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

PCS3635 – LABORATÓRIO DIGITAL I

SEMANA 2 – Implementação

Planejamento da Bancada A3 – Turma 2 – Prof. Reginaldo

Data de Emissão: 14 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**

OS CÓDIGOS DO PROJETO PODE SER ENCONTRADO NO *GITHUB*

ÍNDICE

1. CAPÍTULO 1 - SEMANA 0: ESPECIFICAÇÃO 3
1.1. Introdução 3
1.2. Descrição do Funcionamento do Projeto 3
1.2.1. Descrição textual do jogo 3
1.2.2. Arquitetura Estrutural 5
1.3. Especificação dos Requisitos 6
1.3.1. Requisitos Funcionais 6
1.3.2. Requisitos Não Funcionais 7
1.4. Revisão da Arquitetura da Solução 9
1.5. Cronograma da Solução 9
2. CAPÍTULO 2 - SEMANA 1: IMPLEMENTAÇÃO DO NÍVEL FÁCIL 9

2.1. Introdução..... 9
2.2. Descrição do Projeto..... 10
2.3. Detalhamento do Projeto Lógico..... 10
2.3.1. Fluxo de Dados..... 10
2.3.2. Unidade de Controle..... 11
2.3.3. NeuroSync..... 12
2.4. Testes..... 13
2.4.1. Plano de Teste 1 - Acerto de todas as rodadas antes do timeout..... 13
2.4.2. Plano de Teste 2 - Timeout após 5 minutos..... 13
2.4.3. Plano de Teste 3 - Erro na segunda jogada da segunda rodada..... 13
2.4.4. Plano de Teste 4 - Acerto repetido na segunda jogada da segunda rodada..... 13
2.5. Implementação do Projeto..... 14
2.5.1. Pinagem na Placa FPGA..... 14
2.6. Realização em Laboratório..... 14
2.6.1. Modificação 2.1 - Compartimentalização de partes da lógica..... 14
2.6.2. Modificação 2.2 - Lógica do bigAND..... 14
2.6.3. Modificação 2.3 - Displays Hexadecimais..... 15
2.6.4. Problemas identificados..... 15
2.6.5. Exibição dos Acertos..... 15
2.6.6. Versão Preliminar do Manual..... 15
3. CAPÍTULO 3 - SEMANA 2: IMPLEMENTAÇÃO DA INTERFACE E MANUAL.. 15
3.1. Introdução..... 15
3.2. Descrição do Projeto..... 16
3.3. Detalhamento do Projeto..... 16
3.3.1. Fluxo de Dados..... 16
3.3.1.1. LEDs Acertos..... 16
3.3.1.2. Displays de 7 Segmentos..... 16
3.3.2. Unidade de Controle..... 18
3.3.3. NeuroSync..... 19
3.3.4. Manual..... 19
3.4. Testes..... 20
REFERÊNCIAS BIBLIOGRÁFICAS..... 20

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

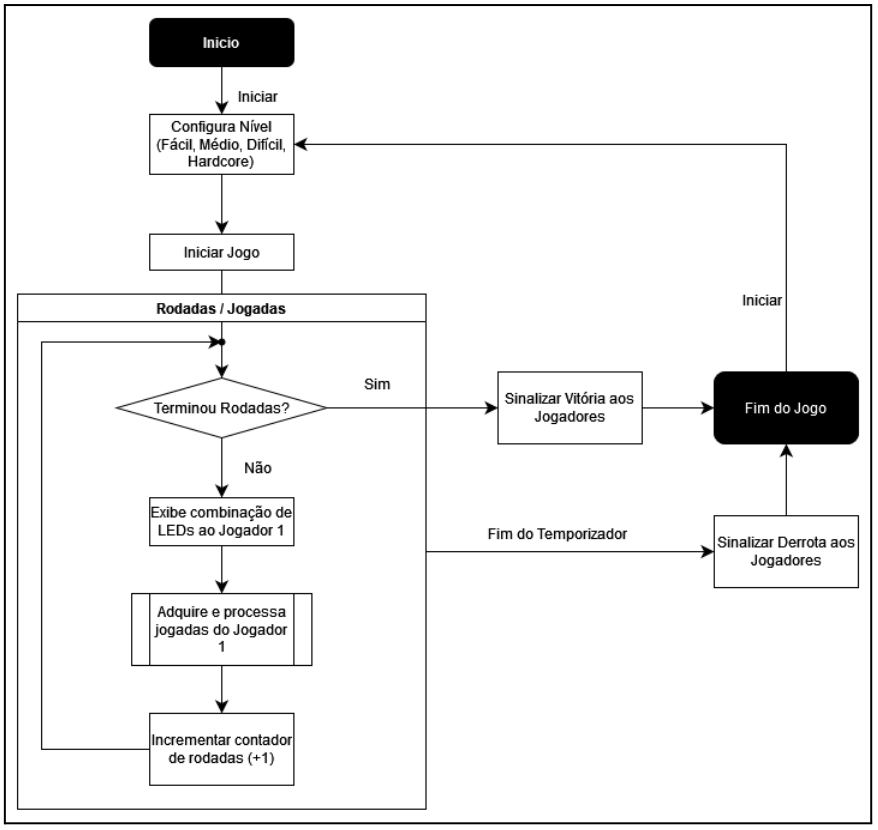

Figura 1.1 - Arquitetura estrutural do Jogo em alto nível.

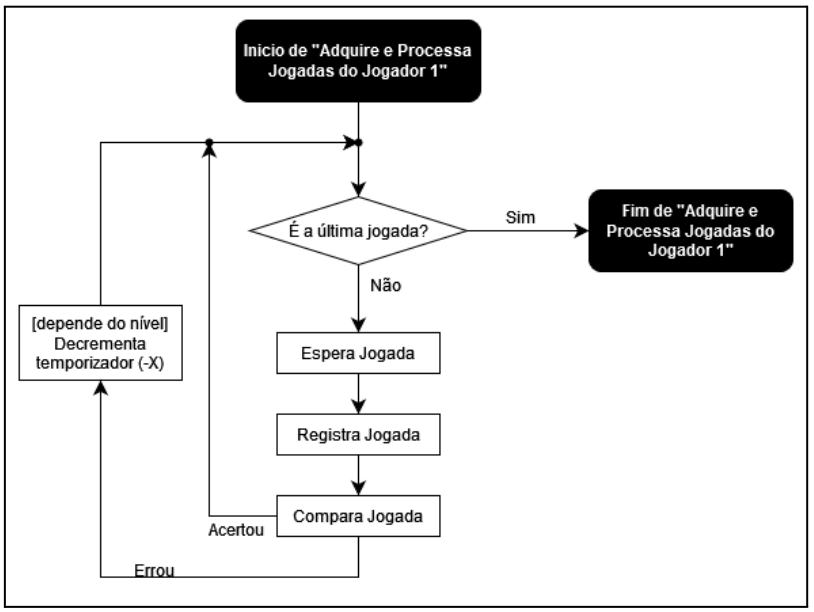

Figura 1.2 - Arquitetura estrutural de "Adquire e Processa Jogadas do Jogador 1".

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

Semana	*Atividade*
0.2	Finalização da Especificação e Planejamento
1	Implementação do Módulo com Nível Fácil
2	Implementação de Interface, Displays HEX e Manual
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).

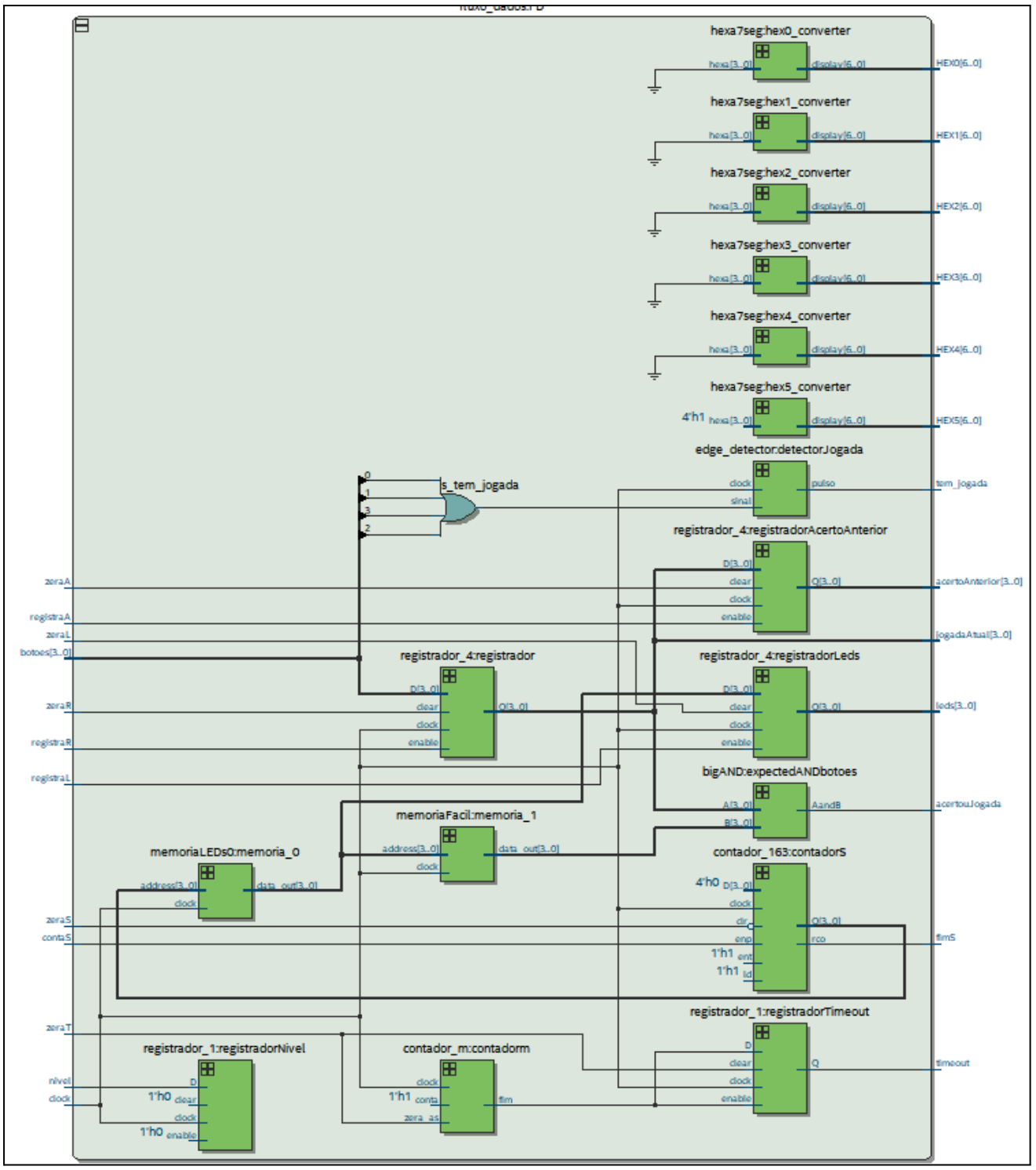

Figura 2.1 - Diagrama de Fluxo de Dados Semana 1

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.

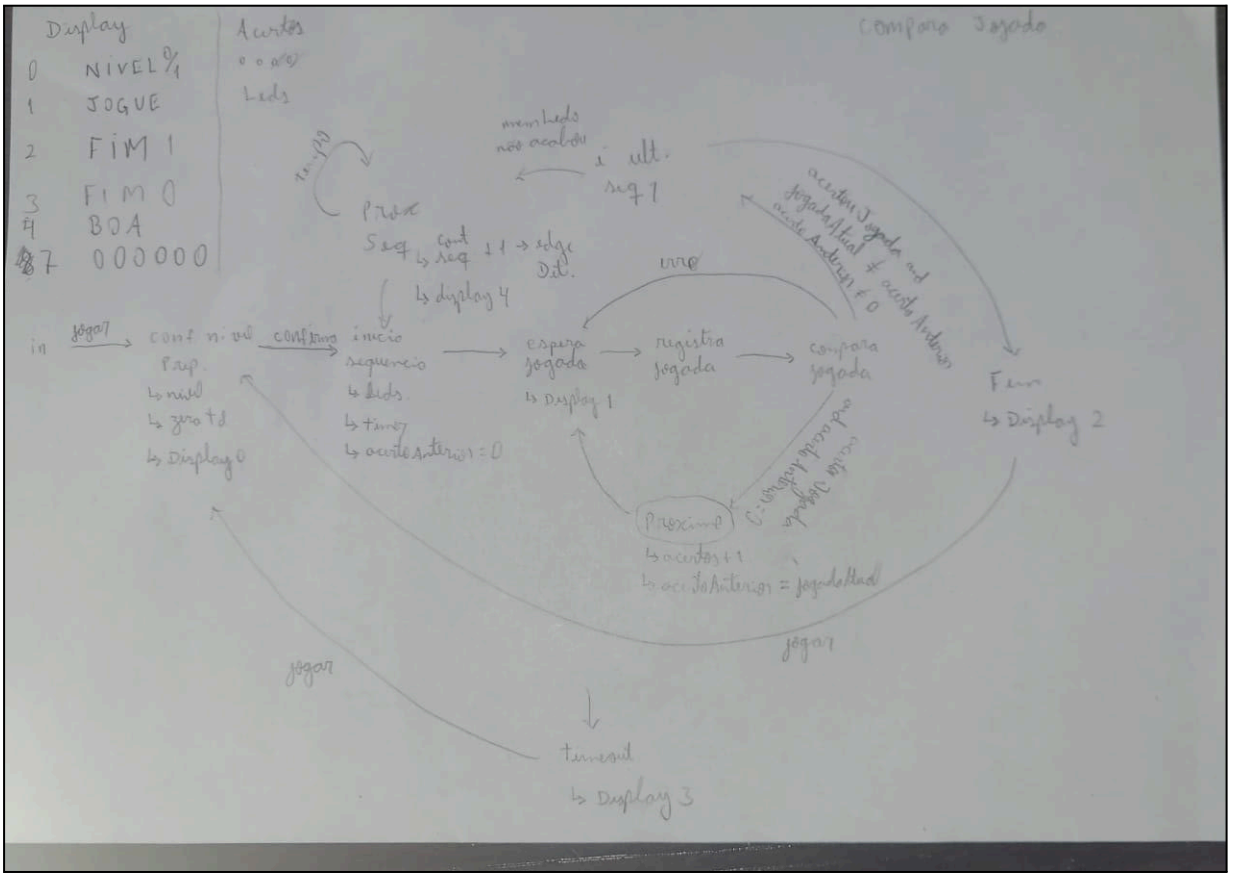

Figura 2.2 - Diagrama de Transição de Estados em Alto Nível da Semana 1

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.

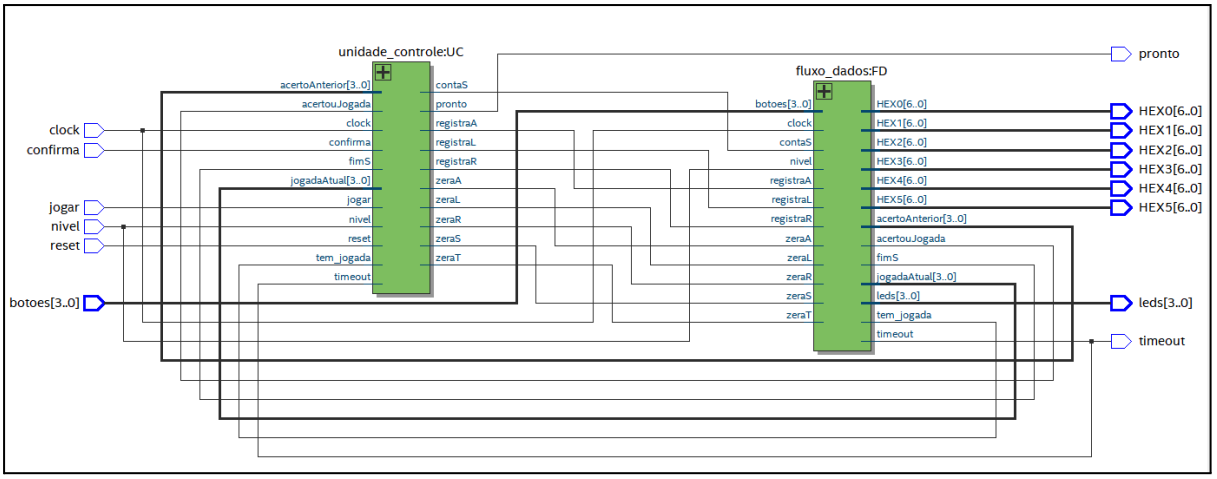

Figura 2.3 - Diagrama RTL-View do módulo NeuroSync Semana 1

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

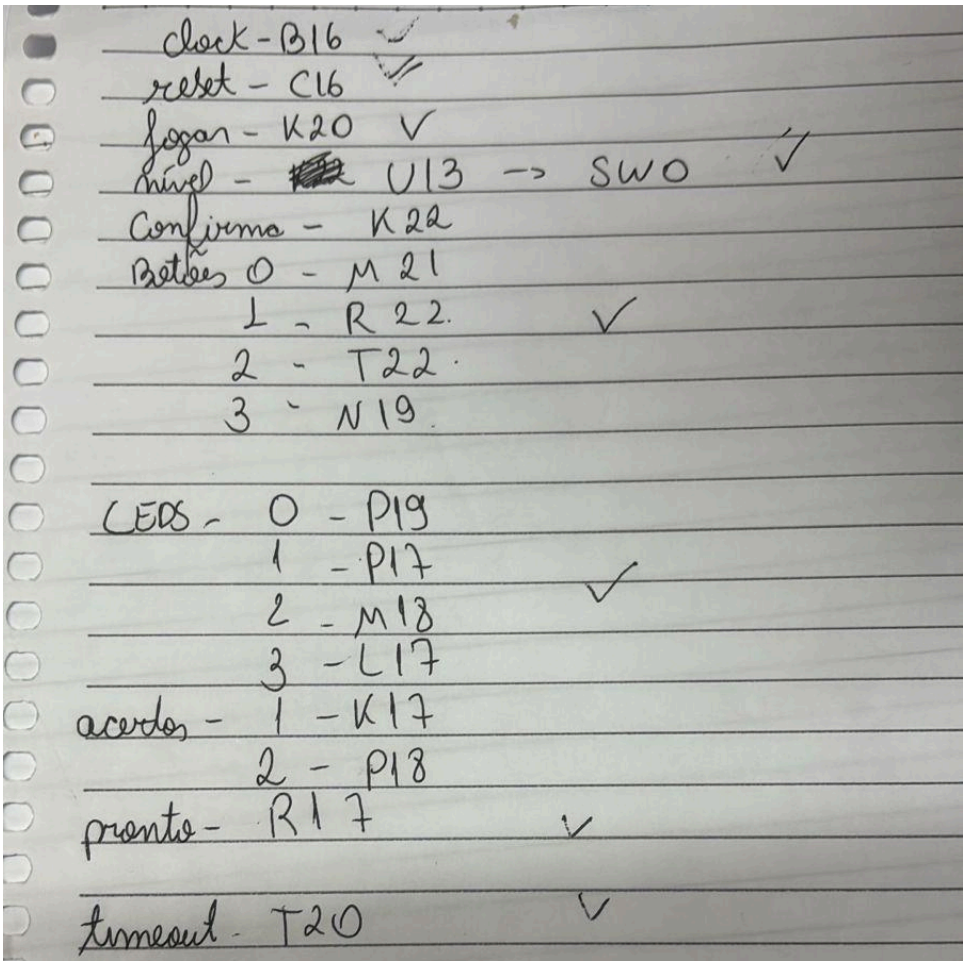

Figura 2.4 - Esquema de Pinagem das Entradas e Saídas principais do NeuroSync.

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: jogadaAtualEQUALSacertoAnterior e acertoAnteriorEQUALSzero.

2.6.2. Modificação 2.2 - Lógica do bigAND

O grupo identificou que o acionamento simultaneo 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 concluida 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. CAPÍTULO 3 - SEMANA 2: IMPLEMENTAÇÃO DA INTERFACE E MANUAL

3.1. Introdução

Nessa semana anterior fizemos o nível fácil (0) do NeuroSync funcionar. Agora vamos torná-lo mais interativo, incorporando estímulos visuais e uma interface (em Verilog) mais incrementada. Se tratam dos requisitos INDICAÇÃO_FEEDBACK (1.3.1.5) e USABILIDADE_INTERFACE (1.3.2.2).

3.2. Descrição do Projeto

O projeto dessa semana contará com uma interface mais amigável ao usuário, com interatividade no acerto de jogadas e descrições nos displays hexadecimais. Para isso, algumas alterações no Fluxo de Dados e na Unidade de Controle serão necessárias. Além disso, também teremos sinais de depuração para o tratamento dos problemas identificados na seção 2.6.4.

3.3. Detalhamento do Projeto

Mais especificamente, a interatividade da interface virá em 2 aspectos principais: novos LEDs que indicarão os acertos dos usuários em cada rodada/sequência e displays de 7 segmentos.

Os LEDs além de indicarem os acertos dos usuários também irão sinalizar quando os jogadores concluírem uma sequência, sendo uma recompensa visual para os acertos dos jogadores.

Já os displays de 7 segmentos irão exibir alguns comandos ao usuário na preparação do jogo e também irão servir para exibir o temporizador do jogo, para um melhor controle de tempo dos jogadores:

- No estado de preparação, exibir: "*nível 0/1*";
- Ao final com timeout (derrota), exibir: "*perdeu*";
- Ao final com acerto de todas as jogadas (vitória), exibir: "*venceu*";
- Nos decorrer do jogo: exibir um *timer regressivo de 300s (5 min)*.

3.3.1. Fluxo de Dados

3.3.1.1. LEDs Acertos

O fluxo de dados conta com elementos adicionais para o tratamento dos LEDs de acerto. Em primeiro lugar, para a exibição dos acertos, há um contador *contadorAcertos* que contabiliza os acertos do jogador e um decodificador *decodificadorAcertos* que transforma a saída em binário do contador para um dado melhor de ser exibido pela interface do NeuroSync.

Além disso, o circuito utilizado para piscar os LEDs de acerto ao fim de cada rodada foi adaptado do Desafio da Experiência 6. Utilizamos 2 timers de 500ms para os LEDs ligados e desligados. Foi também adicionado um contador *contaPiscadas* que limita quantas vezes os LEDs piscam, o limite atual é de 3 vezes. É importante ressaltar que o sinal de enable desse contador passa antes por um detector de borda para evitar contagens excessivas.

3.3.1.2. Displays de 7 Segmentos

Primeiro trataremos dos textos a serem exibidos: *venceu, perdeu e nível 0/1*. Para isso, o grupo criou uma memória (ROM) *displayMem* que recebe um endereço *displayAddr* de 2 bits e a entrada de nível e tem como saída 6 conjuntos de 7 bits

{*HEX5, HEX4, HEX3, HEX2, HEX1 e HEX0*}, um para cada display. A unidade de controle determina qual endereço será enviado para a memória, a depender do estado no qual o jogo se encontra.

Agora para a exibição do timer, utilizamos a saída do *contadorTimeout*, porém encontramos três problemas:

1) A saída do *contadorTimeout* é um sinal em binário, precisamos de um sinal em Binary-Coded Decimal (BCD) para que cada dígito (4 bits) possa ser convertido pelo módulo *hexa7seg* e exibido no display de 7 segmentos.
2) O contador conta de 0 a 300000 uma vez que o clock possui período de 1ms, mas queremos exibir números de 300 a 0.
3) O contador é um contador progressivo (crescente), mas queremos exibir uma contagem regressiva para o usuário.

Para resolver o primeiro problema, convertemos a saída do contador para a base BCD um módulo *binaryTObcd* que implementa o algorítmo de conversão Double Dabble [4], estudado no EP1 da disciplina PCS3225 - Sistemas Digitais 2. As saídas deste módulo são os algarismos (em hexadecimal) da centena, dezena e unidade, que serão convertidos pelos módulos *hex5_converter, hex4_converter e hex3_converter* para saídas adequadas aos displays de 7 segmentos *HEX5, HEX4 e HEX3* respectivamente.

Já o segundo problema foi solucionado com um divisor de clock - *divisorClock* - que pega o clock de 1 kHz e devolve um novo sinal *clockDiv* de 1 Hz. Para nos adaptarmos ao novo clock de 1 Hz, o *contadorTimeout* teve seu limite definido para 300 (uma vez que queremos 300 s).

O terceiro problema, por sua vez, foi solucionado com o módulo *somador*, um somador subtrator que realiza a operação *300 - Saída do contadorTimeout*. Sua saída é a entrada em binário do módulo *binaryTObcd*.

Note que *HEX5, HEX4 e HEX3* possuem duas fontes possíveis: a memória *displayMem* ou a saída dos conversores *hex5_converter, hex4_converter e hex3_converter*. Para decidir qual fonte usar em que momento, cada um dos displays é controlado po um multiplexador, que por sua vez é controlado pelo sinal de controle *displayFromMem*. Os multiplexadores escolhem a saída da memória quando *displayFromMem = 1* e as saídas dos conversores quando *displayFromMem = 0*.

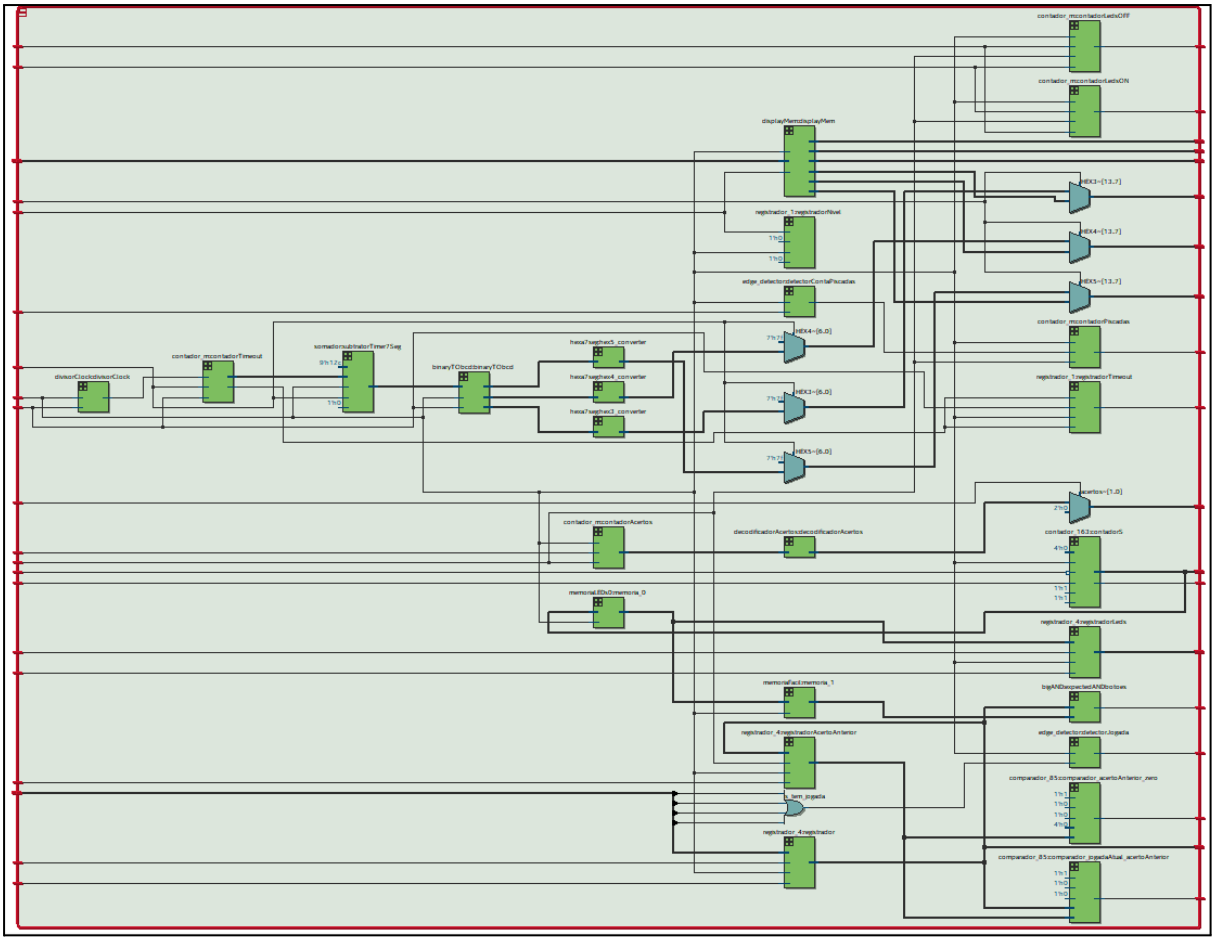

Figura 3.1 - Diagrama de Fluxo de Dados Semana 2

3.3.2. Unidade de Controle

A unidade de controle foi também adaptada do Desafio da Experiência 6, com novos estados *pisca_leds_on* e *pisca_leds_off* para o controle dos LEDs, bem como um novo estado *acende_segundo_acerto* para controlar a exibição de acertos.

Além disso, outros sinais de controle e de condição estão presentes para realizar o controle desses LEDs:

- *contaPiscadas* = enable do módulo *contadorPiscadas*, recebe 1 no estado *pisca_acertos_off*;
- *apagarAcertos* = sinaliza que os LEDs de acertos devem apagar, recebe 1 no estado *pisca_acertos_off*;
- *contaLedsOn e contaLedsOff* = enable dos timers que determinam por quanto temp os leds de acerto acenderão ou ficarão apagados ao piscar. Recebem 1 nos estados de *pisca_acertos_on e pisca_acertos_off* respectivamente.

Já para os displays de 7 segmentos, há uma nova saída *displayAddr* que controla a memória *displayMem*:

- No estado *preparacao - displayAddr = 00 (nível 0/1)*;
    - A própria memória, por uma lógica interna envolvendo o sinal *nivel* determina se o último display (HEX0) exibirá 1 ou 0.

- No estado *final_com_acerto - displayAddr = 01 (venceu)*;
- No estado *estado_timeout - displayAddr = 10 (perdeu)*;
  Além disso, a unidade de controle define *displayFromMem = 1* em qualquer um dos estados citados acima, e *displayFromMem = 0* no restante.

[desenho da máquina de estados]

Figura 3.2 - Diagrama de Transição de Estados em Alto Nível da Semana 2

Eatual	Eprox	Condição de Transição

Tabela 3.1 - Tabela de Transição de Estados da Semana 2

3.3.3. NeuroSync

Uma nova saída de 2 bits *acertos* foi adicionada, bem como novos sinais de depuração para a FPGA: *db_estado, db_sequencia, db_jogada, db_acertouJogada, db_jogadaAtualEQUALSacertoAnterior* e *db_acertoAnteriorEQUALSzero*.

Os sinais *db_acertouJogada, db_jogadaAtualEQUALSacertoAnterior* e *db_acertoAnteriorEQUALSzero* serão cruciais para entendermos o que aconteceu no problema identificado na seção 2.6.4.

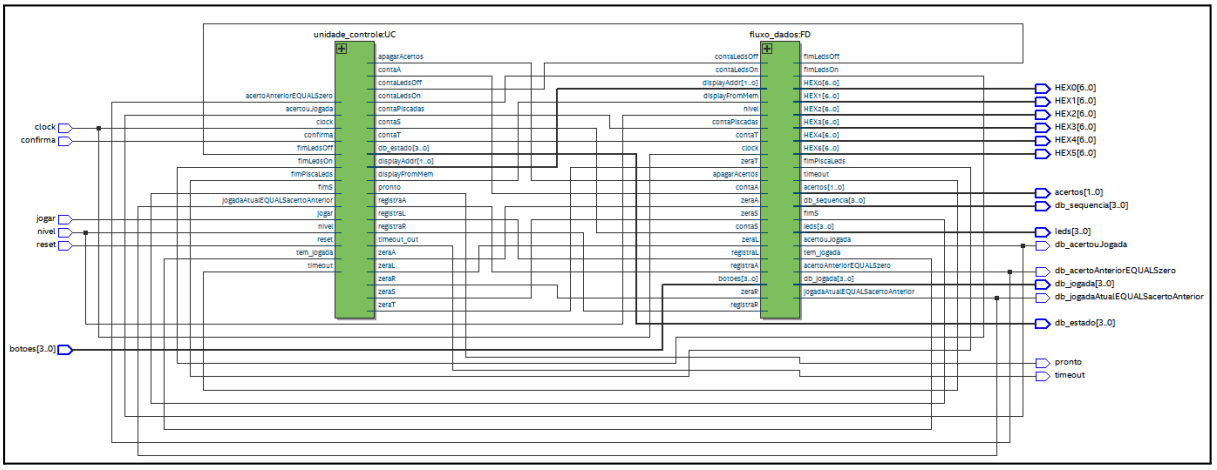

Figura 3.3 - Diagrama RTL View do Módulo NeuroSync da Semana 2

3.3.4. Manual

[Texto sobre o manual]

[Imagem do Manual]

Figura 3.4 - Primeira versão do Manual, Semana 2

3.4. Testes

Repetimos os mesmos planos de teste da seção 2.4 e o funcionamento do jogo se manteve o mesmo, além disso, nas simulações no ModelSim, as saídas dos LEDs acertos e os displays de 7 segmentos estão funcionando como o esperado.

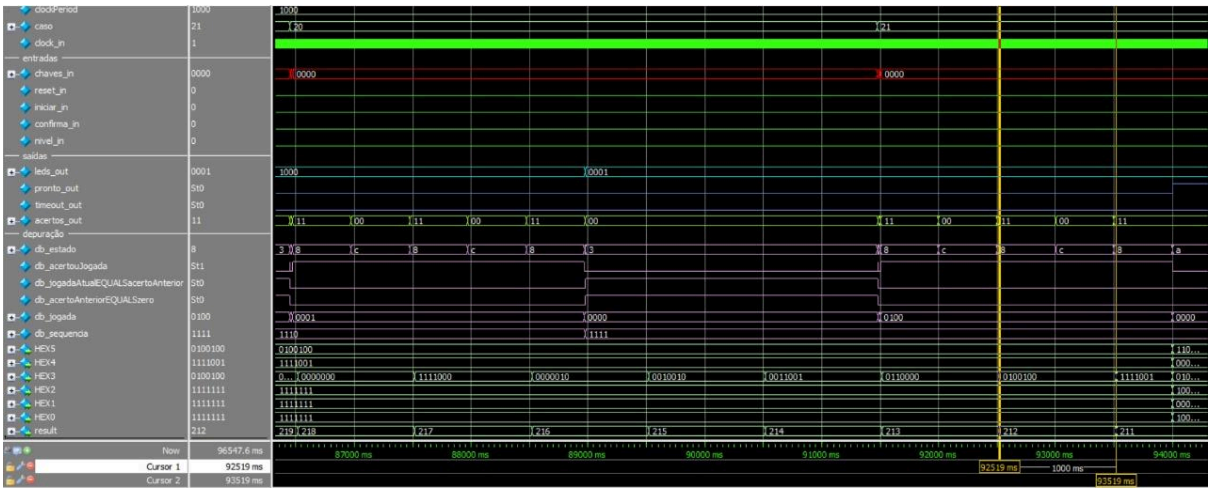

Figura 3.5 - Simulação do ModelSim da tb1.v Semana 2

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.

[4] BRASILFORD, D. Binary to BCD (Double Dabble Algorithm) - Computerphile. Disponível em https://www.youtube.com/watch?v=eXIfZ1yKFIA.