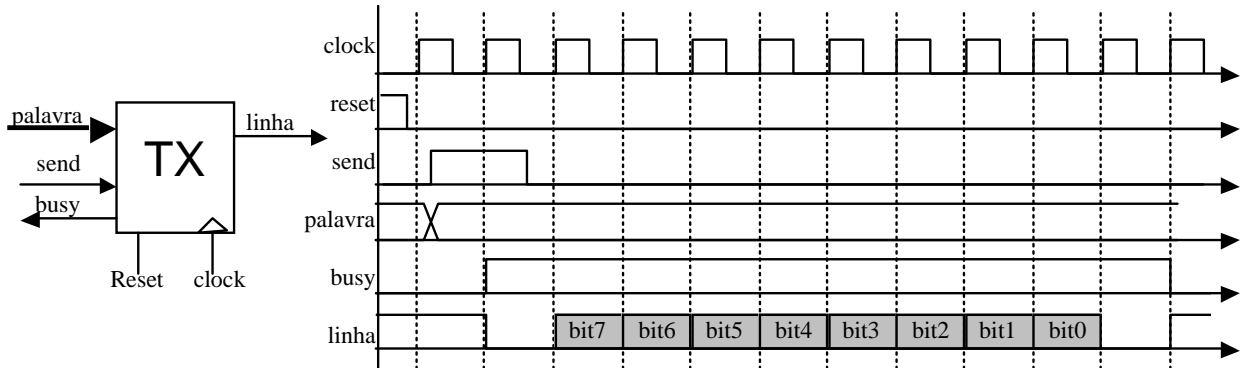


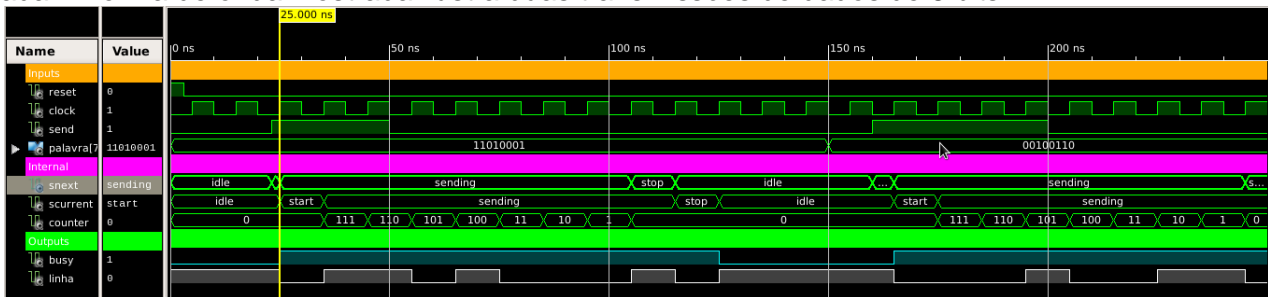
Aluno:

15/julho/2021

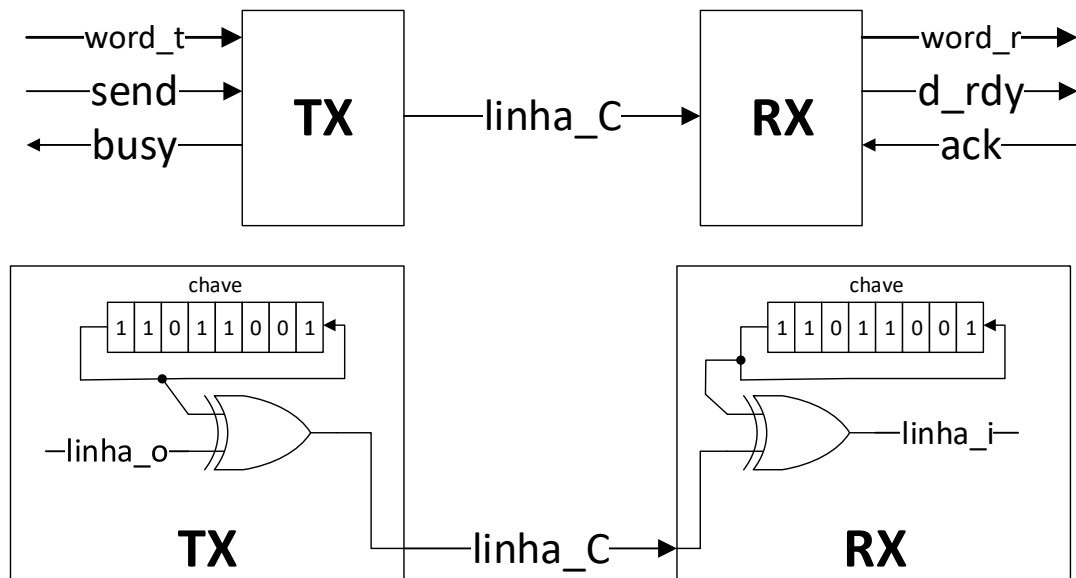
1. (4,0 pontos). Um dos trabalhos da disciplina neste semestre foi implementar um transmissor serial de dados (TX), conforme ilustra a Figura abaixo.



Junto a esta prova encontra-se uma descrição VHDL completa do módulo TX, bem como um *testbench* para ele e um arquivo de configuração de formas de onda. A Figura abaixo mostra a forma de onda capturada ao executar o *testbench* citado. Os alunos devem conferir que esta é compatível com a especificação acima, a menos dos sinais internos adicionais que aparecem na forma de onda dada. A forma de onda mostrada ilustra duas transmissões de dados de 8 bits.



Esta questão versará sobre um sistema de transmissão/recepção serial com características similares ao mostrado acima, ilustrado na Figura abaixo e explicado a seguir:



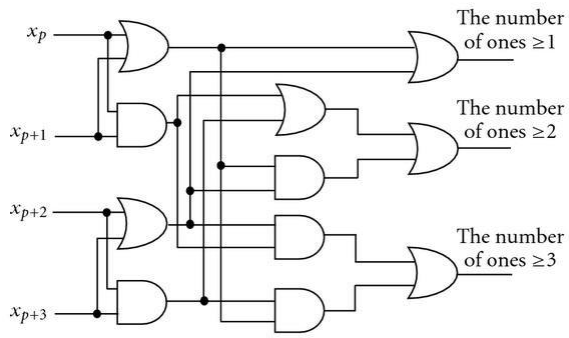
O sistema possui um transmissor (TX) e um receptor (RX) seriais. Ambos, TX e RX são sistemas síncronos com os mesmos sinais **clock** e **Reset**, a exemplo do módulo TX fornecido. Os sinais **word_t** e **word_r** são de 8 bits (8 fios) a exemplo do sinal **palavra** do TX fornecido. Suponha que o fio único que conduz a comunicação serial entre TX e RX (**linha_C**) é um canal de comunicação inseguro. Ele

precisa ser protegido, por isto seu nome é **linha_C** (C de criptografado). O esquema de encriptação usado é simples até demais, e está ilustrado na parte de baixo da figura. Usa-se um registrador de deslocamento realimentado de 8 bits, que contém a mesma chave privada (também chamada de chave secreta compartilhada) em TX e RX e uma porta XOR de duas entradas (Atente para o fato de o registrador de deslocamento realimentado ser síncrono; ele rotaciona os bits da chave um a um. Ele não deve rotacionar bits a todo clock, logo necessita de um processo de controle de rotação.). A figura dá um exemplo de chave. O importante sobre a chave é que ela deve ser a mesma dos dois lados (TX e RX), e é tipicamente carregada no registrador de deslocamento durante o Reset de ambos os subsistemas, TX e RX. Note também que se a chave for “00000000”, não existe encriptação e o canal de comunicação está inseguro. Os sinais **linha_o** e **linha_i** são gerados internamente respectivamente em TX e RX. O sinal **linha_o**, pode-se notar, é equivalente ao sinal **linha** do TX fornecido. Claramente, o módulo RX realiza o trabalho inverso do módulo TX, primeiro descriptografando a informação vinda do canal serial (gerando **linha_i**) e remontando a palavra originalmente enviada (**word_t**), disponibilizando-a em **word_r**. Os sinais de controle **d_rdy** (do inglês *data ready*) e **ack** (do inglês *acknowledge*) proveem um protocolo de comunicação síncrona com quem usa o dado gerado por RX (o ambiente do sistema). Quando **word_r** está pronta, **d_rdy** vai para ‘1’ para avisar o fato. Quem usa o dado **word_t** avisa quando capturou o sinal, colocando **ack** em ‘1’. Depois que **ack** está em ‘1’, eventualmente o sinal **d_rdy** volta a ‘0’ e quando isto acontece, **ack** deve eventualmente voltar a ‘0’ também, e o ciclo de comunicação pode se repetir. Observe que depois que **d_rdy** sobe e até que ambos, **d_rdy** e **ack** retornem a ‘0’ nenhuma transmissão deve iniciar em **linha_C**. Mas, como não há troca de sinais de controle na interface entre TX e RX, isto é uma imposição no ambiente de operação (*testbench*), não no protocolo de comunicação. Como exemplo, imagine que com a chave secreta dada acima, queremos transmitir a letra ‘A’ em código ASCII-E de TX para RX. Ora, como ‘A’ é o vetor de 8 bits “0100 0001” ASCII-E, é fácil definir como este valor vai ser encriptado e colocado em **linha_C**, certo? Verifique que neste caso **linha_C** vai apresentar serialmente o dado “1001 1000” na sua interface, começando com o bit mais significativo. Perceba que este é o código ASCII-E do caractere ~ (veja isto por exemplo em <https://www.ascii-code.com/>). Estamos ignorando aqui os *start* e *stop bits* que respectivamente iniciam e concluem a transmissão. Do lado do RX o tratamento de ~ vai recuperar ‘A’, certo? Deve-se verificar isto

- a) Cada aluno realizando a prova deve desenvolver uma parte do hardware do novo sistema apresentado e um *testbench* para testar o bloco desenvolvido. Cada aluno deve solicitar ao professor em qual parte do sistema trabalhar e qual a especificação detalhada deste.
 - b) Deve-se descrever na resposta da prova como foi desenvolvido o código VHDL e o seu *testbench*.
 - c) Deve-se mostrar uma nova forma de onda de simulação, que deve conter a transmissão ou a recepção de pelo menos dois dados de 8 bits, demonstrando o comportamento do novo circuito.
 - d) Deve-se anexar na resposta os novos arquivos .vhd e/ou .wcfg, identificando-os adequadamente para fins de avaliação.
2. (3,0 pontos). Um circuito combinacional interessante é um que seja capaz de contar o número de dígitos em 1 (ou 0) em uma entrada de múltiplos bits. Referir-se à Figura abaixo. No lado direito, ela apresenta um diagrama de portas lógicas que gera esta informação para vetores de 4 bits. No lado esquerdo ela possui uma tabela de valores de atrasos para cada tipo de porta. Os valores **X** e **Y** serão definidos de forma individual para cada aluno realizando esta prova. Interajam com o professor da disciplina para obter os valores específicos de **X** e **Y** a usar.
- a) Deve-se descrever em VHDL um módulo que implemente este circuito, usando os dados de atraso mostrados na Tabela do lado esquerdo da Figura.
 - b) Deve-se gerar um *testbench* em VHDL para este circuito e deve-se testar sua funcionalidade por simulação da forma mais extensa possível, documentando o comportamento, com formas de onda, nas respostas da prova, com comentários.
 - c) Deve-se responder o seguinte: (1) O circuito pode gerar *glitches* na saída ou não? (2) É possível estimar a velocidade máxima de operação correta deste circuito com os atrasos aqui definidos? Explique como fazer isto e responda isto para o seu caso de circuito.

Deve-se anexar os arquivos .vhd e/ou .wcfg gerados à resposta, identificando-os adequadamente para fins de avaliação.

Porta	Atraso (ns)
Inversor	2
Nand e Nor - 2 entradas	4
Nand e Nor - 3 entradas	7
And e Or - 2 entradas	X
And e Or - 3 entradas	Y
Xor - 2 entradas	5
Xor - 3 entradas	10



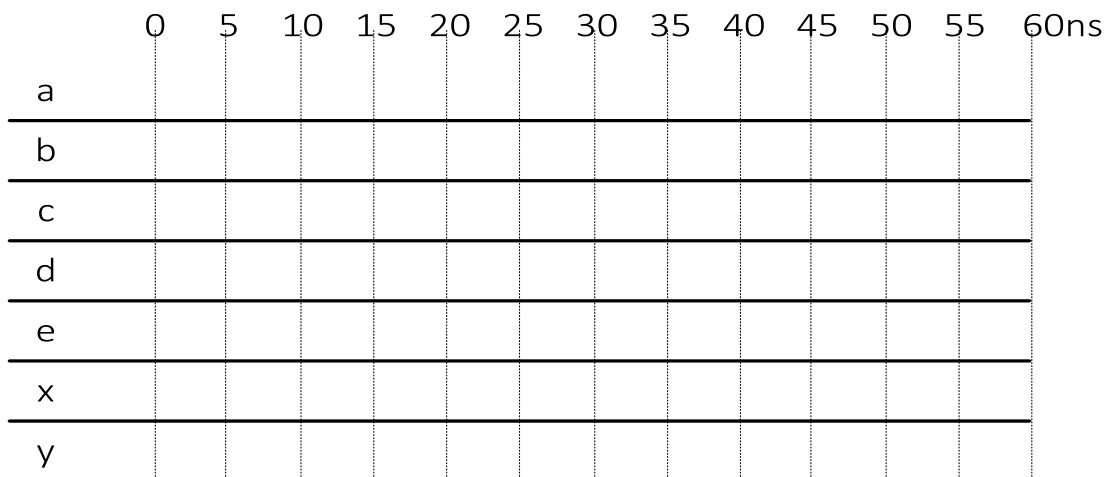
3. (3 pontos). Dada a descrição VHDL abaixo, deve-se fazer o que se pede. Os valores **M** e **N** serão definidos de forma individual para cada aluno realizando esta prova. Deve-se interagir com o professor da disciplina para obter os valores específicos de **M** e **N** a usar.

```

1.  entity Dut is
2.      port (a, b: in std_logic;
3.            c: out std_logic;
4.            d: in std_logic_vector(3 downto 0);
5.            e: out std_logic_vector(3 downto 0));
6.  end Dut;
7.  architecture Dut of Dut is
8.      signal x: std_logic;
9.      signal y: std_logic_vector(3 downto 0) := "0001";
10. begin
11.     x <= a nand b after Mns;
12.     c <= x;
13.     y <= "000" & x after 5ns;
14.     e <= d + y after Nns;
15. end Dut;

```

- a) Deve-se desenhar o circuito acima, usando convenções adequadas para representar o número de fios em cada sinal e para representar o atraso de portas e/ou de fios do circuito;
- b) Deve-se descrever um *testbench* em VHDL para testar este circuito o mais extensamente possível, e simular o mesmo por 60ns;
- c) Deve-se desenhar no diagrama abaixo o que se espera obter como comportamento para cada um dos sinais ao simular o *testbench*. Claro, sempre é possível gerar uma forma de onda via simulação do *testbench* criado e mostrá-la como resposta.



Bom Trabalho!