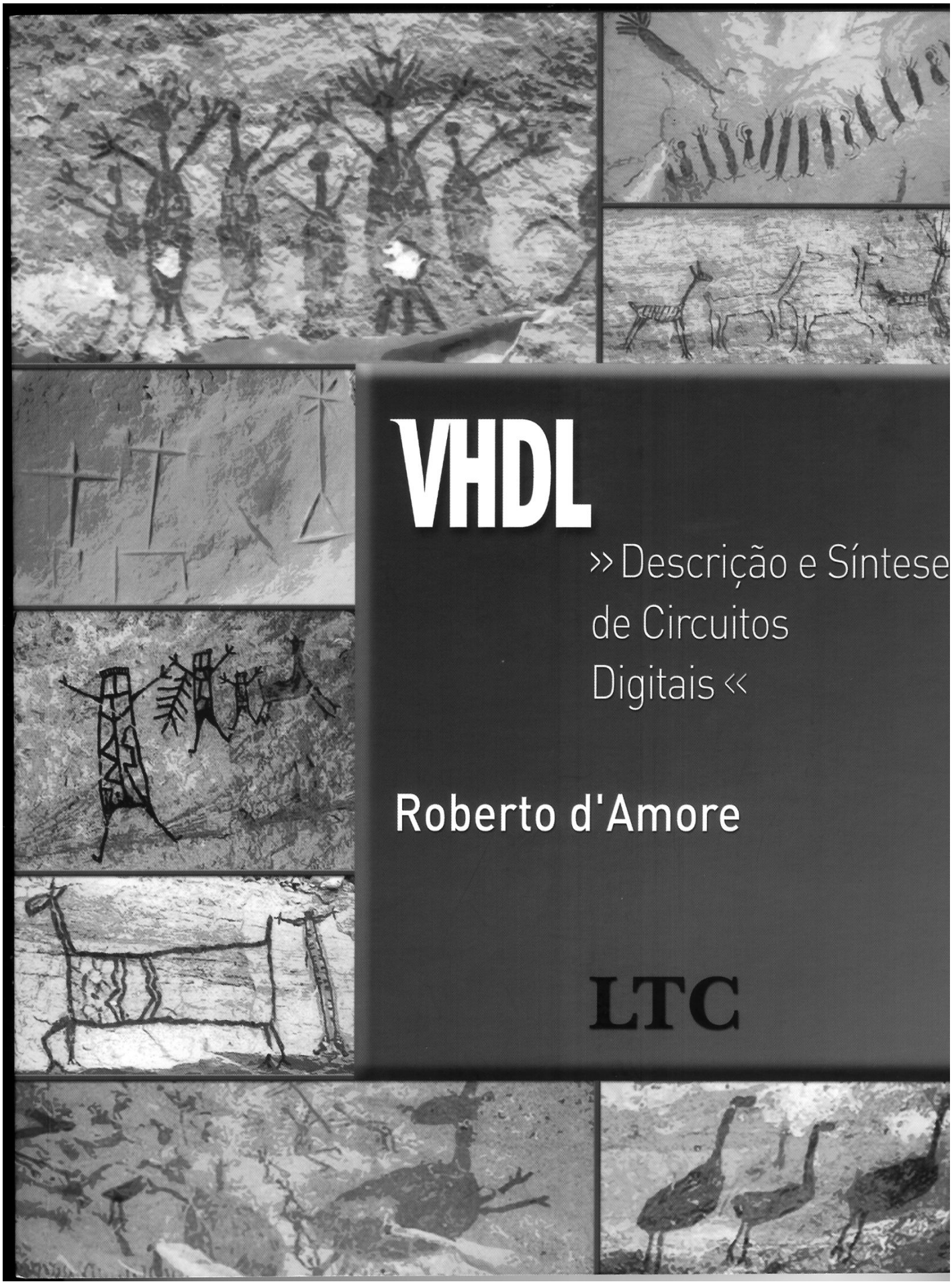
Lista de Exercícios – SEL0632

Capítulo 10



construção " CASE WHEN " não foi empregada no processo de síntese (ver Quadro 10.4.10). A remoção dessa cláusula, entretanto, impede a síntese, devido ao erro de sintaxe, conforme observado anteriormente.

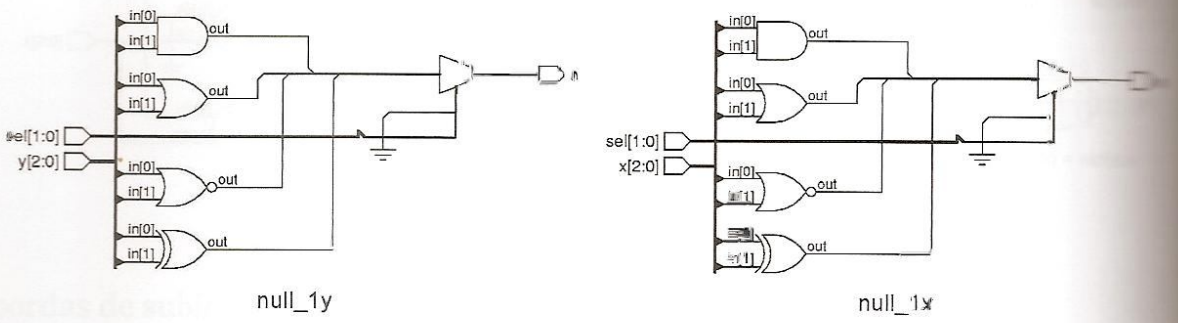


Figura 10.4.7 Resultado da síntese, descrições do Quadro 10.4.9.

```
-- Compiling root entity null_1x(teste)
"\vhdl\null_1x.vhd",line 14: Info, others clause is never selected for synthesis.
```

Quadro 10.4.10 Mensagem gerada pela ferramenta de síntese.

10.5 Principais pontos abordados

- O tipo " std_ulogic " definido no pacote " std_logic_1164 ":
  - pode assumir os valores " U ", " X ", " 0 ", " 1 ", " Z ", " W ", " L ", " H " e " - ";
  - não tem uma função de resolução incorporada;
  - somente um controlador pode acionar um sinal " std_ulogic ".
- O tipo " std_logic ":
  - é um subtipo " std_ulogic " que contém o mesmo conjunto de valores;
  - é declarado com uma função de resolução;
  - mais de um controlador pode acionar um sinal " std_logic ".
- No pacote " std_logic_1164 ", são definidas funções que realizam operações lógicas sobrecarregando as funções lógicas definidas no pacote padrão. Funções de conversão de tipos são, também, definidas.
- As ferramentas de síntese empregam os valores sem um nível lógico válido do modo mais conveniente para a elaboração do circuito.
- A ferramenta de síntese infere a necessidade de portas com saídas de estado de alta impedância pela atribuição do valor " Z ".
- A função " std_match " é mais adequada, caso se deseje considerar o significado correto do valor " - " em uma comparação. Ela está contida no pacote " numeric_std ", apresentado no Capítulo 11.
- As funções " rising_edge " e " falling_edge " devem ser empregadas para a detecção de bordas de subida e descida. Elas permitem manter uma compatibilidade com a simulação e síntese.

10.6 Exercícios

10.6.1 Na entidade " std1164c ", apresentada no Quadro 10.3.2, altere o tipo dos sinais " s " e " s_out " de " std_logic_vector " para " std_ulogic_vector ". Compile a nova descrição e comente o resultado obtido.

10.6.2 Remova a palavra reservada " BUS " na linha 7 da entidade " std1164f " apresentada no Quadro 10.4.3. Compile e simule a entidade conforme os estímulos propostos na Figura 10.4.1. Explique a nova operação do circuito.

10.6.3 Apresente uma descrição para o circuito mostrado na Figura 10.6.1 empregando a construção " WHEN ELSE ".

10.6.4 Apresente uma descrição para o circuito mostrado na Figura 10.6.1 empregando a construção " IF ELSE ".

10.6.5 Apresente uma descrição para o circuito mostrado na Figura 10.6.1 empregando a construção " BLOCK GUARDED ".

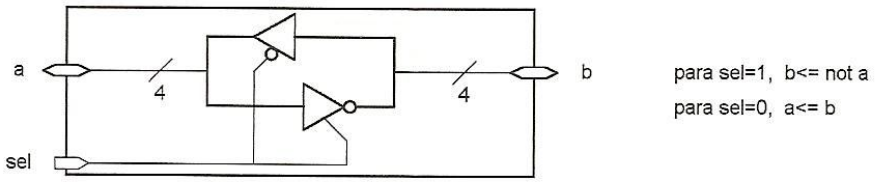


Figura 10.6.1 Circuito dos Exercícios 10.6.3, 10.6.4 e 10.6.5.

10.6.6 Sintetize as descrições propostas para os Exercícios 10.6.3, 10.6.4 e 10.6.5 e verifique se o código foi interpretado corretamente pela ferramenta de síntese.

10.6.7 O código do Quadro 10.6.1 descreve uma máquina de estados para o controle de velocidade. Conforme o valor do sinal " acelera_b ", o valor do sinal " velocidade " é incrementado ou decrementado a cada borda de subida de " ck ". Altere o tipo dos sinais " ck " e " rst " para " std_logic ", e o tipo do sinal " velocidade " para " std_logic_vector ". Considere as implicações na construção " CASE WHEN ".

```
1  ENTITY null_al0 IS
2    PORT( ck, rst     : IN  BIT;
3          acelera_b   : IN  BOOLEAN;
4          velocidade  : BUFFER BIT_VECTOR (1 DOWNTO 0));
5    END null_al0;
6
7  ARCHITECTURE exemplo OF null_al0 IS
8  BEGIN
9    abc: PROCESS
10   BEGIN
11     WAIT UNTIL (ck'EVENT AND ck='1');
12     IF rst = '1' THEN velocidade <= "00";  -- parado
13     ELSE
14       CASE velocidade IS
15         WHEN "00" =>
16           IF acelera_b THEN velocidade <= "01"; ELSE velocidade <= "00"; END IF;
17         WHEN "01" =>
18           IF acelera_b THEN velocidade <= "10"; ELSE velocidade <= "00"; END IF;
19         WHEN "10" =>
20           IF acelera_b THEN velocidade <= "11"; ELSE velocidade <= "01"; END IF;
21         WHEN "11" =>
22           IF acelera_b THEN velocidade <= "11"; ELSE velocidade <= "10"; END IF;
23       END CASE;
24     END IF;
25   END PROCESS abc;
26 END exemplo;
```

Quadro 10.6.1 Descrição de uma máquina de estados — Exercício 10.6.7.

10.6.8 Sintetize a descrição do Quadro 10.6.1 e a descrição proposta para o Exercício 10.6.7. Observe que o processo não tem uma lista de sensibilidade, e o sinal de relógio que controla a máquina é definido no comando " WAIT ". Compare e comente os resultados.

10.6.9 A Figura 10.6.2 contém o diagrama de blocos de uma interface de comunicação serial, e a Figura 10.6.3 ilustra a operação da interface. A comunicação dos dados é feita por uma linha de dados bidirecional denominada " sda ", *serial data*, e o sinal de relógio para o controle da transmissão é feito pela linha " scl ", *serial clock*. A direção da transmissão é definida pela linha " rec_env ", na condição " rec_env=1 " os dados são recebidos pela interface. A cada borda de subida do sinal " scl " um dado é transmitido através da linha " sda ". A leitura ou escrita de dados para o interior da interface na forma paralela é feita pela linha denominada " dado ". O controle das operações com a linha " dado " é feito pelos terminais " ce " e " rd_wr ". O primeiro habilita a interface na condição " ce=0 "; e o segundo define a operação: " rd_wr=0 " escrita, " rd_wr=1 " leitura.

Apresente uma descrição para a interface proposta. No diagrama da Figura 10.6.2 o registrador " ds " armazena os dados a serem transferidos, e o seu controle é executado pelos sinais " ce " e " rd_wr ". O registrador " di " é um registrador de deslocamento controlado por " scl ". A entrada " reset " pode ser empregada para inicializar o controle interno da máquina.

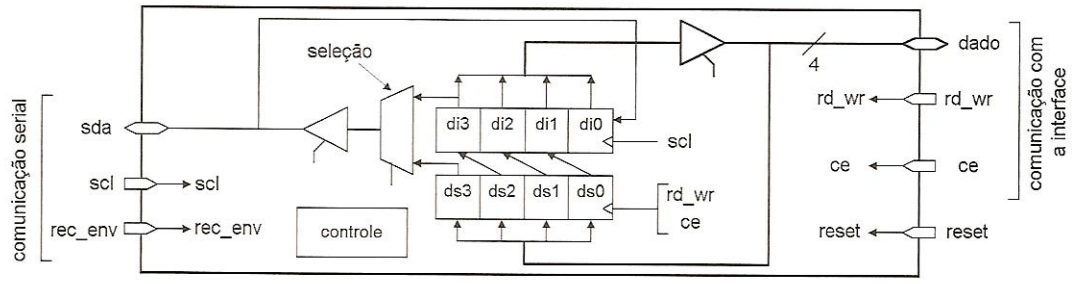


Figura 10.6.2 Diagrama de blocos da interface de comunicação serial.

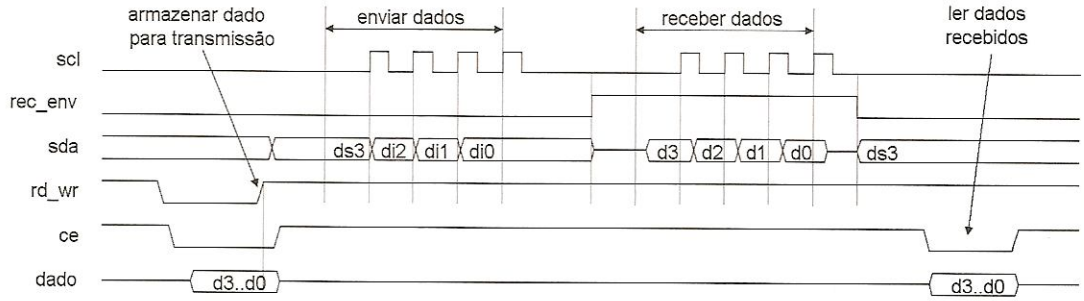


Figura 10.6.3 Esquema de operação e controle da interface.

10.6.10 A linha " rec_env " da interface proposta no exercício anterior pode ser eliminada inserindo-se um mecanismo de controle de início e fim de transmissão, e um bit de dados definindo o sentido da comunicação dos dados. A Figura 10.6.4 ilustra o esquema de controle proposto baseado no padrão I²C de comunicação. Na carta de tempos, a linha " sda " foi dividida em " sda entrada " e " sda saida " para identificar o sentido da comunicação num dado instante. Isto é, a informação está sendo recebida ou transmitida pela interface.

O primeiro dado recebido, ciclo 1, define a operação da interface. Caso esse bit tenha um nível lógico baixo, a interface deve enviar dados; na outra condição, a interface deve receber dados. O início de uma operação é definido por uma transição de um nível lógico alto para um nível lógico baixo na linha " sda " com " scl=1 " (ver ciclo 0). O fim de uma operação é identificado por uma transição de um nível lógico baixo para um nível lógico alto em " sda " com " scl=1 ", conforme mostrado no ciclo 7. Esse mecanismo de detecção de início e fim impõe a seguinte condição na linha " sda ": os dados presentes nessa linha devem permanecer estáveis, se " scl=1 ". Se essa condição não for respeitada, poderá ser identificada uma falsa condição de início ou fim, devido a uma transição na linha " sda ". Devido a esse fato, no esquema de operação proposto os dados são transmitidos a cada borda de descida de " scl ", assegurando, assim, a manutenção da informação na linha " sda " com " scl=1 ".

Proponha uma descrição que receba e transmita dados segundo o esquema apresentado na Figura 10.6.4. Alguns cuidados devem ser tomados para que a descrição seja possível de ser sintetizada. Um elemento de memória não pode ser sensível tanto à borda de descida como a uma borda de subida; assim, a recepção dos dados e a transmissão dos dados devem empregar registradores distintos (ver Figura 10.6.5). O mecanismo de identificação do início de operação e fim de operação sofre do mesmo problema. Um é ativado na borda de descida, e o outro na borda de subida.

Um outro ponto que deve ser levado em consideração é quanto à linha " sda ". Na proposta do padrão I²C, essa linha é do tipo dreno aberto para manter um nível lógico definido nos intervalos que nenhum periférico está controlando " sda " (ver Figuras 10.6.4 e 10.6.5). Note que, no esquema proposto, a linha " sda " é acionada por uma saída com estado de alta impedância. Assim, logo após o ciclo 1, numa operação para enviar dados, a interface " A " libera a linha " sda " para que a interface " B " assuma o controle. Nesse intervalo de tempo que o barramento fica em estado de alta impedância, uma interferência pode causar uma perturbação na linha " sda " levando à detecção de uma falsa condição de início ou fim. Desse modo, é necessário que a implementação do circuito contenha uma resistência para impor o nível lógico alto quando a linha " sda " se encontra em estado de alta impedância. Como uma saída do tipo dreno aberto não é normalmente inferida pelas ferramentas de síntese, a descrição deve empregar uma saída com estado da alta impedância.

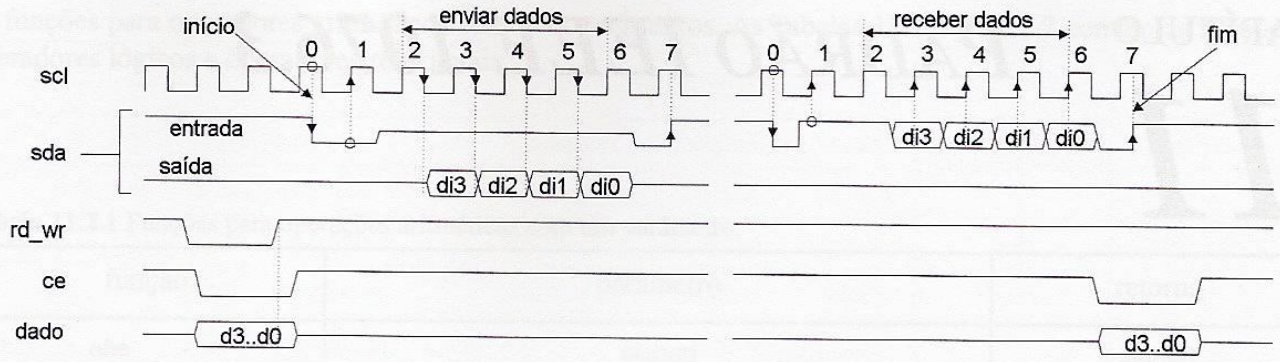


Figura 10.6.4 Esquema de operação e controle da interface II.

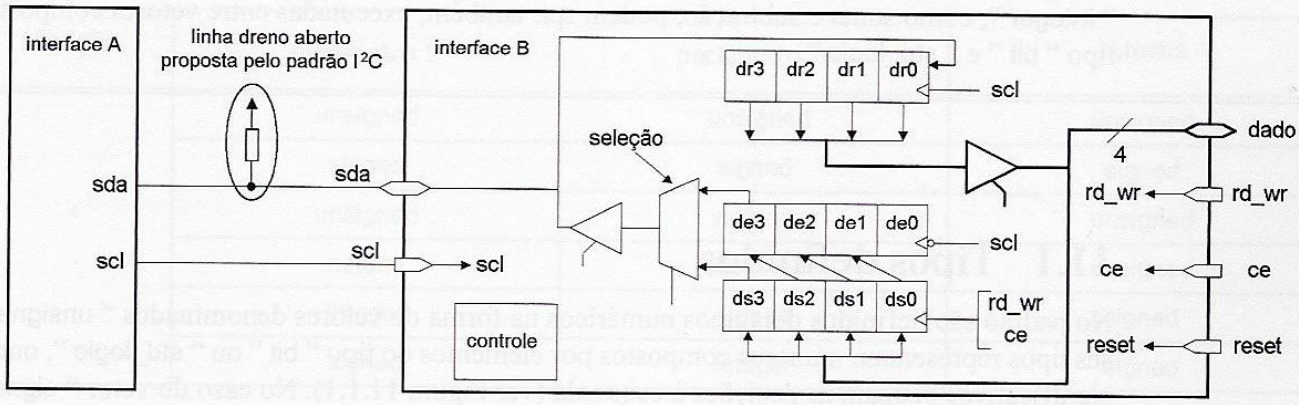


Figura 10.6.5 Diagrama de blocos de uma segunda proposta de comunicação serial.