# Laboratório 5

**1**. Um somador binário de **n** bits é um sistema combinacional que tem duas entradas (**x** e ***y***) de ***n*** bits que representam os operandos da operação de adição e uma saída **f** de ***n*** bits que representa o resultado. Sinais adicionais de entrada e saída, chamados de *carry-in* (**cin**) e *carry-out* (**cout**) são usados para facilitar a implementação de somadores maiores.

**a)** Projete com VHDL estrutural um somador chamado **fullAdder** completo de um bit, (com os *carries*), utilizando somente portas lógicas. **[Entregar fullAdder.vhd]**

**b)** Elabore uma simulação para testar o funcionamento do seu circuito.

**c)** Usando o circuito do item a implemente um somador *ripple-carry adder* de 4 bits. O circuito deve ter entradas **x** e **y**, saída **f** e uma saída adicional **v** para indicar overflow. **[Entregar ripple4.vhd]**

d) Instancie o componente *ripple-carry adder* do item anterior no circuito demo_setup dado e faça as seguintes ligações de entrada e saída: 8 switches para as duas entradas de 4 bits; resultado da soma num display de 7 segmentos hexadecimal; overflow em um LED.

e) Implemente um somador *ripple-carry* de 8 bits e meça o tempo de atraso utilizando o timing analyzer.

f) Implemente um somador *ripple-carry* de 32 bits e meça o tempo de atraso.

g) Implemente um somador *ripple-carry* de 64 bits e meça o tempo de atraso.

*h)* Reflita sobre relação tempo de atraso e tamanho da entrada. **[Entregar Respostas.txt com os tempos dos itens (e), (f) e (g) ]**

2. Uma unidade aritmética lógica (*arithmetic – logic* unit) ALU é um módulo capaz de realizar um conjunto de funções aritméticas e lógicas. Para isto, uma ALU tem vetores de entrada/saída de dados, bem como entradas e saídas de controle. A Figura 1 mostra uma possível especificação de uma ALU de 4 bits.

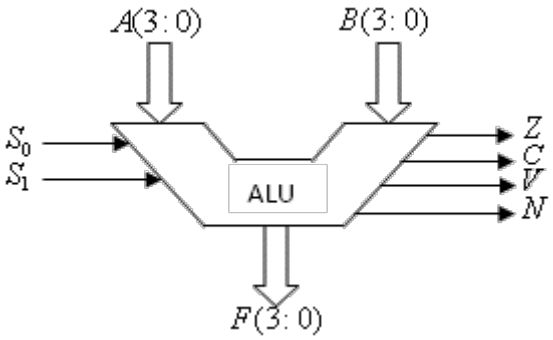

**Figura 1:** ALU de 4 bits

A **ALU** da Figura 1 tem entradas (**A** e **B**) de 4 bits, $s_{0..1}$ sinais de controle para selecionar a operação. A saída F de 4 bits mostra o resultado da operação. **Z** é 1 se o resultado da operação for zero, e 0 caso contrário.

Os sinais **C**, **V** e **N** são don´t care para operações lógicas e tem o seguinte significado para operações aritméticas. **C** é 1 se houver um *carry* em operações de soma. **V** é 1 se houver *overflow*. **N** é igual a 1 se o resultado for negativo.

Os sinais de controle e suas respectivas operações são mostradas na Tabela 1:

| $S_0$ | $S_1$ | **Operação (F)** |
|---|---|---|
| 0 | 0 | A+B |
| 1 | 0 | A-B |
| 0 | 1 | AND |
| 1 | 1 | OR |

**Tabela 1:** Operações ALU

**a)** Estenda a implementação do seu display de 7 segmentos para mostrar, em decimal, números negativos de quatro bits em complemento de 2 no formato sinal magnitude (se o resultado for negativo será necessário complementá-lo). Utilize dois displays, o da esquerda para indicar o sinal e o da direita; a magnitude. **[Entregar display7seg.vhd]**

**b)** Usando o somador de 4 bits da questão 1, implemente uma ALU de quatro bits com as operações descritas pela Tabela 1. **[Entregar ALU.vhd]**

**c)** Simule o funcionamento da sua ALU para verificar se sua implementação está correta.

**d)** Faça a simulação funcional para alguns casos extremos (i.e. *overflow*).

e) Instancie o circuito obtido no demo_setup. Faça as seguintes ligações: 8 toggle switches para as duas entradas de 4 bits; resultado f no display de 7 segmentos hexadecimal; overflow em um led.; 2 *toggle switches* para selecionar a operação da ALU; 4 leds vermelhos para os sinais de status da ALU (Z, C, V, N).