universidade de aveiro
theoria poiesis praxis

Relatório - Controlador da intensidade dos 18 LEDs vermelhos

Turma P11 – Ricardo Azevedo (84730) e Fábio Alves (84734)

Universidade de Aveiro - Laboratórios de Sistemas Digitais

07 de Maio de 2017

Introdução

Neste trabalho foi proposto a criação de um código vhdl que quando implementado numa FPGA, através dos botões desta ocorressem mudanças de brilho e que os diferentes leds ligassem ou desligassem com esse brilho. Sendo possível usar protocolos de comunicação entre FPGAs para obter classificações superiores.

Desta forma foi definido que iria-mos usar a base que foi pedida, o protocolo de comunicação, os displays de 7 segmentos e o LCD sendo estes 2 últimos para representar o nível de brilho em que se encontram os leds a operação que esta a ocorrer e um mini menu.

Por fim o trabalho foi dividido em 2 partes sem do a base do trabalho atribuída a Fábio Alves e a comunicação entre kits a Ricardo Azevedo.

Arquitetura

Controlo de leds

Nesta fase foram usados 3 timers, 2 contadores e um comparador de forma a variar o duty-cicle, de acordo com os tempos pretendidos e por um LEDG a piscar.

Timer_1: Este bloco é responsável por garantir que cada mudança só ocorre apos 1 segundo, servindo de start para o bloco responsável por piscar a cada mudança de nível e reset para o contador_1 que vai estar constantemente a contar entre o 5 e 965, de forma a que quando a mudança ocorra não exista uma variação no duty-cicle do ciclo seguinte.

Timer_2: Controla o tempo que o LEDG(8) fica aceso após cada mudança de brilho

Timer_3: É responsável por após 3 segundos -1 ciclo (ou seja no 3º segundo não incluído) com o enable a ser premido(KEY(0) OU KEY(1) não simultaneamente) enviar o sinal para passar a fazer 5 mudanças em vez de 2. Se o nível de brilho for max ou min não envia sinal para 5 mudanças;

Contador_2: Neste contador é retirado se o nível de brilho é máximo ou mínimo de forma a controlar o timer_2 para que não seja reconhecida nenhuma mudança e o time_3 pelas razoes referidas no próprio e ocorre mudanças de 2 ou 5 níveis dependendo do timer_3 e das KEY(0 e 1), mudança para cima ou para baixo, apenas quando o timer 1 se encontra a 1.

Comparador: Compara os 2 contadores de forma a definir quantos ciclos a saída fica a 1 e a 0 variando assim o duty-cicle

Hex

Nesta fase foram usados 2 selecções e um decoder de a representar nos displays de 7 segmentos a operação e o nível.

Seleção_L: Apenas verifica se a tecla clicada foi a KEY(0 ou 1) e devolve os valores que representam AU ou rE nos displays respectivamente.

Seleção_N: Recebe o valor do contador_2 retira 5 unidades (5 apenas para os leds no nível 0 não estarem desligados) e divide-o pelo valor de uma mudança, posteriormente converte o nível de binário para hexadecimal de acordo com a formula num=6+nivel variando aquele 6 de acordo com quantas dezenas o nível tem (para retirar A…F).

Decoder 7_4: Converte os valores devolvidos anteriormente em valores que representam o que é pretendido nos displays.

LCD

Lcd_data: Este bloco guarda na RAM um conjunto de valores que representam os valores que pretendemos escrever no LCD assim como a instrução para escrever cada uma das linhas, estando constantemente a tentar escrever estes valores mas editando os 4 que representam a operação e o nível caso o sinal de aceitar esteja a 1 .Este bloco esta sempre a pedir para escrever.

Lcd_controller: Foi usado o bloco fornecido pelo professor.

Controlo de 1 FPGA

Leds_v0: Neste bloco é usado o valor que é retirado do comparador para em conjunto com as KEYS (2 e 3) fazer o brilho apenas sair nos SW ligados ou se a KEY2 estiver a 0 ligar todos, porem se a key (3) estiver a 0 desliga tudo. O resto das atribuições foi referido anteriormente.

Comunicação

Esta entidade tem como objetivo controlar a comunicação/transmissão de dados entre os dois kits usados. Para tal é usado o protocolo RS-232 utilizando UART. UART é um transmissor/recetor assíncrono universal implementado no kit tendo como principal função a conversão de dados entre as formas paralela e serial.

Após os dados de entrada serem recebidos é formado um vetor de bits com todos os dados de entrada, começando com os dados pinos SW e finalizando com as KEYS. Assim todos os dados poderão ser transmitidos numa única sequencia de bits. Estes irão ser transmitidos para o segundo kit através do pino de saída UART_TXD. No inicio da transmissão é acrescentado um bit com valor 1 no bit mais significativo e 0 no bit menos significativo. Logo de seguida são enviados os dados bit a bit num pequeno processo onde o sinal que contém os dados a ser enviados sofre uma constante deslocação para a direita bit a bit (neste processo é sempre acrescentado o valor 1 no bit mais significativo para ser realizada a deslocação). Como o pino UART_TXD só pode enviar bit a bit isto facilita a transmissão pois será sempre enviado o bit menos significativo que irá conter todos os bits dos dados. No final do envio de cada bit é incrementado num contador que será usado para finalizar a transmissão assim que o seu valor for igual ao total de bits que se pretende enviar. (O total dos bits é uma constante que está implementada em ambos os processos de envio e receção). O bit de valor 0 que foi inserido no sinal será usado no final para verificar se a receção dos dados ocorreu como planeado sendo então esse bit um bit de validação.

A receção dos dados é feita através do pino UART_RXD que, tal como no processo de transmissão, irá receber os dados bit a bit e guarda-los. Também possui um contador que será

incrementado até que este seja igual ao total de bits que se pretende receber. No final é feita a validação, já referida, sendo verificado se o bit menos significativo é 0 e o bit mais significativo é 1. Se tal ocorrer é enviado um sinal de validação com o valor 1 caso tudo tenha ocorrido devidamente ou 0 caso contrario. É retirado de seguida o bit de valor 0 e 1 do sinal onde se encontram guardados os dados. Estes serão encaminhados para a próxima entidade.

É importante também referir que, durante o processo de envio e receção dos dados, é usado um contador que é incrementado ao longo do envio/receção para calcular a duração do envio do bit. Na transmissão serve apenas para determinar quando o envio fui concluído (o contador é igual á duração de envio de um bit, calculada através da divisão da frequência do relógio e a baund_range (nº de bits transmitidos por segundo) que neste caso terá um valor predefinido de 115200.0 (melhor valor possível para a realização deste protocolo). No caso da receção, o contador, além de controlar o inicio e o fim da receção do bit, pode, caso a receção esteja a ocorrer mais rápido do que o previsto, decrementar a meio do processo de modo a manter o envio e a receção a ocorrerem em instantes iguais.

Nota o debouncer utilizado é o disponibilizado pelo stor

Conclusão

O projeto realizado cumpriu com os que nos fui proposto. Sentimos que foi um projeto que nos fez aprender bastante sobre sistemas digitais e programação para FPGA's. Escolhemos este projeto porque queríamos algo simples mas ao mesmo tempo algo que nos pudesse dar um desafio e ao mesmo tempo que tivesse alguma funcionalidade interessante. O nosso objetivo foi completamente alcançado. De tal maneira que, depois de feito, passámos alguns minutos apenas a "brincar" no que tínhamos criado. Não tivemos grandes dificuldades a realizar o pretendido ou mesmo nos extras que decidimos acrescentar. Conseguimos com que o projeto fosse feito de forma faseada, tal como pretendido, e cumprimos assim as datas impostas.

Manual do utilizador

FPGA_1 → FPGA_2 ou FPGA_1 ← FPGA_2

SW[17..0] – cada SW liga o respectivo LEDR no nível em que se encontra a intensidade.

KEY[0]- Aumenta 2 níveis de intensidade no 1º e 2º segundos e depois 5 por segundo durante o tempo que a tecla seja premida. Cada mudança faz o LEDG[8] piscar durante 1/10 segundos.

KEY[1]- Reduz um 2 níveis de intensidade no 1º e 2º segundos e depois 5 por segundo durante o tempo que a tecla seja premida. Cada mudança faz o LEDG[8] piscar durante 1/10 segundos.

KEY[2]-acende todos os LEDR mesmo que o SW esteja desligado no nível em que se encontra a intensidade.

KEY[3]- apaga todos os LEDR mesmo que o SW esteja ligado.

O nível e a operação aparecem nos segmentos de 7 bits e no LCD da FPGA onde ocorre a mudança.

No LCD existe um mini manual do efeito das KEY :

(0.+) KEY[0] - aumento

(1.-) KEY[1] - redução

(2.1) KEY[2] - all on

(3.0) KEY[3] - all of

Divisão do trabalho

Fábio Alves – 84734

Blocos criados:

1- Timers

2- Contadores

3- Seleções

4- Comparador

5- decoder

6- Leds_v0

7- lcd_dados

8- Simmulações

Ricardo Azevedo-84730

Blocos criados:

1- rs232_controller.vhd

2- debouncer.vhd

3- Leds_v1.vhd