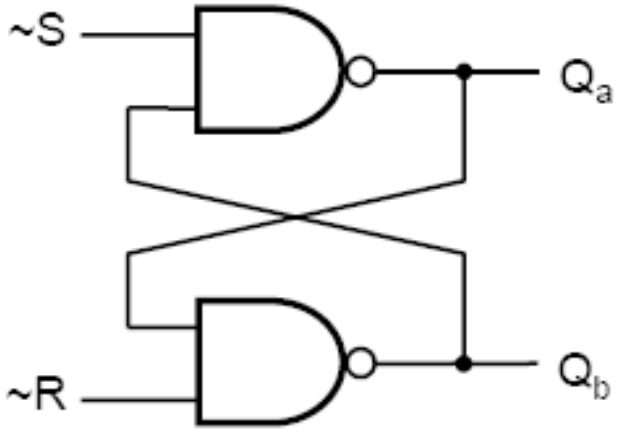
Projeto com Circuitos Reconfiguráveis

Projeto lógico sequencial
Flip-flops e latches

Prof. Daniel M. Muñoz Arboleda

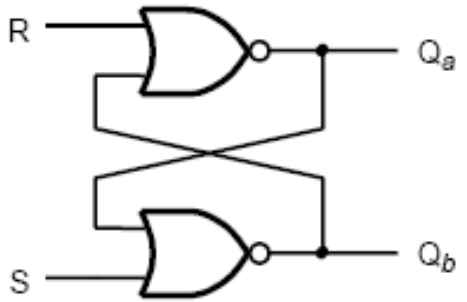
Novembro de 2012

Latch SR com NAND

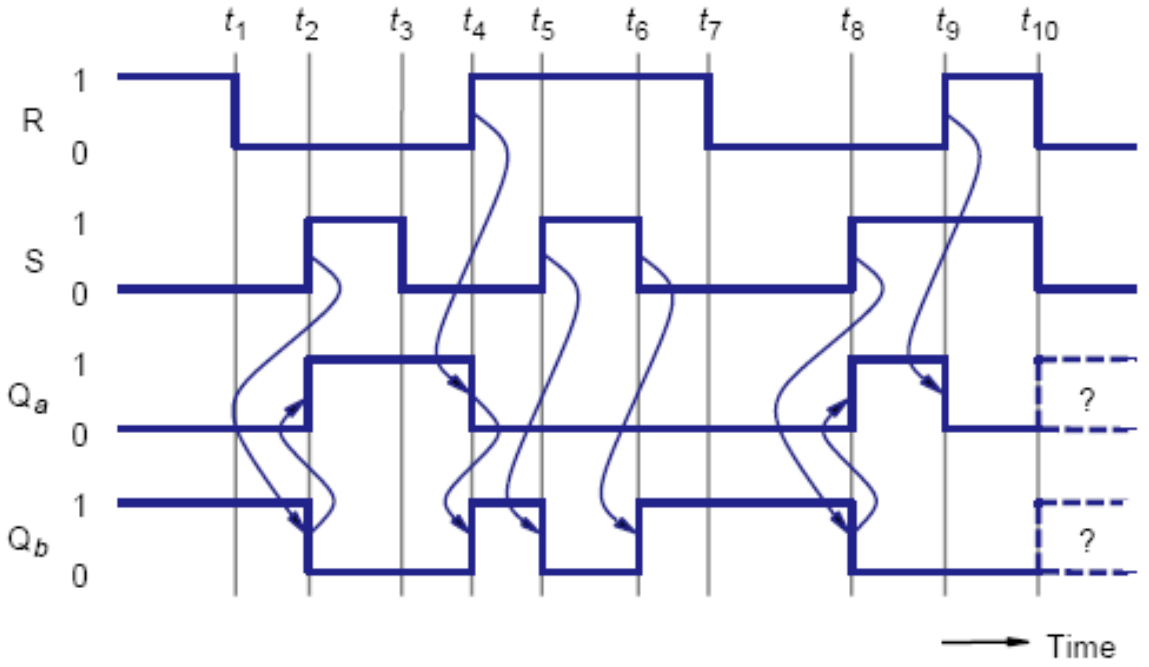


~S	~R	Q_a	Q_b	
1	1	0/1	1/0	(no change)
0	1	1	0	
1	0	0	1	
0	0	1	1	

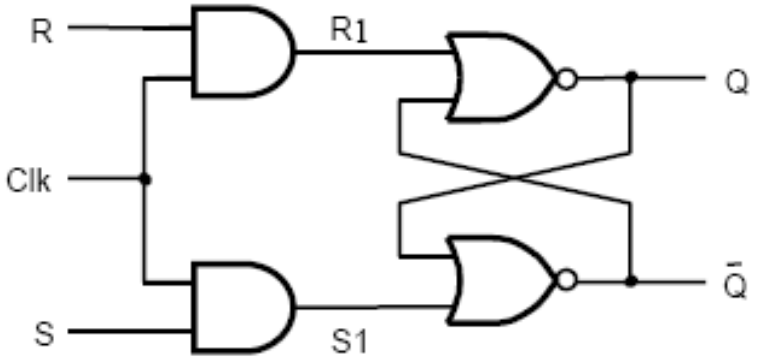
Latch SR com NORs



S	R	Q_a	Q_b	
0	0	0/1	1/0	(no change)
0	1	0	1	
1	0	1	0	
1	1	0	0	

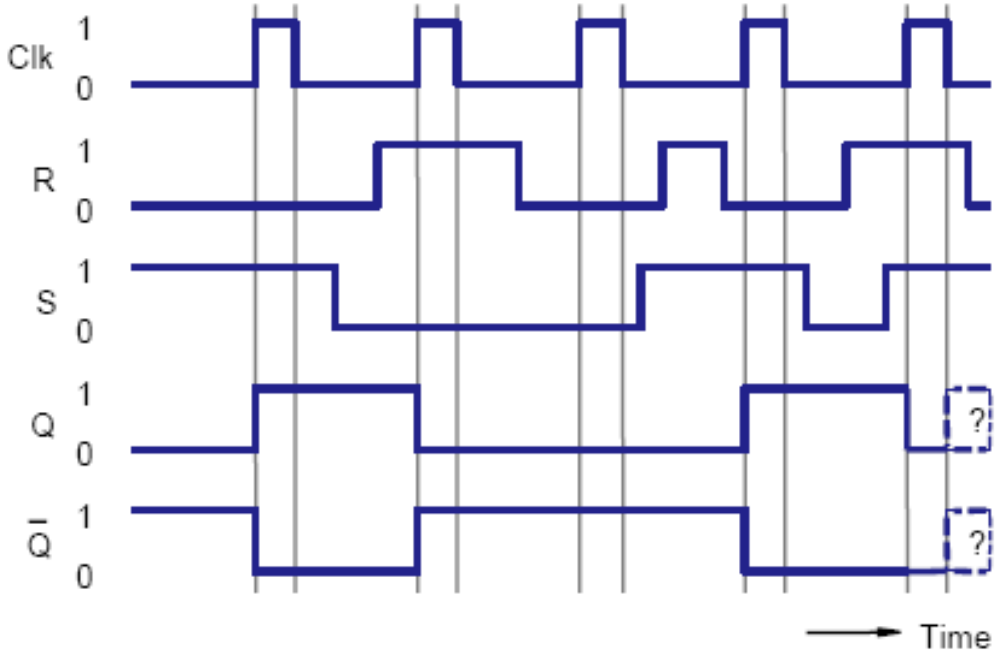


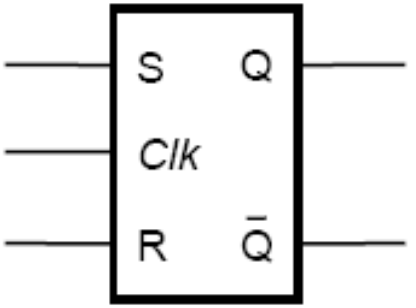
Latch SR chaveado



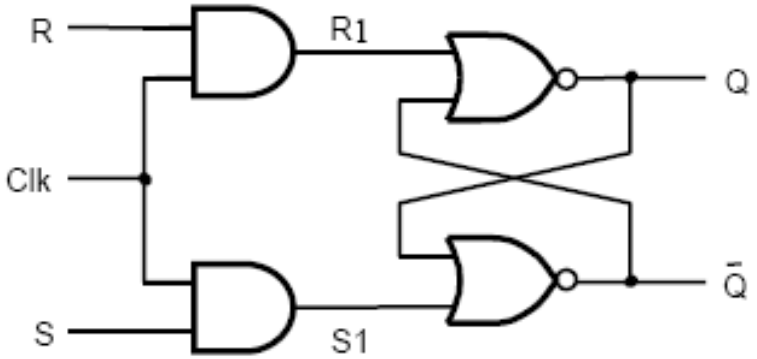
Clk	S	R	Q(t + 1)
0	x	x	Q(t) (no change)
1	0	0	Q(t) (no change)
1	0	1	0
1	1	0	1
1	1	1	x

Por que?





Latch SR chaveado



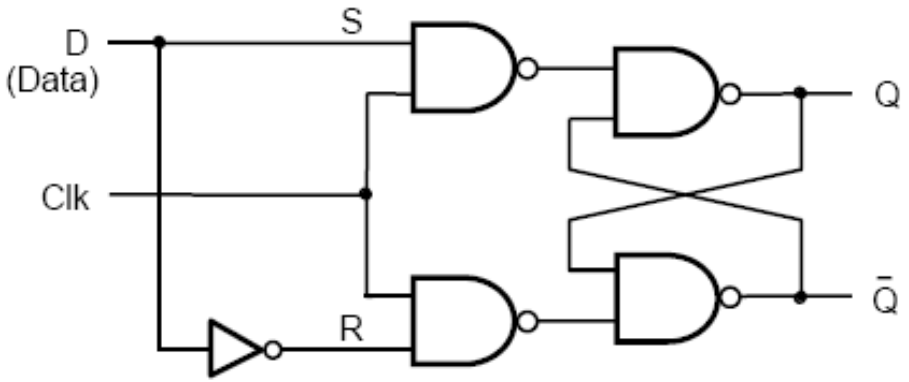
Clk	S	R	Q(t + 1)
0	x	x	Q(t) (no change)
1	0	0	Q(t) (no change)
1	0	1	0
1	1	0	1
1	1	1	x

Por que?

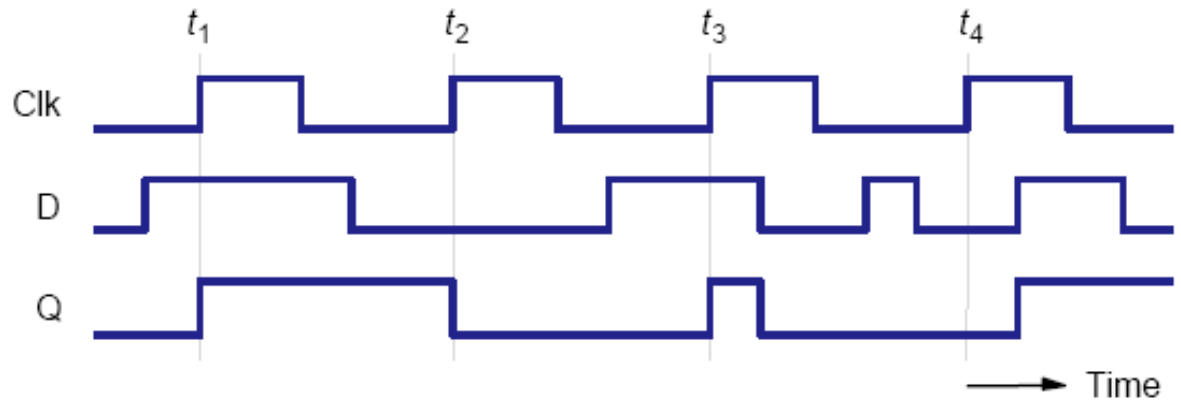
Um comportamento *oscilante* se apresenta quando S e R mudam de '1' para '0' exatamente ao mesmo tempo. Se as duas portas lógicas possuem o mesmo delay então aparecerá um '0' nas duas saídas exatamente ao mesmo tempo. Este estado será realimentado produzindo um '1' nas saídas exatamente ao mesmo tempo e posteriormente um '0' , '1', … Este comportamento oscilante continuará para sempre.

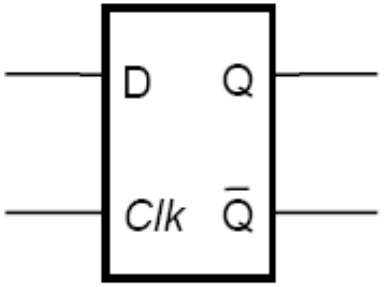
Se as portas lógicas não tem o mesmo delay, então o comportamento oscilante desaparece, porém, na prática não sabemos qual porta é mais rápida do que a outra. Portanto, não se sabe qual será o estado do Latch. Assim, é dito que o estado é *indefinido*.

Latch tipo D chaveado

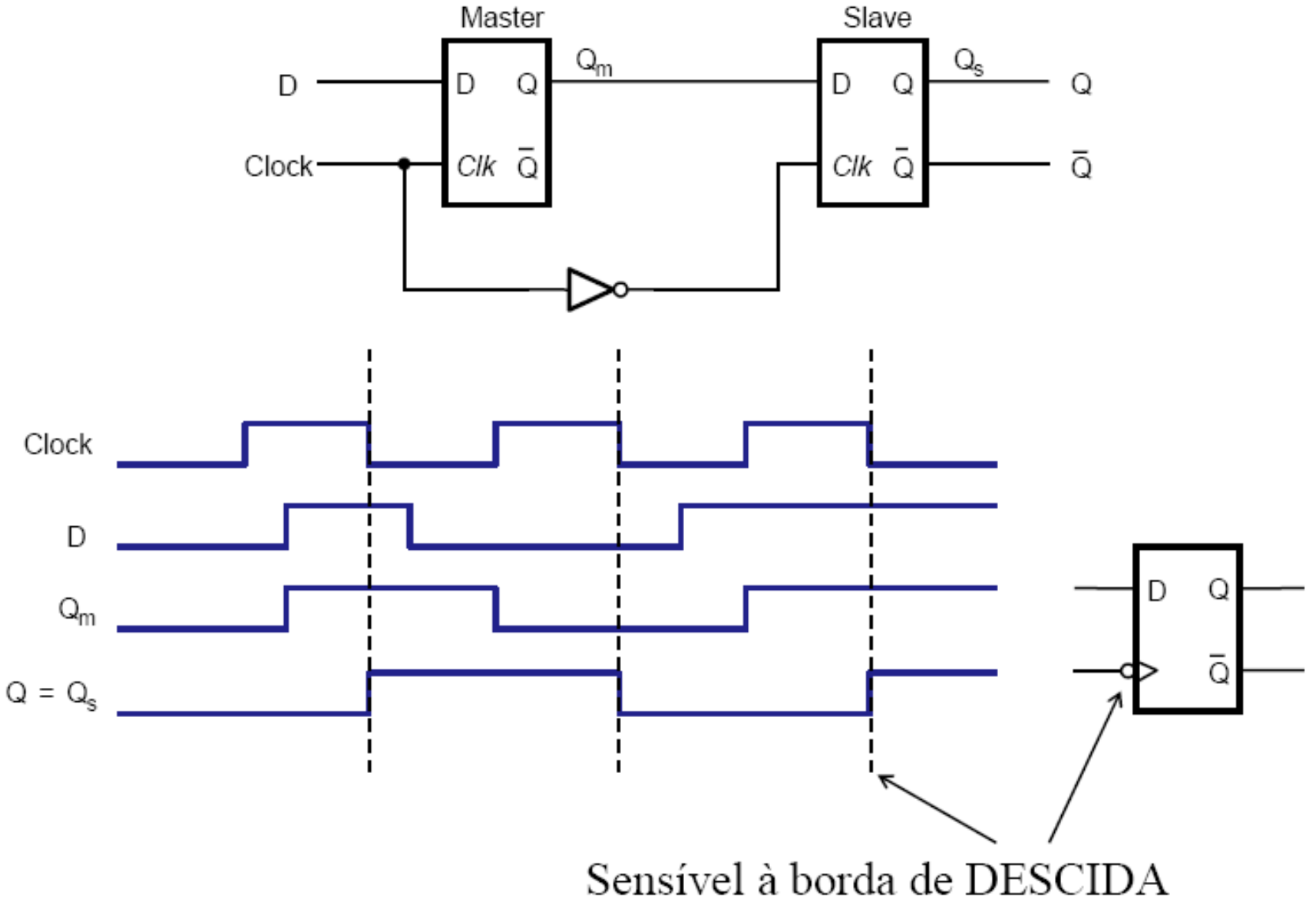


Clk	D	Q(t + 1)
0	x	Q(t)
1	0	0
1	1	1

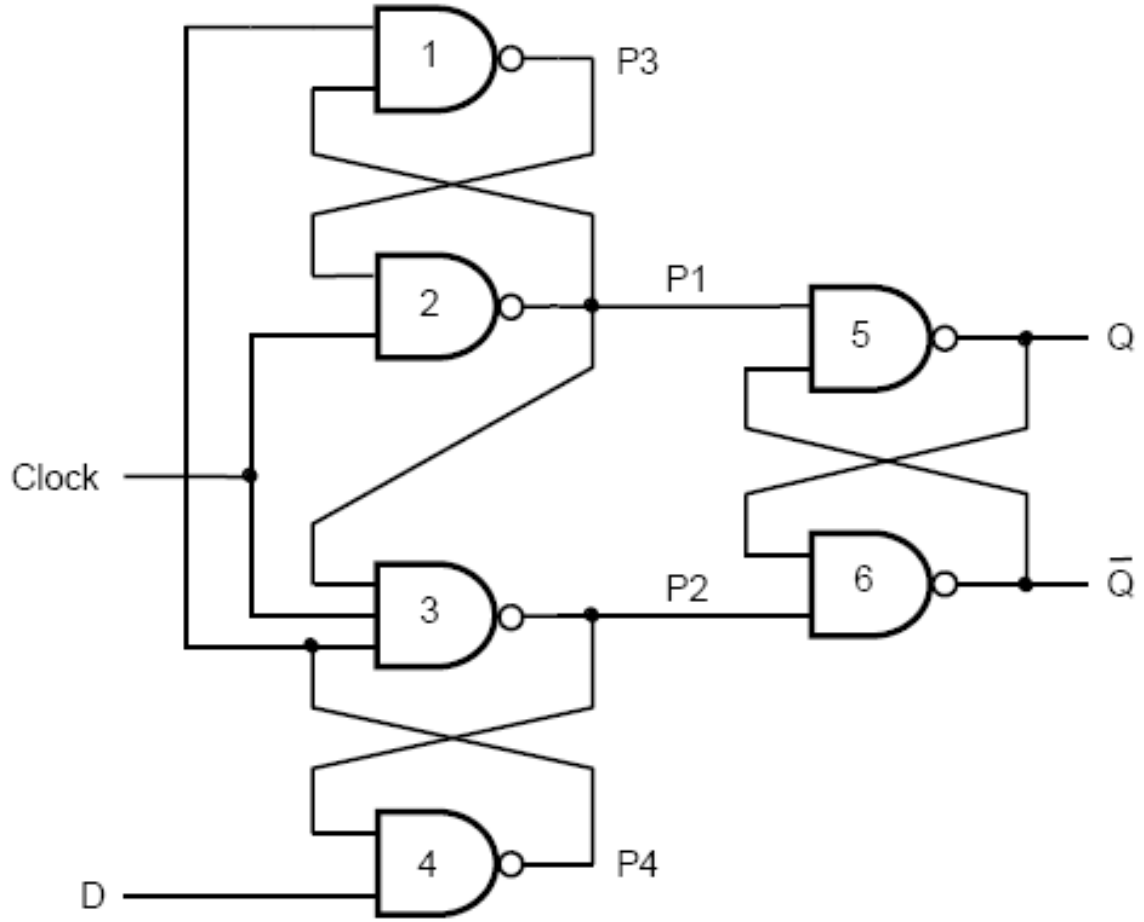


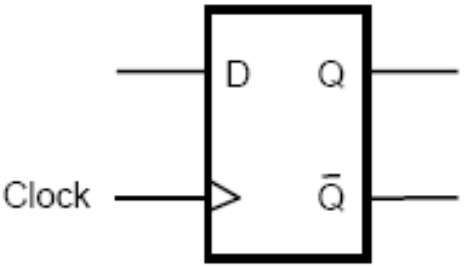


Flip-flop mestre escravo



Flip-flop D clássico

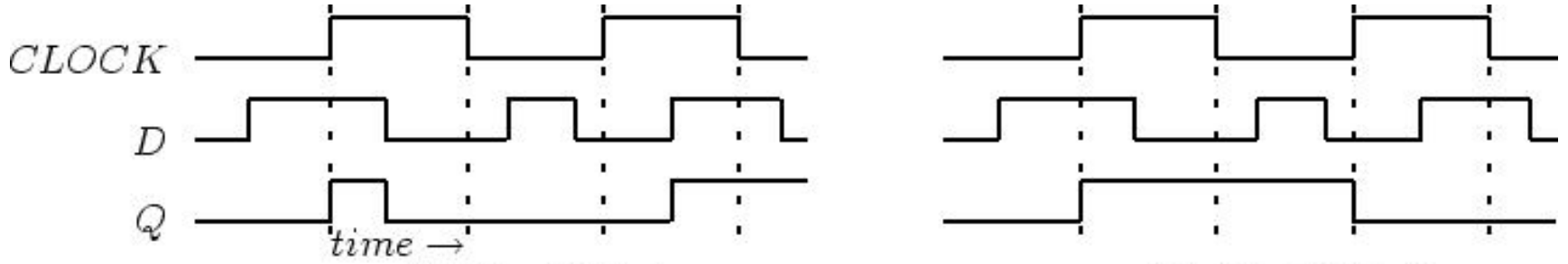




Flip-flop D vs Latch D

• Manifestação da saída Q em função de variações na entrada D:
  - Latch: transparente durante EN (ou Ck) ativos, ou seja, entrada D passa diretamente para a saída Q
  - Flip-Flop: na borda do clock, o valor presente na entrada D é transferido para Q

• Instante em que o valor da entrada D é armazenado:
  - Latch: valor armazenado é o presente na entrada D no instante em que EN (ou Ck) é desativado (operação de latch ou travamento)
  - Flip-Flop: na borda do Clock, o valor presente na entrada D é armazenado



(a) The D latch (b) The D flip flop

Flip-flop D e Latch D. Descrições em VHDL

• Latches ou Flip-flops são inferidos se todas as possibilidades de um condicional IF não estão explícitas na descrição de *hardware*.

• Latch é inferido quando a sentença IF é usada por *nível lógico*

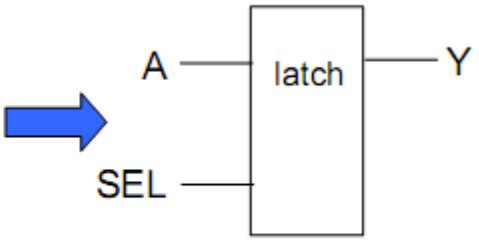
• Flip-Flop é inferido quando a sentença IF é usada por borda de clock

• As ferramentas de simulação precisam manter a saída prévia (criar uma memória) sob certas condições se as sentenças ELSE não são incluídas

Latches em VHDL

Um Latch é inferido quando um condicional IF é usado por nível lógico e não todas as possibilidades foram incluidas.

```
process (SEL, A)
begin
  if (SEL = '1') then Y <= A;
  end if ;
end process;
```

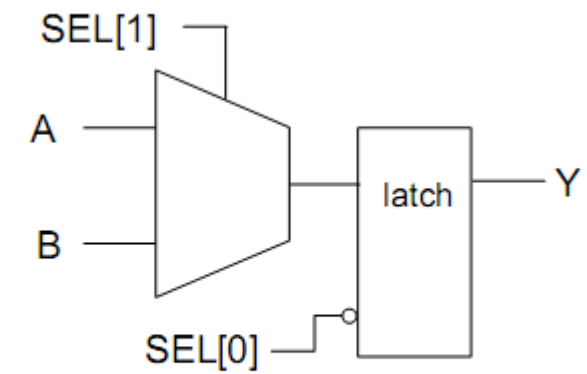


Para evitar latches, coloque as sentenças ELSE

Latches em VHDL

Um Latch pode ser inferido se quando as sentenças CASE usam
when others => null se o tipo de dado é std_logic ou std_logic_vector

```
-- sel, A, B are std_logic
process (SEL, A, B)
begin
case SEL is
  when "00" => Y <= A;
  when "10" => Y <= B;
  when others => null;
end case;
end process;
```



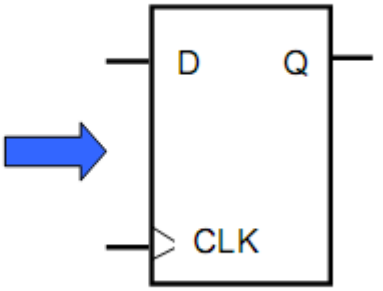
Para evitar latches, defina a saída para as outras condições, por exemplo,
when others => Y <= '0' ;

Flip-flops em VHDL

Dica: use Processos e condicionais IF para descrever lógica sequencial.

Um flip-flip é inferido se os condicionais IF detectam bordas de clock.

```
architecture BEHAVE of DF is
begin
  INFER: process (CLK) begin
    if (CLK'event and CLK ='1') then
         Q <= D;
    end if ;
  end process INFER;
end BEHAVE;
```

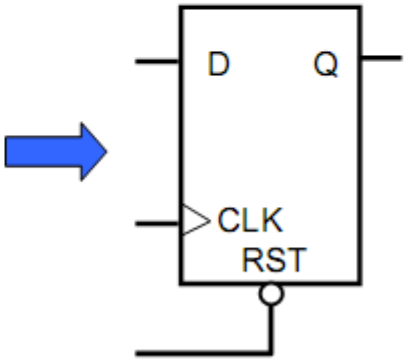


Flip-flops em VHDL (exemplo 1)

Um flip-flip é inferido se os condicionais IF detectam bordas de clock.

```
architecture FLOP of DFCLR is
begin
  INFER: process (CLK, RST)
  begin
    if (RST ='0') then
        Q <=  '0';
    elsif (CLK'event and CLK ='1') then
         Q  <= D;
   end if ;
  end process INFER;
end FLOP;
```

D flip-flop with asynchronous low reset and active high clock edge

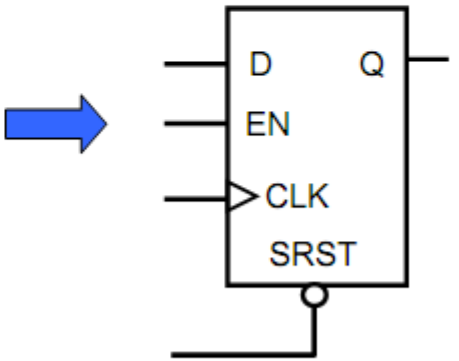


Flip-flops em VHDL (exemplo 2)

Um flip-flip é inferido se os condicionais IF detectam bordas de clock.

```
architecture FLOP of DFSLRHE is
begin
  INFER: process (CLK)
  begin
    if (CLK'event and CLK ='1') then
       if (SRST = '0') then
         Q <= '0';
       elsif (EN = '1') then
         Q  <= D;
       end if ;
    end if ;
  end process INFER;
end FLOP;
```

D flip-flop with synchronous low reset, active high enable and rising edge clock

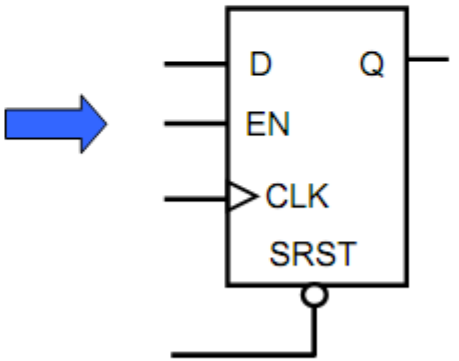


Flip-flops em VHDL (exemplo 3)

Dica: descreva lógica sequencial usando *reset* síncrono !

```
architecture FLOP of DFSLRHE is
begin
  INFER: process (CLK)
  begin
    if (CLK'event and CLK ='1') then
       if (SRST = '0') then
         Q <= '0';
       elsif (EN = '1') then
         Q  <= D;
       end if ;
    end if ;
  end process INFER;
end FLOP;
```

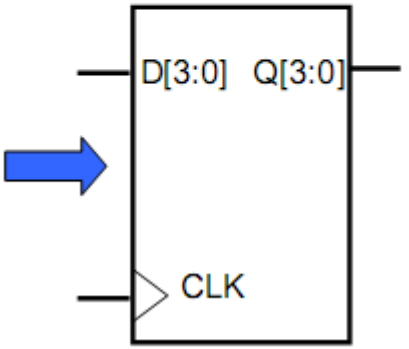
D flip-flop with synchronous low reset, active high enable and rising edge clock



Flip-flops em VHDL (exemplo 4)

```
library ieee;
use ieee.std_logic_1164.all;
entity DF_4 is
port (D: in std_logic_vector(3 downto 0);
      CLK: in std_logic;
      Q: out std_logic_vector(3 downto 0));
end DF_4 ;
architecture FLOP of DF_4 is
begin
  INFER: process ←— Where's the sensitivity list?
  begin
    wait until (CLK'event and CLK ='1');
         Q <= D;
  end process INFER;
end FLOP;
```

4-bit register using WAIT statement



Flip-flops em VHDL (recapitulando)

```
architecture FLOP of EN_FLOP is
begin
  INFER:process (CLK) begin
    if (CLK'event and CLK ='0') then
      if (EN = '0') then
        Q  <= D;
      end if ;
    end if ;
  end process INFER;
end FLOP;
```

1. flip-flop ativo em borda de subida ou descida ?
2. O *enable* é síncrono ou assíncrono ?
3. O *enable* é ativo em nível lógico alto ou baixo ?

Flip-flops em VHDL (exemplo 4)

```
library ieee;
use ieee.std_logic_1164.all;
entity DF_4 is
port (D: in std_logic_vector(3 downto 0);
      CLK: in std_logic;
      Q: out std_logic_vector(3 downto 0));
end DF_4 ;
architecture FLOP of DF_4 is
begin
  INFER: process ←— Where's the sensitivity list?
  begin
    wait until (CLK'event and CLK ='1');
         Q <= D;
  end process INFER;
end FLOP;
```

4-bit register using WAIT statement