**Universidade de São Paulo**

Escola Politécnica, Engenharia de Computação
PCS3645 - Laboratório Digital II
Turma 3 - Professor Paulo Cugnasca
Bancada B6

# Experiência 02

Circuitos de Comunicação Serial Assíncrona – Planejamento

| **Nome Completo** | **N USP** |
|---|---|
| Henrique Freire da Silva | 12555551 |
| Mariana Dutra Diniz Costa | 12550841 |

**São Paulo, 11 de Setembro de 2023**

# Sumário

# 1 INTRODUÇÃO

Esta experiência consiste no desenvolvimento de um circuito digital para a comunicação de dados de forma serial e assíncrona, adotando a norma EIA-RS-232C e o padrão de comunicação 7O1. Com o objetivo de projetar tanto a transmissão quanto o recebimento de dados, o projeto compreende uma etapa de familiarização com os mecanismos de envio, seguida da aplicação do circuito de recepção em VHDL, simulação com o *ModelSim*, síntese com o *Quartus Prime* e, finalmente, verificação por meio de testes práticos.

# 2 DESCRIÇÃO DO PROJETO

## 2.1 Estudo do Projeto do Circuito de Transmissão Serial

O conjunto de arquivos *tx_serial_7O1.zip* descreve o circuito de transmissão serial assíncrona, sendo a estrutura do projeto baseada na relação entre o Fluxo de Dados e a Unidade de Controle.

### 2.1.1 Fluxo de Dados

Dois principais módulos adequadamente estruturados e relacionados são os responsáveis pela composição do Fluxo de Dados (entidade *tx_serial_7O1_fd*) do circuito: um Contador e um Deslocador. O primeiro deles conta cada borda de subida do *clock* até o parâmetro definido $M = 12$, reiniciando a contagem e transmitindo uma *flag* de aviso ao ultrapassá-lo. Já o módulo Deslocador recebe um vetor de dados de 11 bits e o desloca uma posição para a direita, adicionando no bit mais significativo a entrada serial 1.

A Figura 1 apresenta o diagrama de blocos do Fluxo de Dados do circuito de Transmissão, a partir do qual é possível compreender a dinâmica estabelecida entre o Deslocador e o Contador. Enquanto o primeiro transmite, a cada ciclo de *clock*, um bit dos dados ASCII configurados ao padrão 7O1, o segundo módulo conta as passagens de *clock* e sinaliza o final do processo de transmissão para o restante do circuito.

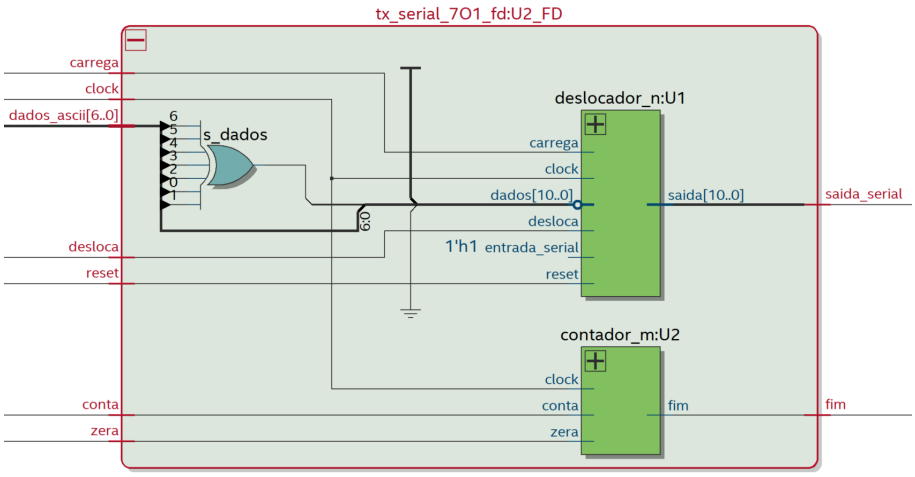


Figura 1: Diagrama de blocos do Fluxo de Dados do circuito de Transmissão Serial.

É importante destacar, também, o papel do Fluxo de Dados na aplicação da especificação 7O1 para a transmissão do código ASCII. É possível notar, inicialmente, que o código a ser transmitido possui 7 bits, enquanto o vetor de dados que efetivamente participa do processo é composto por 11 bits. Conforme representado no trecho de código VHDL da Figura 2, a transmissão sempre começa com um bit de repouso 1, seguida do *start bit* 0 e dos dados do caractere ASCII. Ao final, ainda são passados um bit de paridade ímpar e um *stop bit* 1 para indicar o fim da transmissão.

```
71
72      s_dados(0)          <= '1';    -- repouso
73      s_dados(1)          <= '0';    -- start bit
74      s_dados(8 downto 2) <= dados_ascii;
75      -- bit de paridade da transmissao impar (dados + paridade)
76      s_dados(9)          <= not (dados_ascii(0) xor dados_ascii(1)
77                                  xor dados_ascii(2) xor dados_ascii(3)
78                                  xor dados_ascii(4) xor dados_ascii(5)
79                                  xor dados_ascii(6));
80      s_dados(10)         <= '1';    -- stop bit
81
```

Figura 2: Trecho da descrição VHDL da composição dos dados de Transmissão.

### 2.1.2 Unidade de Controle

A entidade *tx_serial_uc* descreve a Unidade de Controle do circuito de transmissão, que pode ser adequadamente representada pela máquina de estados diagramada na Figura 3, com a exposição de seus devidos estados, condições de transição e saídas dos sinais de controle.

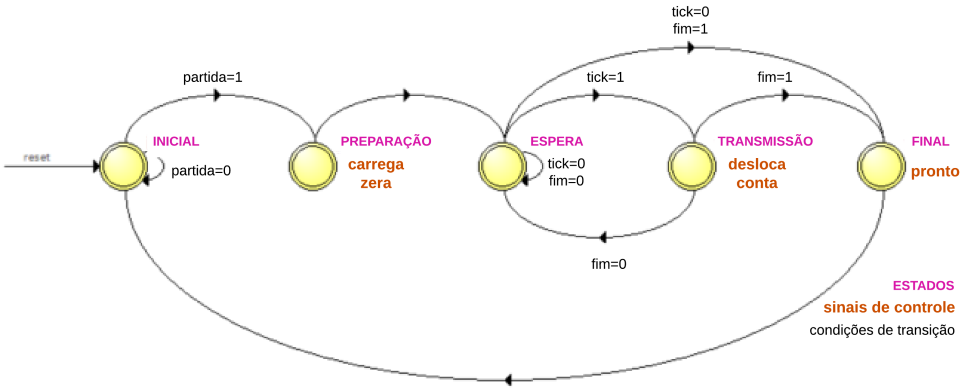


Figura 3: Máquina de estados da Unidade de Controle do circuito de Transmissão Serial.

A partir dos cinco estados presentes na Figura 3, percebe-se a relevância das transições entre "espera", "transmissão"e "final", ditadas pelo sinal de controle *tick*. O *tick* é um sinal periódico que determina o instante em que deve ser iniciada a transmissão de um novo *bit* dentro do processo serial, com o objetivo de controlar e padronizar o intervalo entre o envio de cada um.

Para alcançar esse funcionamento, a arquitetura principal do circuito possui mais um contador que atua como um "gerador de *tick*". O parâmetro $M$ é calculado por meio da divisão da frequência do *clock* do circuito pelo nível de *bauds* (bits transmitidos por segundo) desejado. Como, nesse caso, deseja-se transmitir com 115.200 *bauds* e o *clock* é de 50MHz, o contador é configurado para terminar sua contagem em $M = \frac{50M}{115200} \approx 434$.

Já com o sinal interno de *tick* gerado periodicamente, a Unidade de Controle o utiliza como um Sinal de Controle, ou seja, uma condição de transição do estado de "espera"para o de "transmissão". É esse ciclo de aguardo periódico regular que sustenta as bases da comunicação serial.

### 2.1.3 Teste de funcionamento do circuito

Para verificar o funcionamento do circuito de Transmissão Serial Assíncrona, alguns casos de teste foram definidos em VHDL de modo a formar um *testbench* adequado no arquivo *tx_serial_7O1_tb.vhd*. Esse banco de testes abrange quatro casos de transmissão, para os caracteres ASCII definidos na Tabela 1 a seguir.

| Caractere transmitido | Código ASCII | Paridade esperada |
|---|---|---|
| 5 | 0110101 | 1 |
| U | 1010101 | 1 |
| ~ | 1111110 | 1 |
| DEL | 1111111 | 0 |

Tabela 1: Casos de teste de transmissão definidos no *testbench*.

Com os casos de teste bem definidos e o arquivo de *testbench* fornecido, foi possível simular o circuito com o *software ModelSim* e analisar as formas de onda obtidas. Em cada uma das Figuras de 4 a 7, é mostrado um dos cenários de teste apresentados na Tabela 1. O destaque em verde nas imagens indica qual é o caso que está sendo iniciado, e a onda "Saída Serial"modula de fato a transmissão periódica e sequencial dos *bits* codificados para cada caractere.

Em cada figura, estão destacados os *bits* correspondentes ao padrão de comunicação 7O1, ou seja, o *start bit*, os 7 *bits* de dado, a paridade ímpar e o *stop bit*. É importante verificar que o *start bit* está corretamente posicionado em 0 nos quatro casos, enquanto o *stop bit* segue o valor lógico alto 1 e o *bit* de paridade varia conforme a paridade ímpar prevista na Tabela 1. Além disso, é importante notar que todos os casos de teste finalizam com a ativação da saída "Pronto", que indica o término bem sucedido de cada transmissão serial.

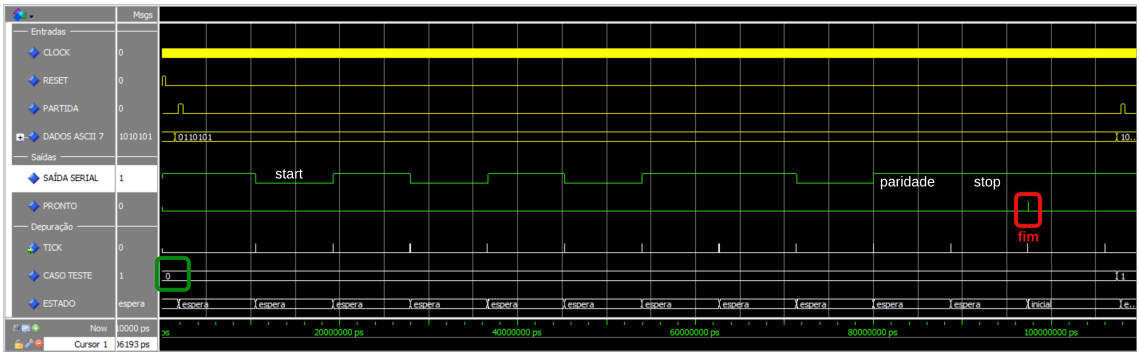


Figura 4: Formas de onda para o teste de transmissão do caractere "5".

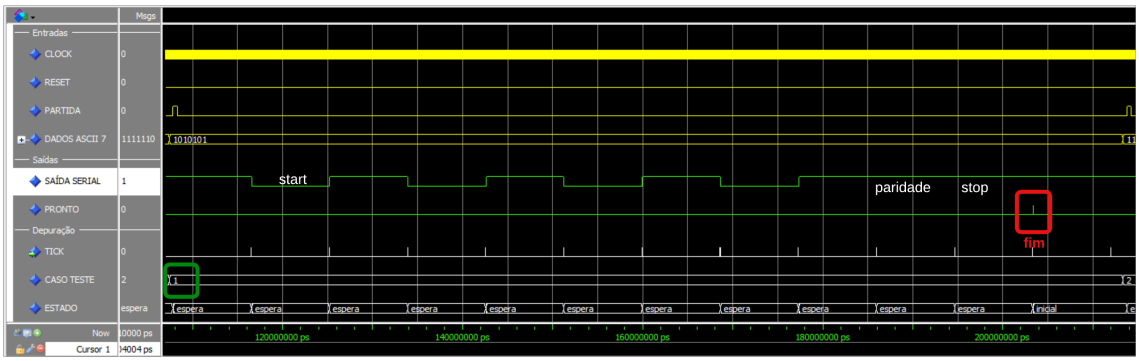


Figura 5: Formas de onda para o teste de transmissão do caractere "U".

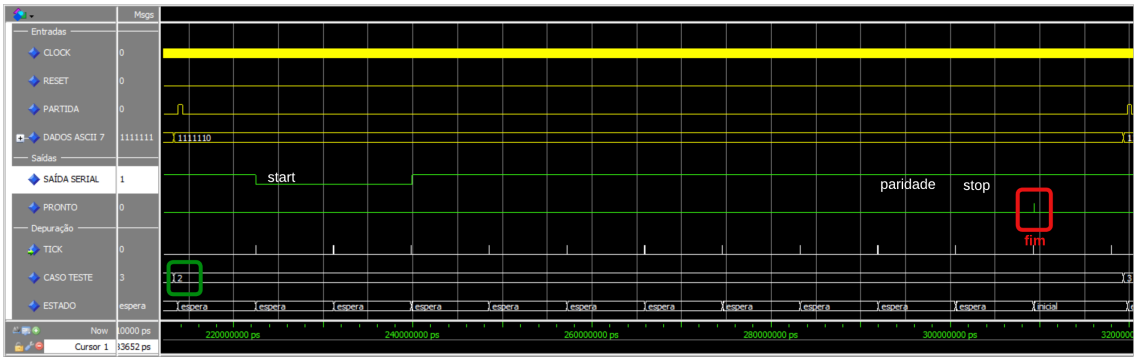


Figura 6: Formas de onda para o teste de transmissão do caractere "∼".

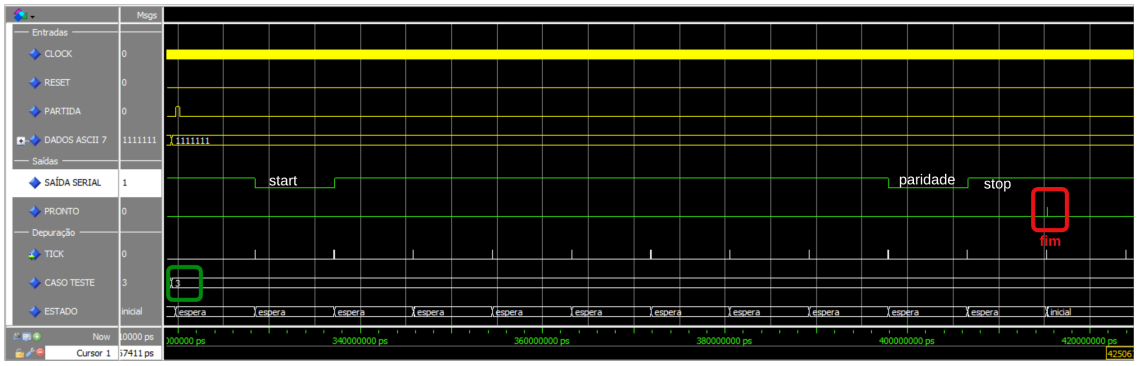


Figura 7: Formas de onda para o teste de transmissão do caractere "DEL".

É possível perceber, também, os *ticks* periódicos que estão sendo gerados pelo contador externo ao Fluxo de Dados e atuam como marcadores da transmissão recorrente do código. Por meio da medida de tempo entre os cursores da Figura 8, pode-se verificar que o intervalo entre a transmissão de cada *bit* segue a especificação previamente definida de $\frac{1}{8680000ps} \approx 115200$ *bauds*.

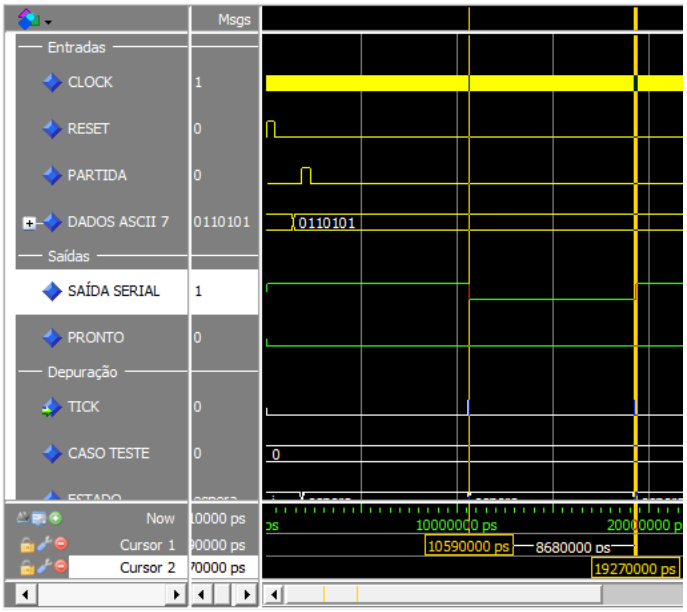


Figura 8: Medida do intervalo entre a transmissão de cada $bit = 8680000ps$.

## 2.2 Projeto do Circuito de Recepção Serial

O circuito de recepção serial corresponde ao dual do transmissor. Ele foi desenvolvido também com a separação em módulos do fluxo de dados e da unidade de controle integrados em uma arquitetura estrutural e gerando um sinal próprio para a amostragem do canal de transmissão, discorrido na seção de 2.2.3.

Na Figura 9, é possível visualizar a conexão dos componentes no *top level* do projeto, em que estão contidos a unidade de controle; o fluxo de dados; o gerador do sinal de amostragem (contador) e os *displays* de depuração dos dados e estado interno do sistema.

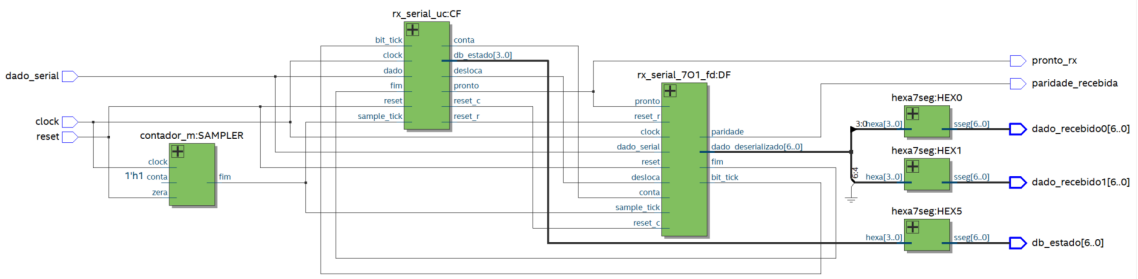


Figura 9: Diagrama RTL do circuito de Recepção Serial sintetizado.

### 2.2.1 Unidade de Controle

Em um circuito simplificado, no qual a taxa de transmissão se iguala à frequência do *clock* interno, inicialmente se esperaria pela sinalização de início da transmissão ao abaixar o nível lógico da linha, transitando para um estado de recepção dos bits, seguido de um estado final para sinalizar a prontidão do dado e, finalmente, retornar ao estado inicial de espera.

Enquanto no estado de recepção, um sinal de controle para contagem do número de bits recebidos ficaria ativo e performaria o incremento no contador do fluxo de dados a cada ciclo de *clock* (nesse modelo equivalendo também à chegada de um novo bit de dado).

A unidade de controle efetivamente implementada não depende diretamente do *clock* do sistema na definição de um próximo estado, apenas de dois sinais gerados a partir dele: o de tempo de bit (*bit_tick*) e o de tempo de amostra (*sample_tick*), conforme o código no Apêndice A e o diagrama da Figura 10 da máquina de estados sintetizada.

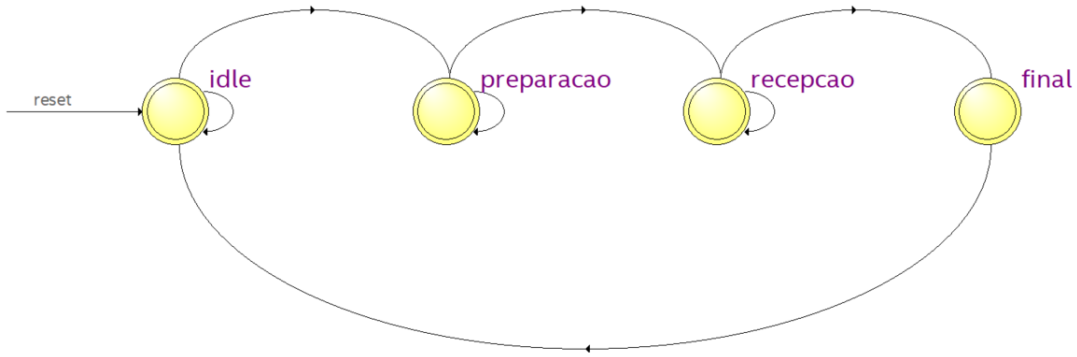


Figura 10: Maquina de estados da Unidade de Controle do circuito de Recepção Serial.

O *bit_tick* corresponde ao sinal de frequência igual à transmissão serial, porém defasado em aproximadamente $180^{o}$, **i.e.** gerando pulsos no ponto médio do intervalo entre pulsos de transmissão de dados. Já o *sample_tick* corresponde ao sinal de frequência múltipla inteira do *bit_tick*, tal que a todo período de bit seja executada um número fixo de amostragem, o qual é parametrizado na descrição elaborado do dispositivo.

### 2.2.2 Fluxo de Dados

Adotando novamente o modelo de transmissão de um bit por ciclo de *clock* do receptor, os principais componentes que compõem o módulo são: um contador de dados recebidos e dois registradores - um deslocador, para o registro dos bits à medida que são recebidos; e outro de carga paralela, para armazenamento do último dado enviado.

O contador mantém o registro do número de bits (dados, paridade e bit de parada) já recebidos, de forma a sinalizar a troca de estado quando atingir seu fim (i.e. dado presente). O deslocador, a cada ciclo de *clock* no estado de recepção, insere o dado amostrado na linha de transmissão à esquerda do dado já armazenado, de forma a gradualmente montar o dado transmitido e para carga parelela no registrador que contém o dado para leitura, isso quando for sinalizado o fim da transmissão pela unidade de controle.

No circuito proposto para transmissão a taxas inferiores à frequência de *clock*, há a inclusão de um contador extra, que se referencia na taxa de amostragem para gerar um pulso a cada período de bit, usando o sinal "meio"para defasear o sinal em relação ao transmissor; e um detector de borda para tornar esse pulso a princípio com largura de $L = \frac{período\ de\ bit}{n\ de\ amostras}$ para $20ns$ (um ciclo de *clock* de $50MHz$).

Na Figura 11 percebe-se a conexão desses componentes. No canto superior direito, em azul, tem-se os registradores dos dados recebidos, alimentados diretamente do registrador de deslocamento; também se encontram os contadores - *bit_ticker* e *data_counter* para gerar o *tick* de transmissão e

contar o número de bits recebidos, respectivamente - e, no canto inferior direito, tem-se o detector de borda do pulso do *bit_ticker*.

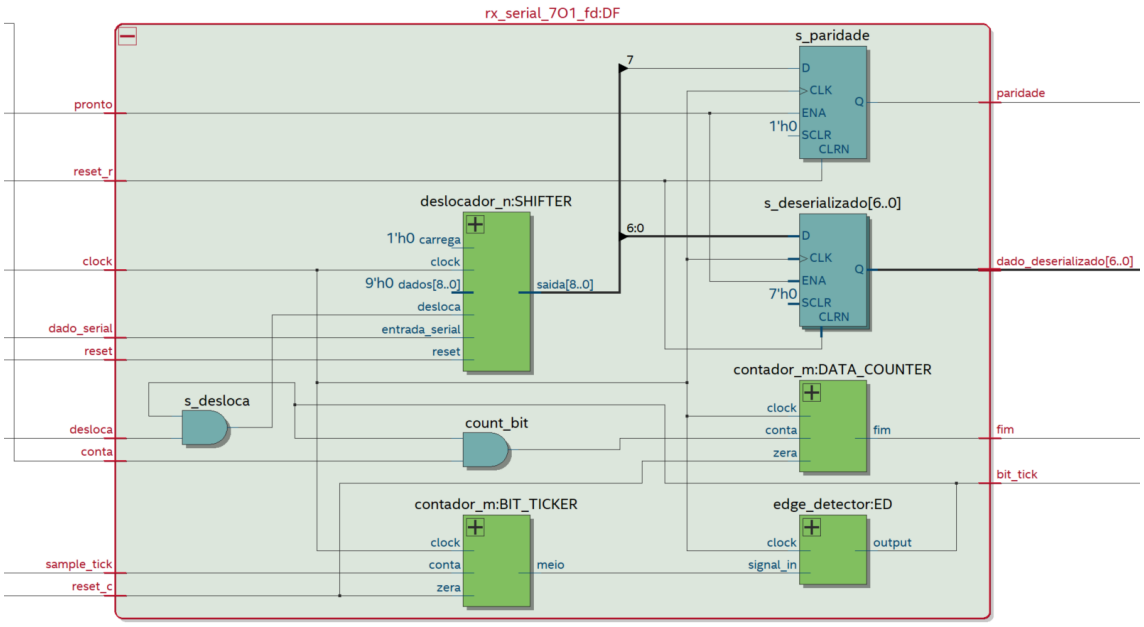


Figura 11: Diagrama de blocos do Fluxo de Dados do circuito de Recepção Serial.

### 2.2.3 Superamostragem

A técnica de superamostragem foi aplicada ao projeto do receptor de forma a garantir melhor alinhamento da leitura ao meio tempo do bit enviado. Como o nome indica, a superamostragem refere-se à amostragem exaustiva de um mesmo dado, ou no caso, em um mesmo período de bit.

Ao se iniciar a transmissão serial, a linha é chaveada para o estado lógico baixo, porém a velocidade com que vai ser verificado esse *start bit* deve ser superior à taxa de transmissão por se tratar de uma operação assíncrona, em que cada ponta da comunicação tem um *clock* interno próprio. O valor base adotado no projeto é de 16x a frequência de transmissão (115200 bauds), porém o módulo pode ser instanciado com uma constante de amostras por período de bit diferente desde que seja inferior ou igual a 434, limiar em que utiliza-se o próprio *clock* do receptor na amostragem. Valores maiores trazem maior precisão no alinhamento com o dado transmitido, visto que a contagem é baseada em múltiplos inteiros do período de *clock*.

Como mencionado na seção 2.2.1, o sinal de *clock* era a referência para amostragem, assim não necessitando do alinhamento ao meio do período de transmissão, o que implica que a solução nesse caso dispensaria o estado "preparacao"do Apêndice A, podendo sair do "idle"direto para a "recepcao", com os devidos ajustes no fluxo de dados para contabilizar o *start bit* no contador indicado na seção 2.2.2.

Dessa forma, a frequência do *tick de bit* é a mesma que a do transmissor (115200Hz) e a frquência de amostragem é 16 vezes esse valor, assim se aproximando de 1,84MHz.

### 2.2.4 Simulação do Circuito de Recepção Serial

De forma análoga ao circuito transmissor, foram propostos testes com caracteres ASCII diversos para testar a recepção dos dados enviados por uma *procedure* no *testbench* que simula o próprio transmissor serial. O banco de testes escolhido é como segue a Tabela 2

| Caso de teste | Caractere transmitido | Código ASCII | Paridade transmitida |
|---|---|---|---|
| 1 | 9 | 0111001 | 1 |
| 2 | k | 1101011 | 0 |
| 3 | @ | 1000000 | 0 |
| 4 | DEL | 1111111 | 0 |

Tabela 2: Casos de teste de recepção definidos no *testbench*.

Vale notar que a sequência de dados enviados não contém *resets* globais do circuito e testam a capacidade do *design* de detectar o primeiro bit de dado, mesmo que quando '0' (igual *start bit*), e de diferenciar a paridade do *stop bit*. As Figuras 12 a 15 apresentam a execução de cada caso de teste listado na Tabela 2.

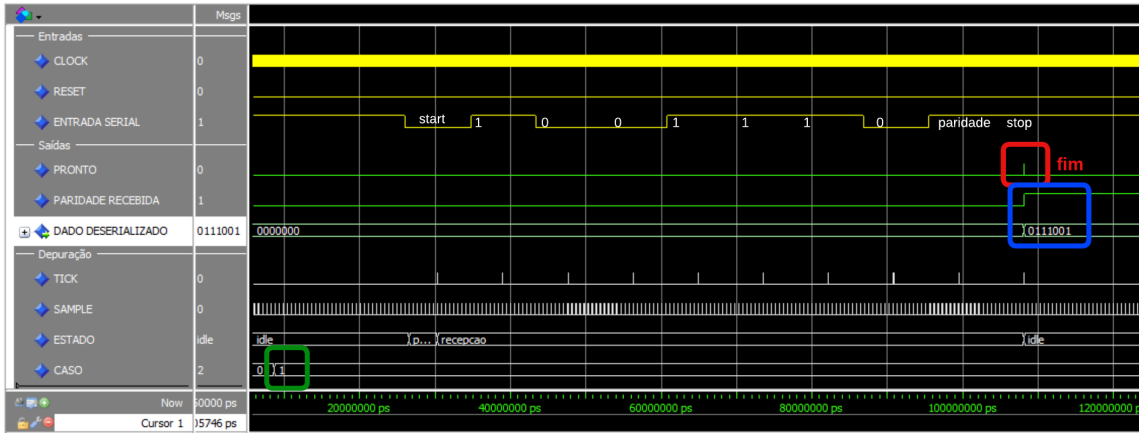


Figura 12: Formas de onda para o teste de recepção do caractere "9".

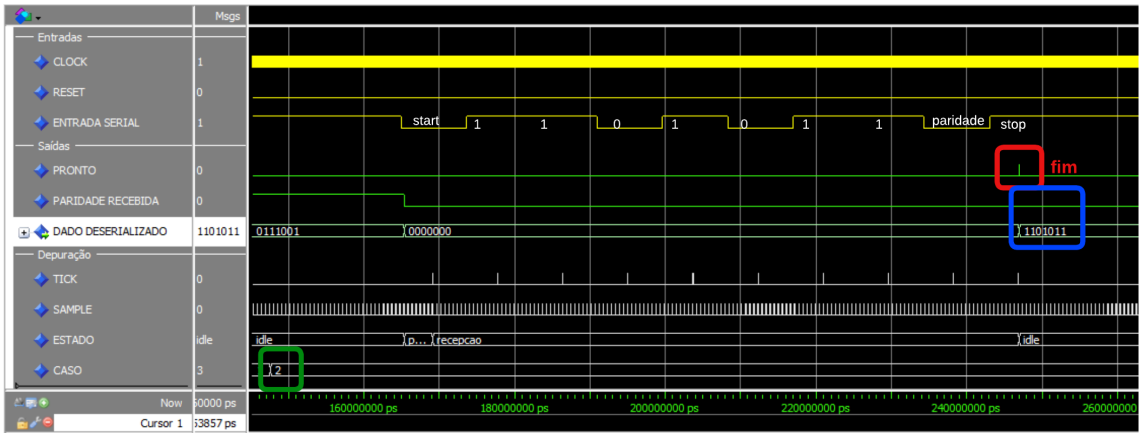


Figura 13: Formas de onda para o teste de recepção do caractere "k".

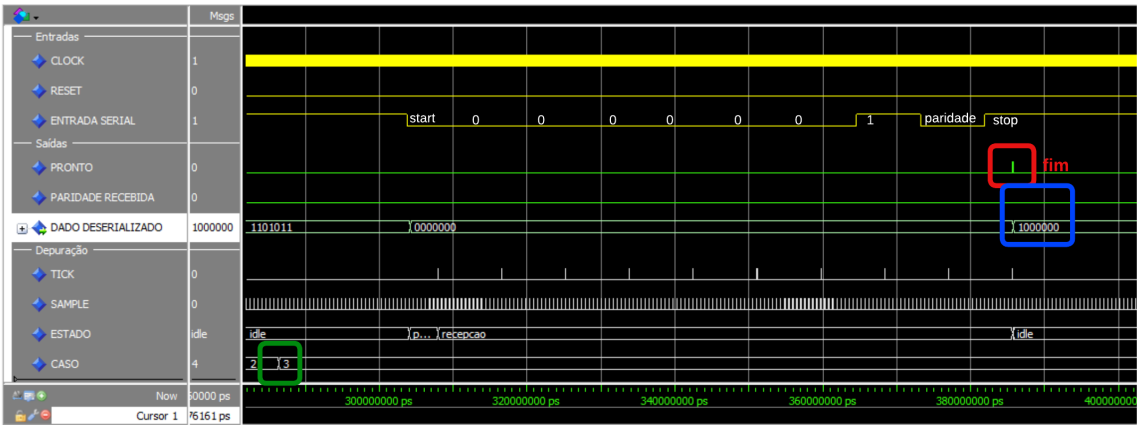


Figura 14: Formas de onda para o teste de recepção do caractere "@".

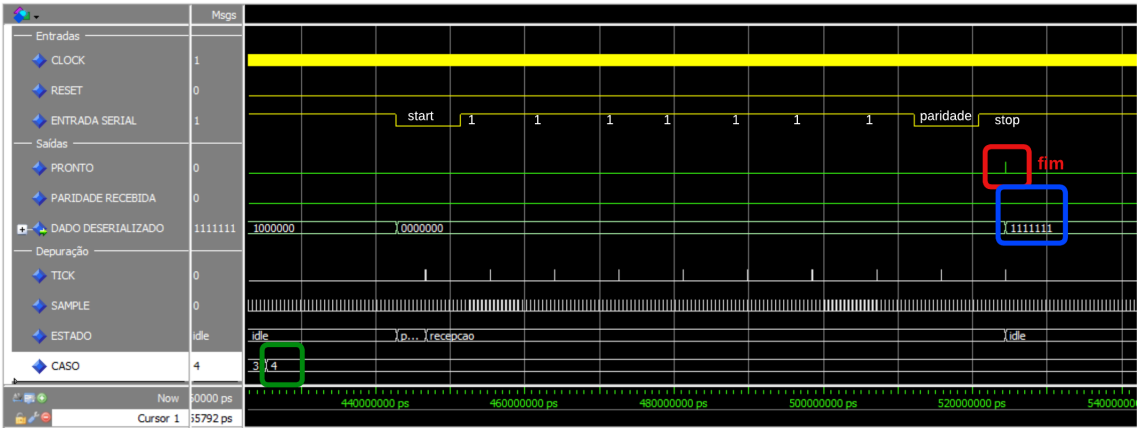


Figura 15: Formas de onda para o teste de recepção do caractere "DEL".

Para cada caso ilustrado, é destacado: o sinal de fim da transmissão (vermelho); o caractere ASCII recebido (azul) e o caso de teste simulado (verde). Além disso, os bits recebidos, de dado e de redundância para transmissão, estão devidamente anotados nas figuras.

Percebe-se que o sinal de amostragem (*sample*) é constante, enquanto *tick* é acionado apenas durante a transmissão, de forma a ser resetado a cada novo envio de caractere e diminuir possibilidade de desalinhamentos.

A paridade e o dado coletado da transmissão são resetados a zero a cada início de transmissão e ficam nesse estado até ela ser concluída. Vale ressatar que o sinal registrado como "dado deserializado"é um sinal interno ao fluxo de dados, mas que apenas é diretamente codificado pelo *display* de sete segmentos para as saídas *dado_recebido0* e *dado_recebido1*.

Na Figura 16, é apresentado em maiores detalhes os dados dos sinais de referência na amostragem do canal serial. A diferença de tempos entre os cursores corresponde a um período de transferência ($\frac{1}{115200}$), o sinal *sample* é aquele constante e 16x a frequência de transmissão, e o *tick* inicia sua contagem com base no *sample* a partir do momento em que é detectada a primeira borda de descida da transmissão.

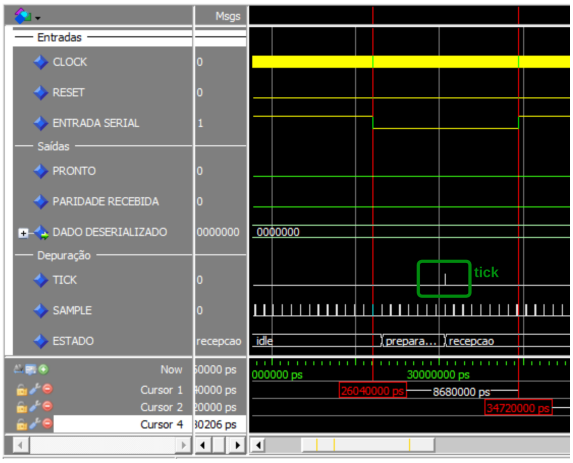


Figura 16: Medida do intervalo entre a transmissão de cada $bit = 8680000ps$.

# 3 PLANEJAMENTO DA AULA PRÁTICA

Em laboratório, será realizada a síntese do projeto arquivado (extensão *qar*) segundo a pinagem da Tabela 3. Além disso, será conectado o GND do Analog Discovery ao respectivo pino na ponte da FPGA.

| **Sinal** | **Ligação na placa FPGA** | **Pino na FPGA** | **Analog Discovery** |
|---|---|---|---|
| clock | CLOCK_50 | PIN_M9 | - |
| reset | chave SW0 | PIN_U13 | - |
| dado_serial | GPIO_0_D1 | PIN_B16 | Protocol - UART - DIO15 |
| dado_recebido0 | display HEX0 | PIN_U21<br>PIN_V21<br>PIN_W22<br>PIN_W21<br>PIN_Y22<br>PIN_Y21<br>PIN_AA22 | - |
| dado_recebido1 | display HEX1 | PIN_AA20<br>PIN_AB20<br>PIN_AA19<br>PIN_AA18<br>PIN_AB18<br>PIN_AA17<br>PIN_U22 | - |
| paridade_recebida | led LEDR[0] | PIN_AA2 | - |
| pronto_rx | led LEDR[9] | PIN_L1 | - |
| db_estado | display HEX5 | PIN_N9<br>PIN_M8<br>PIN_T14<br>PIN_P14<br>PIN_C1<br>PIN_C2<br>PIN_W19 | - |

Tabela 3: Pinagem para a montagem experimental.

Com auxílio da ferramenta *Protocol* do WaveForms, será configurada a transmissão na porta especificada para o envio do dado serial com o protocolo RS232C e comunicação 7O1 a 115200 bauds (aspecto utilizado no projeto em questão). Ao fim dessa etapa, o Analog Discovery será conectado à GPIO da placa para realizar a conexão serial pelo sinal "dado_serial".

A partir disso, a placa DE0-CV será ligada e programada para a execução do plano de testes contido na Tabela 2, com possível adição de novos sinais de depuração a depender do resultado dos testes executados.

Ao fim dessa primeira parte, a conexão com o Analog Discovery será trocada pelo circuito conversor de nível de tensão (MAX3232) para interfacear a saída USB do computador e a GPIO da FPGA.A comunicação serial será controlada com o programa TeraTerm e, após as devidas alterações de montagem, o projeto será reprogramado na placa e retestado segundo os casos de teste anteriores.

# Apêndices

## A Estados e condições de transição da unidade de controle do receptor serial

```
1   process (dado, fim, sample_tick, bit_tick, Eatual)
2   begin
3
4     case Eatual is
5
6       when idle       => if dado='0' and sample_tick='1' then
7                                        Eprox <= preparacao;
8                          else          Eprox <= idle;
9                          end if;
10
11      when preparacao => if bit_tick='1' then Eprox <= recepcao;
12                         else                 Eprox <= preparacao;
13                         end if;
14
15      when recepcao   => if    fim='1' then Eprox <= final;
16                         else               Eprox <= recepcao;
17                         end if;
18
19      when final      => Eprox <= idle;
20
21      when others     => Eprox <= idle;
22
23    end case;
24
25  end process;
```