UNIVERSIDADE FEDERAL DO RIO GRANDE DO NORTE
CENTRO DE TECNOLOGIA
DEPARTAMENTO DE ENGENHARIA ELÉTRICA

Relatório 7 – Máquina de Estados em VHDL I

Disciplina: ELE0518 – Laboratório de Sistemas Digitais
Alunos: Jefferson H. Silva
Kathleen N. D. Rego
Data: 20/05/2019

1. Introdução

O presente relatório consiste nos relatos e observações acerca da aula laboratorial 7 do curso de *Laboratório de Sistemas Digitais*, executada no dia 20 do mês de maio. Nesta aula, basicamente, foi designada a projeção de uma máquina de estados (MdE), utilizando a linguagem de descrição de hardware VHDL, a partir do software *Quartus*. Este sistema era, basicamente, composto por 2 entradas e 5 saídas.

Além da projeção, também foi realizada a simulação em um FPGA da *Altera* para atestar o funcionamento do sistema. Os principais objetivos da aula estavam focados no reconhecimento da nova linguagem, VHDL, introduzindo os alunos ao primeiro projeto com a linguagem.

2. Referencial teórico

1. VHDL

VHDL é uma linguagem de descrição de hardwares para circuitos integrados de altíssima velocidade. O conceito de linguagem de programação é um tanto quanto abstrato, pois em um senso mais comum a linguagem de programação é aquela capaz de criar um programa - sequência de comandos que instruem e guiam a execução em hardware para efetuar uma determinada tarefa. Nesta definição, VHDL não é uma linguagem de programação, pois o resultado de um código VHDL não é um programa, mas um circuito eletrônico - na verdade um mapeamento de rotas que definirão o circuito quando gravado em um circuito integrado tal como ASIC ou FPGA.

É muito comum ver pessoas iniciando no VHDL escrevendo códigos da forma que escrevem programas em C/Python/etc e isso causa muita confusão. Enquanto nessas linguagens você sabe que o código é sequencial, um comando executado após o outro, no VHDL ocorre tudo de forma paralela, pois como trata-se de um circuito eletrônico, você não está trabalhando com valores armazenados em uma memória, mas sim de sinais elétricos.

Uma das aplicações mais comuns e simples da linguagem é no estudo de circuitos lógicos e álgebra booleana. Muitos livros da área utilizam exemplos em VHDL e muitas universidades utilizam a linguagem como ferramenta no estudo inicial de sistemas digitais. Sua aplicação em si não tem como ser descrita, varia de acordo com a necessidade e criatividade de cada um, pois qualquer circuito digital pode ser implementado utilizando VHDL. Pode ir de uma simples porta lógica, até um modem para transmissões em altas velocidades através de fibra ótica ou mais complexo que isso. A título de exemplo, é possível recriar

microcontroladores, permitindo até que você o programe utilizando outras linguagens, como C ou Assembly. É possível criar drivers de vídeo do zero, drivers de áudio, memórias etc. Basicamente qualquer circuito digital. O circuito gerado em VHDL que faz o cálculo da Transformada de Fourier, por exemplo, é tão grande que é impossível representá-lo de forma visível, como mostrado na Figura 1.

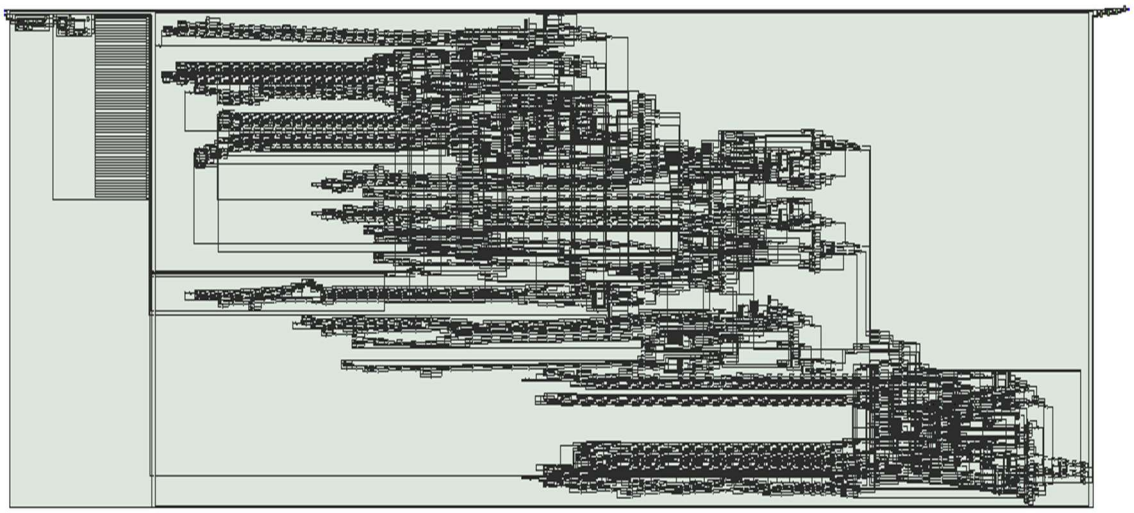

Figura 1. Cálculo da Transformada de Fourier em VHDL.

2. FPGA

FPGA é um chip reprogramável a nível de portas lógicas, ou seja, um hardware reconfigurável, o qual têm as suas funcionalidades definidas exclusivamente pelos usuários e não pelos fabricantes. Na Figura 2, temos um exemplo de FPGA da marca Altera.

Uma de suas inúmeras vantagens é o fato de que não é necessário desenvolver todo o circuito para testes feitos em uma placa ASIC. Assim, com o uso do FPGA, é possível fazer todos os testes necessários prevendo erros mínimos, diminuindo os gastos. Entretanto, seu custo é bastante elevado para a utilização em projetos pequenos. Ademais, seu desempenho entra em desvantagem em comparação com placas ASIC pois geralmente, o circuito projetado não é o esperado, necessitando de versões mais eficientes de FPGA e de projetistas com grandes habilidades na construção do circuito.

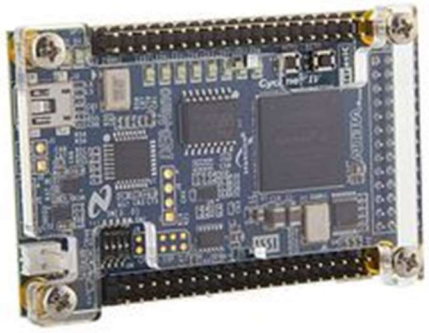

Figura 2. Exemplo de FPGA

3. Metodologia

Esta aula laboratorial foi fundamentada em uma problemática dada pelo docente da disciplina. Basicamente, esta consistia em uma máquina que possuía 5 saídas, todas elas permaneciam desligadas até que uma entrada *H* era acionada. Enquanto H continuasse ativada, as saídas deviam ter o seguinte comportamento: 10000 → 01000 → 00100 → 00010 → 00001, e retornava para o estado inicial. Caso H fosse desacionada ao longo do processo, no próximo estado, o sistema também retornava para o estado inicial.

Além disso, o sistema devia contar quantos ciclos completos dava, pois, ao completar 3 ciclos completos, as saídas deviam ficar todas ligadas até que H fosse desacionada, quando o sistema voltava ao estado inicial e zerava a contagem dos ciclos. Também existia uma entrada de reset a qual servia, basicamente, para forçar o sistema para o estado inicial e zerar a contagem dos ciclos.

A partir disso, definiu-se os estados conforme ilustra a Tabela 1 e se construiu um diagrama de transição de estados para facilitar o entendimento do funcionamento da máquina, ilustrado na Figura 2 de forma simplificada, suprimindo a atuação da entrada de reset, a qual tem prioridade, mandando a máquina para o estado inicial (E0) e zerando o contador de ciclos (count).

Tabela 1. Definição dos Estados com suas respectivas saídas.

Estado	Saídas
E0	00000
E1	10000
E2	01000
E3	00100
E4	00010
E5	00001
E6	11111

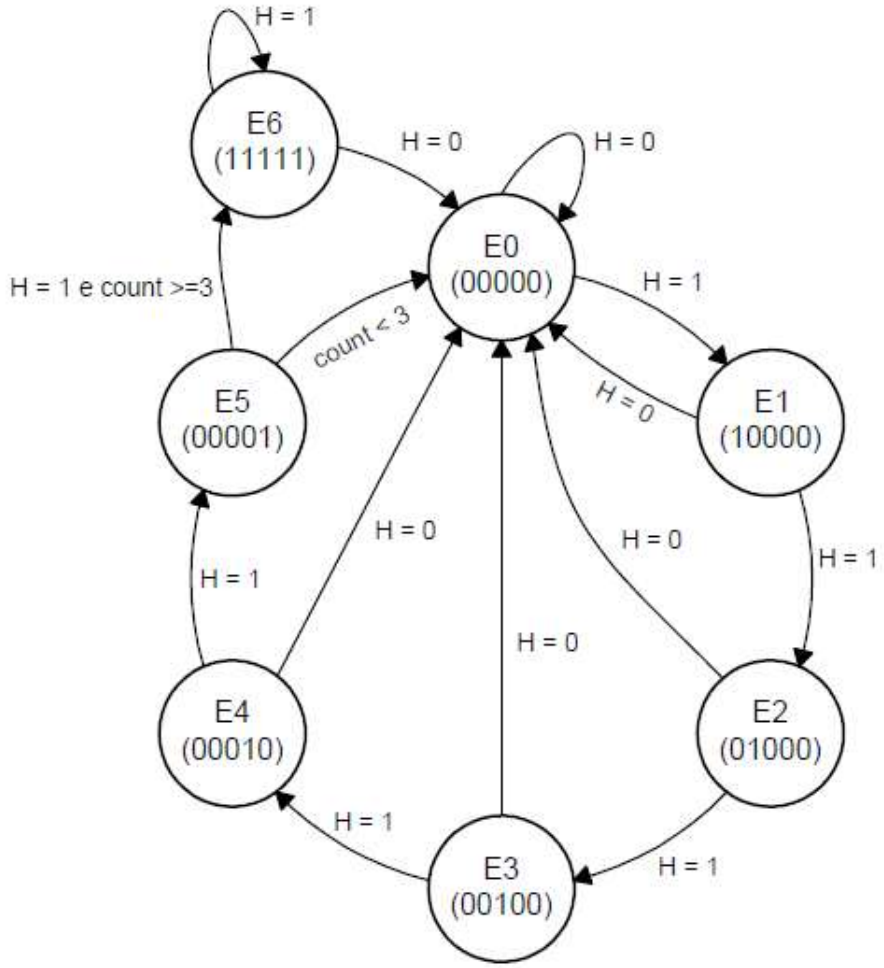

Figura 2. Diagrama de Transição de Estados simplificado.

Então, com o diagrama, a partir da aula ministrada anteriormente pelo docente, realizou-se a implementação da lógica por trás do funcionamento da MdE em VHDL utilizando a abordagem sequencial, aplicando diretivas como if/elsif/else e case/when, valendo-se do software *Quartus* da Intel. Primeiramente, criou-se a entidade com as entradas clk, rst e H e com o vetor de saídas s, com 5 bits. Então, para descrever a arquitetura da entidade, criou-se um novo tipo chamado **estado** que podia assumir os valores (E0, E1, E2, E3, E4, E5 e E6), os estados possíveis da máquina e se criou um signal *state* do tipo estado, que iria armazenar o estado atual da MdE, e outro signal *count* do tipo integer para realizar a contagem dos ciclos completos.

Isto posto, usou-se a diretiva process, sensível a clk, com um if para realizar ações apenas com a transição positiva de clk. Então, com o uso da diretiva case, listou-se todos os casos possíveis para o signal state e suas respectivas ações (definição do próximo estado e atuação das saídas), conforme o diagrama de transição de estados. E, por fim, introduziu-se um if para a atuação da entrada rst, colocando o estado para o E0, saídas 00000 e zerando o contador count.

Posteriormente, realizou-se diversas simulações no próprio software e, por fim, carregou-se o código em uma FPGA da *Altera*, modelo DE2, e se efetuou os testes na placa.

4. Resultados práticos

Durante a implementação da MdE em VHDL, surgiram diversos erros e problemas, principalmente, em decorrência de ser o primeiro contato com a linguagem. Todavia, estes erros serviram para ajudar a compreender um pouco melhor como a linguagem funciona, atingindo um dos objetivos da aula. Após as correções, foi realizada uma simulação pela própria ferramenta *Quartus* e, então obteve-se o resultado dentro do esperado.

Após a simulação ser executada com sucesso, adicionou-se um arquivo enviado pelo docente para que fosse utilizado na implementação com a placa FPGA, aplicando a abordagem hierárquica com a diretiva *port map*. Então, configurou-se o *pinout* para a placa e se efetuou os testes e tudo funcionou dentro do esperado.

5. Conclusão

Portanto, apesar de muitas dificuldades com a adaptação de uma nova linguagem, foi possível construir a máquina de estados em VHDL e ter a introdução ao software *Quartus* e à linguagem VHDL, como objetivado pela atividade. Além disso, o código foi testado várias vezes no simulador até chegar na lógica certa e ser possível simular em um FPGA da Altera, atestando, assim, o funcionamento do sistema.