

Computação Reconfigurável

Aula prática 1

Resumo:

1. Introdução ao sistema Vivado (versão 2015.4).
2. Introdução à placa de prototipagem *Nexys-4*.
3. Projetos simples para FPGA.

Objetivos

Realização de um conjunto de exercícios práticos introdutórios para:

- Perceber o nível básico do sistema Vivado 2015.4;
- Perceber utilização da placa *Nexys 4*;
- Estudar as construções introdutórias da linguagem VHDL através da criação de projetos simples;
- Desenvolvimento, síntese, implementação e verificação de circuitos simples em FPGA.

Materiais adicionais úteis:

- WebPack de Vivado: <http://www.xilinx.com/products/design-tools/vivado.html> .
- Xilinx Design Constraints File XDC: [elearning](http://www.xilinx.com/support/answers/69202.html).
- Manual de utilização da placa *Nexys-4* :
<http://www.digilentinc.com/Products/Detail.cfm?NavPath=2,400,1184&Prod=NEXYS4>.

Problemas para resolver

~~Exercício 1.1.~~ Mostrar os valores dos interruptores nos LEDs (verificar a instrução $LED \leq SW$);). Mostrar os valores dos botões nos LED (verificar instruções $LED(0) \leq BTNL$; $LED(1) \leq BTNC$; $LED(2) \leq BTNR$; $LED(3) \leq BTNU$; $LED(4) \leq BTND$);).

~~Exercício 1.2.~~ Use os interruptores Sw_0, \dots, Sw_3 para escolher um botão (BTNU, BTNC, BTND, BTNL, BTNR) ou um interruptor Sw_4, \dots, Sw_{16} . Mostre o valor do interruptor ou do botão escolhido no LED_0 . Exemplo: quando $Sw_0, \dots, Sw_3 = "000"$ mostrar o valor BTNU; quando $Sw_0, \dots, Sw_3 = "001"$ mostrar o valor BTNC; etc.

Exercício 1.3. Implementar um somador completo para dois operandos de um bit A (Sw_0) e B (Sw_1). Usar interruptor Sw_2 para sinal de entrada *carry in*. Mostrar o valor da soma no LED_0 e o valor do sinal *carry out* em LED_1 .

Exercício 1.4. Fazer ligação com um divisor de frequência $100 \text{ MHz} \rightarrow 1 \text{ Hz}$. Mostrar a saída do divisor no LED_3 .

Exercício 1.5. Ligar/desligar LED_3 com frequência aproximada de 1 Hz. Ligar/desligar LED_4 com frequência aproximada de 0.5 Hz. Ligar/desligar LED_5 com frequência aproximada de 0.25 Hz.

Exercício 1.6. Implemente um contador de 4 bits. Entradas são a) relógio com frequência 1 Hz (mais ou menos); b) *clock enable* (Sw_0) para permitir relógio na entrada; c) *reset* (BTNC) para fazer reset (i.e. todas as saídas são iguais a zero); d) *inc* ($Sw_1 = 0$) para incrementar o valor de saída com frequência de relógio; d) *dec* ($Sw_1 = 1$) para decrementar o valor de saída com frequência de relógio. Mostrar o conteúdo do contador nos LEDs (4 LEDs totalmente).

Exercício 1.7. Implemente uma função Booleana que permite verificar se o valor marcado nos interruptores SW_0, \dots, SW_{15} é par ou ímpar. Mostre o valor da função no LED_0 .

Exercício 1.8. Mostre nos LEDs o valor $(SW_0, \dots, SW_7) + (SW_8, \dots, SW_{15})$ quando o botão BTNU está pressionado. Mostre nos LEDs o valor $(SW_0, \dots, SW_7) - (SW_8, \dots, SW_{15})$ quando o botão BTND está pressionado. Mostre nos LEDs o valor $(SW_0, \dots, SW_7) * (SW_8, \dots, SW_{15})$ quando o botão BTNL está pressionado. Mostre nos LEDs o valor $(SW_0, \dots, SW_7) / (SW_8, \dots, SW_{15})$ quando o botão BTNR está pressionado. Mostre nos LEDs o valor $(SW_0, \dots, SW_7) \% (SW_8, \dots, SW_{15})$ quando o botão BTNC está pressionado, onde % é o resto da divisão.

Exercício 1.9. Implemente um registo de deslocamento de 16 bits. Entradas são a) relógio com frequência 1 Hz (mais ou menos); b) *clock enable* para permitir relógio na entrada; c) *reset* para fazer reset (i.e. todas as saídas são iguais a zero); d) SW_0, \dots, SW_{15} entradas paralelas; e) *set* para gravar no registo entradas paralelas; f) *left* para fazer *shift* à esquerda com frequência de relógio; g) *right* para fazer *shift* à direita com frequência de relógio. Mostrar o conteúdo do registo nos LEDs (16 LEDs totalmente).

Exercício 1.10. Implemente um sistema de funções Booleanas para encontrar o *Hamming weight* de vetor binário com 3 bits: $(SW_0, \dots, SW_2) \rightarrow (LED_0, LED_1)$.

Exercício 1.11. Implemente um sistema de funções Booleanas para fazer conversão de códigos binários para valor hexadecimal nos segmentos dum display: $(SW_0, \dots, SW_3) \rightarrow (SEG_0, \dots, SEG_7)$. Usar várias possibilidades, nomeadamente a) definir uma constante; b) usar instrução **when ... else** dentro de uma arquitetura; c) usar instrução **case ... when** dentro de um processo; d) usar instrução **with ... select** dentro de uma arquitetura.