Universidade Federal De Campina Grande Departamento De Engenharia Elétrica
Laboratório De Arquitetura De Sistemas Digitais – LASD
Prof. Rafael B. C. Lima

Universidade Federal de Campina Grande

Aluno: _____
Matrícula: _____ Data: _____

Sprint 1 – Revisão de Verilog/SystemVerilog – Simulação (Combinacional)

Descrição geral do problema: Revisão de Verilog/SystemVerilog. Implemente uma série de módulos combinacionais e seus respectivos testbenches. Simule-os no ambiente https://edaplayground.com/ .

Preparação para a sprint:

1. Revise os conceitos aprendidos na disciplina de Circuitos Lógicos. Sugestão: capítulos 1 ao 5 do livro de referência, "*Sarah Harris and David Harris. Digital Design and Computer Architecture, RISC-V Edition. 1. Morgan Kaufmann. 2021*".
2. Crie uma conta no https://edaplayground.com/ e inicie um projeto novo.
   - Language: *SystemVerilog/Verilog*.
   - Tools & Simulator: *Aldec Riviera Pro 2023.04*.
   - Habilite a opção: Open *EPWave after run*.
3. Assista os vídeos de revisão de Verilog, na seguinte playlist (infelizmente ainda não temos material gravado em SystemVerilog): https://youtube.com/playlist?list=PLKM6TRQ8YHKfgnljhJUu1f4dJsxcXkhom&si=zEG01mgT9oXouUz4
4. Assista o tutorial de como criar testbenches e simular módulos no EDAPlayground: https://www.youtube.com/watch?v=VsP6zHarUSM&list=PLKM6TRQ8YHKfgnljhJUu1f4dJsxcXkhom&index=8

Requisitos mínimos:

1. Segue um código simples para implementar uma porta XOR e simulá-la por meio de um *testbench*. Inspecione o código e copie-o para o EDA Playground. *Dica: assista o video*.

```
// Porta XOR 2x1
module XOR2x1 (input in0, in1, output out);
  assign out = in0 ^ in1; //out = (in0 & ~in1) | (~in0 & in1)
endmodule
```

Módulo de uma porta XOR 2x1

```
// Testbench para o XOR 2x1 - Entradas manuais
module XOR2x1_TB;

 // Sinais
 logic  Ia, Ib;
 logic Out;

 // Instância da porta XOR 2x1
 XOR2x1 tes(.in0(Ia), .in1(Ib), .out(Out));

 initial
  begin
   //Salva os sinais do TB em um arquivo .vcd
   $dumpfile("test.vcd"); $dumpvars(1);

   //Monitora os sinais de entrada e saída
   $monitor("time=%3d, Ia=%h, Ib=%h, Out=%h",$time,Ia,Ib,Out);

   //Geração dos sinais de entra (estímulos)
   Ia=1'b0; Ib=1'b0;
```

```
    #10;
    Ia=1'b1; Ib=1'b0;
    #10;
    Ia=1'b0; Ib=1'b1;
    #10;
    Ia=1'b1; Ib=1'b1;
    #10;
    $display("Terminou");
  end
endmodule
```

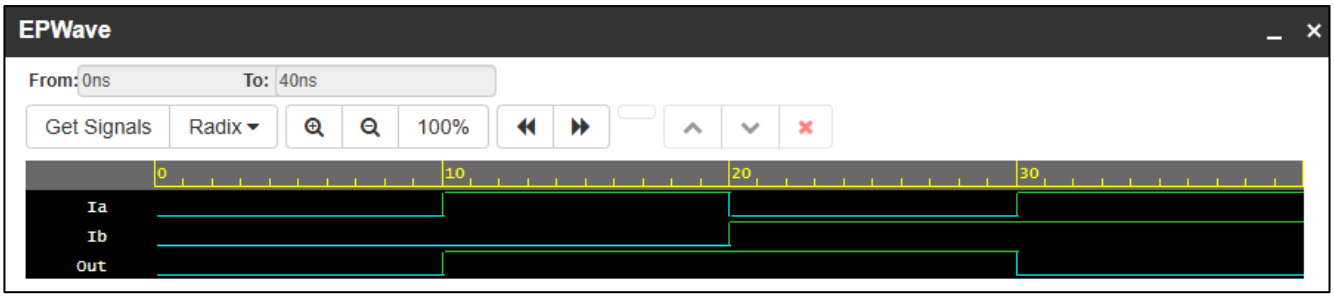
Testbench simples para a porta XOR 2x1

2. Simule o testbench (RUN) e inspecione os sinais de entradas e saídas da porta XOR. Estão conforme o esperado?

LOG:

```
# KERNEL: time=  0, Ia=0, Ib=0, Out=0
# KERNEL: time= 10, Ia=1, Ib=0, Out=1
# KERNEL: time= 20, Ia=0, Ib=1, Out=1
# KERNEL: time= 30, Ia=1, Ib=1, Out=0
# KERNEL: Terminou
# KERNEL: Simulation has finished. There are no more test vectors to simulate.
# VSIM: Simulation has finished.
```

Log de saída do EDA Playground



Sinais de entrada e saída

3. Implemente um **módulo somador de 4bits**, sem entrada de carry-in, nem saída de carry-out. Escolha o nível de abstração que mais lhe convém. *Dica: assista o video*.

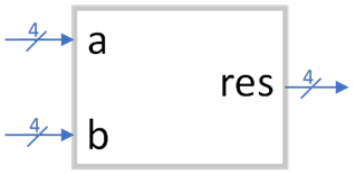


Diagrama de blocos do somador de 4bits

4. Implemente um testbench simples, por meio de delays, para validar um cenário de 5 somas distintas utilizando seu **módulo somador de 4bits**. Simule-o no EDA Playground e cheque se os resultados estão de acordo com o previsto. Chame o professor para Validar seu progresso até esse ponto.

5. Implemente um **módulo MUX 2x1 de 4bits**. *Dica: assista o video*.

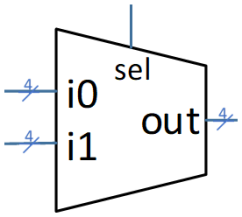


Diagrama de blocos do MUX 2x1 de 4bits

6. Implemente um testbench para validar um conjunto representativo de entradas e saídas do seu **módulo MUX 2x1 de 4bits**. Simule-o no EDA Playground e cheque se os resultados estão de acordo com o previsto. Chame o professor para Validar seu progresso até esse ponto.

7. Utilize os módulos de **somador** e **MUX 2x1** para construir um terceiro módulo que possibilite operar entre duas entradas **A** e **B** ou entre **A** e uma constante **C**. A seleção ente os dois modos de operação, deve ser realizada por uma entrada **S**.

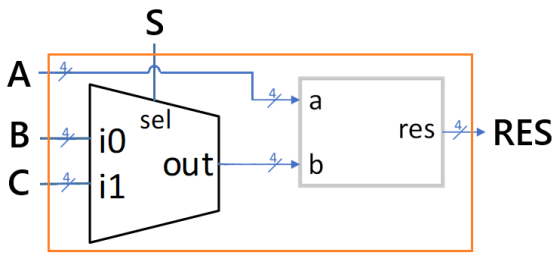


Diagrama de blocos da montagem com o MUX e o somador

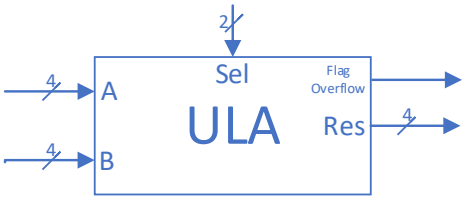
8. Implemente um testbench para validar a sua montagem do item 7. Simule-o no EDA Playground e cheque se os resultados estão de acordo com o previsto. Chame o professor para Validar seu progresso até esse ponto.

Após o professor conferir seus testes, compacte todos os arquivos em um **.zip** e submeta-o no **SIGAA**

Desafio (Valendo +0,1 na média geral)

1. Modifique seu somador para possibilitar a execução de 4 operações (ULA): Soma, Subtração, deslocamento para direita e para esquerda. Além da saída do resultado, inclua um flag de status para indicar overflow.

Sel	Res
00	A + B
01	A – B
10	A >> B
11	A << B



2. Implemente um testbench para validar os cenários mais significativos do seu novo módulo de ULA.