

Implementação de Latches e FlipFlops

Professores: Kennedy/Emanoel

Email: [\[kenneurison/emanoel.chaves\]@dca.ufrn.br](mailto:[kenneurison/emanoel.chaves]@dca.ufrn.br)

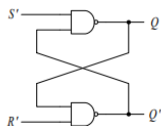
Departamento de Engenharia de Computação e Automação
Universidade Federal do Rio Grande do Norte
DCA0202 - Circuitos Digitais

Latches e FlipFlops

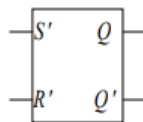
Latches e FlipFlops são dispositivos básicos de armazenamento de informação. Armazenando um ou mais bits de informação. A principal diferença entre eles consiste na maneira que suas saídas são atualizadas.

- O Latch tem suas saídas constantemente modificadas quando estiver habilitado.
- O FlipFlop precisa de um sinal de Clock.

Tipos de Latches (L-SR)



(a) LSR

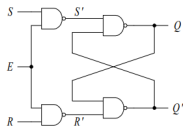


(b) LSR-Diagram

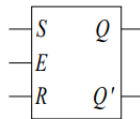
S	R	Q	Q_{next}	Q_{next}'
0	0	\times	1	1
0	1	\times	1	0
1	0	\times	0	1
1	1	0	0	1
1	1	1	1	0

(c) LSR-Table

Tipos de Latches (L-SR com Habilitador)



(a) LSR-Enable

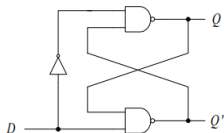


(b) LSR-Enable-Diagram

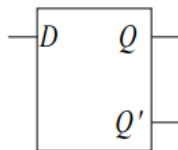
E	S	R	Q	Q_{next}	Q_{next}'
0	x	x	0	0	1
0	x	x	1	1	0
1	0	0	0	0	1
1	0	0	1	1	0
1	0	1	x	0	1
1	1	0	x	1	0
1	1	1	x	1	1

(c) LSR-Enable-Table

Tipos de Latches (L-D)



(a) LD

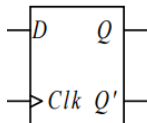


(b) LD-Diagram

D	Q	Q_{next}	Q_{next}'
0	\times	0	1
1	\times	1	0

(c) LD-Table

Tipos de Latches (FlipFlop D)

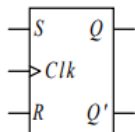


(a) FFD

Clk	D	Q	Q_{next}	Q_{next}'
0	\times	0	0	1
0	\times	1	1	0
1	\times	0	0	1
1	\times	1	1	0
\uparrow	0	\times	0	1
\uparrow	1	\times	1	0

(b) FFD-Table

Tipos de Latches (FlipFlop SR)

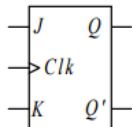


(a) FF SR

S	R	Q	Q_{next}	Q_{next}'
0	0	0	0	1
0	0	1	1	0
0	1	0	0	1
0	1	1	0	1
1	0	0	1	0
1	0	1	1	0
1	1	0	×	×
1	1	1	×	×

(b) FF SR-Table

Tipos de Latches (FlipFlop JK)



(a) FFJK

J	K	Q	Q_{next}	Q_{next}'
0	0	0	0	1
0	0	1	1	0
0	1	0	0	1
0	1	1	0	1
1	0	0	1	0
1	0	1	1	0
1	1	0	1	0
1	1	1	0	1

(b) FFJK-Table




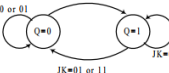




Name / Symbol	Characteristic (Truth) Table	State Diagram / Characteristic Equations	Excitation Table																																																								
SR 	<table><tr><th>S</th><th>R</th><th>Q</th><th>Q_{next}</th></tr><tr><td>0</td><td>0</td><td>0</td><td>0</td></tr><tr><td>0</td><td>0</td><td>1</td><td>1</td></tr><tr><td>0</td><td>1</td><td>0</td><td>0</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>0</td><td>1</td><td>1</td></tr><tr><td>1</td><td>1</td><td>0</td><td>×</td></tr><tr><td>1</td><td>1</td><td>1</td><td>×</td></tr></table>	S	R	Q	Q_{next}	0	0	0	0	0	0	1	1	0	1	0	0	0	1	1	0	1	0	0	1	1	0	1	1	1	1	0	×	1	1	1	×	<p>SR=10</p>  <p>$Q_{next} = S + R'Q$ $SR = 0$</p>	<table><tr><th>Q</th><th>Q_{next}</th><th>S</th><th>R</th></tr><tr><td>0</td><td>0</td><td>0</td><td>×</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>×</td><td>0</td></tr></table>	Q	Q_{next}	S	R	0	0	0	×	0	1	1	0	1	0	0	1	1	1	×	0
S	R	Q	Q_{next}																																																								
0	0	0	0																																																								
0	0	1	1																																																								
0	1	0	0																																																								
0	1	1	0																																																								
1	0	0	1																																																								
1	0	1	1																																																								
1	1	0	×																																																								
1	1	1	×																																																								
Q	Q_{next}	S	R																																																								
0	0	0	×																																																								
0	1	1	0																																																								
1	0	0	1																																																								
1	1	×	0																																																								
JK 	<table><tr><th>J</th><th>K</th><th>Q</th><th>Q_{next}</th></tr><tr><td>0</td><td>0</td><td>0</td><td>0</td></tr><tr><td>0</td><td>0</td><td>1</td><td>1</td></tr><tr><td>0</td><td>1</td><td>0</td><td>0</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>0</td><td>1</td><td>1</td></tr><tr><td>1</td><td>1</td><td>0</td><td>1</td></tr><tr><td>1</td><td>1</td><td>1</td><td>0</td></tr></table>	J	K	Q	Q_{next}	0	0	0	0	0	0	1	1	0	1	0	0	0	1	1	0	1	0	0	1	1	0	1	1	1	1	0	1	1	1	1	0	<p>JK=10 or 11</p>  <p>$Q_{next} = J'K'Q + JK' + JKQ'$ $= J'K'Q + JK'Q + JK'Q' + JKQ'$ $= K'Q(J'+J) + JQ'(K'+K)$ $= K'Q + JQ'$</p>	<table><tr><th>Q</th><th>Q_{next}</th><th>J</th><th>K</th></tr><tr><td>0</td><td>0</td><td>0</td><td>×</td></tr><tr><td>0</td><td>1</td><td>1</td><td>×</td></tr><tr><td>1</td><td>0</td><td>×</td><td>1</td></tr><tr><td>1</td><td>1</td><td>×</td><td>0</td></tr></table>	Q	Q_{next}	J	K	0	0	0	×	0	1	1	×	1	0	×	1	1	1	×	0
J	K	Q	Q_{next}																																																								
0	0	0	0																																																								
0	0	1	1																																																								
0	1	0	0																																																								
0	1	1	0																																																								
1	0	0	1																																																								
1	0	1	1																																																								
1	1	0	1																																																								
1	1	1	0																																																								
Q	Q_{next}	J	K																																																								
0	0	0	×																																																								
0	1	1	×																																																								
1	0	×	1																																																								
1	1	×	0																																																								
D 	<table><tr><th>D</th><th>Q</th><th>Q_{next}</th></tr><tr><td>0</td><td>×</td><td>0</td></tr><tr><td>1</td><td>×</td><td>1</td></tr></table>	D	Q	Q_{next}	0	×	0	1	×	1	<p>D=0</p>  <p>$Q_{next} = D$</p>	<table><tr><th>Q</th><th>Q_{next}</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	Q_{next}	D	0	0	0	0	1	1	1	0	0	1	1	1																																
D	Q	Q_{next}																																																									
0	×	0																																																									
1	×	1																																																									
Q	Q_{next}	D																																																									
0	0	0																																																									
0	1	1																																																									
1	0	0																																																									
1	1	1																																																									
T 	<table><tr><th>T</th><th>Q</th><th>Q_{next}</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>	T	Q	Q_{next}	0	0	0	0	1	1	1	0	1	1	1	0	<p>T=1</p>  <p>$Q_{next} = TQ' + T'Q = T \oplus Q$</p>	<table><tr><th>Q</th><th>Q_{next}</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	Q_{next}	T	0	0	0	0	1	1	1	0	1	1	1	0																										
T	Q	Q_{next}																																																									
0	0	0																																																									
0	1	1																																																									
1	0	1																																																									
1	1	0																																																									
Q	Q_{next}	T																																																									
0	0	0																																																									
0	1	1																																																									
1	0	1																																																									
1	1	0																																																									

Figura: Resumo dos FlipFlops

Flip-Flops em VHDL

- O Flip-Flop somente deve atualizar o seu estado quando ocorrer uma transição do estado do CLOCK.

Flip-Flops em VHDL

- O Flip-Flop somente deve atualizar o seu estado quando ocorrer uma transição do estado do CLOCK.
- Torna-se necessário implementar uma desvio condicional IF no VHDL.

Flip-Flops em VHDL

- O Flip-Flop somente deve atualizar o seu estado quando ocorrer uma transição do estado do CLOCK.
- Torna-se necessário implementar uma desvio condicional IF no VHDL.
- O ambiente PROCESS no VHDL permite implementar condicionais, laços e muitas outras funções típicas de algoritmos de programação.

O ambiente PROCESS é descrito com as variáveis envolvidas no processo em parênteses ().

```
1  library ieee;
2  use ieee.std_logic_1164.all;
3
4  entity ffjk is
5  port(j,k : in std_logic;
6       clock, preset, clear : in std_logic;
7       q : out std_logic);
8  end ffjk;
9
10 architecture ffjk of ffjk is
11 signal estado : std_logic;
12 begin
13 process(clock,preset,clear)
14 begin
15     if (preset = '1') then estado <= '1';
16     elsif (clear = '0') then estado <= '0';
17     elsif (clock='1' and clock'EVENT) then
18         if (j='0' and k='1') then estado <='0';
19         elsif (j='1' and k='0') then estado <='1';
20         elsif (j='1' and k='1') then estado <= not estado;
21         end if;
22     end if;
23 end process;
24 q <= estado;
25 end ffjk;
```

O comando *'EVENT* descreve uma transição de alguma variável binária.

Durante uma viagem de avião, os passageiros tem acesso a um botão para solicitar algo aos comissários de bordo. E os comissários tem acesso a outro para finalizar a solicitação.

- O botão A (dos passageiros) indica o chamado pendente, mantendo a luz correspondente ao seu acento ligado.
- O botão B (dos comissários) indica que a solicitação foi concluída, desligando a mesma luz.

Utilizando flip-flop tipo D, implemente VHDL e embarque na FPGA este circuito sequencial. Considere que o botão A tem prioridade sobre o botão B.