Ministério da Educação
Universidade Tecnológica Federal do Paraná
Câmpus Apucarana
Bacharelado em Engenharia de Computação

Universidade Tecnológica Federal do Paraná

Engenharia de Computação

PROJETO DE UM MULTIPLEXADOR GENÉRICO

Maria Eduarda Pedroso

Professor orientador: Marcelo de Oliveira

Apucarana

Outubro/2023

SUMÁRIO

1. RESUMO

Neste trabalho, focamos no desenvolvimento e implementação de um multiplexador genérico, cujas características permitiam a configuração tanto do número de entradas quanto da quantidade de bits por entrada. A ênfase foi dada na flexibilidade e adaptabilidade do design proposto. Utilizando uma placa FPGA, uma versão específica do multiplexador foi concretizada, apresentando 4 entradas de 2 bits cada. As entradas foram associadas às chaves da FPGA, enquanto as saídas foram mapeadas para LEDs ou SSDs. A simulação do projeto confirmou sua funcionalidade e eficiência, evidenciando sua aplicabilidade prática em contextos que requerem multiplexação adaptável.

Palavras-chave: multiplexador genérico, FPGA, implementação, simulação.

2. INTRODUÇÃO

No domínio da eletrônica digital, o multiplexador (mux) desempenha um papel crucial como um dispositivo que seleciona uma de várias entradas e a direciona para uma única saída. Um multiplexador genérico, ao contrário dos designs fixos tradicionais, oferece uma abordagem mais flexível, permitindo a configuração de seu número de entradas e a quantidade de bits por entrada. Esta adaptabilidade tem implicações significativas.

A concepção de um multiplexador genérico é motivada por duas principais vantagens. Primeiramente, proporciona uma solução única que pode ser adaptada para atender a diferentes requisitos de projeto sem a necessidade de redesenhar ou adquirir um novo componente. Isso significa que, ao invés de manter diferentes versões de multiplexadores para diferentes aplicações, um único design genérico pode ser reconfigurado conforme necessário.

Em segundo lugar, em um ambiente de prototipagem rápida ou em cenários onde as especificações podem mudar, ter um design que possa ser facilmente ajustado é uma vantagem considerável. Reduz o tempo de desenvolvimento e permite que os engenheiros e designers adaptem-se rapidamente a novos requisitos ou corrijam problemas sem ter que retornar ao estágio inicial de design.

Neste relatório, exploraremos em profundidade a implementação e aplicação de um multiplexador genérico.

2.1. Objetivos

O presente trabalho tem como objetivo:

- Desenvolvimento de um Multiplexador Genérico: Projetar e codificar um multiplexador que permita flexibilidade na definição do número de entradas e da quantidade de bits por entrada.
- Implementação em Hardware: Utilizar uma placa FPGA para concretizar e testar uma instância específica do multiplexador genérico, com 4 entradas de 2 bits cada.
- Interação com Componentes da Placa: Associar as entradas do multiplexador às chaves disponíveis na FPGA e mapear as saídas para LEDs ou SSDs, proporcionando uma interface visual e tátil para o funcionamento do multiplexador.
- Simulação da Implementação: Conduzir simulações para validar a correta funcionalidade do multiplexador genérico, assegurando que o design proposto opere conforme as especificações definidas.
- Análise e Avaliação: Avaliar o desempenho, eficiência e adaptabilidade do multiplexador genérico, identificando potenciais melhorias ou ajustes necessários.
- Demonstração Prática: Apresentar o funcionamento do multiplexador implementado na placa FPGA, evidenciando sua operação e resultados em um cenário real.

3. METODOLOGIA

O presente trabalho foi desenvolvido utilizando os conceitos e técnicas estudados durante as aulas da disciplina. Bem como foi empregado o uso do software Quartus, de modo a facilitar e implementar na placa nossa lógica.

3.1. Código multiplexador genérico

Em um primeiro momento, foi feita a implementação do multiplexador para teste com um número de 4 entradas de 8 bits, para esse código utilizamos de variáveis genéricas e também lógicas para contabilizar e criar nossos vetores de entrada, saída e seletor. Como mostrado abaixo no trecho do código no qual temos a entidade:

```
entity aula230830 is
    generic (
        N_SELETOR : integer := 2; --Numero de bits seletor
        BITS_POR_ENTRADA : integer := 2  -- Número de bits por entrada
    );
    port (
        Seletor : in STD_LOGIC_VECTOR(N_SELETOR-1 downto 0); -- Seletor
        X : in STD_LOGIC_VECTOR(BITS_POR_ENTRADA*2**N_SELETOR-1
downto 0); -- Entradas do multiplexador
        Y : out STD_LOGIC_VECTOR(BITS_POR_ENTRADA-1 downto 0); -- Saída
do multiplexador
                    LED: out STD_LOGIC_VECTOR(9-N_SELETOR downto 0) -- Saída
do multiplexador
    );
end aula230830;
```

Como podemos observar temos duas variáveis genéricas (BITS_POR_ENTRADA e N_SELETOR), com elas fazemos toda a lógica para o seletor, entrada e saída sendo a saída o tamanho de BITS_POR_ENTRADA, seletor o tamanho de N_SELETOR, que seria o número de bits do seletor e por fim a entrada que vai ser um array com o número de bits por entrada multiplicado pela quantidade de escolhas do seletor, ou seja $2^{N_{SELETOR}}$.

Para o funcionamento do multiplexador foi desenvolvido a lógica matemática para que a saída fosse correta, abaixo temos o trecho no qual isso é demonstrado.

```
architecture MUX_GENERICO of aula230830 is
begin

    Y <= X((to_integer(unsigned(Seletor)) * BITS_POR_ENTRADA) +
BITS_POR_ENTRADA - 1 downto to_integer(unsigned(Seletor)) *
BITS_POR_ENTRADA);

end MUX_GENERICO;
```

Como sabemos pela teoria do multiplexador o seletor escolhe uma faixa de x de acordo com o número que está no seletor juntamente com a quantidade de bits necessários para cada faixa, precisamos também nos atentar que a linguagem VHDL inicia seu array em 0.

3.2. Implementação na placa

Como nosso código é genérico para a placa, apenas foi preciso pensar em quantos switches tinhamos e quais seriam os seletores e como mostraremos a saída. Foi escolhido utilizar 2 switchs para seletores e o resto para a entrada, a saída foi feita através dos leds em cima dos switchs dos seletores e os outros leds foram apagados, abaixo podemos ver o pinPlaner e o código.

```
-- biblioteca
library ieee; -- Importa a biblioteca padrão IEEE.
use ieee.std_logic_1164.all; -- Usa o pacote que define os tipos de lógica padrão.
use ieee.numeric_std.all; -- Importa o pacote para operações numéricas padrão.

-- temos 3: entradas x, seletor, saida y
-- X == 16 entradas com 8 bits
-- sel == 4 bits 2^4==16
-- Y == 1 saida com 8 bits

entity aula230830 is -- Declaração da entidade chamada "aula230830".
    generic (
        N_SELETOR : integer := 2; -- Define o número de bits do seletor, por padrão
2.
```

```
        BITS_POR_ENTRADA : integer := 2  -- Define o número de bits por entrada, por padrão 2.
    );
    port (
        Seletor : in STD_LOGIC_VECTOR(N_SELETOR-1 downto 0); -- Declara a entrada Seletor com N_SELETOR bits.
        X : in STD_LOGIC_VECTOR(BITS_POR_ENTRADA*2**N_SELETOR-1 downto 0); -- Declara a entrada X com BITS_POR_ENTRADA * 2^N_SELETOR bits.
        Y : out STD_LOGIC_VECTOR(BITS_POR_ENTRADA-1 downto 0); -- Declara a saída Y com BITS_POR_ENTRADA bits.
                    LED: out STD_LOGIC_VECTOR(9-N_SELETOR downto 0) -- Declara a saída LED com 10 - N_SELETOR bits.
    );
end aula230830; -- Final da declaração da entidade.

architecture MUX_GENERICO of aula230830 is -- Início da arquitetura chamada "MUX_GENERICO" da entidade "aula230830".
begin
    -- A linha abaixo implementa a função do multiplexador.
    -- Y recebe a parte do vetor X selecionada pelo valor de Seletor.
    -- A fórmula matemática calcula qual segmento do vetor X deve ser selecionado.
    Y <= X((to_integer(unsigned(Seletor)) * BITS_POR_ENTRADA) + BITS_POR_ENTRADA - 1 downto to_integer(unsigned(Seletor)) * BITS_POR_ENTRADA);

    LED <= (others=>'0'); -- Define todos os bits da saída LED para '0'.

end MUX_GENERICO; -- Final da arquitetura MUX_GENERICO.
```

Figura 1 - PinPlaner

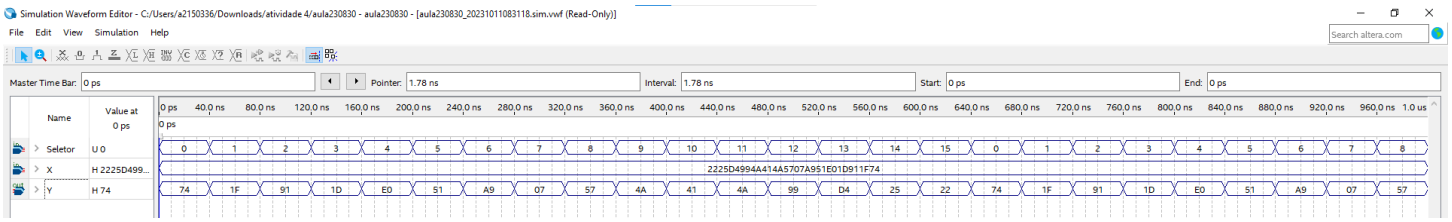
Node Name	Direction	Location	I/O Bank	VREF Group	Fitter Location	I/O Standard	Reserved	Current Strength	Slew Rate
LED[7]	Output	PIN_B10	7	B7_N0	PIN_B10	2.5 V		12mA (default)	2 (default)
LED[6]	Output	PIN_D13	7	B7_N0	PIN_D13	2.5 V		12mA (default)	2 (default)
LED[5]	Output	PIN_C13	7	B7_N0	PIN_C13	2.5 V		12mA (default)	2 (default)
LED[4]	Output	PIN_E14	7	B7_N0	PIN_E14	2.5 V		12mA (default)	2 (default)
LED[3]	Output	PIN_D14	7	B7_N0	PIN_D14	2.5 V		12mA (default)	2 (default)
LED[2]	Output	PIN_A11	7	B7_N0	PIN_A11	2.5 V		12mA (default)	2 (default)
LED[1]	Output	PIN_B11	7	B7_N0	PIN_B11	2.5 V		12mA (default)	2 (default)
LED[0]	Output	PIN_A10	7	B7_N0	PIN_A10	2.5 V		12mA (default)	2 (default)
Seletor[1]	Input	PIN_C11	7	B7_N0	PIN_C11	2.5 V		12mA (default)	
Seletor[0]	Input	PIN_C10	7	B7_N0	PIN_C10	2.5 V		12mA (default)	
X[7]	Input	PIN_D12	7	B7_N0	PIN_D12	2.5 V		12mA (default)	
X[6]	Input	PIN_C12	7	B7_N0	PIN_C12	2.5 V		12mA (default)	
X[5]	Input	PIN_A12	7	B7_N0	PIN_A12	2.5 V		12mA (default)	
X[4]	Input	PIN_B12	7	B7_N0	PIN_B12	2.5 V		12mA (default)	
X[3]	Input	PIN_A13	7	B7_N0	PIN_A13	2.5 V		12mA (default)	
X[2]	Input	PIN_A14	7	B7_N0	PIN_A14	2.5 V		12mA (default)	
X[1]	Input	PIN_B14	7	B7_N0	PIN_B14	2.5 V		12mA (default)	
X[0]	Input	PIN_F15	7	B7_N0	PIN_F15	2.5 V		12mA (default)	
Y[1]	Output	PIN_A8	7	B7_N0	PIN_A8	2.5 V		12mA (default)	2 (default)
Y[0]	Output	PIN_A9	7	B7_N0	PIN_A9	2.5 V		12mA (default)	2 (default)
<<new node>>									

Fonte: Autoria Própria

4. RESULTADOS E DISCUSSÃO

Após a criação do código foi feito uma simulação, a fim de testar e validar o código antes de inserir o mesmo na placa, abaixo conseguimos ver que o código funcionou da maneira esperada, retornando os valores corretos para cada número adicionado ao seletor.

Figura 2 - Simulação dentro do Quartus

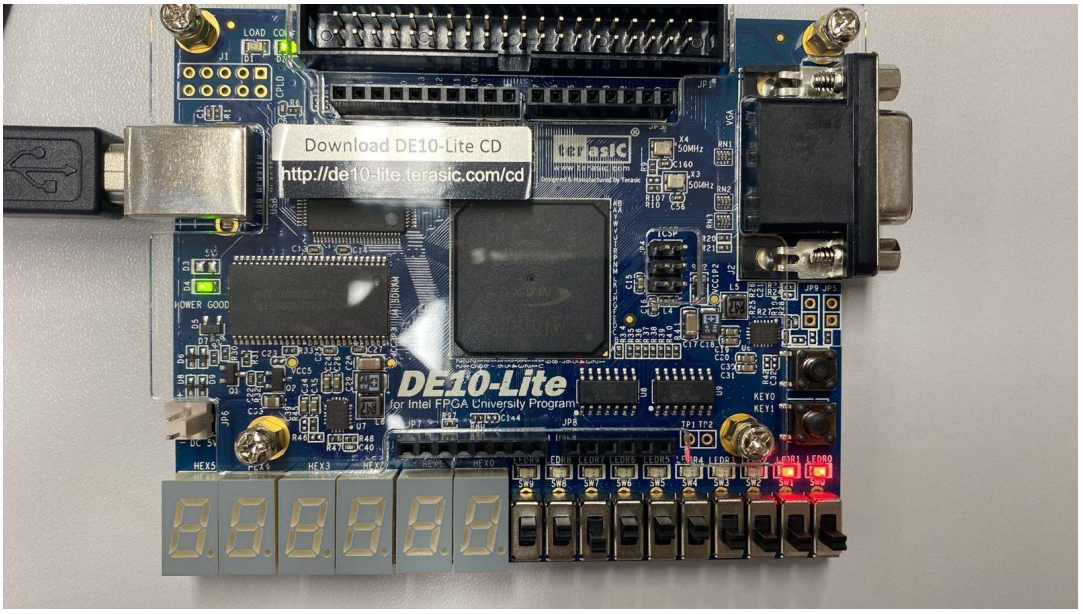


Fonte: Autoria Própria

Após isto, foi alterado o código a fim de conseguir adicionar na placa, para a placa utilizamos 4 entradas de 2 bits cadas e um seletor com 2 bits para conseguir se adequar ao número necessário de entradas. Cada entrada foi ligada a um switch da nossa placa, bem como o seletor. Para visualizar se a saída estava condizente com a entrada foi utilizado dois leds, como mostrado na figura 1.

Com a implementação e pinPlaner prontos foi startado o código na placa, os resultados foram condizentes com a entrada como podemos visualizar nessa imagem abaixo

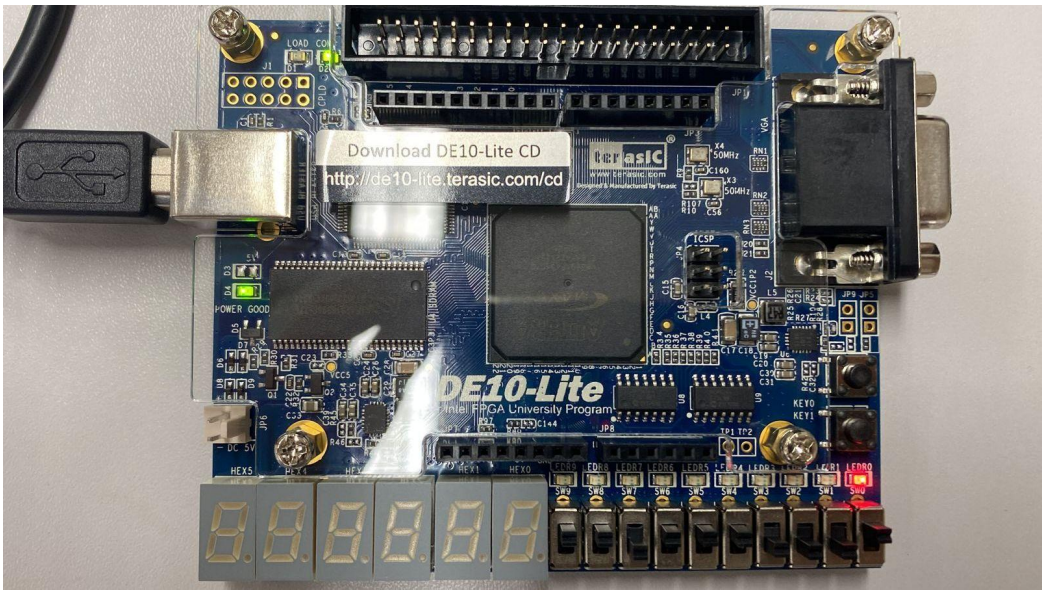
Figura 3 - Saída para quando seletor é zero



Fonte: Autoria Própria

Como podemos observar, o seletor está em zero já transformando ele em inteiro, nossa faixa de seleção é a primeira e como podemos ver nossos 2 switches estão em 1 e 1 o que condiz com a nossa saída nos leds.

Figura 4 - Saída para quando seletor é um



Fonte: Autoria Própria

Para fim de validação foi testado o seletor em outra faixa, no caso faixa um e também obtivemos um resultado satisfatório, como podemos ver na figura 4.

5. CONSIDERAÇÕES FINAIS

Ao longo da realização dessa prática, exploramos o desenvolvimento, implementação e avaliação de um multiplexador genérico. A flexibilidade introduzida por este design oferece benefícios inegáveis, permitindo uma adaptabilidade sem precedentes em comparação com designs tradicionais fixos de multiplexadores.

A implementação bem-sucedida na placa FPGA confirmou a viabilidade prática do projeto. Através da interface com componentes da placa, como chaves e LEDs ou SSDs, conseguimos observar de maneira tangível o desempenho do multiplexador, validando sua operação conforme especificado.

No entanto, como qualquer projeto técnico, reconhecemos que sempre há espaço para refinamentos e otimizações. A tecnologia e as necessidades do setor estão em constante evolução, e é essencial permanecer adaptável e receptivo a novas ideias e inovações.

Em conclusão, a atividade proporcionou insights valiosos sobre a importância e aplicabilidade dos multiplexadores genéricos no campo da eletrônica digital. O conhecimento e a experiência adquiridos servirão como base sólida para futuras investigações e projetos nesta área, sempre buscando combinar inovação com funcionalidade prática.