O pacote STD_LOGIC_ARITH

Em VHDL, os operadores $+$ e $-$ só funcionam de forma *built in* com tipos inteiros e reais. Especificamente, eles **não** funcionam com arrays to tipo STD_LOGIC_VECTOR e BIT_VECTOR. O pacote std_logic_arith define os tipos SIGNED e UNSIGNED, e define os operadores $+$ e $-$ para estes tipos. A representação SIGNED utiliza a notação em complemento de 2.

Quando usamos esses operadores para os mesmos tipos, o funcionamento é simples (isto é, a soma de operandos do tipo UNSIGNED é UNSIGNED, e a soma de operandos do tipo SIGNED é SIGNED). Quando usamos esses operadores com tipos diferentes, o resultado é um pouco mais complicado. Em geral, se um dos operandos for SIGNED, o resultado é SIGNED.

Um cuidado especial deve ser tomado com relação ao *overflow*: a soma de dois números de n bits é retornada com n bits. Se desejamos a soma completa, com $n+1$ bits, devemos somar números de 9 bits (concatenando um 0 ao vetor de 8 bits).

Algoritmo 1 Adicionando inteiros de tipo UNSIGNED com saída de 8 bits: há risco de overflow.

```
1:  library IEEE;
2:  use IEEE.std_logic_1164.all;
3:  use IEEE.std_logic_arith.all;
4:
5:  entity vadd is
6:          port (
7:              A8, B8 : in UNSIGNED (7 downto 0);
8:              S8 : out UNSIGNED (7 downto 0)) ;
9:  end vadd;
10:
11: architecture arch of vadd is
12: begin
13:         S8 <= A8 + B8;
14: end arch;
```

Algoritmo 2 Adicionando inteiros de tipo UNSIGNED com saída de 9 bits: resolve-se a questão do overflow.

```
1:  library IEEE;
2:  use IEEE.std_logic_1164.all;
3:  use IEEE.std_logic_arith.all;
4:
5:  entity vadd is
6:          port (
7:              A8, B8 : in UNSIGNED (7 downto 0);
8:              S9 : out STD_LOGIC_VECTOR (8 downto 0)) ;
9:  end vadd;
10:
11: architecture arch of vadd is
12: begin
13:         S9 <= ('0' & A8) + ('0' & B8);
14: end arch;
```

Finalmente, como era de se esperar, podemos utilizar o pacote std_logic_arith para realizar a multiplicação de tipos *signed* e *unsigned* utilizando o operador *****. O código fica um pouco mais complicado do que os códigos anteriores, mas em geral vale a pena o esforço. Note que as varíaveis $\mathbf{X}$ e $\mathbf{Y}$ são do tipo STD_LOGIC_VECTOR, e fazemos um *cast* para determinar o tipo de multiplicação que queremos efetuar (no caso, multiplicação de números sem sinal).

Algoritmo 3 Multiplicador de 8 bits

```
1: library IEEE;
2: use IEEE.std_logic_1164.all;
3: use IEEE.std_logic_arith.all;
4:
5: entity vmul_8bits_arith is
6:         port (
7:             X,Y : in STD_LOGIC_VECTOR (7 downto 0);
8:             P : out STD_LOGIC_VECTOR (15 downto 0)) ;
9: end vmul_8bits_arith;
10:
11: architecture arch of vmul_8bits_arith is
12: begin
13:         P <= unsigned(X) * unsigned(Y);
14: end arch;
```