



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

## PCS3635 – LABORATÓRIO DIGITAL I

### SEMANA 1 – Implementação

Relato da Bancada A3 – Turma 2 – Prof. Reginaldo

Data de Emissão: 10 de Março de 2025.

Nome: Pedro Henrique Zanato da Costa	Número USP: 13874761
Nome: Enzo Koichi Jojima	Número USP: 14568285
Nome: Eduardo Ribeiro do Amparo Rodrigues de Souza	Número USP: 14567346

O PROJETO PODE SER ENCONTRADO NO [GITHUB](#)

## 1. CAPÍTULO 1 - SEMANA 0: ESPECIFICAÇÃO

### 1.1. Introdução

O projeto realizado será um jogo sério, denominado *NeuroSync*, de caráter cooperativo voltado para o desenvolvimento social e cognitivo de pessoas no espectro autista. O Transtorno do Espectro Autista (TEA) “*é um distúrbio caracterizado pela alteração das funções do neurodesenvolvimento do indivíduo, interferindo na capacidade de comunicação, linguagem, interação social e comportamento*” [1]. Nesse contexto, o jogo irá contribuir para a interação social e desenvolvimento lógico, cognitivo e motor dos jogadores, de modo a auxiliar na inclusão social e educação dos indivíduos. O jogo terá como inspiração o jogo *Keep Talking and Nobody Explodes* [2], bem como no projeto desenvolvido na disciplina até então, o *Jogo Desafio Memória*.

Desse modo, tratando-se de um jogo sério cooperativo, o escopo do projeto é, principalmente, auxiliar no desenvolvimento social dos jogadores, estimulando a interação entre os players através de algo dinâmico como o jogo.

## 1.2. Descrição do Funcionamento do Projeto

### 1.2.1. Descrição textual do jogo

A ideia básica do projeto é desenvolver um jogo de caráter cooperativo. O jogo será jogado por dois jogadores que deverão se unir para vencer o jogo.

Um dos players, digamos o player 1, ficará de frente a um **display de leds ou display numérico** e uma **bancada de botões numerados/coloridos**. O player 2 terá em suas mãos um **manual contendo um conjunto de jogadas** (botões a serem apertados) para cada combinação de leds que podem aparecer.

Uma partida será determinada por um número N de rodadas e terá um tempo limite T. Ao iniciar um jogo, um **temporizador será iniciado com o tempo T**. Em cada rodada, o sistema exibirá uma **combinação aleatória de LEDs ou um número** para o player 1, que por sua vez deverá comunicar ao player 2 essa combinação de LEDs. O player 2 deve procurar no manual o conjunto de jogadas relacionado a essa combinação de leds ou número e comunicar ao player 1 esse conjunto de jogadas. O player 1 deverá realizar esse conjunto de jogadas, passando assim para a próxima rodada. Para vencer, os jogadores deverão terminar as N rodadas antes do temporizador chegar em 0 (ou seja, em um intervalo de tempo inferior a T), caso contrário, os jogadores perdem. Se o player 2 erra uma jogada, o temporizador T é **decrementado de um valor X** e ele deve refazer a jogada.

O jogo também contará com diferentes **níveis de dificuldade**, que alteram o número N de jogadas e o tempo limite T, e o decremento X.

*Obs:* O professor Antonio recomendou buscar alguma forma de flexibilização das regras do jogo. Por exemplo, nos níveis mais fáceis, o jogador não necessariamente precisa acertar as jogadas na ordem correta, mas apenas acertar as jogadas especificadas. Uma ideia preliminar seria a seguinte:

- Nível Fácil: maior tempo T, menor número N de jogadas:
  - Player 1 deve acertar as jogadas independentemente da ordem e pode apertar os botões quantas vezes quiser.
  - Rodadas mais simples, de 2 jogadas.
  - Combinações de LEDs simples, apenas 1 LED por vez.
  - Sem penalidade para erro, focado em ensinar as mecânicas do jogo.
- Nível Médio: menor tempo T, maior decremento X, maior número N de jogadas:
  - Player 1 deve acertar as jogadas independentemente da ordem e pode apertar os botões quantas vezes quiser.
  - Rodadas mais simples, de 2 jogadas.
  - Combinações de LEDs simples, apenas 1 LED por vez.

- Cada botão errado decrementa o temporizador apenas uma vez, se o mesmo botão errado for acionado novamente, o temporizador não é decrementado novamente.
- Nível Difícil: menor tempo T, maior decremento X, maior número N de jogadas:
  - Player 1 deve acertar as jogadas independentemente da ordem.
  - Cada acionamento adicional acarretará em um erro de jogada, decrementando o temporizador.
  - Rodadas mais complexas, de 3 ou 4 jogadas.
  - Combinações de LEDs complexas, 2 LEDs por vez.
  - Cada erro de jogada decrementa o temporizador apenas uma vez, se o mesmo botão errado for acionado novamente (seja um botão certo duplicado ou um botão errado acionado), o temporizador não é decrementado novamente.
- Nível Hardcore: menor tempo T, maior decremento X, maior número N de jogadas:
  - Player 1 deve acertar as jogadas independentemente da ordem.
  - Cada acionamento adicional acarretará em um erro de jogada, decrementando o temporizador.
  - Rodadas mais complexas, de 3 ou 4 jogadas.
  - Combinações de LEDs complexas, 2 LEDs por vez.
  - Cada erro adicional de jogada decrementa o temporizador novamente, ou seja, se o mesmo botão errado for acionado novamente (seja um botão certo duplicado ou um botão errado acionado), o temporizador é decrementado novamente.

Fomos orientados pelo monitor a focar, na disciplina de Laboratório Digital 1, no nível Fácil do jogo, e como desafio tentar implementar o nível Médio. E deixar os outros níveis para Laboratório Digital 2. Também houve a ideia de implementar uma interface gráfica para o projeto, mas também fomos orientados a deixar isso para Laboratório Digital 2 uma vez que não vimos ainda comunicação serial com a FPGA, que será o foco do Laboratório Digital 2.

### 1.2.2. Arquitetura Estrutural

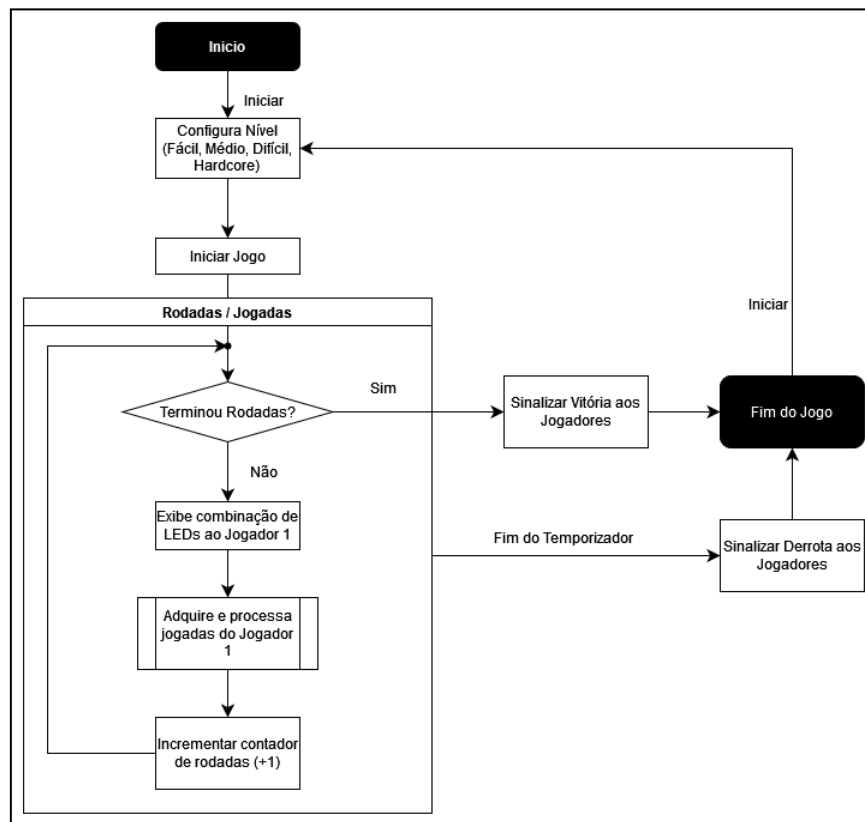


Figura 1.1 - Arquitetura estrutural do Jogo em alto nível.

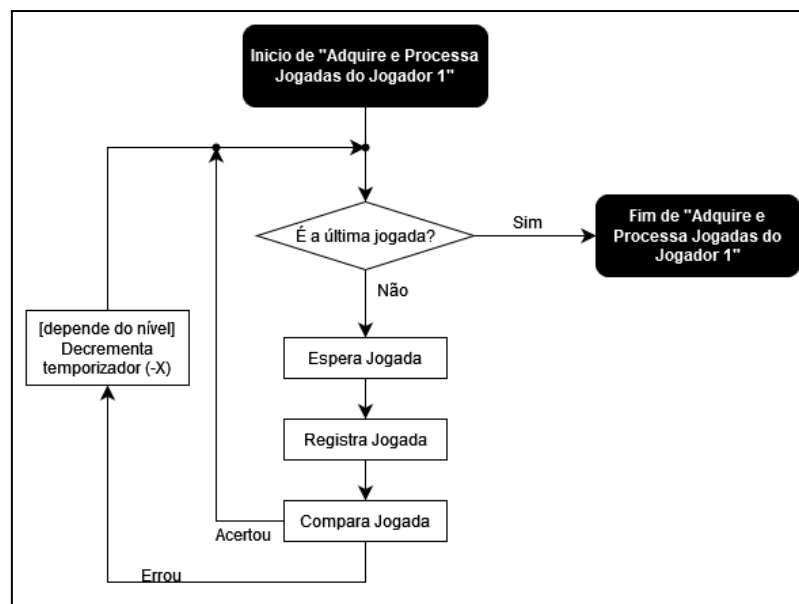


Figura 1.2 - Arquitetura estrutural de "Adquire e Processa Jogadas do Jogador 1".

### 1.3. Especificação dos Requisitos

#### 1.3.1. Requisitos Funcionais

##### 1.3.1.1. **Código:** MODO\_COOPERATIVO

**Requisito:** Modo Cooperativo

**Descrição:** O jogo deve permitir que dois jogadores interajam para resolver desafios.

**Prioridade:** ☒ Alta ☐ Média ☐ Baixa

**Estabilidade:** ☒ Alta ☐ Média ☐ Baixa

**Rationale:** Essencial para garantir a jogabilidade cooperativa.

**Requisitos associados:** COMUNICACAO\_INTERACAO

##### 1.3.1.2. **Código:** EXIBICAO\_COMANDOS

**Requisito:** Exibição de Comandos

**Descrição:** O sistema deve apresentar ao jogador 1 sequências de LEDs ou números.

**Prioridade:** ☒ Alta ☐ Média ☐ Baixa

**Estabilidade:** ☒ Alta ☐ Média ☐ Baixa

**Rationale:** Permite a comunicação eficiente entre os jogadores.

**Requisitos associados:** MANUAL\_INSTRUcoes

##### 1.3.1.3. **Código:** MANUAL\_INSTRUcoes

**Requisito:** Manual de Instruções

**Descrição:** O jogador 2 deve ter acesso a um manual com as instruções para cada combinação apresentada.

**Prioridade:** ☒ Alta ☐ Média ☐ Baixa

**Estabilidade:** ☒ Alta ☐ Média ☐ Baixa

**Rationale:** Fundamental para guiar o jogador 2 na interação com o jogo.

**Requisitos associados:** EXIBICAO\_COMANDOS

##### 1.3.1.4. **Código:** CONTROLE\_TEMPO

**Requisito:** Controle de Tempo

**Descrição:** O jogo deve incluir um temporizador regressivo.

**Prioridade:** ☒ Alta ☐ Média ☐ Baixa

**Estabilidade:** ☒ Alta ☐ Média ☐ Baixa

**Rationale:** Define o ritmo do jogo e evita partidas

infinitas.

**Requisitos associados:** PENALIZACAO\_ERROS

- 1.3.1.5. **Código:** INDICACAO\_FEEDBACK  
**Requisito:** Indicação de Feedback Visual  
**Descrição:** LEDs devem indicar o estado atual do jogo (acertos, erros e tempo restante).  
**Prioridade:** ☐ Alta ☒ Média ☐ Baixa  
**Estabilidade:** ☒ Alta ☐ Média ☐ Baixa  
**Rationale:** Garante que os jogadores recebam retorno visual imediato.  
**Requisitos associados:** EXIBICAO\_COMANDOS

- 1.3.1.6. **Código:** NIVEL\_DIFICULDADE  
**Requisito:** Níveis de Dificuldade  
**Descrição:** O jogo deve permitir selecionar entre níveis Fácil, Médio, Difícil e Hardcore.  
**Prioridade:** ☒ Alta ☐ Média ☐ Baixa  
**Estabilidade:** ☒ Alta ☐ Média ☐ Baixa  
**Rationale:** Ajusta a complexidade do jogo conforme a experiência do usuário.  
**Requisitos associados:** CONTROLE\_TEMPO

### 1.3.2. Requisitos Não Funcionais

- 1.3.2.1. **Código:** DESEMPENHO\_SISTEMA  
**Requisito:** Desempenho  
**Descrição:** O sistema deve processar entradas em menos de 100ms.  
**Prioridade:** ☒ Alta ☐ Média ☐ Baixa  
**Estabilidade:** ☒ Alta ☐ Média ☐ Baixa  
**Rationale:** Evita atrasos que possam prejudicar a experiência do usuário.  
**Requisitos associados:** CONTROLE\_TEMPO

- 1.3.2.2. **Código:** USABILIDADE\_INTERFACE  
**Requisito:** Usabilidade  
**Descrição:** A interface deve ser clara e intuitiva para jogadores neurodivergentes.

**Prioridade:** ☒ Alta ☐ Média ☐ Baixa  
**Estabilidade:** ☒ Alta ☐ Média ☐ Baixa  
**Rationale:** Garante que o jogo seja acessível a um público mais amplo.  
**Requisitos associados:** EXIBICAO\_COMANDOS

1.3.2.3. **Código:** SEGURANCA\_JOGO  
**Requisito:** Segurança  
**Descrição:** O jogo não pode gerar estimulação excessiva prejudicial a pessoas com TEA.  
**Prioridade:** ☒ Alta ☐ Média ☐ Baixa  
**Estabilidade:** ☒ Alta ☐ Média ☐ Baixa  
**Rationale:** Garante que o jogo seja adequado ao público-alvo.  
**Requisitos associados:** USABILIDADE\_INTERFACE

1.3.2.4. **Código:** MODULARIDADE\_SOFTWARE  
**Requisito:** Modularidade do Software  
**Descrição:** O código deve ser estruturado de forma modular para facilitar manutenção e expansões.  
**Prioridade:** ☐ Alta ☒ Média ☐ Baixa  
**Estabilidade:** ☒ Alta ☐ Média ☐ Baixa  
**Rationale:** Garante a flexibilidade para futuras melhorias.  
**Requisitos associados:** DESEMPENHO\_SISTEMA

1.3.2.5. **Código:** SONS\_AUXILIARES  
**Requisito:** Sons Auxiliares  
**Descrição:** O jogo pode emitir sons para indicar ações, como erros e sucessos.  
**Prioridade:** ☐ Alta ☐ Média ☒ Baixa  
**Estabilidade:** ☒ Alta ☐ Média ☐ Baixa  
**Rationale:** Melhora a imersão e feedback do jogador.  
**Requisitos associados:** INDICACAO\_FEEDBACK

## 1.4. Revisão da Arquitetura da Solução

### Componentes Principais

- **Módulo de Exibição:** Controla os LEDs e displays numéricos.
- **Módulo de Entrada:** Processa comandos do jogador 1 (botões).
- **Módulo de Lógica de Jogo:** Avalia os comandos e verifica se estão corretos.
- **Temporizador:** Controla o tempo restante na partida.
- **Sistema de Níveis:** Determina as regras para cada dificuldade.
- **Feedback ao Usuário:** Fornece indicações visuais e sonoras.

## 1.5. Cronograma da Solução

<b>Semana</b>	<b>Atividade</b>
0.2	Finalização da Especificação e Planejamento
1	Implementação do Módulo com Nível Fácil
2	Implementação de Interface e Displays HEX
3	Testes e Ajustes da Interface e Feedback ao Usuário
4	Integração e Testes Finais
5	Apresentação na Feira de Projetos

*O cronograma permite rastrear cada requisito, garantindo que o desenvolvimento ocorra de forma estruturada e eficiente.*

## 2. CAPÍTULO 2 - SEMANA 1: IMPLEMENTAÇÃO DO NÍVEL FÁCIL

### 2.1. Introdução

Nessa semana, tentaremos implementar o nível fácil do jogo em Verilog. Iremos manter funcionalidades de nível para posteriormente tentarmos implementar o nível médio, conforme desafio proposto pelo monitor na Semana 0 (seção 1.2.1 do documento).

### 2.2. Descrição do Projeto

O projeto terá como base a arquitetura do jogo desafio memória, desenvolvido até agora nas aulas da disciplina. O fluxograma base do projeto pode



ser visto na seção 1.2.2 do documento, tomaremos esse fluxograma como base para a máquina de estados do projeto.

O timeout do projeto está programado para 5 minutos (300000 ms) após o início do jogo.

## **2.3. Detalhamento do Projeto Lógico**

### **2.3.1. Fluxo de Dados**

Foi implementado o funcionamento básico do jogo, apenas as funcionalidades mais fundamentais.

O projeto é baseado no jogo desenvolvido até então na disciplina, mas conta com certas modificações.

A memória de jogadas agora é controlada a partir da memória de LEDs, ou seja, de acordo com os LEDs exibidos, teremos um conjunto de jogadas esperadas diferente. Além disso, como no nível fácil não estamos nos importando com a ordem das jogadas, cada conjunto de jogadas é armazenado em um único endereço na memória de jogadas. Isso, é claro, traz alguns problemas.

O primeiro é saber quando uma jogada certa é repetida, isso é importante pois o NeuroSync só passará para a próxima sequência quando o jogador acertar as duas jogadas distintas da rodada, e o segundo é saber quando o jogador acertou as duas jogadas. Para isso, introduzimos um registrador `acertoAnterior` que armazena 0 quando não houve acerto e armazena o acerto quando há o primeiro acerto. Desse modo, podemos comparar se a jogada atual é igual ao acerto anterior e usar isso para a transição de estados na Unidade de Controle. Posteriormente, isso será feito com um comparador a parte no Fluxo de Dados (modificação 2.1).

Além disso, introduzimos um novo módulo chamado `bigAND` que realiza um `and bit a bit` nas entradas para verificar se a jogada atual possui algum bit em comum com a jogada esperada (saída da memória de jogadas). Posteriormente, será tratado o caso de dois botões serem acionados ao mesmo tempo, aprimorando esse módulo `bigAND` ou introduzindo uma lógica adicional para tratar isso (modificação 2.2).

Por fim, a ideia para os displays de 7 segmentos é realizar o display de textos de acordo com o estado ou exibir o timer. Isso será discutido e implementado posteriormente (modificação 2.3).

Em rosa, os “*TO-DOS*” do projeto.

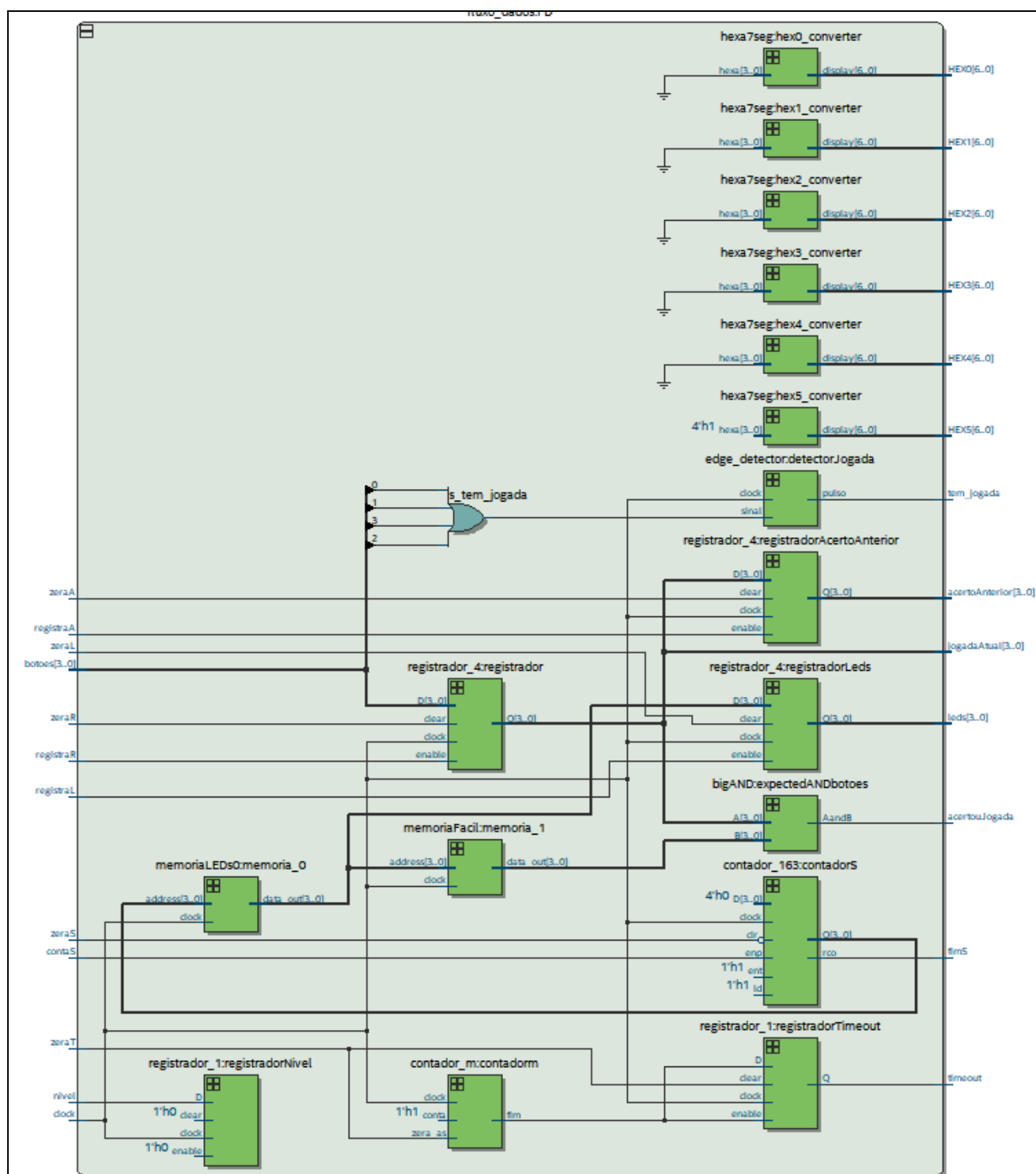
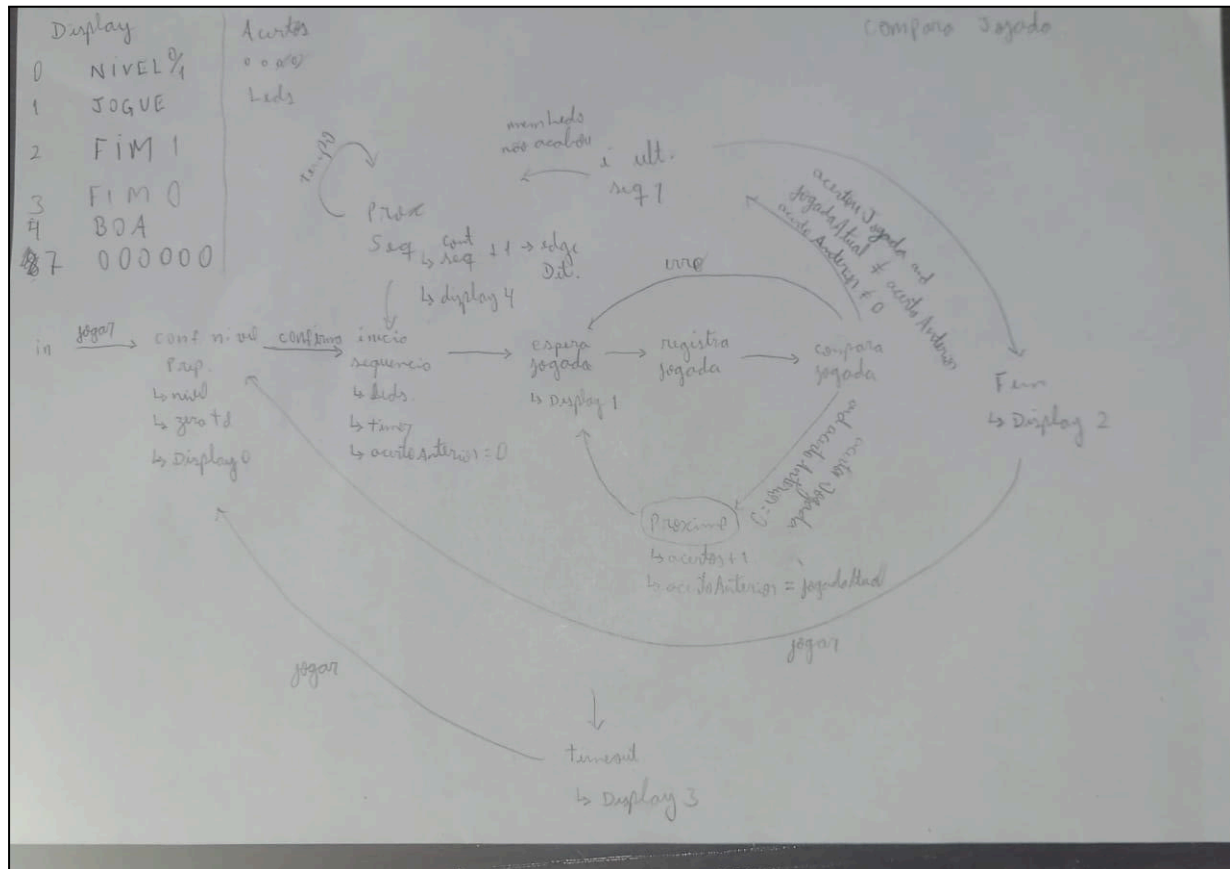


Figura 2.1 - Diagrama de Fluxo de Dados Semana 1

### 2.3.2. Unidade de Controle

A unidade de controle também foi adaptada do jogo feito até então na disciplina. Note que o estado de preparação está sendo usado para esperar a configuração de nível, a transição de estados está mais simples em relação à última experiência, uma vez que há menos estados pois a exibição de leds e obtenção de

jogadas está mais simples. Porém a lógica de transição de estados é mais complexa e pode ser vista melhor no código em Verilog da unidade de controle.



*Figura 2.2 - Diagrama de Transição de Estados em Alto Nível da Semana 1*

### 2.3.3. NeuroSync

Aqui vemos a implementação do NeuroSync como um todo. O funcionamento básico do jogo como timeout, exibição de leds, obtenção e tratamento de jogadas, está funcionando. Apesar disso, há detalhes a serem melhorados na implementação de tais funcionalidades como incorporação de boas práticas e compartimentalização de certas lógicas. O objetivo dessa semana foi tirar do papel tudo que planejamos na Semana 0 e ter um esqueleto base para o NeuroSync.

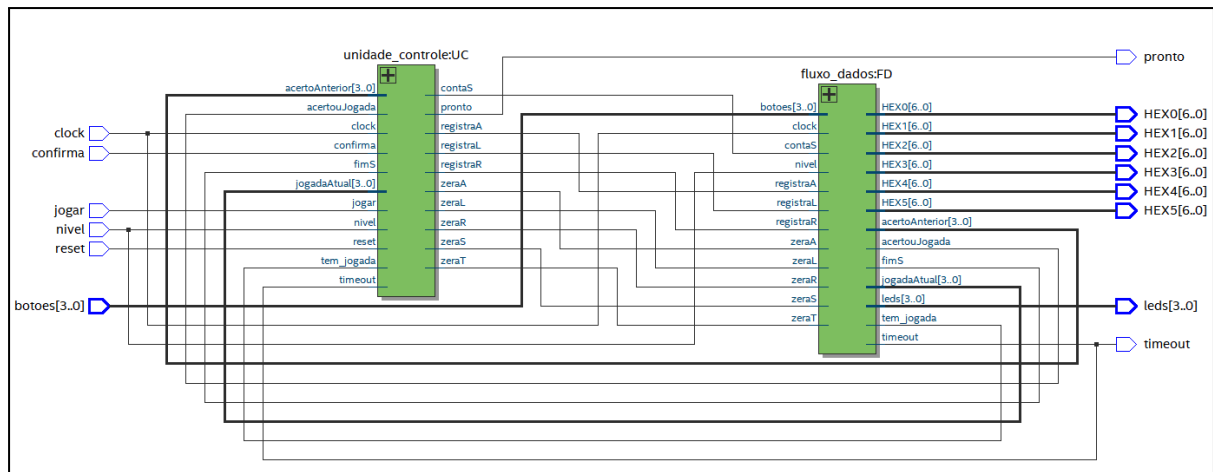


Figura 2.3 - Diagrama RTL-View do módulo NeuroSync Semana 1

## 2.4. Testes

### 2.4.1. Plano de Teste 1 - Acerto de todas as rodadas antes do timeout

Saída esperada: Pronto = 1 e Timeout = 0

### 2.4.2. Plano de Teste 2 - Timeout após 5 minutos

Saída esperada: Pronto = 1 e Timeout = 1

### 2.4.3. Plano de Teste 3 - Erro na segunda jogada da segunda rodada

Saída esperada: db\_estado = 3 (espera\_jogada), Timeout = 0 e Pronto = 0 (sem penalização por erro).

### 2.4.4. Plano de Teste 4 - Acerto repetido na segunda jogada da segunda rodada

Saída esperada: db\_estado = 3 (espera\_jogada), Timeout = 0 e Pronto = 0 (sem penalização por repetição).

## 2.5. Implementação do Projeto

### 2.5.1. Pinagem na Placa FPGA

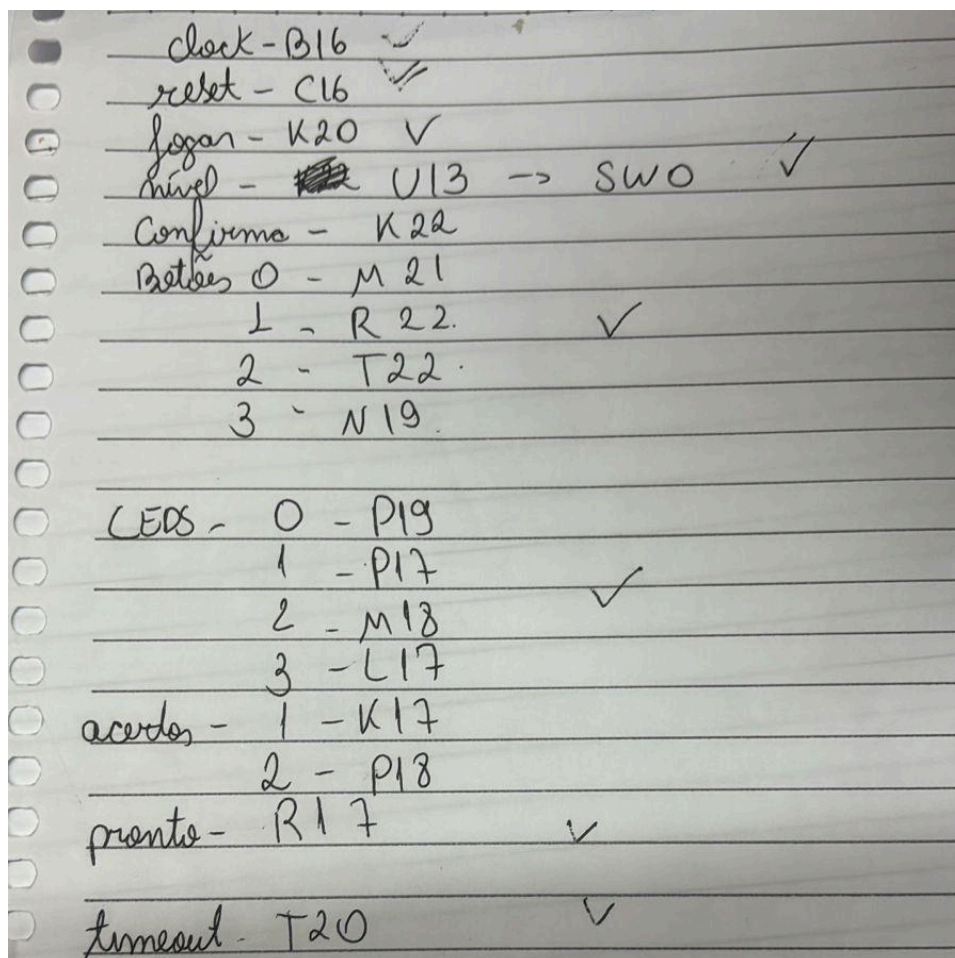


Figura 2.4 - Esquema de Pinagem das Entradas e Saídas principais do NeuroSync.

## 2.6. Realização em Laboratório

### 2.6.1. Modificação 2.1 - Compartimentalização de partes da lógica

Realizamos a implementação da lógica no fluxo de dados com o uso de dois contadores. Um contador compara a jogada atual com o acerto anterior e o outro compara o acerto anterior com zero (0000).

Novas saídas de condição: `jogadaAtualEQUALS acertoAnterior` e `acertoAnteriorEQUALS zero`.

### 2.6.2. Modificação 2.2 - Lógica do bigAND

O grupo identificou que o acionamento simultâneo de dois botões em um tempo menor que um período de clock é praticamente impossível e decidiu manter o módulo bigAND como estava.

### 2.6.3. Modificação 2.3 - Displays Hexadecimais

O monitor orientou ao grupo deixar essa modificação para o futuro, uma vez que não se trata de um requisito com uma prioridade muito alta, devemos então focar em coisas mais prioritárias.

### 2.6.4. Problemas identificados

Ao implementarmos a montagem na placa FPGA, notamos que o jogo não funcionou como na simulação no ModelSim. O grupo irá posteriormente tirar saídas de depuração para a FPGA para poder depurar o circuito.

### 2.6.5. Exibição dos Acertos

O grupo começou a trabalhar no requisito funcional INDICAÇÃO\_FEEDBACK (1.3.1.5) e USABILIDADE\_INTERFACE (1.3.2.2), iniciando a implementação da exibição de acertos e de um feedback luminoso ao acertar uma sequência. Foram adicionadas:

- saída adicional **acertos** de 2 bits - leds indicarão os acertos do jogador irão piscar ao acertar uma sequência inteira;
- novos estados para piscar os leds ao final de uma sequência;
- lógica adicional de transição de estados e componentes no fluxo de dados para controlar a exibição dos leds.

A implementação não foi concluída e nem testada, o que será realizado nas atividades da Semana 2.

### 2.6.6. Versão Preliminar do Manual

O grupo começou o desenvolvimento de uma versão preliminar do manual que será de posse do player 2. Por enquanto o manual conta com uma tabela em binário para as 4 sequências possíveis do nível fácil. A ideia é criar um manual mais didático e lúdico, com muitas imagens e cores.

## 3. REFERÊNCIAS BIBLIOGRÁFICAS

[1] TEA: saiba o que é o Transtorno do Espectro Autista e como o SUS tem dado assistência a pacientes e familiares, Ministério da Saúde: <https://www.gov.br/saude/pt-br/assuntos/noticias/2022/abril/tea-saiba-o-que-e-o-transtorno-do-espectro-autista-e-como-o-sus-tem-dado-assistencia-a-pacientes-e-familiares>.

[2] Jogo *Keep Talking and Nobody Explodes*: <https://keeptalkinggame.com/>, Steel Crate Games®.

[3] OBERG, R.; PROBASCO, L.; ERICSSON, M.; *Applying Requirements Management with Use Cases*. Rational Software Corporation. Technical Paper TP505 (Version 1.4), 2001.

