Escola Politécnica da USP

Departamento de Engenharia de Computação e Sistemas Digitais

PCS2011 Laboratório Digital I

Turma 1 – Prof. Edson Midorikawa

EXPERIÊNCIA 7

TRANSMISSÃO SERIAL ASSÍNCRONA

FELIPE HAMAMOTO TOYODA

MI CHE LI LEE

BANCADA: A-03

DATA: 27/02/2013

1. OBJETIVOS

A experiência visa implementar um circuito de comunicação serial assíncrona usando o padrão RS-232C, em outras palavras, desenvolve-se um circuito digital para a transmissão de dados para um terminal serial, usando a placa de desenvolvimento FPGA da Altera.

2. PROJETO

ESPECIFICAÇÃO DO PROJETO

O projeto da parte experimental consiste em projetar, implementar e documentar um circuito digital que faça uma transmissão serial assíncrona para um terminal, fazendo com que um dado caractere em código ASCII, colocado em 7 chaves de dados seja apresentado no terminal, ao pressionar-se um botão de partida.

O circuito foi projetado de forma a permitir a apresentação do caractere colocado nas chaves a cada acionamento do botão de partida e também gerar o bit de paridade automaticamente (foi-se optado por paridade ímpar).

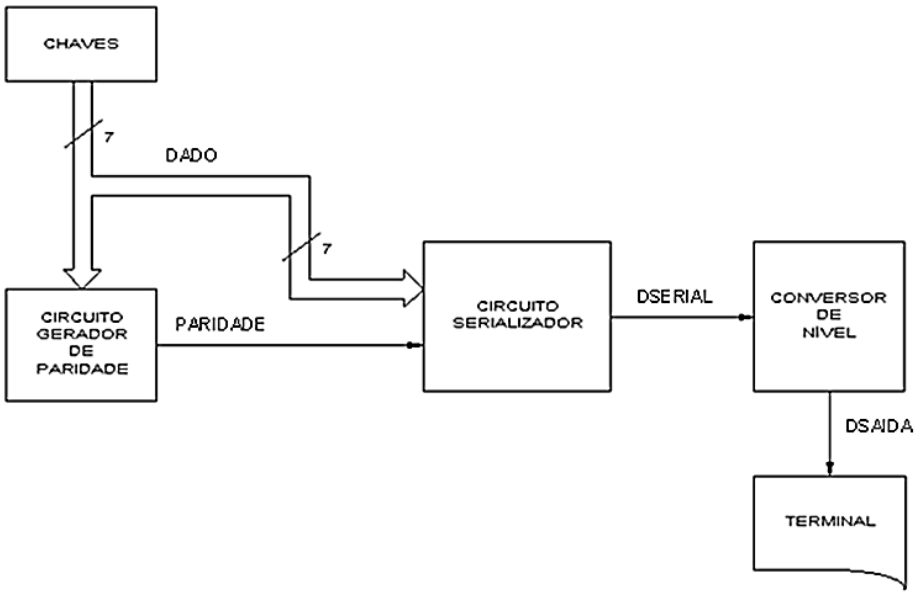

Figura 1 - Diagrama de blocos do circuito de transmissão de Dados

FLUXO DE DADOS

Para o fluxo de dados, precisa-se de deslocadores de entrada paralela e saída em série, pois se tem 6 bits de dados de entrada e uma saída do sistema para o terminal. É necessário que os dados passem por um processo de cálculo de bit de paridade (paridade ímpar), sendo este, colocado numa posição posterior aos bits de dados e antes dos stop bits na mensagem serial após este processo. Ressalta-se também que é muito importante a sincronia da entrada de dados e a sua detecção pelo pulso do start bit, pra isso foi fornecido um divisor de frequência configurável em vhdl para o projeto.

MÁQUINA DE ESTADOS

Para este projeto precisou-se de apenas 3 estados, como descritos a seguir, sendo INIT o estado inicial. Quando o comando para a inicialização é dado, entra-se no estado A, no qual a transmissão serial ocorre até se completar,

parando no estado WAIT, para prevenir a execução do algoritmo repetidas vezes, enquanto o comando de iniciar ainda não é desativado (quando o indivíduo deixa pressionado um botão de início, por exemplo).

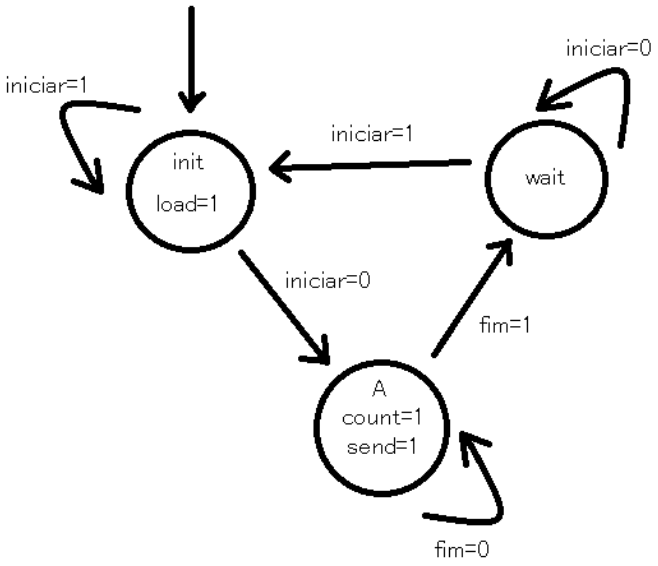

Figura 2 - Máquina de estados

IMPLEMENTAÇÃO

Para a implementação do circuito, optou-se por utilizar 2 registradores deslocadores de entrada paralela e saída serial para a transmissão da mensagem de 6 bits de dados de entrada, 1 bit de paridade, 1 stop bit e 1 start bit. Para o processo de geração de bit de paridade, foi desenvolvida uma descrição estrutural em vhdl (geradordeparidade.vhd), o qual utiliza portas xor (oux2.vhd) e not (inv.vhd). Tratando-se da Unidade de Controle (deslocador_uc.vhd), esta permite a entrada de dados para os registradores de deslocamento e a saída de dados dos mesmos através do sinal INICIAR; inicializa o contador, o qual determina a condição de parada (finalização da transmissão) .

Seguem-se os códigos de programação em vhdl e o diagrama lógico do circuito final:

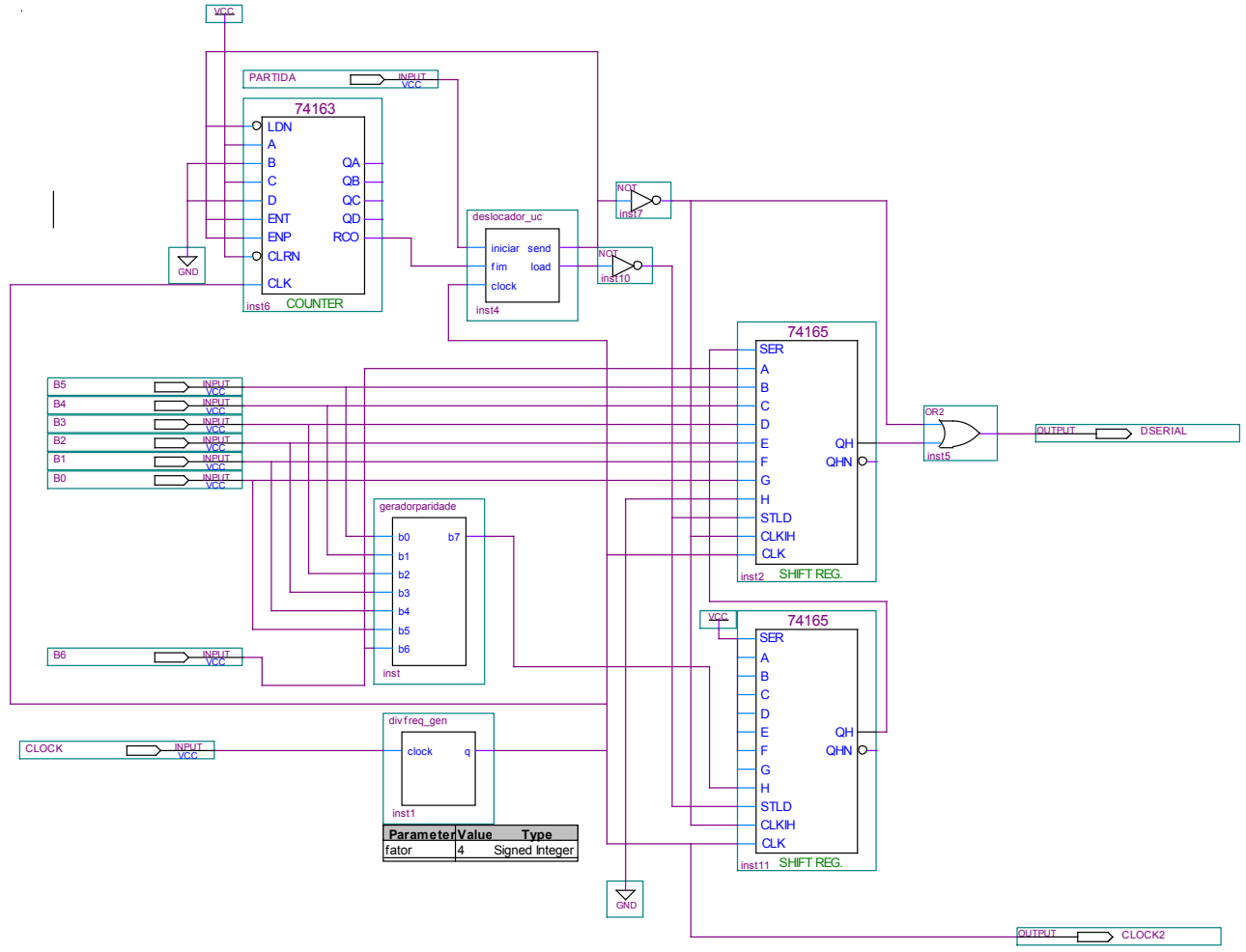

Figura 3 - Circuito completo de transmissão serial assíncrona

INV1.VHD

```
library ieee;
use ieee.std_logic_1164.all;

entity INV1 is
      port(x: in std_logic;
                  y: out std_logic);
end INV1;

architecture yay of INV1 is
begin
      y <= not x;
end yay;
```

OUX2.VHD

```
library ieee;
use ieee.std_logic_1164.all;

entity OUX2 is
      port(x,y: in std_logic;
                  z: out std_logic);
end OUX2;
```

```
architecture yay of OUX2 is
      begin
            z <= x xor y;
end yay;
```

GERADORDEPARIDADE.VHD

```
library ieee;
use ieee.std_logic_1164.all;

entity geradorparidade is
      port(b0,b1,b2,b3,b4,b5,b6 : in std_logic;
                  b7: out std_logic);
end geradorparidade;

architecture yay of geradorparidade is

component OUX2 is
      port(x,y: in std_logic;
                  z: out std_logic);
end component;

component INV1 is
      port(x: in std_logic;
                  y: out std_logic);
end component;

signal n0,n1,n2,n3,n4,n5: std_logic;

begin
      X1: OUX2 port map (b0,b1,n0);
      X2:   OUX2 port map (b2,b3,n1);
      X3:   OUX2 port map (b4,b5,n2);
      X4:   OUX2 port map (n0,n1,n3);
      X5:   OUX2 port map (n2,b6,n4);
      X6:   OUX2 port map (n3,n4,n5);
      I1:   INV1 port map (n5,b7);
end yay;
```

DESLOCADOR_UC.VHD

```
library ieee;
use ieee.std_logic_1164.all;

entity deslocador_uc is
      port(iniciar,fim,clock: in std_logic;
                  send,load: out std_logic);
end deslocador_uc;

architecture yay of deslocador_uc is
type state_type is (init,a,esp);
signal snext, sreg: state_type;
begin
      process(clock)
      begin
            if clock'event and clock='1' then
                  sreg<=snext;
```

```
            end if;
      end process;

      process(iniciar,fim,sreg)
      begin
            case sreg is
                  when init   => if iniciar='1' then snext<=init;
                                       else snext<=a;
                                       end if;

                  when a      => if fim='0' then snext<=a;
                                       else snext<=esp;
                                       end if;

                  when esp    => if iniciar='0' then snext<=esp;
                                       else snext<=init;
                                       end if;
            end case;
      end process;


      with sreg select
            send <= '0' when init | esp, '1' when a;

      with sreg select
            load <= '1' when init, '0' when a | esp;


end yay;
```

CARTAS DE TEMPO

Os testes que serão realizados no circuito na placa FPGA serão baseados nas cartas de tempo das simulações no Quartus II:

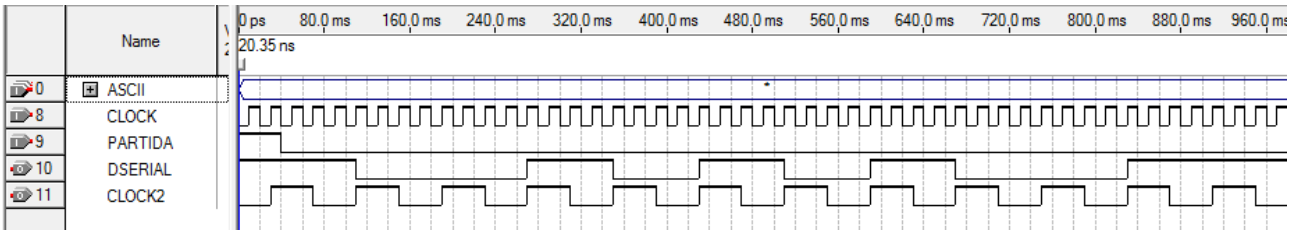

Figura 4 - Carta de tempo - Caractere usado: *

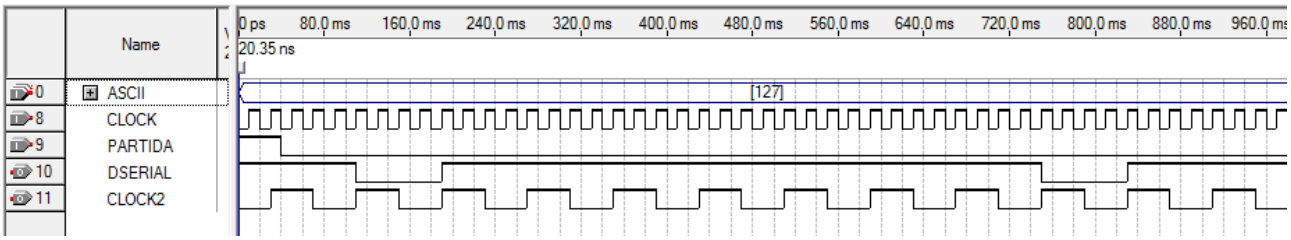

Figura 5 - Carta de tempo - Caractere usado: [DEL]

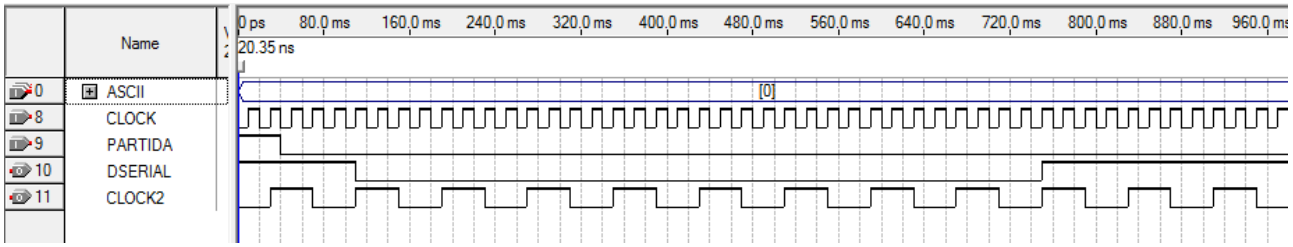

Figura 6 - Carta de tempo - Caractere usado: [NULL]

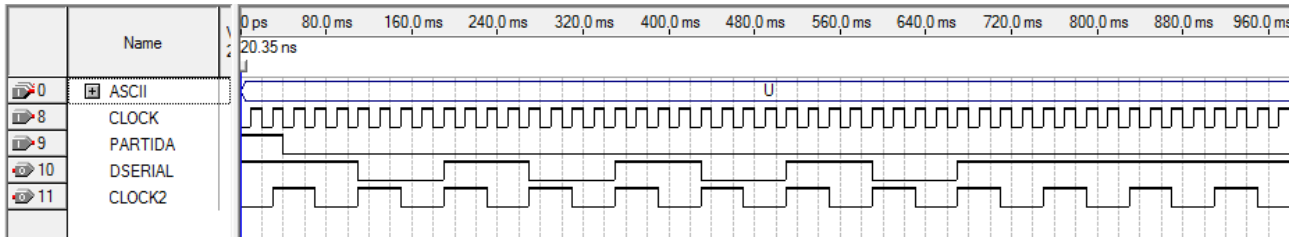

Figura 7 - Carta de tempo - Caractere usado: U

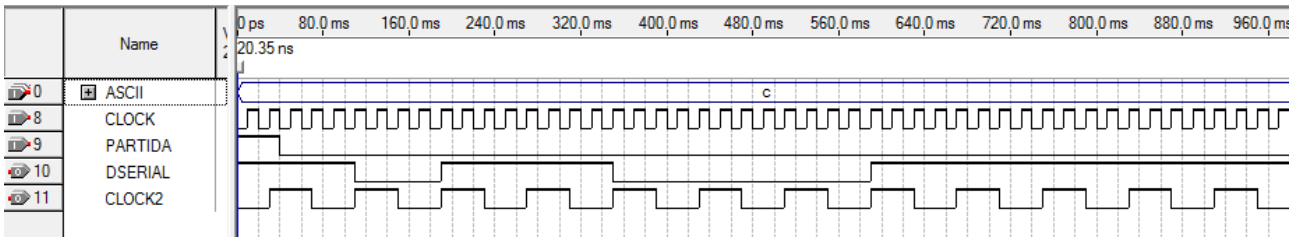

Figura 8 - Carta de tempo - Caractere usado: c

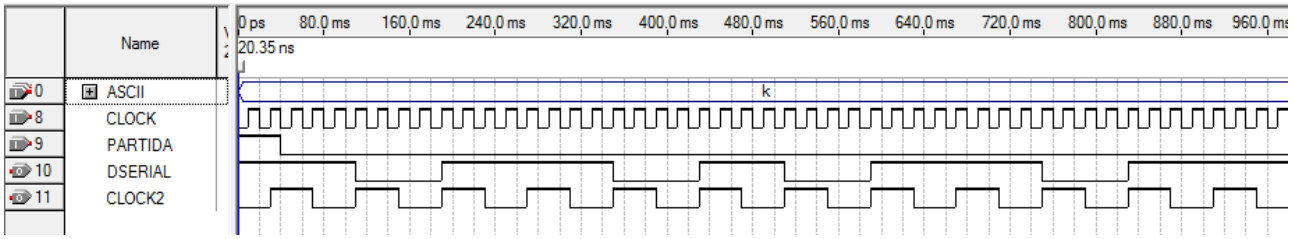

Figura 9 - Carta de tempo - Caractere usado: k

3. ATIVIDADES EXPERIMENTAIS

Tabela 1- Identificação dos pinos

Nome do Sinal	Pino	Descrição
UART_RXD	PIN_C25	UART Receiver
UART_TXD	PIN_B25	UART Transmitter

Tabela 2 - Tabela de fatores de divisão de frequência

Baud Rate	Fator de Divisão	Frequência do Clock (Para o bom funcionamento do circuito)
110		
300		
1200		
9600		
19200		

4. RESUMO DAS ATIVIDADES EXECUTADAS

5. CONCLUSÕES

6. APÊNDICES

Questões da apostila

A. Há alguma limitação de funcionamento do circuito projetado? Elabore uma discussão sobre as frequências mínima e máxima de funcionamento do circuito.

   R: Sim, a frequência do oscilador deve ser pelo menos 4 vezes mais alta que a frequência do clock do registrador de deslocamento. Se a frequência for menor que isso, é possível que o circuito não detecte o pulso do START BIT.

B. Construa uma tabela mostrando para diferentes baud rates (p.ex. 110, 300, 4800 e 19200 bauds), a frequência do relógio (clock de transmissão) correspondente que é necessária para o correto funcionamento do circuito.

R:

Baud Rate	**Frequência do Clock (Para o bom funcionamento do circuito)**
110	
300	
1200	
9600	
19200	

C. Os dados de teste (caracteres a serem mostrados no circuito) influem nos testes efetuados? Caso afirmativo explique por que.

R: