Departamento de Engenharia Elétrica e de Computação

SEL0606 – Laboratório de Sistemas Digitais

Prof. Dr. Maximiliam Luppe

PRÁTICA Nº6

Aprendizado baseado em problema (PBL)

PBL01 - Circuitos Combinacionais - ULA

Problema:

Implementar uma ULA de N bits utilizando linguagem de descrição de hardware Verilog em projeto parametrizável

Equipamentos necessários:

- Kit DE10-Lite

Introdução:

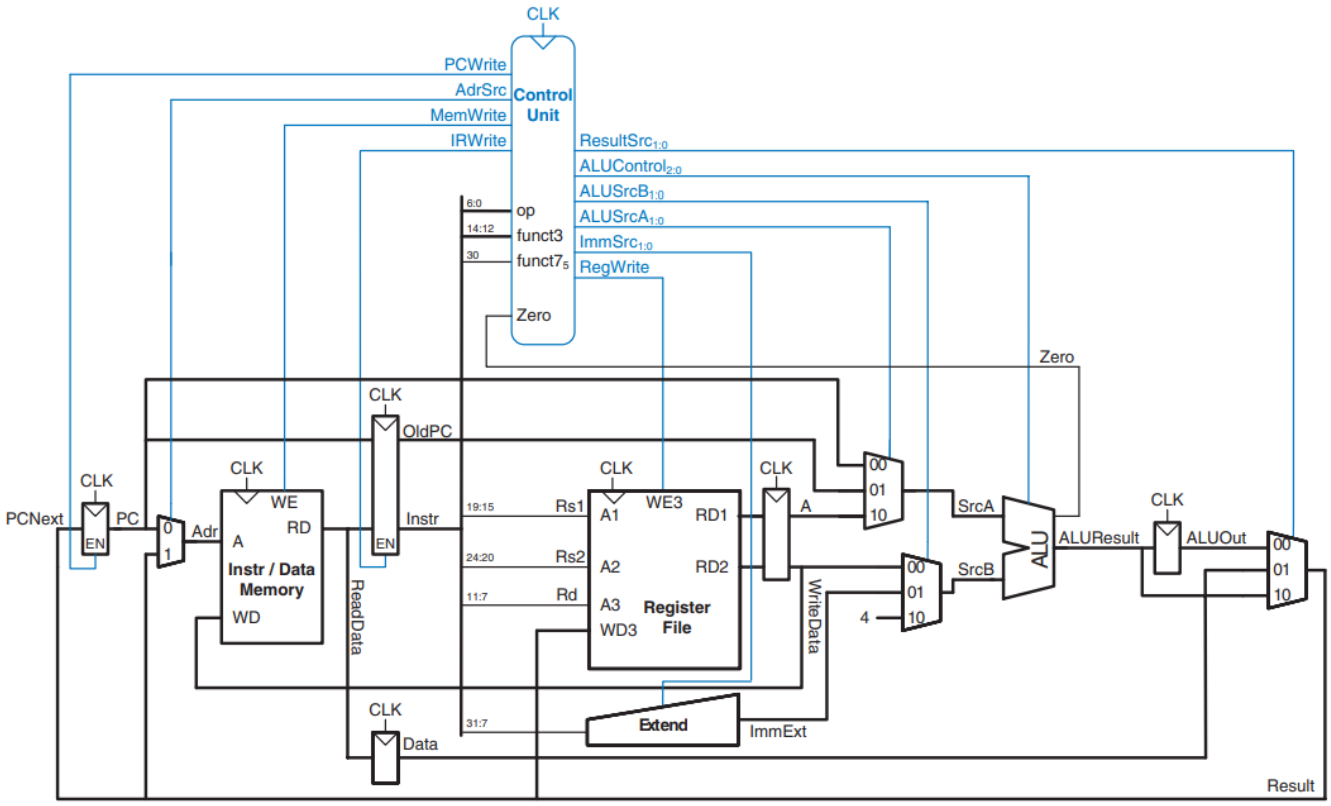
Valendo-se da natureza prática da disciplina SEL0606 – Laboratório de Sistemas Digitais e visando proporcionar aos alunos uma experiência mais ativa e dinâmica nas atividades de laboratório, será adotada a metodologia de aprendizado ativo baseado em solução de problemas (PBL - *Problem Based Learning*) ao longo das próximas práticas.

O PBL consiste em apresentar ao aluno um problema como não resolvido, de modo que este busque as informações necessárias para as causas e para as soluções do problema. O aluno poderá utilizar todo o conhecimento adquirido ao longo das demais disciplinas relacionadas (Sistemas Digitais, Organização e Arquitetura de Computadores, etc.), colocando-os em prática. Para a disciplina SEL0606, o problema central será a construção de um processador simples, baseado no RISC-V, na versão multiciclo.

O RISC-V é um conjunto de instruções (ISA) baseado em princípios RISC (Reduced Instruction Set Computing), desenvolvido desde 2010, na Universidade da Califórnia, em Berkeley. Um esquemático simples do caminho de dados para a versão multiciclo pode ser visto na figura 1. Nesta primeira parte do curso, estaremos interessados apenas na Unidade Lógica-Aritmética (ULA). A ULA, assim como o banco de registradores e a unidade de controle, é um dos componentes centrais do processador. Sua função é realizar operações aritméticas (soma, subtração, etc.) e lógicas (AND, OR, etc.) entre dois operandos. A largura N do barramento de dados da ULA é, em algumas classificações, utilizado para discriminar o tipo de arquitetura: 8 bits, 16 bits, 32 bits, etc

.

Figura 1 - Processador RISC-V multiciclo



Fonte: Digital Design and Computer Architecture - RISC-V Edition

https://doi.org/10.1016/C2019-0-00213-0

Sob o ponto de vista de Sistemas Digitais, a ULA é um circuito combinacional que realiza operações aritméticas (principalmente soma e subtração) e operações lógicas (AND, OR, NOT) entre dois valores de N bits, resultando em outro valor de N bits. Eventualmente a ULA pode gerar alguns

sinais adicionais, como *flags* de Zero (quando o resultado é igual a 0), de Negativo (quando o resultado é menor do que 0, considerando complemento de 2), de Carry, de Overflow etc. Operações mais complexas, como multiplicação, divisão, deslocamento, rotação e operações em ponto flutuante também são possíveis de serem implementadas.

O desenvolvimento da ULA se dará em duas partes. Na primeira parte será feito um levantamento da estrutura de uma ULA levando em consideração o uso de blocos lógicos básicos para a sua implementação (portas lógicas, somadores completos, mux21, etc.), subindo até níveis hierárquicos mais altos (circuito somador de N bits, comparador de N bits, multiplexador de N bits, etc.). Na segunda parte, a mesma será implementada em Verilog, utilizando os componentes selecionados na etapa anterior, com a apresentação do circuito implementado e seu devido funcionamento em relatório (teoria) e em fotos (funcionamento).

Para a primeira parte, deverá ser levada em consideração a seguinte tabela de funcionamento:

Tabela 1 - Conjunto de operações da ULA

$ALUControl_{2:0}$	Função
000	A + B
001	A - B
010	A and B
011	A or B
100	-
101	SLT
110	-
111	-

.

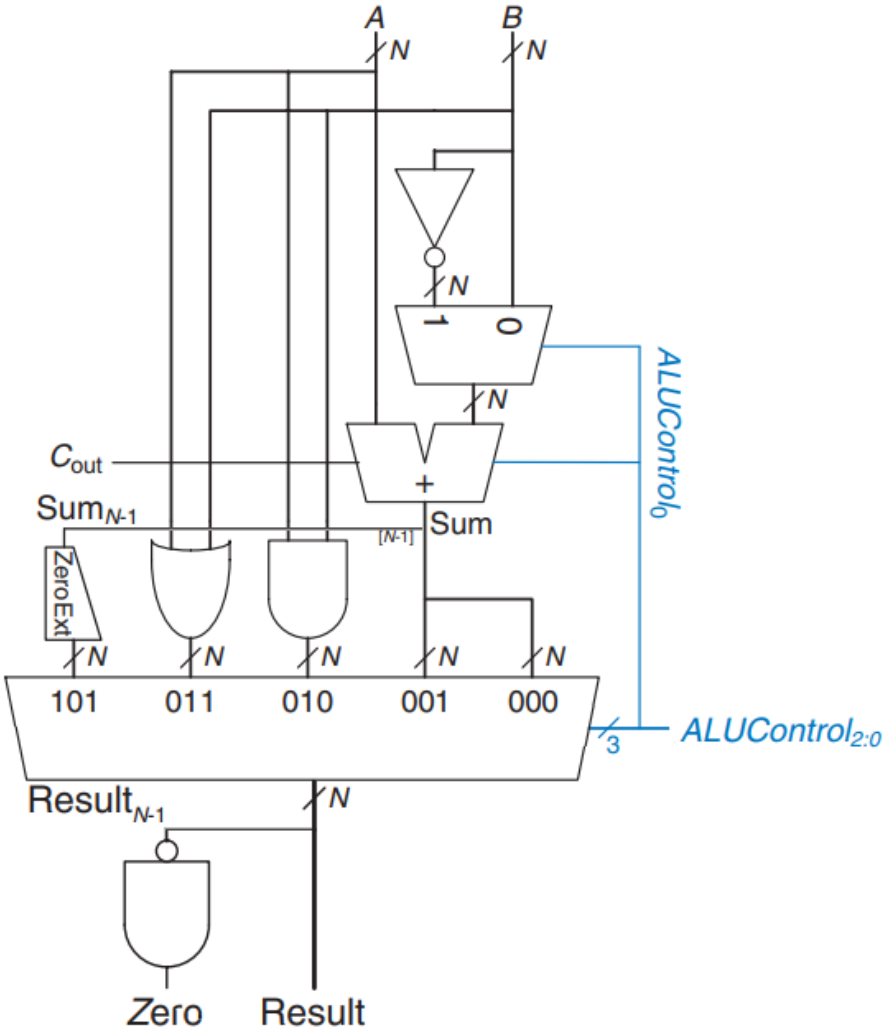
Onde $ALUControl_{2:0}$ são três sinais de controle que selecionam a operação que a ULA realizará, A e B são os dois operando de entrada, e SLT significa *Set if Less Than*, ou seja, verifica se A é menor do que B, de acordo com a Tabela 2.

Tabela 2 - Resultado da operação SLT para N=8

SLT	
A >= B	A < B
00000000	00000001

Na figura 2 é apresentado o esquemático em alto nível de uma possível implementação de uma ULA de N bits. Além do resultado descrito na Tabela 1, a ULA deverá ter como saída uma *flag* de Zero.

Figura 2 - ULA de N bits com suporte para SLT e *flag* de Zero



Fonte: Digital Design and Computer Architecture - RISC-V Edition

https://doi.org/10.1016/C2019-0-00213-0

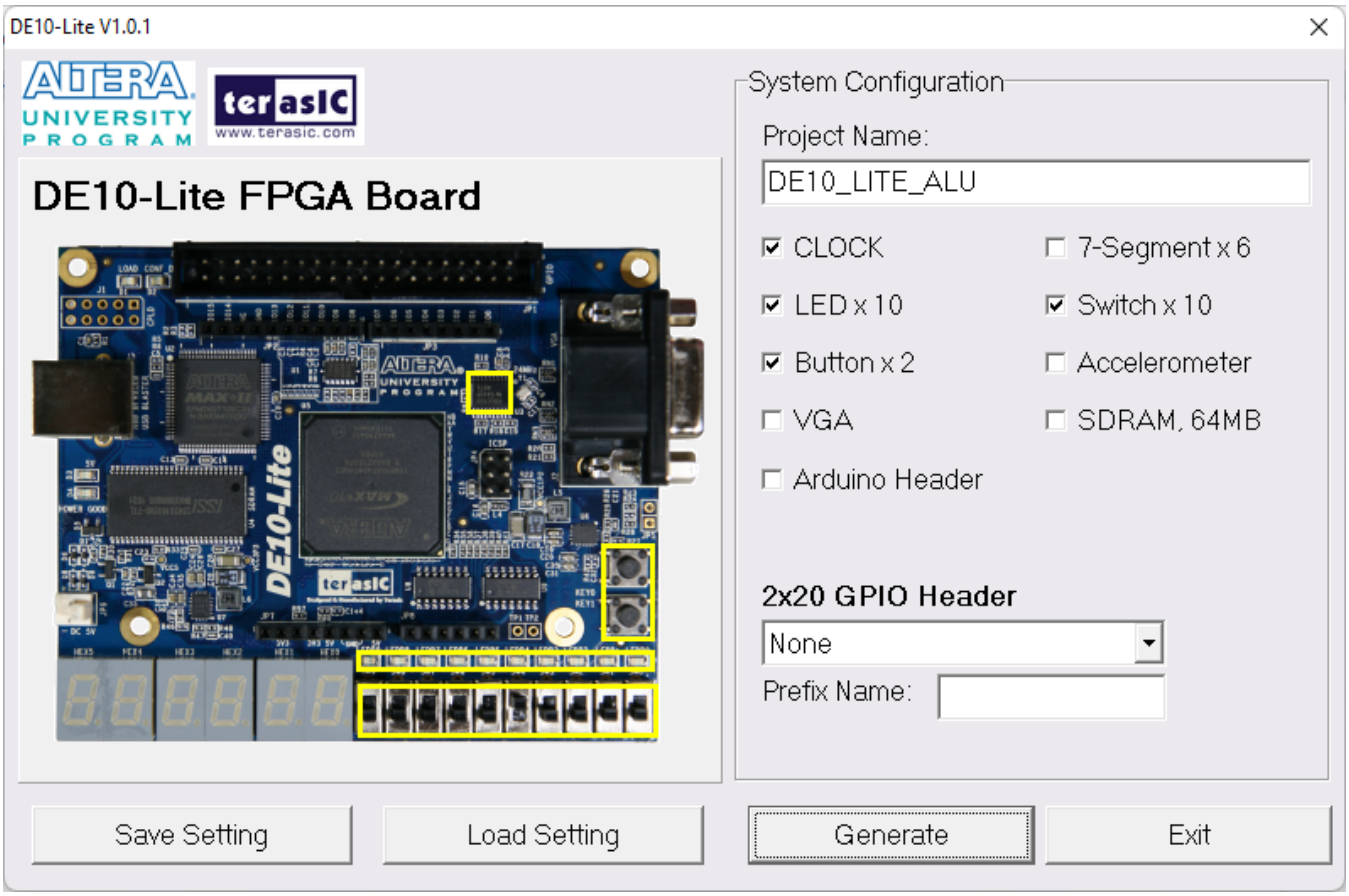
O relatório deve descrever, de forma sucinta, o funcionamento de um somador e de um multiplexador: teoria, soma em complemento de 2, equações booleanas, estrutura lógica, implementação hierárquica, etc., e de um multiplexador: teoria, equações booleanas, estrutura lógica, implementação hierárquica, etc., e como estes componentes foram utilizados para implementar a ULA.

Procedimento Experimental:

Apresentar a implementação de uma ULA baseada na arquitetura RISC-V, com barramento de dados de tamanho parametrizável, utilizando a linguagem de descrição de hardware Verilog.

Criar uma pasta denominada `DE10_LITE_ALU`, com as subpastas `docs`, `modelsim`, `quartus` e `src`, e, utilizando o DE10_Lite_SystemBuilder.exe, criar um projeto na pasta `quartus`, também denominado DE10_LITE_ALU, ativando apenas o clock, os chaves, os push-buttons e os LEDs, conforme apresentado na figura 3.

Figura 3 - Configuração do DE10_Lite_SystemBuilder.exe



Fonte: o autor

Abrir o projeto no Quartus Lite e implementar a ULA utilizando Verilog, denominado `ALU.v`, armazenando o código na pasta `src`. Incorporar o código da ULA ao projeto principal (DE10_LITE_ALU), com N=4, ligando as chaves SW[3:0] às entradas A da ULA, as chaves SW[7:4] às entradas B da ULA, o push-button KEY[0] à entrada ALUControl[2] e as chaves SW[9:8] às entradas ALUControl[1:0], as saídas Result aos LEDR[3:0] e Zero ao LEDR[4], e executar o projeto no kit DE10_LITE.

Apresentar código Verilog, circuito RTL, número de células lógicas utilizadas e foto do kit com o circuito funcionando.

Exemplo de projeto parametrizável:

```
module ALU
#(
     parameter WIDTH = 4)
(
     input [WIDTH-1:0] SrcA,
     input [WIDTH-1:0] SrcB,
     input [2:0] ALUControl,
     output reg [WIDTH-1:0] ALUResult,
     output Zero
);

     // your code goes here…

endmodule
```

Exemplo de solicitação de projeto parametrizável:

```
// Instantiate ALU design and connect with Testbench variables
ALU #(
           .WIDTH (4))
     ALU0 ( .SrcA (SW[3:0]),
            .SrcB (SW[7:4]),
            .ALUControl ({KEY[0],SW[9:8]}),
            .ALUResult (LEDR[3:0]),
            .Zero(LEDR[4]));
```