Experimento 5. Somadores de palavras binárias e simulação de circuitos em VHDL

OBJETIVOS:

- Construir um somador de palavras binárias usando somadores completos em cascata.
- Familiarizar-se com o pacote STD_LOGIC_ARITH.
- Desenvolver um testbench para simulação e teste de circuitos em VHDL.

PRÉ-RELATÓRIO:

- Apresente o diagrama de blocos do circuito do visto 1.
- Apresente o código VHDL das entidades e das arquiteturas dos vistos 1, 2 e 3: somador do visto 1, somador do visto 2, testbench e top module.
- Apresente o arquivo UCF para os vistos 1 e 2. Note que, se as entidades do visto 1 e 2 forem criadas usando sinais com os menos nomes, você poderá usar o mesmo arquivo UCF para ambos os vistos.
- Lembre-se: seu pré-relatório deve ser entregue como um único arquivo PDF. Não anexe arquivos .vhdl nem arquivos .ucf!

VISTOS:

1. Escrever em VHDL e implementar em FPGA um somador de palavras de 4 bits, construído utilizando somente somadores completos (entidade desenvolvida no visto 1 do experimento 2, utilizada aqui como "component"). A nova entidade deve ter como entrada dois vetores *A* e *B* (com 4 bits cada) e, como saída, um vetor *S* (com 5 bits). Respeitando a ordem de significância, associe cada um dos bits de entrada a diferentes chaves (SW0 a SW7) e os de saída a diferentes LEDs (LD0 a LD7).
2. Escrever em VHDL e implementar em FPGA um somador de palavras de 4 bits, construído utilizando o operador '+' do pacote STD_LOGIC_ARITH. A nova entidade deve ter como entrada dois vetores *A* e *B* (com 4 bits cada) e, como saída, um vetor *S* (com 5 bits). Respeitando a ordem de significância, associe cada um dos bits de entrada a diferentes chaves (SW0 a SW7) e os de saída a diferentes LEDs (LD0 a LD7). Dica: declare as entradas *A* e *B* como sendo do tipo STD_LOGIC_VECTOR (como feito no visto 1) e, na arquitetura, faça a conversão para o tipo UNSIGNED usando o comando `unsigned(.)` de modo a poder usar o operador '+'.
3. Escrever em VHDL um testbench para simular e testar o somador de palavras de 4 bits desenvolvido no visto 1. Esse testbench deve gerar todas as 256 combinações possíveis de valores para *A* e *B*, aguardando 500 ns entre combinações e, para cada combinação, comparar a saída do somador do visto 1 (utilizado aqui como *device under test* ou DUT) com a saída do somador do visto 2 (utilizado aqui como *golden model*), imprimindo uma mensagem de erro se as saídas não concordarem.