UNIVERSIDADE FEDERAL DO RIO GRANDE DO NORTE
CENTRO DE TECNOLOGIA - CT
DEPARTAMENTO DE ENGENHARIA ELÉTRICA - DEE
BACHARELADO ENGENHARIA MECATRÔNICA

RELATÓRIO SISTEMAS DIGITAIS – FIFO – PROJETO 02

NATAL
2020

UNIVERSIDADE FEDERAL DO RIO GRANDE DO NORTE
CENTRO DE TECNOLOGIA - CT
DEPARTAMENTO DE ENGENHARIA ELÉTRICA - DEE
BACHARELADO ENGENHARIA MECATRÔNICA
SISTEMAS DIGITAIS

ATYSON JAIME DE SOUSA MARTINS

RELATÓRIO SISTEMAS DIGITAIS – FIFO – PROJETO 02

NATAL
2020

Sumário

1. Introdução

Neste relatório estará presente todo o procedimento para escolha e construção de uma FIFO em VHDL utilizando máquina RTL, para ser usado em uma FPGA. Seguiremos a arquitetura desejada pelo professor, apresentada na figura abaixo (imagem 1).

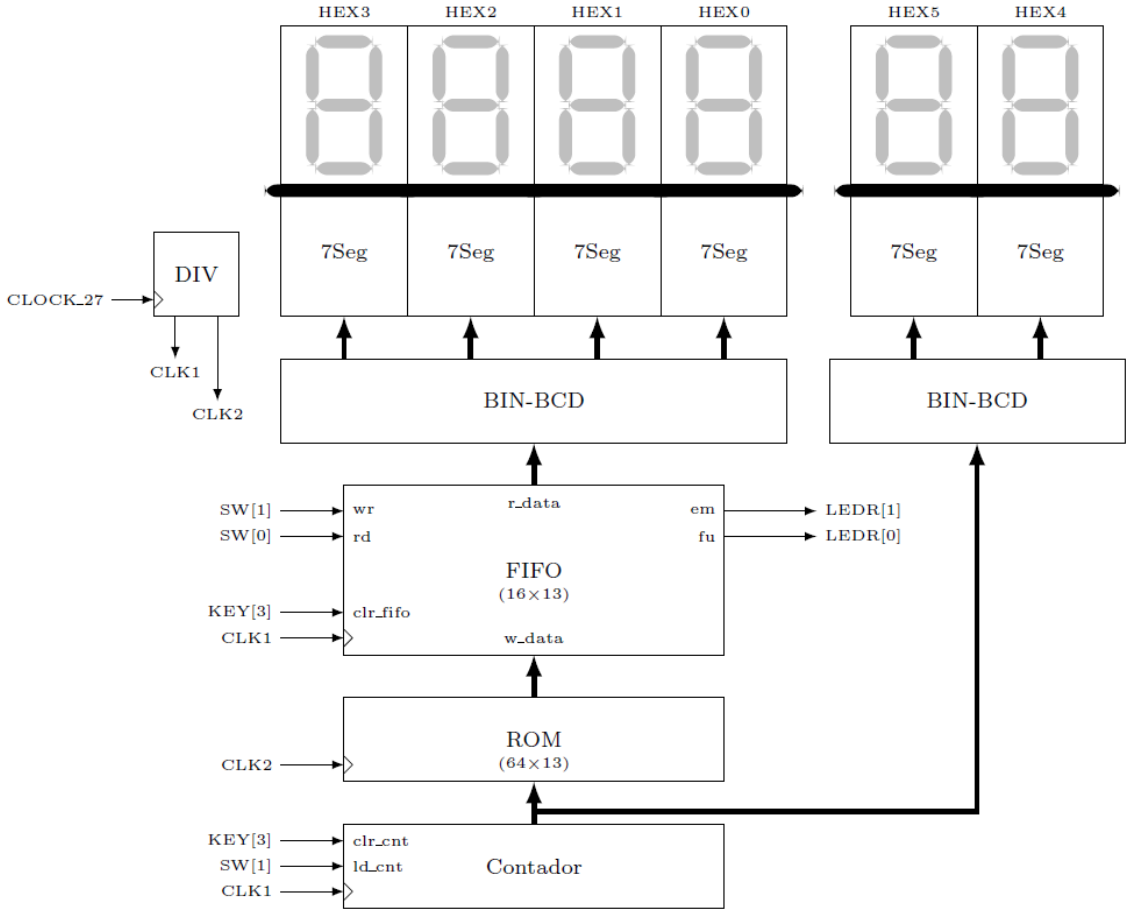

Imagem 1 - Arquitetura desejada para ser implementada

Todavia, antes de apresentar o procedimento, iremos falar sobre o que é uma Máquina RTL (Register Transfer Level); essas máquinas são métodos usados para criar processadores que são a junção de um bloco de controle (onde fica toda a parte que controla o processador) com um bloco operacional (onde fica toda a parte que mexe com dados do processador). Para se projetar uma máquina dessa é preciso seguir alguns passos: Obter a máquina de estados de nível alto; criar o bloco operacional; obter a máquina de estados finito do bloco de controle (FSM).

No seguinte estudo, foi necessária a utilização de outros componentes juntos a máquina RTL para a perfeito funcionamento desejado e adequado da estrutura referida na imagem 1.

2. Desenvolvimento

Para a realização do projeto proposto decidi fazer de forma estruturada, ou seja, destrinchar o projeto em partes e ao final juntar todas elas em um único arquivo. Desse modo, o código fica mais enxuto e mais fácil de ser testado. Assim, seguindo como roteiro a imagem 1, dividi os blocos da seguinte maneira: divisor de clock, projeto FIFO (funcionalmente, bloco de controle e datapath), BIN-BCD com display 7 segmentos, ROM e contador.

2.1. Divisor de Clock

Considerando que a FPGA utilizada para os testes possui em sua circuitaria interna um clock de 27MHz, foi necessário a criação de um bloco no qual reduza esse clock de operação da placa para um clock de operação de 10Hz.

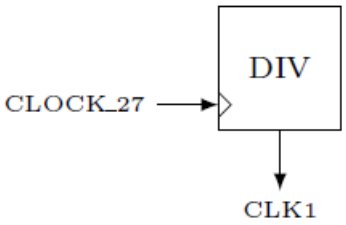

Imagem 2 – Bloco Divisor de Clock

Para a confecção desse, utilizou-se o código disponibilizado pelo professor, o qual a posteriori foi implementado em VHDL e simulado na plataforma.

2.2 – FIFO

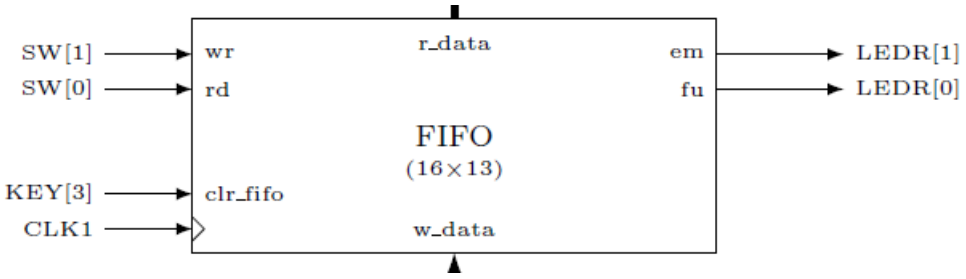

Imagem 3 – Bloco FIFO

A FIFO, do inglês *First-in First-out*, é um tipo de memória que tem como definição o seguinte dilema: o primeiro dado que entra na memória é sempre o primeiro a sair. No projeto proposto, tínhamos que projetar uma FIFO com 16 posições que suportasse uma leitura de 13Bits.

Para tal, tiramos como base o projeto RTL presente no livro Sistemas Digitais – Projeto, Otimização e HDL's pelo autor por Frank Vahid, com uma adicional de um botão de reset na máquina de estados. Em vista disso, ficamos com a seguinte formatação para a máquina de estado de alto nível.

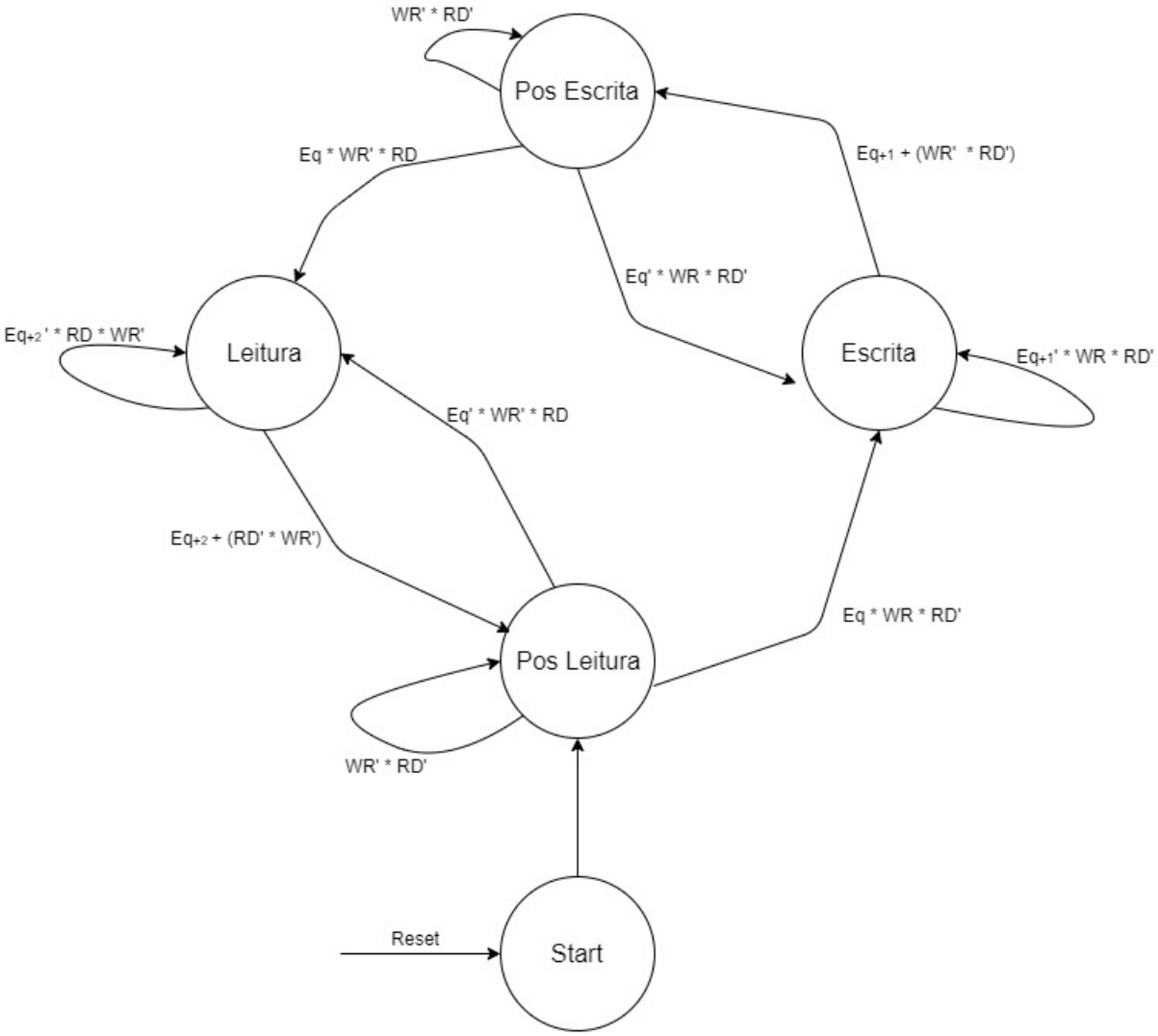

Imagem 4 – Máquina de estado de alto nível

Funcionalmente da máquina se dar da seguinte maneira:

Start: Estado no qual a máquina inicia;

Pós Leitura: Nesse estado, a máquina espera o acionamento do botão de escrita na memória (WR), caso acionado, vai para o estado de Escrita. Como também, se a comparação dos contadores de posição de escrita e leitura forem iguais, acionará um LED avisando que a fila se encontra vazia. Caso esse LED não esteja acionado, e for apertado o botão de leitura (RD) a máquina irá para o estado de leitura.

Escrita: Nesse estado, a máquina mandará um sinal para gravar o dado que está sendo enviado para a memória. Se o botão WR continuar acionado, EQ_{+1} (comparação dos contadores de posições de escrita mais um e leitura. Basicamente, serve para dizer se a próxima posição do contador de escrita encherá a memória FIFO) e RD estiverem barrados a máquina continuará no estado de escrita. Caso EQ_{+1} vinher a se tornar *true* ou WR parar de ser acionado, a máquina irá para o estado de pós escrita.

Pós Escrita: Nesse estado, a máquina espera o acionamento do botão de leitura na memória (RD), caso acionado, vai para o estado de Leitura. Como também, se a comparação dos contadores de posição de escrita e leitura forem iguais, acionará um LED avisando que a fila se encontra cheia. Caso esse LED não esteja acionado, e for apertado o botão de escrita (WR) a máquina irá para o estado de leitura.

Leitura: Nesse estado, a máquina mandará um sinal para ler os dados que estão gravados na memória. Se o botão RD continuar acionado, EQ_{+2} (comparação dos contadores de posições de escrita e leitura mais um. Basicamente, serve para dizer se a próxima posição do contador de leitura esvaziará a memória FIFO) e RD estiverem barrados a máquina continuará no estado de escrita. Caso EQ_{+2} vinher a se tornar *true* ou WR parar de ser acionado, a máquina irá para o estado de pós escrita.

Com esse esquemático pronto, dividimos o projeto RTL em datapath (local onde fica a parte que manipula dados) e bloco de controle (local onde fica a parte que controla a máquina). Para o datapath, foi necessário criar: Um Banco de Registradores no qual, em sua estrutura interna se encontra um decodificador e um multiplexador ambos de 16bits, como também, dezesseis registradores de 13Bits feitos com flip-flop D; Dois contadores de 4bits, que servem para informar ao bloco operacional qual as posições que iram ser escritas ou lidas pelo bloco de controle; Dois somadores de 4bits, que exercem a função de somar a saída dos contadores mais um para ser usado nos comparadores; Três comparadores de 4bits, onde desempenham a função de comparar as posições dadas pelos contadores. Portando, ficando com a seguinte estrutura para o bloco de controle e datapath.

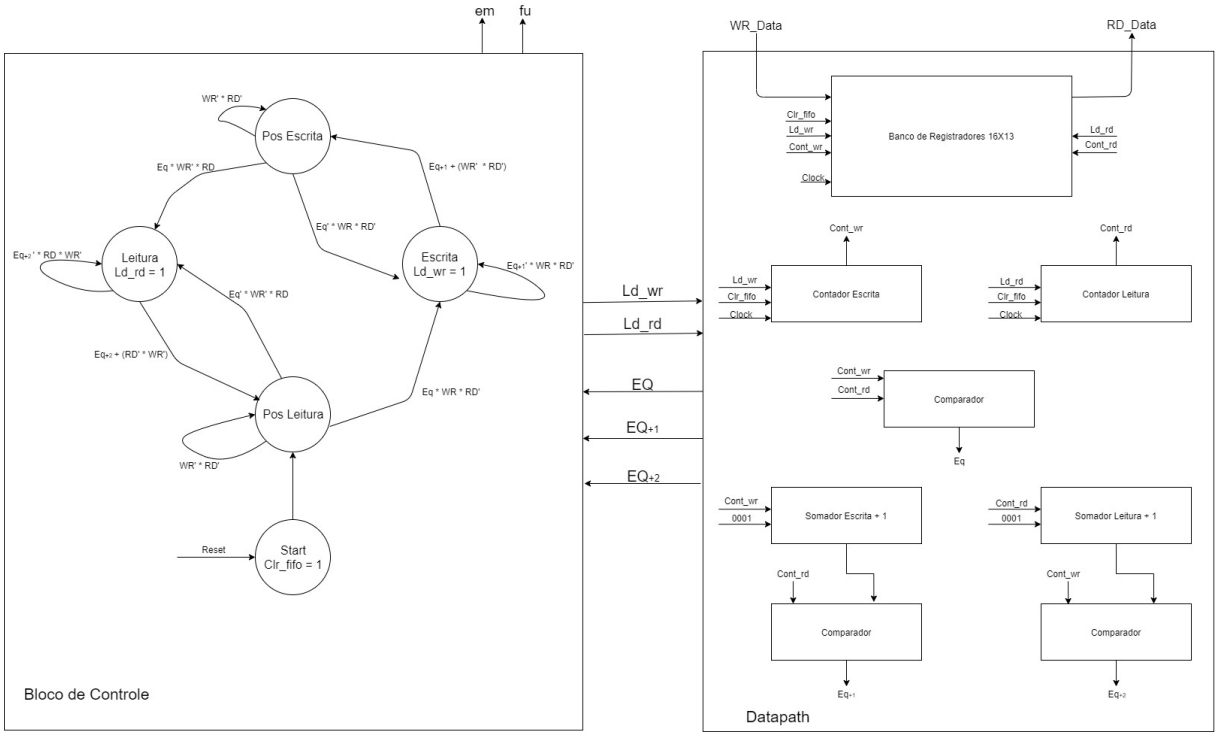

Imagem 5 – Bloco de controle e datapath da máquina de vendas

Após implementação em VHDL, o mesmo foi sumulado na plataforma.

2.3 – BIN-BCD com display7Segmentos

Se olharmos para a imagem 1 novamente vemos que a exigência da arquitetura é que, a saída tanto do contador da ROM, como a saída da memória localizada na FIFO tenham seu resultado mostrado em um display 7 segmentos. Para isso ocorrer, precisou-se primeiramente fazer uma conversão de binário para BCD.

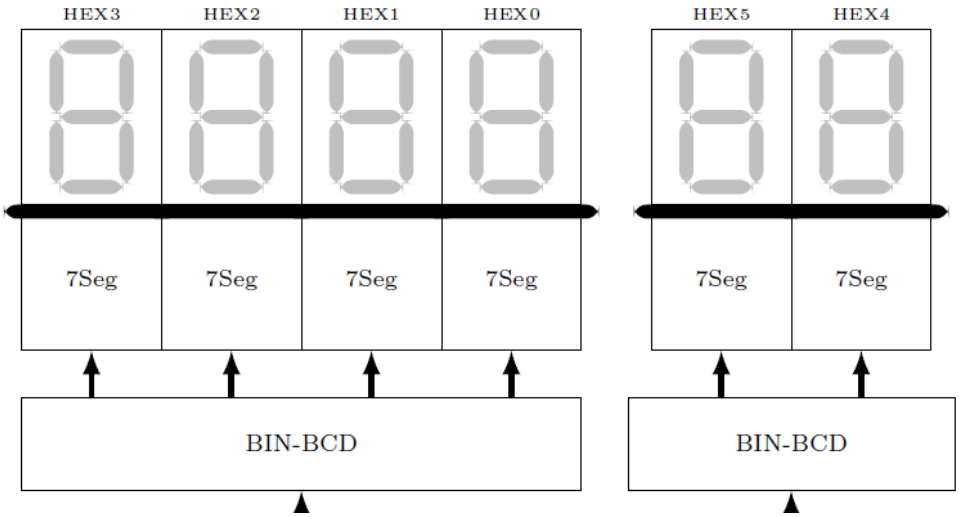

Imagem 6 – Bloco BINBCD – Display7Segmentos

Os códigos utilizados para fazer todas essas partes, tanto para a conversão de binário para BCD, quanto BCD para display 7 segmentos, foram retirados das atividades antes realizadas na matéria de Circuitos Digitais, a qual, é requisito obrigatório para se pagar Sistemas Digitais, com alterações no código de binário para BCD a fim de que ele consiga converte até a casa do milhar.

2.4 – ROM

O bloco de memória vista na arquitetura da imagem 1, funcionará basicamente como um guardado dos valores que serão enviados para a FIFO, retirando a necessidade de o usuário ficar colocando valores. Por isso, que sua saída vai direito para a entrada da FIFO. Visto que, como no projeto só queremos ler os valores que estão salvos nela, a ROM é a melhor escolha. Já que, pela sua definição a ROM (*Read-Only Memory)* é uma memória apenas de leitura.

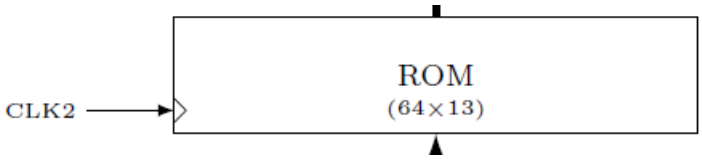

Imagem 7 – Bloco da ROM

Para a implementação da mesma em VHDL na plataforma, utilizou-se do passo a passo em pdf disponibilizado pelo professor por meio da turma virtual.

2.5 – Contador

No projeto, o contador foi empregado para acessar a posição da ROM e assim, pegar o valor que serão usados como valores para a FIFO.

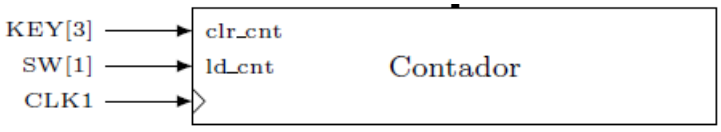

Imagem 8 – Bloco Contador

Dado que o tamanho da ROM é possível ter 64 posições diferentes, foi necessário criar um contador de 6Bits, pois quando fazemos $2^6 = 64$. Internamente, seu funcionamento se dar por flip flop JK juntos, com pequena diferença empregada devido a imposição quista na arquitetura final, no qual, foi a colocação de um load e um clear no contador. Desse modo, o contador só funcionará quando o botão for apertado, ou seja, quando o usuário quiser gravar um novo valor na FIFO. Quando isso ocorrer, o valor que está no contador ditará a posição da ROM no qual terá a quantia X guardada na FIFO. Para sabermos em qual posição a ROM se encontra a saída do contador também vai para dois displays.

3. Resultados Obtidos

Após testes exaustivos em simulador o código foi posteriormente testado em FPGA, utilizando interfaces de saída de dados destinados aos displays da placa, aliado a um código divisor de clock utilizado para reduzir o clock de operação da placa, 27 MHz, para um clock de operação de 10Hz, previamente citado no projeto. Desse modo, começamos com a seguinte interface no FPGA:

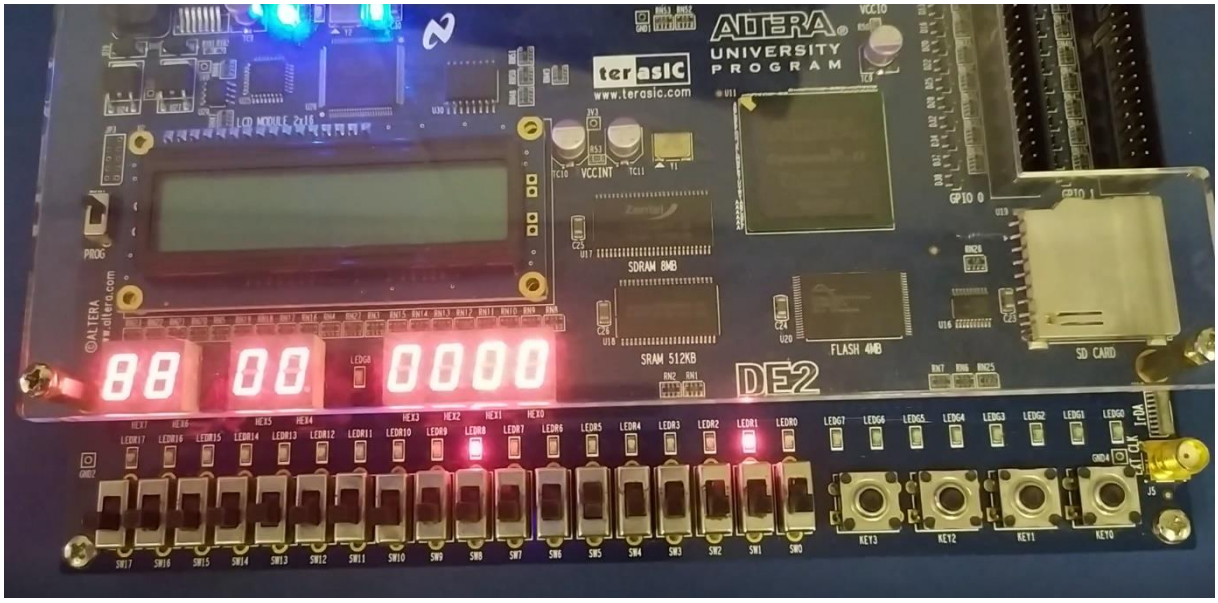

Imagem 9 – momento inicial do teste em FPGA

Como podemos ver na imagem 11, os 2 displays que se encontram no meio (o qual possuem valor 00) mostra a posição da ROM. Já os 4 displays da ponta (o qual possuem valor 0000 inicialmente) mostra o valor da saída da FIFO, ou seja, a ordem dos valores inseridos será mostrada em ordem nesses displays. Os LEDs LEDR10, LEDR9 e LEDR8 foram usados a fins didáticos para se visualizar em qual estado a máquina se encontra.

Nesse primeiro momento, vemos a FIFO completamente vazia, pois o LEDR1 está aceso. Quando levantamos a chave SW1, mandamos o sinal para a máquina para escrever o valor da primeira posição da ROM na fila. Quando isso ocorre, o LEDR1 se apaga, informando que a fila não está mais vazia, como também, os displays do meio mostram a nova posição que o contador da ROM se encontra.

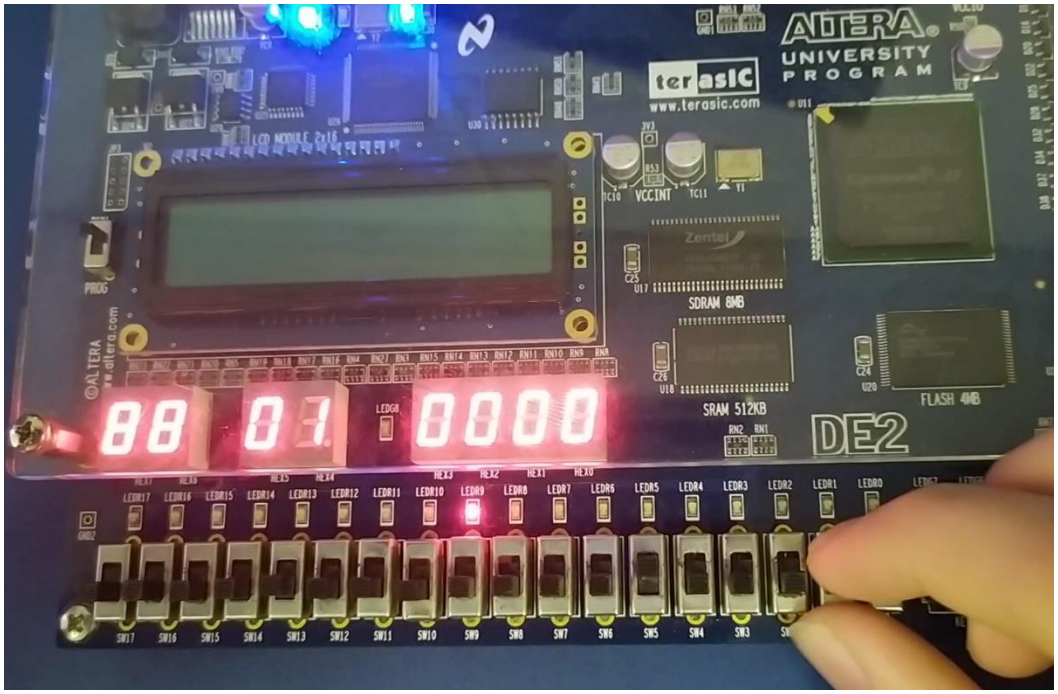

Imagem 10 – Funcionalmente da escrita na FIFO

Para fins de mostrar seu funcionalmente quando a mesma se encontra cheia, continuemos a dar o sinal de escrita para a máquina, até que chegamos à posição de 16 do contador da ROM. Porquê, como a exigência da arquitetura vista na imagem 1 desse relatório, a FIFO tinha como tamanho 16x13, ou seja, 16 posições sendo cada uma delas de 13Bits. Assim, ao chegar nesse ponto a FIFO se encontrou cheia, visto que, o LEDR0 se acendeu.

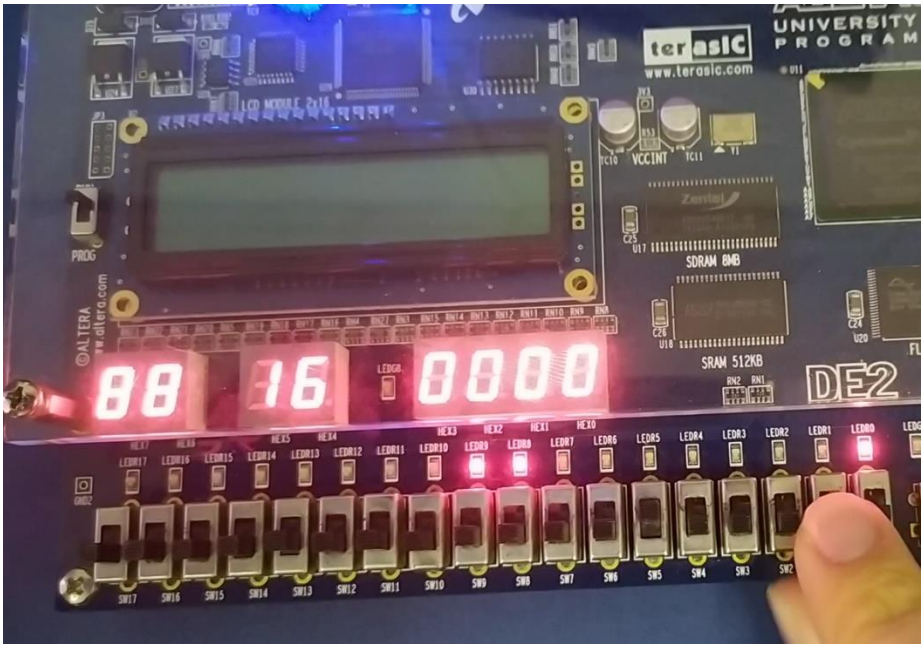

Imagem 11 – Funcionamento da FIFO quando cheia

]A partir daqui, começamos a acionar a leitura da fila. Ao levantar a chave SW0 mandamos um sinal a máquina informando-a que desejamos fazer a leitura dos valores salvos na FIFO. Quando isso ocorre, o led LEDR0 se apaga, comunicando que a fila não se encontra mais cheia e assim o primeiro valor inserido na fila e mostrado nos 4 displays da ponta. Em meu caso, a primeira posição da ROM tinha valor 1000 guardado nela.

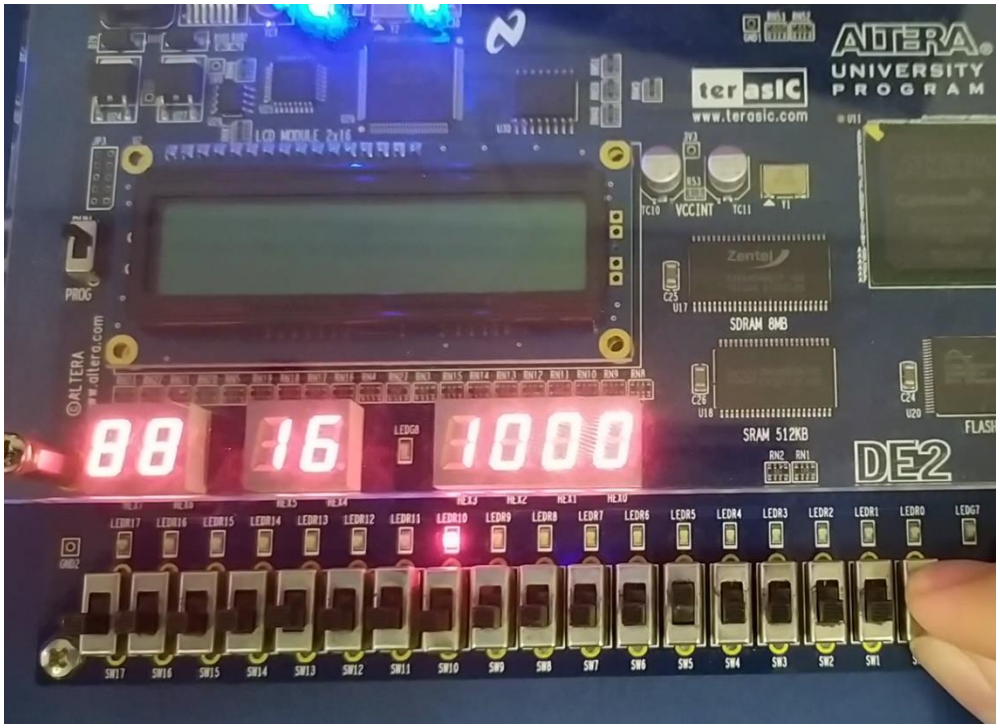

Imagem 12 – Funcionamento de leitura da FIFO

Se prosseguimos acionando a chave SW0, até que todos os valores guardados FIFO sejam lidos, voltaremos basicamente a configuração inicial, onde tínhamos o LEDR1 aceso, nos mostrando que a FIFO se encontra vazia novamente. Com a diferença que, caso não tenha pedido nenhuma nova escrita, a posição do contador continuará na 16.

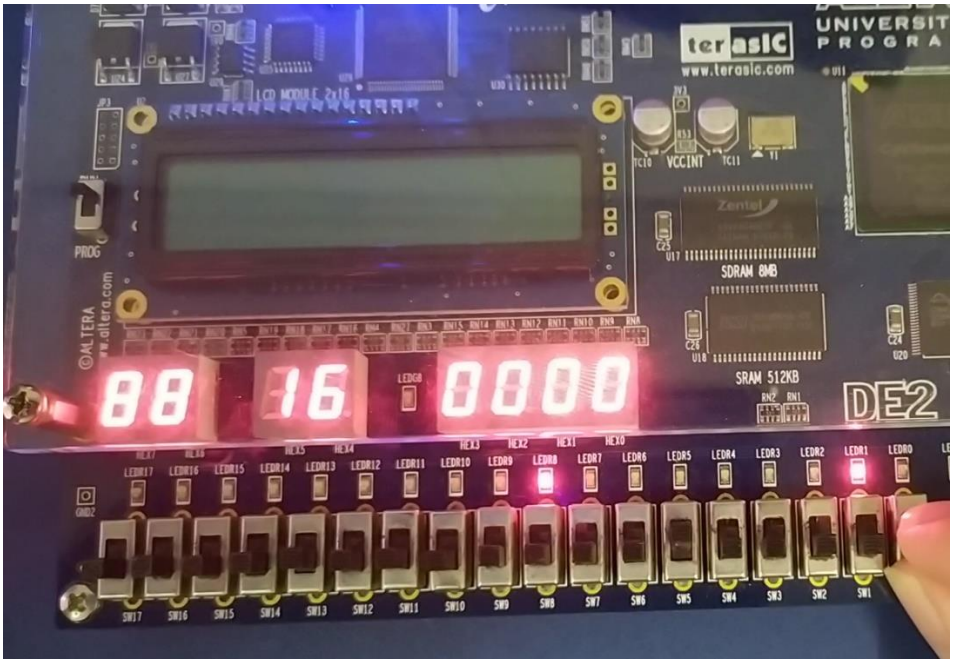

Imagem 13 – FIFO novamente vazia, após leitura de todas os valores guardados.

Os testes realizaram também mostraram o perfeito funcionalmente, caso se deseja-se fazer uma leitura sem antes ter enchido a fila, ou fazer uma escrita sem antes ter esvaziado a fila. Adicionalmente, ao apertamos no botão KEY3 da FPGA independentemente das posições que a FIFO ou o contador da ROM se encontram, todo o projeto sofre reset. Voltando assim, as suas configurações iniciais, presente na imagem 9 para esse respectivo teste.

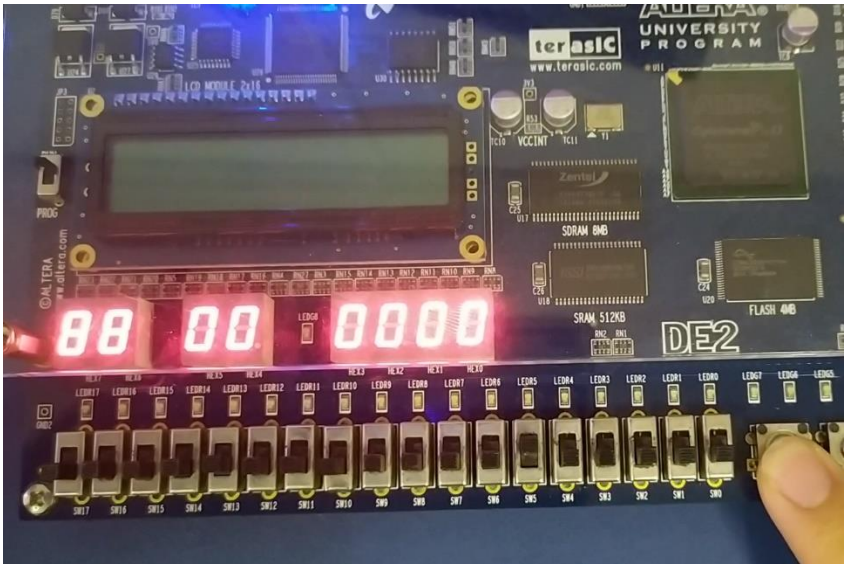

Imagem 14 – FIFO após botão reset ser pressionado.

Portanto, todos os testes físicos retornaram os resultados obtidos em simulador, constituindo um projeto implementável em FPGA.

4. Conclusão

Nos dias atuais, os projetos de circuitos aumentaram significativamente. Como também, a utilização de memória seja RAM ou ROM junto das mesmas. Para tal, a utilização de maquinas RTL para a projeção de circuitos tornou-se sua construção e codificação mais fáceis de serem entendidas e realizadas.

Nesse documento, foi exposto o passo a passo de como projetar uma máquina RTL para uma FIFO, com uma memória ROM para salvar os valores que serão inseridos na fila. Ademais, foi mostrado os resultados obtidos quando simulado o projeto utilizando a FPGA Ciclone II.

Disto podemos retirar as seguintes conclusões, dado a quantidade de componentes exigidos para a FIFO foi uma importante revisão dos conceitos visto anteriormente na matéria de Circuitos Digitais que agora serão de extrema importância para o que iremos enfrentar pela frente em Sistemas Digitais. Como também, o fato de utilizamos a FPGA para manipular e olhar nosso código funcionando como se fosse uma situação real, nos mostra a importância da matéria para os desafios que iremos encontrar no futuro da nossa profissão.

5. Referências Bibliográficas

D’AMORE, R.; VHDL: descrição e síntese de circuitos digitais. 2.ed. Rio de Janeiro: LTC, 2015. 292p.

STALLINGS, William. Arquitetura e organização de computadores. 8. ed. São Paulo: Pearson Prentice Hall, 2010.

VAHID, F.; Sistemas Digitais: projeto, otimização e HSLs; tradução Anatólio Laschuk. – Porto Alegre: Artmed, 2008. 560p.