

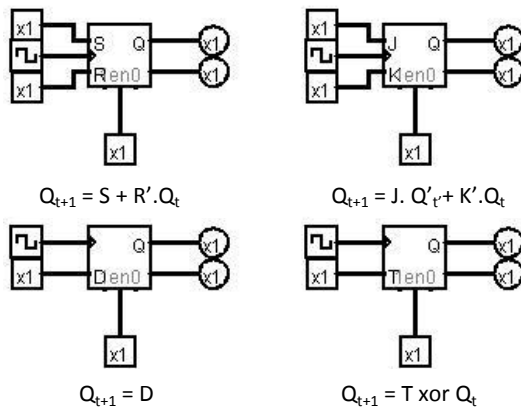
Tema: Introdução à linguagem Verilog
Atividade: Circuitos sequenciais – Flip-Flops
Todos os circuitos deverão ser simulados no Logisim.

- 01.) Projetar e descrever em Verilog um módulo para implementar um registrador de deslocamento para a esquerda, com 4 bits (estágios), com carga unitária no primeiro estágio.
DICA: Ver modelo anexo.
- 02.) Projetar e descrever em Verilog um módulo para implementar um registrador de deslocamento para a esquerda, com 4 bits (estágios), com carga inicial em todos os estágios.
- 03.) Projetar e descrever em Verilog um módulo para implementar um registrador de deslocamento circular para a direita, com 4 bits (estágios), com carga unitária no primeiro estágio.
- 04) Projetar e descrever em Verilog um módulo para implementar um registrador de deslocamento circular, em anel torcido, para a esquerda, com 4 bits (estágios), com carga unitária no primeiro estágio.
DICA: Ver modelo anexo.
- 05.) Projetar e descrever em Verilog um módulo para implementar um conversor paralelo-série para 04 bits.
DICA: Ver modelo anexo.

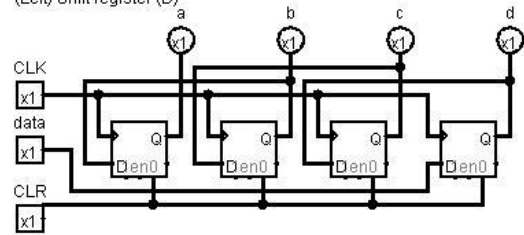
Extras

- 06.) Projetar e descrever em Verilog um módulo para implementar um registrador de deslocamento circular para a direita, com 4 bits (estágios), com carga inicial em todos os estágios.
- 07.) Projetar e descrever em Verilog um módulo para implementar um registrador de deslocamento circular, em anel torcido, para a direita, com 4 bits (estágios), com carga inicial em todos os estágios.

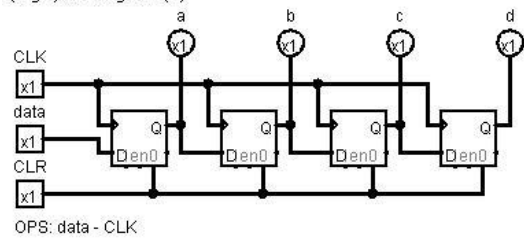
Flip-flops



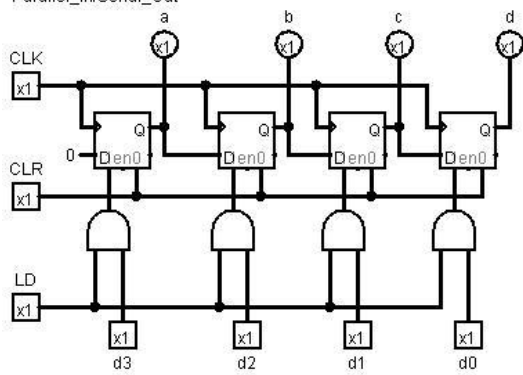
(Left) Shift register (D)



(Right) Shift register (D)

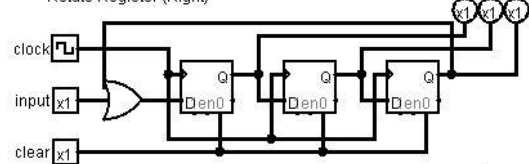


Parallel_In/Serial_Out

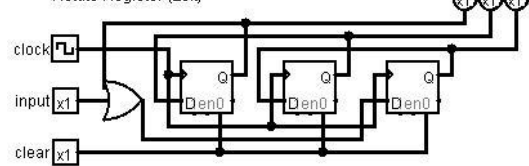


OPS: data - CLR - LD - CLK

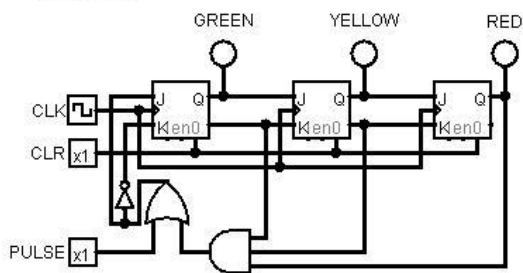
Rotate Register (Right)



Rotate Register (Left)

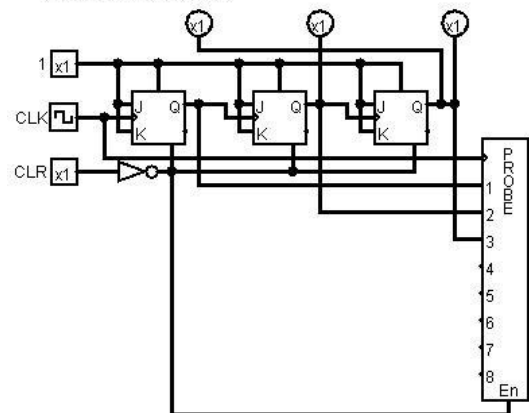


Semaphore



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

Frequency Division (by 2)



```

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

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

```