

ARQ1 \_ Aula\_14

Tema: Introdução à linguagem Verilog e simulação em Logisim (circuitos sequenciais)

Orientação geral:

Apresentar todas as soluções em apenas um arquivo com formato texto (.txt).  
Sugere-se usar como nome Guia\_xx.txt, onde xx indicará o guia, exemplo Guia\_01.txt.  
Todos os arquivos deverão conter identificações iniciais com o nome e matrícula, no caso de programas, usar comentários.  
As implementações e testes dos exemplos em Verilog (.v) fornecidos como pontos de partida, também fazem parte da atividade e deverão ter os códigos fontes entregues **separadamente**, com o código fonte e os módulos de testes, a fim de que possam ser compilados e verificados.  
Sugere-se usar como nomes Guia\_01yy.v, onde yy indicará a questão, exemplo Guia\_0101.v  
As saídas de resultados, opcionalmente, poderão ser copiadas ao final do código, em comentários.  
Quaisquer outras anotações, observações ou comentários poderão ser colocadas em arquivo texto (README.txt) acompanhando a entrega.

Outras formas de solução serão **opcionais**; não servirão para substituir as atividades a serem avaliadas. Caso entregues, poderão contar apenas como atividades extras.  
Os programas com funções desenvolvidas em C, Java ou Python (c, .java, py), como os modelos usados para verificação automática de testes das respostas; caso entregues, também deverão estar em arquivos **separados**, com o código fonte e os módulos de testes, a fim de que possam ser compilados e verificados.  
As execuções deverão, preferencialmente, serão testadas mediante uso de redirecionamento de entradas e saídas padrões, cujos dados/resultados deverão ser armazenados em arquivos textos.  
Os resultados poderão ser anexados ao código, ao final, como comentários  
Planilhas, caso venham a ser solicitadas, deverão ser **programadas** e/ou usar funções nativas.  
Serão suplementares e opcionais, e deverão ser entregues em formato texto, preferencialmente, com colunas separadas por tabulações ou no formato (.csv), acompanhando a solução em texto.  
Arquivos em formato (.pdf), fotos, cópias de tela ou soluções manuscritas também poderão ser aceitos como recursos suplementares para visualização, mas não servirão como substitutos e **não** terão validade para fins de avaliação.  
Os *layouts* de circuitos deverão ser entregues no formato (.circ), identificados internamente.  
Os *layouts* de diagramas deverão ser entregues no formato (.jff), identificados externamente.  
Figuras exportadas pela ferramenta serão aceitas apenas como arquivos para visualização, mas **não** terão validade para fins de avaliação. Separar versões completas (a) e simplificadas (b).

## Atividade: Circuitos sequenciais – *Flip-Flops*

Todos os circuitos deverão ser simulados no Logisim.

- 01.) Projetar e descrever em Logisim e Verilog um módulo para implementar um registrador de deslocamento para a esquerda, com 6 bits (estágios), com carga de 1 bit (load=LD) no *preset* do primeiro estágio.  
DICA: Ver modelo anexo.
- 02.) Projetar e descrever em Logisim e Verilog um módulo para implementar um registrador de deslocamento para a esquerda, com 6 bits (estágios), com carga inicial (load=LD) em todos *preset* dos estágios.
- 03.) Projetar e descrever em Logisim e Verilog um módulo para implementar um registrador de deslocamento circular ("*ring*") para a direita, com 6 bits (estágios), com carga unitária no primeiro estágio.
- 04.) Projetar e descrever em Logisim e Verilog um módulo para implementar um registrador de deslocamento circular, em anel torcido ("*twisted ring*"), para a esquerda, com 6 bits (estágios), com carga unitária no primeiro estágio.  
DICA: Ver modelo anexo.
- 05.) Projetar e descrever em Logisim e Verilog um módulo para implementar um conversor paralelo-série para 6 bits.  
DICA: Ver modelo anexo.

## Extras

- 06.) Projetar e descrever em Logisim e Verilog um módulo para implementar um registrador de deslocamento circular ("*ring*") para a esquerda, com 5 bits (estágios), com carga inicial em todos os estágios.
- 07.) Projetar e descrever em Logisim e Verilog um módulo para implementar um registrador de deslocamento circular, em anel torcido ("*twisted ring*"), para a direita, com 5 bits (estágios), com carga inicial ("*load/preset*") em todos os estágios.

```

module dff ( output q, output qnot,
             input  d, input clk,
             input  preset, input clear );
reg q, qnot;

always @( posedge clk )
begin
  if ( clear )    begin q <= 0; qnot <=  1; end
  else
    if ( preset ) begin q <= 1; qnot <=  0; end
    else
      begin q <= d; qnot <= ~d; end
end

endmodule // dff

module jkff ( output q, output qnot,
             input  j, input k,
             input clk, input preset, input clear );

reg  q, qnot;

always @( posedge clk or
         posedge preset or
         posedge clear )
begin
  if ( clear )    begin q <= 0; qnot <= 1; end
  else
    if ( preset ) begin q <= 1; qnot <= 0; end
    else
      if ( j & ~k ) begin q <= 1; qnot <= 0; end
      else
        if ( ~j & k ) begin q <= 0; qnot <= 1; end
        else
          if ( j & k )
            begin q <= ~q; qnot <= ~qnot; end
end

endmodule // jkff

```

```

module tff ( output q, output qnot,
             input  t, input  clk,
             input  preset, input clear );

reg q, qnot;

always @( posedge clk or ~preset or ~clear)
begin
  if ( ~clear )
    begin  q <= 0;          qnot <= 1;  end
  else
    if ( ~preset )
      begin  q <= 1;          qnot <= 0;  end
    else
      begin
        if ( t ) begin q <= ~q; qnot <= ~qnot; end
        end
      end

endmodule // tff

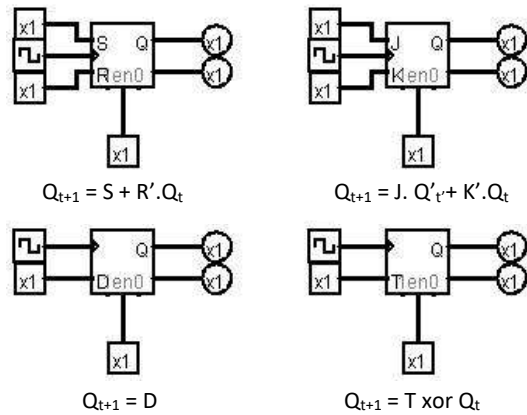
module srff ( output q, output qnot,
             input  s, input r, input clk,
             input preset, input clear );
reg q, qnot;

always @( posedge clk )
begin
  if ( clear )    begin q <= 0; qnot <= 1; end
  else
    if ( preset )  begin q <= 1; qnot <= 0; end
    else
      if ( s & ~r ) begin q <= 1; qnot <= 0; end
      else
        if ( ~s & r ) begin q <= 0; qnot <= 1; end
        else
          if ( s & r )
            begin  q <= 0; qnot <= 0;  end // arbitrary
end

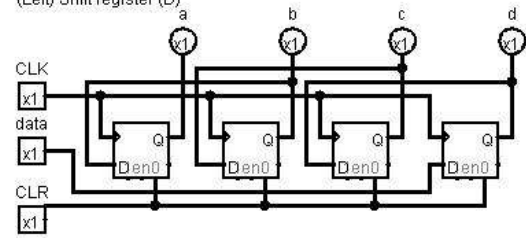
endmodule // srff

```

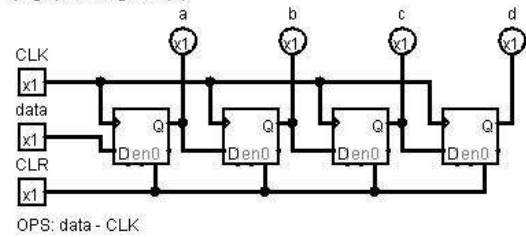
## Flip-flops



(Left) Shift register (D)

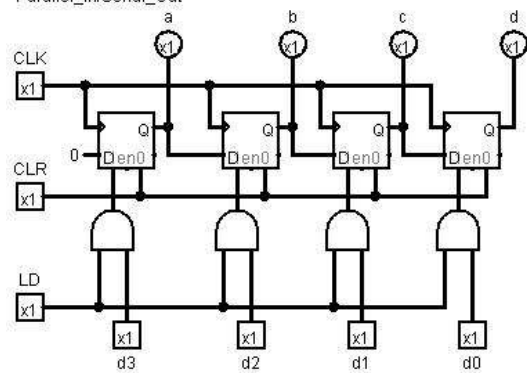


(Right) Shift register (D)



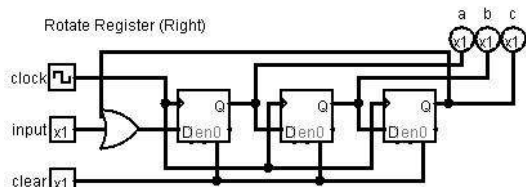
OPS: data - CLK

## Parallel\_In/Serial\_Out

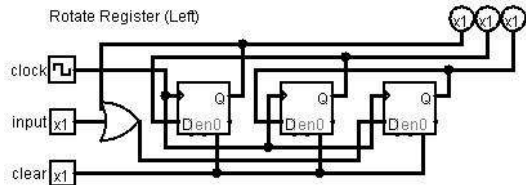


OPS: data - CLR - LD - CLK

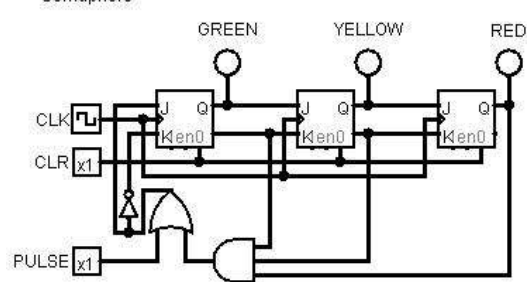
Rotate Register (Right)



Rotate Register (Left)

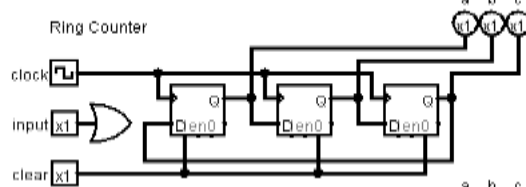


## Semaphore



Operation: CLR - PULSE (UP) - CLK - PULSE (DOWN) - CLK ...

Ring Counter



Twisted Ring Counter

