Departamento de Engenharia Elétrica e de Computação

SEL0632 – Linguagens de Descrição de Hardware

SEL5752 - Dispositivos Reconfiguráveis e Linguagem de Descrição de Hardware

Prof. Dr. Maximiliam Luppe

PROJETO FINAL

Processador RISC-V

Adendo

Utilizando o kit DE10-LITE para testar as unidades funcionais da implementação em VHDL da arquitetura RISC-V, versão RV32I, de ciclo único, baseado na implementação descrita por Harris e Harris [1]

Atividades

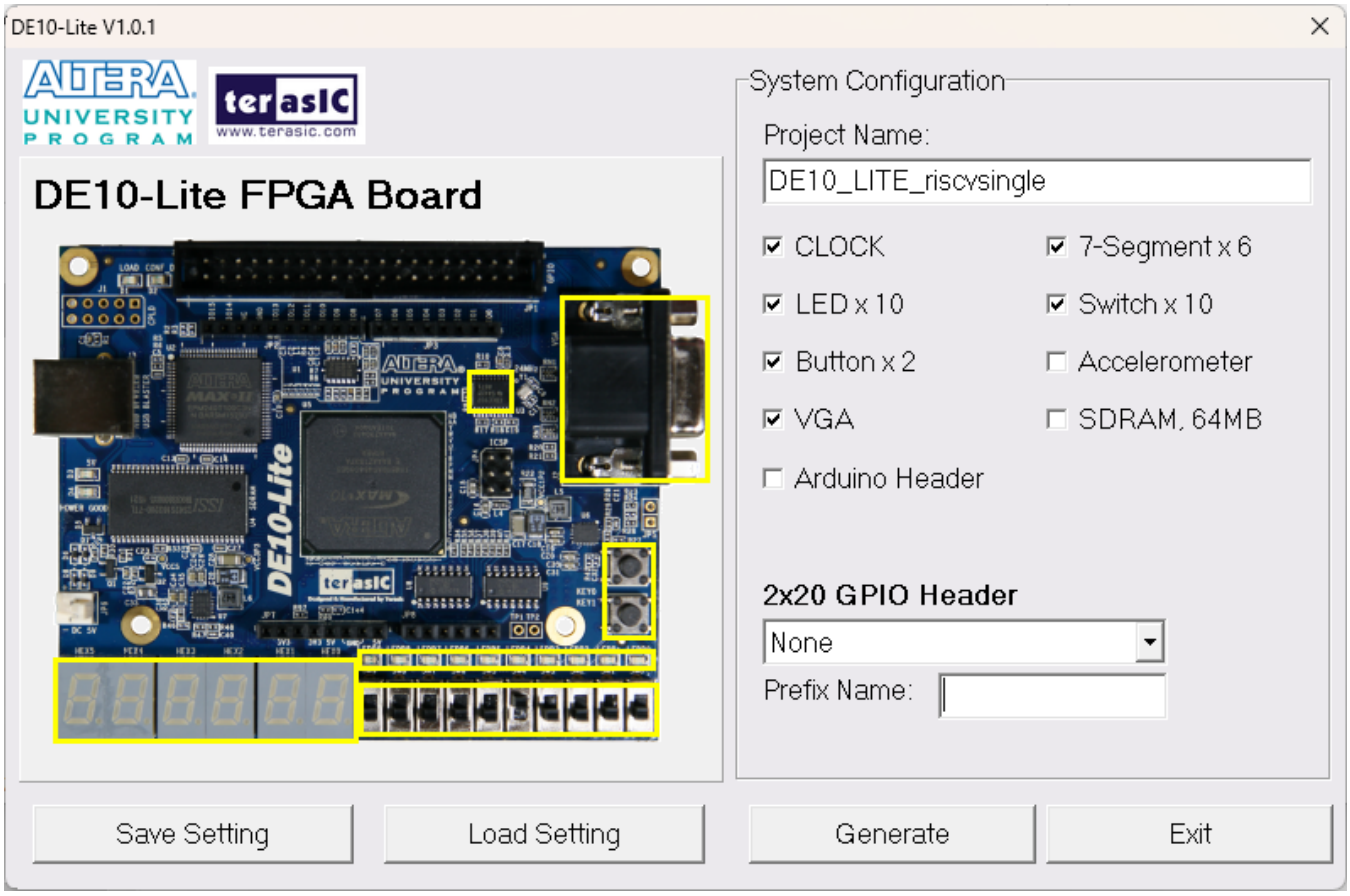
Para testar a implementação em VHDL da arquitetura RISC-V, versão RV32I, de ciclo único, baseado na implementação descrita por Harris e Harris [1], utilizando o kit DE10-LITE é necessário fazer algumas inclusões no projeto original. A primeira é incluir uma pasta denominada `DE10_LITE` às estrutura de arquivos original, de acordo com a listagem 1:

Listagem 1 - Estrutura de arquivos modificada para o projeto

```
SEL0632 (ou SEL5752)
└── Projeto2023
    ├── DE10_LITE   (acrescentar)
    ├── docs        (manter as demais inalteradas)
    ├── modelsim
    ├── quartus
    └── src
```

A pasta `DE10_LITE` será utilizada como base para realizar os testes das estruturas da arquitetura utilizando o kit DE10-LITE. Uma vez criada a pasta, o próximo passo é criar um projeto denominado `DE10_LITE_riscvsingle`, utilizando o aplicativo `DE10_Lite_SystemBuilder.exe`, com as configurações marcadas de acordo com a figura 1.

Figura 1 - Configuração para utilizar o kit DE10-LITE para testar as partes do projeto



Pressione o botão Generate e salve o novo projeto na pasta recém criada (`DE10_LITE`). Seguindo passo a passo as atividades descritas originalmente para a implementação da arquitetura, complemente as ações com as atividades a seguir, adicionando os arquivos solicitados ao projeto da DE10-LITE, após tê-lo concluído na atividade original. Os dados a serem apresentados nos relatórios devem ser obtidos pelo projeto original, armazenado na pasta quartus, uma vez que serão utilizadas estruturas extras para uma melhor visualização do funcionamento das estruturas no kit, e elas afetarão o resultado original esperado.

A primeira atividade é incluir apenas o arquivo `alu.vhd` ao projeto da pasta `DE10_LITE`, e o código descrito na listagem 2 à entidade *toplevel* (`DE10_LITE_riscvsingle.v`). Este código, escrito em Verilog, fará o instanciamento da entidade `alu` e decodificará o resultado para os displays de 7 segmentos. Os dados de entrada serão mapeados nas chaves `SW`, de acordo com a tabela 1, e também serão visualizados nos displays de 7 segmentos.

Tabela 1 - Configuração dos sinais de entrada e saída

Sinal	Chave	Display
a	SW[7:4]	HEX5
b	SW[3:0]	HEX4
ALUControl	KEY[1],SW[9],SW[8]	HEX3, HEX2, HEX1
ALUResult		HEX0
Zero		HEX0[7]

O sinal `ALUControl` será obtido concatenando o valor push-button `KEY[1]` com as chaves `SW[9]` e `SW[8]`. Desta forma, para obter a operação de `SLT`, `KEY[1]` deverá ser pressionada e as chaves `SW[9]` e `SW[8]` deverão estar na posição 0 e 1, respectivamente.

Listagem 2 - Modificação no arquivo para testar a unidade `alu` no kit DE10-LITE

```
//=======================================================
//  REG/WIRE declarations
//=======================================================

wire [3:0]  Result;
wire        Zero;

function [6:0] hexa27seg (input [3:0] hexa, input blank);
      // Declaration of local variables
      begin
      // function code
      if (blank)
            hexa27seg = 7'b1111111;
      else
```

```
            hexa27seg =  (hexa == 4'h0) ? 7'b1000000 :
                                 (hexa == 4'h1) ? 7'b1111001 :
                                 (hexa == 4'h2) ? 7'b0100100 :
                                 (hexa == 4'h3) ? 7'b0110000 :
                                 (hexa == 4'h4) ? 7'b0011001 :
                                 (hexa == 4'h5) ? 7'b0010010 :
                                 (hexa == 4'h6) ? 7'b0000010 :
                                 (hexa == 4'h7) ? 7'b1111000 :
                                 (hexa == 4'h8) ? 7'b0000000 :
                                 (hexa == 4'h9) ? 7'b0010000 :
                                 (hexa == 4'hA) ? 7'b0001000 :
                                 (hexa == 4'hB) ? 7'b0000011 :
                                 (hexa == 4'hC) ? 7'b1000110 :
                                 (hexa == 4'hD) ? 7'b0100001 :
                                 (hexa == 4'hE) ? 7'b0000110 :
                                7'b0001110;
      end
endfunction

//=======================================================
//  Structural coding
//=======================================================

alu #(.Width(4)) uut (.a(SW[7:4]), .b(SW[3:0]),
                      .ALUControl({~KEY[1],SW[9:8]}),
                      .ALUResult(Result), .Zero(Zero));

//a
assign HEX5[7] = 1'b1;
assign HEX5[6:0] = hexa27seg(SW[7:4], 1'b0);

//b
assign HEX4[7] = 1'b1;
assign HEX4[6:0] = hexa27seg(SW[3:0], 1'b0);

//ALUContro[2] (~KEY[1])
assign HEX3[7] = 1'b1;
assign HEX3[6:0] = {~KEY[1], 1'b1, ~KEY[1], ~KEY[1], 1'b0, 1'b1, 1'b1};
```

```
//ALUContro[1] (SW[9])
assign HEX2[7] = 1'b1;
assign HEX2[6:0] = {SW[9], 1'b1, SW[9], SW[9], 1'b0, 1'b1, 1'b1};

//ALUControl[0] (SW[8])
assign HEX1[7] = 1'b1;
assign HEX1[6:0] = {SW[8], 1'b1, SW[8], SW[8], 1'b0, 1'b1, 1'b1};

//ALUResult (Result)
assign HEX0[7] = ~Zero;
assign HEX0[6:0] = hexa27seg(Result, 1'b0);
```

A segunda atividade é incluir o arquivo `extend.vhd` ao projeto `DE10_LITE_riscvsingle`. Não há nada para testar no kit DE10-LITE.

A terceira atividade é incluir o arquivo `riscv_pkg` ao projeto `DE10_LITE_riscvsingle`. Não há nada para testar no kit DE10-LITE.

A quarta atividade é incluir o arquivo `regfile.vhd` ao projeto `DE10_LITE_riscvsingle`. Para testá-lo, terá que substituir o código da listagem 2 pelo código da listagem 3, na entidade *toplevel* (`DE10_LITE_riscvsingle.v`). Este código, escrito em Verilog, fará o instanciamento da entidade `regfile` e decodificará tanto o dado de entrada wd3, que será escrito no banco de registradores, no endereço indicado por a3, como os dados de saída rd1 e rd2, endereçados por a1 e a2, nos displays de 7 segmentos. Os dados de entrada serão mapeados nas chaves `SW` e `KEY`, de acordo com a tabela 2.

Tabela 2 - Configuração dos sinais de entrada e saída

Sinal	Chave	Display
clk	KEY[0]	HEX5[7]
we3	KEY[1]	HEX4[7]
a3	SW[5:4]	HEX5
a2	SW[3:2]	HEX3
a1	SW[1:0]	HEX1

wd3	SW[9:6]	HEX4
rd2		HEX2
rd1		HEX0

Para realizar uma operação de escrita do dado presente em wd3 (SW[9:6]) no registrador indicado por a3 (SW[5:4]), sinal de we3 (KEY[1]) deverá ser mantido pressionado ao mesmo tempo que é gerado um pulso de CLOCK (KEY[0]). O valor dos registradores indicados por a1 (SW[1:0]) e a2 (SW[3:2]) são visualizados nos displays HEX0 (rd1) e HEX2 (rd2), respectivamente.

Listagem 3 - Modificação no arquivo para testar a unidade regfile no kit DE10-LITE

```
//=======================================================
//  REG/WIRE declarations
//=======================================================

wire [3:0]  Result1, Result2;

function [6:0] hexa27seg (input [3:0] hexa, input blank);
      // Declaration of local variables
      begin
      // function code
      if (blank)
            hexa27seg = 7'b1111111;
      else
            hexa27seg =  (hexa == 4'h0) ? 7'b1000000 :
                              (hexa == 4'h1) ? 7'b1111001 :
                              (hexa == 4'h2) ? 7'b0100100 :
                              (hexa == 4'h3) ? 7'b0110000 :
                              (hexa == 4'h4) ? 7'b0011001 :
                              (hexa == 4'h5) ? 7'b0010010 :
                              (hexa == 4'h6) ? 7'b0000010 :
                              (hexa == 4'h7) ? 7'b1111000 :
                              (hexa == 4'h8) ? 7'b0000000 :
                              (hexa == 4'h9) ? 7'b0010000 :
                              (hexa == 4'hA) ? 7'b0001000 :
                              (hexa == 4'hB) ? 7'b0000011 :
                              (hexa == 4'hC) ? 7'b1000110 :
```

```
                                (hexa == 4'hD) ? 7'b0100001 :
                                (hexa == 4'hE) ? 7'b0000110 :
                                7'b0001110;
      end
endfunction

//=======================================================
//  Structural coding
//=======================================================

regfile #(.Width(4)) uut (
            .clk(~KEY[0]), .we3(~KEY[1]),
            .a1({2'b00,SW[1:0]}), .a2({2'b00,SW[3:2]}), .a3({2'b00,SW[5:4]}),
            .wd3(SW[9:6]), .rd1(Result1), .rd2(Result2));

//a3 (SW[5:4])
assign HEX5[7] = KEY[0];
assign HEX5[6:0] = hexa27seg({2'b00,SW[5:4]}, 1'b0);

//wd3
assign HEX4[7] = KEY[1];
assign HEX4[6:0] = hexa27seg(SW[9:6], 1'b0);

//a2 (SW[3:2])
assign HEX3[7] = 1'b1;
assign HEX3[6:0] = hexa27seg({2'b00,SW[3:2]}, 1'b0);

//rd2 (Result2)
assign HEX2[7] = 1'b1;
assign HEX2[6:0] = hexa27seg(Result2, 1'b0);

//a1 (SW[1:0])
assign HEX1[7] = 1'b1;
assign HEX1[6:0] = hexa27seg({2'b00,SW[1:0]}, 1'b0);

//rd1 (Result1)
assign HEX0[7] = 1'b1;
assign HEX0[6:0] = hexa27seg(Result1, 1'b0);
```

A quinta atividade é incluir o arquivo `mux3.vhd` ao projeto `DE10_LITE_riscvsingle`. Para testá-lo, terá que substituir o código da listagem 3 pelo código da listagem 4, na entidade *toplevel* (`DE10_LITE_riscvsingle.v`). Este código, escrito em Verilog, fará o instanciamento da entidade `mux3` e decodificará tanto os dados de entrada `d0`, `d1`, `d2` e `s`, como o dado de saída `y`, para os displays de 7 segmentos. Os dados de entrada serão mapeados nas chaves `SW` e `KEY`, de acordo com a tabela 3.

Tabela 3 - Configuração dos sinais de entrada e saída

Sinal	Chave	Display
d2	SW[8:6]	HEX5
d1	SW[5:3]	HEX4
d0	SW[2:0]	HEX3
s	KEY[1],KEY[0]	HEX1
y		HEX0

Por meio de `s` (`KEY[1]`, `KEY[0]`), o valor presente em uma das entradas, `d0` (`SW[2:0]`), `d1` (`SW[5:3]`) e `d3` (`SW[8:6]`), será visualizado no display `HEX0` (`y`).

Listagem 4 - Modificação no arquivo para testar a unidade `mux3` no kit DE10-LITE

```
//=======================================================
//  REG/WIRE declarations
//=======================================================

wire [2:0]  Result;

function [6:0] hexa27seg (input [3:0] hexa, input blank);
      // Declaration of local variables
      begin
      // function code
      if (blank)
            hexa27seg = 7'b1111111;
      else
            hexa27seg =  (hexa == 4'h0) ? 7'b1000000 :
                              (hexa == 4'h1) ? 7'b1111001 :
```

```
                                (hexa == 4'h2) ? 7'b0100100 :
                                (hexa == 4'h3) ? 7'b0110000 :
                                (hexa == 4'h4) ? 7'b0011001 :
                                (hexa == 4'h5) ? 7'b0010010 :
                                (hexa == 4'h6) ? 7'b0000010 :
                                (hexa == 4'h7) ? 7'b1111000 :
                                (hexa == 4'h8) ? 7'b0000000 :
                                (hexa == 4'h9) ? 7'b0010000 :
                                (hexa == 4'hA) ? 7'b0001000 :
                                (hexa == 4'hB) ? 7'b0000011 :
                                (hexa == 4'hC) ? 7'b1000110 :
                                (hexa == 4'hD) ? 7'b0100001 :
                                (hexa == 4'hE) ? 7'b0000110 :
                                7'b0001110;
      end
endfunction

//=======================================================
//  Structural coding
//=======================================================

mux3 #(.width(3)) uut (.d0(SW[2:0]), .d1(SW[5:3]), .d2(SW[8:6]),
                  .s({~KEY[1],~KEY[0]}), .y(Result));

//d2 (SW[8:6])
assign HEX5[7] = 1'b1;
assign HEX5[6:0] = hexa27seg({1'b0,SW[8:6]}, 1'b0);

//d1 (SW[5:3])
assign HEX4[7] = 1'b1;
assign HEX4[6:0] = hexa27seg({1'b0,SW[5:3]}, 1'b0);

//d0 (SW[2:0])
assign HEX3[7] = 1'b1;
assign HEX3[6:0] = hexa27seg({1'b0,SW[2:0]}, 1'b0);

//Blank
assign HEX2[7] = 1'b1;
assign HEX2[6:0] = hexa27seg(4'h0, 1'b1);
```

```
//s (KEY[1],KEY[0])
assign HEX1[7] = 1'b1;
assign HEX1[6:0] = hexa27seg({2'b00,~KEY[1],~KEY[0]}, 1'b0);

//y (Result)
assign HEX0[7] = 1'b1;
assign HEX0[6:0] = hexa27seg({1'b0,Result}, 1'b0);
```

A sexta atividade é incluir o arquivo `mux2.vhd` ao projeto `DE10_LITE_riscvsingle`. Para testá-lo, terá que substituir o código da listagem 4 pelo código da listagem 5, na entidade *toplevel* (`DE10_LITE_riscvsingle.v`). Este código, escrito em Verilog, fará o instanciamento da entidade `mux2` e decodificará tanto os dados de entrada `d0`, `d1` e `s`, como o dado de saída `y`, para os displays de 7 segmentos. Os dados de entrada serão mapeados nas chaves `SW` e `KEY`, de acordo com a tabela 4.

Tabela 4 - Configuração dos sinais de entrada e saída

Sinal	Chave	Display
d1	SW[7:4]	HEX4
d0	SW[3:0]	HEX3
s	KEY[0]	HEX1
y		HEX0

Por meio de `s` (`KEY[0]`), o valor presente em uma das entradas, `d0` (`SW[3:0]`) e `d1` (`SW[7:4]`), será visualizado no display `HEX0` (`y`).

Listagem 5 - Modificação no arquivo para testar a unidade `mux2` no kit DE10-LITE

```
//=======================================================
//  REG/WIRE declarations
//=======================================================

wire [3:0]  Result;

function [6:0] hexa27seg (input [3:0] hexa, input blank);
```

```
        // Declaration of local variables
        begin
        // function code
        if (blank)
              hexa27seg = 7'b1111111;
        else
              hexa27seg =  (hexa == 4'h0) ? 7'b1000000 :
                                 (hexa == 4'h1) ? 7'b1111001 :
                                 (hexa == 4'h2) ? 7'b0100100 :
                                 (hexa == 4'h3) ? 7'b0110000 :
                                 (hexa == 4'h4) ? 7'b0011001 :
                                 (hexa == 4'h5) ? 7'b0010010 :
                                 (hexa == 4'h6) ? 7'b0000010 :
                                 (hexa == 4'h7) ? 7'b1111000 :
                                 (hexa == 4'h8) ? 7'b0000000 :
                                 (hexa == 4'h9) ? 7'b0010000 :
                                 (hexa == 4'hA) ? 7'b0001000 :
                                 (hexa == 4'hB) ? 7'b0000011 :
                                 (hexa == 4'hC) ? 7'b1000110 :
                                 (hexa == 4'hD) ? 7'b0100001 :
                                 (hexa == 4'hE) ? 7'b0000110 :
                                 7'b0001110;
        end
endfunction

//=======================================================
//  Structural coding
//=======================================================

mux2 #(.width(4)) uut (.d0(SW[3:0]), .d1(SW[7:4]),
                       .s(~KEY[0]), .y(Result));

//Blank
assign HEX5[7] = 1'b1;
assign HEX5[6:0] = hexa27seg(4'h0, 1'b1);

//d1 (SW[7:4])
assign HEX4[7] = 1'b1;
assign HEX4[6:0] = hexa27seg(SW[7:4], 1'b0);
```

```
//d0 (SW[3:0])
assign HEX3[7] = 1'b1;
assign HEX3[6:0] = hexa27seg(SW[3:0], 1'b0);

//Blank
assign HEX2[7] = 1'b1;
assign HEX2[6:0] = hexa27seg(4'h0, 1'b1);

//s (KEY[0])
assign HEX1[7] = 1'b1;
assign HEX1[6:0] = hexa27seg({3'b000,~KEY[0]}, 1'b0);

//y (Result)
assign HEX0[7] = 1'b1;
assign HEX0[6:0] = hexa27seg(Result, 1'b0);
```

A sétima atividade é incluir o arquivo `adder.vhd` ao projeto `DE10_LITE_riscvsingle`. Para testá-lo, terá que substituir o código da listagem 5 pelo código da listagem 6, na entidade *toplevel* (`DE10_LITE_riscvsingle.v`). Este código, escrito em Verilog, fará o instanciamento da entidade `adder` e decodificará tanto os dados de entrada `a` e `b`, como o dado de saída `y`, para os displays de 7 segmentos. Os dados de entrada serão mapeados nas chaves `SW`, de acordo com a tabela 5.

Tabela 5 - Configuração dos sinais de entrada e saída

Sinal	Chave	Display
a	SW[9:5]	HEX4
b	SW[4:0]	HEX2
y		HEX0

O resultado da soma das entradas `a` (`SW[7:4]`) e `b` (`SW[3:0]`) será visualizado no display `HEX0` (`y`).

Listagem 6 - Modificação no arquivo para testar a unidade `adder` no kit DE10-LITE

```
//=======================================================
```

```
//  REG/WIRE declarations
//=======================================================

wire [3:0]   Result;

function [6:0] hexa27seg (input [3:0] hexa, input blank);
      // Declaration of local variables
      begin
      // function code
      if (blank)
            hexa27seg = 7'b1111111;
      else
            hexa27seg =  (hexa == 4'h0) ? 7'b1000000 :
                              (hexa == 4'h1) ? 7'b1111001 :
                              (hexa == 4'h2) ? 7'b0100100 :
                              (hexa == 4'h3) ? 7'b0110000 :
                              (hexa == 4'h4) ? 7'b0011001 :
                              (hexa == 4'h5) ? 7'b0010010 :
                              (hexa == 4'h6) ? 7'b0000010 :
                              (hexa == 4'h7) ? 7'b1111000 :
                              (hexa == 4'h8) ? 7'b0000000 :
                              (hexa == 4'h9) ? 7'b0010000 :
                              (hexa == 4'hA) ? 7'b0001000 :
                              (hexa == 4'hB) ? 7'b0000011 :
                              (hexa == 4'hC) ? 7'b1000110 :
                              (hexa == 4'hD) ? 7'b0100001 :
                              (hexa == 4'hE) ? 7'b0000110 :
                              7'b0001110;
      end
endfunction

//=======================================================
//  Structural coding
//=======================================================

adder #(.width(4)) uut (.a(SW[7:4]), .b(SW[3:0]),
                  .y(Result));

//a (SW[9])
```

```
assign HEX5[7] = 1'b1;
assign HEX5[6:0] = hexa27seg({3'h0, SW[9]}, 1'b0);

//a (SW[8:5])
assign HEX4[7] = 1'b1;
assign HEX4[6:0] = hexa27seg(SW[8:5], 1'b0);

//b (SW[4])
assign HEX3[7] = 1'b1;
assign HEX3[6:0] = hexa27seg({3'h0, SW[4]}, 1'b0);

//b (SW[3:0])
assign HEX2[7] = 1'b1;
assign HEX2[6:0] = hexa27seg(SW[3:0], 1'b0);

//y (Result[4]
assign HEX1[7] = 1'b1;
assign HEX1[6:0] = hexa27seg({3'h0, Result[4]}, 1'b0);

//y (Result[3:0])
assign HEX0[7] = 1'b1;
assign HEX0[6:0] = hexa27seg(Result[3:0], 1'b0);
```

A oitava atividade é incluir o arquivo `flopr.vhd` ao projeto `DE10_LITE_riscvsingle`. Para testá-lo, terá que substituir o código da listagem 6 pelo código da listagem 7, na entidade *toplevel* (`DE10_LITE_riscvsingle.v`). Este código, escrito em Verilog, fará o instanciamento da entidade `flopr` e decodificará tanto os dados de entrada clk, reset, a e b, como o dado de saída q, para os displays de 7 segmentos. Os dados de entrada serão mapeados nas chaves `SW` e `KEY`, de acordo com a tabela 6.

Tabela 6 - Configuração dos sinais de entrada e saída

Sinal	Chave	Display
clk	KEY[0]	
reset	KEY[1]	
d	SW[9:0]	HEX5, HEX4, HEX3

q		HEX2, HEX1, HEX0

Tanto o valor de d (SW[9:0]), quanto o valor de q, serão visualizados nos display de 7 segmentos, de acordo com a tabela 7.

Listagem 7 - Modificação no arquivo para testar a unidade flopr no kit DE10-LITE

```
//=======================================================
//  REG/WIRE declarations
//=======================================================

wire [9:0]   Result;

function [6:0] hexa27seg (input [3:0] hexa, input blank);
	// Declaration of local variables
	begin
	// function code
	if (blank)
		hexa27seg = 7'b1111111;
	else
		hexa27seg =  (hexa == 4'h0) ? 7'b1000000 :
					(hexa == 4'h1) ? 7'b1111001 :
					(hexa == 4'h2) ? 7'b0100100 :
					(hexa == 4'h3) ? 7'b0110000 :
					(hexa == 4'h4) ? 7'b0011001 :
					(hexa == 4'h5) ? 7'b0010010 :
					(hexa == 4'h6) ? 7'b0000010 :
					(hexa == 4'h7) ? 7'b1111000 :
					(hexa == 4'h8) ? 7'b0000000 :
					(hexa == 4'h9) ? 7'b0010000 :
					(hexa == 4'hA) ? 7'b0001000 :
					(hexa == 4'hB) ? 7'b0000011 :
					(hexa == 4'hC) ? 7'b1000110 :
					(hexa == 4'hD) ? 7'b0100001 :
					(hexa == 4'hE) ? 7'b0000110 :
					7'b0001110;
	end
endfunction
```

```
//=======================================================
//  Structural coding
//=======================================================

flopr #(.width(10)) uut (.clk(~KEY[0]), .reset(~KEY[1]), .d(SW[9:0]),
                  .q(Result));

//d (SW[9:8])
assign HEX5[7] = 1'b1;
assign HEX5[6:0] = hexa27seg({2'b00, SW[9:8]}, 1'b0);

//d (SW[7:4])
assign HEX4[7] = 1'b1;
assign HEX4[6:0] = hexa27seg(SW[7:4], 1'b0);

//d (SW[3:0])
assign HEX3[7] = 1'b1;
assign HEX3[6:0] = hexa27seg(SW[3:0], 1'b0);

//q (Result[9:8])
assign HEX2[7] = 1'b1;
assign HEX2[6:0] = hexa27seg({2'b00, Result[9:8]}, 1'b0);

//q (Result[7:4])
assign HEX1[7] = 1'b1;
assign HEX1[6:0] = hexa27seg(Result[7:4], 1'b0);

//q (Result[3:0])
assign HEX0[7] = 1'b1;
assign HEX0[6:0] = hexa27seg(Result[3:0], 1'b0);
```

A nona atividade é incluir o arquivo `datapath.vhd` ao projeto `DE10_LITE_riscvsingle`. Não há nada para testar no kit DE10-LITE.

A décima atividade é incluir o arquivo `maindec.vhd` ao projeto `DE10_LITE_riscvsingle`. Para testá-lo, terá que substituir o código da listagem 7 pelo código da listagem 8, na entidade *toplevel*

(DE10_LITE_riscvsingle.v). Este código, escrito em Verilog, fará o instanciamento da entidade maindec e decodificará tanto os dados de entrada op, como os dados de saída RegWrite, ImmSrc[1:0], ALUSrc, MemWrite, ResultSrc[1:0], Branch, ALUOp[1:0] e Jump, para o segmento a do display de 7 segmentos HEX0 e para os leds (LEDR[9:0]), respectivamente. Os dados de entrada serão mapeados nas chaves SW, de acordo com a tabela 7.

Tabela 7 - Configuração dos sinais de entrada e saída

Sinal	Chave	Display
op	SW[9:0]	
ImmSrc[1:0]		LEDR[9:8]
ALUSrc		LEDR[7]
MemWrite		LEDR[6]
ResultSrc[1:0]		LEDR[5:4]
Branch		LEDR[3]
ALUOp[1:0]		LEDR[2:1]
RegWrite		HEX0[0]
Jump		LEDR[0]

O resultado esperado poderá ser comparado com a tabela 2 do arquivo Projeto2023.pdf.

Listagem 8 - Modificação no arquivo para testar a unidade maindec no kit DE10-LITE

```
//=======================================================
//  REG/WIRE declarations
//=======================================================

wire        RegWrite;

function [6:0] hexa27seg (input [3:0] hexa, input blank);
      // Declaration of local variables
      begin
      // function code
      if (blank)
```

```
            hexa27seg = 7'b1111111;
      else
            hexa27seg =  (hexa == 4'h0) ? 7'b1000000 :
                              (hexa == 4'h1) ? 7'b1111001 :
                              (hexa == 4'h2) ? 7'b0100100 :
                              (hexa == 4'h3) ? 7'b0110000 :
                              (hexa == 4'h4) ? 7'b0011001 :
                              (hexa == 4'h5) ? 7'b0010010 :
                              (hexa == 4'h6) ? 7'b0000010 :
                              (hexa == 4'h7) ? 7'b1111000 :
                              (hexa == 4'h8) ? 7'b0000000 :
                              (hexa == 4'h9) ? 7'b0010000 :
                              (hexa == 4'hA) ? 7'b0001000 :
                              (hexa == 4'hB) ? 7'b0000011 :
                              (hexa == 4'hC) ? 7'b1000110 :
                              (hexa == 4'hD) ? 7'b0100001 :
                              (hexa == 4'hE) ? 7'b0000110 :
                             7'b0001110;
      end
endfunction

//=======================================================
//  Structural coding
//=======================================================

maindec uut (.op(SW[6:0]),
                        .ResultSrc(LEDR[5:4]),
                        .MemWrite(LEDR[6]),
                        .Branch(LEDR[3]),
                        .ALUSrc(LEDR[7]),
                        .RegWrite(RegWrite),
                        .Jump(LEDR[0]),
                        .ImmSrc(LEDR[9:8]),
                        .ALUOp(LEDR[2:1]));

//Blank
assign HEX5[7] = 1'b1;
assign HEX5[6:0] = hexa27seg(4'h0, 1'b1);
```

```
//Blank
assign HEX4[7] = 1'b1;
assign HEX4[6:0] = hexa27seg(4'h0, 1'b1);

//Blank
assign HEX3[7] = 1'b1;
assign HEX3[6:0] = hexa27seg(4'h0, 1'b1);

//Blank
assign HEX2[7] = 1'b1;
assign HEX2[6:0] = hexa27seg(4'h0, 1'b1);

//Blank
assign HEX1[7] = 1'b1;
assign HEX1[6:0] = hexa27seg(4'h0, 1'b1);

//Blank
assign HEX0[7] = 1'b1;
assign HEX0[6:0] = {1'b1, 1'b1, 1'b1, 1'b1, 1'b1, 1'b1, ~RegWrite};
```

A décima primeira atividade é incluir o arquivo `aludec.vhd` ao projeto `DE10_LITE_riscvsingle`. Para testá-lo, terá que substituir o código da listagem 7 pelo código da listagem 8, na entidade *toplevel* (`DE10_LITE_riscvsingle.v`). Este código, escrito em Verilog, fará o instanciamento da entidade `aludec` e decodificará tanto os dados de entrada opb5, funct3, funct7b5 e ALUOp, como o dado de saída ALUControl para os leds (LEDR[2:0]). Os dados de entrada serão mapeados nas chaves `SW`, de acordo com a tabela 8.

Tabela 8 - Configuração dos sinais de entrada e saída

Sinal	Chave	Display
ALUOp	SW[6:5]	
funct3	SW[4:2]	
opb5	SW[1]	
funct7b5	SW[0]	
ALUControl		LEDR[2:0]

O resultado esperado poderá ser comparado com a tabela 5 do arquivo Projeto2023.pdf.

Listagem 9 - Modificação no arquivo para testar a unidade `aludec` no kit DE10-LITE

```
//=======================================================
//  REG/WIRE declarations
//=======================================================

function [6:0] hexa27seg (input [3:0] hexa, input blank);
	// Declaration of local variables
	begin
	// function code
	if (blank)
		hexa27seg = 7'b1111111;
	else
		hexa27seg =  (hexa == 4'h0) ? 7'b1000000 :
					(hexa == 4'h1) ? 7'b1111001 :
					(hexa == 4'h2) ? 7'b0100100 :
					(hexa == 4'h3) ? 7'b0110000 :
					(hexa == 4'h4) ? 7'b0011001 :
					(hexa == 4'h5) ? 7'b0010010 :
					(hexa == 4'h6) ? 7'b0000010 :
					(hexa == 4'h7) ? 7'b1111000 :
					(hexa == 4'h8) ? 7'b0000000 :
					(hexa == 4'h9) ? 7'b0010000 :
					(hexa == 4'hA) ? 7'b0001000 :
					(hexa == 4'hB) ? 7'b0000011 :
					(hexa == 4'hC) ? 7'b1000110 :
					(hexa == 4'hD) ? 7'b0100001 :
					(hexa == 4'hE) ? 7'b0000110 :
					7'b0001110;
	end
endfunction

//=======================================================
//  Structural coding
//=======================================================
```

```
aludec uut (.opb5(SW[1]),
                        .funct3(SW[4:2]),
                        .funct7b5(SW[0]),
                        .ALUOp(SW[6:5]),
                        .ALUControl(LEDR[2:0]));

//Blank
assign HEX5[7] = 1'b1;
assign HEX5[6:0] = hexa27seg(4'h0, 1'b1);

//Blank
assign HEX4[7] = 1'b1;
assign HEX4[6:0] = hexa27seg(4'h0, 1'b1);

//Blank
assign HEX3[7] = 1'b1;
assign HEX3[6:0] = hexa27seg(4'h0, 1'b1);

//Blank
assign HEX2[7] = 1'b1;
assign HEX2[6:0] = hexa27seg(4'h0, 1'b1);

//Blank
assign HEX1[7] = 1'b1;
assign HEX1[6:0] = hexa27seg(4'h0, 1'b1);

//Blank
assign HEX0[7] = 1'b1;
assign HEX0[6:0] = hexa27seg(4'h0, 1'b1);
```

A décima segunda atividade é incluir o arquivo `controller.vhd` ao projeto `DE10_LITE_riscvsingle`. Não há nada para testar no kit DE10-LITE.

A décima terceira atividade é incluir o arquivo `riscvsingle.vhd` ao projeto `DE10_LITE_riscvsingle`. Não há nada para testar no kit DE10-LITE.

Referência bibliográfica

[1] Harris, S. L., Harris, D., “Digital Design and Computer Architecture RISC-V Edition”, Morgan Kaufmann, 2022