Pontifícia Universidade Católica de Minas Gerais
Instituto de Ciências Exatas e Informática – ICEI
Arquitetura de Computadores I

ARQ1 _ Aula_14
Tema: Introdução à linguagem Verilog e simulação em Logisim

Orientação geral:
Os arquivos para simulação em Logisim (.circ) poderão ser entregues como complementação, deverão ser identificados internamente e entregues, acompanhados (ou não) de figuras equivalentes exportadas pela ferramenta.
Programas em Verilog deverão ser entregues em formato (.v) com previsão de testes.

Atividade: Circuitos sequenciais – Flip-Flops
Todos os circuitos deverão ser simulados no Logisim.

01.) Projetar e descrever em Logisim e Verilog um módulo,
com portas e flip-flops tipo JK apenas,
para implementar um contador assíncrono decrescente
com 5 bits de comprimento.
DICA: Ver modelo anexo.

02.) Projetar e descrever em Logisim e Verilog um módulo
com portas e flip-flops tipo JK apenas,
para implementar um contador assíncrono crescente
com 5 bits de comprimento.

03.) Projetar e descrever em Logisim e Verilog um módulo,
com portas lógicas e flip-flops tipo JK apenas,
para implementar um contador assíncrono decádico crescente
com 4 bits de comprimento.
DICA: Ver modelo anexo.

04.) Projetar e descrever em Logisim e Verilog um módulo
com portas e flip-flops tipo JK apenas,
para implementar um contador assíncrono decádico decrescente
com 4 bits de comprimento.

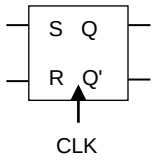
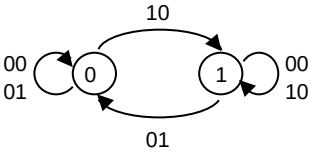
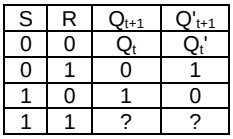
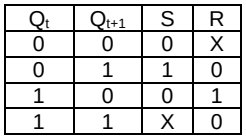
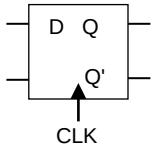
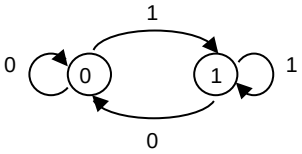
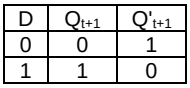
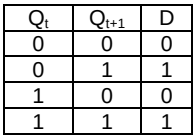
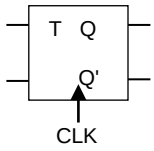
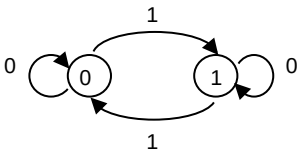
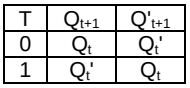
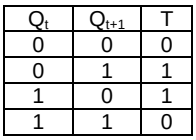
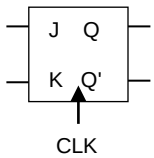
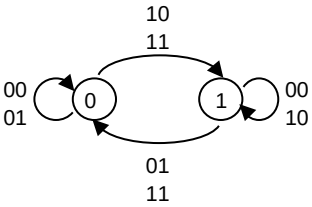
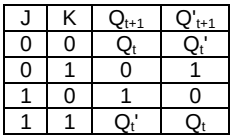
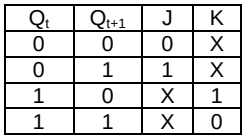
05.) Projetar e descrever em Logisim e Verilog um módulo,
com portas e flip-flops tipo T apenas,
para implementar um contador síncrono módulo 8.
DICA: Ver modelo anexo.

Extras

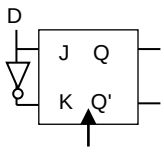
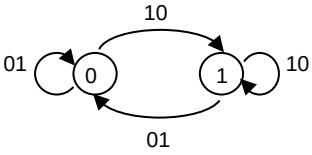
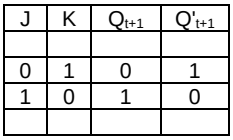
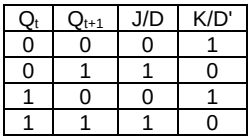
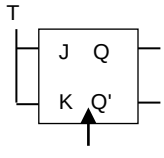
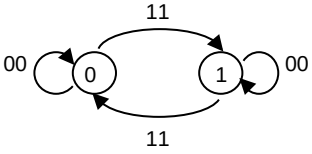
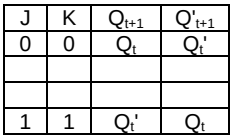
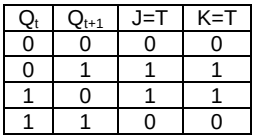
06.) Projetar e descrever em Logisim e Verilog um módulo,
com portas e flip-flops tipo JK apenas,
para implementar um contador em anel com 6 bits de comprimento.
DICA: Ver modelo anexo.

07.) Projetar e descrever em Logisim e Verilog um módulo
com portas e flip-flops tipo JK apenas,
para implementar um contador em anel torcido com 6 bits de comprimento.
DICA: Ver modelo anexo.

Flip-flops

Flip-flop	Estados	Característica	Transição	Equação
				$Q_{t+1}=S+R'.Q_t$
				$Q_{t+1} = D$
				$Q_{t+1} = T \oplus Q_t$
				$Q_{t+1}=J.Q_t'+K'.Q_t$

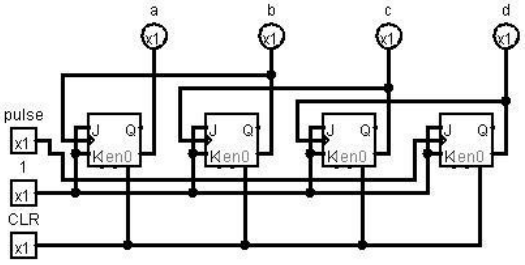
Configurações especiais

Flip-flop	Estados	Característica	Transição	Equação
				$Q_{t+1}=1.Q_t'+0'.Q_t$ $Q_{t+1}=1$ $Q_{t+1}=0.Q_t'+1'.Q_t$ $Q_{t+1}=0$
				$Q_{t+1}=0.Q_t'+0'.Q_t$ $Q_{t+1}=0'.Q_t = Q_t$ $Q_{t+1}=1.Q_t'+1'.Q_t$ $Q_{t+1}=1 .Q_t' = Q_t'$

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

always @( posedge clk )
begin
   q <= d;          qnot <= ~d;
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 preset or 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
```

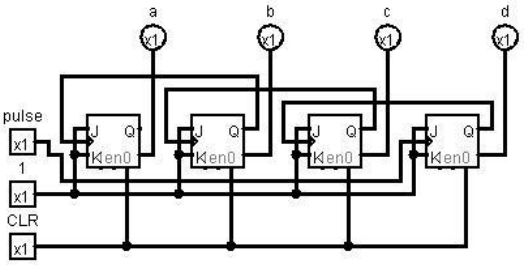
```
module tff ( output q, output qnot,
                input    t, input    clk,
                input    preset, input clear );

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 );
reg q, qnot;

always @( posedge clk )
begin
   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; // arbitrary end
end

endmodule // srff
```

(Down) Asynchronous counter



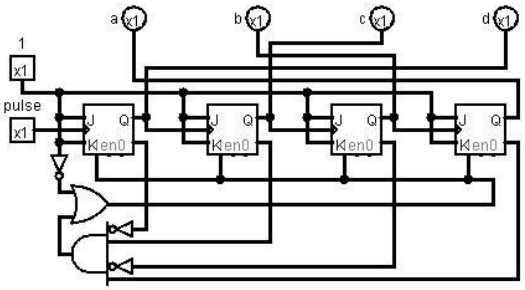
OPS: CLR - 1 - pulse

(Up) Asynchronous counter

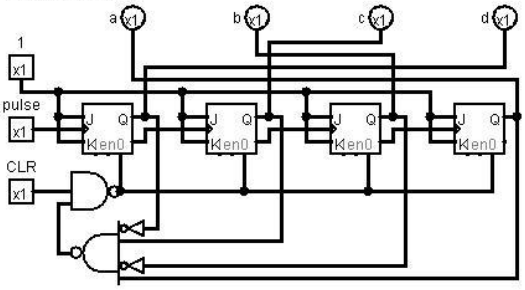


OPS: CLR - 1 - pulse

(Down) Decade counter

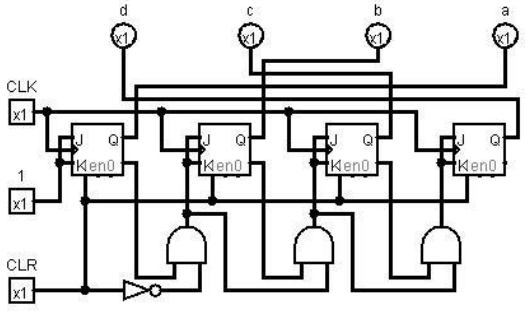


(Up) Decade counter



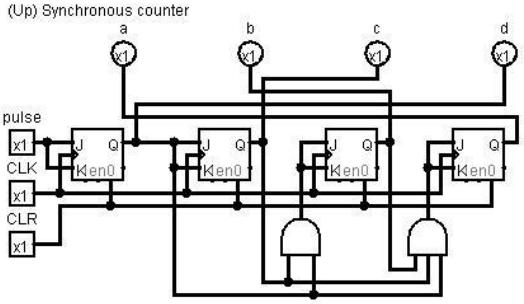
OPS: CLR - 1 - pulse

(Down) Synchronous counter

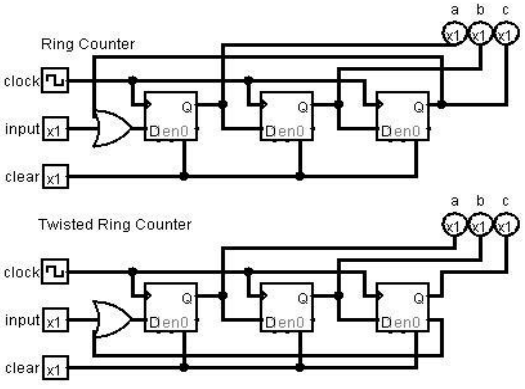


OPS: CLR - 1 - CLK

(Up) Synchronous counter



OPS: CLR - pulse - CLK



Counter base 5