

Aluno: _____

Matrícula: _____ Data: _____

Sprint 1 – Primeiro projeto no Quartus II

Descrição geral do problema: Criar o primeiro projeto no Quartus II e implementar uma Unidade Lógica e Aritmética (ULA) com 2 operações. Soma e Subtração.

Requisitos mínimos:

1. Setup do ambiente de trabalho individual
 - a) Crie o diretório “c:\LASD\20XX.X\SEUNOME_MATRICULA”. Cada aluno deverá usar o mesmo computador em todas as aulas do LASD. Sempre salvar seus arquivos nessa pasta;
 - b) Abra o Quartus II 13.0;
 - c) Menu “File -> New Project Wizard”;
 - d) Crie o Projeto, com nome “Mod_Teste”, no seu diretório;
 - e) Ignore o pedido de inclusão de arquivos já existentes;
 - f) Selecione a família **CycloneII**, FPGA **EP2C35F672C6** e finish;
 - g) Copie os seguintes arquivos, do google classroom, para a pasta local do seu projeto: **Mod_Teste.v**, **LCD_TEST2.v**, **LCD_CONTROLLER.v** e **DE2_PIN_ASSIGNMENT.CSV**
 - h) Use o menu “Assignments > import assignment” para incluir o arquivo **DE2_PIN_ASSIGNMENT.CSV**
 - i) Adicione, em seu projeto, os arquivos .v do passo g). Na janela “Project Navigator > Files > Botão direito > Add Remove files in project”. Selecione os arquivos, Add All e OK.



Obs: Nas aulas subsequentes devemos abrir este projeto usando “File > Open Project” e NÃO “File > Open”.
2. Dentro do módulo **Mod_Teste**, faça uma atribuição contínua entre o LEDG[0] e o botão KEY[1]. Compile , carregue na placa  e teste seu primeiro código!
3. Implemente um circuito somador/subtrator de 4bits (ULA com 2 operações)
 - a) O circuito deve possuir duas entradas de operandos de 4bits, conectadas as chaves **SW[3:0]** e **SW[7:4]**, uma entrada de seleção de operação, conectada a chave **SW[17]** e uma saída de 4bits para o resultado, conectada aos **LEDR[3:0]**. A lógica de operação da ULA é resumida na tabela 1.

Tabela 1

Entrada de seleção de Operação (SW[17])	Saída do resultado (LEDR[3:0])
0	SW[3:0] + SW[7:4]
1	SW[3:0] - SW[7:4]

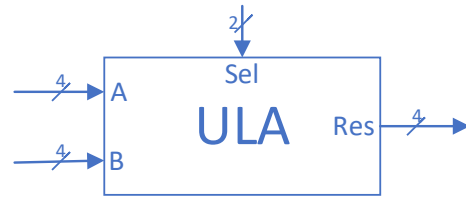
- b) **Fica a seu critério:** modelar o circuito em nível de portas lógicas ou subir o nível de abstração? implementar com atribuição contínua ou procedural? Criar um módulo ou implementar diretamente no Mod_Teste? Teste diferentes formas!
- c) Deve ser um circuito combinacional.
- d) **Roteiro de testes:**
 - i. $4'd3 + 4'd1 =$ _____ (decimal) ou _____ (binário)
 - ii. $4'd15 + 4'd2 =$ _____ (decimal) ou _____ (binário)
 - iii. $4'd7 - 4'd3 =$ _____ (decimal) ou _____ (binário)
 - iv. $4'd7 - 4'd8 =$ _____ (decimal) ou _____ (binário)
 - v. Caso o seu circuito passe por todos os testes, chame o professor para receber sua nota.
- e) Após o professor conferir seus testes, compacte o projeto em um .zip e submeta-o no Google Classroom da disciplina
- f) **Dica:** Aproveite o elevado nível de abstração que Verilog proporciona e escreva um código simples e enxuto. Daria para resolver essa sprint com uma só linha de código?

Desafio (Valendo +0,1 na média geral)

1. Incremente a sua ULA para efetuar 4 operações: Soma, Subtração, AND bit a bit e OR bit a bit
2. Crie um módulo para implementar a lógica da tabela 2.

Tabela 2

Sel	Res
00	A + B
01	A – B
10	A & B
11	A B



3. Instancie, no Mod_Teste, um bloco ULA com o mapeamento da tabela 3

Tabela 3

A	SW[3:0]
B	SW[7:4]
Sel	SW[17:16]
Res	LEDR[3:0]

4. Crie cenários de testes para todas as operações

Módulo topo – Mod_Teste

```
`default_nettype none //Comando para desabilitar declaração automática de wires
module Mod_Teste (
//Clocks
input      CLOCK_27, CLOCK_50,
//Chaves e Botoes
input  [3:0] KEY,
input  [17:0] SW,
//Displays de 7 seg e LEDs
output [0:6] HEX0, HEX1, HEX2, HEX3, HEX4, HEX5, HEX6, HEX7,
output [8:0] LEDG,
output [17:0] LEDR,
//Serial
output      UART_TXD,
input      UART_RXD,
inout  [7:0] LCD_DATA,
output      LCD_ON, LCD_BLON, LCD_RW, LCD_EN, LCD_RS,
//GPIO
inout  [35:0] GPIO_0, GPIO_1
);

assign GPIO_1      =      36'hzzzzzzzzzz;
assign GPIO_0      =      36'hzzzzzzzzzz;
assign LCD_ON      =      1'b1;
assign LCD_BLON    =      1'b1;

wire  [7:0]  w_d0x0, w_d0x1, w_d0x2, w_d0x3, w_d0x4, w_d0x5,
          w_d1x0, w_d1x1, w_d1x2, w_d1x3, w_d1x4, w_d1x5;

LCD_TEST MyLCD (
    .iCLK  ( CLOCK_50 ),
    .iRST_N( KEY[0] ),
    .d0x0(w_d0x0),.d0x1(w_d0x1),.d0x2(w_d0x2),.d0x3(w_d0x3),.d0x4(w_d0x4),.d0x5(w_d0x5),
    .d1x0(w_d1x0),.d1x1(w_d1x1),.d1x2(w_d1x2),.d1x3(w_d1x3),.d1x4(w_d1x4),.d1x5(w_d1x5),
    .LCD_DATA( LCD_DATA ),
    .LCD_RW  ( LCD_RW ),
    .LCD_EN  ( LCD_EN ),
    .LCD_RS  ( LCD_RS )
);
//----- modifique a partir daqui -----

endmodule
```