

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

## PCS3635 – LABORATÓRIO DIGITAL I

### Relatório Técnico

Relatório da Bancada 07 – Turma 1 – Prof. Edson Midorikawa

Data de Emissão: 15 de Abril de 2025.

<b>Nome:</b> Mourtaza Akil	<b>Número USP:</b>
<b>Nome:</b> Gabriel Chaves Lopes Silva	<b>Número USP:</b>
<b>Nome:</b> Luiz Mariano dos Santos Silva	<b>Número USP:</b>
<b>Nome:</b> Luca Bompiani	<b>Número USP:</b>

### 1 INTRODUÇÃO

Nosso objetivo é incorporar os conceitos desenvolvidos na disciplina de **Laboratório Digital 1** na criação de uma solução que atenda às necessidades de pessoas com **Transtorno do Espectro Autista (TEA)**, proporcionando maior acessibilidade e suporte por meio de um modelo de prova de conceito.

### 2 CONTEXTO DA APLICAÇÃO

Inicialmente, nos foi apresentado a problemática de se desenvolver a partir do jogo da memória, previamente desenvolvido, uma solução que pudesse atender pessoas com **Transtorno do Espectro Autista (TEA)** e que para além de um jogo, fosse uma solução de jogo sério, logo o objetivo deixa de ser o entretenimento. A partir, desse desafio, antes de avaliarmos o que poderíamos ou não fazer para atender ao requisitos, desenvolvemos algumas pesquisas sobre os tópicos:

1. Jogos Sérios
2. TEA
3. Exercícios de Auto Regulação

Todo processo para a realização desta pré etapa, fora dividido em duas partes, **Aprofundamento nos conceitos de jogos sérios e TEA & Pesquisas de campo.** Cada qual detalhada logo abaixo:

### **Aprofundamento nos conceitos de jogos sérios**

Notamos logo de cara o quanto o grupo estava por fora do cenário do problema apresentado, para isso o primeiro foco foi entender melhor com quem estaríamos trabalhando e lidando. Com isso, nessa primeira etapa nosso foco foi utilizar buscadores e indexadores acadêmicos como **Web of Science**, **Engineering Village** e **PBI** para localizar artigos relevantes. Para otimizar a busca, definimos **strings** (conjuntos de termos) que incluímos nos campos de Título (Title), Resumo (Abstract) e Palavras-Chave (Keywords/TAK). Dessa forma, conseguimos refinar a pesquisa e encontrar materiais específicos sobre.

### **Pesquisas de campo**

Tendo uma visão mais ampla do que estaríamos trabalhando, buscamos entender como isso se aplicava ao mundo real. Embora pesquisas acadêmicas sejam excelentes bases para elucidação e direcionamento, elas nem sempre apresentam aplicabilidade direta ou exemplos concretos do cotidiano.

Com isso em mente, realizamos algumas pesquisas. A primeira delas foi uma entrevista com uma professora da rede municipal de ensino de São Paulo, que possui mais de 30 anos de experiência na educação infantil. Embora seu foco não seja especificamente crianças com TEA, sua experiência contribuiu significativamente para o projeto, especialmente na definição da faixa etária ideal para a solução proposta.

A partir dessa entrevista, fomos direcionados a dar uma atenção especial aos exercícios de autorregulação, pois eles se mostraram fundamentais para o desenvolvimento infantil. Além disso, essa abordagem se alinha perfeitamente com o material acadêmico consultado, reforçando a importância desse aspecto no nosso projeto.

### **PRINCIPAIS FONTES DE CONSULTA**

- Professora Simone da Rede Municipal de São Paulo
- Motivacao Autismo - <https://www.motivacaoautismo.com.br/single-post/2020/07/30/exerc%C3%ADcios-divertidos-para-regula%C3%A7%C3%A3o-emocional>
- Família Tagarela - Autismo & TDAH - <https://youtube.com/shorts/vWNPKdC4iEs?feature=shared>
- Cartilha de Regulacao Emocional - Projeto Joga Aurora - <https://www.feevale.br/Comum/midias/26f80400-4d56-4a4a-a130-02cc28e03a99/Cartilha%20de%20Regulacao%20Emocional%20-%20Projeto%20Joga%20Aurora.pdf>
- ZALAPA, R.; TENTORI, M. Movement-Based and Tangible Interactions to Offer Body Awareness to Children with Autism. *Lecture Notes in Computer Science*, p. 127–134, 2013.
- ZAKARI, H. M.; MA, M.; SIMMONS, D. A Review of Serious Games for Children with Autism Spectrum Disorders (ASD). *Serious Games Development and Applications*, p. 93–106, 2014.
- REYES, S. V. et al. Serious Game Prototype Based on Strategies for Children with Autism Spectrum Disorder: Learning with “TEA”. p. 140–149, 6 nov. 2023.
- Como a Tecnologia Ajuda no Desenvolvimento de Crianças com Autismo: Ferramentas e Apps Essenciais. Disponível em: <https://autismovr.com.br/uso-de-tecnologia-no-autismo/>. Acesso em: 17 fev. 2025.

### 3 SOLUÇÃO PROPOSTA

A solução proposta envolve o desenvolvimento de um tapete interativo (similar com o jogo Twister), baseado em brincadeiras de autorregulação, como o jogo "Pulo do Sapo". Esse tapete será projetado para ajudar crianças com TEA a melhorar o foco/atenção e desenvolver habilidades motoras por meio de desafios. O conceito principal seria:

- **Estimulação Multissensorial:** Integração de elementos que estimulem não apenas a visão e a audição, mas também o tato (no caso o tapete).
- **Objetivos Terapêuticos/Educacionais:** Além da diversão, o jogo teria foco em objetivos terapêuticos, como trabalhar coordenação motora, percepção sensorial e atenção sustentada.
- **Personalização:** Permitir ajustes de jogadas, tipo de estímulo, cores e sons, de acordo com a necessidade do usuário e o grau de sensibilidade sensorial.
- **Métricas de Acompanhamento:** Coletar dados de desempenho para que terapeutas e educadores possam monitorar a evolução do usuário e adaptar o jogo conforme necessário.

Essa abordagem se inspira em conceitos de **jogos sérios**, pois, além de entreter, visa contribuir para o desenvolvimento e a melhoria das habilidades sensoriais e cognitivas.

Inicialmente, o MVP será voltado para o grupo B, que inclui crianças com dificuldades motoras e desafios relacionados à atenção. A ideia é que elas realizem movimentos específicos no tapete seguindo instruções visuais e sonoras, promovendo coordenação e concentração. Além disso, pretendemos também atender indivíduos do nível 1, que fazem parte do grupo de suporte, possuindo um grau mais e leve mas que ainda assim demandam de algum suporte para manter uma boa qualidade de vida. Acreditamos que com o foco nesses dois aspectos, conseguimos entregar algo simbolicamente valoroso.

Acreditamos que esse projeto, especialmente o primeiro MVP precisa além de ter impacto real, servir de aprendizado para os membros do grupo, aos quais carecem de experiência/vivência nesse cenário, então atender esse grupo menor e com essas características servem não só para entregar algo mais conciso e direcionado, mas também algo que traga uma base de aprendizado para futuros aprimoramentos e expansões. Quem sabe depois dessa primeira fase em um momento futuro não possa ser desenvolvido algo mais amplo que atenda a toda a comunidade ou que tenha um poder impactador maior.

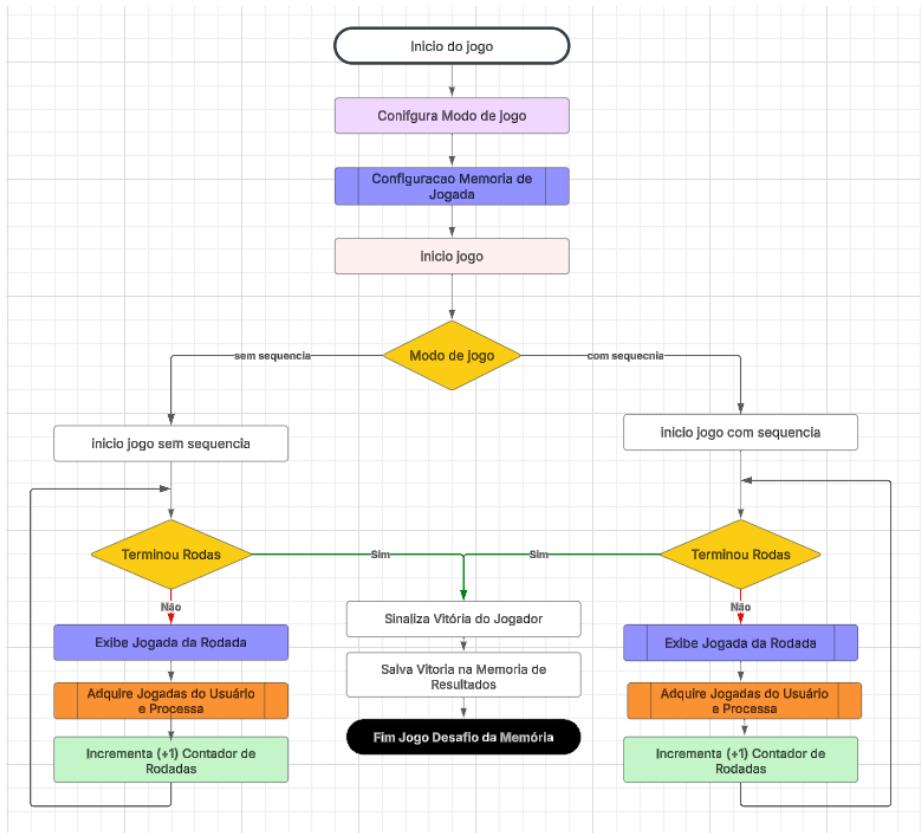
Como complemento a tudo isso, foi pensado em criar uma identidade visual para o projeto dando cara e nome, facilitando assim a familiarização com o mesmo, além de atender uma parte importante de qualquer projeto, ainda mais os que lidam com certos públicos que se trata de criar afeição e ludicidade, sendo o segundo um dos pontos centrais ao que está sendo proposto. Para essa familiarização, damos o nome comercial do projeto de "**PULO DO SAPO**", e a partir de tal seguiremos uma linha de desenvolvimento, que futuramente será exemplificada e desenvolvida melhor.

#### 2.1. Diagramas Comportamental e Estrutural

Listam-se a seguir os principais diagramas do projeto, trazendo uma visão clara do que o projeto pretende realizar e onde almeja chegar.

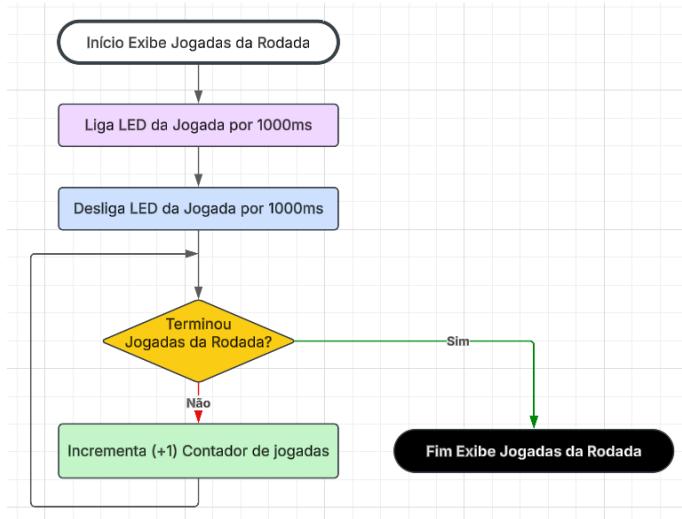
##### 2.1.1 Diagrama Comportamental

Abaixo diagrama geral com todos os estados do jogo proposto.



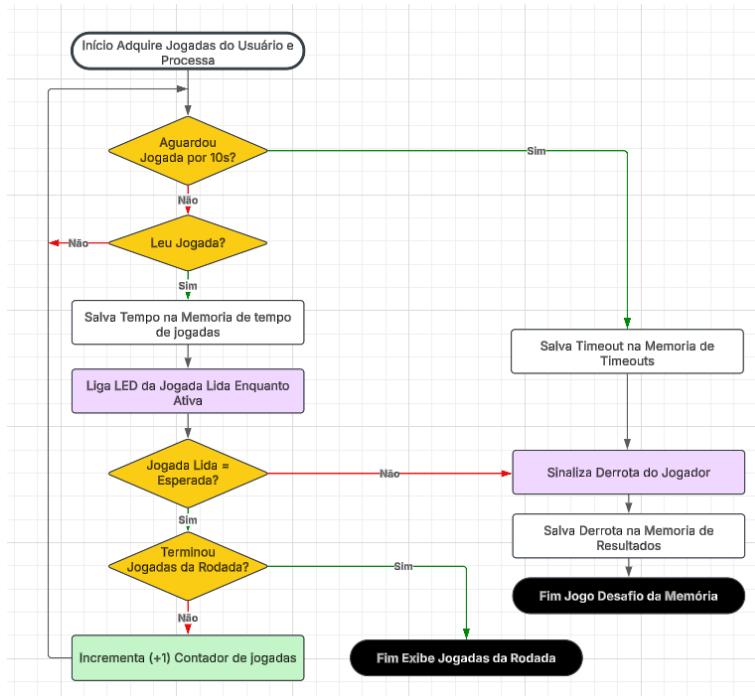
**figura 1. Diagrama Comportamental “Geral”**

Modo que apresenta as jogadas em uma fase inicial, para posteriormente permitir que o jogador tente reproduzir

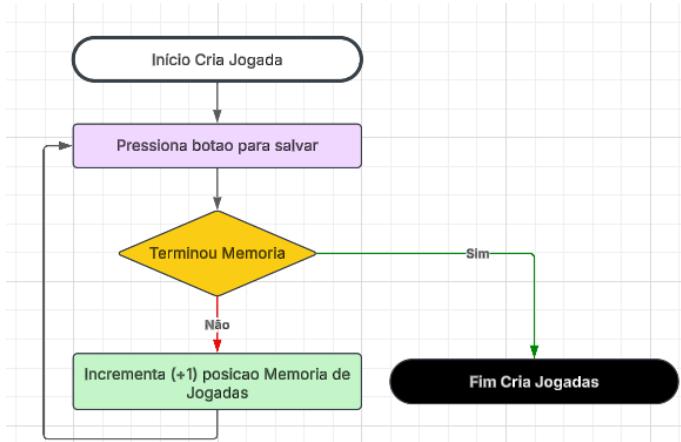


**figura 2. Diagrama Comportamental “Exibe Jogadas da Rodada”**

Descrevesse logo abaixo, a sequência lógica de aquisição de jogadas de um usuário, considerando o timeout e os demais elementos previstos.



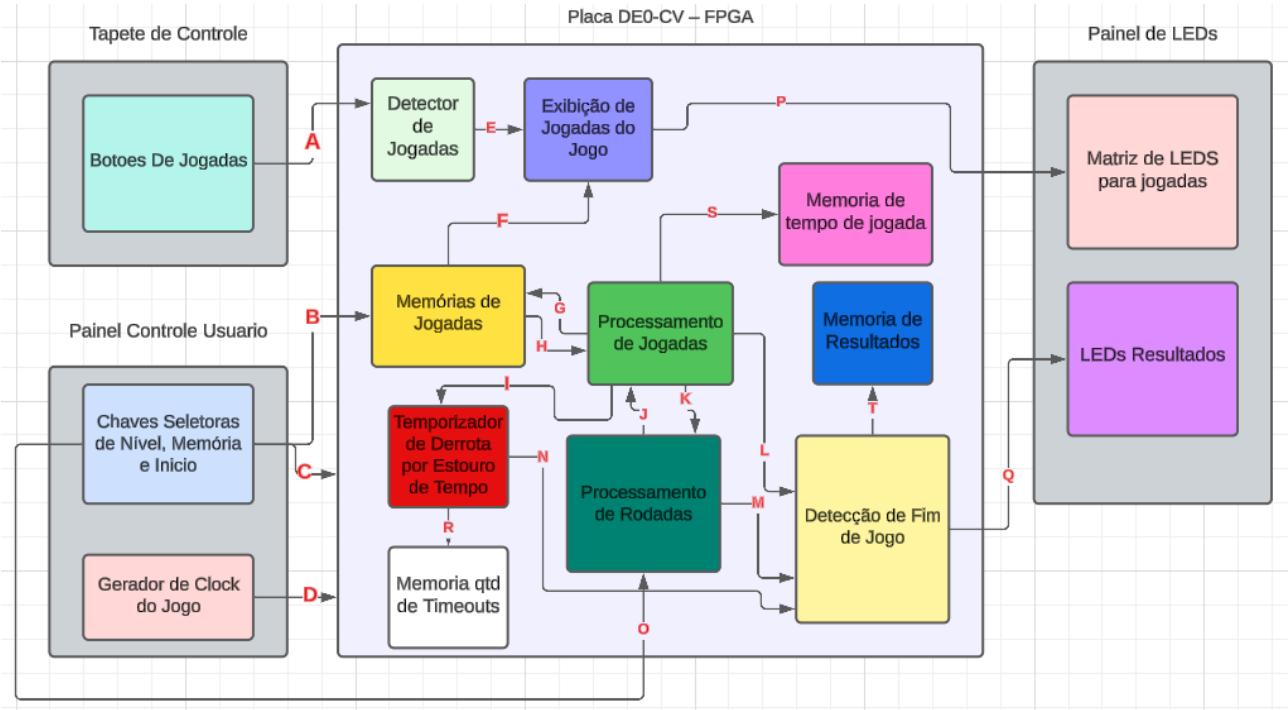
**figura 3. Diagrama Comportamental “Adquire Jogadas do Usuário”**



**figura 4. Diagrama Comportamental “Cria Jogadas”**

### 2.1.2 Diagrama Estrutural

Para o desenvolvimento do diagrama estrutural, imaginamos como a montagem final viria ser, focando na modularização dos elementos, o que facilita tanto o desenvolvimento, teste e eventuais substituições da parte especificada.



**figura 4. Diagrama Estrutural**

A ideia central é em ter um time desenvolvendo a parte da FPGA, enquanto concomitantemente outro desenvolve os demais componentes, sendo esses:

1. **Tapete de controle**, que atua como um controle de botões simples, mas com o diferencial de ser acionado pelos pés, o que traz uma dinamicidade nova ao projeto.
2. **Painel de Controle Usuário + Painel de Leds**, o foco aqui seria unir os dois em uma interface única para o usuário, onde o mesmo controlaria as funções do sistema e receberia feedbacks visuais pertinentes ao que está acontecendo na partida.

Acreditamos que com esses dois três elementos, podem entregar uma solução consideravelmente simples mas que ainda assim e acima de tudo atenda ao cerne do problema proposto, com as justificativas já pré-citadas.

Abaixo segue uma tabela com maiores detalhes do que cada componente representa na incorporação geral da solução.

**tabela 1. Descrição da Arquitetura Estrutural**

Interface	Descrição
A	Jogadas do Usuário
B	Seleção de memória
C	Início

D	<b>clock</b>
E	<b>Jogada filtrada</b>
F	<b>Jogada da memória</b>
G	<b>Endereço da memória</b>
H	<b>Jogada da memória</b>
I	<b>Reinício do temporizador</b>
J	<b>Número da jogada</b>
K	<b>Fim de rodada</b>
L	<b>Erro de jogada</b>
M	<b>Fim de jogo</b>
N	<b>Estouro do temporizador</b>
O	<b>Seleção de nível</b>
P	<b>Indicações de jogadas</b>
Q	<b>Indicações de vitória e derrota</b>
R	<b>Memória para salvar os tempos de jogada</b>
S	<b>Memória para salvar a qtd de Timeout</b>
T	<b>Memória para salvar os resultados</b>

## 2.2. Requisitos

Para guiar todo esse desenvolvimento e entregar uma solução robusta, abaixo listam-se os principais requisitos do projeto, cada qual bem discriminado tanto quanto a funcionalidade, impacto, descrição e dependências.

<b>Código:</b> EXECUCAO	Funcional
<b>Requisito:</b> Execução do Jogo	
<b>Descrição:</b> O jogo deve permitir que o jogador inicie, pause e reinicie uma partida.	
<b>Prioridade:</b> Alta	
<b>Rationale:</b> Este requisito determina a possibilidade de controlar a dinâmica do jogo	

<b>Método de verificação:</b> Simulação e testes .
--

<b>Requisitos associados:</b> não há
--------------------------------------

<b>Código:</b> MEMORIA	Funcional
------------------------	-----------

<b>Requisito:</b> Caminho do jogo jogado
--

<b>Descrição:</b> O sistema proporá dois caminhos diferentes para o jogo.
---

<b>Prioridade:</b> Alta
-------------------------

<b>Estabilidade:</b> Alta
---------------------------

<b>Rationale:</b> Este requisito determina o caminho que será jogado.
---

<b>Método de verificação:</b> Simulação e testes das tentativas de vitória com os diferentes modos.
---

<b>Requisitos asociados:</b> MODO_DE_JOGO
---

<b>Código:</b> MEM_ARMAZENAMENTO	Funcional
----------------------------------	-----------

<b>Requisito:</b> Memórias e Armazenamento
--

<b>Descrição:</b>
-------------------

O jogo deve possuir uma <b>memória ROM</b> para armazenar padrões fixos de sequência.
---

O jogo deve possuir uma <b>memória RAM</b> para criar e armazenar uma partida personalizada temporária.
---

O jogo deve possuir <b>memórias para métricas</b> , armazenando informações como:
---

- Tempo médio de resposta do jogador.
- Número de acertos e erros por rodada.
- Número de timeouts

<b>Prioridade:</b> Alta
-------------------------

<b>Estabilidade:</b> Alta
---------------------------

<b>Rationale:</b> Este requisito determina a dinâmica de cada memória introduzida no sistema
--

<b>Método de verificação:</b> simulação e testes dos diferentes acessos
---

<b>Requisitos associados:</b> GANHA_JOGO, MODO_DE_JOGO
--

<b>Código:</b> LIMITES_RODADAS	Funcional
--------------------------------	-----------

<b>Requisito:</b> Limite de rodadas do jogo.
--

<b>Descrição:</b> O sistema deve possuir um limite de rodadas
---

<b>Prioridade:</b> Alta
-------------------------

<b>Estabilidade:</b> Alta
---------------------------

<b>Rationale:</b> Este requisito determina a quantidade máxima de jogadas que uma partida pode ter
--

<b>Método de verificação:</b> simulação e testes de tentativa de vitória do jogo em todos os níveis
---

de dificuldade.

**Requisitos associados:** GANHA\_JOGO, MODO\_DE\_JOGO

<b>Código:</b> MODO_DE_JOGO	Funcional
<b>Requisito:</b> Modo de jogo.	
<b>Descrição:</b> O sistema deve permitir a seleção de diferentes modos de jogo, incluindo: 1. Um modo onde um conjunto de valores é exibido e o jogador deve repeti-los. 2. Um modo onde um valor é exibido e o jogador o repete; em seguida, outro valor é exibido, e assim por diante.	
<b>Prioridade:</b> Alta	<b>Estabilidade:</b> Alta
<b>Rationale:</b> Este requisito assegura que os jogadores tenham a opção de escolher entre diversos modos de jogo, proporcionando uma experiência variada e adequada às preferências e habilidades individuais.	
<b>Método de verificação:</b> Simulação e testes dos diferentes modos de jogo para garantir o funcionamento correto e a experiência esperada para cada modo.	
<b>Requisitos associados:</b> GANHA_JOGO, NIVEL_JOGO, LIMITE_RODADAS, MODO_DE_JOGO_1, MODO_DE_JOGO_2.	

<b>Código:</b> TAPETE_DE_CONTROLE	Funcional
<b>Requisito:</b> Limite de rodadas do jogo.	
<b>Descrição:</b> O sistema deve possuir um limite de rodadas definido de acordo com o nível de dificuldade configurado.	
<b>Prioridade:</b> Média	<b>Estabilidade:</b> Média
<b>Rationale:</b> Este requisito determina que a quantidade máxima de jogadas depende do nível de dificuldade definido no início de um jogo.	
<b>Método de verificação:</b> simulação e testes de tentativa de vitória do jogo em todos os níveis de dificuldade.	
<b>Requisitos associados:</b> GANHA_JOGO, NIVEL_JOGO,	

<b>Código:</b> TEMPO_DERROTA	Não Funcional
<b>Requisito:</b> Tempo de inatividade do jogador para derrota.	
<b>Descrição:</b> Durante a fase em que o jogo aguarda a próxima jogada, ocorre timeout após 10 segundos de inatividade.	
<b>Prioridade:</b> Alta	<b>Estabilidade:</b> Média
<b>Rationale:</b> Este requisito indica um tempo apropriado para que o usuário possa	

interagir com o jogo entre duas jogadas consecutivas. O tempo de espera pode requerer ajustes em função de realimentações em experimentos práticos com usuários.

**Método de verificação:** simulação e análise com usuários.

**Requisitos associados:** PERDE\_JOGO\_POR\_TEMPO, LOGICA\_RODADA

<b>Código:</b> INTERFACE_VISUAL	Não Funcional
<b>Requisito:</b> Realimentação dos estímulos do jogo ao usuário.	
<b>Descrição:</b> O sistema deve possuir LEDs para exibir ao jogador os estímulos do jogo.	
<b>Prioridade:</b> Alta	<b>Estabilidade:</b> Alta
<b>Rationale:</b> Este requisito serve para determinar a interface visual do jogo.	
<b>Método de verificação:</b> : simulação, teste funcional do jogo e inspeção visual dos LEDs.	
<b>Requisitos associados:</b> Não há.	

<b>Código:</b> ESCALABILIDADE	Não Funcional
<b>Requisito:</b> Escalabilidade	
<b>Descrição:</b> O sistema deve permitir a adição de novos padrões de jogo no futuro sem necessidade de grandes mudanças no hardware.	
<b>Prioridade:</b> Baixa	<b>Estabilidade:</b> Média
<b>Rationale:</b> Este requisito serve para determinar o quanto escalável o sistema pode ser sem prejudicar sua estabilidade	
<b>Método de verificação:</b> : introdução e testes de novos componentes	
<b>Requisitos associados:</b> Não há.	

<b>Código:</b> DURABILIDADE_SEGURANCA	Não Funcional
<b>Requisito:</b> Durabilidade e Segurança	
<b>Descrição:</b> O tapete deve ser resistente e seguro para uso prolongado. Os materiais devem ser atóxicos e confortáveis para pisar. O sistema deve evitar aquecimento excessivo dos componentes eletrônicos.	
<b>Prioridade:</b> Alta	<b>Estabilidade:</b> Alta
<b>Rationale:</b> Este requisito serve para determinar a qualidade do acabamento do produto	
<b>Método de verificação:</b> Testes práticos	
<b>Requisitos associados:</b> Não há.	

<b>Código:</b> ACESSIBILIDADE	Não Funcional
<b>Requisito:</b> Acessibilidade	
<b>Descrição:</b> O sistema deve permitir ajustes sensoriais para jogadores com hipersensibilidade ou hipossensibilidade (controle de volume, intensidade de luz, etc.). O tapete deve ser sensível a diferentes tipos de toque (pulos leves e fortes).	
<b>Prioridade:</b> Alta	<b>Estabilidade:</b> Alta
<b>Rationale:</b> Este requisito serve para determinar a qualidade da experiência do usuário, baseado nas suas necessidades	
<b>Método de verificação:</b> Simulações de casos de usos	
<b>Requisitos associados:</b> Não há.	

<b>Código:</b> TEMPO_RESP	Não Funcional
<b>Requisito:</b> Tempo de Resposta	
<b>Descrição:</b> O tempo entre um comando (pulo) e o feedback visual/sonoro não deve exceder 200ms, garantindo resposta rápida.	
<b>Prioridade:</b> Alta	<b>Estabilidade:</b> Alta
<b>Rationale:</b> Este requisito serve para determinar a qualidade do feedback do sistema	
<b>Método de verificação:</b> simulação, teste funcional do jogo e inspeção visual dos LEDs.	
<b>Requisitos associados:</b> Não há.	

#### 4 LISTA DE MATERIAIS

Segue a relação geral de todos os materiais que serão utilizados na confecção do projeto, separados por componente aos quais se integram. Eventuais atualizações serão indicadas como esperado.

ID	Descrição	Qtd	Especificação	Fornecedor	Custo Unitário	Custo Total
<b>MÓDULO CENTRAL</b>						
1	FPGA	1	Modelo DE0-CV	LABDIG	0	0
2	JUMPER	1	Conjunto de cabos	LABDIG	0	0
3	Fonte alimentacao	1	fonte para FPGA	LABDIG	0	0
TOTAL					0	0
<b>PAINEL DO USUÁRIO</b>						
4	LED Vermelho	4	5mm	Luca	0	0

5	LED Azul	2	5mm	Luca	0	0
6	LED Amarelo	2	5mm	Luca	0	0
7	LED Verde	4	5mm	Luca	0	0
8	Chave	2	2 posições	-	-	-
9	Botao	2	push button	-	-	-
10	Resistor	-	10kOhms	Luca	0	0
11	Botao Arcade	5	push button arcade style	Gabriel	4R\$	4R\$
					TOTAL	4R\$
			<b>TAPETE CONTROLE</b>			
11	Cabos	-	-	-	-	-
12	Resistor	-	-	Luca	-	-
					TOTAL	

A maioria dos componentes listados, especialmente os mais baratos, foram especificados em uma quantidade já pensando em eventuais problemas ou substituições.

## 5 CRONOGRAMA

SEMANA	ATIVIDADES	DETALHES	DATA DE INÍCIO	DATA DE TÉRMINO
1	1. Desenvolvimento e testes do primeiro grupo de requisitos e estrutura geral da solução 2. Busca pelos componentes eletrônicos e não eletrônicos do projeto. 3. Revisões de eventuais mudanças na solução inicial	Para essa primeira semana, o foco principal está em construir a base geral da solução, ou seja, adaptando o projeto desenvolvido até a experiência 6 para os novos requisitos e contexto. Além disso, será nesta primeira semana que será feita a busca e aquisição dos componentes listados na <b>lista de materiais</b> , para que seja possível adquirir em tempo hábil e considerando possíveis atrasos e problemas. Considerasse uma semana mais de alinhamento e start	10/03	16/03
2	1. Desenvolvimento do segundo Grupo de requisitos e seus respectivos testes 2. Confecção dos elementos externos a FPGA (Tapete e painel de usuário) 3. Avaliação geral do progresso	Durante essa segunda semana, a partir da experiência da semana 1 e seus respectivos feedbacks, o foco será em continuar a implementação e testes do projeto proposto e eventuais mudanças decorrentes. Será também, dado início ao desenvolvimento dos componentes externos a FPGA, montagem inicial A partir, da experiência gerada nesta semana será feita uma avaliação de condições, para alinharmos o que será possível ou não desenvolver e caminhos para se aproveitar o máximo	17/03	23/03
3	1. Testes dos componentes externos a FPGA	Semana caótica, onde precisamos garantir que a solução principal do nosso	24/03	30/03

	3. Avaliação geral do progresso	projeto está funcionando devidamente e com tudo que foi previsto implementado. Validação dos componentes junto a solução principal. Verificaremos se está realmente funcionando e se podem ser incorporadas ao projeto ou se precisam de ajustes ou até mesmo uma substituição por componentes mais simples.		
4	1. Preparação da apresentação final 2. Trabalho de Acabamento dos componentes 3. Ajustes Finos 4. Correção de eventuais problemas de última hora	Essa última semana tem como objetivo, polir o trabalho para a feira, então o foco está nos acabamentos finais e correções de possíveis problemas/imprevistos que possam vir a acontecer.	31/03	06/04

## 6 DESENVOLVIMENTO & RELATO DAS ATIVIDADES

Reservasse essa seção para, a partir do planejamento desenvolvido até tal, descrever as atividades práticas e possíveis mudanças ou comentários necessários. Para facilitar e organizar a baixo segue se distribuído para cada semana as atividades desenvolvidas e suas anotações

### 6.1 SEMANA 1

HORAS TRABALHADAS: 10 HORAS

#### 1 DESCRIÇÃO DO MÓDULO DA SEMANA

Para essa primeira semana, separamos um conjunto de requisitos para implementarmos no nosso projeto, além de fazer os devidos testes, todas essas atividades em maior detalhe podem ser consultadas nos módulos seguintes.

##### Módulos:

1. Pausa - Adaptação do jogo para incluir uma pausa em qualquer momento da jogada.
2. Modo Memory - Poder escolher entre a memória padrão, ou a memória personalizada, opção esta que permite que na primeira interação com o sistema o usuário crie uma sequência de jogo.
3. Painel do usuário - Primeiros passos na idealização e listagem de componentes necessários para montar o painel.
4. Tapete de controle - Idealização e modelagem do controle

#### 2 DETALHAMENTO DO PROJETO LÓGICO

O projeto consiste no desenvolvimento de um circuito digital que adapta o jogo da Memória que se baseia na reprodução de sequências de jogadas. Implementado em Verilog e sintetizado na FPGA DE0-CV, o sistema apresenta um desafio progressivo onde o jogador deve repetir corretamente as sequências exibidas. O jogo começa com a criação de uma sequência ou uma jogada pré determinada. O jogador vence ao completar corretamente as 16 rodadas, enquanto um erro ou o tempo limite de resposta resulta na derrota. A arquitetura do circuito é composta por dois módulos principais: o Fluxo de Dados (FD), responsável por armazenar e processar as sequências de jogadas, e a Unidade de Controle (UC), que coordena as operações do jogo,

gerencia as rodadas e verifica a entrada do jogador. A lógica do sistema segue um pseudocódigo estruturado, que busca explicar a correta interação entre os módulos e permitir a implementação eficiente do jogo na FPGA. Além do desenvolvimento do circuito, o projeto envolve testes e depuração para garantir seu correto funcionamento.

A estrutura lógica pode ser melhor observada através dos diagramas fornecidos previamente.

## 2.1 PROJETO DO FLUXO DE DADOS

O fluxo de dados é responsável pelo processamento das jogadas e pelo gerenciamento da lógica interna do jogo. Ele realiza a leitura das entradas do jogador, as compara com a sequência armazenada e controla a progressão do jogo com base nas respostas. A Figura abaixo apresenta o diagrama do fluxo de dados, destacando seus principais componentes e suas interações.

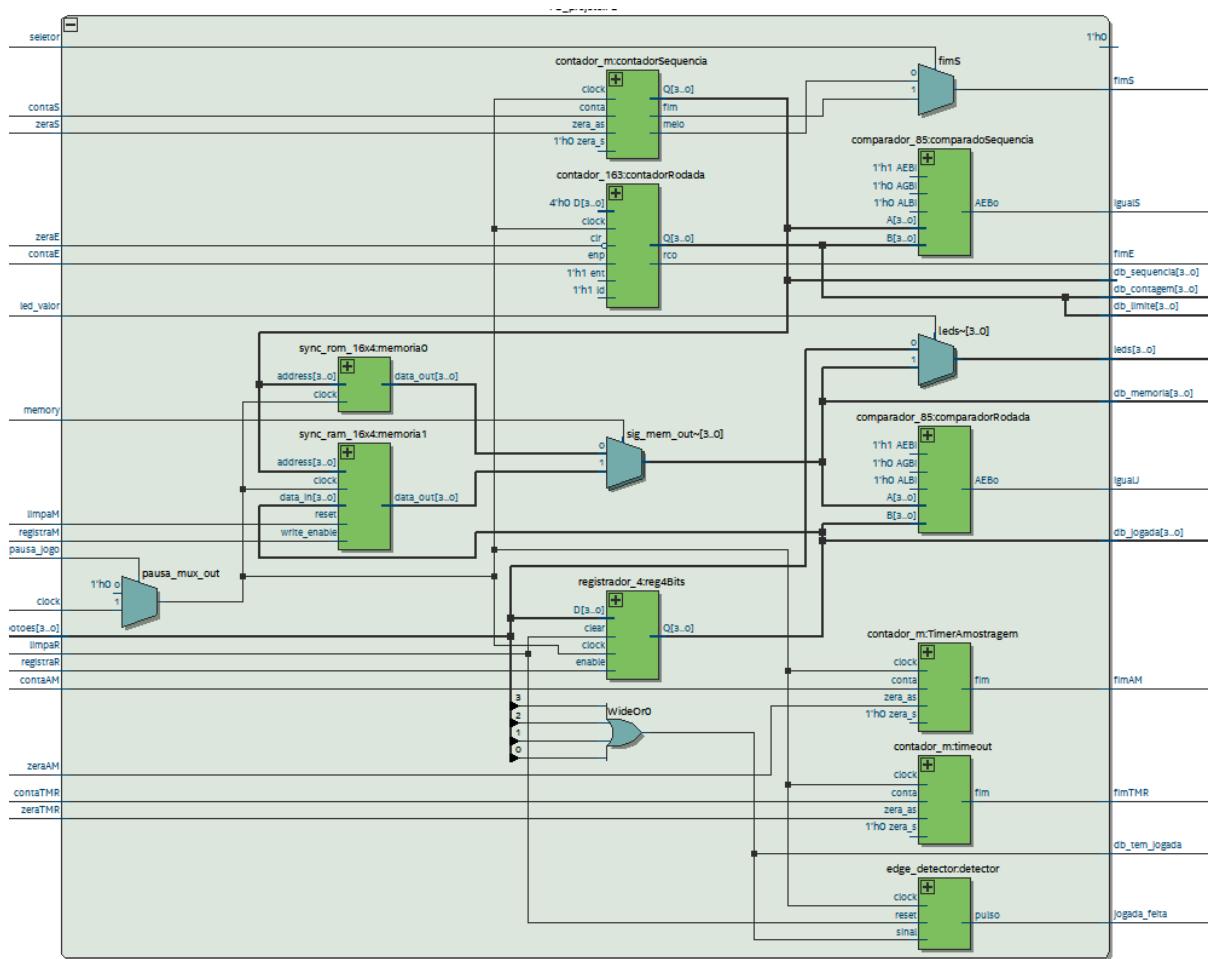


Fig. 1: Fluxo de Dados

Os principais elementos do fluxo de dados incluem:

- **Contador de Sequência:** Controla o número de jogadas realizadas em cada 4 rodadas, garantindo que a progressão do jogo ocorra corretamente.

- **Comparador:** Verifica se a jogada do usuário corresponde à sequência armazenada na memória interna, determinando se a jogada foi correta ou incorreta.
- **Memória Interna:** Responsável por armazenar as sequências de jogadas do jogo, permitindo a verificação da resposta do jogador a cada rodada.
- **Registradores:** Armazenam temporariamente os dados das jogadas e os resultados das comparações, permitindo a execução das verificações e o controle do fluxo do jogo.

O fluxo de dados interage diretamente com a Unidade de Controle (UC), que coordena sua operação com base nos sinais de entrada e nas regras do jogo. Esse design permite um funcionamento eficiente e estruturado do circuito digital.

## 2.2 PROJETO DA UNIDADE DE CONTROLE

A Unidade de Controle (UC) é implementada como uma máquina de estados finita (FSM), responsável por gerenciar a transição entre os diferentes estados do jogo e gerar os sinais de controle necessários para o fluxo de dados. A FSM determina quando exibir a sequência de LEDs, quando aceitar as jogadas do usuário e quando verificar a validade da jogada, além de identificar condições de vitória, derrota ou tempo esgotado. A Figura 2 apresenta o diagrama de transição de estados da Unidade de Controle.

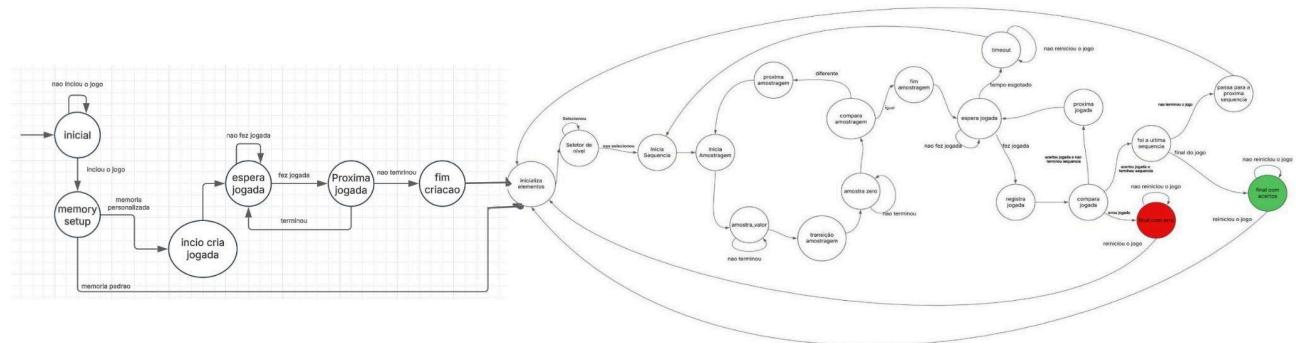


fig 2. Diagrama de estados da Unidade de Controle

O projeto foi estruturado conforme o modelo de Moore, buscando que as saídas dependessem exclusivamente do estado atual. Foram adicionados sinais relacionados ao contador de sequências para melhor gerenciamento da progressão do jogo. A Tabela 1 detalha cada estado da FSM.

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
<b>inicial</b>	Estado inicial do jogo	memory_setup	<b>iniciar = 1</b>
<b>memory_setup</b>	decide qual memória deve acessar	inicializa_elementos	<b>memory = 0</b>
<b>memory_setup</b>	decide qual memória deve acessar	inicia_cria_jogadas	<b>memory =</b>
<b>inicia_cria_jogadas</b>	inicia a criação das jogadas personalizadas	espera_jogada_criac ao	-
<b>espera_jogada_criac ao</b>	espera o usuário inserir uma jogada	proxima_jogada_criacao	<b>jogada = 1</b>
<b>proxima_jogada_criacao</b>	atualiza para a próxima jogada	espera_jogada_criac ao	<b>fimS = 0</b>
<b>proxima_jogada_criacao</b>	atualiza para a próxima jogada	fim_jogada_criacao	<b>fimS = 1</b>
<b>fim_jogada_criacao</b>	finaliza a fase de criacao	inicializa_elementos	-
<b>inicializa_elementos</b>	Inicializa os elementos do jogo	inicia_sequencia	-
<b>inicia_sequencia</b>	Inicia a sequência de jogadas	inicia_amostragem	-
<b>inicia_amostragem</b>	inicia amostragem dos leds	amostra_valor	-
<b>amostra_valor</b>	amostra o valor nos leds	transicao_amostragem	<b>fimAM = 1</b>
<b>transicao_amostragem</b>	reseta a contagem de amostragem	amostra_zero	-
<b>amostra_zero</b>	amostra os leds apagados	compara_amostragem	<b>fimAM=1</b>

Nome do Estado	Descrição do Estado	Próximo Estado	Condições e Justificativas para a Transição entre Estados
<b>compara_amostragem</b>	verifica se já terminou	proxima_amostragem	igualS = 0
<b>compara_amostragem</b>		fim_amostragem	fimS = 1
<b>proxima_amostragem</b>	transiciona para o proximo valor	inicia_amostragem	-
<b>fim_amostragem</b>	finaliza a secção de amostragem	espera_jogada	-
<b>espera_jogada</b>	Aguarda o jogador realizar uma jogada	registra_jogada / timeout	jogada realizada ou timeout
<b>registra_jogada</b>	Registra a jogada do usuário	compara_jogada	-
<b>compara_jogada</b>	Compara a jogada com o valor da memória	proxima_jogada / final_errou / ultima_sequencia	igualS = 1 (jogada correta) ou igualS = 0 (jogada errada)
<b>proxima_jogada</b>	Avança para a próxima jogada na sequência	espera_jogada	-
<b>ultima_sequencia</b>	Verifica se é a última jogada da sequência	proxima_sequencia / final_acertou	fimS = 1 (última jogada) ou fimS = 0 (não é a última)
<b>proxima_sequencia</b>	Avança para a próxima sequência de jogadas	espera_jogada	-
<b>final_errou</b>	Jogador cometeu um erro	inicializa_elementos	reiniciar jogo
<b>final_acertou</b>	Jogador acertou todas as sequências	inicializa_elementos	reiniciar jogo

Nome do Estado	Descrição do Estado	Próximo Estado	Condições e Justificativas para a Transição entre Estados
timeout	Tempo esgotado para realizar a jogada	inicializa_elementos	reiniciar jogo

Para fins de validação, é realizado a partir da vista do FSM no Quartus, uma comparação entre o proposto acima e o gerado. Demonstra-se então o correto funcionamento do circuito, conformando-se com o previsto:

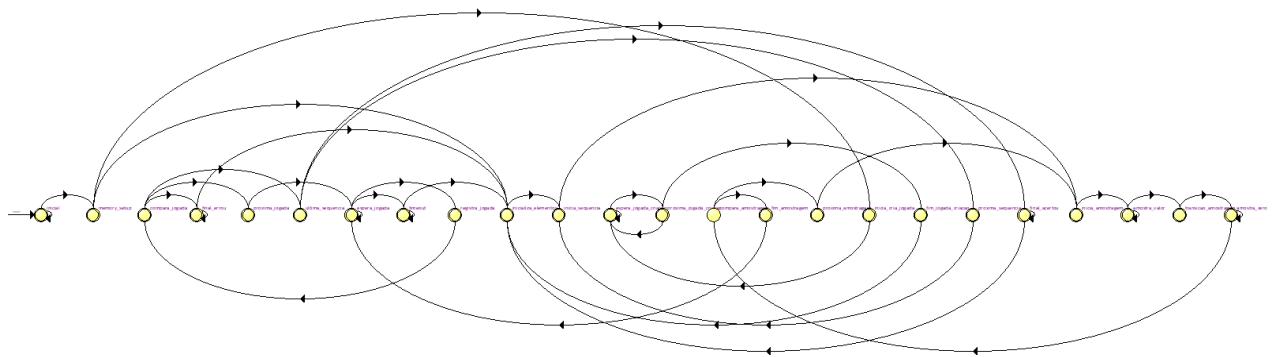


fig 3. FSM Unidade de Controle

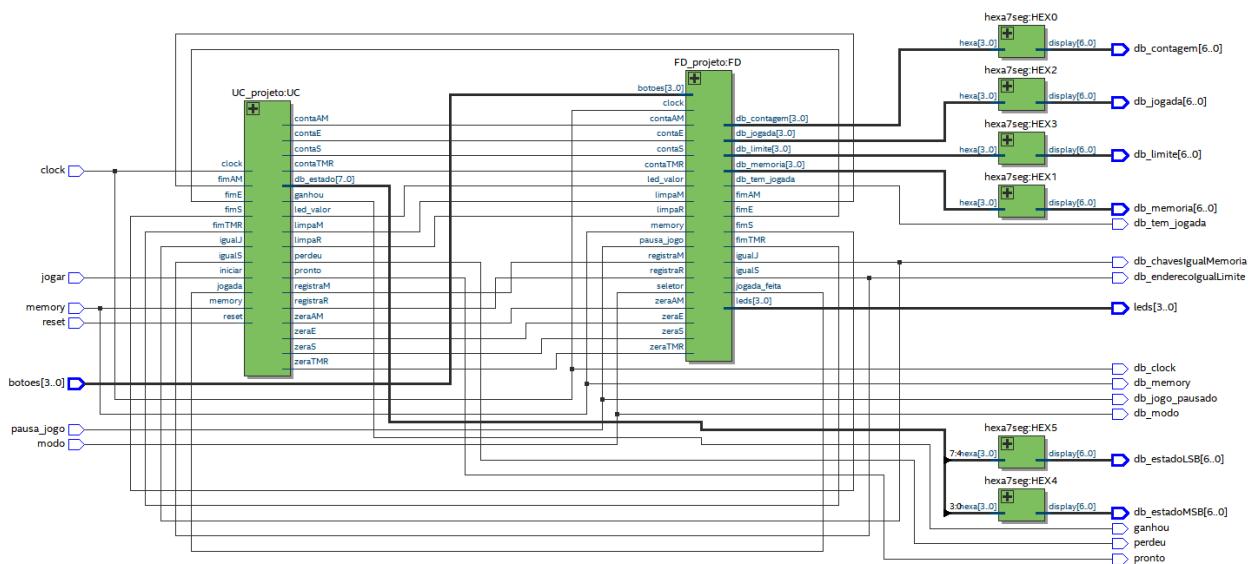
## 2.3 PROJETO DO SISTEMA DIGITAL

A integração entre o **Fluxo de Dados (FD)** e a **Unidade de Controle (UC)** é realizada por meio de sinais de controle e status, garantindo que o sistema funcione de maneira sincronizada. A **UC** emite sinais de controle que orientam a execução das operações no **FD**, enquanto este retorna sinais de status que indicam o progresso do jogo e permitem à **UC** decidir as próximas ações. A Figura 4 apresenta a visão do sistema no **RTL Viewer** a fim de ilustrar essa integração.

### Sinais de Controle da UC para o FD:

- **contaE**: Incrementa o contador de entrada.
- **contaS**: Incrementa o contador de sequência.
- **contaTMR**: Incrementa o temporizador do jogo.
- **contaAM**: Incrementa o temporizador do led.
- **limpaM**: Limpa a memória interna das jogadas.
- **limpaR**: Limpa os registradores do sistema.
- **registraM**: Armazena uma jogada na memória.
- **registraR**: Registra uma jogada do usuário.
- **zeraE**: Reseta o contador de entrada.
- **zeraS**: Reseta o contador de sequência.

- **zeraTMR:** Reseta o temporizador.
- **zeraAM:** Reseta o timer do led Sinais de Status do FD para a UC;
- **iguais:** Indica se a jogada do jogador corresponde à sequência armazenada.
- **igualJ:** Indica se a jogada atual é idêntica à anterior.
- **fimE:** Sinaliza o término da entrada de jogadas.
- **fimS:** Indica se o jogador completou toda a sequência da rodada.
- **fimTMR:** Indica se o tempo limite foi atingido.
- **fimAM:** Indica se o tempo limite de amostragem foi atingido.
- **jogada\_feita:** Confirma que uma jogada foi realizada.



**fig 4. RTL Viewer**

Como na experiência anterior, a interação entre os dois componentes permite que o sistema funcione de maneira sincronizada, pois a unidade de controle gerencia a execução do fluxo de dados em cada ciclo de clock.

## 2.4 PLANO DE TESTES DO SISTEMA E SIMULAÇÕES

Este capítulo tem como objetivo documentar a estratégia de testes adotada para o sistema desenvolvido, conforme descrito no Capítulo 3, bem como apresentar os resultados das simulações realizadas.

**Tabela 2 – Plano de Teste para Acertar com memória personalizada**

Etapa	Descrição do Estado	Sinais de Entrada	Saída Esperada	Deu certo?
1	Inicialização do Sistema – Reset e iniciar cria jogada	reset=1 memory=1 botões=0000	acertou = 0 db_estado = 21 db_memoria = 0	sim
2	Entrar a primeira jogada	botões=0001	db_estado = 15 db_memoria = 0	sim

Etapa	Descrição do Estado	Sinais de Entrada	Saída Esperada	Deu certo?
			db_contagem = 0	
3	Entrar a 2ª jogada	clock: ↑ botões=1000	db_estado = 21 db_memoria =0001 db_contagem = 1	sim
4	Entrar a 3ª jogada	clock: ↑ botões=0001	db_estado = 21 db_memoria =0010 db_contagem = 2	sim
5	Entrar a 4ª jogada	clock: ↑ botões=1000	db_estado = 21 db_memoria =0100 db_contagem = 3	sim
6	Entrar a 5ª jogada	clock: ↑ botões=0001	db_estado = 21 db_memoria =1000 db_contagem = 4	sim
7	Entrar a 6ª jogada	clock: ↑ botões=1000	db_estado = 21 db_memoria =0100 db_contagem = 5	sim
8	Entrar a 7ª jogada	clock: ↑ botões=0001	db_estado = 21 db_memoria =0010 db_contagem = 6	sim
9	Entrar a 8ª jogada	clock: ↑ botões=1000	db_estado = 21 db_memoria =0001 db_contagem = 7	sim
10	Entrar a 9ª jogada	clock: ↑ botões=0001	db_estado = 21 db_memoria =0001 db_contagem = 8	sim
11	Entrar a 10ª jogada	clock: ↑ botões=1000	db_estado = 21 db_memoria =0010 db_contagem = 9	sim
12	Entrar a 11ª jogada	clock: ↑ botões=0001	db_estado = 21 db_memoria =0010 db_contagem = 10	sim
13	Entrar a 12ª jogada	clock: ↑ botões=1000	db_estado = 21 db_memoria =0100 db_contagem = 11	sim
14	Entrar a 13ª jogada	clock: ↑ botões=0001	db_estado = 21 db_memoria =0100 db_contagem = 12	sim
15	Entrar a 14ª jogada	clock: ↑ botões=1000	db_estado = 21 db_memoria =1000 db_contagem = 13	sim
16	Entrar a 15ª jogada	clock: ↑ botões=0001	db_estado = 21 db_memoria =1000 db_contagem = 14	sim
17	Entrar a 16ª jogada (Final)	clock: ↑ botões=1000	db_estado = 3 db_memoria =0001	sim

Etapa	Descrição do Estado	Sinais de Entrada	Saída Esperada	Deu certo?
			db_contagem = 15	
18	Jogar a primeira jogada	Esperar amostragem clock: ↑ botões=0001	acertou = 0 db_estado = 3 db_memoria = 0 db_contagem = 0 Leds=0001	sim
19	Repetir jogada 1, jogar a 2 <sup>a</sup> jogada	Esperar amostragem clock: ↑ botões=1000	acertou = 0 db_estado = 3 db_memoria = 0001 db_contagem = 1 Leds= 0001 + 1000	sim
20	Repetir jogadas 1 e 2, jogar a 3 <sup>a</sup> jogada	Esperar amostragem clock: ↑ botões=0001	leds = acertou = 0 db_estado = 3 db_memoria = 1000 db_contagem = 2 Leds= Jogadas anteriores + 0001	sim
21	Repetir jogadas 1 a 3, jogar a 4 <sup>a</sup> jogada	Esperar amostragem clock: ↑ botões=1000	acertou = 0 db_estado = 3 db_memoria = 0001 db_contagem = 3 Leds= Jogadas anteriores + 1000	sim
22	Repetir jogadas 1 a 4, jogar a 5 <sup>a</sup> jogada	Esperar amostragem clock: ↑ botões=0001	acertou = 0 db_estado = 3 db_memoria = 1000 db_contagem = 4 Leds= Jogadas anteriores + 0001	sim
23	Repetir jogadas 1 a 5, jogar a 6 <sup>a</sup> jogada	Esperar amostragem clock: ↑ botões=1000	acertou = 0 db_estado = 3 db_memoria = 0001 db_contagem = 5 Leds= Jogadas anteriores + 1000	sim
24	Repetir jogadas 1 a 6, jogar a 7 <sup>a</sup> jogada	Esperar amostragem clock: ↑ botões=0001	acertou = 0 db_estado = 3 db_memoria = 1000 db_contagem = 6 Leds= Jogadas anteriores + 0001	sim
25	Repetir jogadas 1 a 7, jogar a 8 <sup>a</sup> jogada	Esperar amostragem clock: ↑ botões=1000	acertou = 0 db_estado = 3 db_memoria = 0001 db_contagem = 7 Leds= Jogadas anteriores + 1000	sim

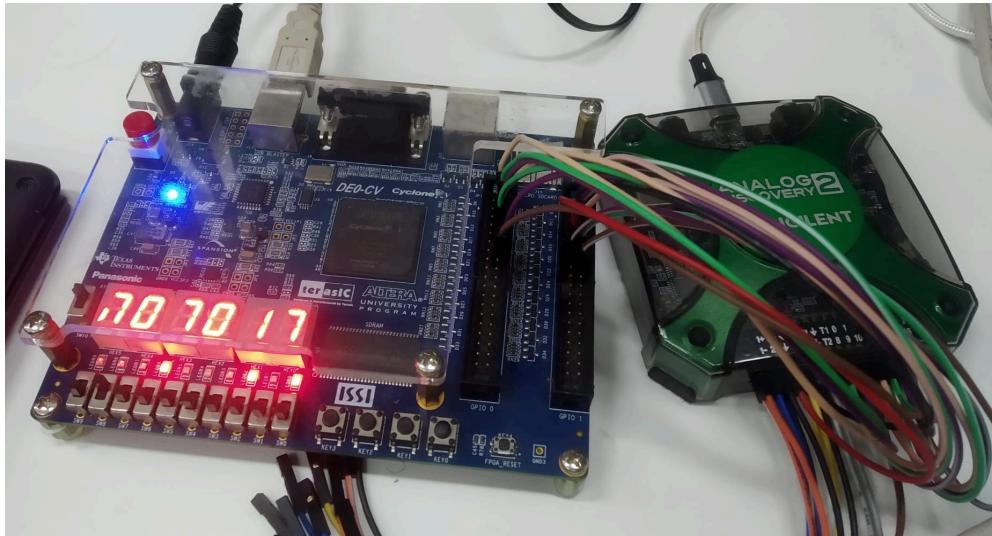
Etapa	Descrição do Estado	Sinais de Entrada	Saída Esperada	Deu certo?
26	Repetir jogadas 1 a 8, jogar a 9 <sup>a</sup> jogada	Esperar amostragem clock: ↑ botões=0001	acertou = 0 db_estado = 3 db_memoria = 1000 db_contagem = 8 Leds= Jogadas anteriores + 0001	sim
27	Repetir jogadas 1 a 9, jogar a 10 <sup>a</sup> jogada	Esperar amostragem clock: ↑ botões=1000	acertou = 0 db_estado = 3 db_memoria = 0001 db_contagem = 9 Leds= Jogadas anteriores + 1000	sim
28	Repetir jogadas 1 a 10, jogar a 11 <sup>a</sup> jogada	Esperar amostragem clock: ↑ botões=0001	acertou = 0 db_estado = 3 db_memoria = 1000 db_contagem = A Leds= Jogadas anteriores + 0001	sim
29	Repetir jogadas 1 a 11, jogar a 12 <sup>a</sup> jogada	Esperar amostragem clock: ↑ botões=1000	acertou = 0 db_estado = 3 db_memoria = 0001 db_contagem = B Leds= Jogadas anteriores + 1000	sim
30	Repetir jogadas 1 a 12, jogar a 13 <sup>a</sup> jogada	Esperar amostragem clock: ↑ botões=0001	acertou = 0 db_estado = 3 db_memoria = 1000 db_contagem = C Leds= Jogadas anteriores + 0001	sim
31	Repetir jogadas 1 a 13, jogar a 14 <sup>a</sup> jogada	Esperar amostragem clock: ↑ botões=1000	acertou = 0 db_estado = 3 db_memoria = 0001 db_contagem = D Leds= Jogadas anteriores + 1000	sim
32	Repetir jogadas 1 a 14, jogar a 15 <sup>a</sup> jogada	Esperar amostragem clock: ↑ botões=0001	db_estado = 3 db_memoria = 1000 db_contagem = E Leds= Jogadas anteriores + 0001	sim
33	Repetir jogadas 1 a 15, jogar a 16 <sup>a</sup> jogada (Final)	Esperar amostragem clock: ↑ botões=1000	db_estado = 3 db_memoria = 0001 db_contagem = F Leds= Jogadas anteriores + 1000 ganhou = 1 pronto = 1	sim

**Tabela 3 – Plano de Teste para Pausar na quarta jogada**

<b>Etapa</b>	<b>Descrição do Estado</b>	<b>Sinais de Entrada</b>	<b>Saída Esperada</b>	<b>Deu certo?</b>
1	Inicialização do Sistema – Reset e preparação.	clock: ↑ botões=0000 jogar=1 pausa_jogo = 0	acertou = 0 db_estado = 3 db_memoria =0 db_contagem = 0 Leds=0001	
2	Jogar a primeira jogada	clock: ↑ botões=0000 pausa_jogo = 0	acertou = 0 db_estado = 3 db_memoria =0001 db_contagem = 1 Leds= 0001 + 0010	
3	Repetir jogada 1, jogar a 2ª jogada	clock: ↑ botões=0001 pausa_jogo = 0	leds = acertou = 0 db_estado = 3 db_memoria =0010 db_contagem = 2 Leds= Jogadas anteriores + 0100	
4	Repetir jogadas 1 e 2, jogar a 3ª jogada	clock: ↑ botões=0010 pausa_jogo = 0	acertou = 0 db_estado = 3 db_memoria =0100 db_contagem = 3 Leds= Jogadas anteriores + 1000	
5	Repetir jogadas 1 a 3, esperar o 2s	clock: ↑ botões=0010 pausa_jogo = 0	errou= 0 pronto = 0 db_estado = A db_memoria =1000 db_contagem = 3	
6	acionar a pausa e espera 5s	pausa_jogo = 1	errou= 0 pronto = 0 db_contagem = 3 db_estado = A db_memoria =1000 jogo_pausado=1	
7	despausa o jogo e espera 1s	clock: ↑ pausa_jogo = 0	errou= 1 pronto = 1 db_contagem = 3 db_estado = 12 db_memoria =1000 jogo_pausado=0	

## 2.5 IMPLANTAÇÃO DO PROJETO

Este capítulo tem o objetivo de documentar as atividades práticas de execução do projeto desenvolvido e documentado no capítulo 2.3 no ambiente do Laboratório Digital.



**fig 5. Montagem FPGA & ANALOG DISCOVERY**

### 2.5.1 PINAGEM DA PLACA FPGA

A pinagem foi definida no Intel Quartus Prime utilizando o Pin Planner, associando os sinais do projeto aos pinos físicos da FPGA Cyclone V (placa DE0-CV). A figura abaixo apresenta o plano de pinagem detalhado:

Pin Planner - C:/Users/gabri/CODE/VERILOG/QUARTUS/PULodSAPO/QUARTUS/circuito_projeto - circuito_projeto																				
Named:		Edit		View		Processing		Tools		Window		Help		Search altera.com						
Tasks	Report	Groups	Node Name	Direction	Location	I/O Bank	VREF Group	I/O Standard	Reserved	Current Strength	Slew Rate	Differential Pair	Analog Setting	(B/V)CC_	er I/O Pin Test	iated Refclk	mon Mode	itter Slew Rate	Differential O	er Cor
			bootees[3]	Input	PIN_T22	5A	B5A_N0	2.5 V		12mA...auto										
			bootees[2]	Input	PIN_R22	5A	B5A_N0	2.5 V		12mA...auto										
			bootees[1]	Input	PIN_M21	5B	B5B_N0	2.5 V		12mA...auto										
			bootees[0]	Input	PIN_K22	5B	B5B_N0	2.5 V		12mA...auto										
			clock	Input	PIN_B16	7A	B7A_N0	2.5 V		12mA...auto										
			db_dpa_emoria	Output	PIN_M2	2A	B2A_N0	2.5 V		12mA...auto	1 (default)									
			db_clock	Output	PIN_M1	2A	B2A_N0	2.5 V		12mA...auto	1 (default)									
			db_contagen[6]	Output	PIN_A422	4A	B4A_N0	2.5 V		12mA...auto	1 (default)									
			db_contagen[5]	Output	PIN_Y21	4A	B4A_N0	2.5 V		12mA...auto	1 (default)									
			db_contagen[4]	Output	PIN_Y22	4A	B4A_N0	2.5 V		12mA...auto	1 (default)									
			db_contagen[3]	Output	PIN_W21	4A	B4A_N0	2.5 V		12mA...auto	1 (default)									
			db_contagen[2]	Output	PIN_W22	4A	B4A_N0	2.5 V		12mA...auto	1 (default)									
			db_contagen[1]	Output	PIN_V21	4A	B4A_N0	2.5 V		12mA...auto	1 (default)									
			db_contagen[0]	Output	PIN_U21	4A	B4A_N0	2.5 V		12mA...auto	1 (default)									
			db_ende_llimit	Output	PIN_W2	2A	B2A_N0	2.5 V		12mA...auto	1 (default)									
			db_estadoLSB[6]	Output	PIN_W19	4A	B4A_N0	2.5 V		12mA...auto	1 (default)									
			db_estadoLSB[5]	Output	PIN_C2	2A	B2A_N0	2.5 V		12mA...auto	1 (default)									
			db_estadoLSB[4]	Output	PIN_C1	2A	B2A_N0	2.5 V		12mA...auto	1 (default)									
			db_estadoLSB[3]	Output	PIN_P14	4A	B4A_N0	2.5 V		12mA...auto	1 (default)									
			db_estadoLSB[2]	Output	PIN_T14	4A	B4A_N0	2.5 V		12mA...auto	1 (default)									
			db_estadoMSB[1]	Output	PIN_M8	3B	B3B_N0	2.5 V		12mA...auto	1 (default)									
			db_estadoMSB[0]	Output	PIN_M9	3B	B3B_N0	2.5 V		12mA...auto	1 (default)									
			db_estadoMSB[6]	Output	PIN_P9	3B	B3B_N0	2.5 V		12mA...auto	1 (default)									
			db_estadoMSB[5]	Output	PIN_Y15	4A	B4A_N0	2.5 V		12mA...auto	1 (default)									
			db_estadoMSB[4]	Output	PIN_U15	4A	B4A_N0	2.5 V		12mA...auto	1 (default)									
			db_estadoMSB[3]	Output	PIN_U16	4A	B4A_N0	2.5 V		12mA...auto	1 (default)									
			db_estadoMSB[2]	Output	PIN_V20	4A	B4A_N0	2.5 V		12mA...auto	1 (default)									
			db_estadoMSB[1]	Output	PIN_Y20	4A	B4A_N0	2.5 V		12mA...auto	1 (default)									
			db_estadoMSB[0]	Output	PIN_U20	4A	B4A_N0	2.5 V		12mA...auto	1 (default)									
			db_jogada[6]	Output	PIN_A01	4A	B4A_N0	2.5 V		12mA...auto	1 (default)									
			db_jogada[5]	Output	PIN_AB22	4A	B4A_N0	2.5 V		12mA...auto	1 (default)									
			db_jogada[4]	Output	PIN_V14	4A	B4A_N0	2.5 V		12mA...auto	1 (default)									
			db_jogada[3]	Output	PIN_Y14	4A	B4A_N0	2.5 V		12mA...auto	1 (default)									
			db_jogada[2]	Output	PIN_AA10	3B	B3B_N0	2.5 V		12mA...auto	1 (default)									
			db_jogada[1]	Output	PIN_AB17	4A	B4A_N0	2.5 V		12mA...auto	1 (default)									
			db_jogada[0]	Output	PIN_Y19	4A	B4A_N0	2.5 V		12mA...auto	1 (default)									
			db_jogado_usado	Output	PIN_Y3	2A	B2A_N0	2.5 V...fault		12mA...auto	1 (default)									
			db_limite[6]	Output	PIN_V19	4A	B4A_N0	2.5 V		12mA...auto	1 (default)									
			db_limite[5]	Output	PIN_V18	4A	B4A_N0	2.5 V		12mA...auto	1 (default)									

**fig 6. Pin planner**

## 2.5.2 ESTRATÉGIA DE MONTAGEM

A implementação do projeto na **placa FPGA DE0-CV** seguiu um fluxo estruturado para assegurar a correta configuração e funcionamento do sistema. As etapas adotadas foram:

### 1. Compilação do projeto no Quartus Prime

- O código Verilog foi desenvolvido e revisado conforme os requisitos da experiência.
- O projeto foi sintetizado e analisado utilizando as ferramentas **RTL Viewer** e **State Machine Viewer**.
- A designação de pinos da FPGA foi configurada no **Pin Planner**, com base nas tabelas fornecidas pela apostila.

### 2. Transferência do arquivo de configuração para a FPGA

- Após a compilação, o arquivo **.sof** foi gerado e carregado na FPGA através do **Intel Quartus Prime Programmer**.
- A programação da placa **DE0-CV** foi realizada via conexão **USB Blaster**.

### 3. Verificação e depuração do circuito

- Os sinais de saída foram analisados utilizando os **LEDs** e os **displays de 7 segmentos** da FPGA.
- Saídas adicionais de depuração foram implementadas para monitorar estados internos do sistema buscando facilitar a identificação e correção de possíveis falhas.
- A validação inicial incluiu testes manuais e simulações no **ModelSim** para verificar a resposta do circuito a diferentes cenários do jogo.

Esse processo garante uma implementação eficiente e confiável, e permite ajustes precisos durante os testes e facilitando a análise do funcionamento do sistema.

## 2.5.3 ESTRATÉGIA DE DEPURAÇÃO

A depuração do projeto ocorre principalmente por meio da análise dos sinais exibidos nos **LEDs** e nos **displays de 7 segmentos** da FPGA, eliminando a necessidade de 25 de equipamentos adicionais. Para facilitar o diagnóstico, foram implementadas diversas saídas de depuração que monitoram o estado interno do circuito digital. Sempre que falhas são identificadas, ajustes devem ser aplicados no código **Verilog**, seguidos de recompilação no **Quartus Prime**. Além disso, simulações no **ModelSim** validam o comportamento lógico antes da implementação física. Outra etapa fundamental é a **verificação das conexões físicas da FPGA**, nota-se que, diversas vezes, há desatenção no momento de associar os pinos no **Pin Planner**. Desse modo, é necessário prevenir problemas de hardware, como ligações incorretas ou mau contato. Após cada modificação, testes específicos asseguram que as correções não comprometem funcionalidades previamente validadas. Esse processo iterativo preserva a integridade do sistema e permite que o circuito opere conforme o esperado na implementação final.

## 2.5.4 EXECUÇÃO PRÁTICA DOS CENÁRIOS DE TESTES

Tabela 4 – Plano de Teste para Pausar na quarta jogada

Etapa	Descrição do Estado	Sinais de Entrada	Saída Esperada	Deu certo?
-------	---------------------	-------------------	----------------	------------

1	Inicialização do Sistema – Reset e preparação.	clock: ↑ botões=0000 jogar=1 pausa_jogo = 0	acertou = 0 db_estado = 3 db_memoria =0 db_contagem = 0 Leds=0001	
2	Jogar a primeira jogada	clock: ↑ botões=0000 pausa_jogo = 0	acertou = 0 db_estado = 3 db_memoria =0001 db_contagem = 1 Leds= 0001 + 0010	
3	Repetir jogada 1, jogar a 2ª jogada	clock: ↑ botões=0001 pausa_jogo = 0	leds = acertou = 0 db_estado = 3 db_memoria =0010 db_contagem = 2 Leds= Jogadas anteriores + 0100	
4	Repetir jogadas 1 e 2, jogar a 3ª jogada	clock: ↑ botões=0010 pausa_jogo = 0	acertou = 0 db_estado = 3 db_memoria =0100 db_contagem = 3 Leds= Jogadas anteriores + 1000	
5	Repetir jogadas 1 a 3, esperar o 2s	clock: ↑ botões=0010 pausa_jogo = 0	errou= 0 pronto = 0 db_estado = A db_memoria =1000 db_contagem = 3	
6	acionar a pausa e espera 5s	pausa_jogo = 1	errou= 0 pronto = 0 db_contagem = 3 db_estado = A db_memoria =1000 jogo_pausado=1	
7	despausa o jogo e espera 1s	clock: ↑ pausa_jogo = 0	errou= 1 pronto = 1 db_contagem = 3 db_estado = 12 db_memoria =1000 jogo_pausado=0	

Tabela 5 – Plano de Teste para Acertar com memória personalizada

Etapa	Descrição do Estado	Sinais de Entrada	Saída Esperada	Deu certo?
1	Inicialização do Sistema – Reset e iniciar cria jogada	reset=1 memory=1 botões=0000	acertou = 0 db_estado = 21 db_memoria = 0	sim

Etapa	Descrição do Estado	Sinais de Entrada	Saída Esperada	Deu certo?
2	Entrar a primeira jogada	botões=0001	db_estado = 15 db_memoria =0 db_contagem = 0	sim
3	Entrar a 2ª jogada	clock: ↑ botões=1000	db_estado = 21 db_memoria =0001 db_contagem = 1	sim
4	Entrar a 3ª jogada	clock: ↑ botões=0001	db_estado = 21 db_memoria =0010 db_contagem = 2	sim
5	Entrar a 4ª jogada	clock: ↑ botões=1000	db_estado = 21 db_memoria =0100 db_contagem = 3	sim
6	Entrar a 5ª jogada	clock: ↑ botões=0001	db_estado = 21 db_memoria =1000 db_contagem = 4	sim
7	Entrar a 6ª jogada	clock: ↑ botões=1000	db_estado = 21 db_memoria =0100 db_contagem = 5	sim
8	Entrar a 7ª jogada	clock: ↑ botões=0001	db_estado = 21 db_memoria =0010 db_contagem = 6	sim
9	Entrar a 8ª jogada	clock: ↑ botões=1000	db_estado = 21 db_memoria =0001 db_contagem = 7	sim
10	Entrar a 9ª jogada	clock: ↑ botões=0001	db_estado = 21 db_memoria =0001 db_contagem = 8	sim
11	Entrar a 10ª jogada	clock: ↑ botões=1000	db_estado = 21 db_memoria =0010 db_contagem = 9	sim
12	Entrar a 11ª jogada	clock: ↑ botões=0001	db_estado = 21 db_memoria =0010 db_contagem = 10	sim
13	Entrar a 12ª jogada	clock: ↑ botões=1000	db_estado = 21 db_memoria =0100 db_contagem = 11	sim
14	Entrar a 13ª jogada	clock: ↑ botões=0001	db_estado = 21 db_memoria =0100 db_contagem = 12	sim
15	Entrar a 14ª jogada	clock: ↑ botões=1000	db_estado = 21 db_memoria =1000 db_contagem = 13	sim
16	Entrar a 15ª jogada	clock: ↑ botões=0001	db_estado = 21 db_memoria =1000 db_contagem = 14	sim

Etapa	Descrição do Estado	Sinais de Entrada	Saída Esperada	Deu certo?
17	Entrar a 16 <sup>a</sup> jogada (Final)	clock: ↑ botões=1000	db_estado = 3 db_memoria =0001 db_contagem = 15	sim
18	Jogar a primeira jogada	Esperar amostragem clock: ↑ botões=0001	acertou = 0 db_estado = 3 db_memoria = 0 db_contagem = 0 Leds=0001	sim
19	Repetir jogada 1, jogar a 2 <sup>a</sup> jogada	Esperar amostragem clock: ↑ botões=1000	acertou = 0 db_estado = 3 db_memoria = 0001 db_contagem = 1 Leds= 0001 + 1000	sim
20	Repetir jogadas 1 e 2, jogar a 3 <sup>a</sup> jogada	Esperar amostragem clock: ↑ botões=0001	leds = acertou = 0 db_estado = 3 db_memoria = 1000 db_contagem = 2 Leds= Jogadas anteriores + 0001	sim
21	Repetir jogadas 1 a 3, jogar a 4 <sup>a</sup> jogada	Esperar amostragem clock: ↑ botões=1000	acertou = 0 db_estado = 3 db_memoria = 0001 db_contagem = 3 Leds= Jogadas anteriores + 1000	sim
22	Repetir jogadas 1 a 4, jogar a 5 <sup>a</sup> jogada	Esperar amostragem clock: ↑ botões=0001	acertou = 0 db_estado = 3 db_memoria = 1000 db_contagem = 4 Leds= Jogadas anteriores + 0001	sim
23	Repetir jogadas 1 a 5, jogar a 6 <sup>a</sup> jogada	Esperar amostragem clock: ↑ botões=1000	acertou = 0 db_estado = 3 db_memoria = 0001 db_contagem = 5 Leds= Jogadas anteriores + 1000	sim
24	Repetir jogadas 1 a 6, jogar a 7 <sup>a</sup> jogada	Esperar amostragem clock: ↑ botões=0001	acertou = 0 db_estado = 3 db_memoria = 1000 db_contagem = 6 Leds= Jogadas anteriores + 0001	sim
25	Repetir jogadas 1 a 7, jogar a 8 <sup>a</sup> jogada	Esperar amostragem clock: ↑ botões=1000	acertou = 0 db_estado = 3 db_memoria = 0001 db_contagem = 7	sim

Etapa	Descrição do Estado	Sinais de Entrada	Saída Esperada	Deu certo?
			Leds= Jogadas anteriores + 1000	
26	Repetir jogadas 1 a 8, jogar a 9 <sup>a</sup> jogada	Esperar amostragem clock: ↑ botões=0001	acertou = 0 db_estado = 3 db_memoria = 1000 db_contagem = 8 Leds= Jogadas anteriores + 0001	sim
27	Repetir jogadas 1 a 9, jogar a 10 <sup>a</sup> jogada	Esperar amostragem clock: ↑ botões=1000	acertou = 0 db_estado = 3 db_memoria = 0001 db_contagem = 9 Leds= Jogadas anteriores + 1000	sim
28	Repetir jogadas 1 a 10, jogar a 11 <sup>a</sup> jogada	Esperar amostragem clock: ↑ botões=0001	acertou = 0 db_estado = 3 db_memoria = 1000 db_contagem = A Leds= Jogadas anteriores + 0001	sim
29	Repetir jogadas 1 a 11, jogar a 12 <sup>a</sup> jogada	Esperar amostragem clock: ↑ botões=1000	acertou = 0 db_estado = 3 db_memoria = 0001 db_contagem = B Leds= Jogadas anteriores + 1000	sim
30	Repetir jogadas 1 a 12, jogar a 13 <sup>a</sup> jogada	Esperar amostragem clock: ↑ botões=0001	acertou = 0 db_estado = 3 db_memoria = 1000 db_contagem = C Leds= Jogadas anteriores + 0001	sim
31	Repetir jogadas 1 a 13, jogar a 14 <sup>a</sup> jogada	Esperar amostragem clock: ↑ botões=1000	acertou = 0 db_estado = 3 db_memoria = 0001 db_contagem = D Leds= Jogadas anteriores + 1000	sim
32	Repetir jogadas 1 a 14, jogar a 15 <sup>a</sup> jogada	Esperar amostragem clock: ↑ botões=0001	db_estado = 3 db_memoria = 1000 db_contagem = E Leds= Jogadas anteriores + 0001	sim
33	Repetir jogadas 1 a 15, jogar a 16 <sup>a</sup> jogada (Final)	Esperar amostragem clock: ↑ botões=1000	db_estado = 3 db_memoria = 0001 db_contagem = F Leds= Jogadas anteriores + 1000	sim

Etapa	Descrição do Estado	Sinais de Entrada	Saída Esperada	Deu certo?
			ganhou = 1 pronto = 1	

### 3 Periféricos

Os componentes periféricos do projetos são divididos como previsto anteriormente em duas partes: **Painel do Usuário** e **Tapete Controle**, partes essas que são descritas em maiores detalhes logo abaixo. Para essa semana 1 focamos em desenvolver o modelo do que acreditamos ser a melhor alternativa dada os recursos e requisitos, além de fazer o levantamento do orçamento que viabilizasse tais componentes, Lista essa que se encontra na seção **4. Lista de Materiais**, essa seção serve apenas para documentar quais novos componentes, fornecedores, preço foram atualizados. Além de mostrar como foi desenvolvido todo o modelo.

#### 3.1 Painel do Usuário

Nosso primeiro foco ao desenvolver o modelo, foi pensar em algo agradável, simples mas que ainda assim mantivesse todos os elementos necessários para que o usuário manipulasse e reagisse ao sistemas da melhor forma possível.

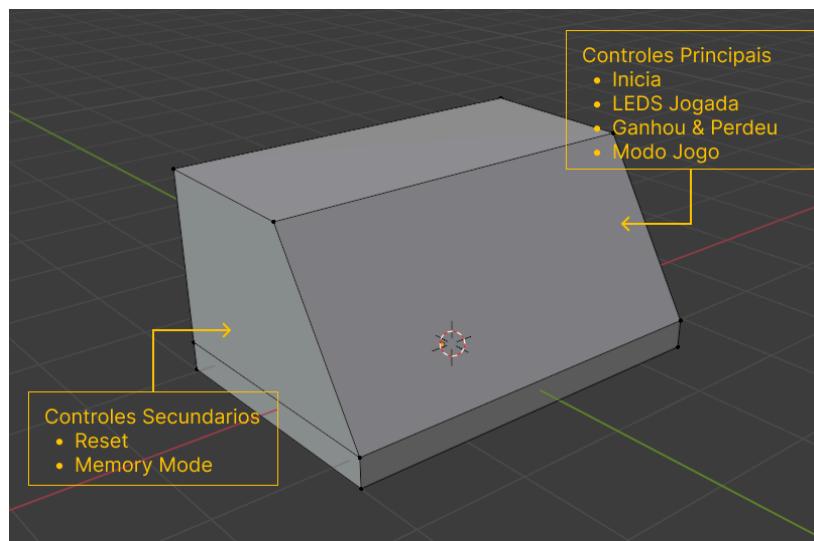


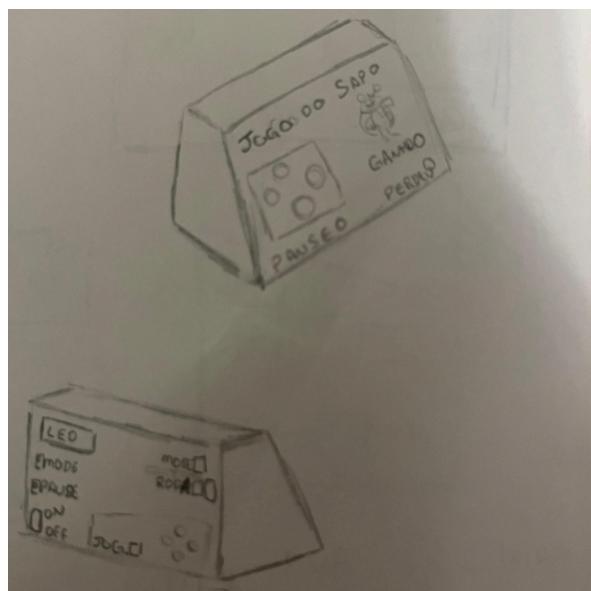
fig 7. Conceito Inicial Painel Usuario

Para confecção vamos cortar algumas peças em MDF e fazer a montagem, nesse primeiro momento, estamos focando em quais componentes vão no painel e em que parte, posteriormente, será adicionado um modelo com mais detalhes de layout, indicando corretamente onde cada componente será posicionado para o corte e montagem.

Em adição, foi elaborado um esboço do painel a ser construído. Este possuirá 2 interfaces, cada uma em uma posição diferente do painel:

- **Interface de configuração:** Destinado ao cuidador ou educador da criança. Possui leds e displays para saída, bem como botões e switches que permitirão o controle do jogo, pause e seleção do modo.
- **Interface de jogo:** Mais colorido e lúdico, com maior identidade visual. Destinado às crianças que estiverem jogando o jogo. Contém apenas os leds de jogada, bem como ganhou, perdeu e pause

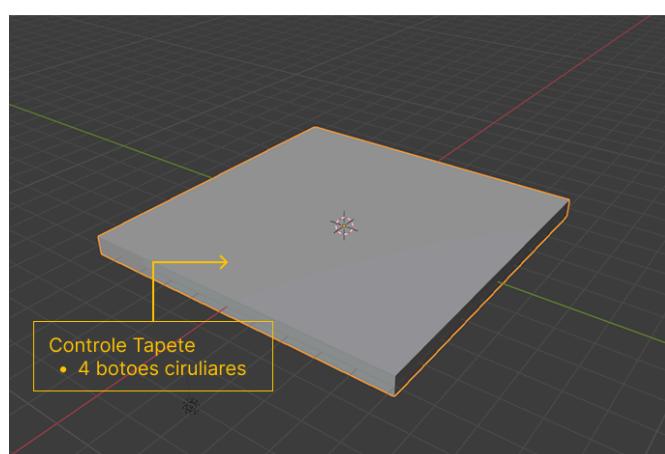
Para o painel, utilizaremos leds simples e displays de 7 segmentos para saída de informações, switches e botões para entrada do usuário e possivelmente um display led LCD que será controlado por um arduino UNO utilizando um adaptador I2C. A possibilidade de sua implementação ainda deve ser discutida e melhor analisada.



**fig 8. Esboço Conceitual**

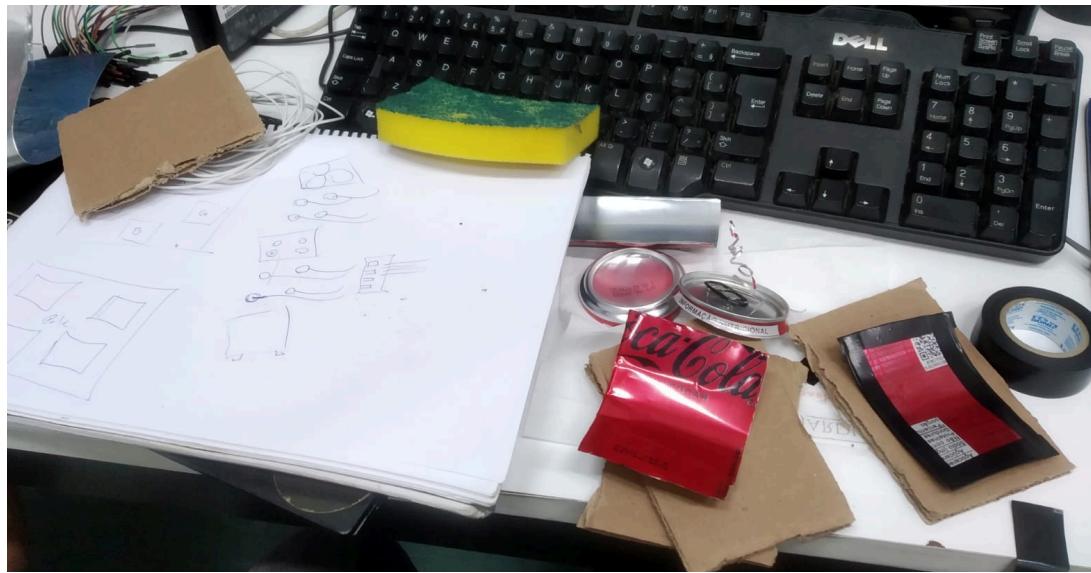
### 3.2 Tapete Controle

Para o tapete, nossa ideia inicial seria utilizar uma plataforma mais sólida para que o usuário pudesse ficar de pé e acionasse os 4 botões que estarão dispostos nela.



**fig 9. Conceito Inicial Tapete**

Durante a atividade laboratorial do dia 10/03/2025, foi desenvolvido nosso primeiro POC, uma versão bem mais simples para poder validar o tipo de estrutura mecânica que acreditamos atender ao projeto final. Além disso, pretendemos utilizar esse modelo para um futuro teste em que pretendemos integrar todos os componentes dos dois periféricos ao módulo central, antes de finalmente, desenvolver uma versão em maior escala ou definitiva.



**fig 10. POC tapete**

#### **4. CONCLUSÃO SEMANA 1**

Conseguimos desenvolver tudo que nos propomos para essa primeira semana, alguns detalhes ainda precisam ser validados durante o laboratório, mas para o planejamento acreditamos que está o mais completo possível, não só rico em detalhes descritivos mas também em pontos técnicos.

## 6.2 SEMANA 2

HORAS TRABALHADAS: 8H

### 1. DESCRIÇÃO DO MÓDULO DA SEMANA

Para essa segunda semana, separamos um conjunto de requisitos para implementarmos no nosso projeto, além de fazer os devidos testes, todas essas atividades em maior detalhe podem ser consultadas nos módulos seguintes.

**Módulos:**

1. Modo de jogo - Permite escolher entre o modo sequencial, onde as jogadas são cumulativas para cada round, ou o modo onde a cada amostragem é feita uma jogada.
2. Métricas - Permite acompanhar os desempenho do usuário para, número de timeouts, número de acertos, tempo médio.
5. Painel do usuário - Fazer a montagem e validação da interface em uma protoboard
6. Tapete de controle - Fazer a montagem simplificada e testar junto da interface na protoboard.

### 2. DETALHAMENTO DO PROJETO LÓGICO

#### 2.1 PROJETO DO FLUXO DE DADOS

O fluxo de dados é responsável pelo processamento das jogadas e pelo gerenciamento da lógica interna do jogo. Ele realiza a leitura das entradas do jogador, as compara com a sequência armazenada e controla a progressão do jogo com base nas respostas. A Figura abaixo apresenta o diagrama do fluxo de dados, destacando seus principais componentes e suas interações.

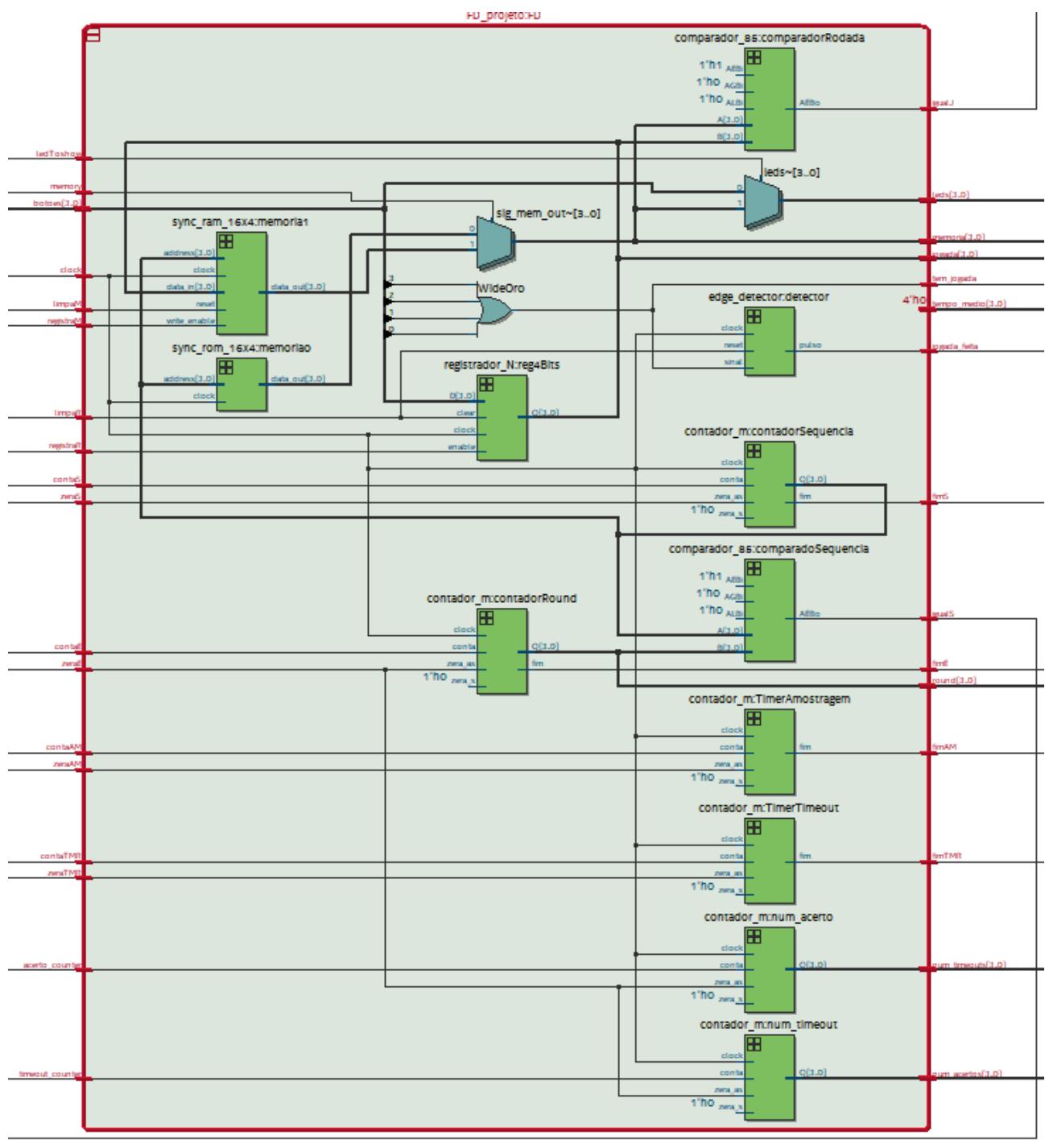


Fig. 1: Fluxo de Dados

Os principais elementos do fluxo de dados incluem:

- **Contador de Sequência:** Controla o número de jogadas realizadas em cada 4 rodadas, garantindo que a progressão do jogo ocorra corretamente.
- **Comparador:** Verifica se a jogada do usuário corresponde à sequência armazenada na memória interna, determinando se a jogada foi correta ou incorreta.
- **Memória Interna:** Responsável por armazenar as sequências de jogadas do jogo, permitindo a verificação da resposta do jogador a cada rodada.

- **Registradores:** Armazenam temporariamente os dados das jogadas e os resultados das comparações, permitindo a execução das verificações e o controle do fluxo do jogo.

O fluxo de dados interage diretamente com a Unidade de Controle (UC), que coordena sua operação com base nos sinais de entrada e nas regras do jogo. Esse design permite um funcionamento eficiente e estruturado do circuito digital.

### 3.3 PROJETO DA UNIDADE DE CONTROLE

A Unidade de Controle (UC) é implementada como uma máquina de estados finita (FSM), responsável por gerenciar a transição entre os diferentes estados do jogo e gerar os sinais de controle necessários para o fluxo de dados. A FSM determina quando exibir a sequência de LEDs, quando aceitar as jogadas do usuário e quando verificar a validade da jogada, além de identificar condições de vitória, derrota ou tempo esgotado. A Figura 2 apresenta o diagrama de transição de estados da Unidade de Controle.

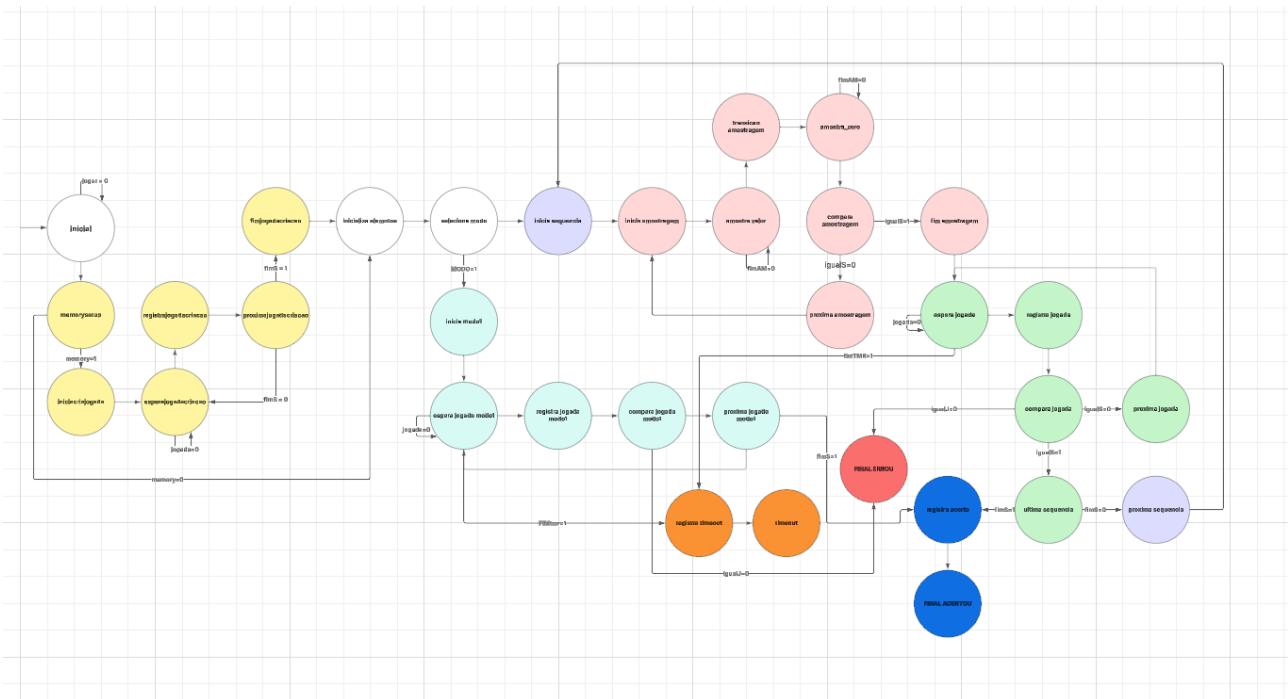


fig 2. Diagrama de estados da Unidade de Controle

O projeto foi estruturado conforme o modelo de Moore, buscando que as saídas dependessem exclusivamente do estado atual. Foram adicionados sinais relacionados ao contador de sequências para melhor gerenciamento da progressão do jogo. A Tabela 1 detalha cada estado da FSM.

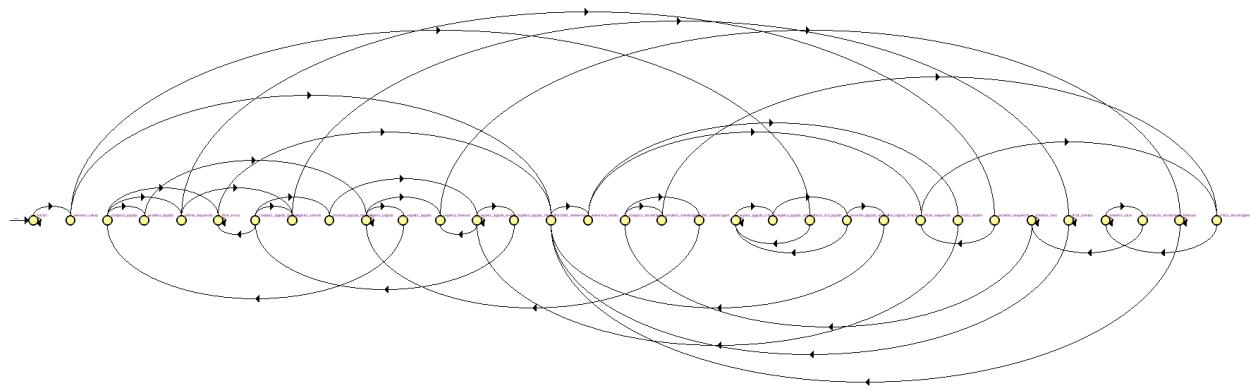
Tabela 1 – Descrição da Unidade de Controle do Sistema

Nome do Estado	Descrição do Estado	Próximo Estado	Condições e Justificativas para a Transição entre Estados
<b>inicial</b>	Estado inicial do jogo	memory_setup	<b>iniciar = 1</b>
<b>memory_setup</b>	decide qual memória deve acessar	inicializa_elementos	<b>memory = 0</b>
<b>memory_setup</b>	decide qual memória deve acessar	inicia_cria_jogadas	<b>memory = 1</b>
<b>inicia_cria_jogadas</b>	inicia a criação das jogadas personalizadas	espera_jogada_criac ao	-
<b>espera_jogada_criac ao</b>	espera o usuário inserir uma jogada	proxima_jogada_criacao	<b>jogada = 1</b>
<b>proxima_jogada_criacao</b>	atualiza para a próxima jogada	espera_jogada_criac ao	<b>fimS = 0</b>
<b>proxima_jogada_criacao</b>	atualiza para a próxima jogada	fim_jogada_criacao	<b>fimS = 1</b>
<b>fim_jogada_criacao</b>	finaliza a fase de criacao	inicializa_elementos	-
<b>inicializa_elementos</b>	Inicializa os elementos do jogo	seleciona modo	-
<b>seleciona modo</b>	escolhe qual jeito vai jogar	inicia sequencia	modo = 0
<b>seleciona modo</b>	escolhe qual jeito vai jogar	inicia modo1	modo = 1
<b>espera jogada modo1</b>	aguarda jogada	registra_timeout / registra jogada modo1	timerTMR = 1 jogada
<b>registra jogada</b>	salva no registrador	compara jogada	
<b>compara jogada</b>	compara os valores	final errou / próxima jogada modo1	igualj=0 igualj=1

Nome do Estado	Descrição do Estado	Próximo Estado	Condições e Justificativas para a Transição entre Estados
<b>proxima_jogada</b>	vai para o proximo	espera_jogada	
<b>inicia_sequencia</b>	Inicia a sequência de jogadas	inicia_amostragem	-
<b>inicia_amostragem</b>	inicia amostragem dos leds	amostra_valor	-
<b>amostra_valor</b>	amostra o valor nos leds	transicao_amostragem	fimAM = 1
<b>transicao_amostragem</b>	reseta a contagem de amostragem	amostra_zero	-
<b>amostra_zero</b>	amostra os leds apagados	compara_amostragem	fimAM=1
<b>compara_amostragem</b>	verifica se já terminou	proxima_amostragem	igualS = 0
<b>compara_amostragem</b>		fim_amostragem	fimS = 1
<b>proxima_amostragem</b>	transiciona para o proximo valor	inicia_amostragem	-
<b>fim_amostragem</b>	finaliza a secção de amostragem	espera_jogada	-
<b>espera_jogada</b>	Aguarda o jogador realizar uma jogada	registra_jogada / registra_timeout	jogada realizada ou timeout
<b>registra_jogada</b>	Registra a jogada do usuário	compara_jogada	-
<b>compara_jogada</b>	Compara a jogada com o valor da memória	proxima_jogada / final_errou / ultima_sequencia	igualS = 1 (jogada correta) ou igualS = 0 (jogada errada)
<b>proxima_jogada</b>	Avança para a	espera_jogada	-

Nome do Estado	Descrição do Estado	Próximo Estado	Condições e Justificativas para a Transição entre Estados
	próxima jogada na sequência		
<b>ultima_sequencia</b>	Verifica se é a última jogada da sequência	proxima_sequencia / registra_acerto	<code>fimS = 1</code> (última jogada) ou <code>fimS = 0</code> (não é a última)
<b>proxima_sequencia</b>	Avança para a próxima sequência de jogadas	espera_jogada	-
<b>final_errou</b>	Jogador cometeu um erro	inicializa_elementos	<code>reiniciar jogo</code>
<b>registra_acerto</b>	aumenta o contador de acertos	final_acertou	
<b>final_acertou</b>	Jogador acertou todas as sequências	inicializa_elementos	<code>reiniciar jogo</code>
<b>registra_timeout</b>	aumenta o contador de timeouts	timeout	
<b>timeout</b>	Tempo esgotado para realizar a jogada	inicializa_elementos	<code>reiniciar jogo</code>

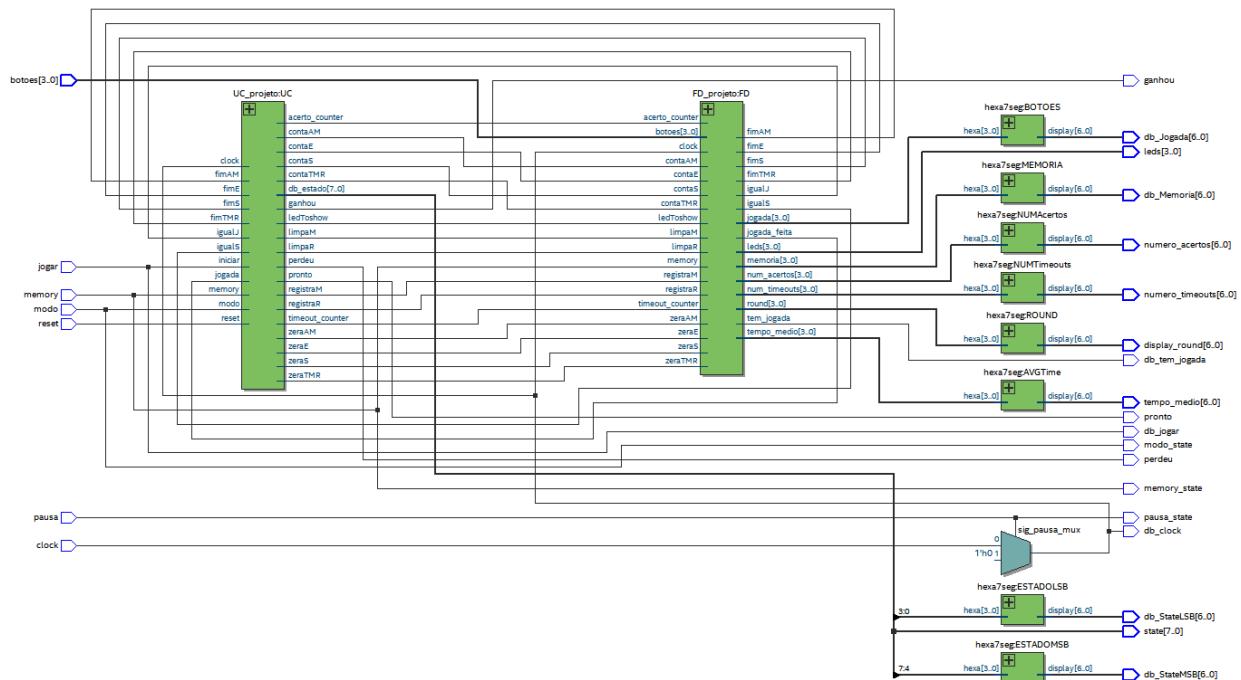
Para fins de validação, é realizado a partir da vista do FSM no Quartus, uma comparação entre o proposto acima e o gerado. Demonstra-se então o correto funcionamento do circuito, conformando-se com o previsto:



**fig 3. FSM Unidade de Controle**

### 3.4 PROJETO DO SISTEMA DIGITAL

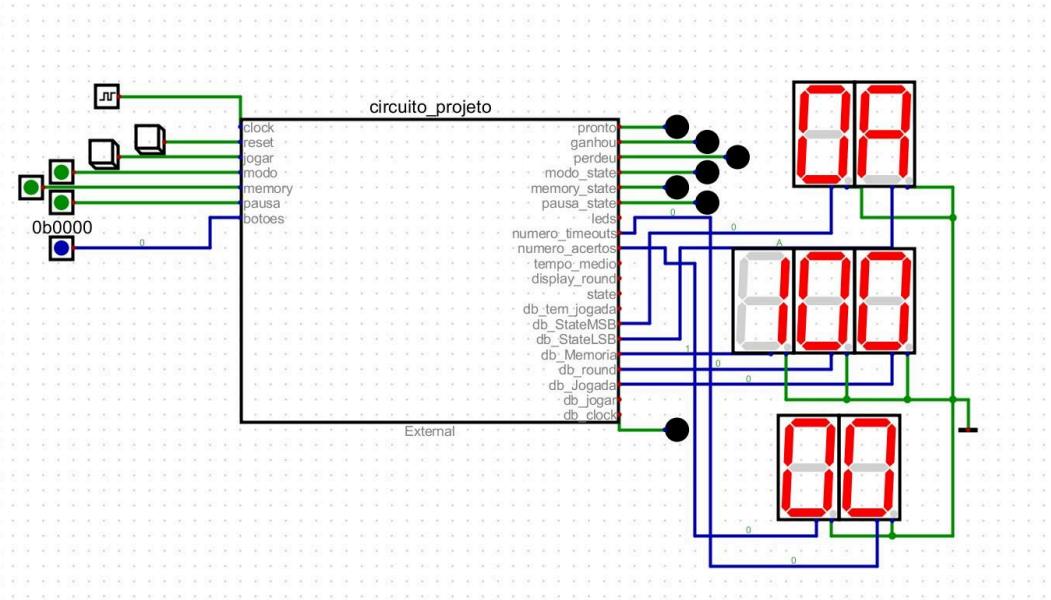
A integração entre o **Fluxo de Dados (FD)** e a **Unidade de Controle (UC)** é realizada por meio de sinais de controle e status, garantindo que o sistema funcione de maneira sincronizada. A **UC** emite sinais de controle que orientam a execução das operações no **FD**, enquanto este retorna sinais de status que indicam o progresso do jogo e permitem à **UC** decidir as próximas ações. A Figura 4 apresenta a visão do sistema no **RTL Viewer** a fim de ilustrar essa integração.



**fig 4. RTL Viewer**

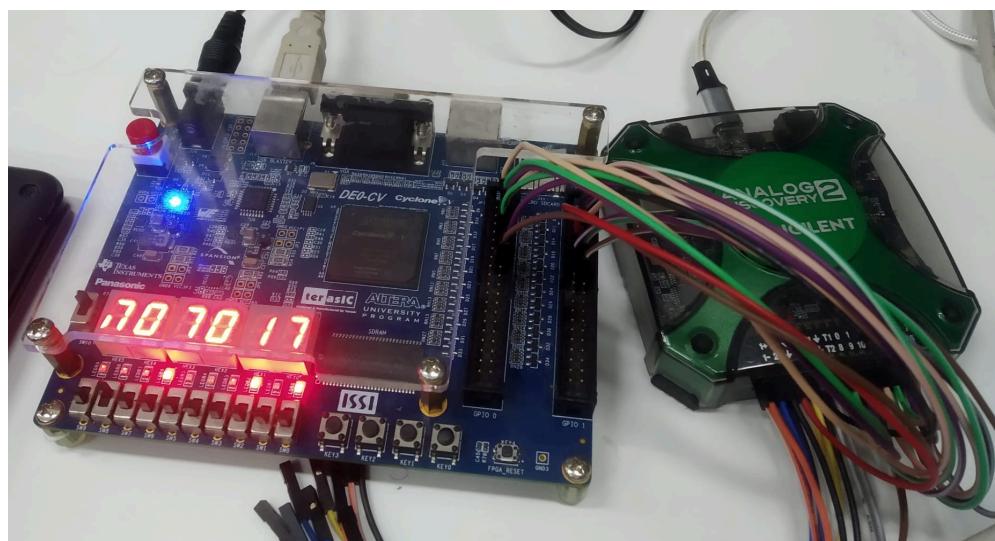
Como na experiência anterior, a interação entre os dois componentes permite que o sistema funcione de maneira sincronizada, pois a unidade de controle gerencia a execução do fluxo de dados em cada ciclo de clock.

Para podermos validar o funcionamento antes mesmo de testar na FPGA, utilizamos o Digital que pode ser verificado logo abaixo:



#### 4 IMPLANTAÇÃO DO PROJETO

Este capítulo tem o objetivo de documentar as atividades práticas de execução do projeto desenvolvido e documentado no capítulo 2.3 no ambiente do Laboratório Digital.



**fig 5. Montagem FPGA & ANALOG DISCOVERY**

### 4.1.1 PINAGEM DA PLACA FPGA

A pinagem foi definida no Intel Quartus Prime utilizando o Pin Planner, associando os sinais do projeto aos pinos físicos da FPGA Cyclone V (placa DE0-CV). A figura abaixo apresenta o plano de pinagem detalhado:

Node Name	Direction	Location	I/O Bank	VREF Group	I/O Standard	Reserved	Current Strength	Slew Rate	Differential Pairs	Analog Setting	(B)VCCT_GX	Driver I/O Pin Termination	Isolated Refclk	Common Mode	Driver Slew Rate	Differential Output	Driver Current
botoes[3]	Input	PIN_M21	5B	B5B_NO	2.5 V		12mA ..auto										
botoes[2]	Input	PIN_K22	5B	B5B_NO	2.5 V		12mA ..auto										
botoes[1]	Input	PIN_K20	7A	B7A_NO	2.5 V		12mA ..auto										
botoes[0]	Input	PIN_C16	7A	B7A_NO	2.5 V		12mA ..auto										
clock	Input	PIN_B16	7A	B7A_NO	2.5 V		12mA ..auto										
db_Jogada[6]	Output	PIN_A22	4A	B4A_NO	2.5 V ...fault		12mA ..auto	1 (default)									
db_Jogada[5]	Output	PIN_Y21	4A	B4A_NO	2.5 V ...fault		12mA ..auto	1 (default)									
db_Jogada[4]	Output	PIN_Y22	4A	B4A_NO	2.5 V ...fault		12mA ..auto	1 (default)									
db_Jogada[3]	Output	PIN_W21	4A	B4A_NO	2.5 V ...fault		12mA ..auto	1 (default)									
db_Jogada[2]	Output	PIN_W22	4A	B4A_NO	2.5 V ...fault		12mA ..auto	1 (default)									
db_Jogada[1]	Output	PIN_V21	4A	B4A_NO	2.5 V ...fault		12mA ..auto	1 (default)									
db_Jogada[0]	Output	PIN_U21	4A	B4A_NO	2.5 V ...fault		12mA ..auto	1 (default)									
db_Memoria[6]	Output	PIN_AB21	4A	B4A_NO	2.5 V ...fault		12mA ..auto	1 (default)									
db_Memoria[5]	Output	PIN_AB22	4A	B4A_NO	2.5 V ...fault		12mA ..auto	1 (default)									
db_Memoria[4]	Output	PIN_V14	4A	B4A_NO	2.5 V ...fault		12mA ..auto	1 (default)									
db_Memoria[3]	Output	PIN_Y14	4A	B4A_NO	2.5 V ...fault		12mA ..auto	1 (default)									
db_Memoria[2]	Output	PIN_AA10	3B	B3B_NO	2.5 V ...fault		12mA ..auto	1 (default)									
db_Memoria[1]	Output	PIN_AB17	4A	B4A_NO	2.5 V ...fault		12mA ..auto	1 (default)									
db_Memoria[0]	Output	PIN_Y19	4A	B4A_NO	2.5 V ...fault		12mA ..auto	1 (default)									
db_StateLSB[6]	Output	PIN_P9	3B	B3B_NO	2.5 V ...fault		12mA ..auto	1 (default)									
db_StateLSB[5]	Output	PIN_Y15	4A	B4A_NO	2.5 V ...fault		12mA ..auto	1 (default)									
db_StateLSB[4]	Output	PIN_U15	4A	B4A_NO	2.5 V ...fault		12mA ..auto	1 (default)									
db_StateLSB[3]	Output	PIN_U16	4A	B4A_NO	2.5 V ...fault		12mA ..auto	1 (default)									
db_StateLSB[2]	Output	PIN_V20	4A	B4A_NO	2.5 V ...fault		12mA ..auto	1 (default)									
db_StateLSB[1]	Output	PIN_Y20	4A	B4A_NO	2.5 V ...fault		12mA ..auto	1 (default)									
db_StateLSB[0]	Output	PIN_U20	4A	B4A_NO	2.5 V ...fault		12mA ..auto	1 (default)									
db_StateMSB[6]	Output	PIN_W19	4A	B4A_NO	2.5 V ...fault		12mA ..auto	1 (default)									
db_StateMSB[5]	Output	PIN_C2	2A	B2A_NO	2.5 V ...fault		12mA ..auto	1 (default)									
db_StateMSB[4]	Output	PIN_C1	2A	B2A_NO	2.5 V ...fault		12mA ..auto	1 (default)									
db_StateMSB[3]	Output	PIN_P14	4A	B4A_NO	2.5 V ...fault		12mA ..auto	1 (default)									
db_StateMSB[2]	Output	PIN_T14	4A	B4A_NO	2.5 V ...fault		12mA ..auto	1 (default)									
db_StateMSB[1]	Output	PIN_M8	3B	B3B_NO	2.5 V ...fault		12mA ..auto	1 (default)									
db_StateMSB[0]	Output	PIN_N9	3B	B3B_NO	2.5 V ...fault		12mA ..auto	1 (default)									
db_clock	Output	PIN_L2	2A	B2A_NO	2.5 V ...fault		12mA ..auto	1 (default)									
db_jogar	Output	PIN_U1	2A	B2A_NO	2.5 V ...fault		12mA ..auto	1 (default)									
db_ten_jogada	Output	PIN_L1	2A	B2A_NO	2.5 V		12mA ..auto	1 (default)									
display_round[6]	Output	PIN_U22	4A	B4A_NO	2.5 V ...fault		12mA ..auto	1 (default)									
display_round[5]	Output	PIN_AA17	4A	B4A_NO	2.5 V ...fault		12mA ..auto	1 (default)									
display_round[4]	Output	PIN_AB18	4A	B4A_NO	2.5 V ...fault		12mA ..auto	1 (default)									

fig 6. Pin planner

### 4.1.2 ESTRATÉGIA DE MONTAGEM

A implementação do projeto na placa **FPGA DE0-CV** seguiu um fluxo estruturado para assegurar a correta configuração e funcionamento do sistema. As etapas adotadas foram:

#### 1. Compilação do projeto no Quartus Prime

- O código Verilog foi desenvolvido e revisado conforme os requisitos da experiência.
- O projeto foi sintetizado e analisado utilizando as ferramentas **RTL Viewer** e **State Machine Viewer**.
- A designação de pinos da FPGA foi configurada no **Pin Planner**, com base nas tabelas fornecidas pela apostila.

#### 2. Transferência do arquivo de configuração para a FPGA

- Após a compilação, o arquivo **.sof** foi gerado e carregado na FPGA através do **Intel Quartus Prime Programmer**.
- A programação da placa **DE0-CV** foi realizada via conexão **USB Blaster**.

#### 3. Verificação e depuração do circuito

- Os sinais de saída foram analisados utilizando os **LEDs** e os **displays de 7 segmentos** da FPGA.
- Saídas adicionais de depuração foram implementadas para monitorar estados internos do sistema buscando facilitar a identificação e correção de possíveis falhas.
- A validação inicial incluiu testes manuais e simulações no **ModelSim** para verificar a resposta do circuito a diferentes cenários do jogo.

Esse processo garante uma implementação eficiente e confiável, e permite ajustes precisos durante os testes e facilitando a análise do funcionamento do sistema.

#### 4.1.3 ESTRATÉGIA DE DEPURAÇÃO

A depuração do projeto ocorre principalmente por meio da análise dos sinais exibidos nos **LEDs** e nos **displays de 7 segmentos** da FPGA, eliminando a necessidade de 25 de equipamentos adicionais. Para facilitar o diagnóstico, foram implementadas diversas saídas de depuração que monitoram o estado interno do circuito digital. Sempre que falhas são identificadas, ajustes devem ser aplicados no código **Verilog**, seguidos de recompilação no **Quartus Prime**. Além disso, simulações no **ModelSim** validam o comportamento lógico antes da implementação física. Outra etapa fundamental é a **verificação das conexões físicas da FPGA**, nota-se que, diversas vezes, há desatenção no momento de associar os pinos no **Pin Planner**. Desse modo, é necessário prevenir problemas de hardware, como ligações incorretas ou mau contato. Após cada modificação, testes específicos asseguram que as correções não comprometem funcionalidades previamente validadas. Esse processo iterativo preserva a integridade do sistema e permite que o circuito opere conforme o esperado na implementação final.

### 5 PERIFÉRICOS

Os componentes adicionais do sistema passaram por melhorias significativas nesta semana, o que viabilizou a execução dos primeiros testes, tanto individuais quanto de integração com a placa FPGA. Esses avanços são fundamentais para garantir o correto funcionamento do sistema e a comunicação eficiente entre os diferentes módulos.

#### Painel de controle

O desenvolvimento do painel de comunicação entre o sistema e o ambiente externo foi iniciado, com foco especial na configuração do display de LED 16x2. Esse dispositivo exibe mensagens de instrução ao usuário com base no estado (8 bits) fornecido pela Unidade de Controle. Abaixo, alguns exemplos de mensagens e seus respectivos estados:

- **Início do jogo (Estado 0):** *Pulo do sapo Bem-vindo!*
- **Aguardando jogada (Estado 10):** *Aguardando próxima jogada*
- **Fim - Perdeu (Estado 16):** *Essa não! Você errou!*
- **Fim - Ganhou (Estado 17):** *Parabéns! Você venceu!*
- **Timeout (Estado 18):** *Essa não! O tempo acabou!*

Para a configuração do display, foi utilizado um **Arduino UNO**, que recebe os 8 bits do estado como entrada, realiza a conversão dos valores e envia a frase correspondente ao dispositivo de cristal líquido. Além disso, um módulo **I2C** foi integrado para simplificar a comunicação com o display.

Vale destacar que cada mensagem pode conter, no máximo, 32 caracteres, devido às limitações físicas do dispositivo de saída. O código utilizado para configuração do mesmo é apresentado abaixo:

```
#include <Wire.h>
#include <LiquidCrystal_I2C.h>
```

```

// Define o endereço I2C do display (0x27 é o endereço mais comum, mas
// pode ser 0x3F em alguns modelos)
LiquidCrystal_I2C lcd(0x27, 16, 2);
// SDA = A4
// SLC = A5

// Define os pinos para as 4 entradas digitais
const int pinoBit0 = 2; // LSB
const int pinoBit1 = 3;
const int pinoBit2 = 4;
const int pinoBit3 = 5;
const int pinoBit4 = 6;
const int pinoBit5 = 7;
const int pinoBit6 = 8;
const int pinoBit7 = 9; // MSB

int estadoAtual = -1; // Estado atual da máquina
int estadoAnterior = -1; // Estado anterior para detectar mudanças

void setup() {
    // Inicializa a comunicação serial para debugging
    Serial.begin(9600);
    Serial.println("Iniciando sistema...");
    // Inicializa o display LCD
    lcd.init();
    lcd.backlight();
    // Configura os pinos de entrada para lógica de 3.3V (não necessita
    // de pull-up)
    pinMode(pinoBit0, INPUT);
    pinMode(pinoBit1, INPUT);
    pinMode(pinoBit2, INPUT);
    pinMode(pinoBit3, INPUT);
    pinMode(pinoBit4, INPUT);
    pinMode(pinoBit5, INPUT);
    pinMode(pinoBit6, INPUT);
    // Exibe mensagem inicial
    lcd.clear();
    lcd.setCursor(0, 0);
    lcd.print("Sistema");
    lcd.setCursor(0, 1);
    lcd.print("Inicializando...");
    delay(2000); // Aguarda 2 segundos para inicialização
}

void loop() {
    // Lê o estado dos pinos com lógica direta (3.3V = HIGH = 1)
    int bit0 = digitalRead(pinoBit0);
    int bit1 = digitalRead(pinoBit1);
    int bit2 = digitalRead(pinoBit2);
    int bit3 = digitalRead(pinoBit3);
    int bit4 = digitalRead(pinoBit4);
    int bit5 = digitalRead(pinoBit5);
    int bit6 = digitalRead(pinoBit6);
    // Calcula o número do estado atual (0-127) com 7 bits
}

```

```

    int novoEstado = bit0 + (bit1 << 1) + (bit2 << 2) + (bit3 << 3) +
(bit4 << 4) + (bit5 << 5) + (bit6 << 6);
Serial.println(novoEstado);
// Se detectamos uma mudança, atualiza o estado e o display
if (novoEstado != estadoAnterior) {
    estadoAtual = novoEstado;
    Serial.print("Novo estado detectado: ");
    Serial.println(estadoAtual);

    // Atualiza o display
    exibirEstado(estadoAtual);
    estadoAnterior = estadoAtual;
}
// Delay curto para estabilidade sem prejudicar a responsividade
delay(100);
}

void exibirEstado(int estado) {
lcd.clear();
switch (estado) {
case 0: // Estado 00000000
    lcd.setCursor(0, 0);
    lcd.print("Pulo do Sapo");
    lcd.setCursor(0, 1);
    lcd.print("Bem vindo!");
    break;

case 10: // Estado 0001010
    lcd.setCursor(0, 0);
    lcd.print("Aguardando");
    lcd.setCursor(0, 1);
    lcd.print("proxima jogada");
    break;

case 16: // Estado 0010000
    lcd.setCursor(0, 0);
    lcd.print("Essa nao!");
    lcd.setCursor(0, 1);
    lcd.print("Voce errou!");
    break;

case 17: // Estado 0010001
    lcd.setCursor(0, 0);
    lcd.print("Parabens!");
    lcd.setCursor(0, 1);
    lcd.print("Voce venceu!");
    break;

case 18: // Estado 0010010
    lcd.setCursor(0, 0);
    lcd.print("Essa nao!");
    lcd.setCursor(0, 1);
    lcd.print("O tempo acabou!");
    break;
}
}

```

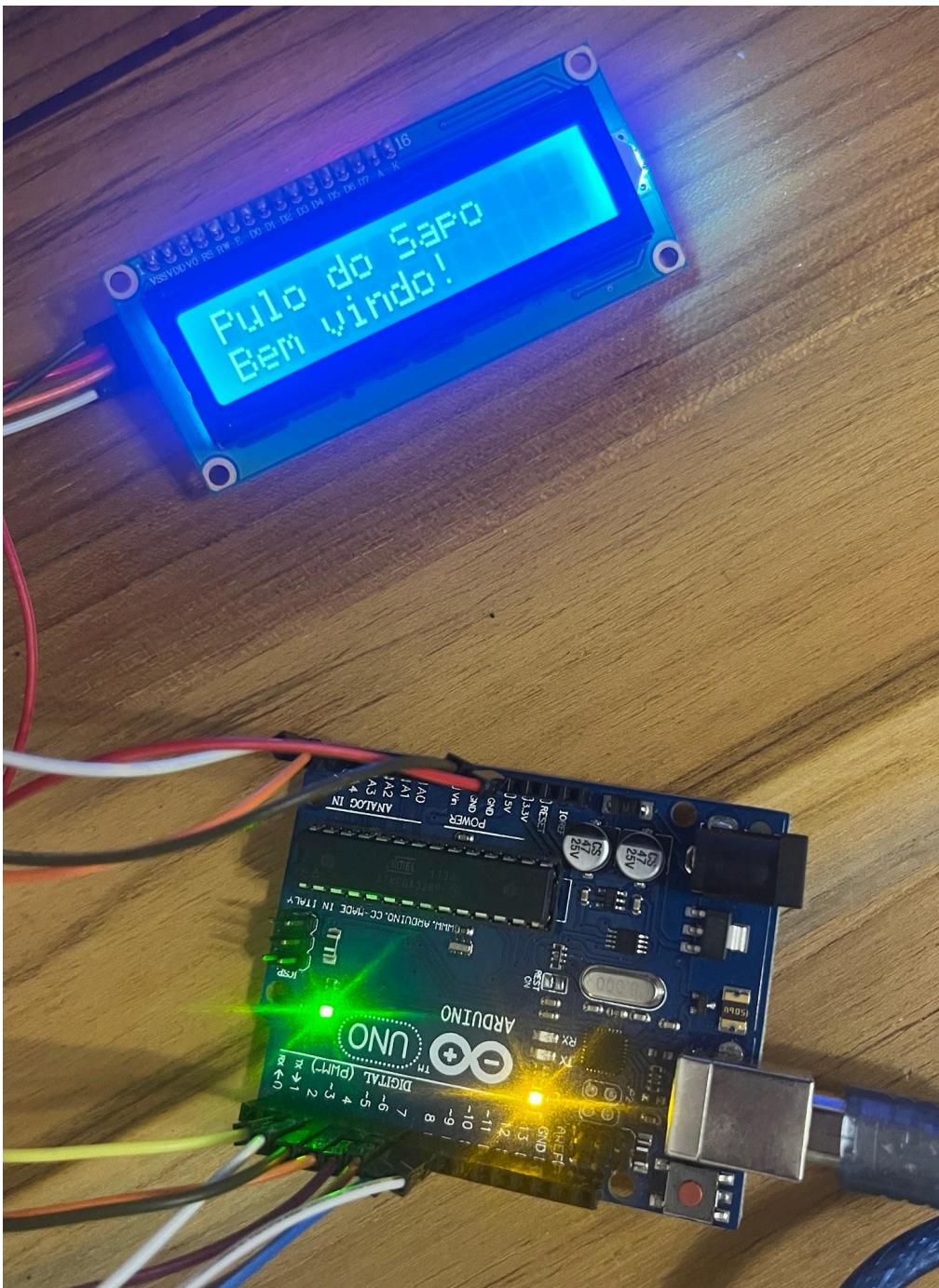
```
case 19: // Estado 0010011
    lcd.setCursor(0, 0);
    lcd.print("Seleccione a");
    lcd.setCursor(0, 1);
    lcd.print("memoria");
    break;

case 21: // Estado 0010101
    lcd.setCursor(0, 0);
    lcd.print("Insira o proximo");
    lcd.setCursor(0, 1);
    lcd.print("movimento");
    break;

case 25: // Estado 0011001
    lcd.setCursor(0, 0);
    lcd.print("Seleccione o modo");
    lcd.setCursor(0, 1);
    lcd.print("de jogo");
    break;

case 27: // Estado 0011011
    lcd.setCursor(0, 0);
    lcd.print("Aguardando");
    lcd.setCursor(0, 1);
    lcd.print("proxima jogada");
    break;

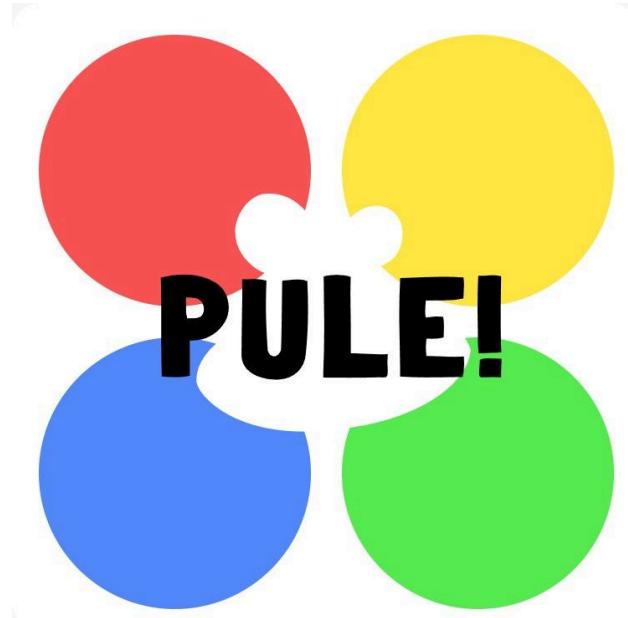
default: // Para todos os outros estados
    lcd.setCursor(0, 0);
    lcd.print("Estado ");
    lcd.print(estado);
    break;
}
```



**fig 7. Arduino e display no estado 0**

## Tapete

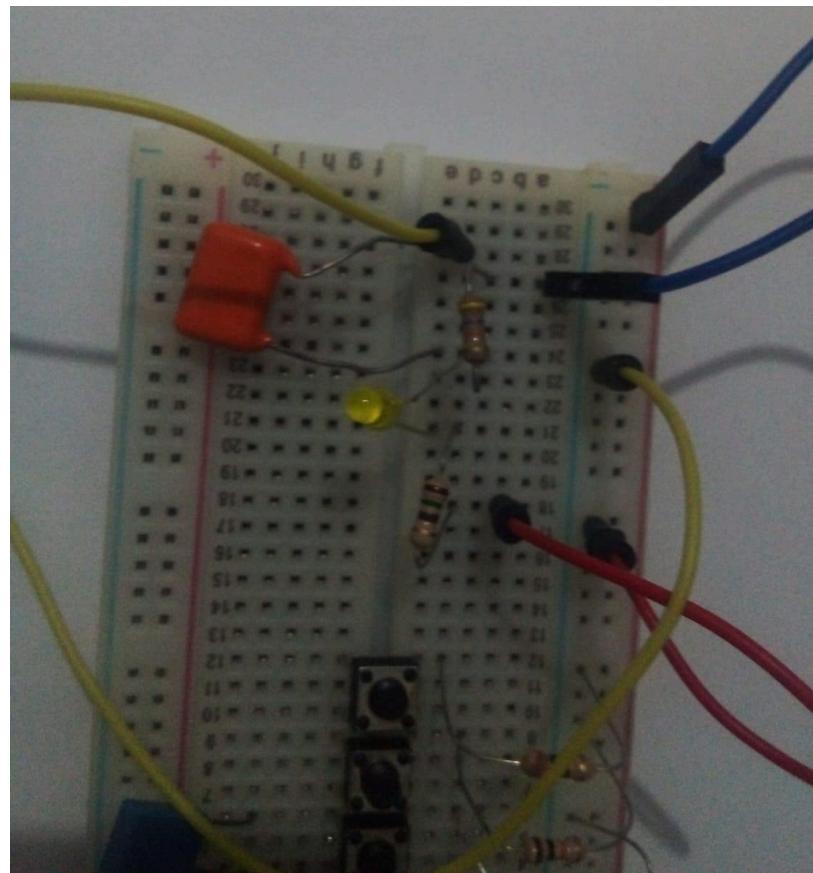
Após a validação do uso de placas como botões, foram definidas a disposição e as dimensões do tapete interativo (90cm x 90cm). Como o sistema depende da precisão na detecção dos toques dos usuários, foi necessário um estudo detalhado sobre a estabilidade dos sinais elétricos gerados pelas placas ao serem pressionadas.



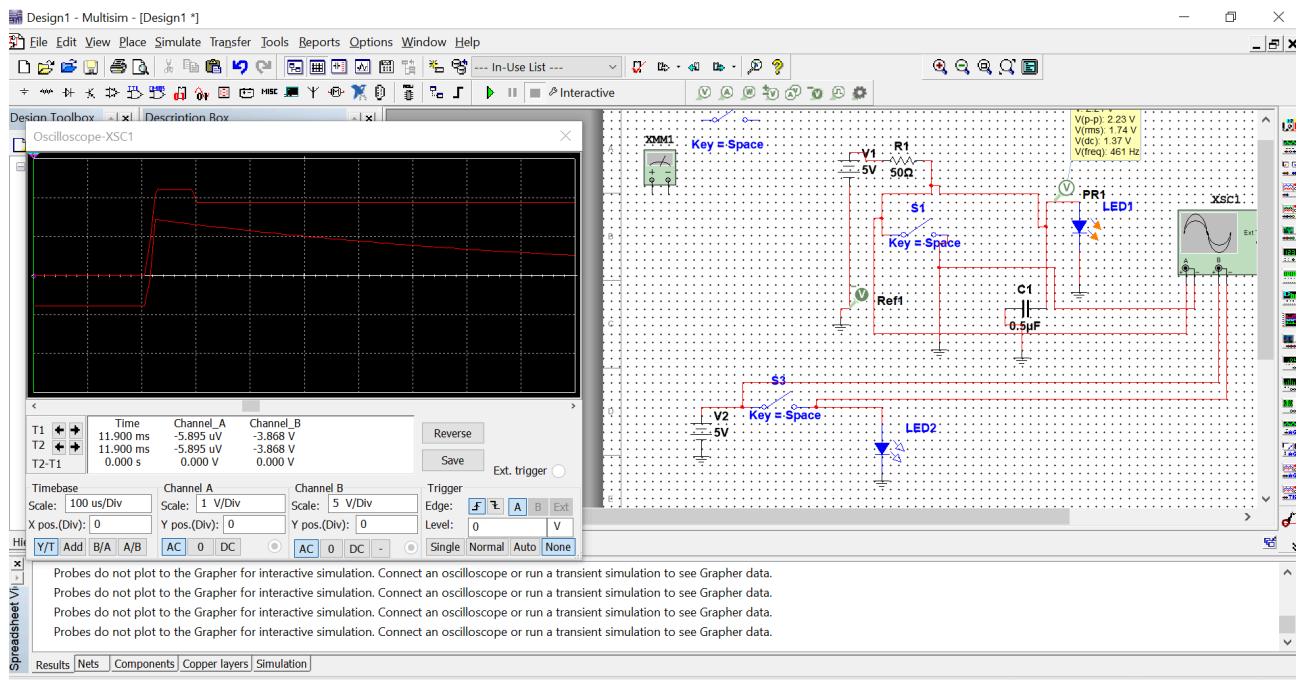
**fig 8. Design do tapete interativo**

Identificamos a necessidade de um circuito de debouncing para evitar leituras incorretas. Esse circuito é essencial porque, ao pressionar um botão físico (ou uma placa sensível no caso do tapete), os contatos mecânicos internos podem oscilar antes de se estabilizarem, gerando múltiplas transições elétricas em um curto período de tempo. Esse efeito, conhecido como bouncing, pode resultar em leituras erradas pelo sistema, registrando toques duplicados ou falhando na detecção correta dos comandos do usuário, assim levando-nos a pesquisa e primeiras tentativas de implementação.

Inicialmente, testamos um circuito com um resistor de **47kΩ** e um capacitor de **100μF**. Em seguida, utilizamos o software **Multisim** para analisar sua validade e realizamos as devidas modificações conforme necessário.



**fig 9. Tentativa de montagem de circuito debouncing em lab**



**fig 7. Modelagem do circuito deboucing no Multisim**

## 6 CONCLUSÃO SEMANA 2

Conseguimos nos manter no cronograma para essa semana, o circuito foi testado na FPGA e agora podemos focar única e exclusivamente na integração dos componentes externos.

### **6.3 SEMANA 3**

**HORAS TRABALHADAS: 6H**

#### **1. DESCRIÇÃO DO MÓDULO DA SEMANA**

Para essa terceira semana, focamos em validar o funcionamento geral do projeto e definir se precisamos ou não mudar a abordagem de algum componente.

**Módulos:**

3. Montagem de todos os módulos de maneira integrada, além de seus respectivos testes
7. Painel do usuário - Fazer a montagem e validação da interface em uma protoboard e elaborar o layout na caixa
8. Tapete de controle - Validar junto ao restante do circuito e começar a confeccionar a montagem final.

#### **2. DETALHAMENTO DO PROJETO LÓGICO**

##### **2.1 PROJETO DO FLUXO DE DADOS**

O fluxo de dados (FD), durante essa terceira semana, não sofreu qualquer alteração, visto que seu funcionamento já estava bem estabelecido desde a semana passada.

##### **2.2 PROJETO DA UNIDADE DE CONTROLE**

A Unidade de Controle (UC), durante essa terceira semana, não sofreu qualquer alteração, visto que seu funcionamento já estava bem estabelecido desde a semana passada.

##### **2.3 PROJETO DO SISTEMA DIGITAL**

Durante essa semana, o código responsável por integrar os componentes gerais do circuito, foi refatorado para uma versão final e pronta para se usar, essas alterações podem ser consultadas logo abaixo:

```
module circuito_projeto (
    //ENTRADAS
    input clock,
    input reset,
    input jogar,
    input modo,
    input memory,
    input pausa,
    input [3:0] botoes,
```

```
//SAIDAS
    output pronto,
    output ganhou,
    output perdeu,
    output modo_state,
    output memory_state,
    output pausa_state,
    output [6:0] numero_timeouts,
    output [6:0] numero_acertos,
    output [13:0] tempo_medio,
    output [3:0] leds,
    output [6:0] display_round,
    output [7:0] state
);

//Pausa Universal
wire sig_pausa_mux;
assign sig_pausa_mux = pausa ? 0 : clock;

//Sinais Intermediario Display
//ENTRADA
wire [3:0] sig_num_timeouts;
wire [3:0] sig_num_acertos;
wire [3:0] sig_round;
wire [3:0] sig_jogada;
//SAIDA
wire [7:0] sig_estadoTotal;

//Sinais Intermediario UC & FD
//FD - UC
//CONTADORES
wire sig_fimE;
wire sig_fimS;
wire sig_fimTMR;
wire sig_fimAM;
//COMPARADORES
wire sig_igualJ;
wire sig_iguals;
//DETECTOR
wire sig_jogada_feita;

//UC - FD
//CONTADORES
wire sig_contaE;
```

```

    wire sig_contaS;
    wire sig_contaTMR;
    wire sig_contaAM;
    wire sig_zeraE;
    wire sig_zeraS;
    wire sig_zeraTMR;
    wire sig_zeraAM;
    //REGISTRADOR & RAM
    wire sig_limpaM;
    wire sig_limpaR;
    wire sig_registraM;
    wire sig_registraR;
    //OUTROS
    wire sig_ledToShow;
    wire sig_acerto_counter;
    wire sig_timeout_counter;

    // Fluxo de Dados
FD_projeto FD (
    .clock          ( sig_pausa_mux ),
    .memory         ( memory ),
    .botoes         ( botoes ),
    //CONTADORES
    .contaE          ( sig_contaE ),
    .contaS          ( sig_contaS ),
    .contaTMR        ( sig_contaTMR ),
    .contaAM         ( sig_contaAM ),
    .zeraE           ( sig_zeraE ),
    .zeraS           ( sig_zeraS ),
    .zeraAM          ( sig_zeraAM ),
    .zeraTMR         ( sig_zeraTMR ),
    .fimE            ( sig_fimE ),
    .fimS            ( sig_fimS ),
    .fimTMR          ( sig_fimTMR ),
    .fimAM           ( sig_fimAM ),
    //COMPARADORES
    .igualJ          ( sig_igualJ ),
    .iguals          ( sig_iguals ),
    //REGISTRADOR & RAM
    .limpaM          ( sig_limpaM ),
    .limpaR          ( sig_limpaR ),
    .registraM       ( sig_registraM ),
    .registraR       ( sig_registraR ),
    //DISPLAY

```

```

.round              ( sig_round ),
.jogada            ( sig_jogada ),
.memoria          ( sig_memoria ),
.num_acertos      ( sig_num_acertos ),
.num_timeouts     ( sig_num_timeouts ),
.tempo_medio       ( tempo_medio ),
//OUTROS
.ledToshow        ( sig_ledToshow ),
.acerto_counter   ( sig_acerto_counter ),
.timeout_counter  ( sig_timeout_counter ),
.tem_jogada        ( db_tem_jogada ),
.jogada_feita     ( sig_jogada_feita ),
.leds              ( leds )

);

// Unidade de Controle
UC_projeto UC (
    .clock           ( sig_pausa_mux ),
    .iniciar         ( jogar ),
    .reset           ( reset ),
    .modo            ( modo ),
    .memory          ( memory ),
//CONTADORES
    .contaE          ( sig_contaE ),
    .contaS          ( sig_contaS ),
    .contaTMR         ( sig_contaTMR ),
    .contaAM          ( sig_contaAM ),
    .acerto_counter  ( sig_acerto_counter ),
    .timeout_counter ( sig_timeout_counter ),
    .zeraE           ( sig_zeraE ),
    .zeraS           ( sig_zeraS ),
    .zeraTMR          ( sig_zeraTMR ),
    .zeraAM          ( sig_zeraAM ),
    .fimE            ( sig_fimE ),
    .fimS            ( sig_fimS ),
    .fimTMR          ( sig_fimTMR ),
    .fimAM          ( sig_fimAM ),
//COMPARADORES
    .igualJ          ( sig_igualJ ),
    .igualS          ( sig_igualS ),
//REGISTRADOR & RAM
    .limpaM          ( sig_limpaM ),
    .limpaR          ( sig_limpaR ),
    .registraM       ( sig_registraM ),

```

```

.registraR          ( sig_RegistraR ),
//DISPLAY
.db_estado          ( sig_EstadoTotal ),
//OUTROS
.ledToshow         ( sig_LedToshow ),
.jogada             ( sig_Jogada_Feita ),
.perdeu              ( Perdeu ),
.ganhou              ( Ganhou ),
.pronto              ( Pronto )

);

// Display 7 segmentos ROUND
hexa7seg ROUND (
    .hexa      ( sig_Round ),
    .display   ( display_Round )
);

//METRICAS
// Display 7 segmentos Numero Timeouts
hexa7seg NUMTimeouts (
    .hexa      ( sig_Num_Timeouts ),
    .display   ( numero_Timeouts )
);

// Display 7 segmentos Numero de Acertos
hexa7seg NUMAcertos (
    .hexa      ( sig_Num_Acertos ),
    .display   ( numero_Acertos )
);

//Sinais Gerais
assign modo_state     = modo;
assign memory_state   = memory;
assign pausa_state    = pausa;
assign state           = sig_EstadoTotal;

endmodule

```

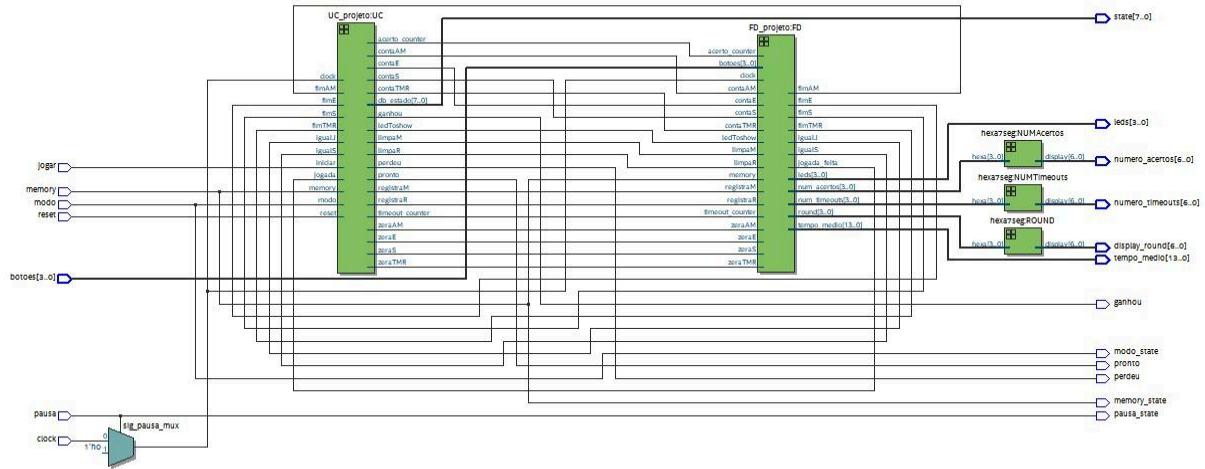


fig 1. RTL Viewer

Como na experiência anterior, a interação entre os dois componentes permite que o sistema funcione de maneira sincronizada, pois a unidade de controle gerencia a execução do fluxo de dados em cada ciclo de clock.

Para podermos validar o funcionamento antes mesmo de testar na FPGA, utilizamos o Digital que pode ser verificado logo abaixo:

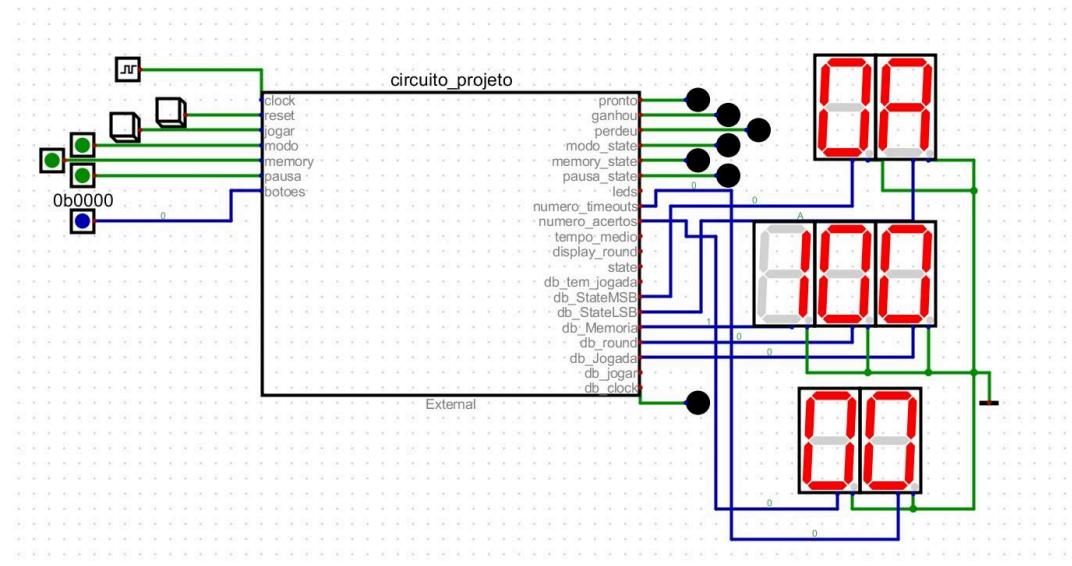


fig 2. Simulação no Digital

### 3. IMPLANTAÇÃO DO PROJETO

Este capítulo tem o objetivo de documentar as atividades práticas de execução do projeto desenvolvido e documentado no capítulo 2.3 no ambiente do Laboratório Digital.

fig 3. Montagem FPGA &amp; Periféricos

### 3.1 PINAGEM DA PLACA FPGA

A pinagem foi definida no Intel Quartus Prime utilizando o Pin Planner, associando os sinais do projeto aos pinos físicos da FPGA Cyclone V (placa DE0-CV). A figura abaixo apresenta o plano de pinagem detalhado:

Node Name	Direction	Location	I/O Bank	VREF Group	I/O Standard	Reserved	Current Strength	Slew Rate	Differential Pairs	Analog Setting	(B)VCCT_GXF	Termination	Refclk	Common Mode	Output Slew Rate	Differential Output	Output Configuration
botoes[3]	Input	PIN_M21	5B	B5B_NO	2.5 V		12mA ..auto										
botoes[2]	Input	PIN_K22	5B	B5B_NO	2.5 V		12mA ..auto										
botoes[1]	Input	PIN_K20	7A	B7A_NO	2.5 V		12mA ..auto										
botoes[0]	Input	PIN_C16	7A	B7A_NO	2.5 V		12mA ..auto										
clock	Input	PIN_B16	7A	B7A_NO	2.5 V		12mA ..auto										
db_Jogada[6]	Output	PIN_A22	4A	B4A_NO	2.5 V ...fault		12mA ..auto	1 (default)									
db_Jogada[5]	Output	PIN_Y21	4A	B4A_NO	2.5 V ...fault		12mA ..auto	1 (default)									
db_Jogada[4]	Output	PIN_Y22	4A	B4A_NO	2.5 V ...fault		12mA ..auto	1 (default)									
db_Jogada[3]	Output	PIN_W21	4A	B4A_NO	2.5 V ...fault		12mA ..auto	1 (default)									
db_Jogada[2]	Output	PIN_W22	4A	B4A_NO	2.5 V ...fault		12mA ..auto	1 (default)									
db_Jogada[1]	Output	PIN_V21	4A	B4A_NO	2.5 V ...fault		12mA ..auto	1 (default)									
db_Jogada[0]	Output	PIN_U21	4A	B4A_NO	2.5 V ...fault		12mA ..auto	1 (default)									
db_Memoria[6]	Output	PIN_A621	4A	B4A_NO	2.5 V ...fault		12mA ..auto	1 (default)									
db_Memoria[5]	Output	PIN_A622	4A	B4A_NO	2.5 V ...fault		12mA ..auto	1 (default)									
db_Memoria[4]	Output	PIN_V14	4A	B4A_NO	2.5 V ...fault		12mA ..auto	1 (default)									
db_Memoria[3]	Output	PIN_V14	4A	B4A_NO	2.5 V ...fault		12mA ..auto	1 (default)									
db_Memoria[2]	Output	PIN_AA10	3B	B3B_NO	2.5 V ...fault		12mA ..auto	1 (default)									
db_Memoria[1]	Output	PIN_AB17	4A	B4A_NO	2.5 V ...fault		12mA ..auto	1 (default)									
db_Memoria[0]	Output	PIN_Y19	4A	B4A_NO	2.5 V ...fault		12mA ..auto	1 (default)									
db_Status[6]	Output	PIN_P9	3B	B3B_NO	2.5 V ...fault		12mA ..auto	1 (default)									
db_Status[5]	Output	PIN_Y15	4A	B4A_NO	2.5 V ...fault		12mA ..auto	1 (default)									
db_Status[4]	Output	PIN_U15	4A	B4A_NO	2.5 V ...fault		12mA ..auto	1 (default)									
db_Status[3]	Output	PIN_U16	4A	B4A_NO	2.5 V ...fault		12mA ..auto	1 (default)									
db_Status[2]	Output	PIN_V20	4A	B4A_NO	2.5 V ...fault		12mA ..auto	1 (default)									
db_Status[1]	Output	PIN_Y20	4A	B4A_NO	2.5 V ...fault		12mA ..auto	1 (default)									
db_Status[0]	Output	PIN_U20	4A	B4A_NO	2.5 V ...fault		12mA ..auto	1 (default)									
db_StatusMS[6]	Output	PIN_W19	4A	B4A_NO	2.5 V ...fault		12mA ..auto	1 (default)									
db_StatusMS[5]	Output	PIN_C2	2A	B2A_NO	2.5 V ...fault		12mA ..auto	1 (default)									
db_StatusMS[4]	Output	PIN_C1	2A	B2A_NO	2.5 V ...fault		12mA ..auto	1 (default)									
db_StatusMS[3]	Output	PIN_P14	4A	B4A_NO	2.5 V ...fault		12mA ..auto	1 (default)									
db_StatusMS[2]	Output	PIN_LT14	4A	B4A_NO	2.5 V ...fault		12mA ..auto	1 (default)									
db_StatusMS[1]	Output	PIN_M8	3B	B3B_NO	2.5 V ...fault		12mA ..auto	1 (default)									
db_StatusMS[0]	Output	PIN_N9	3B	B3B_NO	2.5 V ...fault		12mA ..auto	1 (default)									
db_clock	Output	PIN_L2	2A	B2A_NO	2.5 V ...fault		12mA ..auto	1 (default)									
db_jogar	Output	PIN_U1	2A	B2A_NO	2.5 V ...fault		12mA ..auto	1 (default)									
db_tensao_jogada	Output	PIN_L1	2A	B2A_NO	2.5 V		12mA ..auto	1 (default)									
display_round[6]	Output	PIN_U22	4A	B4A_NO	2.5 V ...fault		12mA ..auto	1 (default)									
display_round[5]	Output	PIN_AA17	4A	B4A_NO	2.5 V ...fault		12mA ..auto	1 (default)									
display_round[4]	Output	PIN_AB18	4A	B4A_NO	2.5 V ...fault		12mA ..auto	1 (default)									

fig 4. Pin planner

### 3.2 ESTRATÉGIA DE MONTAGEM

A implementação do projeto na **placa FPGA DE0-CV** seguiu um fluxo estruturado para assegurar a correta configuração e funcionamento do sistema. As etapas adotadas foram:

#### 1. Compilação do projeto no Quartus Prime

- O código Verilog foi desenvolvido e revisado conforme os requisitos da experiência.
- O projeto foi sintetizado e analisado utilizando as ferramentas **RTL Viewer** e **State Machine Viewer**.
- A designação de pinos da FPGA foi configurada no **Pin Planner**, com base nas tabelas fornecidas pela apostila.

#### 2. Transferência do arquivo de configuração para a FPGA

- Após a compilação, o arquivo **.sof** foi gerado e carregado na FPGA através do **Intel Quartus Prime Programmer**.
- A programação da placa **DE0-CV** foi realizada via conexão **USB Blaster**.

#### 3. Verificação e depuração do circuito

- Os sinais de saída foram analisados utilizando os **LEDs** e os **displays de 7 segmentos** da FPGA.

- Saídas adicionais de depuração foram implementadas para monitorar estados internos do sistema buscando facilitar a identificação e correção de possíveis falhas.
- A validação inicial incluiu testes manuais e simulações no **ModelSim** para verificar a resposta do circuito a diferentes cenários do jogo.

Esse processo garante uma implementação eficiente e confiável, e permite ajustes precisos durante os testes e facilitando a análise do funcionamento do sistema.

### 3.3 ESTRATÉGIA DE DEPURAÇÃO

A depuração do projeto ocorre principalmente por meio da análise dos sinais exibidos nos **LEDs** e nos **displays de 7 segmentos** da FPGA, eliminando a necessidade de 25 de equipamentos adicionais. Para facilitar o diagnóstico, foram implementadas diversas saídas de depuração que monitoram o estado interno do circuito digital. Sempre que falhas são identificadas, ajustes devem ser aplicados no código **Verilog**, seguidos de recompilação no **Quartus Prime**. Além disso, simulações no **ModelSim** validam o comportamento lógico antes da implementação física. Outra etapa fundamental é a **verificação das conexões físicas da FPGA**, nota-se que, diversas vezes, há desatenção no momento de associar os pinos no **Pin Planner**. Desse modo, é necessário prevenir problemas de hardware, como ligações incorretas ou mau contato. Após cada modificação, testes específicos asseguram que as correções não comprometem funcionalidades previamente validadas. Esse processo iterativo preserva a integridade do sistema e permite que o circuito opere conforme o esperado na implementação final.

## 4 PERIFÉRICOS

Esta semana teve como objetivo a verificação geral do sistema e sua integração com os periféricos. Durante as aulas de laboratório, priorizamos a execução de testes de comunicação entre os dispositivos de controle, os módulos de visualização externa e a placa FPGA, garantindo o correto funcionamento do conjunto.

### Painel de controle

O painel de controle foi dividido em dois módulos principais:

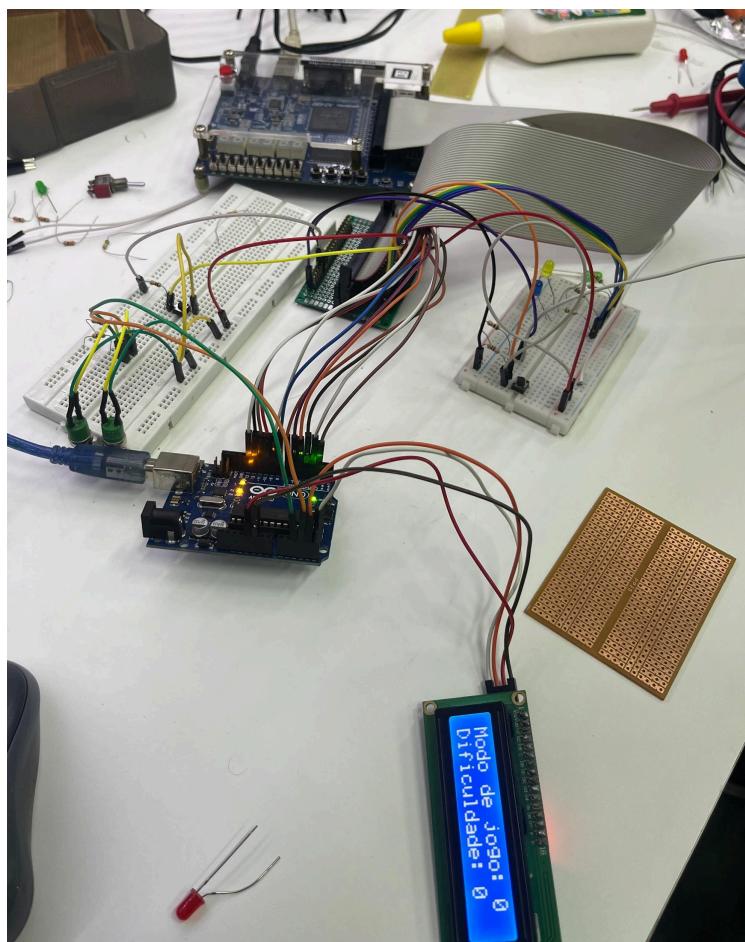
1. **Tela de Jogo** (destinada à criança)
  - Exibe as jogadas e contém o botão de início.
  - Para melhorar a identidade visual, estão sendo desenvolvidos elementos gráficos como logo, mascote e textos personalizados.
  - **Componentes necessários:**
    - 4 LEDs (com resistores adequados)
    - 1 botão de fliperama (alta durabilidade para suportar uso intensivo)
2. **Tela de Configuração** (destinada ao cuidador/responsável)
  - Utiliza um display LCD para apresentar comandos e configurações.
  - Possui botão de reset e chaves para seleção de dificuldade e modo de jogo.
  - **Componentes necessários:**
    - 2 chaves simples
    - 1 botão de reset
    - 1 display LCD 16x2 (com módulo adaptador I2C para simplificar a comunicação)
    - 1 Arduino Uno (responsável pela interface com o LCD e alimentação)

É necessário prestar atenção a alguns detalhes do sistema, devido à fragilidade de alguns componentes. Para isso, deve-se garantir a robustez do botão de fliperama para evitar falhas por uso excessivo, bem como otimizar a comunicação I2C entre o Arduino e o display para evitar latência.

### Tapete

O tapete é um dos componentes mais críticos do projeto, pois deve ser resistente e responsivo. Desenvolvemos um sistema de ativação por pressão baseado em três componentes metálicos:

- **Estrutura do Tapete:**
  - Duas placas inferiores em formato de "S" que entram em curto quando pressionadas por uma placa superior.
  - Material reforçado com papelão, papel panamá e folha plástica para resistência a impactos e pisoteamento.
- **Circuitos e Melhorias:**
  - Fios trançados para reduzir interferência eletromagnética.
  - Implementação de um circuito de *debounce* para filtrar ruídos mecânicos e garantir sinais limpos.



**fig 3. Teste de integração com a FPGA**

## 5 CONCLUSÃO SEMANA 3

Ao longo desta semana, consolidamos a integração dos módulos do projeto, validando o funcionamento do sistema digital na FPGA e a comunicação com os periféricos, como o painel de controle e o tapete sensível. A refatoração do código Verilog garantiu uma estrutura robusta para a unidade de controle e o fluxo de dados, enquanto os testes no Digital e no hardware confirmaram a sincronia entre os componentes. A atenção aos detalhes de montagem — como a robustez do tapete, a filtragem de ruídos e a otimização da interface com o display — foi crucial para assegurar usabilidade e durabilidade. Com a pinagem definida e a estratégia de depuração implementada, o projeto está pronto para a fase final de ajustes e validações, mantendo o foco na confiabilidade e na experiência do usuário.

## 6.4 SEMANA 4

**HORAS TRABALHADAS: 17H**

### 2. DESCRIÇÃO DO MÓDULO DA SEMANA

Para essa quarta semana, seguimos validando o funcionamento geral do projeto e definimos que não é mais necessário alterar nenhum componente.

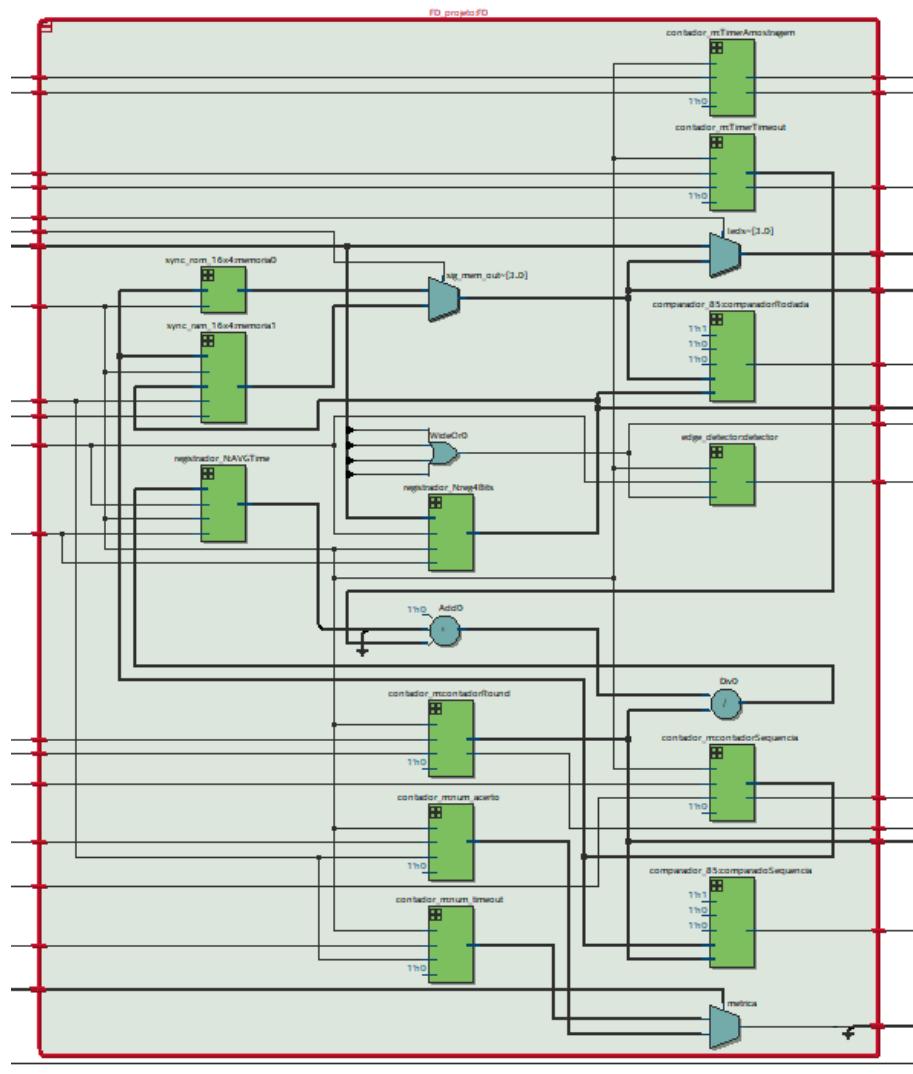
#### Módulos:

1. Montagem de todos os módulos de maneira integrada, além de seus respectivos testes
2. Painel do usuário - Fazer a soldagem do sistema e validação da interface em uma protoboard e elaborar o layout na caixa
3. Tapete de controle - Terminar a montagem final.

### 2. DETALHAMENTO DO PROJETO LÓGICO

#### 2.1 PROJETO DO FLUXO DE DADOS

O fluxo de dados (FD), durante essa quarta semana, foi alterado para primeiro, reduzir o número de GPIOs utilizadas e segundo, permitir mostrar de maneira simplificada as métricas, o que antes exigia três display 7 segmentos, foi reduzido para apenas um.

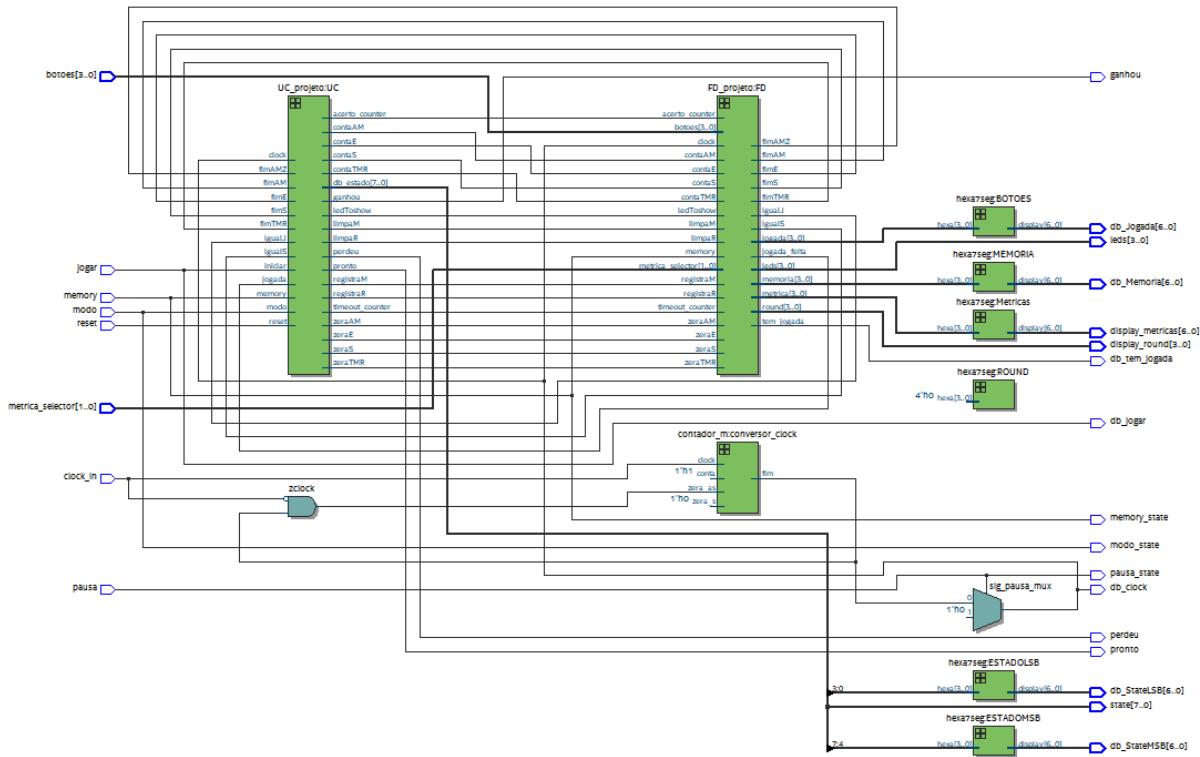


## 2.2 PROJETO DA UNIDADE DE CONTROLE

A Unidade de Controle (UC), durante essa quarta semana, não sofreu qualquer alteração, visto que seu funcionamento já estava bem estabelecido desde a semana passada.

## 2.3 PROJETO DO SISTEMA DIGITAL

Durante essa semana, o código responsável por integrar os componentes gerais do circuito necessitou de algumas alterações, sendo elas a inserção de um módulo que converte o clock padrão de 50MHz da placa FPGA, para 1kHz.

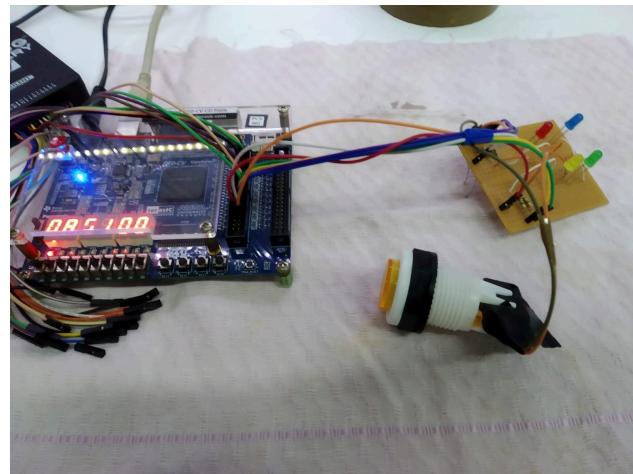


## **fig 1. RTL Viewer**

Como na experiência anterior, a interação entre os dois componentes permite que o sistema funcione de maneira sincronizada, pois a unidade de controle gerencia a execução do fluxo de dados em cada ciclo de clock.

### **3. IMPLANTAÇÃO DO PROJETO**

Este capítulo tem o objetivo de documentar as atividades práticas de execução do projeto desenvolvido e documentado no capítulo 2.3 no ambiente do Laboratório Digital.



**fig 2. Montagem FPGA & Periféricos**

### 3.1 PINAGEM DA PLACA FPGA

A pinagem foi definida no Intel Quartus Prime utilizando o Pin Planner, associando os sinais do projeto aos pinos físicos da FPGA Cyclone V (placa DE0-CV). A figura abaixo apresenta o plano de pinagem detalhado:

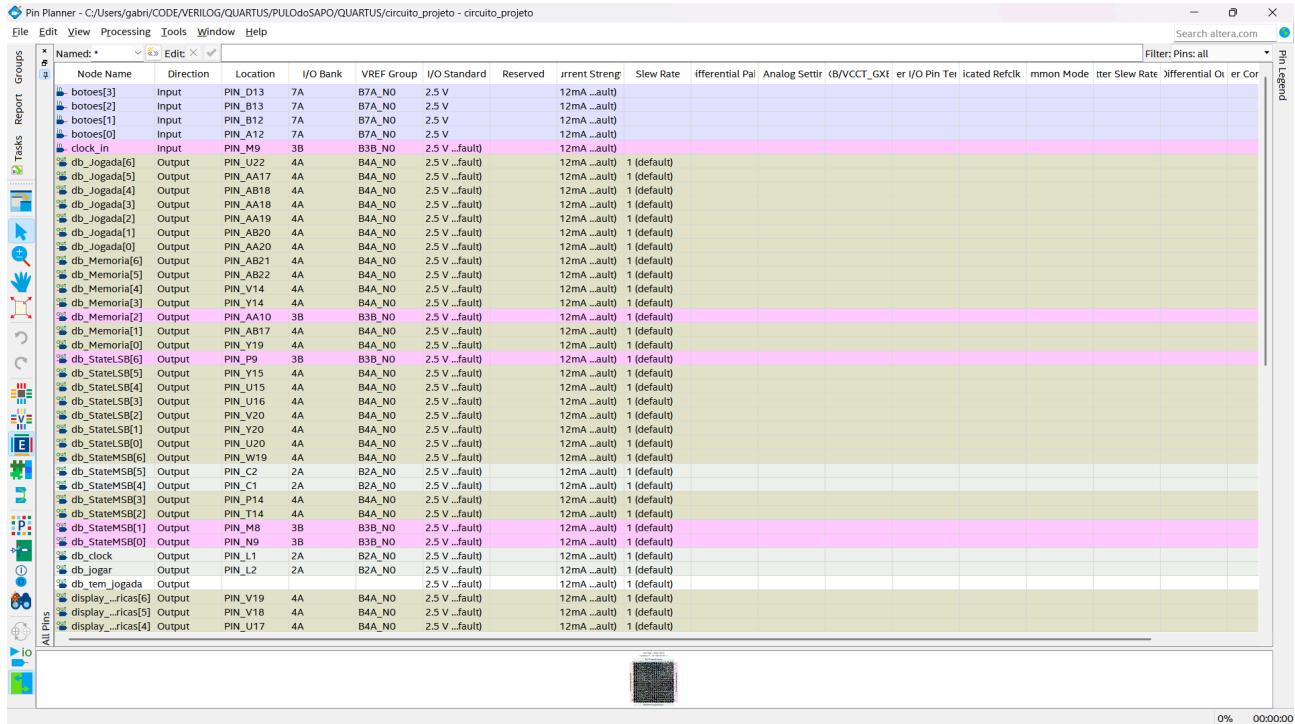


fig 3. Pin planner

### 3.2 ESTRATÉGIA DE MONTAGEM

A implementação do projeto na **placa FPGA DE0-CV** seguiu um fluxo estruturado para assegurar a correta configuração e funcionamento do sistema. As etapas adotadas foram:

#### 1. Compilação do projeto no Quartus Prime

- O código Verilog foi desenvolvido e revisado conforme os requisitos da experiência.
- O projeto foi sintetizado e analisado utilizando as ferramentas **RTL Viewer** e **State Machine Viewer**.
- A designação de pinos da FPGA foi configurada no **Pin Planner**, com base nas tabelas fornecidas pela apostila.

#### 2. Transferência do arquivo de configuração para a FPGA

- Após a compilação, o arquivo **.sof** foi gerado e carregado na FPGA através do **Intel Quartus Prime Programmer**.
- A programação da placa **DE0-CV** foi realizada via conexão **USB Blaster**.

#### 3. Verificação e depuração do circuito

- Os sinais de saída foram analisados utilizando os **LEDs** e os **displays de 7 segmentos** da FPGA.

- Saídas adicionais de depuração foram implementadas para monitorar estados internos do sistema buscando facilitar a identificação e correção de possíveis falhas.
- A validação inicial incluiu testes manuais e simulações no **ModelSim** para verificar a resposta do circuito a diferentes cenários do jogo.

Esse processo garante uma implementação eficiente e confiável, e permite ajustes precisos durante os testes e facilitando a análise do funcionamento do sistema.

### 3.3 ESTRATÉGIA DE DEPURAÇÃO

A depuração do projeto ocorre principalmente por meio da análise dos sinais exibidos nos **LEDs** e nos **displays de 7 segmentos** da FPGA, eliminando a necessidade de 25 de equipamentos adicionais. Para facilitar o diagnóstico, foram implementadas diversas saídas de depuração que monitoram o estado interno do circuito digital. Sempre que falhas são identificadas, ajustes devem ser aplicados no código **Verilog**, seguidos de recompilação no **Quartus Prime**. Além disso, simulações no **ModelSim** validam o comportamento lógico antes da implementação física. Outra etapa fundamental é a **verificação das conexões físicas da FPGA**, nota-se que, diversas vezes, há desatenção no momento de associar os pinos no **Pin Planner**. Desse modo, é necessário prevenir problemas de hardware, como ligações incorretas ou mau contato. Após cada modificação, testes específicos asseguram que as correções não comprometem funcionalidades previamente validadas. Esse processo iterativo preserva a integridade do sistema e permite que o circuito opere conforme o esperado na implementação final.

## 4 PERIFÉRICOS

Esta semana teve como objetivo a verificação geral do sistema e sua integração com os periféricos. Durante as aulas de laboratório, priorizamos a execução de testes de comunicação entre os dispositivos de controle, os módulos de visualização externa e a placa FPGA, garantindo o correto funcionamento do conjunto.

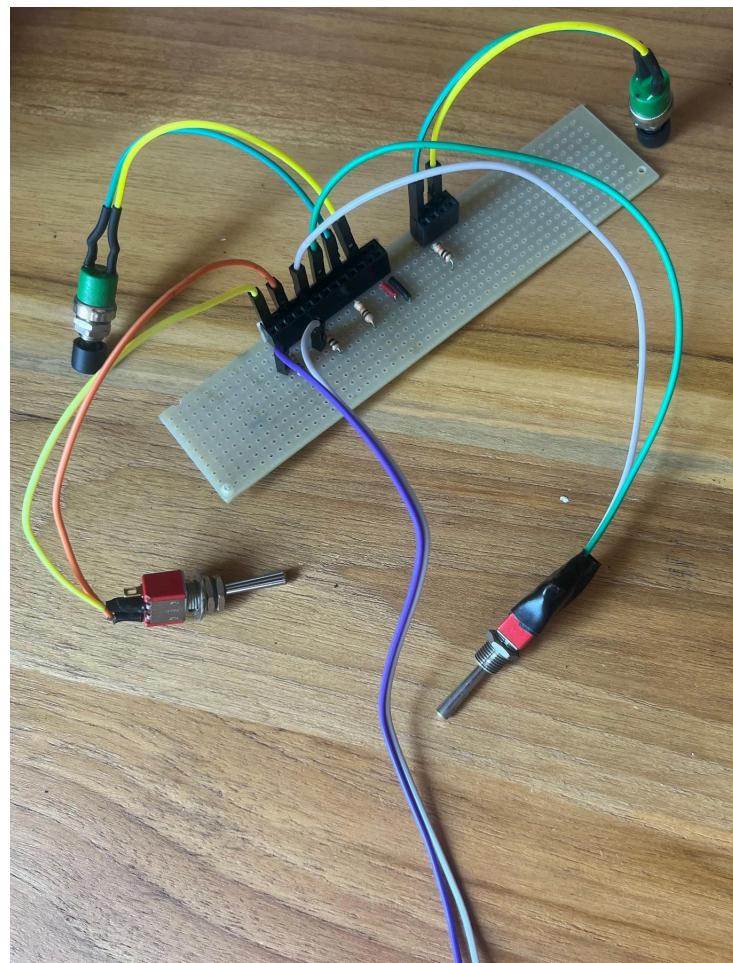
### Painel de controle

O painel de controle segue dividido em dois módulos principais, como nas semanas anteriores:

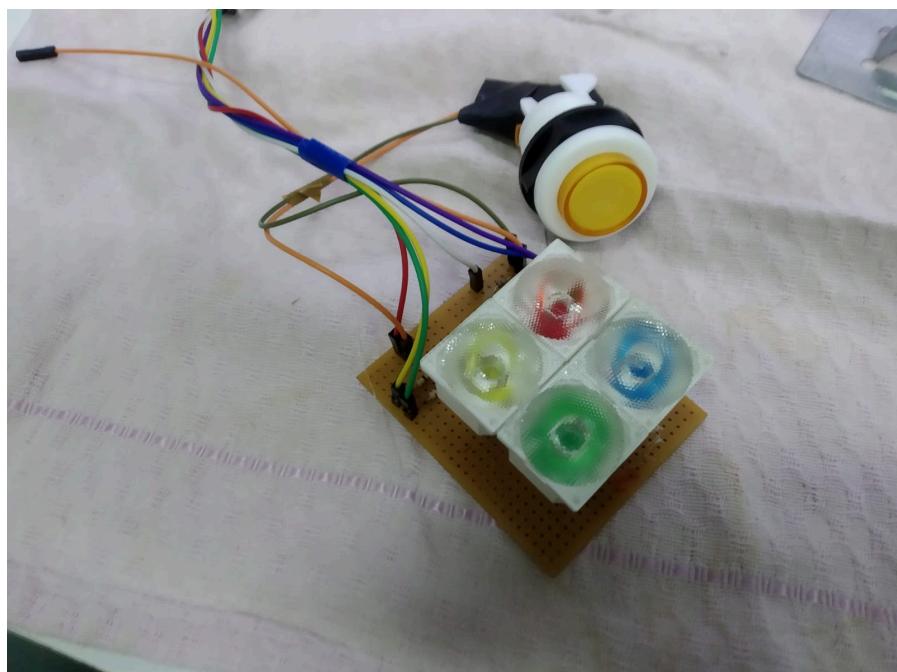
3. **Tela de Jogo** (destinada à criança)
  - Exibe as jogadas e contém o botão de início.
  - Para melhorar a identidade visual, estão sendo desenvolvidos elementos gráficos como logo, mascote e textos personalizados.
  - **Componentes necessários:**
    - 4 LEDs (com resistores adequados)
    - 1 botão de fliperama (alta durabilidade para suportar uso intensivo)
4. **Tela de Configuração** (destinada ao cuidador/responsável)
  - Utiliza um display LCD para apresentar comandos e configurações.
  - Possui botão de reset e chaves para seleção de dificuldade e modo de jogo.
  - **Componentes necessários:**
    - 2 chaves simples
    - 1 botão de reset
    - 1 display LCD 16x2 (com módulo adaptador I2C para simplificar a comunicação)
    - 1 Arduino Uno (responsável pela interface com o LCD e alimentação)

É necessário prestar atenção a alguns detalhes do sistema, devido à fragilidade de alguns componentes. Para isso, deve-se garantir a robustez do botão de fliperama para evitar falhas por uso excessivo, bem como otimizar a comunicação I2C entre o Arduino e o display para evitar latência.

Nesta semana, foi necessário soldar os componentes em diferentes placas com o fim de modularizar o jogo e torná-lo mais fácil de implementar e depurar.



**fig 4. Módulo para chaves de entrada**



**fig 5. Interface frontal para o usuário Jogador**

### **Tapete**

Foram feitas algumas montagens do sistema e testes, apesar de estar funcionando e respondendo bem ao sistema, devido a questões maiores de montagem e organização do grupo, esse elemento provavelmente será retirado do projeto final, uma pena visto ser o componente mais legal do projeto.

## **5 CONCLUSÃO SEMANA 4**

A semana 4 foi uma semana exaustiva, alemã da carga alta de trabalho ainda existem muitas coisas que não foram efetuadas, especialmente material de apresentação de documentação, alem dos componentes básicos ainda não estarem validados e o projeto ainda necessitar de uma apresentação visual mais sofisticada.

### **6.5 SEMANA 5 - FEIRA DE PROJETOS**

**HORAS TRABALHADAS: 12**

#### **1. DESCRIÇÃO DAS ATIVIDADES**

Na semana final, dedicamo-nos aos ajustes finais e à preparação para a apresentação na Feira de Projetos. Como não foram feitas alterações ao software do projeto (incluindo o sistema digital e a unidade de controle), omitimos a apresentação dessa parte. O essencial para a finalização do projeto se deu no design do tapete interativo e seu funcionamento, bem como o painel do usuário. As principais atividades incluíram:

- Finalização do Tapete Interativo:**

Resolvemos os problemas de bounce e validamos o funcionamento do tapete, que terminou por responder adequadamente aos comandos do usuário, embora ainda apresente pequenos ruídos dependendo da intensidade do toque.

Além disso, realizamos testes de integração com a FPGA, o que nos garantiu a correta sincronia entre os movimentos no tapete e os estímulos visuais propostos pelos leds na interface.

- **Interface do Usuário:**

Concluímos a montagem da interface, que está funcional e de fácil utilização. O display LCD exibe mensagens claras, e as chaves de configuração respondem conforme o esperado. Adicionamos também um LED indicador para sinalizar ao usuário o momento de realizar a jogada, melhorando a experiência e acessibilidade.



**fig 1. Interface traseira para o usuário Cuidador**

- **Testes Finais e Ajustes:**

Realizamos testes exaustivos em todos os módulos (FPGA, tapete, painel de controle) para garantir robustez e confiabilidade. Apesar de não funcionar perfeitamente como um todo (especialmente devido a interferência e mal contato dos cabos), o jogo apresentou boa resposta às entradas dos usuários, além de uma resposta decente aos estímulos no tapete.

Após isso, desenvolvemos um pôster explicativo com formato de cartão e um vídeo demonstrativo do jogo. Foi necessário explicar aos avaliadores as diversas possibilidades de uso deste aparelho e inclusive convidamos alguns a experimentar pessoalmente o seu funcionamento.

A imagem abaixo apresenta a organização da bancada para exibir o projeto de forma clara e interativa durante o evento.



**fig 2. Projeto completo para apresentação no evento**

## **7 CONCLUSÃO GERAL**

O projeto **Pulo do Sapo** foi desenvolvido com o objetivo de criar uma ferramenta lúdica e acessível para auxiliar crianças com Transtorno do Espectro Autista (TEA) no desenvolvimento de habilidades motoras e de autorregulação. Ao longo das cinco semanas de trabalho, enfrentamos

desafios técnicos e operacionais, mas também alcançamos conquistas significativas que validaram nossa proposta.

Um dos principais avanços foi a integração bem-sucedida entre hardware e software, que se deu na união da FPGA, do tapete interativo e da interface do usuário em um sistema funcional e coeso. O tapete, após ajustes finais, responde adequadamente aos comandos, embora ainda apresente um leve *bounce* em algumas situações. A interface, por sua vez, foi finalizada com êxito, garantindo uma experiência intuitiva por meio de um display LCD claro e um LED indicador que sinaliza o momento correto para a jogada.

Além disso, conseguimos assegurar a robustez do sistema com testes exaustivos realizados na última semana que permitiram identificar e corrigir falhas. A preparação para a Feira de Projetos também foi concluída, com a criação de materiais explicativos e a organização da bancada para demonstrar o funcionamento do sistema de forma interativa.

Este trabalho não apenas cumpriu os requisitos técnicos da disciplina, mas também proporcionou um aprendizado valioso em desenvolvimento de sistemas digitais, trabalho em equipe e adaptação a imprevistos. Apesar das limitações, o **Pulo do Sapo** mostrou-se uma solução viável e com potencial para expansões futuras, como a inclusão de novos níveis de dificuldade e métricas mais detalhadas para acompanhamento terapêutico.