

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

ARQ1 \_ Aula\_13

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

Orientação geral:

Atividades previstas como parte da avaliação

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**,  
a fim de que possam ser compilados e testados.

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,  
como comentários.

Atividades extras e opcionais

Outras formas de solução serão **opcionais**; não servirão para substituir as atividades  
a serem avaliadas. Caso entregues, poderão contar apenas como atividades extras.

Os *layouts* de circuitos deverão ser entregues no formato (.circ), identificados internamente.  
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).

Arquivos em formato (.pdf), fotos, cópias de tela ou soluções manuscritas também serão aceitos  
como recursos suplementares para visualização, e **não** terão validade para fins de avaliação.

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

### Análise e síntese de circuitos sequenciais

As técnicas para análise de circuitos sequenciais que implementam uma certa máquina de estados finitos, em geral, dividem-se em duas etapas:

1. determinar as funções que determinam o próximo estado e as saídas
  - 1.1 especificar as equações que representem a lógica do circuito e as saídas de cada **flip-flop** (estado corrente);
  - 1.2 especificar as equações que determinem as transições entre dois pulsos de **clock**;
  - 1.3 construir a *tabela de transições* para cada uma das combinações das entradas, indicando quais os próximos estados;
  - 1.4 identificar todas as combinações que representem um mesmo estado e reescrevê-las em uma *tabela de estados*;
2. construir as tabelas de estados/saídas que especifiquem o comportamento do circuito para todas as combinações das entradas e do estado corrente:
  - 2.1 verificar as funções das saídas em relação às entradas e aos estados correntes;
  - 2.2 após avaliar todas as combinações de entradas e estados, combinar a tabela de estados com essas informações e criar a tabela de estados/saídas, relacionando cada saída ao próximo estado.

Exemplo 1:

Considerar o circuito abaixo com um **flip-flop** tipo D.

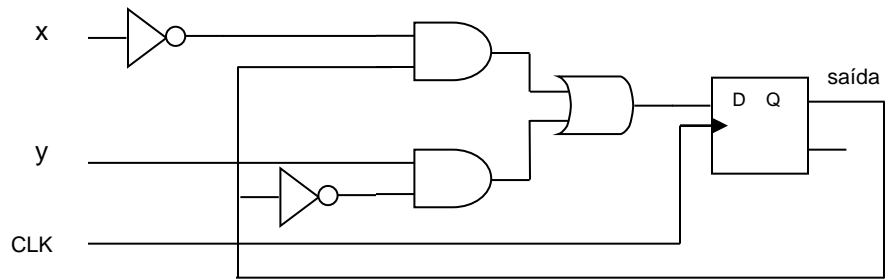


Tabela de transições

$Q_t \backslash xy$	00	01	10	11
0	0	1	0	1
1	1	1	0	0

$Q_{t+1}$

Equações de transições

$$D = x' \cdot Q + y \cdot Q'$$

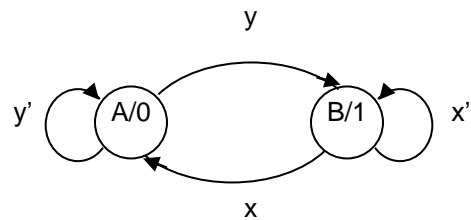
$$Q_{t+1} = x' \cdot Q_t + y \cdot Q'_t$$

Tabela de estados/saídas

$Q_t \backslash xy$	00	01	10	11
A	A,0	B,1	A,0	B,1
B	B,1	B,1	A,0	A,0

$Q_{t+1},$  saída

Diagrama de estados



Considerar o circuito abaixo com dois **flip-flops** tipo JK.

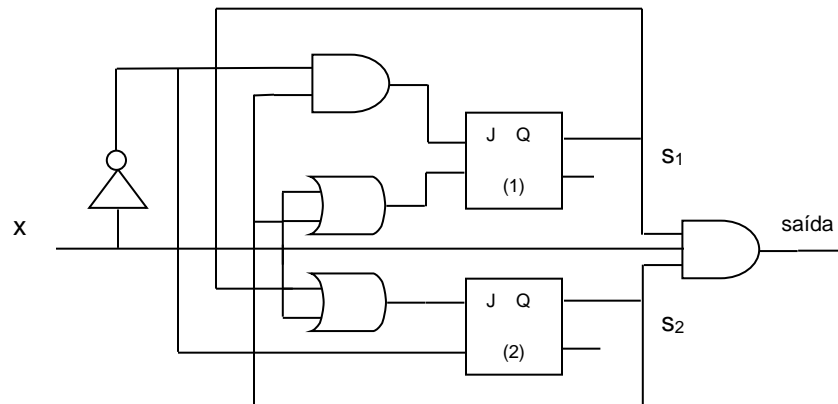


Tabela de transições

S <sub>1</sub> (t)	S <sub>2</sub> (t)	X	S <sub>1</sub> (t+1)	S <sub>2</sub> (t+1)	saída
0	0	0	0	0	0
0	0	1	0	1	0
0	1	0	1	0	0
0	1	1	0	1	0
1	0	0	1	1	0
1	0	1	0	1	0
1	1	0	0	0	0
1	1	1	0	1	1

Equações de transições

$$\text{saída} = S_1 \cdot S_2 \cdot X$$

$$J_1 = S_2 \cdot X' \quad \text{e} \quad K_1 = S_2 + X$$

$$J_2 = S_1 + X \quad \text{e} \quad K_2 = X'$$

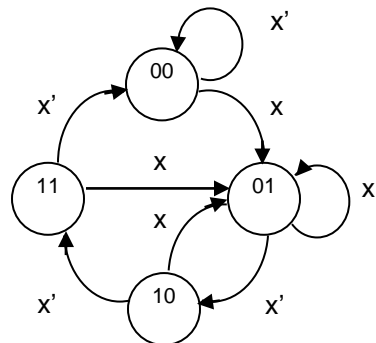
$$\begin{aligned} Q_{t+1} &= J_1 Q_t' + K_1' Q_t \\ S_1 &= S_2 \cdot X' \cdot S_1' + (S_2 + X) \cdot S_1 \\ &= S_2 \cdot X' \cdot S_1' + S_2' \cdot X' \cdot S_1 \\ &= X' \cdot (S_2 \cdot S_1' + S_2' \cdot S_1) \\ &= X' \cdot (S_1 \text{ xor } S_2) \end{aligned}$$

$$\begin{aligned} Q_{t+1} &= J_2 Q_t' + K_2' Q_t \\ S_2 &= (X + S_1) \cdot S_2' + (X')' \cdot S_2 \\ &= (X \cdot S_2') + (S_1 \cdot S_2') + (X \cdot S_2) \\ &= X \cdot (S_2' + S_2) + (S_1 \cdot S_2') \\ &= X + (S_1 \cdot S_2') \end{aligned}$$

Tabela de estados/saídas

S1	S2	x=0		x=1		saída
0	0	0	0	0	1	0
0	1	1	0	0	1	0
1	0	1	1	0	1	0
1	1	0	0	0	1	0/1

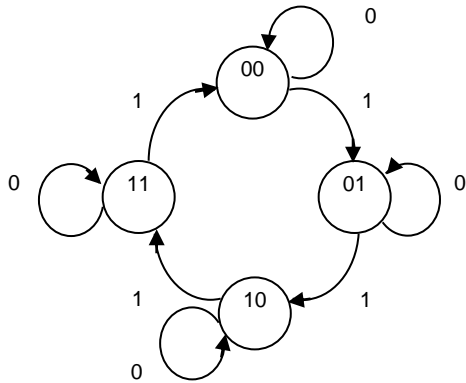
Diagrama de estados



Exemplo 2:

Projetar um contador crescente módulo 4 (0-1-2-3-0) com **flip-flops** tipo D.

### Tabela de transições



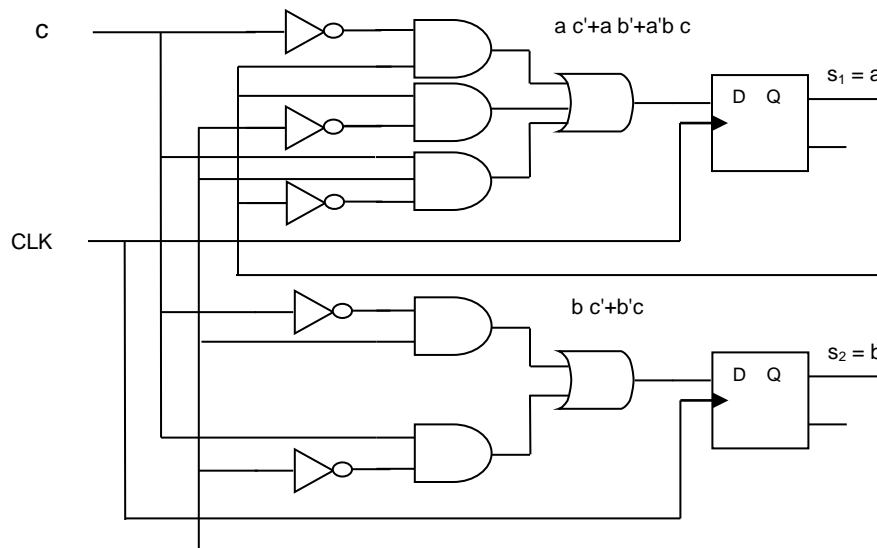
### Diagrama de estados

	S <sub>1</sub> (t)	S <sub>2</sub> (t)	evento	S <sub>1</sub> (t+1)	S <sub>2</sub> (t+1)
	a	b	c	a	b
0	0	0	0	0	0
1	0	0	1	0	1
2	0	1	0	0	1
3	0	1	1	1	0
4	1	0	0	1	0
5	1	0	1	1	1
6	1	1	0	1	1
7	1	1	1	0	0

### Equações de transições

sinais	SoP	mintermos	simplificação
s <sub>1</sub>	3,4,5,6	$a'bc+ab'c+ab'c+abc'$	$ac'+ab'+ab'c$
s <sub>2</sub>	1,2,5,6	$a'b'c+a'bc'+ab'c+abc'$	$bc'+b'c$

## Circuito



Exemplo 3:

Projetar um contador decrescente módulo 4 (0-3-2-1-0) com *flip-flops* tipo D.

Tabela de transições

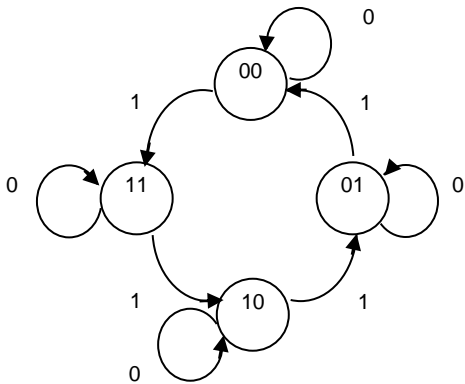


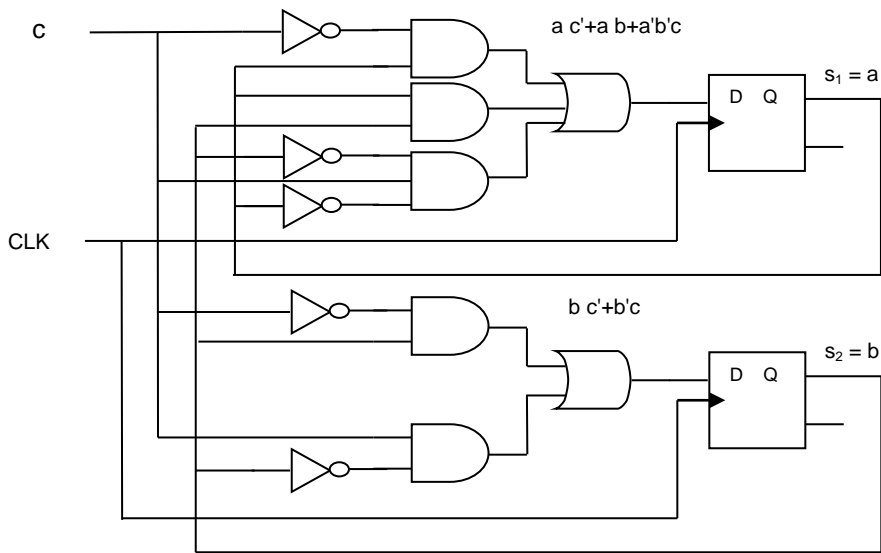
Diagrama de estados

	S1 (t)	S2 (t)	evento c	S1 (t+1)	S2 (t+1)
	a	b		a	b
0	0	0	0	0	0
1	0	0	1	1	1
2	0	1	0	0	1
3	0	1	1	0	0
4	1	0	0	1	0
5	1	0	1	0	1
6	1	1	0	1	1
7	1	1	1	1	0

Equações de transições

sinais	SoP	mintermos	simplificação
S1	1,4,6,7	$a'b'c+ab'c'+abc'+abc$	$ac'+ab+a'b'c$
S2	1,2,5,6	$a'b'c+a'bc'+ab'c+abc'$	$bc'+b'c$

Circuito



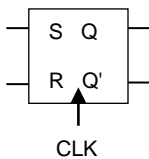
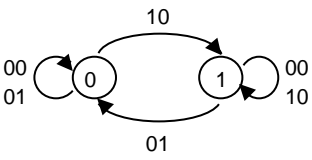
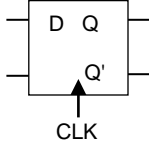
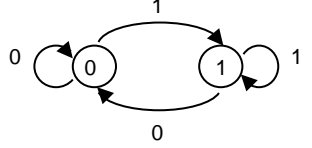
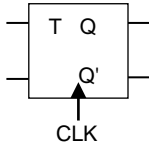
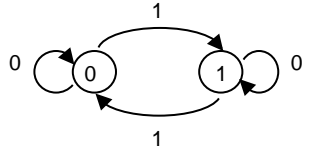
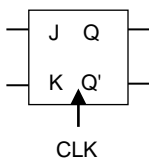
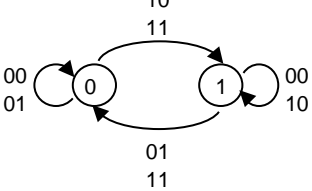
## Exercícios

- 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 9.  
DICA: Ver modelo anexo.

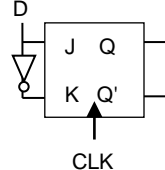
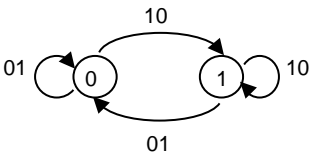
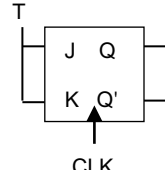
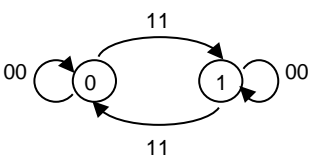
## Extras

- 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 5 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 5 bits de comprimento.  
DICA: Ver modelo anexo.

## Flip-flops

Flip-flop	Estados	Característica	Transição	Equação																																								
		<table><tr><th>S</th><th>R</th><th>Q<sub>t+1</sub></th><th>Q'<sub>t+1</sub></th></tr><tr><td>0</td><td>0</td><td>Q<sub>t</sub></td><td>Q'<sub>t</sub></td></tr><tr><td>0</td><td>1</td><td>0</td><td>1</td></tr><tr><td>1</td><td>0</td><td>1</td><td>0</td></tr><tr><td>1</td><td>1</td><td>?</td><td>?</td></tr></table>	S	R	Q <sub>t+1</sub>	Q' <sub>t+1</sub>	0	0	Q <sub>t</sub>	Q' <sub>t</sub>	0	1	0	1	1	0	1	0	1	1	?	?	<table><tr><th>Q<sub>t</sub></th><th>Q<sub>t+1</sub></th><th>S</th><th>R</th></tr><tr><td>0</td><td>0</td><td>0</td><td>X</td></tr><tr><td>0</td><td>1</td><td>1</td><td>0</td></tr><tr><td>1</td><td>0</td><td>0</td><td>1</td></tr><tr><td>1</td><td>1</td><td>X</td><td>0</td></tr></table>	Q <sub>t</sub>	Q <sub>t+1</sub>	S	R	0	0	0	X	0	1	1	0	1	0	0	1	1	1	X	0	$Q_{t+1}=S+R'.Q_t$
S	R	Q <sub>t+1</sub>	Q' <sub>t+1</sub>																																									
0	0	Q <sub>t</sub>	Q' <sub>t</sub>																																									
0	1	0	1																																									
1	0	1	0																																									
1	1	?	?																																									
Q <sub>t</sub>	Q <sub>t+1</sub>	S	R																																									
0	0	0	X																																									
0	1	1	0																																									
1	0	0	1																																									
1	1	X	0																																									
		<table><tr><th>D</th><th>Q<sub>t+1</sub></th><th>Q'<sub>t+1</sub></th></tr><tr><td>0</td><td>0</td><td>1</td></tr><tr><td>1</td><td>1</td><td>0</td></tr></table>	D	Q <sub>t+1</sub>	Q' <sub>t+1</sub>	0	0	1	1	1	0	<table><tr><th>Q<sub>t</sub></th><th>Q<sub>t+1</sub></th><th>D</th></tr><tr><td>0</td><td>0</td><td>0</td></tr><tr><td>0</td><td>1</td><td>1</td></tr><tr><td>1</td><td>0</td><td>0</td></tr><tr><td>1</td><td>1</td><td>1</td></tr></table>	Q <sub>t</sub>	Q <sub>t+1</sub>	D	0	0	0	0	1	1	1	0	0	1	1	1	$Q_{t+1} = D$																
D	Q <sub>t+1</sub>	Q' <sub>t+1</sub>																																										
0	0	1																																										
1	1	0																																										
Q <sub>t</sub>	Q <sub>t+1</sub>	D																																										
0	0	0																																										
0	1	1																																										
1	0	0																																										
1	1	1																																										
		<table><tr><th>T</th><th>Q<sub>t+1</sub></th><th>Q'<sub>t+1</sub></th></tr><tr><td>0</td><td>Q<sub>t</sub></td><td>Q'<sub>t</sub></td></tr><tr><td>1</td><td>Q'<sub>t</sub></td><td>Q<sub>t</sub></td></tr></table>	T	Q <sub>t+1</sub>	Q' <sub>t+1</sub>	0	Q <sub>t</sub>	Q' <sub>t</sub>	1	Q' <sub>t</sub>	Q <sub>t</sub>	<table><tr><th>Q<sub>t</sub></th><th>Q<sub>t+1</sub></th><th>T</th></tr><tr><td>0</td><td>0</td><td>0</td></tr><tr><td>0</td><td>1</td><td>1</td></tr><tr><td>1</td><td>0</td><td>1</td></tr><tr><td>1</td><td>1</td><td>0</td></tr></table>	Q <sub>t</sub>	Q <sub>t+1</sub>	T	0	0	0	0	1	1	1	0	1	1	1	0	$Q_{t+1} = T \oplus Q_t$																
T	Q <sub>t+1</sub>	Q' <sub>t+1</sub>																																										
0	Q <sub>t</sub>	Q' <sub>t</sub>																																										
1	Q' <sub>t</sub>	Q <sub>t</sub>																																										
Q <sub>t</sub>	Q <sub>t+1</sub>	T																																										
0	0	0																																										
0	1	1																																										
1	0	1																																										
1	1	0																																										
		<table><tr><th>J</th><th>K</th><th>Q<sub>t+1</sub></th><th>Q'<sub>t+1</sub></th></tr><tr><td>0</td><td>0</td><td>Q<sub>t</sub></td><td>Q'<sub>t</sub></td></tr><tr><td>0</td><td>1</td><td>0</td><td>1</td></tr><tr><td>1</td><td>0</td><td>1</td><td>0</td></tr><tr><td>1</td><td>1</td><td>Q'<sub>t</sub></td><td>Q<sub>t</sub></td></tr></table>	J	K	Q <sub>t+1</sub>	Q' <sub>t+1</sub>	0	0	Q <sub>t</sub>	Q' <sub>t</sub>	0	1	0	1	1	0	1	0	1	1	Q' <sub>t</sub>	Q <sub>t</sub>	<table><tr><th>Q<sub>t</sub></th><th>Q<sub>t+1</sub></th><th>J</th><th>K</th></tr><tr><td>0</td><td>0</td><td>0</td><td>X</td></tr><tr><td>0</td><td>1</td><td>1</td><td>X</td></tr><tr><td>1</td><td>0</td><td>X</td><td>1</td></tr><tr><td>1</td><td>1</td><td>X</td><td>0</td></tr></table>	Q <sub>t</sub>	Q <sub>t+1</sub>	J	K	0	0	0	X	0	1	1	X	1	0	X	1	1	1	X	0	$Q_{t+1}=J.Q_t'+K'.Q_t$
J	K	Q <sub>t+1</sub>	Q' <sub>t+1</sub>																																									
0	0	Q <sub>t</sub>	Q' <sub>t</sub>																																									
0	1	0	1																																									
1	0	1	0																																									
1	1	Q' <sub>t</sub>	Q <sub>t</sub>																																									
Q <sub>t</sub>	Q <sub>t+1</sub>	J	K																																									
0	0	0	X																																									
0	1	1	X																																									
1	0	X	1																																									
1	1	X	0																																									

## Configurações especiais

Flip-flop	Estados	Característica	Transição	Equação																																								
		<table><tr><th>J</th><th>K</th><th>Q<sub>t+1</sub></th><th>Q'<sub>t+1</sub></th></tr><tr><td></td><td></td><td></td><td></td></tr><tr><td>0</td><td>1</td><td>0</td><td>1</td></tr><tr><td>1</td><td>0</td><td>1</td><td>0</td></tr><tr><td></td><td></td><td></td><td></td></tr></table>	J	K	Q <sub>t+1</sub>	Q' <sub>t+1</sub>					0	1	0	1	1	0	1	0					<table><tr><th>Q<sub>t</sub></th><th>Q<sub>t+1</sub></th><th>J/D</th><th>K/D'</th></tr><tr><td>0</td><td>0</td><td>0</td><td>1</td></tr><tr><td>0</td><td>1</td><td>1</td><td>0</td></tr><tr><td>1</td><td>0</td><td>0</td><td>1</td></tr><tr><td>1</td><td>1</td><td>1</td><td>0</td></tr></table>	Q <sub>t</sub>	Q <sub>t+1</sub>	J/D	K/D'	0	0	0	1	0	1	1	0	1	0	0	1	1	1	1	0	$Q_{t+1}=1.Q_t'+0'.Q_t$ $Q_{t+1}=1$ $Q_{t+1}=0.Q_t'+1'.Q_t$ $Q_{t+1}=0$
J	K	Q <sub>t+1</sub>	Q' <sub>t+1</sub>																																									
0	1	0	1																																									
1	0	1	0																																									
Q <sub>t</sub>	Q <sub>t+1</sub>	J/D	K/D'																																									
0	0	0	1																																									
0	1	1	0																																									
1	0	0	1																																									
1	1	1	0																																									
		<table><tr><th>J</th><th>K</th><th>Q<sub>t+1</sub></th><th>Q'<sub>t+1</sub></th></tr><tr><td>0</td><td>0</td><td>Q<sub>t</sub></td><td>Q'<sub>t</sub></td></tr><tr><td></td><td></td><td></td><td></td></tr><tr><td></td><td></td><td></td><td></td></tr><tr><td>1</td><td>1</td><td>Q<sub>t</sub>'</td><td>Q<sub>t</sub></td></tr></table>	J	K	Q <sub>t+1</sub>	Q' <sub>t+1</sub>	0	0	Q <sub>t</sub>	Q' <sub>t</sub>									1	1	Q <sub>t</sub> '	Q <sub>t</sub>	<table><tr><th>Q<sub>t</sub></th><th>Q<sub>t+1</sub></th><th>J=T</th><th>K=T</th></tr><tr><td>0</td><td>0</td><td>0</td><td>0</td></tr><tr><td>0</td><td>1</td><td>1</td><td>1</td></tr><tr><td>1</td><td>0</td><td>1</td><td>1</td></tr><tr><td>1</td><td>1</td><td>0</td><td>0</td></tr></table>	Q <sub>t</sub>	Q <sub>t+1</sub>	J=T	K=T	0	0	0	0	0	1	1	1	1	0	1	1	1	1	0	0	$Q_{t+1}=0.Q_t'+0'.Q_t$ $Q_{t+1}=0'.Q_t = Q_t$ $Q_{t+1}=1.Q_t'+1'.Q_t$ $Q_{t+1}=1 .Q_t' = Q_t'$
J	K	Q <sub>t+1</sub>	Q' <sub>t+1</sub>																																									
0	0	Q <sub>t</sub>	Q' <sub>t</sub>																																									
1	1	Q <sub>t</sub> '	Q <sub>t</sub>																																									
Q <sub>t</sub>	Q <sub>t+1</sub>	J=T	K=T																																									
0	0	0	0																																									
0	1	1	1																																									
1	0	1	1																																									
1	1	0	0																																									



```

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

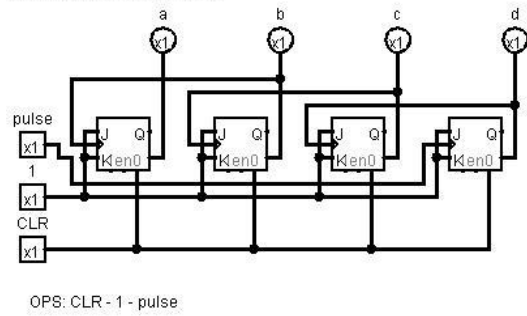
module srff ( output q, output qnot,
             input  s, input  r, input clk,
             input preset, input clear );
reg q, qnot;

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

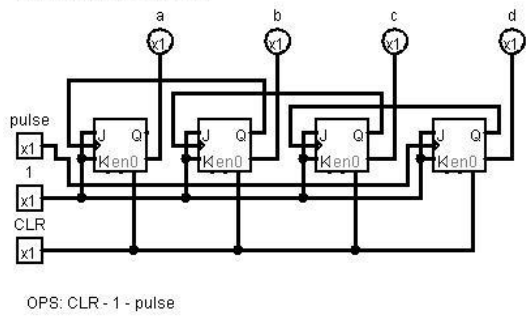
endmodule // srff

```

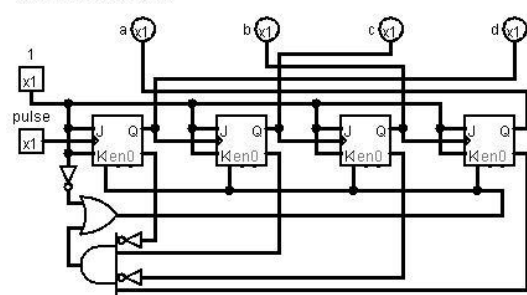
(Down) Asynchronous counter



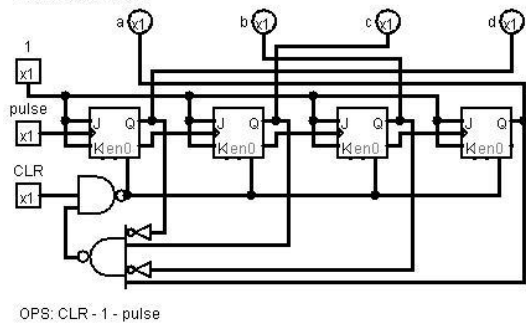
(Up) Asynchronous counter



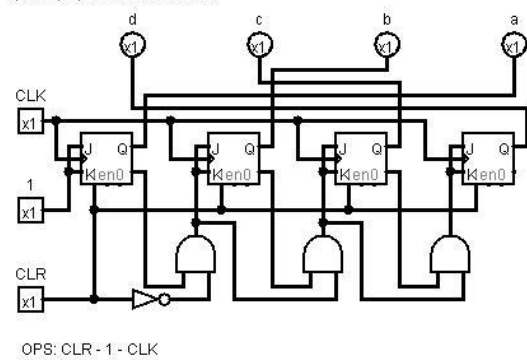
(Down) Decade counter



(Up) Decade counter



(Down) Synchronous counter



(Up) Synchronous counter

