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

|  |
| --- |
| Relatório – Máquina de estados em VHDL II |
| Disciplina: ELE0518 – Laboratório de Sistemas Digitais  Alunos: Bruno Matias de Sousa Data: 31/05/2019  Levy Gabriel da Silva Galvão  Pedro Henrique de Souza Fonsêca dos Santos |
|  |

1. Introdução

|  |
| --- |
| O VHDL se tornou uma opção simples para se desenvolver circuitos que são complexos demais para se implementar em protoboard. Dessa forma, utilizando-o, se diminui a quantidade de erros e o uso de componentes que podem apresentar defeitos.  O objetivo dessa prática é a implementação de uma máquina de estados um pouco mais complexa em VHDL, uma máquina de doces. |

2. Referencial teórico

|  |
| --- |
| VHDL é uma linguagem de descrição de *hardware* e significa *Very High Speed Integrated Circuit Hardware Description Language*. Por ela, é possível implementar um hardware dentro de um FPGA, por exemplo, por meio de um código mais abstrato, dessa forma, diminuindo a necessidade de testes em protoboard. O único problema desse tipo de implementação é apenas a otimização, pois é a placa que se encarregará da configuração das portas lógicas.  Para a prática, foi proposto a implementação de uma MdE de uma máquina de doces. O funcionamento dela está descrito abaixo:  A máquina de doces possui dois bits de entrada, E1 e E0, que representaram os valores de moeda que entraram, caso ‘00’, nenhuma moeda inserida, caso ‘01’, R$ 0,10 inseridos, caso ‘10’, R$ 0,25 inseridos, e caso ‘11’, R$ 0,50 inseridos. A entrada deve receber a moeda e logo depois voltar para o estado ‘00’ para esperar a próxima moeda. A saída S acumula os valores recebidos. Ao passar de R$ 0,80, que é o valor do doce, é possível apertar um botão C que liberará um doce por uma saída D e subtrairá esse valor de S. A máquina não retorna troco. |

3. Metodologia

|  |
| --- |
| O presente projeto fez uso dos seguintes equipamentos:   * Placa do Kit DE2 da Altera; * Quartus.   Inicialmente foram determinados os estados da máquina. O estado I inicia a máquina, enquanto o estado W é o estado de espera. Os estados A’s são estados de adição de valores. S e D são as saídas da máquina, D é a saída de retirar o doce e S a saída de que subtrai o valor do doce.    **Figura 1** - Tabela dos estados e suas respectivas saídas.  Com os estados definidos, fora construída a máquina de estados de alto nível, como mostra a figura abaixo:    **Figura 2** - Diagrama da máquina de estados para o problema.  Agora será explicado o desenvolvimento do código fonte do projeto geral, destinado à placa. No que diz respeito à entidade escolhida, seu nome foi lab8 e possui 5 entrada e 4 saídas. A entrada e refere-se a um vetor de bits com duas posições e ela irá determinar a presença ou não de moedas e seus respectivos valores. A entrada c refere-se ao botão da máquina que quando pressionado deverá sair o doce. A entrada clk\_in vai receber o clock de 50Mhz da placa DE2 da altera e vai retornar um clock de 1Hz no buffer clk para uso interno no código.  O buffer s representa o valor inteiro resultado da soma de todas as moedas inseridas e restantes. Ele foi definido como buffer, pois apesar dele mostrar o valor de dinheiro restante, ele só faz isso eficientemente na simulação no Quartus. Assim se torna necessária uma outra forma de representar o valor em dinheiro quando o código for simulado na FPGA e essa forma é através das saídas dis\_0, dis\_1, dis\_2. Essas saídas são vetores de bits de 7 posições e vão receber a transformação do valor inteiro s referente ao dinheiro restante, em uma representação em três displays de sete segmentos. A obtenção desses valores serão explicados mais abaixo com o uso de outros componentes fora da entidade principal. O código fonte referente à entidade principal pode ser observado abaixo:   1. **entity** lab8 **is** 2. port( 3. e                   : **in** std\_logic\_vector (1 **downto** 0); 4. c                   : **in** std\_logic; 5. d                   : **out** std\_logic; 6. s  : **buffer integer** range 0 **to** 255; 7. clk\_in, rst  : **in** std\_logic; 8. clk                 : **buffer** std\_logic; 9. dis\_0, dis\_1, dis\_2 : **out** std\_logic\_vector (6 **downto** 0) 10. ); 11. **end**   Para que seja realizada a divisão do clock de 50MHz para 1Hz foi usado um componente, como descrito abaixo. A entrada clkIn receberá o clk\_in do código principal, que por sua vez receberá o clock de 50Mhz e retornará o clock de 1Hz na saída clkOut que passará para o código principal como clk.  Para acessar esse novo clock foi usado um port map. Assim, logo após ao begin da architecture, fora colocado o port map.   1. **component** ClockDiv **is** 2. port( 3. clkIn  : **in** std\_logic; 4. clkOut : **out** std\_logic 5. ); 6. **end** **component**; 7. **begin** 8. clock : ClockDiv **port map**(clk\_in, clk);   Para facilitar o projeto foi criado o tipo state que representa os estados válidos da MdE e mais dois sinais que armazenarão o estado atual (actual\_s) e o próximo estado (next\_s), como pertencentes ao tipo state. Outros 3 sinais foram criados: d2, d1, d0; eles são inteiros que receberão a divisão de um número inteiro de 8 bits, em três outros números de 8 bits. O trecho de código pode ser observado abaixo.   1. **type** state **is** (init, waitc, adder, dispenser, subtract); 2. signal actual\_s, next\_s: state; 3. signal d2, d1, d0: **integer** range 0 **to** 255 := 0;   Para o projeto foi utilizado apenas um processo. De acordo com o trecho de código abaixo, ele é sensível apenas ao clock e ao reset.   1. state\_register: **process** (clk, rst)   Esse processo possui uma estrutura condicional principal na forma de um if que para o caso do rst ser acionado ele executa as devidas ações e para quando for detectado a borda de subida do clock por meio do rising\_edge(clk) ele executar as devidas ações por meio de uma estrutura condicional do tipo case para avaliar qual o estado atual e decidir qual o próximo estado de acordo com o diagrama proposto anteriormente.  A cada estado o valor de s será alterado ou permanecerá o mesmo, porém ele deverá ser apresentado por meio dos displays de sete segmentos. Para que isso seja permitido são necessárias duas transformações: a primeira é pegar o número inteiro referente a s, que uma vez que possui 8 bits seu número de algarismos são no máximo 3, e separar cada um desses algarismos nos sinais d2, d1, d0 posições transformá-lo em 3 outros números, sendo associados, respectivamente do número mais significativo para o menos significativo; a outra transformação é feita do código binário correspondente a cada sinal d2, d1 ou d0 para o código do display de sete segmentos em anodo comum.  Essas duas transformações são feitas por meio de componentes. O componente que realiza a primeira transformação, a de divisão, é o disp\_divider, que está descrito no primeiro componente do código abaixo. O componente irá receber uma entrada inteira n e retornará três inteiros: dis\_0, dis\_1, dis\_2, que são binários referentes a cada um dos algarismos correspondentes. O segundo componente que irá transformar de binário para o código dos displays de sete segmentos é o BCD27, ele possui uma entrada bcd e uma saída seg e ele compara por meio de uma estrutura sequencial qual valor inteiro foi inserido e retorna o código em sete segmentos correspondente,   1. **component** disp\_divider **is** 2. port( 3. n                     : **in** **integer** range 0 **to** 255; 4. dis\_0, dis\_1, dis\_2  : **out** **integer** range 0 **to** 255 5. ); 6. **end** **component**; 8. **component** BCD27 **is** 9. port( 10. bcd : **in** **integer** range 0 **to** 255; 11. seg : **out** std\_logic\_vector (6 **downto** 0) 12. ); 13. **end** **component**;   Voltando para a arquitetura da entidade principal, ao final do processo descrito mais acima, ele deverá executar os devidos port maps para atualizar os valores recebidos pelos displays de acordo com o valor total de moedas. Assim ele executa a seguinte sequência de códigos para utilizar os componentes descritos acima:   1. p\_3dnum : disp\_divider **port map** (s, d0, d1, d2); 2. p\_disp2 : BCD27 **port map** (d2, dis\_2); 3. p\_disp1 : BCD27 **port map** (d1, dis\_1); 4. p\_disp0 : BCD27 **port map** (d0, dis\_0);   Após a devida compilação e simulações do código, suas entradas e saídas foram mapeadas segundo a pinagem da placa DE2 da Altera. Assim, foi também simulado na placa. |

4. Resultados práticos

|  |
| --- |
| Os resultados alcançados nesta prática nos mostrou que a máquina de doces implementado o código em VHDL foi alcançado com perfeição, resultando nas operações desejadas. Primeiramente foi se necessário entender o problema proposto, implementar o diagrama de estados em alto nível a ser realizados, para necessitou que ambas implementar uma lógica em cada estado que auxiliasse a realização da soma do dinheiro, após a soma chegar em um valor de um doce foi possível fazer a retirada por C, decrementado ao display de 7 segmentos.  Foi utilizada a simulação do circuito na placa do FPGA e foi realizado a escolha dos pinos através da tabela disponibilizada anteriormente. Com isso foi possível implementar o circuito proposto de modo satisfatório, com todos os estados e funcionamento da máquina de doces proposta.  Em seguida foram realizadas no Quartus II simulações para se verificar se o código funcionava corretamente. Primeiro foi testado a função soma que incrementa valores na máquina sendo R$ 0,10, R$ 0,25 e R$ 0,50, sendo mostrado na figura abaixo.    **Figura 3** - Simulação com entradas variáveis para as moedas, demonstrando que a função soma funciona perfeitamente (a parte inferior é a sequência da parte superior).  Logo após realização da simulação da soma, foi realizada a simulação da retirada do doce, através da utilização do botão C, m. A simulação pode ser vista logo abaixo.    **Figura 4** - Simulação para a retirada de doces após o total em dinheiro superar os 80 centavos (a parte inferior é a sequência da parte superior).  Por fim foi realizada a simulação do reset que mostra a volta ao estado inicial, ou seja em R$ 0,00.    **Figura 5** - Simulação para o acionamento do botão reset em um momento aleatório, demonstrando que o total da soma zerou (a parte inferior é a sequência da parte superior). |

5. Conclusão

|  |
| --- |
| O resultado do trabalho foi satisfatório e correspondeu a todas as expectativas delimitadas no escopo do projeto, mostrando mais uma vez o quanto é facilitada a implementação de uma máquina de estados utilizando o VHDL.  Além disso, o projeto permitiu entender mais um pouco do funcionamento da sintaxe da linguagem, suas limitações e erros. |

6. Referências Bibliográficas

|  |
| --- |
| ABNT, Associação Brasileira de Normas Técnicas. **NBR 10719 – Apresentação de relatórios técnico-científicos**. Rio de Janeiro: ABNT, Copyright © 1989.  MARCONI, Marina de A. & LAKATOS, Eva M. **Fundamentos de metodologia científica**. 5 ed. Editora Atlas. São Paulo, 2003.  TOCCI, Ronald J. **Digital Systems**: principles and applications. 11 ed. Pearson Education India, 1991.  Slide e anotações da Aula. **Linguagem de Descrição de Hardware**.Professor: Carlos Yuri Ferreira Silva. |