Universidade Federal da Bahia
Escola Politécnica
Departamento de Engenharia Elétrica e de Computação
Docente: Wagner Oliveira
Discentes: Caio Mendes, Fábio Miguel, Gerson Daniel, Heverton Reis e José Alves
Semestre 2024.2

Relatório - Release 3

Introdução

O presente documento consolida as etapas finais do fluxo de desenvolvimento de um processador com pipeline de cinco estágios, realizado em linguagem de descrição de hardware (Verilog HDL). O foco se manterá na *validação funcional*, na *prototipação* e na *análise temporal estática (STA)*.

Validação Funcional

A verificação do fluxo das operações determinadas pelas instruções suportadas pelo processador se deu com arquivos de testbench em Verilog HDL que buscaram validar diferentes aspectos do projeto, enquanto este estava em processo de desenvolvimento. Aqui, no entanto, serão exploradas apenas duas entradas desses arquivos.

Testes do Design Principal (TOP/ProcessadorPipeline)

No código Verilog abaixo, encontra-se o *testbench* utilizado para validação do *pipeline* do processador como também algumas de suas operações implementadas, podemos citar, por exemplo, operações com acesso à memória, aritméticas e salto. É importante destacar que as instruções realizadas durante este programa de testes esteve carregada diretamente na memória ROM (*Read-Only Memory*) por meio do arquivo *instruction_memory.mif*.

```
module ProcessadorPipeline_tb;

    // Declaracao dos sinais de teste
    reg clk1;                       // Clock principal
    reg clk2;                       // Clock da memória de instrução
    reg rst;                        // Reset
    wire [31:0] IF_ID_Instruction; // Instrução de saída
    wire [7:0] IF_ID_PC;          // Contador de programa (PC)
    wire [31:0] ID_EX_ReadData1;   // Conteudo do Reg1
    wire [31:0] ID_EX_SignExtImm;   // Conteudo do Sinal Extendido
    wire [4:0] ID_EX_Rd;   // Endereco do Reg Destino
    wire [4:0] ID_EX_Rb;   // Endereco do RegBase
    wire [7:0] ID_EX_PC;          // Contador de programa (PC) ID_EX
    wire ID_EX_RegDst, ID_EX_ALUSrc, ID_EX_MemToReg, ID_EX_RegWrite, ID_EX_MemRead, ID_EX_MemWrite,
ID_EX_Branch;
    wire [4:0] ID_EX_ALUOp;
    wire [31:0] EX_MEM_ALUResult, EX_MEM_WriteData;
    wire [4:0] EX_MEM_WriteReg;
    wire EX_MEM_MemWriteOut, EX_MEM_MemtoRegOut, EX_MEM_RegWrite, EX_MEM_MemReadOut, EX_MEM_Branch;
```

```
wire [7:0] EX_MEM_BranchTarget;
wire [31:0] MEM_WB_ALUResult;
    wire [31:0] MEM_WB_ReadData;
wire [4:0] MEM_WB_WriteReg;
wire MEM_WB_MemToReg;
    wire MEM_WB_RegWrite;
wire [31:0] WriteData;
wire [4:0] WriteRegOut;
wire RegWriteOut;
    wire WB_RegWrite;
    wire [4:0] WB_writeReg;
    wire [31:0] WB_writeData;
wire [31:0] ID_EX_ReadData2;



ProcessadorPipeline dut (
    .clk_ROM(clk2),
    .clk_RAM(clk2),
    .clk_Reg(clk2),
    .clk(clk1),
    .rst(rst)
);

// Instanciacaoo do modulo IF_Stage
IF_STAGE dut_IF_STAGE (
    .clk(clk1),
    .rst(rst),
    .BranchTaken(EX_MEM_Branch),
    .clk_ROM(clk2),
    .BranchTarget(EX_MEM_BranchTarget),
    .IF_ID_Instruction(IF_ID_Instruction),
    .IF_ID_PC(IF_ID_PC)
);

// Instanciando o modulo ID_Stage
ID_STAGE dut_ID_STAGE (
    .clk(clk1),
    .clk_Reg(clk2),
    .rst(rst),
    .IF_ID_PC(IF_ID_PC),
    .IF_ID_Instruction(IF_ID_Instruction),
    .RegWrite(RegWriteOut),
    .writeReg(WriteRegOut),
    .writeData(WriteData),
    .ID_EX_ReadData1(ID_EX_ReadData1),
    .ID_EX_ReadData2(ID_EX_ReadData2),
    .ID_EX_SignExtImm(ID_EX_SignExtImm),
    .ID_EX_Rd(ID_EX_Rd),
    .ID_EX_PC(ID_EX_PC),
    .ID_EX_ALUSrc(ID_EX_ALUSrc),
    .ID_EX_MemToReg(ID_EX_MemToReg),
    .ID_EX_RegWrite(ID_EX_RegWrite),
    .ID_EX_MemRead(ID_EX_MemRead),
    .ID_EX_MemWrite(ID_EX_MemWrite),
    .ID_EX_Branch(ID_EX_Branch),
    .ID_EX_ALUOp(ID_EX_ALUOp)
```

```
);

EX_STAGE dut_EX_STAGE(
    .clk(clk1),
    .rst(rst),
    .ID_EX_ReadData1(ID_EX_ReadData1),
    .ID_EX_ReadData2(ID_EX_ReadData2),
    .ID_EX_SignExtImm(ID_EX_SignExtImm),
    .ID_EX_Rd(ID_EX_Rd),
    .ID_EX_PC(ID_EX_PC),
    .ID_EX_ALUSrc(ID_EX_ALUSrc),
    .ID_EX_MemtoReg(ID_EX_MemToReg),
    .ID_EX_RegWrite(ID_EX_RegWrite),
    .ID_EX_MemRead(ID_EX_MemRead),
    .ID_EX_MemWrite(ID_EX_MemWrite),
    .ID_EX_Branch(ID_EX_Branch),
    .ID_EX_ALUOp(ID_EX_ALUOp),
    .EX_MEM_ALUResult(EX_MEM_ALUResult),
    .EX_MEM_WriteData(EX_MEM_WriteData),
    .EX_MEM_MemWriteOut(EX_MEM_MemWriteOut),
    .EX_MEM_WriteReg(EX_MEM_WriteReg),
    .EX_MEM_MemtoRegOut(EX_MEM_MemtoRegOut),
    .EX_MEM_RegWrite(EX_MEM_RegWrite),
    .EX_MEM_MemReadOut(EX_MEM_MemReadOut),
    .EX_MEM_Branch(EX_MEM_Branch),
    .EX_MEM_BranchTarget(EX_MEM_BranchTarget)
);

MEM_STAGE dut_MEM_STAGE(
    .clk(clk1),
    .clk_RAM(clk2),
    .rst(rst),
    .EX_MEM_ALUResult(EX_MEM_ALUResult),
    .EX_MEM_Data(EX_MEM_WriteData),
    .EX_MEM_MemWrite(EX_MEM_MemWriteOut),
    .EX_MEM_WriteReg(EX_MEM_WriteReg),
    .EX_MEM_MemToReg(EX_MEM_MemtoRegOut),
    .EX_MEM_RegWrite(EX_MEM_RegWrite),
    .EX_MEM_MemRead(EX_MEM_MemReadOut),
    .MEM_WB_ALUResult(MEM_WB_ALUResult),
    .MEM_WB_ReadData(MEM_WB_ReadData),
    .MEM_WB_WriteReg(MEM_WB_WriteReg),
    .MEM_WB_MemToReg(MEM_WB_MemToReg),
        .MEM_WB_RegWrite(MEM_WB_RegWrite)
);

WB_Stage dut_WB_Stage(
    .ALUResult(MEM_WB_ALUResult),
    .MemData(MEM_WB_ReadData),
    .WriteReg(MEM_WB_WriteReg),
    .MemToReg(MEM_WB_MemToReg),
    .RegWrite(MEM_WB_RegWrite),
    .WriteData(WriteData),
    .WriteRegOut(WriteRegOut),
    .RegWriteOut(RegWriteOut)
);

// Geracao do clk1 (periodo = 10 unidades de tempo)
```

```
    initial begin
        clk1 = 1;
        forever #5 clk1 = ~clk1; // Toggle a cada 5 unidades de tempo
    end

    // Geracao do clk2 (período = 5 unidades de tempo)
    initial begin
        clk2 = 1;
        forever #2.5 clk2 = ~clk2; // Toggle a cada 2.5 unidades de tempo
    end

    // Estímulos de teste
    initial begin
        rst = 1;
        $display("Resetando:");
        #10

        rst = 0;
        #10;

        $display("Ler primeira Instrucao:");
        #15;

        $display("Executar primeira Instrucao:");

        #15;

        $display("Ler memoria:");
        #95;

        // Finaliza a simulação
        $finish;
    end

    // Monitoramento dos sinais a cada 5 unidades de tempo, alinhado ao tempo de simulação
        initial begin
        //#5; // Aguarda o primeiro ciclo de 5 unidades de tempo para alinhar
      forever begin
            $display("\nTime=%0t | rst=%b",
                    $time, rst);
          $display("IF_STAGE: BranchTaken=%b | BranchTarget=%d | IF_ID_PC=%d |
IF_ID_Instruction=%b",
                    EX_MEM_Branch, EX_MEM_BranchTarget, IF_ID_PC, IF_ID_Instruction);
          $display("ID_STAGE: ID_EX_ReadData1=%d | ID_EX_ReadData2=%d | ID_EX_SignExtImm=%d |
ID_EX_Rd=%d | ID_EX_PC=%d | ID_EX_Branch=%b | MemToReg=%b | RegWrite=%b | MemRead=%b | ALUOp=%b",
                    ID_EX_ReadData1, ID_EX_ReadData2, ID_EX_SignExtImm, ID_EX_Rd, ID_EX_PC,
ID_EX_Branch, ID_EX_MemToReg, ID_EX_RegWrite, ID_EX_MemRead, ID_EX_ALUOp);
          $display("EX_STAGE: WriteReg=%d | EX_MEM_ALUResult=%d | EX_MEM_WriteData=%d |
EX_MEM_Branch=%b | EX_MEM_BranchTarget=%d | MemRead=%b | ALUSrc=%b | MemToReg=%b",
                    EX_MEM_WriteReg, EX_MEM_ALUResult, EX_MEM_WriteData, EX_MEM_Branch,
EX_MEM_BranchTarget, EX_MEM_MemReadOut, ID_EX_ALUSrc, EX_MEM_MemtoRegOut);
          $display("MEM_STAGE: MEM_WB_ALUResult=%d | MEM_WB_ReadData=%d | MEM_WB_WriteReg=%d |
MemToReg=%b | RegWrite=%b",
                            MEM_WB_ALUResult, MEM_WB_ReadData, MEM_WB_WriteReg, MEM_WB_MemToReg,
MEM_WB_RegWrite);
          $display("WB_STAGE: WriteData=%d | WriteReg=%d | RegWrite=%b",
                            WriteData, WriteRegOut, RegWriteOut);
```

```
        #5; // Espera 5 unidades de tempo antes de imprimir novamente
    end
end

endmodule
```

Testes das Instruções (Interação entre estágios ID)

No código Verilog abaixo, encontra-se o *testbench* utilizado para validação do módulo *Execute* (EX) do processador de 5 estágios. Por ele, testamos a propagação de sinais que são necessários para funcionamento adequado dos estágios que vem em seguida e as operações implementadas na ULA (Unidade Lógica-Aritmética), sejam elas de acesso à memória, aritméticas, lógicas e de saltos. Para isso, simulamos entradas que podem ser geradas pelo estágio anterior (ID).

```
module EX_STAGE_TB;
    // Inputs
    reg clk, rst;
    reg [4:0] ID_EX_ALUOp;
    reg [31:0] ID_EX_ReadData1, ID_EX_ReadData2, ID_EX_SignExtImm;
    reg ID_EX_ALUSrc, ID_EX_RegWrite, ID_EX_MemtoReg;
    reg ID_EX_MemWrite, ID_EX_MemRead, ID_EX_Branch;
    reg [4:0] ID_EX_Rd;
    reg [7:0] ID_EX_PC;
    // Outputs
    wire EX_MEM_MemtoRegOut, EX_MEM_MemWriteOut, EX_MEM_MemReadOut;
    wire EX_MEM_Branch;
    wire [7:0] EX_MEM_BranchTarget;
    wire [31:0] EX_MEM_WriteData, EX_MEM_ALUResult;
    wire [4:0] EX_MEM_WriteReg;
    wire EX_MEM_RegWrite;
    // Unit Under Test (UUT)
    EX_STAGE uut (
        .ID_EX_ALUSrc(ID_EX_ALUSrc),
        .rst(rst),
        .clk(clk),
        .ID_EX_RegWrite(ID_EX_RegWrite),
        .ID_EX_MemtoReg(ID_EX_MemtoReg),
        .ID_EX_MemWrite(ID_EX_MemWrite),
        .ID_EX_MemRead(ID_EX_MemRead),
        .ID_EX_Branch(ID_EX_Branch),
        .ID_EX_ALUOp(ID_EX_ALUOp),
        .ID_EX_PC(ID_EX_PC),
        .ID_EX_Rd(ID_EX_Rd),
        .ID_EX_ReadData1(ID_EX_ReadData1),
        .ID_EX_ReadData2(ID_EX_ReadData2),
        .ID_EX_SignExtImm(ID_EX_SignExtImm),
        .EX_MEM_MemtoRegOut(EX_MEM_MemtoRegOut),
        .EX_MEM_MemWriteOut(EX_MEM_MemWriteOut),
```

```
        .EX_MEM_MemReadOut(EX_MEM_MemReadOut),
        .EX_MEM_Branch(EX_MEM_Branch),
        .EX_MEM_RegWrite(EX_MEM_RegWrite),
        .EX_MEM_ALUResult(EX_MEM_ALUResult),
        .EX_MEM_BranchTarget(EX_MEM_BranchTarget),
        .EX_MEM_WriteData(EX_MEM_WriteData),
        .EX_MEM_WriteReg(EX_MEM_WriteReg)
    );
    // Gerador de Clock
    initial begin
        clk = 1;
        forever #5 clk = ~clk; // Toggle a cada 5 unidades de tempo
    end
    initial begin
        // Inicializa Inputs
        rst = 1;
        ID_EX_ALUOp = 5'b00000;
        ID_EX_ReadData1 = 32'b0;
        ID_EX_ReadData2 = 32'b0;
        ID_EX_SignExtImm = 32'b0;
        ID_EX_ALUSrc = 0;
        ID_EX_PC = 8'b0;
        ID_EX_RegWrite = 0;
        ID_EX_MemtoReg = 0;
        ID_EX_MemWrite = 0;
        ID_EX_MemRead = 0;
        ID_EX_Branch = 0;
        ID_EX_Rd = 5'b00000;
        // Sinal de Reset
        #10;
        rst = 0;
        #10;
        // Teste LW_1 (Load Word com Endereçamento Base-Deslocamento)
        ID_EX_ALUSrc = 1;
        ID_EX_MemtoReg = 1;
        ID_EX_RegWrite = 1;
        ID_EX_MemRead = 1;
        ID_EX_MemWrite = 0;
        ID_EX_Branch = 0;
        ID_EX_Rd = 5'b00001;
        ID_EX_ALUOp = 5'b00000;   // LW_1 15($4), $2
        ID_EX_ReadData1 = 32'd28; // ($4)
        ID_EX_ReadData2 = 32'd8;  // ($2)
        ID_EX_SignExtImm = 32'd15;
        #10;
        // Teste LW_2 (Load Word com Endereçamento Direto)
        ID_EX_ALUSrc = 0;
        ID_EX_MemtoReg = 1;
        ID_EX_RegWrite = 1;
        ID_EX_MemRead = 1;
        ID_EX_MemWrite = 0;
        ID_EX_Branch = 0;
        ID_EX_Rd = 5'b00010;
        ID_EX_ALUOp = 5'b00001; // LW_2 $3, $7
        ID_EX_ReadData1 = 32'd44; // ($3)
        ID_EX_ReadData2 = 32'd2;  // ($7)
        ID_EX_SignExtImm = 32'b0;
        #10;
```

```
        // Teste LW_3 (Load Word com Endereçamento Imediato)
        ID_EX_ALUSrc = 1;
        ID_EX_MemtoReg = 1;
        ID_EX_RegWrite = 1;
        ID_EX_MemRead = 1;
        ID_EX_MemWrite = 0;
        ID_EX_Branch = 0;
        ID_EX_Rd = 5'b00011;
        ID_EX_ALUOp = 5'b00010; // LW_3
        ID_EX_ReadData1 = 32'd0;
        ID_EX_ReadData2 = 32'd10;
        ID_EX_SignExtImm = 32'd17;
        #10;
        // Teste SW_1 (Store Word com Endereçamento Imediato)
        ID_EX_ALUSrc = 1;
        ID_EX_MemtoReg = 0;
        ID_EX_RegWrite = 0;
        ID_EX_MemRead = 0;
        ID_EX_MemWrite = 1;
        ID_EX_Branch = 0;
        ID_EX_Rd = 5'b00100;
        ID_EX_ALUOp = 5'b00011; // SW_1
        ID_EX_ReadData1 = 32'd0;
        ID_EX_ReadData2 = 32'd3;
        ID_EX_SignExtImm = 32'd10;
        #10;
        // Teste SW_2 (Store Word com Endereçamento Direto)
        ID_EX_ALUSrc = 0;
        ID_EX_MemtoReg = 0;
        ID_EX_RegWrite = 0;
        ID_EX_MemRead = 0;
        ID_EX_MemWrite = 1;
        ID_EX_Branch = 0;
        ID_EX_Rd = 5'b00100;
        ID_EX_ALUOp = 5'b00101; // SW_2
        ID_EX_ReadData1 = 32'd10;
        ID_EX_ReadData2 = 32'd10;
        ID_EX_SignExtImm = 32'd0;
        #10;
        // Teste MOV
        ID_EX_ALUSrc = 0;
        ID_EX_MemtoReg = 0;
        ID_EX_RegWrite = 1;
        ID_EX_MemRead = 0;
        ID_EX_MemWrite = 0;
        ID_EX_Branch = 0;
        ID_EX_Rd = 5'b00110;
        ID_EX_ALUOp = 5'b00101; // MOV
        ID_EX_ReadData1 = 32'd22;
        ID_EX_ReadData2 = 32'd0;
        ID_EX_SignExtImm = 32'd0;
        #10

        // Teste ADD
        ID_EX_ALUSrc = 0;
        ID_EX_MemtoReg = 0;
        ID_EX_RegWrite = 1;
        ID_EX_MemRead = 0;
```

```
        ID_EX_MemWrite = 0;
        ID_EX_Branch = 0;
        ID_EX_Rd = 5'b00111;
        ID_EX_ALUOp = 5'b00110; // ADD
        ID_EX_ReadData1 = 32'd10;
        ID_EX_ReadData2 = 32'd5;
        ID_EX_SignExtImm = 32'd0;
        #10;
        // Teste SUB
        ID_EX_ALUSrc = 0;
        ID_EX_MemtoReg = 0;
        ID_EX_RegWrite = 1;
        ID_EX_MemRead = 0;
        ID_EX_MemWrite = 0;
        ID_EX_Branch = 0;
        ID_EX_Rd = 5'b01000;
        ID_EX_ALUOp = 5'b00111; // SUB
        ID_EX_ReadData1 = 32'd10;
        ID_EX_ReadData2 = 32'd5;
        ID_EX_SignExtImm = 32'd0;
        #10;
        // Teste MUL
        ID_EX_ALUSrc = 0;
        ID_EX_MemtoReg = 0;
        ID_EX_RegWrite = 1;
        ID_EX_MemRead = 0;
        ID_EX_MemWrite = 0;
        ID_EX_Branch = 0;
        ID_EX_Rd = 5'b01001;
        ID_EX_ALUOp = 5'b01000; // MUL
        ID_EX_ReadData1 = 32'd10;
        ID_EX_ReadData2 = 32'd5;
        ID_EX_SignExtImm = 32'd0;
        #10;
        // Teste DIV
        ID_EX_ALUSrc = 0;
        ID_EX_MemtoReg = 0;
        ID_EX_RegWrite = 1;
        ID_EX_MemRead = 0;
        ID_EX_MemWrite = 0;
        ID_EX_Branch = 0;
        ID_EX_Rd = 5'b01010;
        ID_EX_ALUOp = 5'b01001; // DIV
        ID_EX_ReadData1 = 32'd10;
        ID_EX_ReadData2 = 32'd5;
        ID_EX_SignExtImm = 32'd0;
        #10;
        // Teste AND
        ID_EX_ALUSrc = 0;
        ID_EX_MemtoReg = 0;
        ID_EX_RegWrite = 1;
        ID_EX_MemRead = 0;
        ID_EX_MemWrite = 0;
        ID_EX_Branch = 0;
        ID_EX_Rd = 5'b01011;
        ID_EX_ALUOp = 5'b01010; // AND
        ID_EX_ReadData1 = 32'd1;
        ID_EX_ReadData2 = 32'd1;
```

```
        ID_EX_SignExtImm = 32'd0;
        #10;
        // Teste OR
        ID_EX_ALUSrc = 0;
        ID_EX_MemtoReg = 0;
        ID_EX_RegWrite = 1;
        ID_EX_MemRead = 0;
        ID_EX_MemWrite = 0;
        ID_EX_Branch = 0;
        ID_EX_Rd = 5'b01100;
        ID_EX_ALUOp = 5'b01011; // OR
        ID_EX_ReadData1 = 32'd1;
        ID_EX_ReadData2 = 32'd0;
        ID_EX_SignExtImm = 32'd0;
        #10;
        // Teste SHL
        ID_EX_ALUSrc = 0;
        ID_EX_MemtoReg = 0;
        ID_EX_RegWrite = 1;
        ID_EX_MemRead = 0;
        ID_EX_MemWrite = 0;
        ID_EX_Branch = 0;
        ID_EX_Rd = 5'b01101;
        ID_EX_ALUOp = 5'b01100; // SHL
        ID_EX_ReadData1 = 32'd15;
        ID_EX_ReadData2 = 32'd3;
        ID_EX_SignExtImm = 32'd0;
        #10;
        // Teste SHR
        ID_EX_ALUSrc = 0;
        ID_EX_MemtoReg = 0;
        ID_EX_RegWrite = 1;
        ID_EX_MemRead = 0;
        ID_EX_MemWrite = 0;
        ID_EX_Branch = 0;
        ID_EX_Rd = 5'b01110;
        ID_EX_ALUOp = 5'b01101; // SHR
        ID_EX_ReadData1 = 32'd15;
        ID_EX_ReadData2 = 32'd3;
        ID_EX_SignExtImm = 32'd0;
        #10;
        // Teste CMP
        ID_EX_ALUSrc = 0;
        ID_EX_MemtoReg = 0;
        ID_EX_RegWrite = 0;
        ID_EX_MemRead = 0;
        ID_EX_MemWrite = 0;
        ID_EX_Branch = 0;
        ID_EX_Rd = 5'b01111;
        ID_EX_ALUOp = 5'b01110; // CMP
        ID_EX_ReadData1 = 32'd15;
        ID_EX_ReadData2 = 32'd15;
        ID_EX_SignExtImm = 32'd0;
        #10;
        // Teste NOT
        ID_EX_ALUSrc = 0;
        ID_EX_MemtoReg = 0;
        ID_EX_RegWrite = 1;
```

```
        ID_EX_MemRead = 0;
        ID_EX_MemWrite = 0;
        ID_EX_Branch = 0;
        ID_EX_Rd = 5'b10000;
        ID_EX_ALUOp = 5'b01111; // NOT
        ID_EX_ReadData1 = 32'd0;
        ID_EX_SignExtImm = 32'd0;
        #10;
        // Teste JR
        ID_EX_ALUSrc = 0;
        ID_EX_MemtoReg = 0;
        ID_EX_RegWrite = 0;
        ID_EX_MemRead = 0;
        ID_EX_MemWrite = 0;
        ID_EX_Branch = 1;
        ID_EX_Rd = 5'b10001;
        ID_EX_ALUOp = 5'b10000; // JR
        ID_EX_ReadData1 = 32'd10;
        ID_EX_SignExtImm = 32'd0;
        #10;
        // Teste JPC
        ID_EX_ALUSrc = 1;
        ID_EX_MemtoReg = 0;
        ID_EX_RegWrite = 0;
        ID_EX_MemRead = 0;
        ID_EX_MemWrite = 0;
        ID_EX_PC = 8'b00010000;
        ID_EX_Branch = 1;
        ID_EX_Rd = 5'b10010;
        ID_EX_ALUOp = 5'b10001; // JPC
        ID_EX_ReadData2 = 32'd5;
        ID_EX_SignExtImm = 32'd8;
        #10;
        // Teste BRFL
        ID_EX_ALUSrc = 0;
        ID_EX_MemtoReg = 0;
        ID_EX_RegWrite = 0;
        ID_EX_MemRead = 0;
        ID_EX_MemWrite = 0;
        ID_EX_Branch = 1;
        ID_EX_Rd = 5'b10011;
        ID_EX_ALUOp = 5'b10010; // BRFL
        ID_EX_ReadData1 = 32'd20;
        ID_EX_ReadData2 = 5'b00001; // Máscara para Flag Igual
        ID_EX_SignExtImm = 32'b0;
        #10;
        // Teste CALL
        ID_EX_ALUSrc = 0;
        ID_EX_MemtoReg = 0;
        ID_EX_RegWrite = 1;
        ID_EX_MemRead = 0;
        ID_EX_MemWrite = 0;
        ID_EX_Branch = 1;
        ID_EX_Rd = 5'b10100;
        ID_EX_ALUOp = 5'b10011; // CALL
        ID_EX_ReadData1 = 32'd10;
        ID_EX_SignExtImm = 32'b0;
        #10;
```

```
        // Teste RET
        ID_EX_ALUSrc = 0;
        ID_EX_MemtoReg = 0;
        ID_EX_RegWrite = 0;
        ID_EX_MemRead = 0;
        ID_EX_MemWrite = 0;
        ID_EX_Branch = 1;
        ID_EX_Rd = 5'b10101;
        ID_EX_ALUOp = 5'b10100; // RET
        ID_EX_ReadData2 = 32'd31;
        ID_EX_SignExtImm = 32'd0;
        #10;
        // Teste NOP
        ID_EX_ALUSrc = 0;
        ID_EX_MemtoReg = 0;
        ID_EX_RegWrite = 0;
        ID_EX_MemRead = 0;
        ID_EX_MemWrite = 0;
        ID_EX_Branch = 0;
        ID_EX_ALUOp = 5'b10101; // NOP
        #10;
        // End simulation
        $stop;
    end
    initial begin
        forever begin
            // Amostra de Resultados
            $display("\nTime=%0t | rst=%b",
                    $time, rst);
            $display("ID_STAGE: ALUSrc =%b | MemToReg =%b | RegWrite =%b | MemRead =%b | MemWrite
=%b | Branch =%b | ALUOp=%b | ReadData1=%d | ReadData2=%d | PC=%d",
                    ID_EX_ALUSrc, ID_EX_MemtoReg, ID_EX_RegWrite, ID_EX_MemRead, ID_EX_MemWrite,
ID_EX_Branch, ID_EX_ALUOp, ID_EX_ReadData1, ID_EX_ReadData2, ID_EX_PC);
            $display("EX_STAGE: ALUResult=%d | WriteData=%d | MemWrite=%b | MemRead=%b | WriteReg=%b
| MemtoReg=%b | RegWrite=%b | Branch=%b | BranchTarget=%d",
                    EX_MEM_ALUResult, EX_MEM_WriteData, EX_MEM_MemWriteOut, EX_MEM_MemReadOut,
EX_MEM_WriteReg, EX_MEM_MemtoRegOut, EX_MEM_RegWrite, EX_MEM_Branch, EX_MEM_BranchTarget);
            #5;
        end
    end
endmodule
```

Universidade Federal da Bahia
Escola Politécnica
Departamento de Engenharia Elétrica e de Computação
Docente: Wagner Oliveira
Discentes: Caio Mendes, Fábio Miguel, Gerson Daniel, Heverton Reis e José Alves
Semestre 2024.2

Prototipação

Visando a execução do modelo de processador com pipeline na plataforma FPGA Altera Cyclone IV modelo tal, o arquivo de pinagem DE2_115.qsf foi adicionado ao projeto para que fosse possível relacionar os pinos de entrada do processador com as conexões reais da placa. Assim, definiu-se os pinos da seguinte forma:

- Clock dos registradores de pipeline foi associado ao PIN_Y2 com o nome CLOCK_50
- Clock das memórias e do Banco de Registradores de 32 bits foi associado ao PIN_AG14 com o nome CLOCK2_50

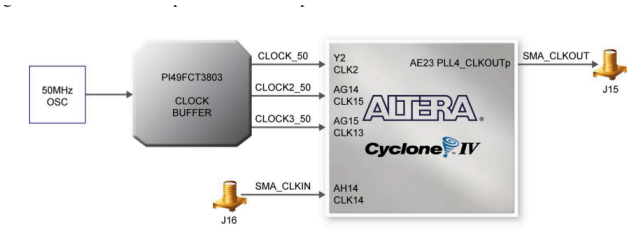

Figure 4-11 Block diagram of the clock distribution

Table 4-5 Pin Assignments for Clock Inputs

Signal Name	FPGA Pin No.	Description	I/O Standard
CLOCK_50	PIN_Y2	50 MHz clock input	3.3V
CLOCK2_50	PIN_AG14	50 MHz clock input	3.3V
CLOCK3_50	PIN_AG15	50 MHz clock input	Depending on JP6
SMA_CLKOUT	PIN_AE23	External (SMA) clock output	Depending on JP6
SMA_CLKIN	PIN_AH14	External (SMA) clock input	3.3V

Figura 1 - Tabela de pinos para os clocks

- Reset do sistema associado ao PIN_M23 com o nome KEY[0]

Table 4-2 Pin Assignments for Push-buttons

Signal Name	FPGA Pin No.	Description	I/O Standard
KEY[0]	PIN_M23	Push-button[0]	Depending on JP7
KEY[1]	PIN_M21	Push-button[1]	Depending on JP7
KEY[2]	PIN_N21	Push-button[2]	Depending on JP7
KEY[3]	PIN_R24	Push-button[3]	Depending on JP7

Figura 2 - Tabela de pinos para os push-buttons

Universidade Federal da Bahia
Escola Politécnica
Departamento de Engenharia Elétrica e de Computação
Docente: Wagner Oliveira
Discentes: Caio Mendes, Fábio Miguel, Gerson Daniel, Heverton Reis e José Alves
Semestre 2024.2

Análise Temporal Estática (STA)

Para garantir que o sistema opere nas restrições de temporização correta, o circuito foi submetido à análise temporal pelo Timing Analyzer. Para tal, foi necessário a criação dos clocks com as seguintes configurações:

```
create_clock -name {CLOCK_50} -period 170.000 -waveform { 0.000 85.000 } [get_ports {CLOCK_50}]

create_clock -name {CLOCK2_50} -period 85.000 -waveform { 0.000 42.500 } [get_ports {CLOCK2_50}]
```

Em seguida, a análise de slack foi realizada para diferentes condições operacionais, após o ajuste de clock, considerando a pior e melhor situação do sistema,

- Slow 1200mV 85C:
    - O slack apresentou valores reduzidos, mas ainda dentro dos limites operacionais.
    - Foram realizados ajustes no input/output delay para melhorar a estabilidade.

Slow 1200mV 85C Model

Command Info | Summary of Paths

	Slack	From Node	To Node	Launch Clock	Latch Clock	Relationship	Clock Skew	Data Delay
1	24.719	ID_STAGE:inst1|ID_EX_Register:inst5|outALUSrc	EX_STAGE:inst3|EX_MEM_REGISTER:inst12|outBranchTaken	CLOCK_50	CLOCK_50	170.000	-0.119	145.160
2	24.720	ID_STAGE:inst1|ID_EX_Register:inst5|outALUSrc	EX_STAGE:inst3|EX_MEM_REGISTER:inst12|outBranchTaken	CLOCK_50	CLOCK_50	170.000	-0.119	145.159
3	24.720	ID_STAGE:inst1|ID_EX_Register:inst5|outALUSrc	EX_STAGE:inst3|EX_MEM_REGISTER:inst12|outBranchTaken	CLOCK_50	CLOCK_50	170.000	-0.119	145.159
4	24.721	ID_STAGE:inst1|ID_EX_Register:inst5|outALUSrc	EX_STAGE:inst3|EX_MEM_REGISTER:inst12|outBranchTaken	CLOCK_50	CLOCK_50	170.000	-0.119	145.158
5	24.725	ID_STAGE:inst1|ID_EX_Register:inst5|outALUSrc	EX_STAGE:inst3|EX_MEM_REGISTER:inst12|outBranchTaken	CLOCK_50	CLOCK_50	170.000	-0.119	145.154
6	24.726	ID_STAGE:inst1|ID_EX_Register:inst5|outALUSrc	EX_STAGE:inst3|EX_MEM_REGISTER:inst12|outBranchTaken	CLOCK_50	CLOCK_50	170.000	-0.119	145.153
7	24.726	ID_STAGE:inst1|ID_EX_Register:inst5|outALUSrc	EX_STAGE:inst3|EX_MEM_REGISTER:inst12|outBranchTaken	CLOCK_50	CLOCK_50	170.000	-0.119	145.153
8	24.726	ID_STAGE:inst1|ID_EX_Register:inst5|outALUSrc	EX_STAGE:inst3|EX_MEM_REGISTER:inst12|outBranchTaken	CLOCK_50	CLOCK_50	170.000	-0.119	145.153
9	24.727	ID_STAGE:inst1|ID_EX_Register:inst5|outALUSrc	EX_STAGE:inst3|EX_MEM_REGISTER:inst12|outBranchTaken	CLOCK_50	CLOCK_50	170.000	-0.119	145.152
10	24.727	ID_STAGE:inst1|ID_EX_Register:inst5|outALUSrc	EX_STAGE:inst3|EX_MEM_REGISTER:inst12|outBranchTaken	CLOCK_50	CLOCK_50	170.000	-0.119	145.152

Figura 3: Caminhos para Slow 1200mV 85c

- Slow 1200mV 0C:
    - A análise mostrou comportamento intermediário entre os dois cenários acima.
    - Os ajustes de delays foram mantidos para garantir operação confiável.

Slow 1200mV 0C Model

Command Info | Summary of Paths

	Slack	From Node	To Node	Launch Clock	Latch Clock	Relationship	Clock Skew	Data Delay
1	38.839	ID_STAGE:inst1|ID_EX_Register:inst5|outALUSrc	EX_STAGE:inst3|EX_MEM_REGISTER:inst12|outBranchTaken	CLOCK_50	CLOCK_50	170.000	-0.107	131.053
2	38.842	ID_STAGE:inst1|ID_EX_Register:inst5|outALUSrc	EX_STAGE:inst3|EX_MEM_REGISTER:inst12|outBranchTaken	CLOCK_50	CLOCK_50	170.000	-0.107	131.050
3	38.848	ID_STAGE:inst1|ID_EX_Register:inst5|outALUSrc	EX_STAGE:inst3|EX_MEM_REGISTER:inst12|outBranchTaken	CLOCK_50	CLOCK_50	170.000	-0.107	131.044
4	38.849	ID_STAGE:inst1|ID_EX_Register:inst5|outALUSrc	EX_STAGE:inst3|EX_MEM_REGISTER:inst12|outBranchTaken	CLOCK_50	CLOCK_50	170.000	-0.107	131.043
5	38.851	ID_STAGE:inst1|ID_EX_Register:inst5|outALUSrc	EX_STAGE:inst3|EX_MEM_REGISTER:inst12|outBranchTaken	CLOCK_50	CLOCK_50	170.000	-0.107	131.041
6	38.852	ID_STAGE:inst1|ID_EX_Register:inst5|outALUSrc	EX_STAGE:inst3|EX_MEM_REGISTER:inst12|outBranchTaken	CLOCK_50	CLOCK_50	170.000	-0.107	131.040
7	38.853	ID_STAGE:inst1|ID_EX_Register:inst5|outALUSrc	EX_STAGE:inst3|EX_MEM_REGISTER:inst12|outBranchTaken	CLOCK_50	CLOCK_50	170.000	-0.107	131.039
8	38.854	ID_STAGE:inst1|ID_EX_Register:inst5|outALUSrc	EX_STAGE:inst3|EX_MEM_REGISTER:inst12|outBranchTaken	CLOCK_50	CLOCK_50	170.000	-0.107	131.038
9	38.855	ID_STAGE:inst1|ID_EX_Register:inst5|outALUSrc	EX_STAGE:inst3|EX_MEM_REGISTER:inst12|outBranchTaken	CLOCK_50	CLOCK_50	170.000	-0.107	131.037
10	38.856	ID_STAGE:inst1|ID_EX_Register:inst5|outALUSrc	EX_STAGE:inst3|EX_MEM_REGISTER:inst12|outBranchTaken	CLOCK_50	CLOCK_50	170.000	-0.107	131.036

Figura 4: Caminhos para Slow 1200mV 0c

- Fast 1200mV 0C:
  - O slack medido foi positivo, indicando que os caminhos críticos respeitam as margens de temporização.
  - Nenhum ajuste adicional foi necessário.

Fast 1200mV 0C Model

Command Info | Summary of Paths

	Slack	From Node	To Node	Launch Clock	Latch Clock	Relationship	Clock Skew	Data Delay
1	98.016	ID_STAGE:inst1|ID_EX_Register:inst5|outALUSrc	EX_STAGE:inst3|EX_MEM_REGISTER:inst12|outBranchTaken	CLOCK_50	CLOCK_50	170.000	-0.066	71.905
2	98.019	ID_STAGE:inst1|ID_EX_Register:inst5|outALUSrc	EX_STAGE:inst3|EX_MEM_REGISTER:inst12|outBranchTaken	CLOCK_50	CLOCK_50	170.000	-0.066	71.902
3	98.020	ID_STAGE:inst1|ID_EX_Register:inst5|outALUSrc	EX_STAGE:inst3|EX_MEM_REGISTER:inst12|outBranchTaken	CLOCK_50	CLOCK_50	170.000	-0.066	71.901
4	98.022	ID_STAGE:inst1|ID_EX_Register:inst5|outALUSrc	EX_STAGE:inst3|EX_MEM_REGISTER:inst12|outBranchTaken	CLOCK_50	CLOCK_50	170.000	-0.066	71.899
5	98.023	ID_STAGE:inst1|ID_EX_Register:inst5|outALUSrc	EX_STAGE:inst3|EX_MEM_REGISTER:inst12|outBranchTaken	CLOCK_50	CLOCK_50	170.000	-0.066	71.898
6	98.023	ID_STAGE:inst1|ID_EX_Register:inst5|outALUSrc	EX_STAGE:inst3|EX_MEM_REGISTER:inst12|outBranchTaken	CLOCK_50	CLOCK_50	170.000	-0.066	71.898
7	98.024	ID_STAGE:inst1|ID_EX_Register:inst5|outALUSrc	EX_STAGE:inst3|EX_MEM_REGISTER:inst12|outBranchTaken	CLOCK_50	CLOCK_50	170.000	-0.066	71.897
8	98.025	ID_STAGE:inst1|ID_EX_Register:inst5|outALUSrc	EX_STAGE:inst3|EX_MEM_REGISTER:inst12|outBranchTaken	CLOCK_50	CLOCK_50	170.000	-0.066	71.896
9	98.026	ID_STAGE:inst1|ID_EX_Register:inst5|outALUSrc	EX_STAGE:inst3|EX_MEM_REGISTER:inst12|outBranchTaken	CLOCK_50	CLOCK_50	170.000	-0.066	71.895
10	98.026	ID_STAGE:inst1|ID_EX_Register:inst5|outALUSrc	EX_STAGE:inst3|EX_MEM_REGISTER:inst12|outBranchTaken	CLOCK_50	CLOCK_50	170.000	-0.066	71.895

Figura 5: Caminhos para Fast 1200mV 0c

Configuração de Incerteza de Clock

A incerteza de clock representa as variações no tempo de propagação de sinais e oscilações internas do clock. Para garantir uma margem de segurança na análise de temporização, foi adicionada uma incerteza de **0.020 ns** entre diferentes transições dos sinais de clock.

```
set_clock_uncertainty -rise_from [get_clocks {altera_reserved_tck}] -rise_to [get_clocks
{altera_reserved_tck}]  0.020
set_clock_uncertainty -rise_from [get_clocks {altera_reserved_tck}] -fall_to [get_clocks
{altera_reserved_tck}]  0.020
set_clock_uncertainty -fall_from [get_clocks {altera_reserved_tck}] -rise_to [get_clocks
{altera_reserved_tck}]  0.020
set_clock_uncertainty -fall_from [get_clocks {altera_reserved_tck}] -fall_to [get_clocks
{altera_reserved_tck}]  0.020
set_clock_uncertainty -rise_from [get_clocks {CLOCK_50}] -rise_to [get_clocks {CLOCK_50}]  0.020
set_clock_uncertainty -rise_from [get_clocks {CLOCK_50}] -fall_to [get_clocks {CLOCK_50}]  0.020
```

```
set_clock_uncertainty -fall_from [get_clocks {CLOCK_50}] -rise_to [get_clocks {CLOCK_50}]  0.020
set_clock_uncertainty -fall_from [get_clocks {CLOCK_50}] -fall_to [get_clocks {CLOCK_50}]  0.020
```

Configuração de Input/Output Delays

Também foi necessário a inserção de inputs/outputs delays para Unconstrained Paths no Timing Analyzer. Abaixo estão as configurações aplicadas para resolver os caminhos não restritos:

3.1. Atrasos de Entrada (Input Delay)

Os seguintes comandos foram utilizados para adicionar atrasos de entrada de **15 ns** a diferentes portas do sistema:

```
set_input_delay -add_delay  -clock [get_clocks {CLOCK_50}]  15.000 [get_ports {CLOCK2_50}]
set_input_delay -add_delay  -clock [get_clocks {CLOCK_50}]  15.000 [get_ports {CLOCK_50}]
set_input_delay -add_delay  -clock [get_clocks {CLOCK_50}]  15.000 [get_ports {KEY[0]}]
set_input_delay -add_delay  -clock [get_clocks {CLOCK_50}]  15.000 [get_ports {altera_reserved_tck}]
set_input_delay -add_delay  -clock [get_clocks {CLOCK_50}]  15.000 [get_ports {altera_reserved_tdi}]
set_input_delay -add_delay  -clock [get_clocks {CLOCK_50}]  15.000 [get_ports {altera_reserved_tms}]
```

3.2. Atrasos de Saída (Output Delay)

Foi também necessário adicionar um atraso de saída de **15 ns** na porta de saída `altera_reserved_tdo`:

```
set_output_delay -add_delay  -clock [get_clocks {CLOCK_50}]  15.000 [get_ports
{altera_reserved_tdo}]
```

Definição de Grupos Assíncronos de Clock

Para evitar que o **Timing Analyzer** interprete caminhos entre domínios de clock que não devem ser analisados, foi necessário definir **grupos assíncronos**. Esses grupos garantem que não haja considerações de temporização entre diferentes sinais de clock que operam de forma independente.

```
set_clock_groups -asynchronous -group [get_clocks {altera_reserved_tck}]
set_clock_groups -asynchronous -group [get_clocks {altera_reserved_tck}]
set_clock_groups -asynchronous -group [get_clocks {altera_reserved_tck}]
set_clock_groups -asynchronous -group [get_clocks {altera_reserved_tck}]
set_clock_groups -asynchronous -group [get_clocks {altera_reserved_tck}]
set_clock_groups -asynchronous -group [get_clocks {altera_reserved_tck}]
set_clock_groups -asynchronous -group [get_clocks {altera_reserved_tck}]
```

```
set_clock_groups -asynchronous -group [get_clocks {altera_reserved_tck}]
```

Definição de Caminhos Falsos (False Paths)

Os caminhos falsos são definidos para que determinadas transições entre domínios de clock sejam ignoradas pelo **Timing Analyzer**, evitando alertas de violações irrelevantes e garantindo que apenas caminhos relevantes sejam considerados.

Os seguintes comandos foram utilizados para definir caminhos falsos entre os sinais de clock do sistema:

```
set_false_path  -from  [get_clocks {CLOCK_50}]  -to  [get_clocks {CLOCK2_50}]
set_false_path  -from  [get_clocks {CLOCK2_50}]  -to  [get_clocks {CLOCK_50}]
```

Impacto na Temporização

A configuração da **incerteza de clock**, dos **grupos assíncronos** e dos **caminhos falsos** garantiu que o projeto fosse analisado corretamente, evitando interferências indesejadas entre diferentes domínios de clock. Essas configurações asseguram que apenas os caminhos válidos sejam considerados na análise de temporização, aumentando a precisão e confiabilidade do circuito.

Conclusão

O desenvolvimento do processador MIPS 32 bits com pipeline de cinco estágios foi concluído, garantindo uma implementação modular e eficiente. A análise temporal estática permitiu avaliar e ajustar a temporização do circuito, garantindo sua estabilidade em diferentes condições operacionais. A configuração de clocks, incertezas, atrasos de entrada e saída, bem como a definição de caminhos assíncronos e caminhos falsos, garantiu que o projeto fosse analisado corretamente sem interferências indesejadas.

A prototipação na FPGA Altera Cyclone IV mostrou que a arquitetura implementada é funcional e pode ser utilizada como base para estudos acadêmicos e experimentos práticos em arquitetura de computadores. O projeto demonstrou um bom equilíbrio entre complexidade e desempenho, sendo uma solução viável para o ensino e a pesquisa no campo de sistemas digitais e microprocessadores.