Tecnologias dos Computadores (2018/2019)

– Trabalho #10 –

Transmissão de Dados Série

Duração: 1 semana

Objetivo

Com este trabalho, pretende-se mostrar como é feita a comunicação de dados série entre dois dispositivos, introduzindo simultaneamente a utilização prática de *flip-flops*.

Módulos Anexos ao Trabalho

- ***filter_keyboard.zip***, disponível no ficheiro .zip fornecido no Material de Apoio da disciplina

Introdução

Existe uma variedade de dispositivos que comunicam entre si transmitindo dados de forma série. Ou seja, existem dois dispositivos interligados pelo menos através de dois fios, sendo os *bits* enviados sequencialmente, até que toda a informação tenha sido transmitida.

Alguns exemplos deste tipo de dispositivos são o rato, o teclado e os periféricos USB (*Universal Serial Bus*).

Vejamos um exemplo prático. Um teclado PS/2 é ligado a um computador utilizando uma ficha MINI-DIN-6, contendo 6 pinos. Existem dois pinos de alimentação do teclado (GND e VCC), um pino de relógio (KB_CLK) e um pino de dados (KB_DATA). Dois dos pinos do conector não são utilizados (N.C.: "*not connected*") – ver figura seguinte.

Figura 1 – Ligações de um teclado PS/2

O computador fornece alimentação ao teclado através dos pinos VCC e GND. O teclado controla as linhas KB_CLK (*keyboard clock*) e KB_DATA (*keyboard data*). Sempre que uma tecla é carregada ou largada, o teclado envia para o computador um código correspondente a essa tecla. Para tal, procede da seguinte forma:

- Normalmente, a linha de KB_CLK está a 1. O teclado certifica-se que esta linha está efetivamente a 1, a fim de poder transmitir os dados[1].

[1] O computador, **que tem sempre precedência**, também pode enviar comandos para o teclado, nomeadamente para controlar os LEDs do mesmo (e nessa altura, obriga KB_CLK a ser 0). O procedimento a partir daí é análogo, mas em sentido inverso.

- Para transmitir os dados, o teclado começa por colocar a linha KB_CLK a 0 durante cerca de 35μs.
- Após a linha KB_CLK ter ido a 0, o teclado envia 10 *bits* em série através da linha KB_DATA, um por cada vertente ativa da linha KB_CLK. O teclado encarrega-se de gerar a onda de relógio em KB_CLK.
- Após os 10 *bits* terem sido transmitidos, a linha de KB_CLK volta a 1, assim como a linha de dados.

O conteúdo dos *bits*, pela ordem que são enviados, é o seguinte:

- ***0*: "*start bit*"**, sempre 0, indicando o início da transmissão de dados.
- ***1*: "*bit de dados 0*"**, o primeiro *bit* do código da tecla.
- ***2*: "*bit de dados 1*"**, o segundo *bit* do código da tecla.
- ***3*: "*bit de dados 2*"**, o terceiro *bit* do código da tecla.
- ***4*: "*bit de dados 3*"**, o quarto *bit* do código da tecla.
- ***5*: "*bit de dados 4*"**, o quinto *bit* do código da tecla.
- ***6*: "*bit de dados 5*"**, o sexto *bit* do código da tecla.
- ***7*: "*bit de dados 6*"**, o sétimo *bit* do código da tecla.
- ***8*: "*bit de dados 7*"**, o oitavo *bit* do código da tecla.
- ***9*: "*bit de paridade (ímpar)*"**, utilizado para detetar erros. Indica que o número total de *bits* a um (1), **incluindo o *bit* de paridade**, é um número ímpar. Caso não seja, houve um erro de transmissão.
- ***10*: "*stop bit*"**, sempre 1, indica o fim da transmissão de dados.

A figura seguinte ilustra o processo.

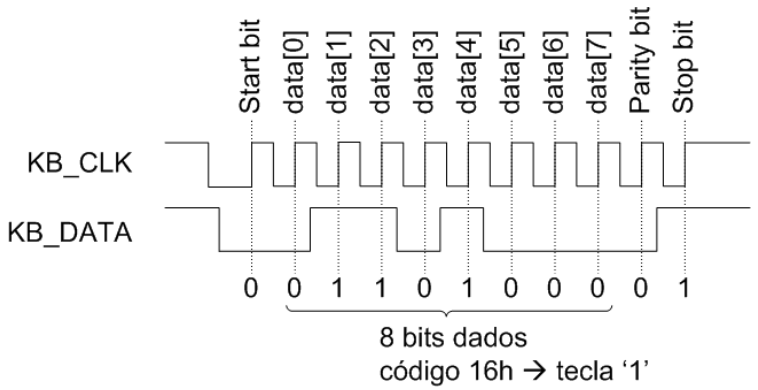

Figura 2 – Transmissão de dados correspondente ao pressionar na tecla '1'

Na documentação anexa ao trabalho encontrará uma tabela contendo os códigos enviados para o computador pelo teclado PS/2. Também encontrará uma descrição mais pormenorizada do funcionamento de um teclado.

Utilize esta tabela no trabalho para se certificar que o mesmo funciona corretamente.

Shift-Register

Imaginemos agora que queremos mostrar num *display* de 7 segmentos o código de uma tecla enviado por um teclado. Para isso, temos de converter os dados no formato série, para um formato paralelo, guardando-os num registo. Para tal, utiliza-se um dispositivo chamado registo de deslocamento ou *shift-register*.

Um *shift-register* é um componente muito simples. Consiste apenas num conjunto de *flip-flops* D em cascata, estando cada saída Q ligada à entrada D seguinte. O sinal de *clock* é comum a todos os *flip-flops* (ver figura seguinte). Repare, ainda, que o *shift-register* é a base dos contadores em anel estudados no trabalho anterior, com a diferença de que a saída do último (mais à direita) *flip-flop* D não está ligada à entrada do primeiro (mais à esquerda).

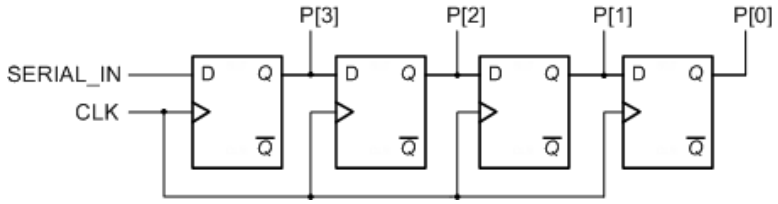

Figura 3 – Shift Register de 4 *bits*

Em cada flanco ascendente do relógio, o valor que se encontra na saída Q de um *flip-flop* é passado ao próximo. Tal significa que se colocarmos um conjunto de dados transmitidos em série na entrada SERIAL_IN, após a primeira vertente de relógio o primeiro *bit* surge em P[3], na segunda vertente esse *bit* passa para P[2], surgindo o segundo *bit* em P[3], e assim sucessivamente, até que o quarto bit enviado estará em P[3], o terceiro em P[2], o segundo em P[1] e o primeiro em P[0]. Se os bits foram enviados na ordem do menos significativo para o mais significativo, isso quer dizer que P[3..0] contém o número correto, sendo P[3] o MSB e P[0] o LSB. A figura seguinte ilustra o processo.

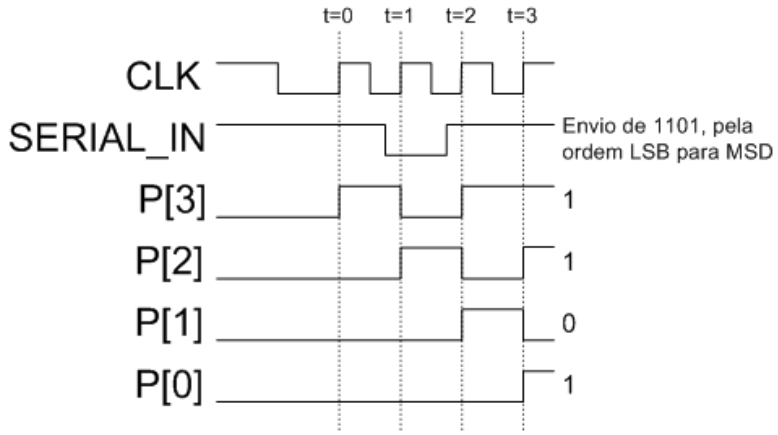

Figura 4 – Entrada do valor 1101 num *shift-register*

Trabalho

Neste trabalho irá novamente utilizar a FPGA **EP2C35F672** existente no kit DE2. A tarefa consiste em programar este dispositivo para mostrar em dois *displays* de 7 segmentos o código hexadecimal que um teclado envia quando se pressiona uma tecla.

Fase 1 (trab06a) – Familiarização com a operação de um teclado

A fim de se familiarizar com a utilização do teclado e com a descodificação para 7 segmentos, faça a montagem indicada abaixo. Note que para fazer a montagem, terá de descompactar o ficheiro *filter_keyboard.zip* para a sua pasta de trabalho. Este ficheiro contém algumas funções que lhe permitirão realizar a montagem.

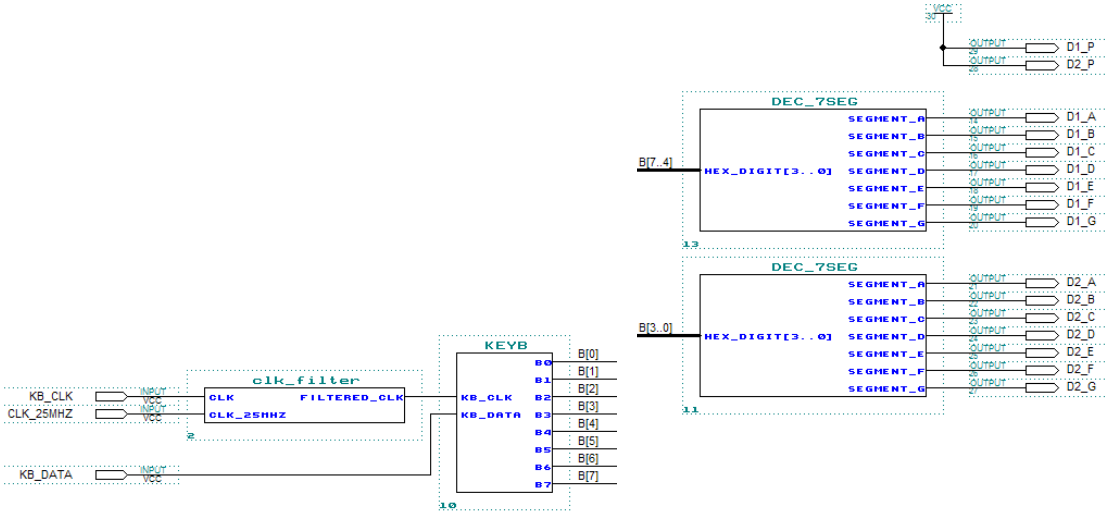

Figura 5 – Descodificador de teclado para *display* 7-segmentos.

O circuito funciona da seguinte forma: o componente KEYB foi definido para este trabalho, recebendo os dados série do teclado, e colocando nas saídas B[7..0] o *byte* correspondente à última tecla carregada. Essas saídas são ligadas a dois descodificadores de binário para hexadecimal em 7-segmentos[2]. Como sabe, é possível converter diretamente um número binário de 8 *bits* para hexadecimal, considerando que um dígito hexadecimal é representado pelos 4 *bits* mais significativos e que o outro é representado pelos 4 menos significativos (consultar os apontamentos da teórica).

O componente CLK_FILTER, embora não seja imprescindível, destina-se a filtrar o sinal de *clock* enviado pelo teclado. Normalmente, em teclados de baixa qualidade, o sinal de relógio é muito ruidoso podendo apresentar "falsas transições". Este componente elimina essas falsas transições. Apesar de não estar representado na Figura 5, utilize o pino correspondente ao relógio de 27 MHz incluído no circuito integrado DE2.

Como em trabalhos anteriores, consulte as tabelas disponibilizadas no Inforestudante para fazer os respetivos "*pin assignments*".

[2] Ou seja, para um número binário de "0000" a "1111", é mostrado o dígito alfanumérico correspondente em hexadecimal.

Experimente o circuito, consultando a tabela de *scan codes* do teclado, que se encontra na documentação anexa ao trabalho. Verifique se os códigos enviados pelo teclado estão corretos.

Fase 2 (trab06b) – Criação do componente KEYB

Após copiar a pasta da fase anterior para uma nova pasta, apague o componente KEYB. Agora, tal como aprendeu no trabalho #3, crie um novo ficheiro gráfico chamado KEYB_DECODE. Esse ficheiro, que será desenhado por si, terá as mesmas entradas e as mesmas saídas que KEYB, realizando a mesma função.

Após a sua implementação, inclua-o no projecto, testando o circuito. Note que, após a sua inclusão e ativação da opção "*Set as Top-Level Entity*", caso pretenda efectuar alguma alteração ao circuito, basta clicar duas vezes sobre o componente, procedendo de seguida às modificações necessárias. Note que o componente a sintetizar deve estar definido como "*Top-Level Entity*".

Para implementar o componente KEYB_DECODE, apenas terá de criar um *shift-register* de tamanho adequado, obtendo os *bits* B[7..0] nas saídas Q dos *flip-flops* que utilizar. Um *flip-flop* D tem o nome "*dff*" no software da Altera. Note que não necessita de aproveitar as saídas dos *flip-flops start-bit*, nem *stop-bit*, mas deve ligar o *bit* de paridade a um LED à sua escolha para verificar a existência ou inexistência de erros na transmissão.

Fase 3 (trab06c) – Interação de LEDs com o KEYBOARD

Ative agora 8 leds vermelhos do kit DE2 (LEDR0-LEDR17), de modo a que estes acendam ou apaguem em função das saídas dos *flip-flops*. Em função da tecla pressionada no teclado, os leds deverão apresentar o *scan code* correspondente.

Antes de dar o trabalho por concluído ou de sair da aula, chame o docente e apresente o resultado das várias tarefas que realizou neste trabalho.

	Tarefas	
1	Implementação do circuito de comunicação PS/2 com visualização em 7-segmentos	□
2	Teste e interpretação do seu funcionamento	□
3	Projecto do componente KEYB_DECODE	□
4	Implementação e teste do circuito com o novo componente, e interpretação do funcionamento do circuito final	□
5	Implementação da funcionalidade que permite ativar LEDs em função da tecla premida	□