Tempo de Reação: Competição por média

Laboratório de Sistemas Digitais-2023/2024

Dinis Oliveira NMEC:119193

Guilherme Escórcio NMEC:118648

Introdução

O projeto desenvolvido consiste em um sistema para medir e analisar o tempo de reação dos utilizadores. O principal objetivo deste projeto é criar um sistema preciso e eficiente que permita avaliar o tempo de resposta a estímulos visuais. As funcionalidades do sistema incluem a geração de estímulos, a medição do tempo de reação e o registo dos resultados para análise posterior.

Arquitetura

A arquitetura do sistema foi projetada para ser modular e escalável, permitindo a fácil integração de novos componentes e funcionalidades. A estrutura conceptual do sistema é ilustrada na Figura 1.

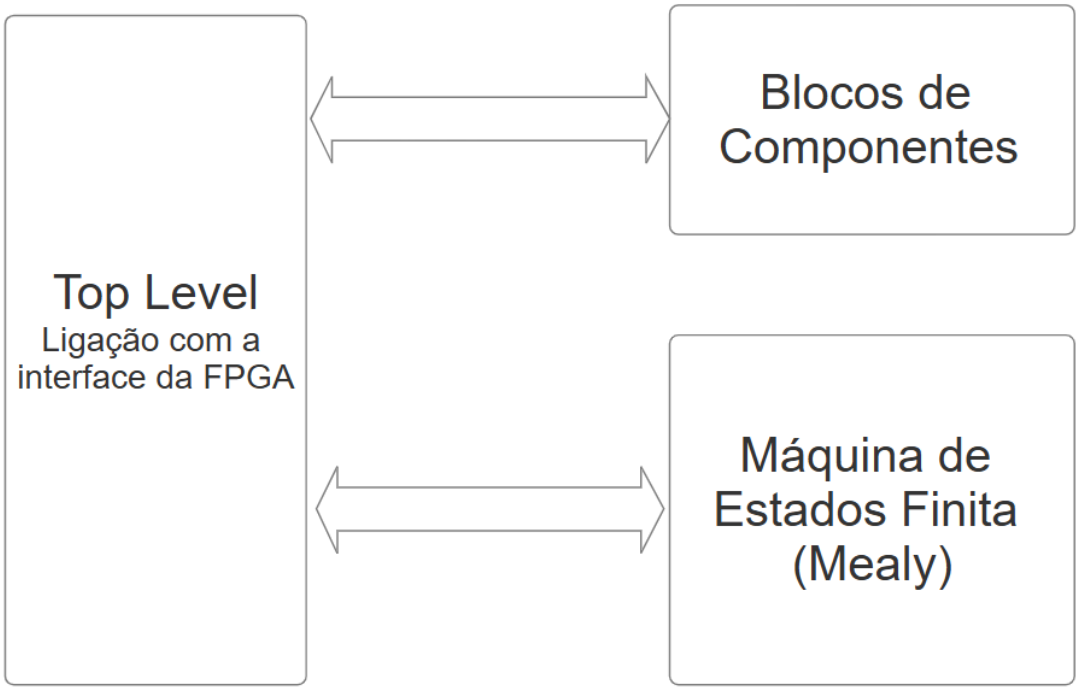

Figura 1: Arquitetura do Sistema

Componentes Modelados:

AccN.vhd - acumulador digital parametrizável
Bin2BCD - um conversor de binário para BCD
Bin7SegDecoderExt - decodificador binário para display de sete segmentos
ClkDivider - divisor de clock
Debouncer - debouncer
FreeRun - contador de execução livre
Nrounds - módulo em VHDL que implementa uma lógica condicional acionada por um sinal de clock
RandomDownsCounter - contador decrescente
ReactTimeFSM - máquina de estados finitos
ReactTimePack - define constantes que representam padrões para a exibição de dígitos em um display de sete segmentos
ReactTimeTop – top level/instanciação dos outros componentes
RoundCount – conta os ciclos de um sinal de relógio quando a entrada ‘up’ está alta
Treact - contador ascendente que é usado para medir o tempo de reação
TimerN – temporizador com um contador que conta até ‘N’ ciclos de relógio antes de sinalizar a conclusão
Blink_gen - gera um sinal de saída de pulso que alterna entre '1' e '0'

Implementação

A implementação do sistema envolveu a integração de diversos componentes de hardware e software. Um aspeto crucial foi a implementação de máquinas de estado finitos para gerenciar as transições entre diferentes estados do sistema.

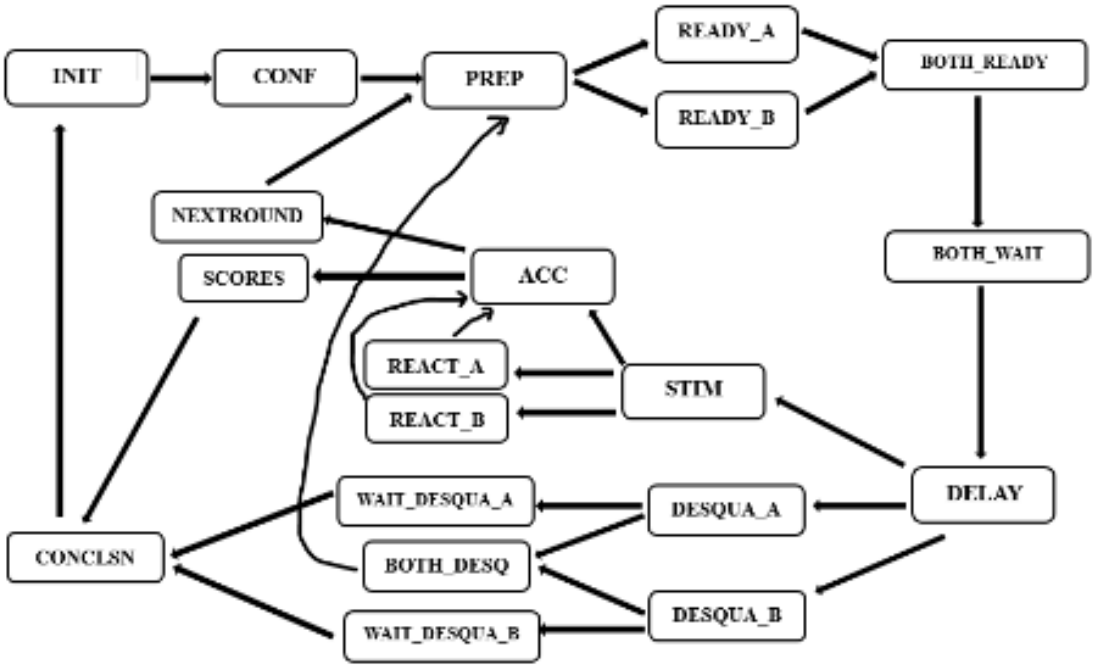

Figura 2: Máquina de Estados Finita

Definição e Transições dos Estados

1. **INIT (Inicialização)**
   - **Ações**: Reseta vários contadores e somas.
   - **Próximo Estado**: Vai para CONF.
   - **Descrição**: Inicializa o sistema resetando contadores e outros componentes necessários.
2. **CONF (Configuração)**
   - **Ações**: Habilita a seleção do número de rodadas.
   - **Próximo Estado**: Vai para PREP quando o jogador B confirma o número de rodadas.
   - **Descrição**: Configura o número de rodadas para o jogo.
3. **PREP (Preparação)**
   - **Ações**: Reseta os contadores de reação e acende todos os LEDs verdes.

- **Próximo Estado**: Vai para READY_A se o jogador A estiver pronto ou READY_B se o jogador B estiver pronto.
- **Descrição**: Prepara o sistema para a próxima rodada, garantindo que todos os contadores sejam resetados.

4. **READY_A (Jogador A Pronto)**
   - **Ações**: Acende um subconjunto de LEDs verdes.
   - **Próximo Estado**: Vai para BOTH_READY se o jogador B estiver pronto.
   - **Descrição**: Indica que o jogador A está pronto e a aguardar o jogador B.
5. **READY_B (Jogador B Pronto)**
   - **Ações**: Acende um subconjunto de LEDs verdes.
   - **Próximo Estado**: Vai para BOTH_READY se o jogador A estiver pronto.
   - **Descrição**: Indica que o jogador B está pronto e a aguardar o jogador A.
6. **BOTH_READY (Ambos Prontos)**
   - **Ações**: Inicia o temporizador de atraso.
   - **Próximo Estado**: Vai para BOTH_WAIT.
   - **Descrição**: Ambos os jogadores estão prontos, e o sistema inicia um atraso antes do estímulo.
7. **BOTH_WAIT (Aguardando Atraso)**
   - **Ações**: Aguarda o término do temporizador de atraso.
   - **Próximo Estado**: Vai para DELAY quando o atraso termina.
   - **Descrição**: Garante que o sistema espere um atraso pré-definido antes de prosseguir.
8. **DELAY (Atraso)**
   - **Ações**: Gere o contador de atraso e verifica reações prematuras.
   - **Próximo Estado**: Vai para STIM se o estímulo for acionado, DESQUA_A se o jogador A reagir prematuramente, ou DESQUA_B se o jogador B reagir prematuramente.
   - **Descrição**: Lida com o período de atraso antes de apresentar o estímulo.
9. **DESQUA_A (Desqualificação Jogador A)**
   - **Ações**: Lida com a reação prematura do jogador A.
   - **Próximo Estado**: Vai para BOTH_DESQ se o jogador B também reagir, ou WAIT_DESQUA_A se o estímulo for acionado.
   - **Descrição**: Gere a situação onde o jogador A reage antes do estímulo.
10. **WAIT_DESQUA_A (Aguardando Desqualificação Jogador A)**
    - **Ações**: Indica que o jogador A está desqualificado e aguarda o temporizador de desqualificação.
    - **Próximo Estado**: Vai para CONCLSN após o término do temporizador.
    - **Descrição**: Aguarda a conclusão do temporizador de desqualificação após a reação prematura do jogador A.
11. **DESQUA_B (Desqualificação Jogador B)**
    - **Ações**: Lida com a reação prematura do jogador B.
    - **Próximo Estado**: Vai para BOTH_DESQ se o jogador A também reagir, ou WAIT_DESQUA_B se o estímulo for acionado.
    - **Descrição**: Gere a situação onde o jogador B reage antes do estímulo.
12. **WAIT_DESQUA_B (Aguardando Desqualificação Jogador B)**
    - **Ações**: Indica que o jogador B está desqualificado e aguarda o temporizador de desqualificação.
    - **Próximo Estado**: Vai para CONCLSN após o término do temporizador.

- **Descrição**: Aguarda a conclusão do temporizador de desqualificação após a reação prematura do jogador B.

13. **BOTH_DESQ (Ambos Desqualificados)**
    - **Ações**: Indica que ambos os jogadores estão desqualificados e aguarda o temporizador de desqualificação.
    - **Próximo Estado**: Vai para PREP após o término do temporizador.
    - **Descrição**: Ambos os jogadores são desqualificados devido a reações prematuras.
14. **STIM (Estimulação)**
    - **Ações**: Inicia os temporizadores de reação para ambos os jogadores.
    - **Próximo Estado**: Vai para ACC se ambos os jogadores reagirem ou se o tempo máximo de reação for atingido, REACT_A se o jogador A reagir primeiro, ou REACT_B se o jogador B reagir primeiro.
    - **Descrição**: Apresenta o estímulo e inicia a cronometragem das reações.
15. **REACT_A (Reação Jogador A)**
    - **Ações**: Continua o temporizador de reação para o jogador B.
    - **Próximo Estado**: Vai para ACC se o jogador B reagir ou se o tempo máximo de reação for atingido.
    - **Descrição**: Gere a situação onde o jogador A reage primeiro.
16. **REACT_B (Reação Jogador B)**
    - **Ações**: Continua o temporizador de reação para o jogador A.
    - **Próximo Estado**: Vai para ACC se o jogador A reagir ou se o tempo máximo de reação for atingido.
    - **Descrição**: Gere a situação onde o jogador B reage primeiro.
17. **ACC (Acumulação)**
    - **Ações**: Resume os resultados da rodada e atualiza os pontos.
    - **Próximo Estado**: Vai para NEXTROUND se houver mais rodadas, caso contrário, vai para SCORES.
    - **Descrição**: Acumula os tempos de reação e determina o resultado da rodada.
18. **NEXTROUND (Próxima Rodada)**
    - **Ações**: Incrementa o contador de rodadas.
    - **Próximo Estado**: Vai para PREP.
    - **Descrição**: Prepara para a próxima rodada incrementando o contador de rodadas.
19. **SCORES (Pontuações)**
    - **Ações**: Exibe as pontuações e gere as rodadas empatadas.
    - **Próximo Estado**: Vai para CONCLSN se todas as rodadas forem concluídas.
    - **Descrição**: Exibe as pontuações e lida com a lógica para determinar os vencedores das rodadas.
20. **CONCLSN (Conclusão)**
    - **Ações**: Exibe os resultados finais e aguarda o temporizador de conclusão.
    - **Próximo Estado**: Vai para INIT após o término do temporizador.
    - **Descrição**: Conclui o jogo e exibe os resultados finais.

Processos adicionais

- **sync_proc**: Lida com transições síncronas de estado baseadas no sinal de clock e na condição de reset.
- **comb_proc**: Gere a lógica combinacional para transições de estado e sinais de saída baseados no estado atual e nas entradas.
- **Processo de Gerenciamento de LEDs**: Controla a exibição dos LEDs baseados no estado atual e nas rodadas restantes

Validação

Durante o desenvolvimento do nosso projeto, enfrentámos diversos desafios relacionados com a simulação e a validação. A principal metodologia de verificação utilizada foi prática e realizada diretamente na placa FPGA.

Para validar a lógica de empate, configurámos a nossa testbench para que ambos os jogadores atingissem o tempo máximo permitido para reação em todas as rodadas. Esta configuração garantiu tempos de reação iguais, resultando em um empate. Este teste específico demonstrou que o sistema reconhece corretamente a situação de empate, confirmando a robustez da nossa implementação.

"Manual do utilizador"

Instalação

1. Abra o Intel Quartus Prime Lite e abra o projeto destinado a ser usado.
2. Importe o master.qsf para o projeto, faça Compile Design e abra o Program Device.
3. Adicione a FPGA no Hardware Setup.

Operação

1. Ligue a FPGA.
2. Escolha o número de rondas usando os interruptores SW[0..3] em binário.
3. Ambos os jogadores têm de estar prontos clicando em KEY[0] e KEY[3].
4. Vai aparecer um estímulo visual e os jogadores têm de clicar nos seus botões o mais rápido possível.
5. O jogador que for mais rápido ganha a ronda e sabe-se, pois, o lado do jogador que ganhou vai ter leds verdes a piscar.
6. A ronda seguinte começa automaticamente assim que os jogadores se encontrarem prontos novamente.

7. O jogo acaba quando as rondas terminarem e o vencedor é o que teve menor tempo de reação médio.

Conclusão

O trabalho realizado cumpriu os objetivos definidos inicialmente, proporcionando um sistema funcional para medir o tempo de reação. Durante o desenvolvimento, adquirimos conhecimentos valiosos sobre o desenvolvimento de um programa em VHDL e as suas aplicações práticas. Em termos de autoavaliação, o projeto pode ser considerado um sucesso, pois conseguimos realizar o programa de modo a funcionar como desejado.

As percentagens de desenvolvimento do projeto são:
Dinis Oliveira – 60%
Guilherme Escórcio – 40%.