ESCOLA POLITÉCNICA DA UNIVERSIDADE DE SÃO PAULO

Departamento de Engenharia de Computação e Sistemas Digitais

PCS3635 – LABORATÓRIO DE PROJETO DE SISTEMAS DIGITAIS I

EXPERIÊNCIA 3 – Projeto de uma Unidade de Controle

Relatório da Bancada B6 – Turma 2 – Prof. Antônio

Data de Emissão: 26 de Janeiro de 2026.

Nome: Gabriel Agra de Castro Motta	**Número USP: 15452743**
Nome: Mateus Silva de Araújo	**Número USP: 15497076**
Nome: Vinicius Akira Durante Tahara	**Número USP: 12547002**

1 Introdução **3**
2 Descrição do Projeto **3**
3 Detalhamento do Projeto Lógico **4**
3.1 Projeto do Fluxo de Dados **4**
3.2 Projeto da Unidade de Controle **5**
3.3 Projeto do Sistema Digital **7**
4 Plano de Testes do Sistema e Simulações **8**
4.1 Cenário de Teste – Validação Do Fluxo De Dados **8**
4.2 Cenário de Teste – Validação Do Sistema Completo 9
5 Implantação do Projeto **14**
5.1 Pinagem da Placa FPGA **14**
5.2 Estratégia de Montagem **15**
5.3 Estratégia de Depuração **16**
5.4 Resultados Práticos em Bancada **16**
6 Perguntas da Apostila **17**
7 Projeto do Desafio da Experiência **21**
7.1 Descrição do Desafio **21**
7.2 Descrição do Projeto Lógico **21**
7.2.1 Alterações do Fluxo de Dados **21**
7.2.2 Alterações da Unidade de Controle **21**
7.2.3 Alterações do Sistema Digital **21**
7.3 Verificação e Validação do Desafio **21**
8 Conclusões **23**

1 Introdução

Esta experiência tem como objetivo o projeto, simulação e síntese de um sistema digital hierárquico composto por uma Unidade de Controle (UC) e um Fluxo de Dados (FD), utilizando a linguagem de descrição de hardware Verilog. O foco prático é a implementação de uma Máquina de Estados Finitos (FSM) que orquestra a leitura de chaves, armazenamento em registrador e comparação com dados pré-gravados em uma memória ROM, validando o funcionamento através de simulação no ModelSim e execução física na placa FPGA DE0-CV.

2 Descrição do Projeto

O sistema digital projetado realiza a comparação sequencial entre um valor de entrada (chaves) e uma série de 16 valores armazenados em uma memória interna. O funcionamento baseia-se em um ciclo onde o circuito espera um sinal de início, armazena o valor das chaves, compara com o dado atual da memória, exibe o resultado e avança para o próximo endereço de memória .

A lógica de controle garante que a varredura da memória ocorra passo a passo. O sistema deve apresentar nos displays de 7 segmentos os valores atuais de contagem (endereço), dado da memória e estado da UC para fins de depuração. Ao final da varredura dos 16 endereços, o sistema sinaliza o término através do sinal pronto e retorna ao estado de espera.

```
Algoritmo: um sistema digital simples
    entradas: iniciar, chaves
    saídas: pronto
    depuração: chaves, contagem, memória, estado, igual, iniciar
1.  {
2.     while (verdadeiro) {
3.        espera acionamento do sinal iniciar
4.        inicia circuito para condições iniciais
5.        while (não atingiu o último dado) {
6.            compara chaves de entrada com dados armazenados e atualiza saídas
7.            incrementa contador interno
8.        }
9.        ativa saída pronto
10.    }
11. }
```

Figura 1: Pseudocódigo do Sistema Digital da Experiência

3 Detalhamento do Projeto Lógico

Este capítulo do documento tem a finalidade de descrever detalhes do projeto lógico desenvolvido na presente experiência.

3.1 Projeto do Fluxo de Dados

O Fluxo de Dados (FD) foi adaptado da experiência anterior para incluir capacidade de armazenamento e comparação. A estrutura central consiste em um contador síncrono (contador_163) que gera o endereço para uma memória ROM (sync_rom_16x4), a qual fornece o dado "gabarito". Paralelamente, um registrador de 4 bits (registrador_4) captura a entrada das chaves quando habilitado pela UC .

Um comparador de magnitude (comparador_85) verifica a igualdade entre a saída do registrador (chaves salvas) e a saída da ROM. O FD recebe sinais de controle (zeraR, registraR, contaC, zeraC) e devolve sinais de condição (fimC, chavesIgualMemoria) para a Unidade de Controle, além de exportar os barramentos para os displays de depuração.

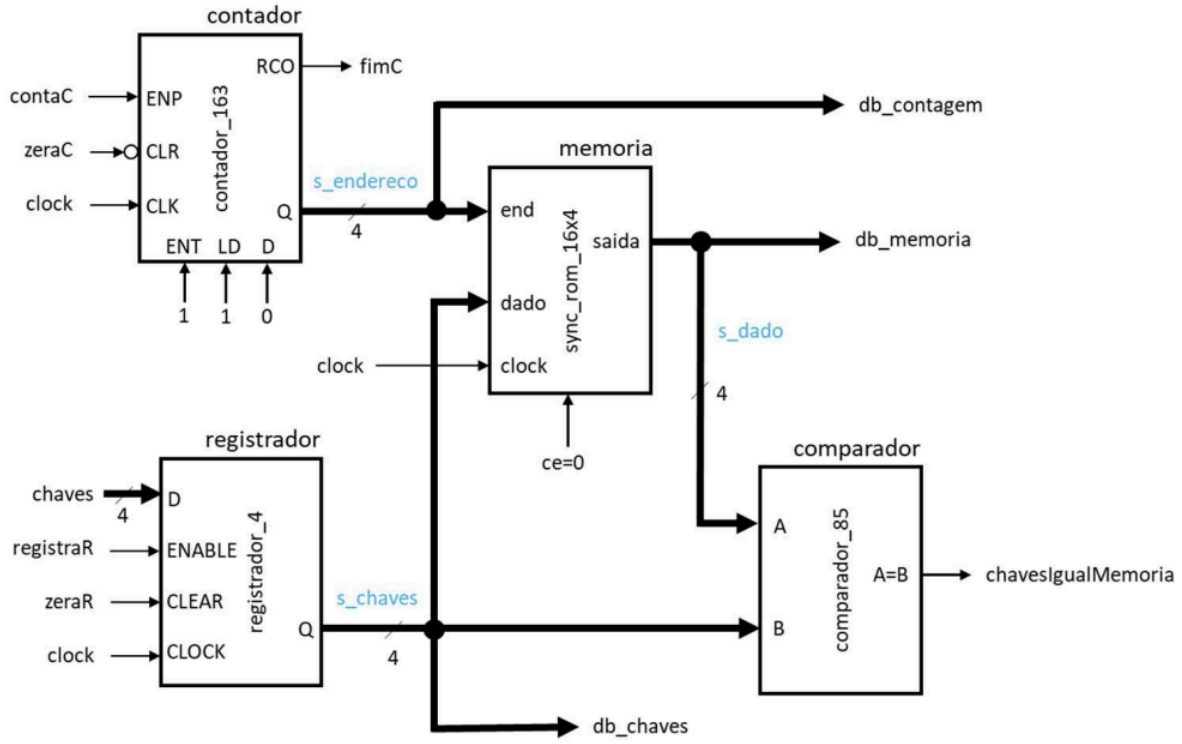

Figura 2: Diagrama de Blocos da Estrutura Interna do Fluxo de Dados

3.2 Projeto da Unidade de Controle

A Unidade de Controle é uma Máquina de Estados Finitos (FSM) que gerencia a sequencialização das operações. Ela foi projetada para traduzir comandos de alto nível (como "Zerar Contador" ou "Registrar Chaves") em sinais elétricos digitais para o FD. O diagrama de estados garante que o tempo de setup do registrador e da memória sejam respeitados antes que a comparação seja avaliada.

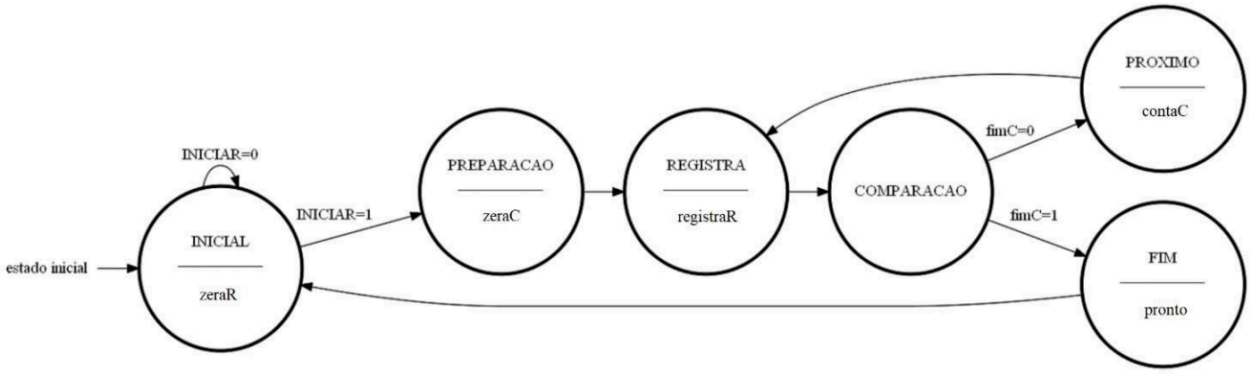

Figura 3: Diagrama de Transição de Estados da Unidade de Controle do Circuito para a Atividade 1

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	INICIAL Estado de reset e espera. Mantém zeraR=1 (limpa registrador).	INICIAL ou PREPARAÇÃO	Permanece se iniciar=0. Se iniciar=1, transita para PREPARAÇÃO para começar o ciclo.
PREPARAÇÃO	Estado de configuração inicial. Ativa zeraC=1 para garantir endereço 0 na memória.	REGISTRA	Transição incondicional no próximo clock. O contador precisa ser zerado apenas uma vez no início.
REGISTRA	Estado de captura. Ativa registraR=1 para salvar o valor das chaves no registrador interno.	COMPARAÇÃO	Transição incondicional. Garante estabilidade do dado das chaves antes da comparação.
COMPARAÇÃO	Estado de espera/análise. Nenhuma saída ativa (padrão de retenção). O comparador do FD atua aqui.	PRÓXIMO ou FIM	Se fimC=0 (não acabou memória), vai para PROXIMO. Se fimC=1 (último dado), vai para FIM.

Nome do Estado	Descrição do Estado	Próximo Estado	Condições e Justificativas para a Transição entre Estados
PRÓXIMO	Estado de incremento. Ativa contaC=1 para avançar endereço da memória.	REGISTRA	Retorna para REGISTRA para capturar as chaves novamente para o novo dado da memória.
FIM	Estado de conclusão. Ativa pronto=1.	INICIAL	Transição incondicional para reiniciar o sistema e aguardar novo comando.

3.3 Projeto do Sistema Digital

A integração é realizada no módulo topo de hierarquia, conectando as saídas de controle da UC às entradas do FD e as saídas de condição do FD (fimC) de volta à UC. Sinais de depuração essenciais (db_estado, db_contagem, db_memoria, db_chaves) foram roteados para saídas externas para permitir visualização nos displays de 7 segmentos e LEDs, facilitando a identificação de falhas de sincronia ou lógica.

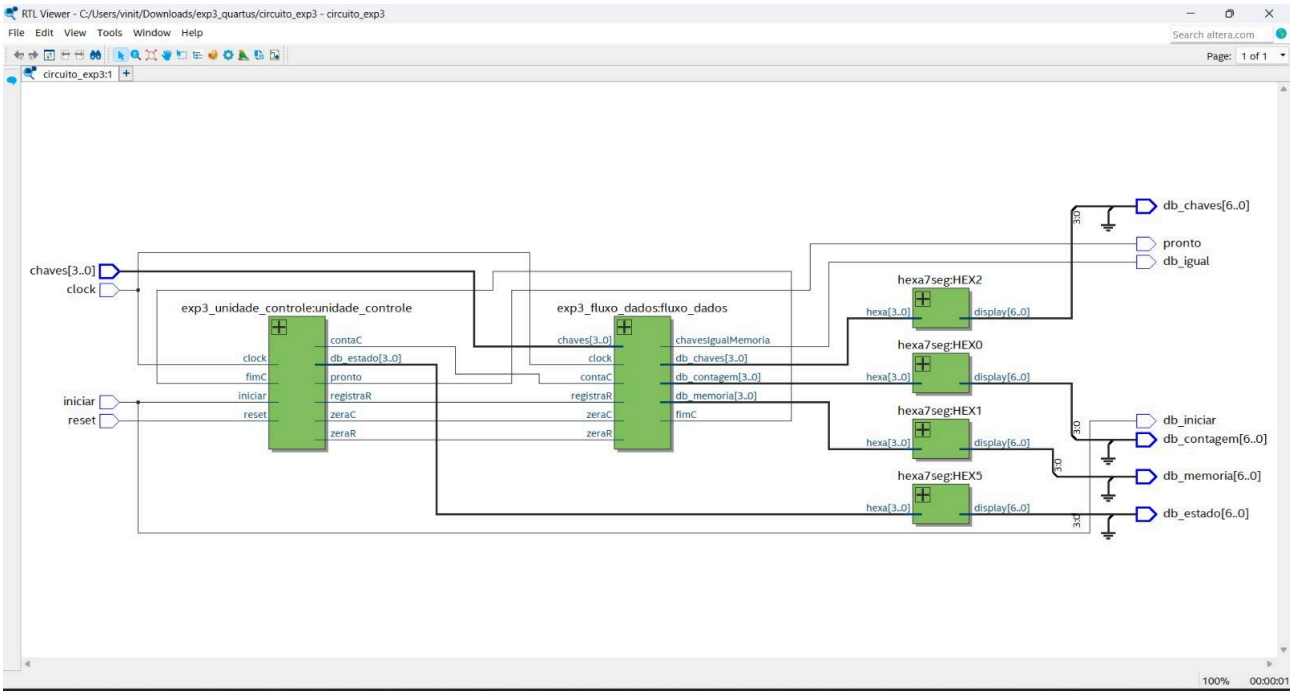

Figura 4: Projeto do RTL Viewer Antes da Adição dos Sinais de db

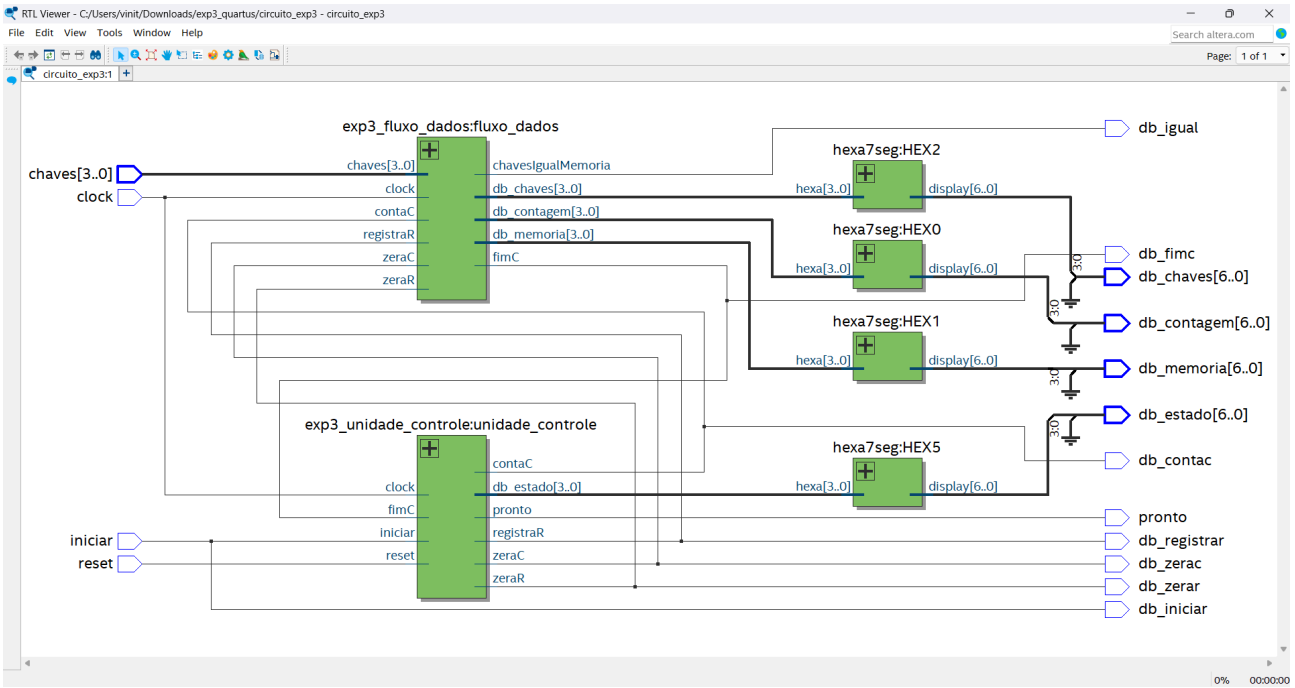

Figura 5: Projeto do RTL Viewer com os Sinais de db

4 Plano de Testes do Sistema e Simulações

A validação foi dividida em teste unitário do Fluxo de Dados e teste integrado do Sistema Completo. As simulações visam garantir que os tempos de propagação e a lógica sequencial estejam corretos antes da síntese física.

4.1 Cenário de Teste – Validação Do Fluxo De Dados

Este cenário visa verificar se o FD responde corretamente aos comandos de controle isolados, garantindo que o registrador armazene o dado no clock correto e que a memória ROM forneça os dados esperados conforme o endereço.

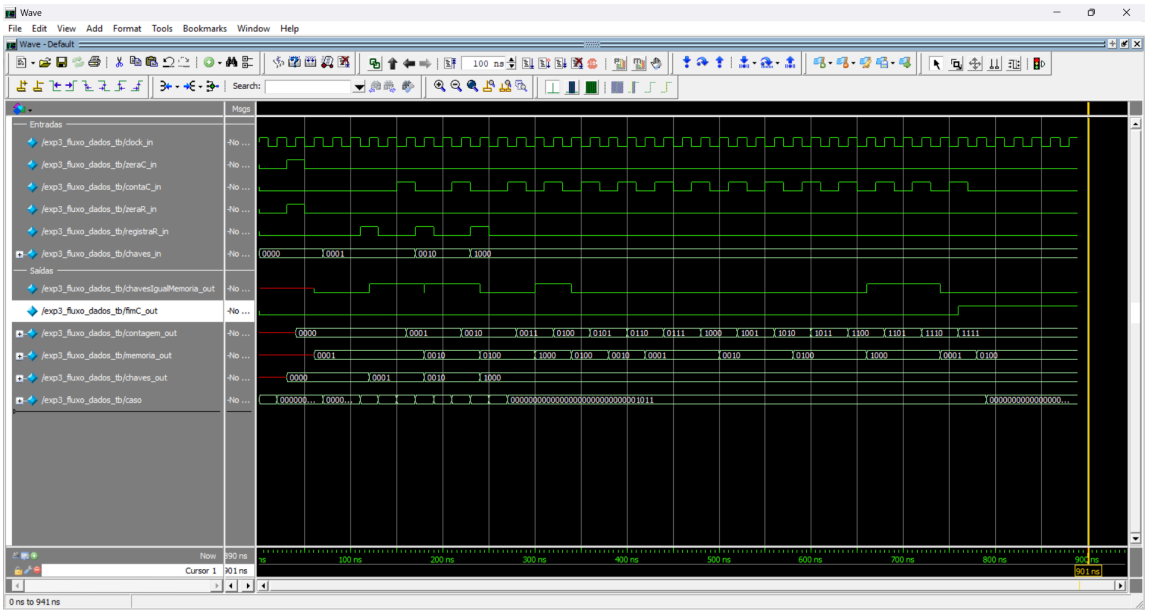

Figura 6: Formas de Onda do Modelsim com Testbench do Fluxo De Dados

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

Entradas	Saídas Esperadas	Resultado Simulado OK?	Análise de Não Conformidades
zeraC=1, zeraR=1	db_contagem=0, db_chaves=0	Sim	O reset síncrono funcionou, zerando os componentes.
chaves=0001, registraR=1	db_chaves passa para 0001 no próx. clock	Sim	O registrador capturou o dado na borda de subida.

Entradas	Saídas Esperadas	Resultado Simulado OK?	Análise de Não Conformidades
contaC=1	db_contagem incrementa, db_memoria muda	Sim	O endereço avançou e o dado da ROM foi atualizado.
contaC até final	fimC=1 quando contagem=1111	Sim	O sinal de *Ripple Carry Out* (RCO) indicou fim da memória corretamente.

4.2 Cenário de Teste – Validação Do Sistema Completo

O objetivo é validar a interação UC-FD. O teste simula um ciclo completo de operação onde o usuário mantém as chaves constantes ou as altera. Verifica-se se a máquina transita pelos estados REGISTRA -> COMPARAÇÃO -> PRÓXIMO 16 vezes e termina em FIM.

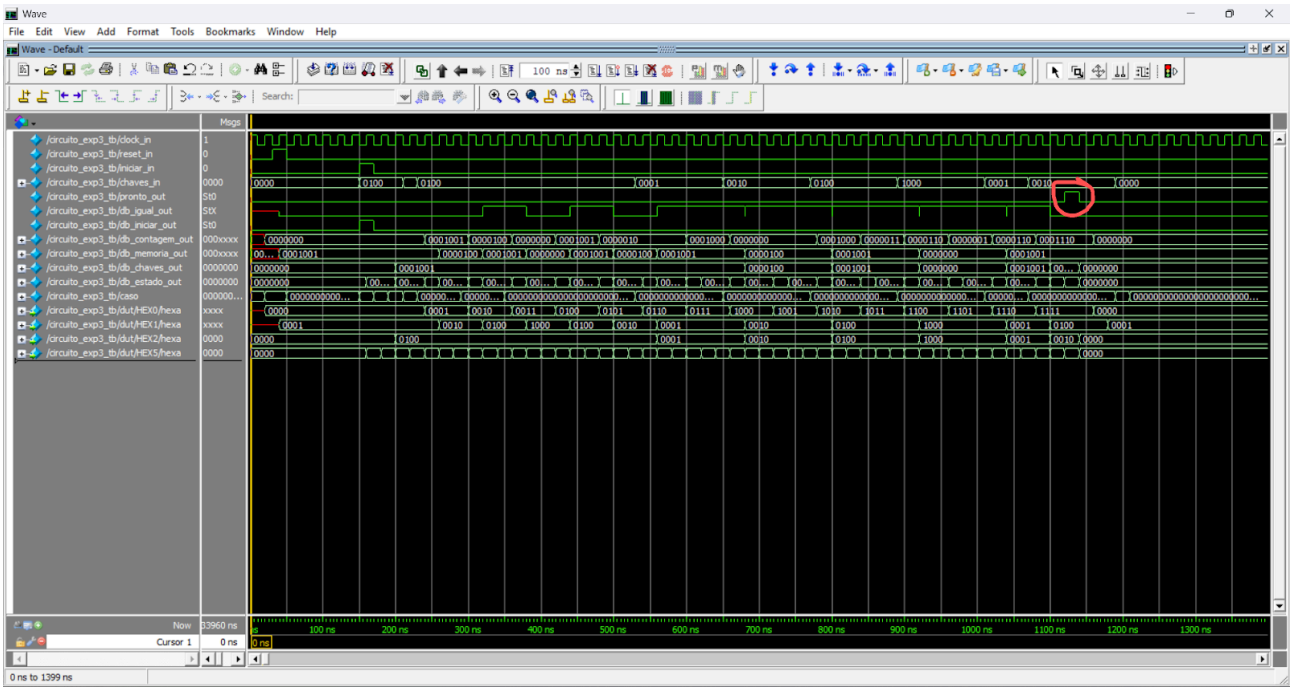

Figura 7: Formas de Onda do Modelsim com Testbench do Sistema

(HEX0 = contagem; HEX1 = memória; HEX2 = chaves; HEX5 = estados)

Sinais de controle	**Resultado esperado**	**Resultado simulado OK?**	**Análise de não conformidades**
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	NOK	db_contagem só muda de xxxx para 0000 na segunda subida de clock. db_memoria só passa de xxxx para 0001 na primeira borda de subida após a ativação do reset.
reset=1, clock ↑	pronto=0, db_igual=0, db_contagem=0000, db_memoria=0001, db_chaves=0000, db_estado=0000	OK	
reset=0, iniciar=0, clock ↑ (5x)	(permanece no estado **inicial**), pronto=0, db_igual=0, db_contagem=0000, db_memoria=0001, db_chaves=0000, db_estado=0000	OK	Com o tb dado, a ativação do clock ocorria só 1x ao invés de 5x; por isso o código foi alterado para cumprir os testes propostos
chaves=0100, iniciar=1, clock ↑	(muda para estado **preparação**), pronto=0, db_igual=0, db_contagem=0000, db_memoria=0001, db_chaves=0000, db_estado=0001	OK	

chaves=0100, iniciar=0, clock ↑	(muda para estado **registra**), pronto=0, db_igual=0, db_contagem=0000, db_memoria=0001, db_chaves=0000, db_estado=0100	OK	
chaves=0100, iniciar=0, clock ↑	(muda para estado **comparação**), pronto=0, **db_igual=0**, db_contagem=0000, db_memoria=0001, **db_chaves=0100**, db_estado=0101	OK	
chaves=0100, clock ↑	(muda para estado **próximo**), pronto=0, db_igual=0, db_contagem=0000, db_memoria=0001, db_chaves=0100, db_estado=0110	OK	
chaves=0100, clock ↑ (3x)	(passa pelos estados registra, comparação e próximo), pronto=0, db_igual=0, db_contagem=0001, db_memoria=0010, db_chaves=0100	OK	
chaves=0100, clock ↑ (3x)	(passa pelos estados registra, comparação e próximo), pronto=0, **db_igual=1**, db_contagem=0010,	OK	

	db_memoria=0100, db_chaves=0100		
chaves=0100, clock ↑ (9x)	(passa 3x pelos estados registra, comparação e próximo), pronto=0, db_igual varia 0-1-0, db_contagem varia 0011-0100-0101, db_memoria varia 1000-0100-0010, db_chaves=0100	OK	
chaves=0001, clock ↑ (6x)	pronto=0, db_igual=1, db_contagem=0111, db_memoria=0001, db_chaves=0001, db_estado=0110	OK	
chaves=0010, clock ↑ (6x)	pronto=0, db_igual=1, db_contagem=1001, db_memoria=0010, db_chaves=0010, db_estado=0110	OK	
chaves=0100, clock ↑ (6x)	pronto=0, db_igual=1, db_contagem=1011, db_memoria=0100, db_chaves=0100, db_estado=0110	OK	
chaves=1000, clock ↑ (6x)	pronto=0, db_igual=1, db_contagem=1101, db_memoria=1000,	OK	

	db_chaves=1000, db_estado=0110		
chaves=0001, clock ↑ (3x)	pronto=0, db_igual=1, db_contagem=1110, db_memoria=0001, db_chaves=0001, db_estado=0110	OK	
chaves=0010, clock ↑ (3x)	(passa pelo estado **fim**), **pronto=1**, db_igual=0, db_contagem=1111, db_memoria=0100, db_chaves=0010, db_estado=1111	OK	
chaves=0000, clock ↑	(termina no estado **inicial**), pronto=0, db_igual=0, db_contagem=0000, db_memoria=0001, db_chaves=0000, db_estado=0000	OK	No tb, após o sinal de pronto retornar para 0, ele demora mais de um clock para retornar as chaves para 0000 (por ser input, não é problema na lógica do código), para isso, foi analisado a primeira borda de subida após a chave retornar para 0.

5 Implantação do Projeto

Este capítulo tem o objetivo de documentar as atividades práticas de execução do projeto desenvolvido e documentado no capítulo 3 no ambiente do Laboratório Digital.

5.1 Pinagem da Placa FPGA

A pinagem foi planejada para facilitar a interação e depuração visual. O Clock e Reset foram mapeados para controle externo via Analog Discovery ou botões da placa.

A pinagem foi feita com base no arquivo disponibilizado no moodle.

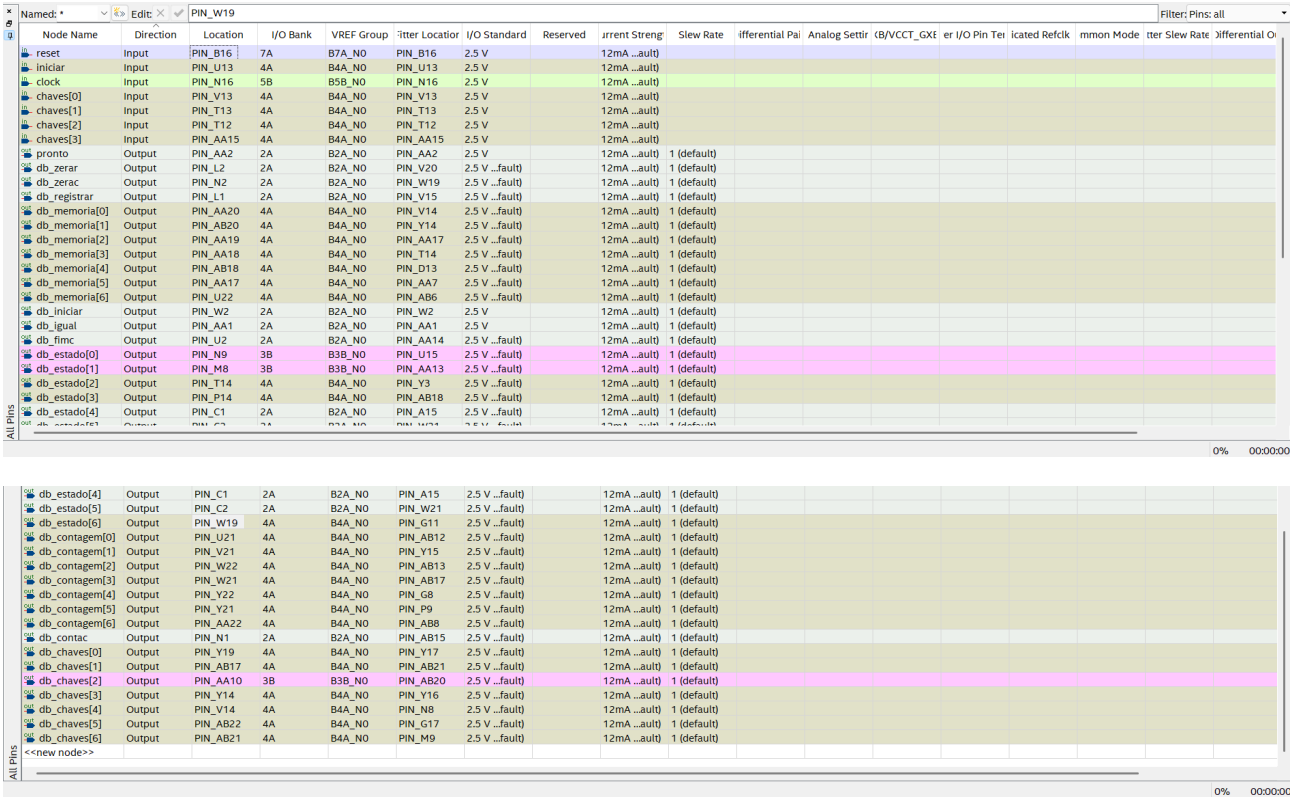

Figura 8: Pinagem do Pin Planner da Experiência

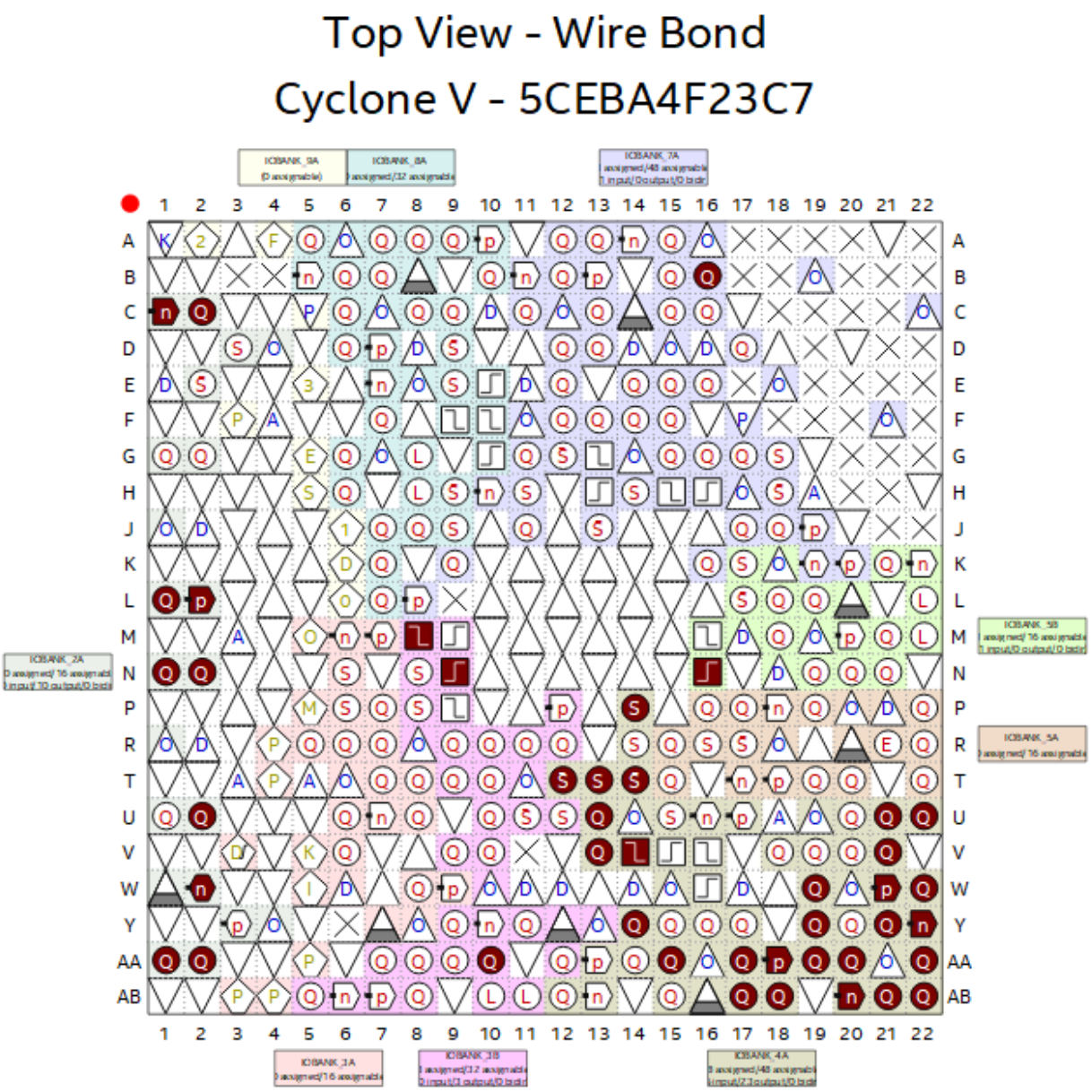

Figura 9: Pinagem do Pin Planner da Experiência

Clock: PIN_M9 (GPIO_0[0]) - Para controle preciso via Analog Discovery.

Reset: PIN_N9 (GPIO_0[1]).

Iniciar: PIN_U13 (SW0).

Chaves [3:0]: PIN_V13, PIN_T13, PIN_T12, PIN_AA15 (SW1 a SW4).

Displays: HEX0 (Contagem), HEX1 (Memória), HEX2 (Chaves), HEX5 (Estado).

5.2 Estratégia de Montagem

a) Carregar o arquivo .sof gerado na placa DE0-CV via USB-Blaster.

b) Conexões Físicas:

i) Conectar o GND do Analog Discovery ao GND da GPIO_0 (Pino 12 ou 30).

ii) Conectar o Canal Digital 0 (DIO 0) ao pino GPIO_0[0] (Clock).

iii) Conectar o Canal Digital 1 (DIO 1) ao pino GPIO_0[1] (Reset).

c) Monitoramento: As chaves SW[4:1] controlam a entrada de dados. O SW0 inicia. O feedback visual será imediato nos displays HEX .

Diagrama das conexões físicas da bancada será incluído em relatório

Figura 10: Esquema de Conexões entre a Placa DE0-CV e o Analog Discovery

5.3 Estratégia de Depuração

A estratégia baseia-se na observabilidade total do estado interno.

a) Os sinais adicionados foram db_zeraC, db_contaC, db_fimC, db_zeraR e db_registraR, assim como recomendado na apostila. Eles servem para verificar como sinais internos (que não poderiam ser vistos normalmente) estão se comportando, o que ajuda a entender possíveis erros ou verificar o comportamento esperado. O sinal db_estado no HEX5 é o mais crítico. Se o sistema travar, o código hexadecimal indicará em qual estado a FSM parou (ex: se travar em '0', não saiu do reset; se travar em '3', está preso na comparação).
b) Utilizar o Analog Discovery para gerar pulsos de clock passo-a-passo (single step) permite verificar a transição de estados em câmera lenta, validando se o dado é registrado antes de ser comparado.
c) Em caso de erro, a primeira verificação será no sinal fimC (se a contagem não termina) e na sensibilidade do Reset.

5.4 Resultados Práticos em Bancada

6 Perguntas da Apostila

a) Segue abaixo o pseudocódigo para o sistema digital simples:

```
Início:
        Esperar acionamento de 'iniciar'
        Zerar contador (Endereço = 0)
        Enquanto (Contador < 16) faça:
                Zerar registrador de chaves
                Registrar entrada das Chaves
                Comparar Chaves com Dado da Memória (Atualiza saídas)
        Incrementar Contador
        Fim Enquanto
        Ativar sinal 'Pronto'
        Voltar para Início
```

b) Segue abaixo o código do fluxo de dados adaptado com registrador e ROM:

```
module exp3_fluxo_dados (
input clock,
input [3:0] chaves,
input zeraR,
input registraR,
input contaC,
input zeraC,
output chavesIgualMemoria,
output fimC,
output [3:0] db_contagem,
output [3:0] db_chaves,
output [3:0] db_memoria
);
   wire   [3:0] s_endereco;
   wire   [3:0] s_dado, s_chaves;

   // contador_163
   contador_163 contador (
     .clock( clock ),
     .clr  ( ~zeraC ),
     .ld   ( 1'b1 ),
```

```
    .ent  ( 1'b1 ),
    .enp  ( contaC ),
    .D    ( 4'b0000 ),
    .Q    ( s_endereco ),
    .rco  ( fimC )
  );

  // comparador_85
  comparador_85 comparador (
    .A   ( s_dado ),
    .B   ( s_chaves ),
    .ALBi( 1'b0 ),
    .AGBi( 1'b0 ),
    .AEBi( 1'b1 ),
        .ALBo(), // não podemos atribuir um valor aqui pois são outputs gerenciados pelo comparador
    .AGBo(), // caso contrário haverá curto-circuito (High Z)
    .AEBo( chavesIgualMemoria )
  );

  registrador_4 registrador(
      .clock(clock),
      .clear(zeraR),
      .enable(registraR),
      .D(chaves),
      .Q(s_chaves)
  );

  sync_rom_16x4 memoria(
      .clock(clock),
      .address(s_endereco),
      .data_out(s_dado)
  );

  // saida de depuracao
```

```
    assign db_contagem = s_endereco;
    assign db_chaves = s_chaves;
    assign db_memoria = s_dado;

endmodule
```

c) O Plano de Testes da Tabela 1 verifica exaustivamente o Fluxo de Dados ao testar isoladamente: 1) O Reset (garantindo estado inicial conhecido); 2) A carga do Registrador (validando a retenção de dados); 3) O incremento do Contador (validando o endereçamento); e 4) O sinal de fim de curso (RCO). Isso cobre todas as funcionalidades críticas do bloco operacional.

d) A análise da máquina de estados revela que as transições dependem estritamente de sinais de controle síncronos. A tradução para Verilog deve usar um bloco always @(posedge clock) para a memória de estado e um bloco always @(*) para a lógica de próximo estado e saídas, garantindo que não haja latches indesejados. A inconsistência potencial seria o atraso na propagação do sinal fimC vindo do contador, o que exige que a condição de saída do estado COMPARAÇÃO seja avaliada com cuidado.

e) Na tradução da unidade de controle, o zeraC não considera que o módulo contador ativa o CLR em nível baixo, o que deve ser corrigido posteriormente (alterando o código da unidade de controle ou a entrada do módulo da UC no código do circuito final). Se compararmos a máquina de estados com sua tradução, há diferenças na ativação de zeraC e zeraR. Enquanto na imagem zeraC e zeraR estão ativados somente em “Inicial” e “Preparação” respectivamente, no código ambos ficam ativados nos dois estados. Esta diferença, entretanto, não altera o funcionamento deste circuito. Por fim, a declaração dos estados não segue uma sequência (p. ex. 1, 2, 3 …), mas os estados são identificados por 0, 1, 4, 5, 6 e F.

f) O componente principal foi elaborado conforme descrito na Seção 3.3. O componente foi estruturado instanciando a UC e o FD. As conexões internas (wires) foram nomeadas explicitamente (ex: wire w_zeraR, wire w_fimC) para facilitar o debug no RTL Viewer e na simulação, seguindo a arquitetura proposta.

g) A simulação do sistema completo está detalhada e analisada na Seção 4.2. Ela confirmou o comportamento cíclico. Observou-se que o sinal pronto é ativado exatamente um ciclo de clock após o fimC do contador ir a nível alto, o que está conforme a especificação de que o sinal deve ser pulsado ao final da operação.

h) A ferramenta RTL Viewer gerou um esquemático onde se visualiza claramente a separação entre Controle e Dados. Não foram observados "latches" inferidos acidentalmente, e as conexões de feedback do fimC aparecem corretamente roteadas.

i) A tabela de pinagem completa está na Seção 5.1.

j) Arquivo QAR da Atividade 1 anexado à entrega.

k) Os testes práticos estão descritos na Seção 5.4

l) Arquivo QAR da Atividade 2 anexado à entrega.

Desafio: (será incluído em relatório final)

m) O sistema deve ser modificado para interromper o loop de comparação assim que uma desigualdade for detectada entre chaves e memória. Isso exige que a Unidade de Controle monitore o sinal de condição chavesIgualMemoria e desvie o fluxo para o estado final de erro caso o valor seja '0'.

n) Foram planejados dois cenários: um de sucesso total (16 acertos) e um de falha crítica (erro no 4º dado). As tabelas detalhadas com os sinais de controle e estados esperados constam na Seção 7.3 deste documento.

o) Os arquivos de testbench serão adaptados para simular a inserção de dados incorretos pelo usuário no tempo de clock correspondente ao quarto endereço de memória. As formas de onda comprobatórias serão geradas em laboratório e incluídas no relatório final.

p) A pinagem mínima para o desafio foi definida conforme a Tabela 5 da apostila, incluindo os LEDs indicadores ACERTOU (LEDR8) e ERROU (LEDR9).

q) Será realizada em relatório. O procedimento experimental seguirá a sequência de estímulos planejada, utilizando o Analog Discovery para fornecer o clock e monitorar a interrupção precoce da FSM em caso de erro.

r) Será realizada em relatório. O arquivo QAR consolidado contendo os módulos circuito_exp3_desafio.v, a nova UC e os testbenches será enviado após essa validação.

7 Projeto do Desafio da Experiência

7.1 Descrição do Desafio

O desafio consiste na implementação de uma lógica de interrupção antecipada ("morte súbita"). O sistema deve validar cada dado inserido e, em caso de erro, ativar a saída errou e encerrar a operação imediatamente, sem percorrer o restante da memória.

7.2 Descrição do Projeto Lógico

7.2.1 Alterações do Fluxo de Dados

O Fluxo de Dados passará a exportar o sinal chavesIgualMemoria (proveniente do comparador_85) como um sinal de condição para a Unidade de Controle. O restante da estrutura (contador, ROM e registrador) permanece inalterado.

* A execução do desafio será incluída no relatório final.

Figura 11: Rtl Viewer do Fluxo de Dados com a Saída chavesIgualMemoria

7.2.2 Alterações da Unidade de Controle

A FSM será modificada para que, no estado COMPARACAO, a transição para o estado PROXIMO só ocorra se chavesIgualMemoria == 1. Caso contrário, a máquina transita para o estado FIM, ativando o bit de erro.

* A execução do desafio será incluída no relatório final.

Figura 12: Diagrama de Estados da Fsm Modificado para Interrupção por Erro

7.2.3 Alterações do Sistema Digital

A integração incluirá a lógica de saída para os novos sinais:

- ACERTOU: Ativado se fimC == 1 e o último dado for igual.
- ERROU: Ativado se chavesIgualMemoria == 0 em qualquer etapa do loop.

Ambos os sinais serão pulsados junto com o sinal pronto por um período de clock.

7.3 Verificação e Validação do Desafio

Foram criados dois cenários:

1. Sucesso: Acerto dos 16 valores -> Saída acertou=1.
2. Falha: Erro no 4º valor -> O sistema para imediatamente, contador não avança mais, saída errou=1.

Cenário de Teste: Falha no 4º Dado

Iniciar sistema, acertar os 3 primeiros valores, inserir valor incorreto no 4º valor.

Resultado Esperado: O sinal db_contagem deve parar em 0011 (decimal 3), o estado deve saltar para FIM, e as saídas devem ser pronto=1 e errou=1.

* Os dados de observação, com as formas de onda do ModelSim para os cenários de sucesso e falha, assim como o RTL Viewer modificado, serão incluídas no relatório final após a conclusão da síntese e validação em hardware.

Tabela 3: Plano de Testes do Desafio - Cenário de Sucesso

Tabela 4: Plano de Testes do Desafio - Cenário de Falha

8 Conclusões

A análise preliminar das simulações indica que o projeto lógico cumpre os objetivos. A separação entre UC e FD permitiu testar os módulos independentemente, reduzindo o risco de erro na integração. O ponto crítico identificado para a prática é a estabilidade dos botões mecânicos (bouncing) na entrada iniciar, o que não aparece na simulação ideal, mas pode causar múltiplos disparos na FPGA. Para o desafio, o planejamento dos cenários de teste para o desafio de "morte súbita" demonstra que a lógica de desvio da FSM é a solução mais eficiente para interromper o fluxo de dados em tempo real. Os objetivos preliminares de projeto e simulação foram atingidos, restando a validação física das temporizações e do comportamento dos sinais na placa FPGA.