

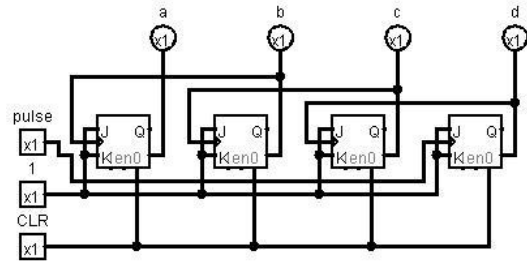
Tema: Introdução à linguagem Verilog  
Atividade: Circuitos sequenciais – Flip-Flops  
Todos os circuitos deverão ser simulados no Logisim.

- 01.) Projetar e descrever em Verilog um módulo,  
com portas e flip-flops tipo JK apenas,  
para implementar um contador assíncrono decrescente  
com 6 bits de comprimento.  
DICA: Ver modelo anexo.
- 02.) Projetar e descrever em Verilog um módulo  
com portas e flip-flops tipo JK apenas,  
para implementar um contador assíncrono crescente  
com 6 bits de comprimento.
- 03.) Projetar e descrever em Verilog um módulo,  
com portas lógicas e flip-flops tipo JK apenas,  
para implementar um contador decádico crescente  
com 6 bits de comprimento.  
DICA: Ver modelo anexo.
- 04.) Projetar e descrever em Verilog um módulo  
com portas e flip-flops tipo JK apenas,  
para implementar um contador decádico decrescente  
com 6 bits de comprimento.
- 05.) Projetar e descrever em Verilog um módulo,  
com portas e flip-flops tipo T apenas,  
para implementar um contador módulo 10.  
DICA: Ver modelo anexo.

#### Extras

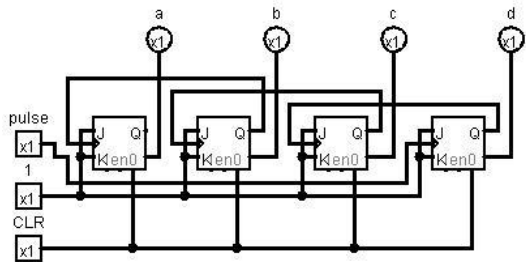
- 06.) Projetar e descrever em Verilog um módulo,  
com portas e flip-flops tipo JK apenas,  
para implementar um contador em anel  
com 6 bits de comprimento.  
DICA: Ver modelo anexo.
- 07.) Projetar e descrever em Verilog um módulo  
com portas e flip-flops tipo JK apenas,  
para implementar um contador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

(Down) Asynchronous counter



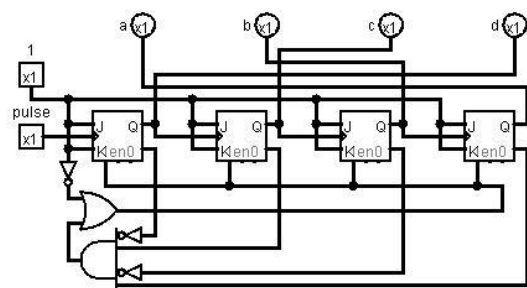
OPS: CLR - 1 - pulse

(Up) Asynchronous counter

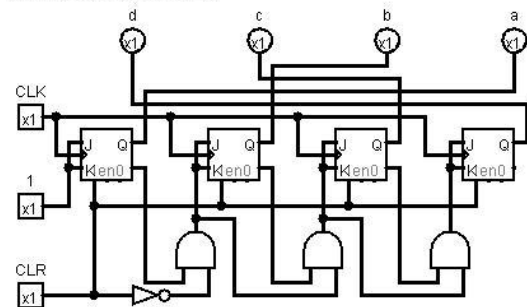


OPS: CLR - 1 - pulse

(Down) Decade counter

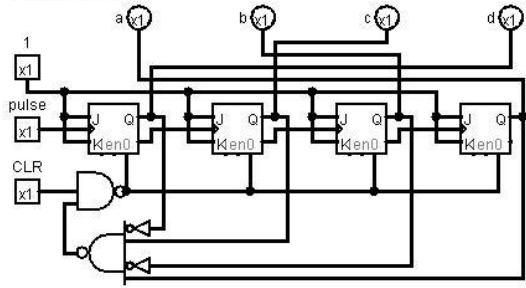


(Down) Synchronous counter



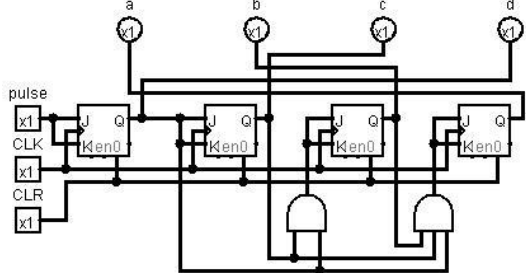
OPS: CLR - 1 - CLK

(Up) Decade counter

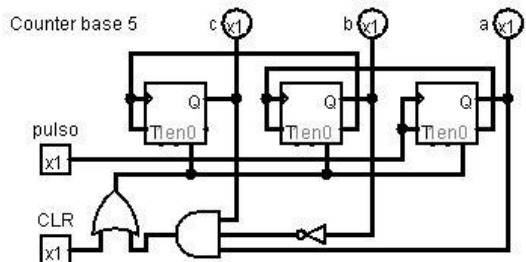
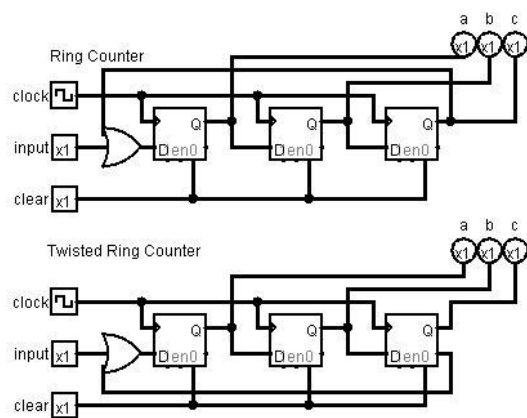


OPS: CLR - 1 - pulse

(Up) Synchronous counter



OPS: CLR - pulse - CLK



```

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 tff ( output q, output qnot,
             input  t, 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
        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

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

always @( posedge clk )
begin
    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

```