UNIVERSIDADE FEDERAL DO RIO GRANDE DO NORTE

CENTRO DE TECNOLOGIA

DEPARTAMENTO DE ENGENHARIA ELÉTRICA

CIRCUITOS DIGITAIS

CIRCUITO DIGITAL PARA UM ALGORITMO DE FILA SIMPLES (FIFO)

ELE2715 - Grupo 03 - Projeto 06

Alysson Ferreira da Silva

Isaac de Lyra Junior

Pedro Henrique de Freitas Silva

Wesley Brito da Silva

Natal, 11 de abril de 2021

RESUMO

Este trabalho apresenta o projeto RTL (*register transfer level*) de um algoritmo de fila simples (FIFO, de *first in, first out*). A FIFO conta com 3 entradas de controle que são: **CLR_FIFO**, **WR** e **RD**, nesta ordem de prioridade. Também possui uma entrada e uma saída ambas de 13 bits para escrita ou leitura de dados controlados pelo usuário. Para registro dos dados inseridos pelo usuário, a FIFO utiliza um banco de registradores de tamanho 16x14. A gestão destes dados é auxiliada por circuitos comparadores, multiplexadores, demultiplexadores, entre outros. Por fim, a FIFO possui duas saídas para representar se a memória do circuito está cheia (**FU**) ou se está totalmente vazia (**EM**).

Palavras-chaves: Circuitos Digitais, RTL, Máquina de Estados, memória, FIFO.

SUMÁRIO

1 INTRODUÇÃO

Processadores específicos são utilizados no armazenamento e manipulação de conteúdos digitais, nos mais básicos podem ser exercitados conceitos de ciclos e conjuntos de instrução assim como armazenamento de variáveis de memória (CARVALHO, 2003; MORANDI, RAABE, ZEFERINO, 2006). Estes processadores são formados a partir da combinação de componentes referidos como blocos operacionais, que são responsáveis por realizar operações lógicas e aritméticas, e blocos de controle que atuam controlando/articulando os blocos operacionais (VAHID, 2008).

Os processadores podem ser resumidos a dois tipos: programáveis e customizados. As estruturas programáveis, ou processadores de propósitos gerais, são facilmente associadas a computadores pessoais onde as tarefas de processamento ficam armazenadas em uma memória. Não obstante, os processadores customizados, ou processadores de propósito único, são construídos no intuito de realizar uma única tarefa. Embora sejam estruturas de atividade simplória, este tipo de processador customizado são circuitos digitais detentores de uma computação mais ágil/eficiente, em termos de consumo energético, se comparado com os processadores de propósito geral (VAHID, 2008).

No tocante a metodologia de construção de processadores, existem diversos métodos. O método mais usual é conhecido como projeto em nível de transferência entre registradores (RTL, do inglês *Register Transfer Level*) (VAHID, 2008). Informalmente, a abordagem RTL consiste em descrever módulos de hardware por meio de um caminho de dados e uma unidade de controle. Nesse sentido, a modelagem RTL assume a precisão do nível do ciclo do relógio nas descrições do caminho de dados e da unidade de controle. Atualmente, as ferramentas de síntese aceitam modelos RTL como entrada para a produção de projetos de alta qualidade(CALAZANS et. al, 2003).

Em conjunto com projeto de processadores, a maioria dos circuitos necessita de elementos de memória. Essa capacidade das memórias torna os circuitos digitais muito versáteis e adaptáveis a várias situações (TOCCI, 2011).

Um banco de registadores MxN é um componente de memória dos blocos operacionais que proporciona um acesso eficiente a um conjunto de M registradores, onde cada registrador tem um tamanho de N bits (VAHID, 2008).

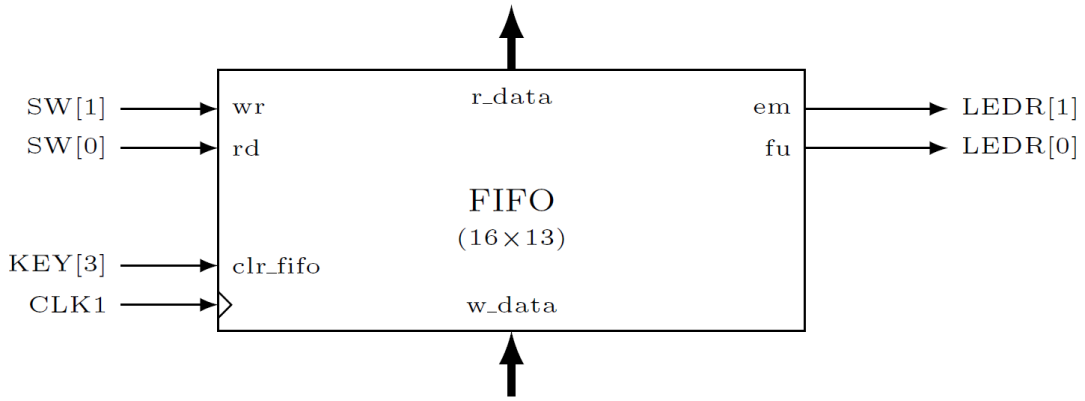
Dito isso, o presente relatório apresentará uma proposta de implementação de um algoritmo de fila simples (FIFO, de *first in, first out*), que utiliza a abordagem de modelagem de processadores RTL e banco de registradores. O projeto descrito na próxima seção

propõe-se a ser exequível, funcional e fundamentado no estado da técnica.

2 DESENVOLVIMENTO

O presente trabalho apresenta o projeto de um circuito lógico para implementar um algoritmo de fila simples. A FIFO conta com 3 entradas que são: **CLR_FIFO**, **WR** e **RD**, nesta ordem de prioridade. Os dados são introduzidos na FIFO quando a entrada **WR** estiver em nível lógico alto, neste momento, o banco de registradores recebe um valor de até 13 bits enviado pelo usuário após um pulso de *clock*. De maneira análoga, a retirada de dados da FIFO é sensível à variável **RD**, que quando recebe nível lógico alto, retira um dado de até 13 bits após um pulso de *clock*. Para reiniciar e limpar todos os dados da FIFO, utiliza-se a entrada **CLR_FIFO**, ou seja, ao ativar esta entrada, o circuito limpa todas as posições de memória e reinicializa os contadores internos. Por fim, de acordo com as saídas **FU** e **EM**, podemos indicar se a FIFO está com a memória cheia ou vazia, respectivamente.

Figura 1 - Proposta de implementação.



Fonte: (DIAS, 2021).

A fim de projetar o circuito da Figura 1, seguimos o método de projeto RTL que segue quatro passos, que são eles: obter uma máquina de estados de alto nível, criação do bloco operacional, conectar o bloco operacional ao bloco de controle e obter a máquina de estados finitos do bloco de controle. Ao mesmo tempo, necessitamos da utilização de circuitos sequenciais, onde serão utilizados como os elementos de memória.

2.1 CONSIDERAÇÕES DE PROJETO

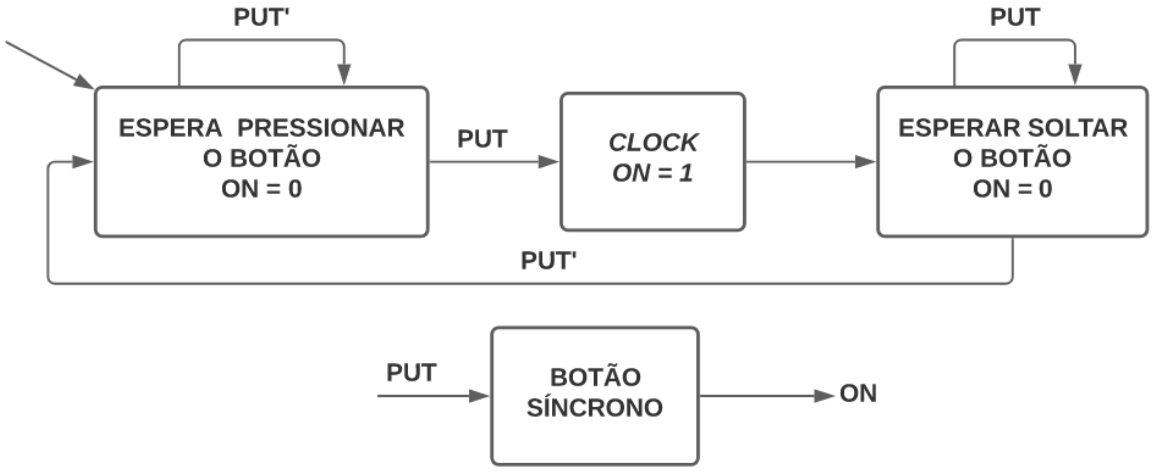
A FIFO foi projetada para continuar fazendo o registro de dados quando atingir o limite máximo de 16 números registrados. Desse modo, haverá uma perda do primeiro número digitado pelo usuário que fará com que o próximo número armazenado passe a ser o novo início. Ademais, observa-se nesta situação que o final da FIFO está coincidindo com o ínicio, motivo este que ocasiona a perda de dados. Por isso, quando a FIFO estiver cheira deverá ser implementada uma lógica que fará o início e o fim sejam os mesmos se o usuário continuar escrevendo dados sem realizar uma leitura.

Nesse sentido, foram utilizados registradores de 14 bits no projeto da máquina. Estes registradores designam 13 bits para o número que será inserido na FIFO, e o bit mais significativo (o décimo quarto bit) deverá indicar se o registrador X possui um número inserido através de um valor lógico alto. Devido a esta implementação a FIFO saberá através de um vetor de 16 posições, composto pelos bits mais significativos de cada registrador, se a máquina encontra-se cheia ou vazia. Além disso, o projeto considera para implementação da ação de Ler e Escrever o uso de botões assíncronos.

2.1.1 BOTÃO

O botão foi implementado para registrar o bit acionador no *clock* atual. Foram elaborados dois deste para que acionasse a leitura ou escrita dos bits. Na tabela 1 temos como variável representativa do **WR** ou **RD** (botões ativadores da escrita e leitura, respectivamente) o **ON**. Na Figura 3, podemos ver os estados desta máquina responsável pelo acionamento do botão.

Figura 2 - Máquina de estados do botão síncrono.



Fonte: Elaborada pelos autores.

A Tabela 1 evidencia o funcionamento do botão síncrono, onde a entrada **PUT** é um *pushbutton* que ao ser pressionado sincroniza apenas um pulso de *clock*, dessa forma, a saída **ON** só envia a máquina de troco um valor a cada pressionamento do *pushbutton* **PUT**, mesmo que esse pressionamento tenha duração maior do que um pulso de *clock* .

Tabela 1 - Equações das saídas do botão assíncrono.

	ESTADO ATUAL		ENTRADA	PRÓXIMO ESTADO		SAÍDA
ESTADOS	QB1	QB0	PUT	DB1	DB0	ON
ESPERA PRESSIONAR O BOTÃO	0	0	0	0	0	0
	0	0	1	0	1	0
CLOCK	0	1	0	1	0	1
	0	1	1	1	0	1
ESPERA SOLTAR O BOTÃO	1	0	0	0	0	0
	1	0	1	0	0	0

Fonte: Elaborada pelos autores.

A fim de simplificar as equações de estado futuro e de saída da Tabela 1, podemos utilizar o dispositivo do mapa de Karnaugh novamente.

Tabela 2 - Equações lógicas para as saídas da Tabela 1.

EQUAÇÃO BOOLEANA
DB1 = QB1' QB0 + QB1 QB0' PUT
DB0 = QB1' QB0' PUT
ON = QB1' QB0

Fonte: Elaborada pelo autor.

2.1.2 CLOCK UTILIZADO

A FIFO foi projetada com um bloco de controle que possui arquitetura de uma máquina de estados finitos do tipo Moore. Por causa disso, há maior segurança na execução da máquina em clocks com maiores frequências. Nesse sentido, recomenda-se que a implementação do projeto proposto seja realizada em altas frequências, e utiliza-se então 512 Hz. Nestas definições, a FIFO torna-se mais célere por ser capaz de fazer o registro dos dados inseridos pelo usuário de maneira mais ágil e eficiente.

2.2 MÁQUINA DE ESTADOS DE ALTO NÍVEL

A máquina de estados de alto nível conta com oito estados distintos que são: início, escrever, contador 1, espera, ler, apagar, contador 2 e contador 1 e 2. O estado “início” (000) é responsável por reiniciar os registradores de estado, liberar as memórias (banco de registradores) e limpar os contadores. Este estado é sensível apenas a variável de entrada **WR**, que quando ativada direciona a máquina de estados para o estado de escrever.

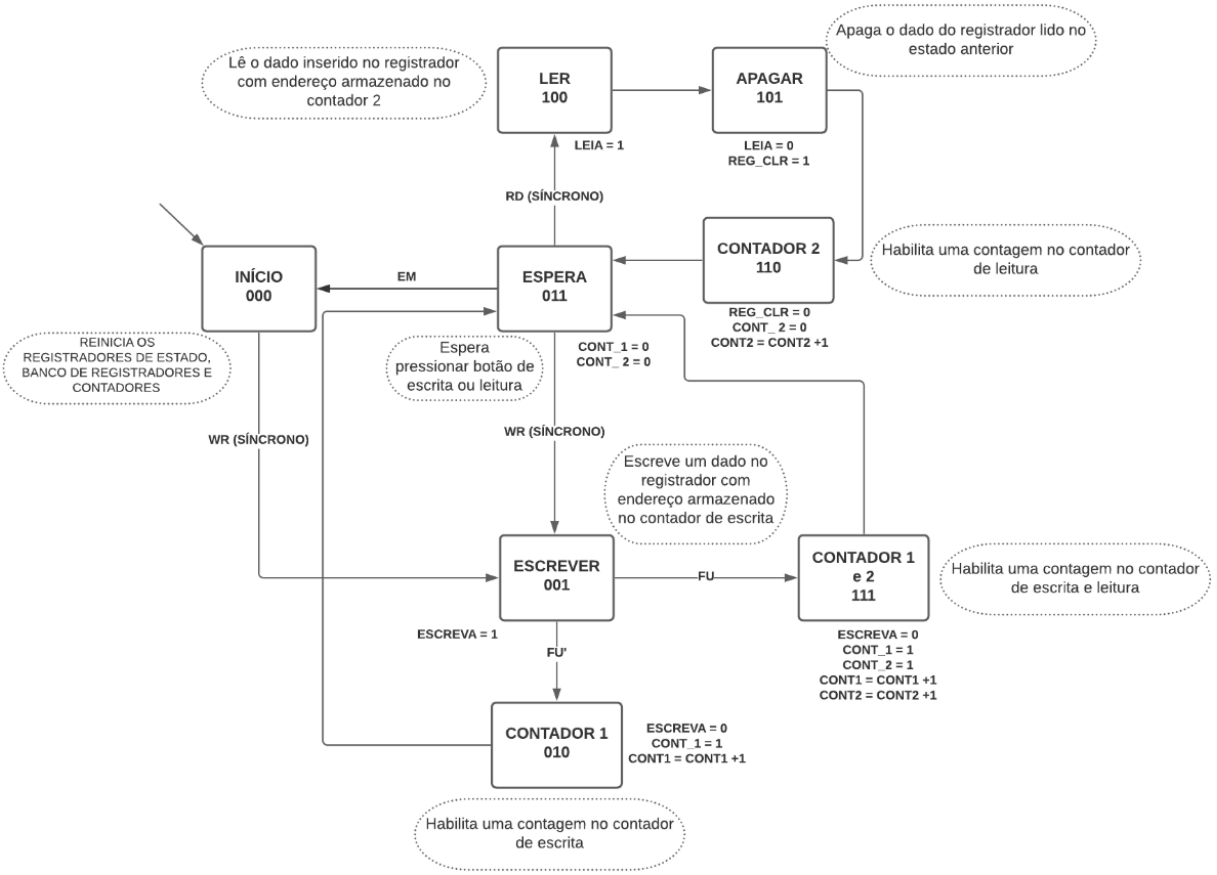
No estado “escrever” (001), o circuito permite a escrita do dado enviado pelo usuário ao mesmo tempo que verifica o nível lógico da variável de entrada **FU**. Caso esta entrada receba nível lógico baixo, a máquina de estados deverá prosseguir para o estado “contador 1” (010). Se a entrada **FU** estiver em nível lógico alto, o circuito deve prosseguir ao estado “contador 1 e 2”.

Ao assumir o estado “contador 1” (010), aciona-se a variável saída **CONT_1** que habilita o contador de escrita e o incrementa com mais 1. Após o pulso de *clock*, a máquina é direcionada ao estado de “espera” (011), este é controlado pelas variáveis de entrada **WR, RD** e **EM**, de forma que se nenhuma delas for ativada, o circuito permanece no mesmo estado. Caso as entradas **WR**, **RD** ou **EM** sejam ativadas, o circuito deve assumir os estados “escrever”, “ler” ou “início", respectivamente.

Ao chegar no estado “ler” (100), o circuito permite a leitura do dado endereçado naquele momento e, logo após o pulso de *clock*, deve prosseguir ao estado “apagar” (101). Neste estado, o registrador endereçado no momento da leitura do dado é reiniciado de modo a extinguir o dado armazenado naquele bloco de memória e, logo após um pulso de clock, deve seguir ao estado “contador 2” (110), onde é habilitada a contagem do contador de leitura. Para isso, a variável de saída **CONT_2** é acionada, e esta habilita o contador de leitura. Após outro tempo de relógio, o bloco de controle deverá voltar para o estado de espera.

Por fim, caso a máquina de estados esteja no estado “escrever” com a variável FU em nível lógico alto, o bloco será encaminhado para o estado “contador 1 e 2”. Neste estado, as variáveis de saída **CONT_1** e **CONT_2** são ativadas, e consequente a memória está cheia, é realizada a contagem igual entre os dígitos do início e do fim. Após, a máquina volta para o estado “espera”.

Figura 3 - Diagrama de blocos da máquina de alto nível.



Fonte: Elaborada pelos autores.

2.4 BLOCO OPERACIONAL

O bloco operacional será responsável por processar todos os dados não booleanos. Por isso, este bloco de operações conta com diversos circuitos, como: registradores, contadores, comparadores, dentre outros.

Primeiramente, o bloco operacional aguarda uma ação do bloco de controle, neste caso, o acionamento das variáveis de saída da máquina de estados ou da variável **CLR_FIFO**.

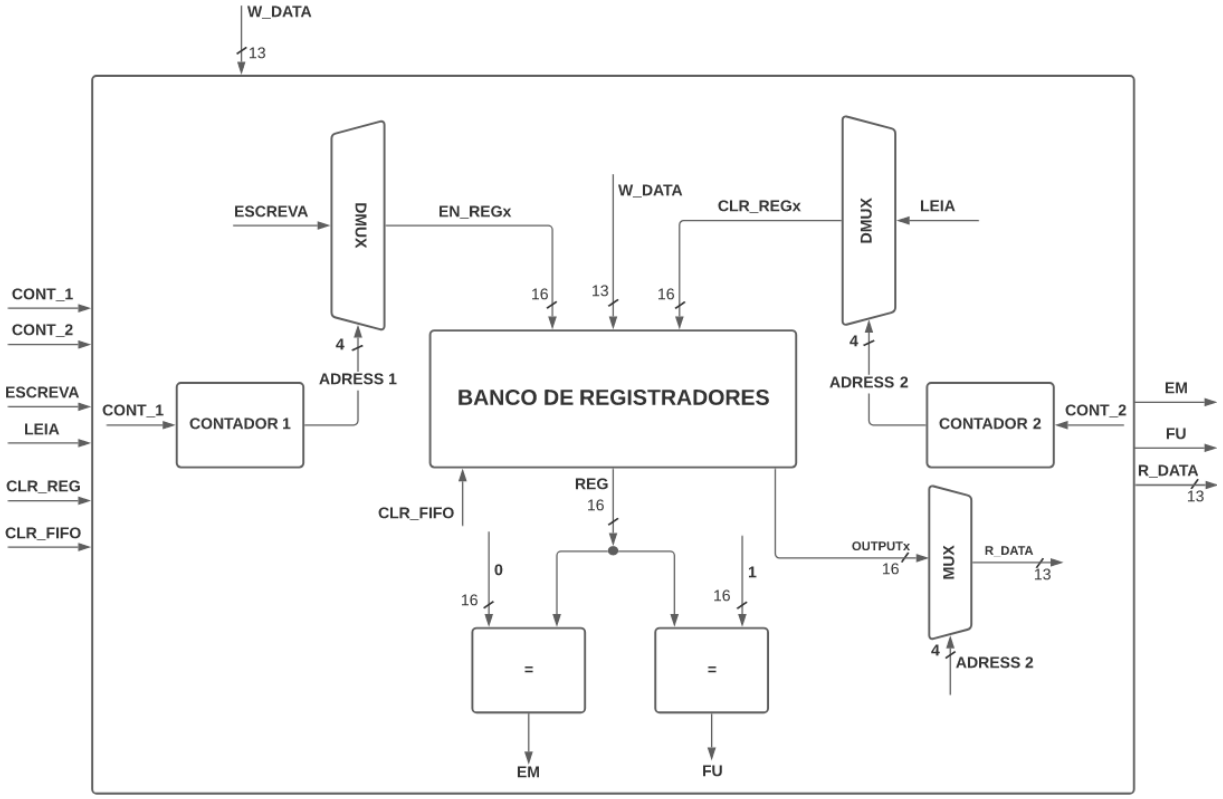
A variável **CONT_1** habilita a contagem no contador 1, onde este retorna o endereço da memória para alocar o próximo dado enviado pelo usuário sempre que a variável **ESCREVA** esteja em nível lógico alto.

De maneira análoga, a variável **CONT_2** habilita a contagem no contador 2, onde este retorna o endereço da memória para retirar o dado que atualmente está no início da fila sempre que a variável **LEIA** esteja em nível lógico alto.

Para verificar se a FIFO está com a memória totalmente cheia ou totalmente livre, podemos utilizar o 14º bit de todos os elementos de memória. A saída **EM** pode verificar igualdade deste 14º bit com 0, se todos forem iguais a zero, significa que a FIFO está com a memória totalmente livre. De maneira semelhante, a saída **FU** verifica se o 14º bit de cada elemento de memória é igual a 1, se todos forem iguais a 1, significa que a FIFO está com a memória totalmente cheia.

Por fim, ao aplicar a leitura de um dado, antes de removê-lo da memória da FIFO, podemos apresentar aquele conteúdo ao usuário através da saída **R_DATA.**

Figura 4 - Diagrama de blocos do bloco operacional.



Fonte: Elaborada pelos autores.

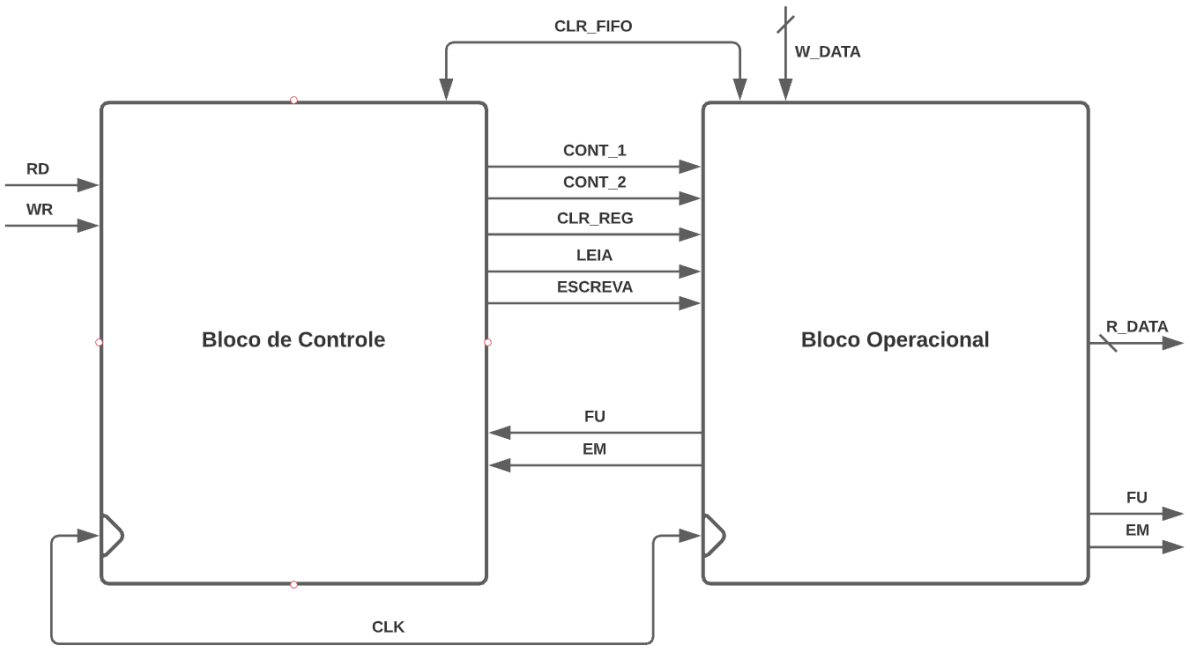
2.5 CONEXÃO DO BLOCO DE CONTROLE COM O BLOCO OPERACIONAL

O bloco operacional irá se comunicar com o controlador através de cinco entradas fornecidas pelo bloco de controle e a partir das entradas fornecidas, o bloco é capaz de gerar duas saídas. Duas entradas serão responsáveis pelos contadores de leitura e escrita, denominadas de **CONT_1** e **CONT_2** respectivamente, tais entradas, quando em nível lógico alto, irá habilitar a contagem durante um pulso de clock nos contadores citados. O **CLR_REG** será responsável por apagar os dados armazenados do registrador de endereço correspondente à saída do contador de leitura após a FIFO ler e fornecer na saída **R_DATA**.

As outras duas entradas restantes, denominadas de **LEIA** e **ESCREVA**, são utilizadas pelo processo de leitura e escrita, como o próprio nome sugere. A entrada **LEIA** é responsável por ler o dado armazenado no registrador de endereço correspondente à saída do contador de leitura e fornecer ao usuário através da saída **R_DATA**, todo este processo só acontecerá quando **LEIA** estiver em nível lógico alto. A entrada **ESCREVA**, quando em nível lógico alto, irá armazenar a entrada de 13 bits **W_DATA** no registrador de endereço correspondente ao contador de escrita.

As duas saídas fornecidas pelo bloco operacional, denominadas de **FU** e **EM**, servem para avisar que o banco de registradores está completamente preenchido ou completamente vazio, respectivamente. Estas devem ser conectadas a um LED cada para informar as ações supracitadas ao usuário.

Figura 5 - Conexões do Bloco de Controle com o Bloco Operacional.



Fonte: Elaborada pelos autores.

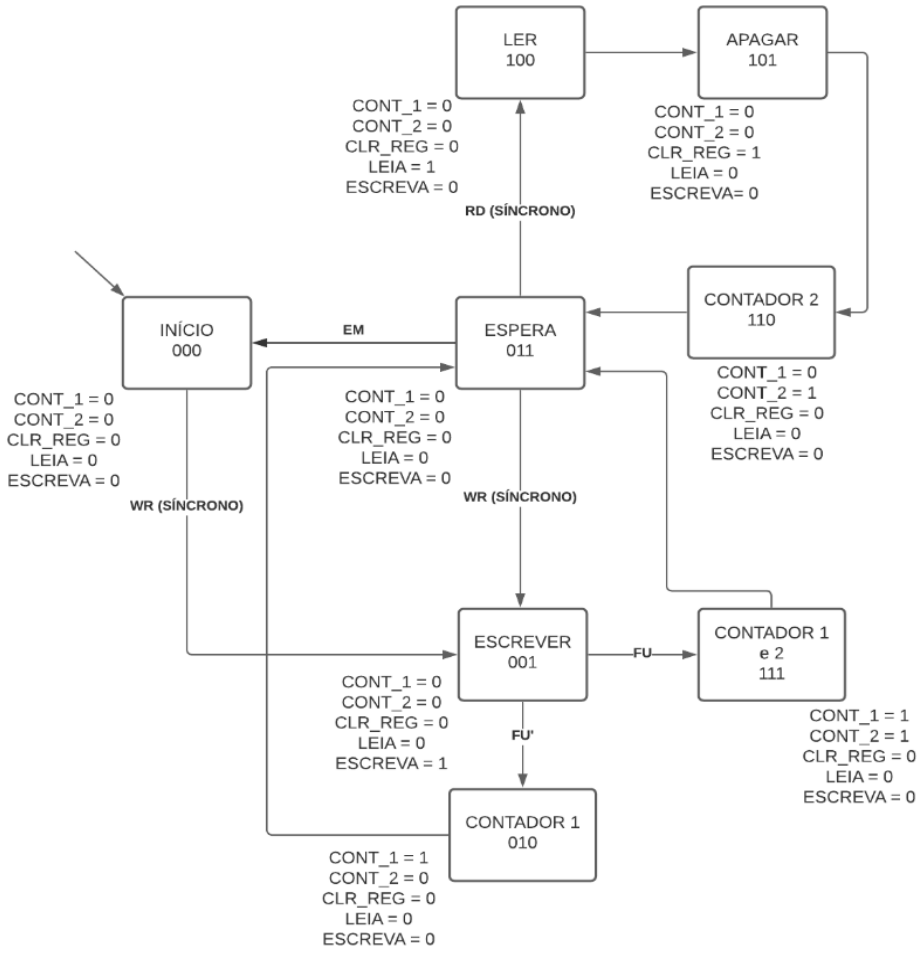
2.6 MÁQUINA DE ESTADOS FINITOS DO BLOCO DE CONTROLE

Após a criação da máquina de estados de alto nível, deve ser desenvolvida a máquina de estados de baixo nível. O estado “início”, após o botão síncrono **WR** ser pressionado, a informação do bit **ESCREVA** será enviado para o estado “escrever”. Este irá acionar o terceiro estado “contador 1”, onde contabilizará a posição dos dados que foram registrados, caso a variável **FU** (esta indica que a memória ficou cheia) seja 0 (zero) irá para o terceiro estado, mas caso seja 1, então acionará os dois contadores. Logo depois, o estado “espera” irá

agir. É nele que vai ocorrer a escolha dos próximos estados após o apertar dos botões síncronos **RD** ou **WR**. O **RD** chamará o "ler" que vai operar o clear dos registradores, apagando o dado selecionado ("apagar"), contabilizando no segundo contador introduzido no "contador 2". Por fim, ele volta para o estado "espera". Se por ventura, a memória estiver vazia, então será retornado ao primeiro estado "início" com a variável **EM** nível lógico alto e se o botão **WR** for apertado, ainda estando no "espera", quando for verificado que a memória estiver cheia (**FU** nível lógico alto), o "contador 1 e 2" demandará a utilização dos dois contadores com **CONT__1** e **CONT_2** sendo iguais a 1.

Na Figura 6, temos a máquina com as declarações do funcionamento de cada estado e as respectivas variáveis parametrizadas, como foi descrito no parágrafo anterior.

Figura 6 - Diagrama de blocos da máquina de baixo nível.



Fonte: Elaborada pelos autores.

Derivado desta figura, as equações lógicas foram elaboradas, posteriormente a utilização do mapa de Karnaugh. Com isso, a tabela 2 possui as expressões booleanas da MDE.

Tabela 3 - Equações das equações de saída do bloco de controle.

VARIÁVEIS	EQUAÇÕES EM VERILOG
D2	**~Q1 Q0 FU + ~Q2 Q1 Q0 ~WR RD ~EM + Q2 ~Q1**
D1	**~Q1 Q0 + Q1 ~Q0 + Q1 ~WR ~RD ~EM + Q2 Q1**
D0	**~Q0 WR + ~Q2 ~Q1 Q0 FU + Q1 ~Q0 + Q1 ~RD ~EM + Q1 WR ~EM + Q2 ~Q0 + Q2 Q1**
CONT_1	**~Q2 Q1 ~Q0 + Q2 Q1 Q0**
CONT_2	**Q2 Q1**
CLR_REG	**Q2 ~Q1 Q0**
LEIA	**Q2 ~Q1 ~Q0**
ESCREVA	**~Q2 ~Q1 Q0**

Fonte: Elaborada pelos autores.

3 CONCLUSÃO

Neste trabalho é apresentado o projeto de um circuito digital para um algoritmo fila simples (FIFO, de *first in, first out*). A FIFO conta com 3 entradas que são: **CLR_FIFO**, **WR** e **RD**, nesta ordem de prioridade. Também conta uma entrada e uma saída de dados ambas de 13 bits para alocação e remoção de dados na memória do circuito. Além disso, a FIFO possui duas saídas (**FU** e **EM**) que informam se a memória do circuito está cheia ou vazia.

O banco de registradores da FIFO é de tamanho 16x14, sendo 16 registradores com largura de 14 bits. Este 14º bit é utilizado para auxiliar o circuito a representar quando as memórias enchem ou esvaziam totalmente. Além disso, os registradores utilizados na memória da FIFO são acessados através de dois endereços diferentes, um referente a inserção de dados e um para leitura de dados.

A gestão de dados internos da FIFO é realizada por um multiplexador, dois de multiplexadores, dois comparadores e dois contadores. O multiplexador é utilizado para selecionar qual dos 16 registradores será a saída do circuito. Os demultiplexadores são utilizados para: realizar a escolha de qual registrador será inserido o número solicitado pelo usuário; realizar a escolha de qual registrador será limpo quando o usuário solicitar leitura. Os contadores registram qual será o início e o final da FIFO, e enviam essa informação para os

demultiplexadores e multiplexadores do sistema. Por fim, os comparadores irão identificar quando o circuito estará com a memória de armazenamento cheia ou vazia.

4 REFERÊNCIAS

CALAZANS, Ney, et al. **From VHDL register transfer level to SystemC transaction level modeling: a comparative case study**. In: 16th Symposium on Integrated Circuits and Systems Design, 2003. SBCCI 2003. Proceedings. IEEE, 2003. p. 355-360.

CARVALHO, José Oscar Fontanini de. **O papel da interação humano-computador na inclusão digital. Transinformação**, 2003, 15.SPE: 75-89.

DIAS, Samaherni Morais. **Notas de aula: FIFO**. 18 de jan. 2021, 30 apr. 2021. 2p.

MORANDI, Diana; RAABE, André Luis Alice; ZEFERINO, Cesar Albenes. **Processadores para ensino de conceitos básicos de arquitetura de computadores**. In: Proceedings of the 18th International Symposium on Computer Architecture and High Performance Computing–Workshops. 2006. p. 17-24.

TOCCI, Ronald J.; WIDMER, Neal S.; MOSS, Gregory L.. **Sistemas Digitais: Princípios e Aplicações**. 11ª ed. São Paulo: Pearson, 2011. 844 p.

VAHID, Frank. **Sistemas digitais: projeto, otimização e HDLs** / Frank Vahid; tradução: Anatólio Laschuk. - Porto Alegre: Artmed, 2008. 560p.

ANEXOS

ANEXO A - RELATO SEMANAL

Líder: Alysson Ferreira da Silva

A.1 Equipe

Função do grupo:	Discente
Redator	Pedro Henrique de Freitas Silva
Debatedor	Isaac de Lyra Junior
Videomaker	Wesley Brito da Silva
Auxiliar	N/A

Fonte: Elaborado pelos autores.

A.2 Defina o problema

O presente trabalho consiste em projetar um circuito digital para implementar um sistema de armazenagem que funciona associado a um algoritmo de fila simples. O algoritmo de fila implementado foi o "*First In, First Out*" (FIFO), onde o primeiro número registrado no circuito é o primeiro a sair quando é realizada a leitura de um dado no sistema.

Nesse sentido, o circuito foi projetado para ter uma saída e uma entrada de dados. Ademais, para auxiliar o sistema na gestão dos dados são implementadas 3 entradas de um bit cada: WR, RD, CLR_FIFO, CLK. Dito isso, o bit de entrada WR é responsável por indicar para o circuito quando o usuário solicita o registro de um bit. O bit RD informa ao circuito quando há solicitação de leitura de um dado, e o CLR_FIFO é uma entrada assíncrona que reseta a FIFO para o estado inicial e limpa os registradores de memória. A entrada CLK recebe o clock do sistema para que todo o circuito funcione.

O circuito ainda conta com duas saídas que fazem interface com o usuário. A primeira saída foi denominada de "EM", quando esta encontra-se em nível lógico alto o sistema estará com os registradores vazios. A segunda nomeia-se "FU", está indica quando a FIFO está cheia através de um valor lógico alto. Ambas as saídas estão conectadas a um LED para informar ao usuário a ação supracitada para cada variável.

Além disso, quando a FIFO encontra-se cheia e é solicitado um novo registro de dado, o circuito sobrescreve o primeiro número registrado pelo novo dado e transforma o segundo dado dado escrito na FIFO no novo primeiro dado. Dessa forma, sempre que o circuito encontra-se cheio e um novo dado é registrado, tem-se uma perda de dados.

A.3 Registro do brainstorming

A primeira reunião do grupo foi utilizada somente para fazer as definições de trabalho durante a semana. Nesse sentido, definiu-se que as reuniões teriam no máximo 100 minutos e que todo o tempo dedicado além disso no dia seria considerado extra. Outro ponto decidido neste momento foi a função que cada membro executaria.

O segundo momento de reunião foi utilizado para construção da máquina de baixo nível. Neste encontro foi decidido que o grupo projetaria uma máquina de estados finitos do tipo Moore, e que esta máquina seria caracterizada por 8 estados. Realizada a construção da máquina, o grupo construiu em um momento posterior a tabela de transição de estados. Na etapa de construção da tabela, a execução do projeto ficou mais palpável. Dessa forma, a 'projetificação' das variáveis de entrada e saída que a máquina precisaria ter foram mais fáceis de identificar.

No terceiro encontro do grupo, dedicou-se a pensar na possível implementação do bloco operacional. Nesse sentido, foram vistas as possibilidades de circuito que combinados pudessem fazer uso das variáveis de saída da tabela e ao mesmo tempo entregar os valores lógicos das variáveis de entrada do bloco de controle.

Por fim, o último momento de encontro foi utilizado para auxiliar o redator na escrita do relatório. Neste procedimento, foram criadas as imagens do presente relatório. Ou seja, as imagens que fazem menção às conexões entre o bloco de controle e bloco operacional, assim como a transformação da máquina de baixo nível criada para um diagrama de alto nível.

A.4 Pontos-chaves

Os principais pontos para a execução do projeto estão centrados nas definições de projeto. Ou seja, na decisão de clock em alta frequência que acarretou na definição de uso de uma máquina de Moore e no uso de um botão síncrono. Assim como, na funcionalidade do circuito que não é travado quando está cheio. Ressalta-se ainda, que a utilização da metodologia de criação de processadores RTL auxiliou o grupo e tornou a prontificação do presente trabalho mais ágil e célere.

A.5 Questões de pesquisa

Os tópicos mais importantes para a implementação deste projeto são os seguintes:

- Banco de registradores
- Algoritmo de fila Simples FIFO
- Data Sheets de FIFOs
- Bloco operacional
- Máquina de alto nível
- Máquina de baixo
- Projeto RTL

A.6 Planejamento da pesquisa

Os 2 primeiros dias da semana foram reservados para estudo e compreensão de funcionamento da FIFO e de banco de registradores. Nos dias subsequentes as pesquisas foram realizadas conforme o grupo encontrava desafios na projetificação do circuito.

ANEXO B - TABELA DE TRANSIÇÃO DE ESTADOS

Figura 1 - Tabela de transição de estados.

Estados	Estado Atual			Entradas				Estado Futuro			Saídas				
	Q2	Q1	Q0	WR	RD	FU	EM	D2	D1	D0	CONT_1	CONT_2	CLR_REG	LEIA	ESCREVA
INICIO	0	0	0	0	X	X	X	0	0	0	0	0	0	0	0
	0	0	0	1	X	X	X	0	0	1	0	0	0	0	0
ESCREVER	0	0	1	X	X	0	X	0	1	0	0	0	0	0	1
	0	0	1	X	X	1	X	1	1	1	0	0	0	0	1
CONTADOR 1	0	1	0	X	X	X	X	0	1	1	1	0	0	0	0
ESPERA	0	1	1	0	0	X	0	0	1	1	0	0	0	0	0
	0	1	1	0	1	X	0	1	0	0	0	0	0	0	0
	0	1	1	1	X	X	0	0	0	1	0	0	0	0	0
	0	1	1	X	X	X	1	0	0	0	0	0	0	0	0
LER	1	0	0	X	X	X	X	1	0	1	0	0	0	1	0
APAGAR	1	0	1	X	X	X	X	1	1	0	0	0	1	0	0
CONTADOR 2	1	1	0	X	X	X	X	0	1	1	0	1	0	0	0
CONTADOR 1 E 2	1	1	1	X	X	X	X	0	1	1	1	1	0	0	0

Fonte: Elaborada pelos autores.