CI INOVADOR RESIDÊNCIA EM MICROELETRÔNICA	**Programa CI Inovador** **Disciplina: Introdução à Verificação** **Profs.: Joseana Macêdo Fechine Régis de Araújo,** **Marcos Morais, Elmar Melcher - UASC/UAEE/UFCG** **PROJETO FINAL**

Observação: Enviar o relatório do projeto em um arquivo denominado "CI_IntroducaoVerificacao_ProjetoFinal_seunome".

OBJETIVO: Análise de técnicas para implementação da etapa de Verificação do projeto de uma Unidade Lógica e Aritmética (ULA).

SUBPROJETO 1: Simular e verificar o funcionamento do circuito de uma ULA, com especificação no documento Simple_Arithmetic_and_Logic_Unit_Spec_V1.pdf (anexo, disponível em Universal_Verification_Methodology/Arithmetic_Logic_Unit_1.0) e descrição apresentada a seguir.

a) Utilizar as seguintes estratégias para implementação do *Testbench*: *Testbench* Simples; *Testbench* - Checagem automática (*self checking*) e *Testbench* - Checagem automática *(self checking) with testvectors*. Utilizar como referência o documento da Atividade Prática 01 (anexo).
b) Elaborar um conjunto de teste com, no mínimo, duas operações de cada tipo (soma, subtração, multiplicação e divisão).
c) Para cada implementação do *Testbench*, apresentar e analisar os resultados obtidos (introduzir também condições de erro). Para complementar a análise dos resultados, apresentar "prints" das entradas e saídas, exibidas na forma textual e por meio das formas de onda/diagramas de tempo.

design.sv (fonte: SV: ALU Verification [LIVE] - EDA Playground)

```
//------------------------------------------------------
//ALU Design Version 1.0
//------------------------------------------------------

/*
ALU Arithmetic and Logic Operations
-----------------------------------------------------------------
|ALU_Sel|   ALU Operation
-----------------------------------------------------------------
| 0000 |   ALU_Out = A + B;
-----------------------------------------------------------------
| 0001 |   ALU_Out = A - B;
-----------------------------------------------------------------
| 0010 |   ALU_Out = A * B;
-----------------------------------------------------------------
| 0011 |   ALU_Out = A / B;
-----------------------------------------------------------------
*/
```

```
//------------------------------------------------------
// NOTE TO A VERIFICATION ENGINEER:
//
// AS A VERIFICATION ENGINEER, ONE SHOULD NEVER PEEK INTO
// DESIGN AND TRY TO REVERSE ENGINEER WHAT'S HAPPENING.
// JUST FOCUS ON GIVEN SPEC AND BUILD YOUR TESTBENCH ACCORDINGLY.
//------------------------------------------------------
module alu(
  input clock,
  input reset,
  input [7:0] A,B,  // ALU 8-bit Inputs
  input [3:0] ALU_Sel,// ALU Selection
  output reg [7:0] ALU_Out, // ALU 8-bit Output
  output bit CarryOut // Carry Out Flag
);

  reg [7:0] ALU_Result;
  wire [8:0] tmp;

  assign tmp = {1'b0,A} + {1'b0,B};

  always @(posedge clock or posedge reset) begin
    if(reset) begin
      ALU_Out  <= 8'd0;
      CarryOut <= 1'd0;
    end
    else begin
      ALU_Out <= ALU_Result;
      CarryOut <= tmp[8];
    end
  end

  always @(*) //introduce a bug here, A B sensitivity only, change ALU_Sel during
    begin
      case(ALU_Sel)
        4'b0000: // Addition
          ALU_Result = A + B ;
        4'b0001: // Subtraction
          ALU_Result = A - B ;
        4'b0010: // Multiplication
          ALU_Result = A * B;
        4'b0011: // Division
          ALU_Result = A/B;
        default: ALU_Result = 8'hAC ; // Give out random BAD value
      endcase
    end

endmodule
```

SUBPROJETO 2: Realizar a Verificação Funcional do circuito da ULA descrita no SUBPROJETO 1, **utilizando a Metodologia de Verificação Funcional UVM**, com os blocos implementados em SV: ALU Verification [LIVE] - EDA Playground.

a) Descrever a função de cada bloco do Ambiente de Verificação UVM utilizado e as ações implementadas no bloco.
b) Apresentar e analisar os resultados obtidos. Para complementar a análise dos resultados, apresentar "prints" das entradas e saídas, exibidas na forma textual e por meio das formas de onda/diagramas de tempo.
c) Apresentar uma análise comparativa entre as estratégias de verificação adotadas nos subprojetos 1 e 2.

Obs.:

- Exemplo de trecho do código do *Testbench* que possibilita gerar as formas de onda dos sinais.

```
88   //------------------------------------------------------------
89   //Generate Waveforms
90   //------------------------------------------------------------
91   initial begin
92     $dumpfile("d.vcd");
93     $dumpvars();
94   end
```

- Para visualizar as formas de onda dos sinais, utilizando o EDA PlayGround, selecionar a opção "Open EPWave after run", conforme figura abaixo.