UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ
ENGENHARIA ELETRÔNICA

ANDERSON VALENGA GUIMARÃES / ANDREAS ANAEL PEREIRA GOMES/
CARLA CRISTINA GOMES / EDUARDO RAMOS DOMINGUES / ELI TEIXEIRA /
FELIPE AUGUSTO SCHWAB
GEOVANA GOMES MACHADO / GUILHERME SCHNEIDER / GUSTAVO
CORDEIRO LIBEL / IAN WILLIAN BOLFARINI ESCOBAR /
JENIFER ANDRESSA CAMILLO / JOÃO FILIPE MORESCA
JOÃO GABRIEL VERONEZI CRESPIM / JULIANO MENEZES /
LUCAS CORADIN RECH / LUIS GUSTAVO DIAS DE SOUZA / MATEUS KAEL
IZAR
OLIVER PENNER / PEDRO BAZIA NETO / RAFAEL NOGARA

MORSE CODE

CURITIBA
2018

Morse code

Trabalho apresentado à disciplina Lógica Reconfigurável, do curso de Engenharia Eletrônica da Universidade Tecnológica Federal do Paraná – UTFPR, como requisito parcial para a obtenção de nota semestral.

Orientador: Prof. Ricardo Pedroni

CURITIBA
2018

Sumário

1. INTRODUÇÃO

Este projeto trata-se do desenvolvimento de um transmissor e receptor de código morse que será desenvolvido com base nos conhecimentos desenvolvidos na disciplina de lógica reconfigurável e alguns conhecimentos adicionais, adquiridos ao longo do curso.

1.1. CÓDIGO MORSE

O código morse é bastante utilizado hoje em dia, e bem conhecido. Samuel Morse foi o inventor deste código. Intrigado com conceitos de eletromagnetismo, Morse se interessou por um instrumento chamado telegrafo. Através disto, desenvolveu um telegrafo elétrico, onde a base de comunicação era o código desenvolvido por ele.
Se trata de representar caracteres através de pontos e traços, que correspondem a impulsos elétricos, que resultam em sinais acústicos ou luminosos de certa duração. Para separar as letras, é utilizado um espaço de 3 pontos (ou o correspondente a um traço), entre palavras o espaço é de 5 pontos.
A seguir a representação de Letras, Números e pontuação com o código.

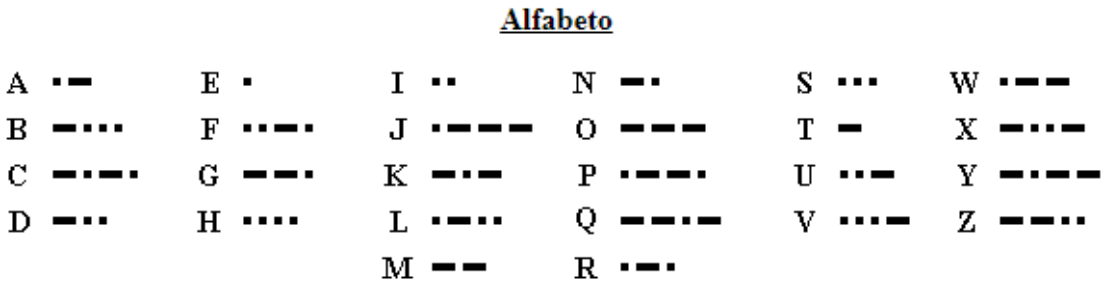

Figura 1 – Alfabeto em código morse

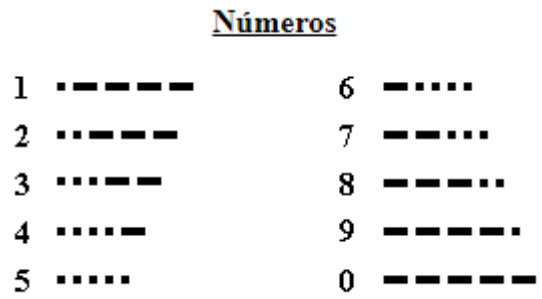

Figura 2 – Numerários em código morse

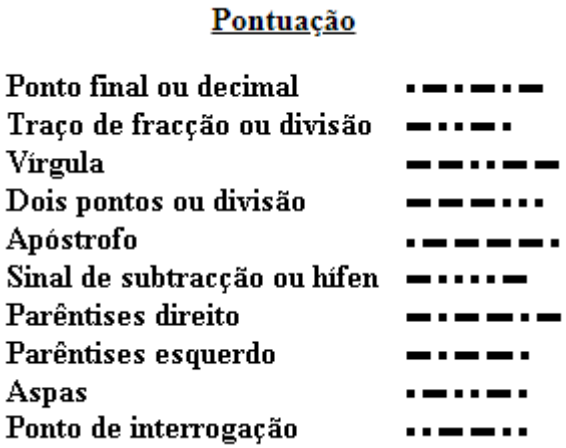

Figura 3 – Pontuação em código morse

1.2. FPGA E VHDL

Uma HDL é uma Hardware Definition Language, ou seja, não é uma linguagem de programação. Usamos a linguagem VHDL para especificar circuitos digitais. É uma linguagem bastante complexa.

A linguagem foi desenvolvida por uma agencia de pesquisa em projetos de defesa norte-americanos, chamada DARPA.

Para trabalhar com VHDL é necessário um editor de código, compilador e um gerador de formas de ondas. Para o presente trabalho foi utilizado o software Quartus II na versão 13.0, disponibilizado nos computadores da universidade e em versões para estudantes nos computadores particulares.

O VHDL possui um vocabulário básico que foi utilizado neste trabalho, o qual será descrito a seguir: Entidade (Entity), Portas (Port), Arquitetura (Architecture), Sinal (Signal), Configuração (Configuration), Pacote (Package), Driver, Bus, Atributos (attribute), Genéricos (generic), Processos (process).

Será utilizado para desenvolvimento deste projeto uma FPGA (Field Programmable Gate Array), que nada mais é que uma microcontrolador reconfigurável, onde possibilita a programação de lógica. O FPGA foi desenvolvido em 1984 pela Xilins para fins militares. Estes dispositivos são baseados ao redor de uma matriz de CLBs, possuindo interconexões programáveis. Neste trabalho foi utilizada uma FPGA da Altera Deo-nano que é um kit didático criado pela altera para incentivo ao conhecimento na área.

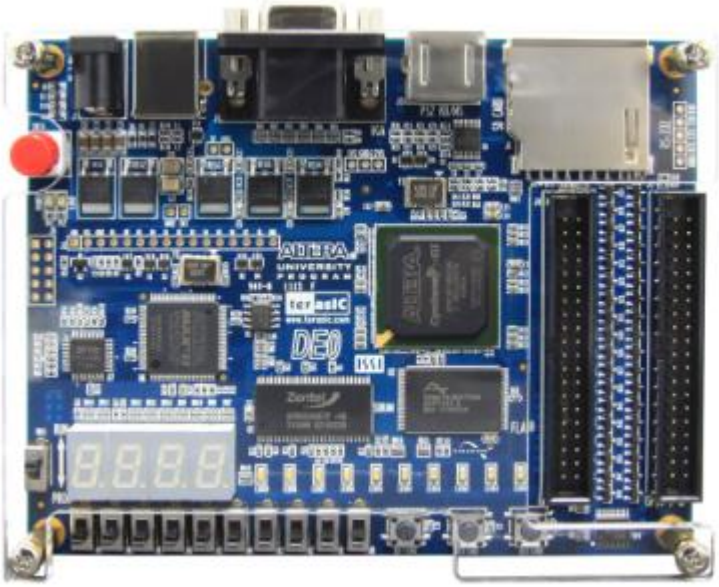

Figura 4 – FPGA

1.3 ESPECIFICAÇÕES

- O controle de entrada de caractere deve ser capaz de inserir caracteres através do padrão Morse (ex: caractere 'S');
- Ter um controle de entrada capaz de inserir caracteres através do padrão morse.
- Exibir cada caractere inserido, na tela, acumulando os caracteres até o limite máximo;

- O sistema deve emitir um som de erro, indicando que não pode receber novos caracteres.
- O sistema deve ser capaz de transmitir os caracteres presentes no buffer para outro sistema compatível, utilizando também código Morse;
- Deve respeitar o seguinte protocolo de comunicação:
    - O sistema deve apenas transmitir se existir outro para receber;
    - O sistema é capaz de transmitir os caracteres presentes no buffer para outro sistema compatível, utilizando também código Morse;
    - O sistema é capaz de transmitir os caracteres presentes no buffer para outro sistema compatível, utilizando também código Morse;

1.4 OBJETIVOS

1.4.1 Objetivo Geral

Desenvolver um equipamento capaz de transmitir e receber código Morse.

1.4.2 Objetivos Específicos

- Desenvolver o código da comunicação VGA
- Desenvolver o código da transmissão e recepção de dados
- Desenvolver o código do looping principal
- Especificar o controle de entrada externo, sinal sonoro e luminoso do sistema

3. MATERIAIS

- FPGA (EP3C16F484C6);
- Display VGA;
- Botão ligado à placa;
- Chave;
- Buzzer;
- Led;

4. METODOLOGIA

A metodologia usada foi de dividir o projeto em partes menores, dividindo a equipe para que cada subequipe trabalhasse em uma parte. Com isto, tivemos as partes de divisão conforme os itens a seguir.

O projeto foi salvo no seguinte endereço de guiHub: https://github.com/cordeirolibel/morse_code_fpga/tree/master/VGA

O projeto seguiu como base o seguinte diagrama:

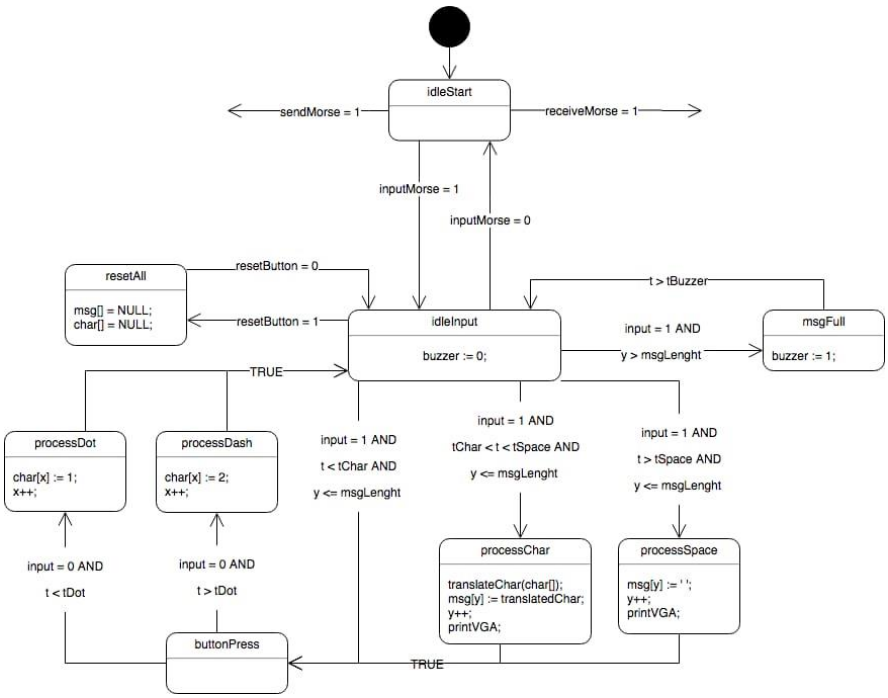

Figura 5 – Diagrama UML

4.1 VGA

Os dados transmitidos via código morse são mostrados na tela do computador via display VGA, existente na placa FPGA utilizada na disciplina. Abaixo, mostramos o exemplo de como os caracteres são exibidos na tela.

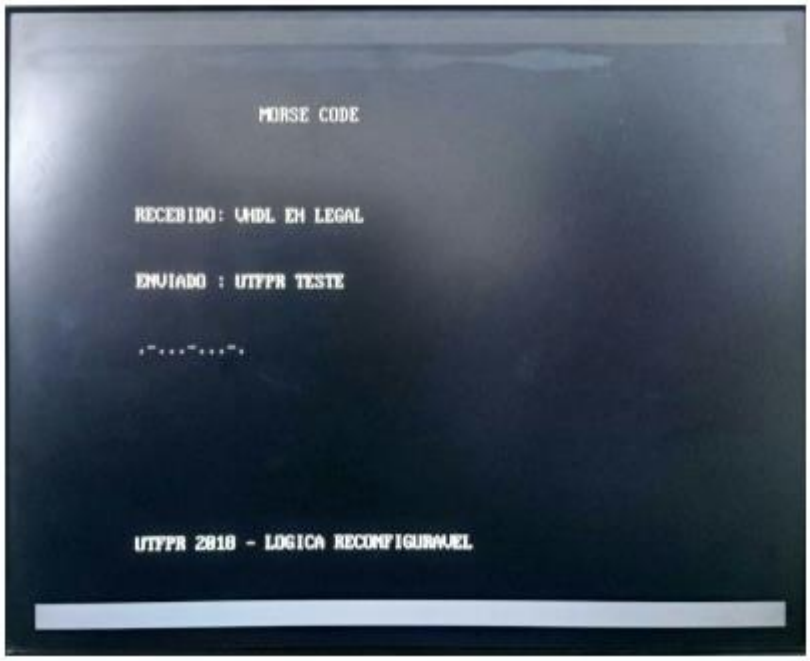

Figura 6 – Comunicação VGA

Basicamente, o código VGA possui nove arquivos (um principal mais oito secundários) que o compõe:

- BlockRamArbiter: Este código tem como objetivo pegar a base de dados do arquivo fontROM (arquivo que possui o conjunto de caracteres disponíveis para serem escritos no display VGA) e fazer o armazenamento desses caracteres em um array de entrada ou saída. É utilizada máquina de estados na parte de arquitetura deste código pois estes são bem definidos e finitos.
- FQDIVIDER: Tem como objetivo fazer a divisão da frequência de saída. O clock de entrada é de 50MHz (CLK_IN), e na saída, temos um clock de 25MHz (CLK_OUT). Faz-se necessário fazer essa divisão para que ocorra uma conexão adequada entre o VGA e o clock de reset da placa FPGA.
- VGA (arquivo principal): É o arquivo principal desta parte do projeto. O funcionamento se resume basicamente a receber três strings (MSG_RECEBIDA, MSG_ENVIADA e MSG_EXTRA) e exibir na tela.

- VGA_LIBRERIA: Inicializa as constantes que definem o tamanho do display VGA e os limites de escrita na tela.
- VGA_SYNC: É um processo que faz a configuração do display VGA para definir em qual local da tela serão escritas as mensagens enviadas e recebidas. Delimita o tamanho máximo do display, de forma que os caracteres escritos neste não ultrapassem o limite determinado.
- basicBlockRam: Este processo lê e escreve dados na memória RAM da placa FPGA.
- commonPack: Ele é um template de arquivos, o qual define tipos suplementares, subtipos, constantes e funções que serão usadas pelo arquivo principal: VGA.
- fontROM: Este arquivo determina a aparência dos caracteres mostrados na tela. Ele contém o padrão de pixels que devem ser mostrados na tela quando um caracter em particular precisa ser mostrado. Neste projeto, utiliza-se uma tela de 640x480. Cada caracter é representado por 8 pixels horizontais e 16 verticais, o que significa que a tela vai poder escrever, no máximo, 80 caracteres por linha (640/8 = 80) e 30 caracteres por coluna (480/16 = 30).
- text_line: Tem como objetivo escrever os dados armazenados nos arrays de saída do código BlockRamArbiter (descrito acima) e mostrá-los na tela. Em sua arquitetura, também possui a descrição de uma máquina de estados.

4.2 GERAÇÃO DE DADOS

Nesta seção o código morse em si é gerado através do arquivo principal chamado APS. Para isto, utiliza-se o template da máquina de estados temporizada para que os caracteres ponto (.) e traço (-) fossem gerados. Além destes, também foram consideradas outras constantes com tempo pré-determinado para compor o código. Como determinado em sala de aula, o tempo padrão para a geração dos dados e constantes é:

T = 3s – constante de tempo para caractere (tChar)

T = 5s – constante de tempo para espaço entre palavras (tSpace)

T = 1s – constante de tempo para ponto (.) (tDot)

T= 3s – constante de tempo para traço (-) (tDash)

T = 5s – constante de tempo para animação do buzzer (tBuzzer)

T = 10s – constante de tempo máximo de espera (tmax)

4.3 TRANSMISSÃO/RECEPÇÃO DE DADOS E LOOPING

Nesta parte é onde ocorre a transmissão e recepção dos dados. Assim como a geração do código morse, este código também utiliza a máquina de estados temporizada em sua arquitetura.

No código foi definida a temporização, conforme citado no item 4.2, e então foi aplicado a cada etapa da maquina de estados desenvolvida.

5. EQUIPE DE DESENVOLVIMENTO

Para a realização deste trabalho, foi necessário dividir a grande equipe em equipes menores, de forma que cada uma ficasse responsável por uma parte do projeto. Desta forma, as subequipes definidas foram:

- Documentação: Carla Gomes e Jeniffer Andressa
- VGA – Andreas Pereira, Gustavo Libel, Eli Teixeira
- Geração de dados – Pedro Bazia, Ian Escobar, Guilherme Schneider, Lucas Rech, Juliano Menezes, Luis de Souza, Rafael Nogara e Geovana Machado

- Transmissão/Recepção de dados e looping – Mateus Izar, Felipe Schwab, Anderson Valenga, João Gabriel, João Moresca, Eduardo Domingues e Oliver Penner

Para a escolha da quantidade de pessoas de cada equipe, levou-se em consideração o nível de dificuldade do assunto, portanto, as maiores equipes ficaram responsáveis pelos assuntos mais difíceis. Já para a escolha dos integrantes de cada equipe, levou-se em conta o nível de conhecimento da disciplina, a afinidade com o assunto e a disponibilidade de tempo dos integrantes. Desta forma, as subequipes apresentadas acima se tornaram a melhor opção na visão da equipe como um todo.

Abaixo, mostramos a reunião de toda equipe e das subequipes montadas:

Figura 7 – Primeira reunião da equipe – definição das subequipes (21/11/2018)

Figura 8 – Equipe desenvolvimento transmissão e recepção de dados (Reunião 23/11/2018)

Figura 9 – Equipe desenvolvimento geração de dados (Reunião 27/11/2018)

Figura 10 - Equipe desenvolvimento geração de dados (Reunião 27/11/2018)

Figura 11 - Equipe desenvolvimento VGA (Reunião 28/11/2018)

Figura 12 - Equipe desenvolvimento geração de dados (Reunião 28/11/2018)

Figura 13 - Equipe desenvolvimento transmissão e recepção de dados (Reunião 28/11/2018)

Figura 14 - Equipes desenvolvimento transmissão e recepção de dados, geração de código (Reunião 11/12/2018)

Figura 15 – Finalizando o trabalho (12/12/2018)

6. CONSIDERAÇÕES FINAIS

Tendo em vista os objetivos citados, este projeto atingiu parcialmente os mesmos. Devido a algumas complicações na junção das partes do projeto, não foi possível obter o resultado esperado. Porém, em grande parte do projeto foi possível executar e atingir a funcionalidade desejada.

Foi possível aplicar o conhecimento adquirido na disciplina, e integrar com outros conhecimentos adquiridos ao longo do curso, até o momento.

Aplicando o que foi aprendido em sala, ocorreu a divisão do trabalho para quebrar em problemas menores, com isto, cada equipe ficou responsável de desenvolver parte do projeto, para ao final, reunir cada parte e então compor o projeto como um todo. Com cada parte funcionando corretamente, partiu-se para a integração das partes, onde não foi possível atingir completamente o objetivo, apesar de grande parte ter funcionado adequadamente.

Com este projeto também foi possível aprender mais sobre o código morse, que foi bastante interessante.

REFERÊNCIAS

PEDRONI, Volnei A. **Eletrônica digital moderna e VHDL**. Rio de Janeiro, RJ: Elsevier, 2010. 619 p. ISBN 9788535234657