

# EA773 LABORATÓRIO DE CIRCUITOS LÓGICOS

## FEEC-UNICAMP - 2º Semestre de 2011 – Turmas C, D, S, T, U e W

Turma: \_\_\_\_\_ Grupo: \_\_\_\_\_ Data: \_\_\_\_\_

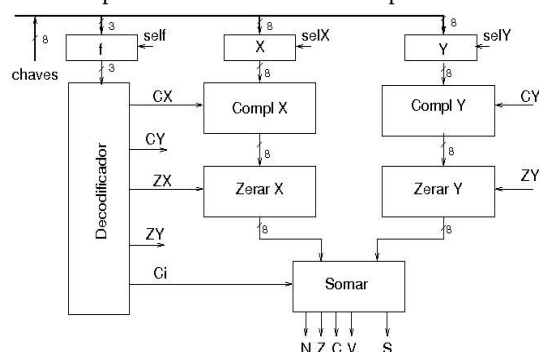
RA: \_\_\_\_\_ Nome: \_\_\_\_\_

RA: \_\_\_\_\_ Nome: \_\_\_\_\_

### Experiência 4: Calculadora em VHDL.

elaborada pela Profa. Ting

**Objetivo:** Projeto de um circuito combinacional lógico-aritmético (ULA) de 8 bits em VHDL, composto de 3 módulos funcionais, conforme esquematiza Figura 1: complementar um valor, *bit a bit*, setar um valor em zero, e somar dois operandos considerando o transporte *Ci*.



**Figura 1: ULA.**

O circuito suporta quatro tipos de operações definidas na Tabela 1.

**Tabela 1: Sinais de Controle.**

Operação	Definição	Sinais de controle
ADD	adição	$s = x + y$
SUB	subtração	$s = x - y$
CMY	complemento	$s = y'$
CSX	troca de sinal	$s = -x$

**1. Preparo** (Referência: Sistemas Digitais: Princípios e Aplicações, R. J. Tocci et al.)

(a) O que é um PLD e um FPGA? (Sec. 4.14) (b) Descreva a diferença entre uma linguagem de programação e uma linguagem de descrição de *hardware* (Sec. 3.17). (c) Descreva a estrutura básica de VHDL (Sec. 3.19). (d) Explique a diferença entre SIGNAL e VARIABLE para declarar os sinais “intermediários”. (Sec. 3.20). (e) Sintetize os tipos de dados comuns em VHDL (Sec. 4.15). (f) Cite a biblioteca que define os tipos de dados *std\_logic* e que contém os blocos primitivos lógicos e componentes mais comuns. (g) Mostre como uma tabela-verdade pode ser representada com uso de VHDL (Sec. 4.16). (h) Quais são as estruturas de controle de decisão em VHDL? (Sec. 4.17) (i) Como se descreve uma multiplexação em VHDL? (Sec. 9.15) (j) Mostre como um componente/bloco primitivo pode ser reutilizado como uma “caixa preta” em VHDL (Sec. 5.26). (l) Como se descreve *flip-flops* e *latches* em VHDL? (Sec. 5.25 e 5.26)

RA: \_\_\_\_\_ Visto: \_\_\_\_\_ Data: \_\_\_\_\_

## 2. Tipos de Dados e Descrição Booleana

Utilizando a linguagem de especificação VHDL, o circuito de **somador completo** do Exeprimento 1 pode ser especificado na seguinte forma:

```
library ieee;
use ieee.std_logic_1164.all;
entity FULLADDER is
    port(
        a      : in bit;    -- porta de entrada;
        b      : in bit;
        cin    : in bit;    -- carry in
        s      : out bit;    -- soma
        cout   : out bit    -- carry out
    );
end FULLADDER;
architecture a of FULLADDER is
    begin
        s <= (a xor b) xor cin;    -- soma
        cout <= (a and b) or (cin and (a xor b));    -- carry
    end a;
end;
```

a) Edite a especificação e compile-a no ambiente MAX+PLUSII. Encapsule o circuito em um componente.

b) Visualize o projeto sintetizado no visualizador de *netlist* do ambiente

Quartus II.

c) Simule e programe-o no *kit* FPGA-SDB para validação em campo.

d) Utilize o componente *FULLADDER* para construir um somador de 8 bits da Figura 1, representando os dados em **vetores de bits**. Verifique o *netlist* sintetizado, simule-o, e teste-o no ambiente MAX+PLUSII para os casos que produzam os valores para os *bits* de condição (N, Z, C, V) mostrados na Tabela 2.

**Tabela 2: Dados de teste.**

- N=0; Z=0; C=1; V=0;
- N=0; Z=0; C=1; V=1;
- N=0; Z=1; C=1; V=0;
- N=0; Z=1; C=0; V=0.
- N=1; Z=0; C=1; V=0;
- N=1; Z=0; C=0; V=0;
- N=1; Z=0; C=0; V=1;
- N=0; Z=0; C=0; V=0.

RA: \_\_\_\_\_ Visto: \_\_\_\_\_ Data: \_\_\_\_\_

## 3. Decodificador

a) Cada operação mostrada na Tabela 1 é representada por um código binário de 3 bits  $f = (f_2 f_1 f_0)$ , conhecido por **Código de Operação**. Tabela 3 apresenta a correspondência entre as operações e os códigos de operação. Descreva em VHDL o decodificador dos bits  $f_2 f_1 f_0$ , verifique o *netlist* sintetizado e simule o decodificador para todas as possíveis combinações de  $f = (f_2 f_1 f_0)$ .

**Tabela 3: Códigos de Operação.**

Operação	Código de Operação ( $f_2 f_1 f_0$ )
ADD	Nº do grupo%8
SUB	(Nº do grupo%8+3)%8
CMY	(Nº do grupo%8+2)%8
CSX	(Nº do grupo%8+1)%8

RA: \_\_\_\_\_ Visto: \_\_\_\_\_ Data: \_\_\_\_\_

b) Utilize VHDL para descrever a tabela-verdade do decodificador de 7 segmentos. Verifique o *netlist* sintetizado, simule-o para todas as possíveis entradas e, como validação em campo, utilize-o para decodificar a saída do somador S e visualizá-la em hexadecimal.

RA: \_\_\_\_\_ Visto: \_\_\_\_\_ Data: \_\_\_\_\_

## 4. Estruturas de Controle de Decisão

a) Utilize a estrutura de decisão IF/THEN/ELSE para descrever o módulo COMPLEMENTAR. Simule-o.

b) Utilize a estrutura de multiplexação para descrever o módulo ZERAR. Simule-o.

c) Pense em uma alternativa de implementação dos módulos ZERAR e COMPLEMENTAR com uso de funções booleanas.

RA: \_\_\_\_\_ Visto: \_\_\_\_\_ Data: \_\_\_\_\_

## 5. Periféricos de Entrada/Saída e Integração

Devido ao número limitado de chaves, os operandos e o código de operação são carregados, em separado, através das botoeiras *sel\** e armazenados em um registrador, conforme mostra Figura 1. Descreva em VHDL estes registradores. Integre em VHDL todos os módulos projetados, exceto o decodificador de 7 segmentos, em um único componente, cujo esquemático é apresentado na Figura 1. Verifique o *netlist* sintetizado e simule o componente com os dados da Tabela 2. Anote os tempos de propagação em cada módulo, estimados pelo **Analisador Temporal** do Quartus II. Para validação em campo, mapeie a saída S em *displays* de 7 segmentos e os bits N, Z, C e V em *leds*.

RA: \_\_\_\_\_ Visto: \_\_\_\_\_ Data: \_\_\_\_\_