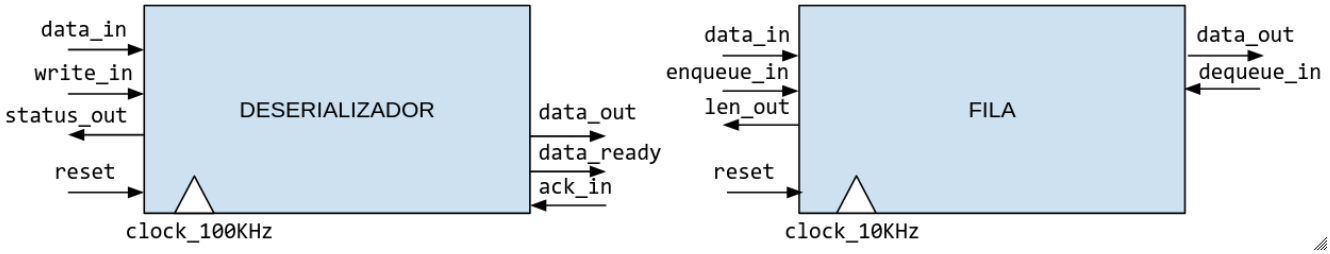
Trabalho 3 (T3) - 2025-1

Objetivo: Exercitar a integração de componentes com múltiplos domínios de relógio em um mesmo projeto de hardware. A organização do trabalho considera as combinações abaixo:

- O trabalho possui 1 entregável: link para o repositório em serviço de versionamento de sua escolha.
    - Entregar APENAS o link para seu repositório via área Moodle da disciplina até a data limite: **05/06/2025 até 23h50**. TODOS OS MEMBROS DO GRUPO DEVERÃO REALIZAR A ENTREGA. Alunos que falharem em entregar o link receberão nota zero. Seu repositório deverá permitir acesso público através do link informado.
    - Projetos sem scripts de execução (e.g., `sim.do`) receberão nota zero. O arquivo README.md do projeto substituirá o relatório nesta atividade. Portanto, projetos sem o arquivo README.md também receberão nota zero. Todos os resultados obtidos, imagens, vídeos e instruções deverão constar neste arquivo.
    - Arquivos não-autorais e/ou gerados por inteligência artificial receberão nota zero e configurarão plágio acadêmico, tanto para o código-fonte, scripts, ou relatório (no caso, o arquivo README.md) .
    - A nota final será atribuída em função da apresentação. Todos os alunos do grupo poderão ser questionados sobre seu projeto. Respostas insatisfatórias e/ou evasivas poderão acarretar na redução da nota creditada.
- O trabalho deverá ser desenvolvido **individualmente** ou em **duplas**.

Enunciado: O trabalho consiste da integração de dois módulos funcionando sobre domínios de relógio diferentes. Estes módulos são um DESERIALIZADOR e uma FILA.

- DESERIALIZADOR: Responsável por receber uma sequência de bits através do sinal **`data_in`** e escrever palavras de 8 bits no sinal **`data_out`**. O sinal **`status_out`** indica se o serializador está em condições de receber dados, enquanto o sinal **`write_in`** indica que o dado deve ser interpretado pelo deserializador. O fio **`data_out`** possui 8 bits e representa a saída de dados do deserializador. Os dados estão prontos para consumo quando o sinal **`data_ready`** está alto. Para confirmar o dado recebido, o sinal **`ack_in`** precisa ser escrito.
- FILA: Representa um container de tamanho limitado do tipo FIFO de 8 bits. Os elementos são inseridos na fila através dos sinais **`data_in`** e **`enqueue_in`**. Para remover um elemento da fila, o sinais **`data_out`** e **`dequeue_in`** deverão ser utilizados. O sinal **`len_out`** possui 8 bits sempre indica o número de elementos da fila. Quando o sinal **`dequeue_in`** sobe, o primeiro dado a ser retirado da fila deve aparecer em **`data_out`** no ciclo subsequente se o número de elementos (**`len_out`**) for maior que zero.



Regras para a implementação:

- [2,5 pontos] Construção do serializador, com testbench, observando as seguintes regras:
    - O serializador deveria receber apenas 1 bit pelo `data_in`. Se o sinal `write_in` estiver alto, o bit recebido deverá ser guardado. Quando houver 8 bits guardados, o sinal `data_ready` deverá estar alto e os bits guardados deverão aparecer em `data_out`, indicando que existem dados para serem transmitidos. Os valores de `data_out` e `data_ready` deverão se manter os mesmos até que o sinal `ack_in` fique alto, indicando que os dados foram recebidos. Enquanto o deserializador não conseguir enviar os dados, deverá manter o sinal de `status_out` alto, indicando que está ocupado. Este módulo deverá funcionar a 100KHz.
- [2,5 pontos] Construção do módulo fila (queue), com testbench, observando as seguintes regras:
    - A fila possui um tamanho fixo de 8 espaços, cada espaço com 8 bits. O sinal `len_out` deve informar o número de espaços utilizados. Para colocar um elemento na fila, o elemento deverá aparecer no sinal `data_in` e o sinal `enqueue_in` deverá estar alto. Para remover um elemento da fila, o sinal `dequeue_in` deve ser levantado e, no ciclo subsequente, o dado removido deverá aparecer no sinal `data_out`. Este módulo deverá funcionar a 10KHz.
- [2 pontos] Conexões dos módulos e criação de um módulo top. A cada 8 bits recebidos pelo deserializador, uma palavra de 8 bits deverá ser colocada na fila. O módulo top deverá receber um sinal de clock de 1MHz. Dois processos internos deverão utilizar o sinal de 1MHz para gerar dois sinais de clock diferentes: um de 100KHz e outro de 10KHz, alimentando os módulos anteriores.
- [1 ponto] Seu testbench deverá demonstrar que o deserializador trava (ocupado, `status_out` alto) quando a fila fica cheia (caso ruim).
- [1 ponto] Seu testbench deverá demonstrar exercitar um cenário onde dados são inseridos no serializador e removidos da fila de forma que a taxa de inserção (e remoção) não cause travamento no deserializador (caso bom)
- [1 ponto] Construção do README.md contendo instruções de como executar o projeto e resultados obtidos.