GTA / UFRJ
GRUPO DE TELEINFORMÁTICA E AUTOMAÇÃO

Politécnica UFRJ

COPPE UFRJ

Circuitos Lógicos

Aula 10

cruz@gta.ufrj.br http://gta.ufrj.br/~cruz

Poli – Dept. de Eng. Eletrônica e de Computação
Universidade Federal do Rio de Janeiro

Na última aula

- Memória
  - Propriedades gerais
    - Volatilidade
    - Mutabilidade
    - Acesso
  - Tipos existentes no mercado
    - Fita (velharia)
    - Disco rígido
    - SRAM
    - DRAM

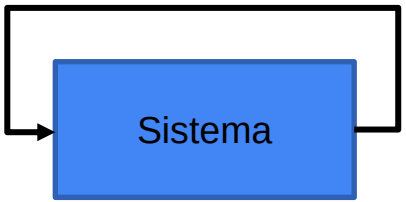
Hoje

- Latch
- Flip-flop
  - SR

Realimentação

Saída realimentando entrada

- Ligar a saída a uma das entradas pode ser vantajoso
  - Regulação
  - Recursão
  - Realimentação

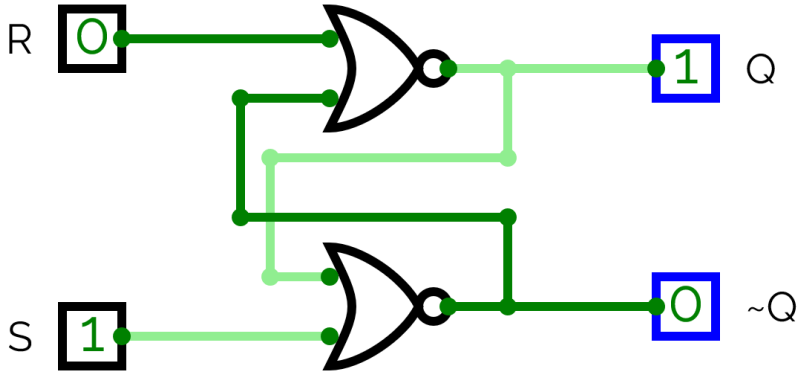


Latch SR (set/*reset*) com NOR

O *latch* SR com NOR

$$Q = \overline{R + \bar{Q}}$$

$$\bar{Q} = \overline{S + Q}$$

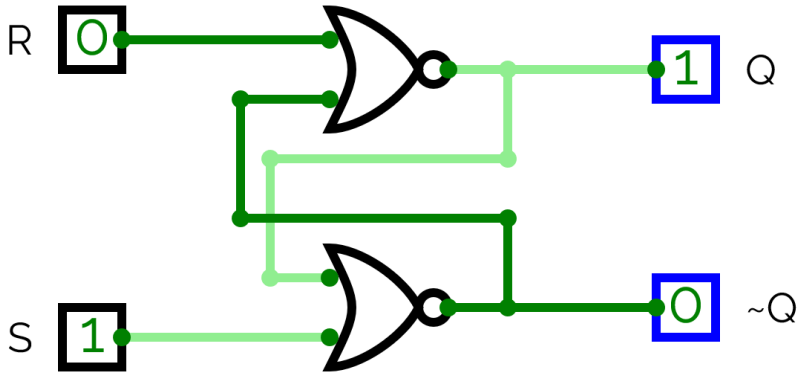


O *latch* SR com NOR

- $S = 1, \mathrm{R} = 0$
  - $\bar{Q} = \overline{1 + Q}$
    - $\boldsymbol{\bar{Q} = 0}$
  - $Q = \overline{0 + 0}$
    - $\boldsymbol{Q = 1}$

$$Q = \overline{R + \bar{Q}}$$

$$\bar{Q} = \overline{S + Q}$$



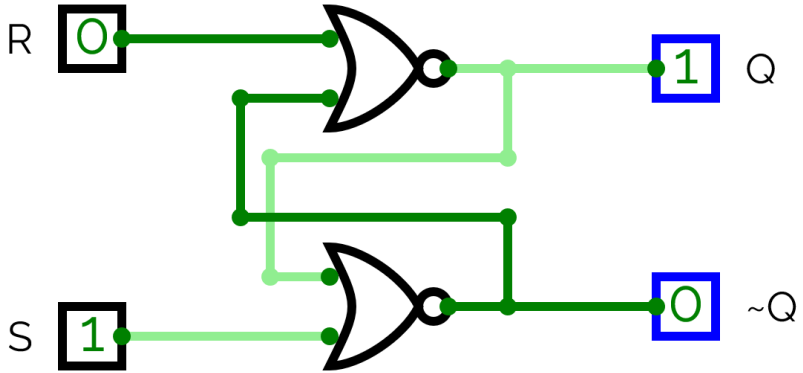
O *latch* SR com NOR

$$Q = \overline{R + \bar{Q}}$$

$$\bar{Q} = \overline{S + Q}$$

- $S = 1, \mathrm{R} = 0$
  - $\bar{Q} = \overline{1 + Q}$
    - $\boldsymbol{\bar{Q} = 0}$
  - $Q = \overline{0 + 0}$
    - $\boldsymbol{Q = 1}$

Q=1
Latch está “setado”

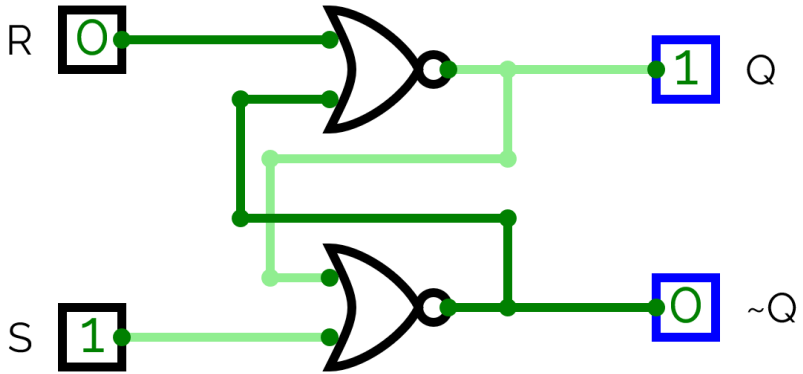


O *latch* SR com NOR

$$Q = \overline{R + \bar{Q}}$$

$$\bar{Q} = \overline{S + Q}$$

- $S = 1, \mathrm{R} = 0$
  - $\bar{Q} = \overline{1 + Q}$
    - $\boldsymbol{\bar{Q} = 0}$
  - $Q = \overline{0 + 0}$
    - $\boldsymbol{Q = 1}$
- $S = 0, \mathrm{R} = 0$
  - $\bar{Q} = \overline{0 + 1}$
    - $\boldsymbol{\bar{Q} = 0}$
  - $Q = \overline{0 + 0}$
    - $\boldsymbol{Q = 1}$

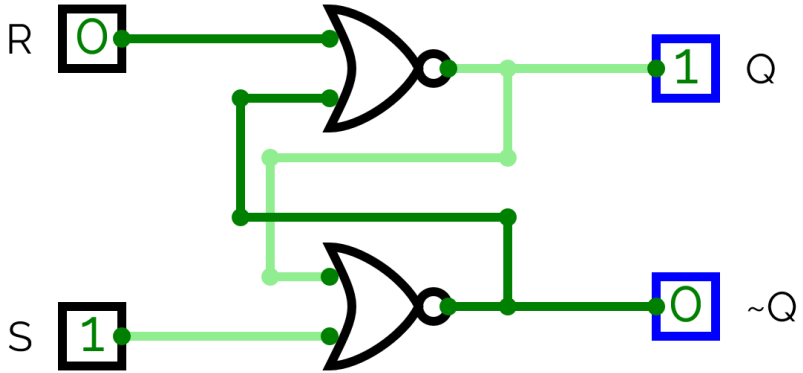


O *latch* SR com NOR

- $S = 0, \mathrm{R} = 1$
  - $Q = \overline{1 + \bar{Q}}$
    - $\boldsymbol{Q = 0}$
  - $\bar{Q} = \overline{0 + 0}$
    - $\boldsymbol{\bar{Q} = 1}$

$$Q = \overline{R + \bar{Q}}$$

$$\bar{Q} = \overline{S + Q}$$



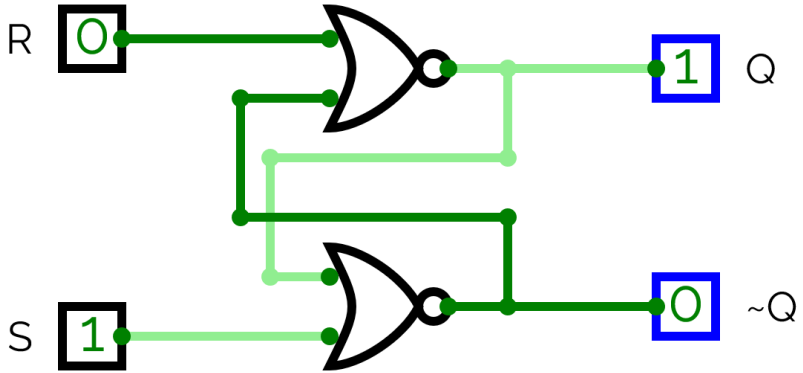
O *latch* SR com NOR

$$Q = \overline{R + \bar{Q}}$$

$$\bar{Q} = \overline{S + Q}$$

- $S = 0, \mathrm{R} = 1$
  - $Q = \overline{1 + \bar{Q}}$
    - $\boldsymbol{Q = 0}$
  - $\bar{Q} = \overline{0 + 0}$
    - $\boldsymbol{\bar{Q} = 1}$

Q=0
Latch está “resetado”



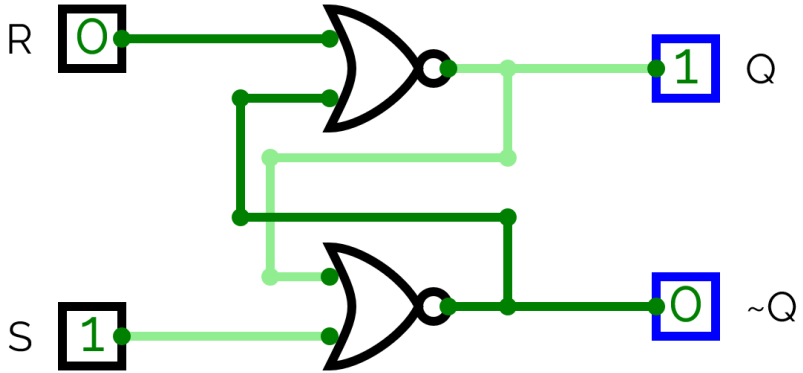
O *latch* SR com NOR

$$Q = \overline{R + \bar{Q}}$$

$$\bar{Q} = \overline{S + Q}$$

- $S = 0, \mathrm{R} = 1$
  - $Q = \overline{1 + \bar{Q}}$
    - $\boldsymbol{Q = 0}$
  - $\bar{Q} = \overline{0 + 0}$
    - $\boldsymbol{\bar{Q} = 1}$

- $S = 0, \mathrm{R} = 0$
  - $Q = \overline{0 + 1}$
    - $\boldsymbol{Q = 0}$
  - $\bar{Q} = \overline{0 + 0}$
    - $\boldsymbol{\bar{Q} = 1}$

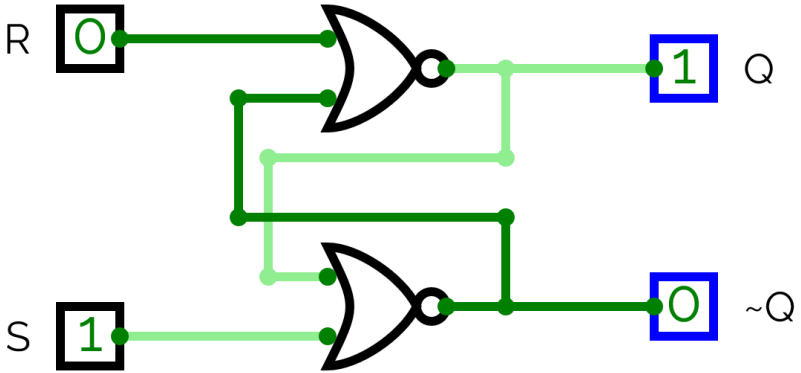


Recapitulando...

$$Q = \overline{R + \bar{Q}}$$

$$\bar{Q} = \overline{S + Q}$$

R	S	Q	$\bar{Q}$
0	1	1	0
0	0	1	0
1	0	0	1
0	0	0	1



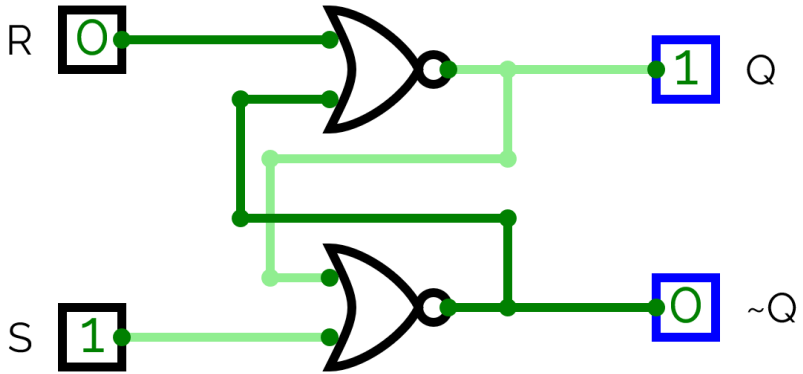
Recapitulando...

- $S = 0$ e $R = 0$
  - Q e $\bar{Q}$ são definidos pelo estado anterior!

$$Q = \overline{R + \bar{Q}}$$

$$\bar{Q} = \overline{S + Q}$$

R	S	Q	$\bar{Q}$
0	1	1	0
0	0	1	0
1	0	0	1
0	0	0	1

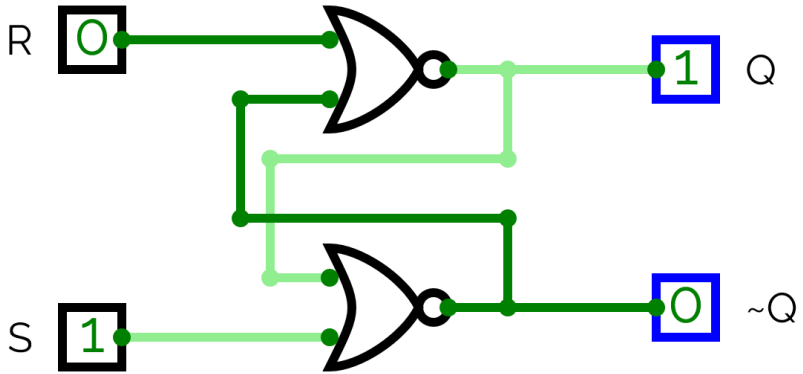


O *latch* SR com NOR

$$Q = \overline{R + \bar{Q}}$$

$$\bar{Q} = \overline{S + Q}$$

- E se colocarmos $S = 1, \mathrm{R} = 1$?
  - $Q = \overline{1 + \bar{Q}}$
    - $\boldsymbol{Q = 0}$
  - $\bar{Q} = \overline{1 + Q}$
    - $\boldsymbol{\bar{Q} = 0}$
- Estado **não permitido**
  - Propriedade de Q e $\bar{Q}$ é violada!
- *Latch* não é um *latch*

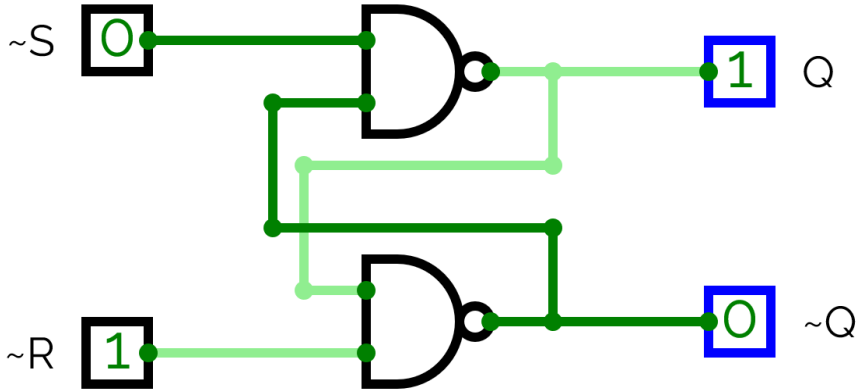


Latch SR (set/*reset*) com NAND

O *latch* SR com NAND

$$Q = \overline{\bar{S} \cdot \bar{Q}}$$

$$\bar{Q} = \overline{\bar{R} \cdot Q}$$

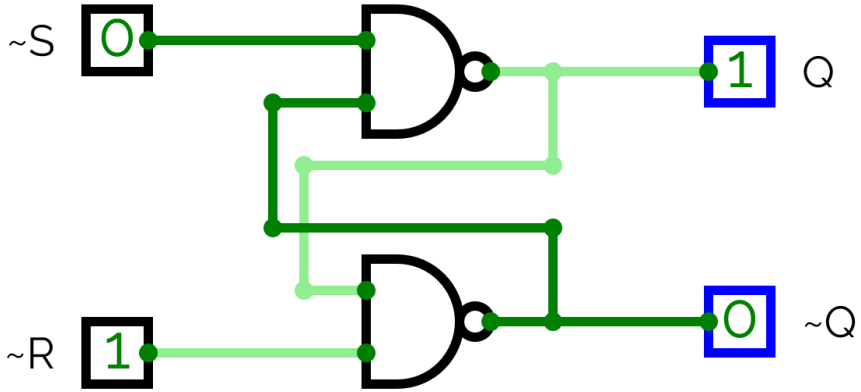


O *latch* SR com NAND

$$Q = \overline{\bar{S} \cdot \bar{Q}}$$

$$\bar{Q} = \overline{\bar{R} \cdot Q}$$

- $\bar{S} = 0, \bar{R} = 1$
  - $Q = \overline{0 \cdot \bar{Q}}$
    - $\boldsymbol{Q = 1}$
  - $\bar{Q} = \overline{1 \cdot 1}$
    - $\boldsymbol{\bar{Q} = 0}$



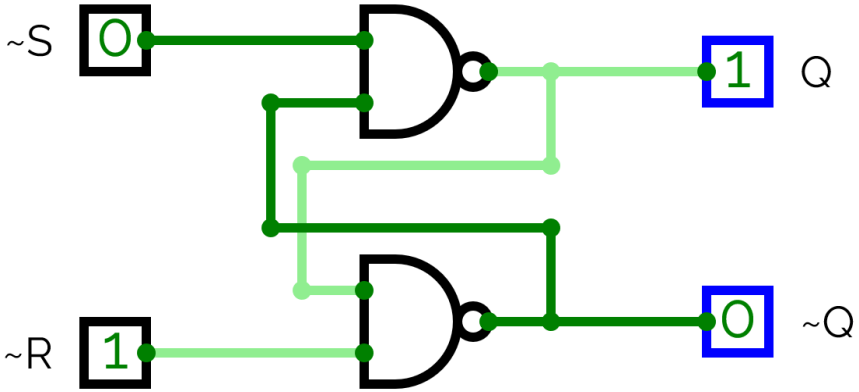
O *latch* SR com NAND

$$Q = \overline{\bar{S} \cdot \bar{Q}}$$

$$\bar{Q} = \overline{\bar{R} \cdot Q}$$

- $\bar{S} = 0, \bar{R} = 1$
  - $Q = \overline{0 \cdot \bar{Q}}$
    - $\boldsymbol{Q = 1}$
  - $\bar{Q} = \overline{1 \cdot 1}$
    - $\boldsymbol{\bar{Q} = 0}$

Q=1
Latch está "setado"

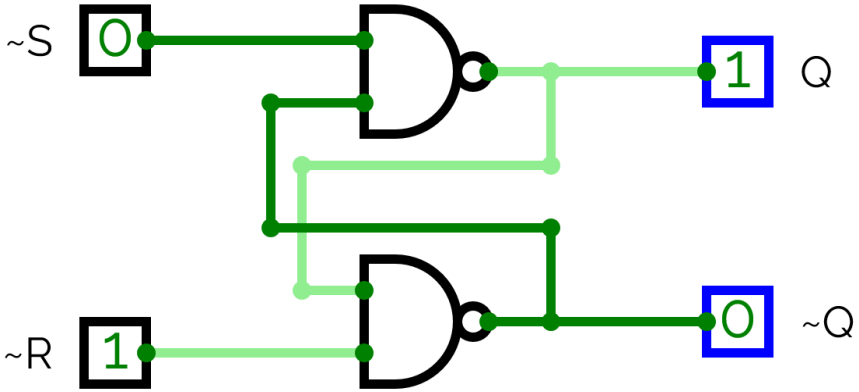


O *latch* SR com NAND

$$Q = \overline{\bar{S} \cdot \bar{Q}}$$

$$\bar{Q} = \overline{\bar{R} \cdot Q}$$

- $\bar{S} = 0, \bar{R} = 1$
  - $Q = \overline{0 \cdot \bar{Q}}$
    - $\boldsymbol{Q = 1}$
  - $\bar{Q} = \overline{1 \cdot 1}$
    - $\boldsymbol{\bar{Q} = 0}$
- $\bar{S} = 1, \bar{R} = 1$
  - $Q = \overline{1 \cdot 0}$
    - $\boldsymbol{Q = 1}$
  - $\bar{Q} = \overline{1 \cdot 1}$
    - $\boldsymbol{\bar{Q} = 0}$

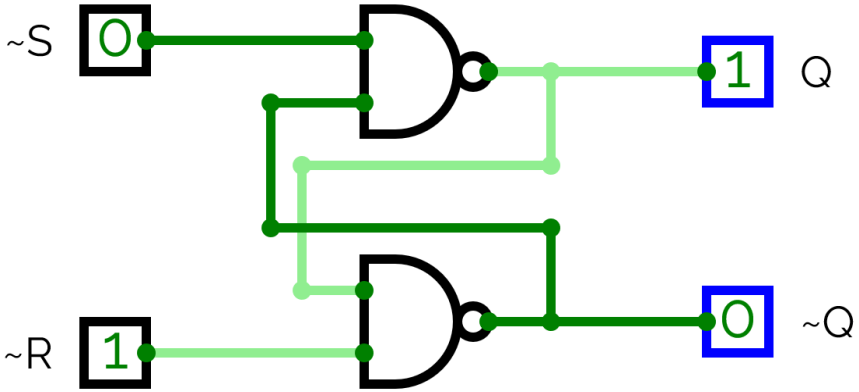


O *latch* SR com NAND

$$Q = \overline{\bar{S} \cdot \bar{Q}}$$

$$\bar{Q} = \overline{\bar{R} \cdot Q}$$

- $\bar{S} = 1, \bar{R} = 0$
  - $\bar{Q} = \overline{0 \cdot Q}$
    - $\boldsymbol{\bar{Q} = 1}$
  - $Q = \overline{1 \cdot 1}$
    - $\boldsymbol{Q = 0}$



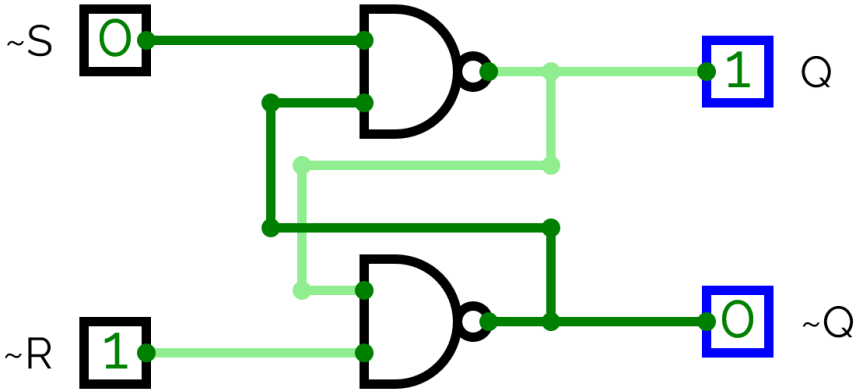
O *latch* SR com NAND

$$Q = \overline{\bar{S} \cdot \bar{Q}}$$

$$\bar{Q} = \overline{\bar{R} \cdot Q}$$

- $\bar{S} = 1, \bar{R} = 0$
  - $\bar{Q} = \overline{0 \cdot Q}$
    - $\boldsymbol{\bar{Q} = 1}$
  - $Q = \overline{1 \cdot 1}$
    - $\boldsymbol{Q = 0}$

Q=0
Latch está “resetado”

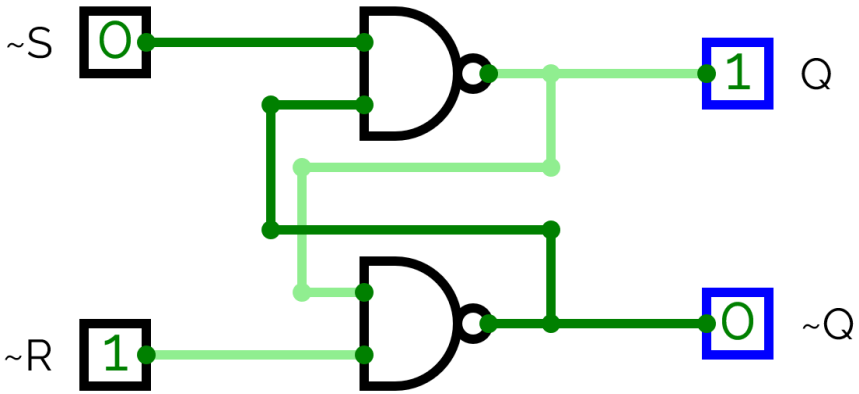


O *latch* SR com NAND

$$Q = \overline{\bar{S} \cdot \bar{Q}}$$

$$\bar{Q} = \overline{\bar{R} \cdot Q}$$

- $\bar{S} = 1, \bar{R} = 0$
  - $\bar{Q} = \overline{0 \cdot Q}$
    - $\boldsymbol{\bar{Q} = 1}$
  - $Q = \overline{1 \cdot 1}$
    - $\boldsymbol{Q = 0}$
- $\bar{S} = 1, \bar{R} = 1$
  - $\bar{Q} = \overline{1 \cdot 0}$
    - $\boldsymbol{\bar{Q} = 1}$
  - $Q = \overline{1 \cdot 0}$
    - $\boldsymbol{Q = 0}$



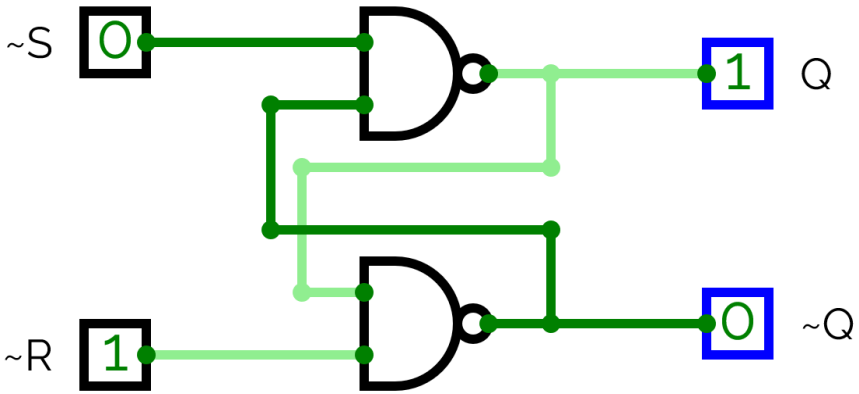
O *latch* SR com NAND

- $\bar{S} = 1, \bar{R} = 1$
  - Q e $\bar{Q}$ são definidos pelo estado anterior!

$$Q = \overline{\bar{S} \cdot \bar{Q}}$$

$$\bar{Q} = \overline{\bar{R} \cdot Q}$$

$\bar{R}$	$\bar{S}$	Q	$\bar{Q}$
1	0	1	0
1	1	1	0
0	1	0	1
1	1	0	1



Tá, mas e daí?

- Em ambos os casos, saída depende do resultado anterior
  - Circuito “lembra” do resultado anterior
    - Memória

R	S	Q
1	0	0
0	1	1
0	0	$Q_{anterior}$

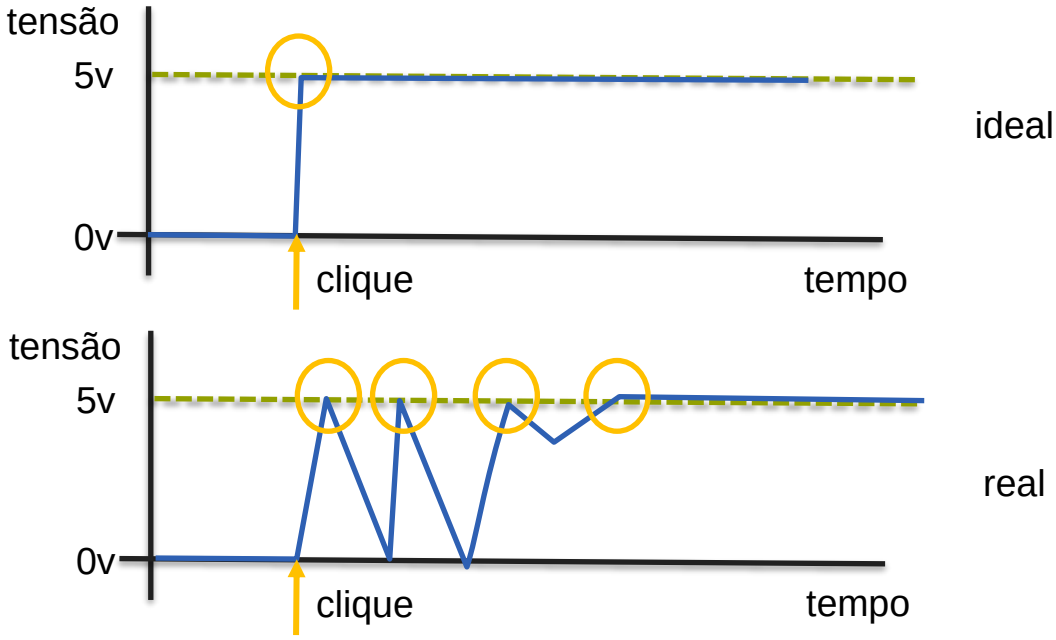
Tabela verdade para o *latch* com portas NOR

$\overline{R}$	$\overline{S}$	Q
1	0	1
0	1	0
1	1	$Q_{anterior}$

Tabela verdade para o *latch* com portas NAND

Exemplo de aplicação: *debouncing**

- Ao apertar um botão, o valor lógico não vai de 0 a 1 imediatamente
  - A tensão “quica”
- Necessário um circuito que entenda
  - Segure o valor em 1 por alguns instantes

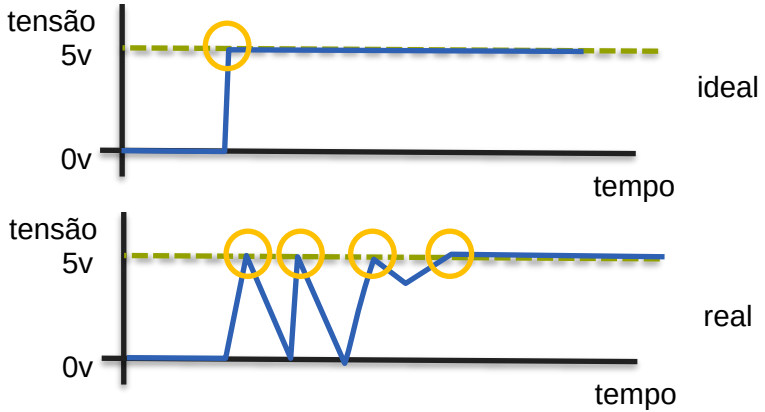


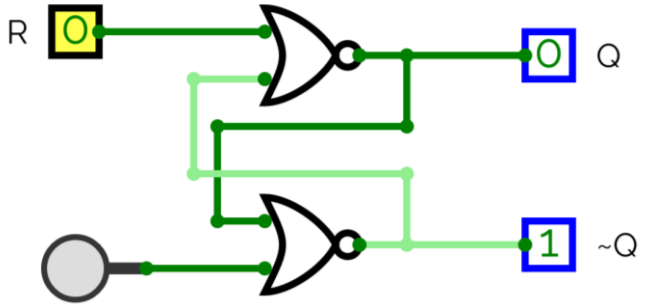
Um clique gerou 4 idas e voltas de tensão – e uma leitura de 4 cliques

**To bounce* significa “quicar”

Exemplo de aplicação: *debouncing**

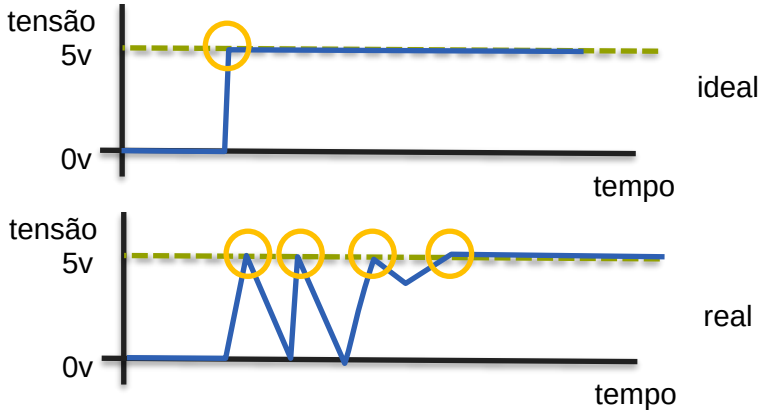
- Botão ligado ao S
  - 0 quando solto
  - 1 quando apertado
- Circuito é “resetado”
  - $S = 0, \mathrm{R} = 1$
- Circuito é colocado no estado de “mantém”
  - $S = 0, \mathrm{R} = 0$
- Quando botão for apertado
  - Circuito é “setado”
- Quando tensão “quicar”
  - Circuito **mantém**
- Temporizador deve resetar o circuito

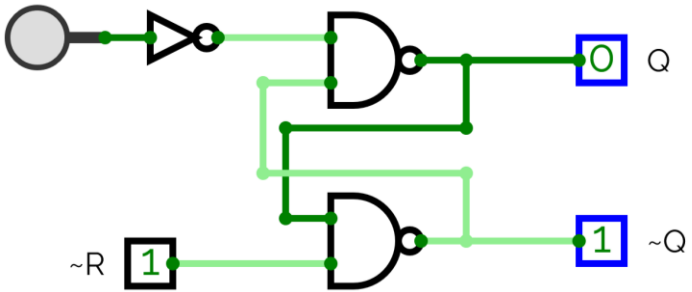




Exemplo de aplicação: *debouncing**

- Botão ligado ao $\bar{S}$ com uma porta inversora
  - 1 quando solto
  - 0 quando apertado
- Circuito é “resetado”
  - $\bar{S} = 1, \bar{R} = 0$
- Circuito é colocado no estado de “mantém”
  - $\bar{S} = 1, \bar{R} = 1$
- Quando botão for apertado
  - Circuito é “setado”
- Quando tensão “quicar”
  - Circuito **mantém**
- Temporizador deve resetar o circuito



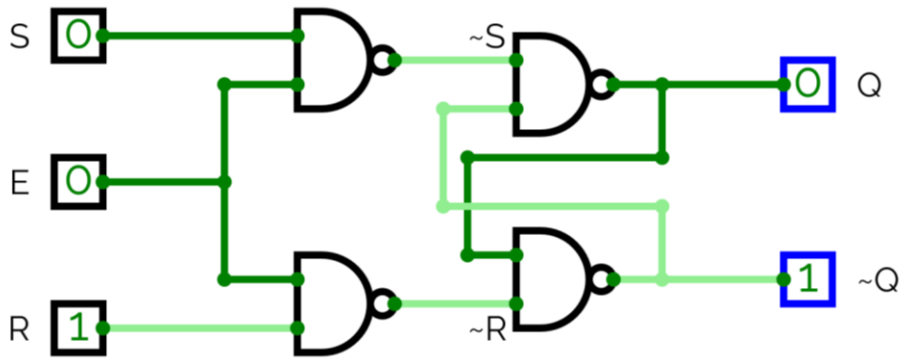


Enable

- *Latch* guarda o estado anterior
  - Onde começa o **antes**?
  - Onde começa o **agora**?

Enable

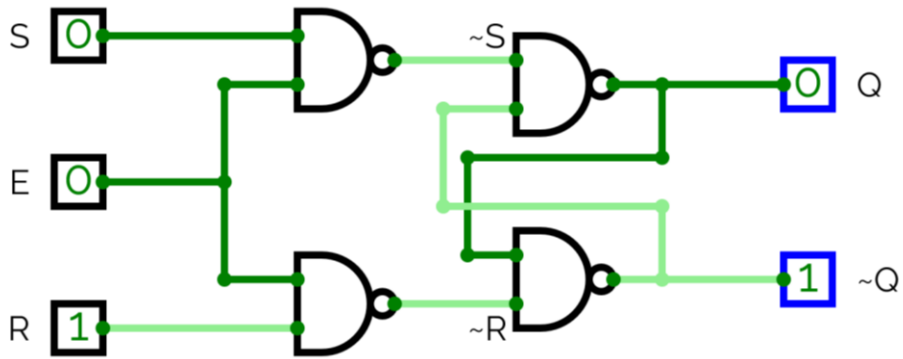
- *Latch* guarda o estado anterior
  - Se existe **antes** e **agora**, deve haver uma forma de delimitar quando é o **antes** e quando é o **agora**



Enable

- *Latch* guarda o estado anterior
  - Se existe **antes** e **agora**, deve haver uma forma de delimitar quando é o **antes** e quando é o **agora**

- *Enable* controla entrada do *latch*
  - *A*ceita entrada quando *enable* é 1
  - Ignora entrada quando *enable* é 0



O módulo *latch SR*

- Módulo contendo o *latch* SR
  - Entrada S
  - Entrada R
  - Entrada E (*enable*)
  - Saída Q
  - Saída $\bar{Q}$

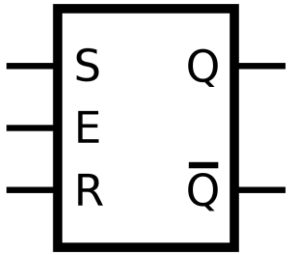
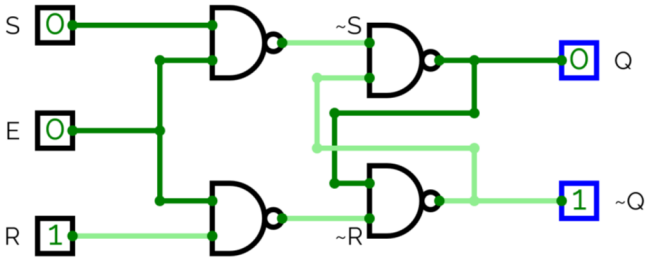


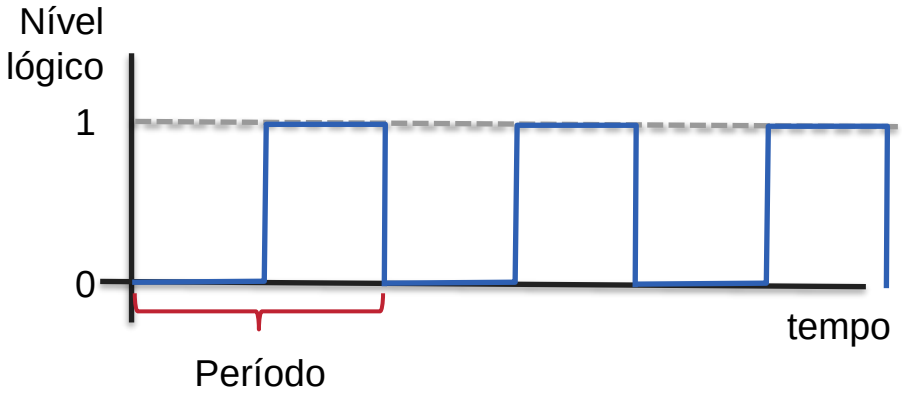
Imagem retirada da Wikipedia



Clock

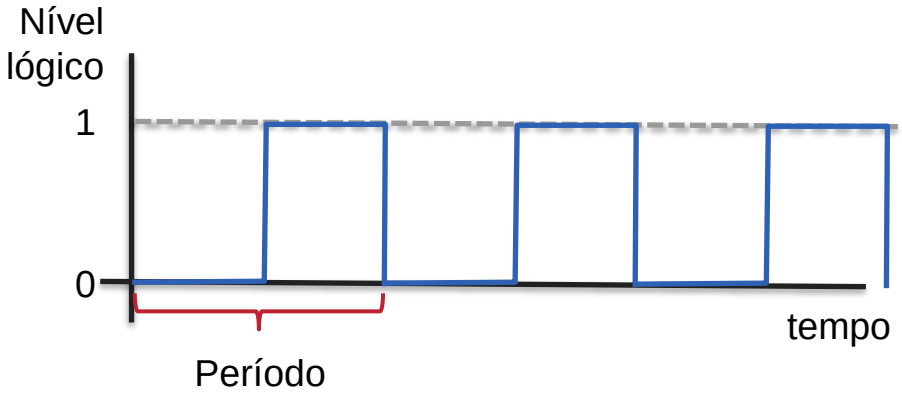
Clock

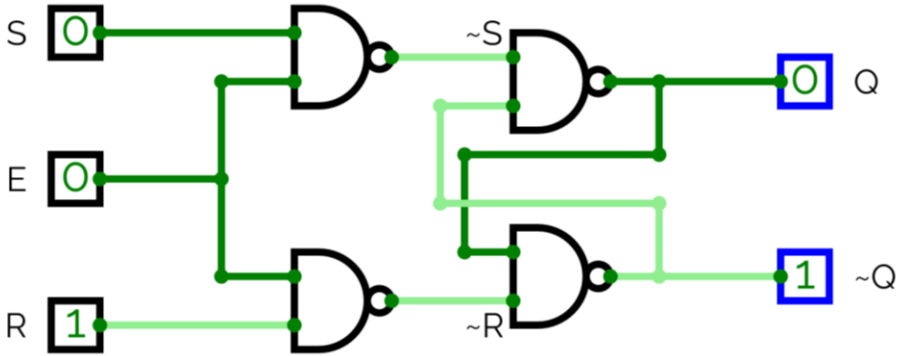
- Sinal que oscila entre 0 e 1
- Garante a sincronia dos circuitos
- Fornecido por um gerador de *clock*
- Ciclo de trabalho 50%
  - (ou seja: mesmo tempo em nível lógico 1 e 0)



Clock

- Ligar o *clock* no *enable*
  - Limitar o tempo que o circuito “aceita” entradas

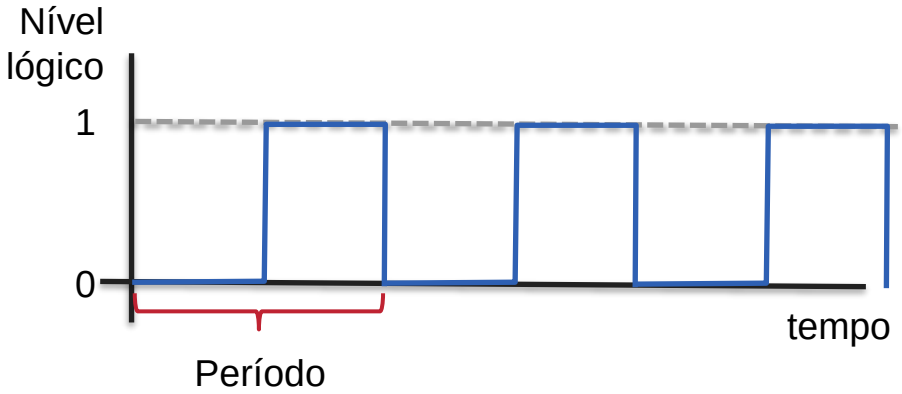


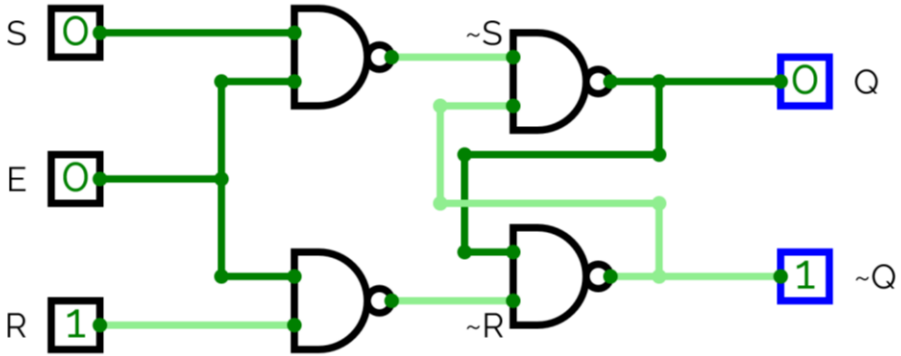


Problema

- Pulso do *clock* é muito longo
  - Circuito está ativo 50% do tempo

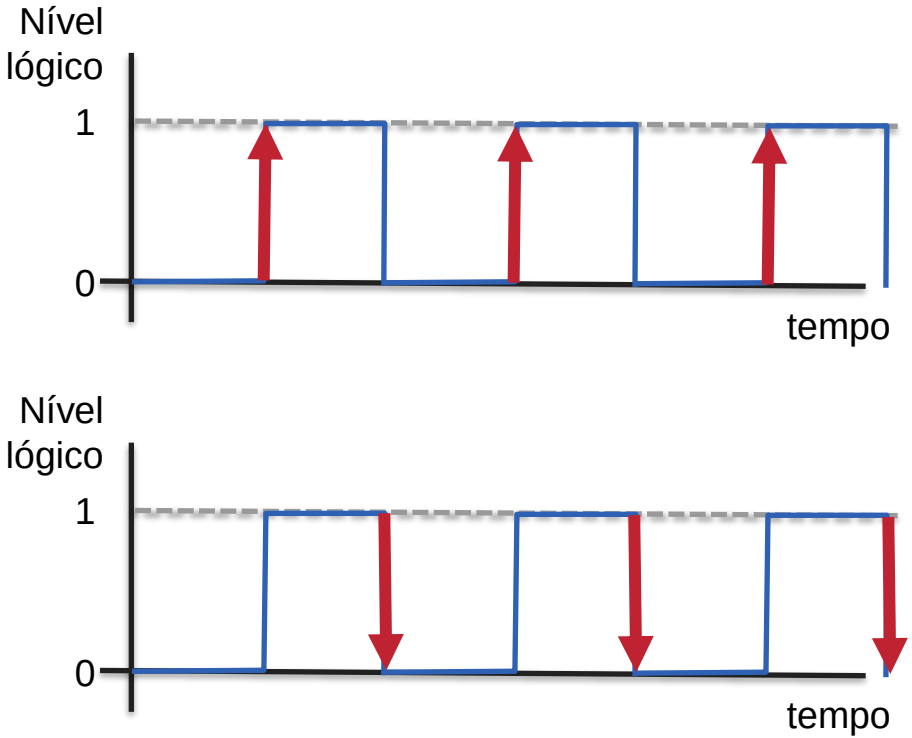
- Quero uma forma de aceitar entradas por apenas uma fração do *clock*
  - Só quando o *clock* **mudar de estado**





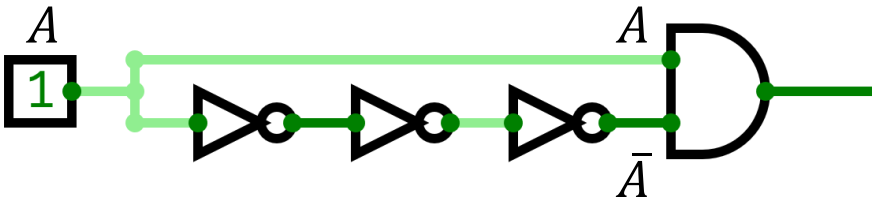
Ativação por subida de *clock*

1. Detectar uma mudança do *clock*
   - 0 -> 1
   - 1 -> 0
2. Permitir UMA alteração de estado
   - Por um período de tempo muito mais curto do que as mudanças nas variáveis de entrada



Detecção de subida de *clock*

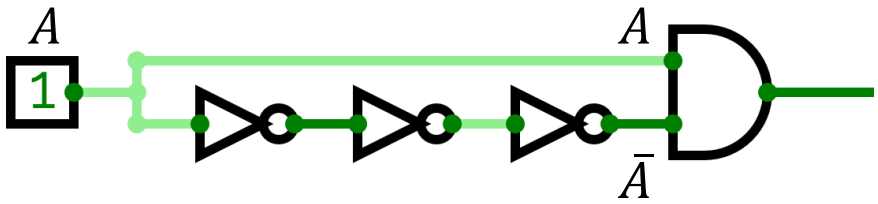
- Portas possuem atraso (*delay*)
- Sinal que passa por muitas portas chega mais atrasado que sinal 'direto'
- AND de variável com a negação dela mesma
  - Variável não tem atraso
  - Negação tem atraso
  - Porta AND vê $A \cdot A$ por fração de segundo
    - $1 \cdot 1$ quando variável vai de 0 para 1

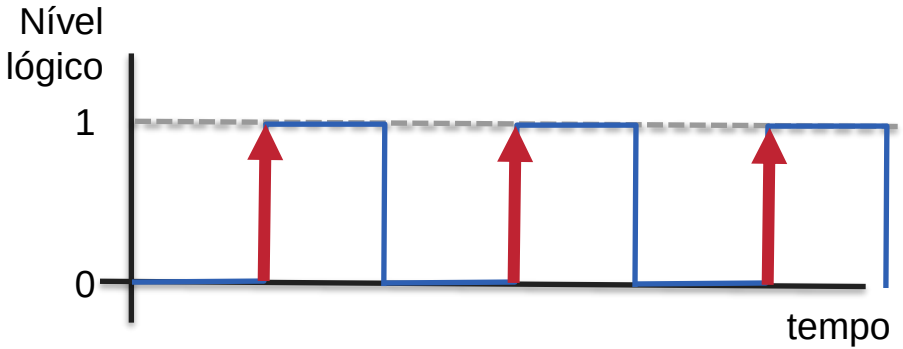


*Rising edge

Detecção de subida de *clock*

- Portas possuem atraso (*delay*)
- Sinal que passa por muitas portas chega mais atrasado que sinal 'direto'
- AND de variável com a negação dela mesma
  - Variável não tem atraso
  - Negação tem atraso
  - Porta AND vê $A \cdot A$ por fração de segundo
    - $1 \cdot 1$ quando variável vai de 0 para 1





Detecção de descida de clock

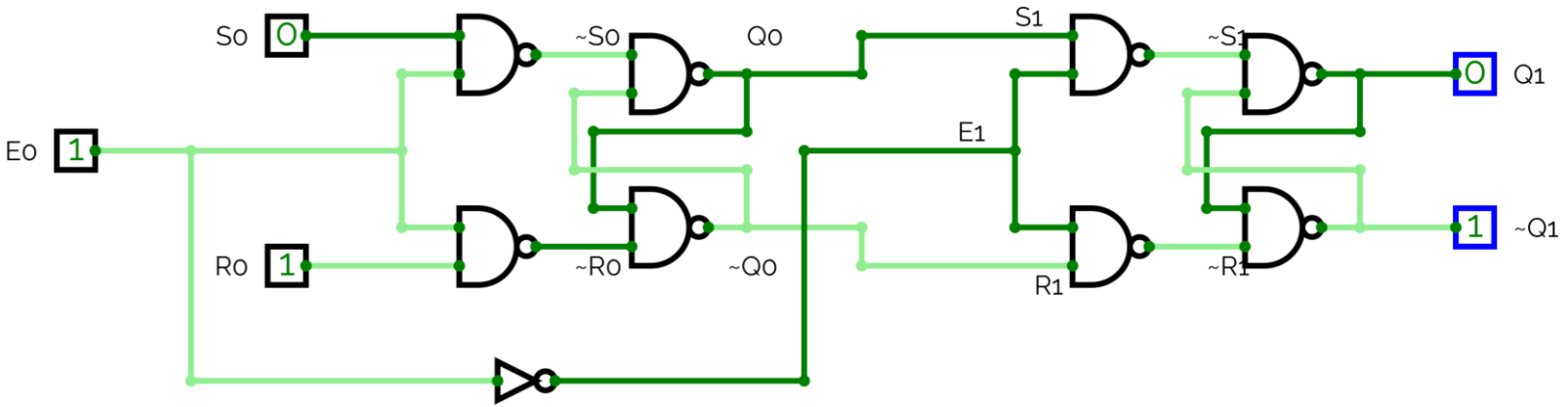
- Solução trivial
  - Colocar mais uma negação em cada entrada do AND

E se....?

- Não confio nos atrasos das portas
- Quero uma solução que não dependa dos atrasos

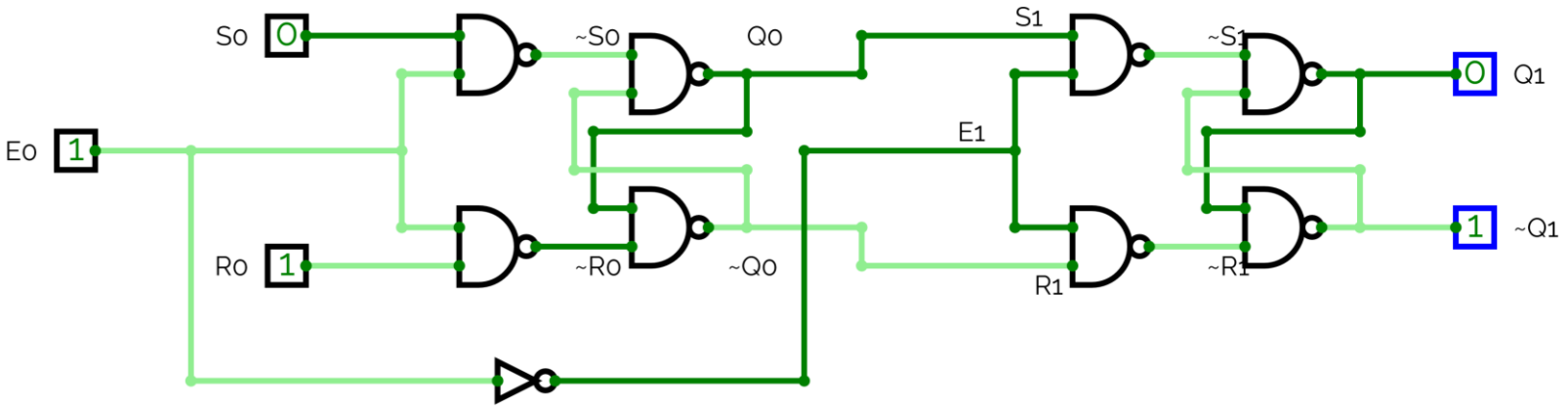
Detecção de descida com *latches*

- Dois *latches* são concatenados
- *Enable* de um é a negação do *enable do outro*



Detecção de descida com *latches*

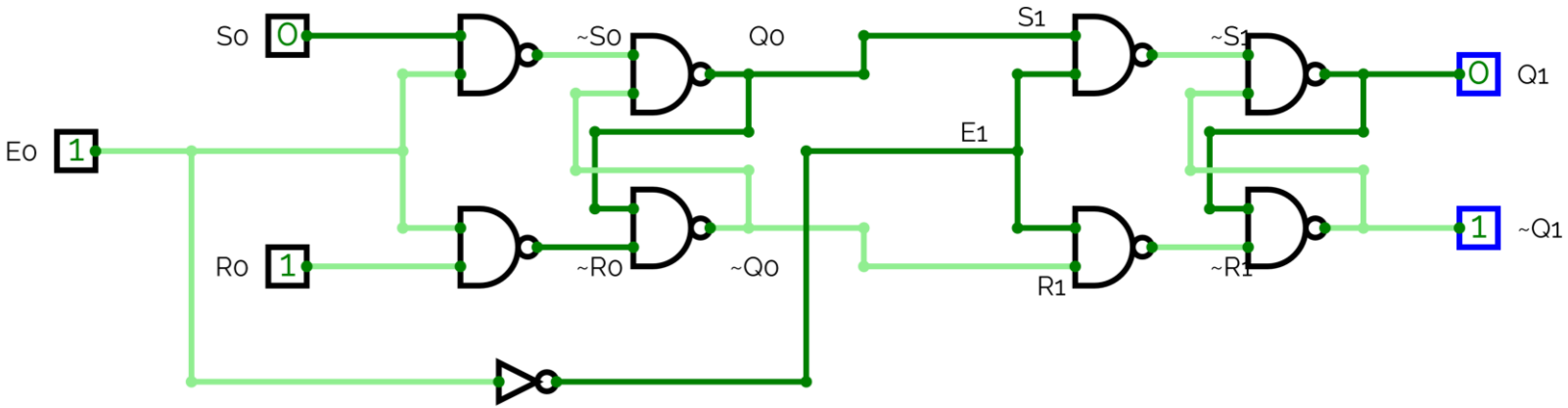
- *Enable* = 1
  - Valor é “carregado” no primeiro
- *Enable* = 0
  - Valor é carregado no segundo
  - Primeiro não pode mudar



Detecção de descida com *latches*

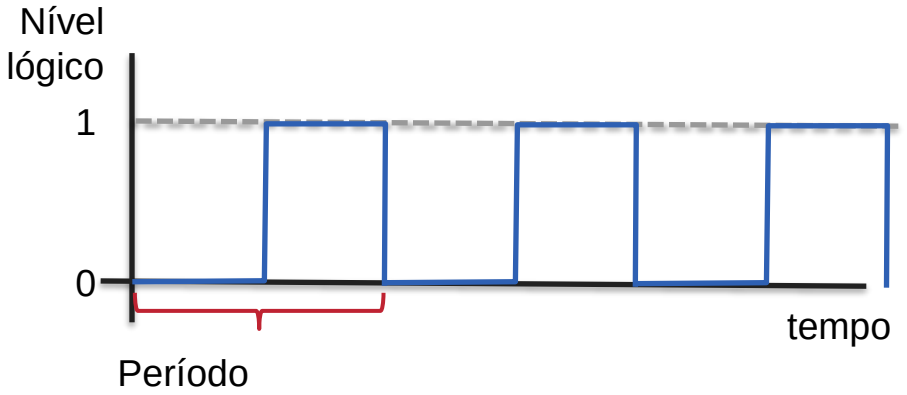
- Enable = 1
  - Valor é “carregado” no primeiro
- Enable = 0
  - Valor é carregado no segundo
  - Primeiro não pode mudar

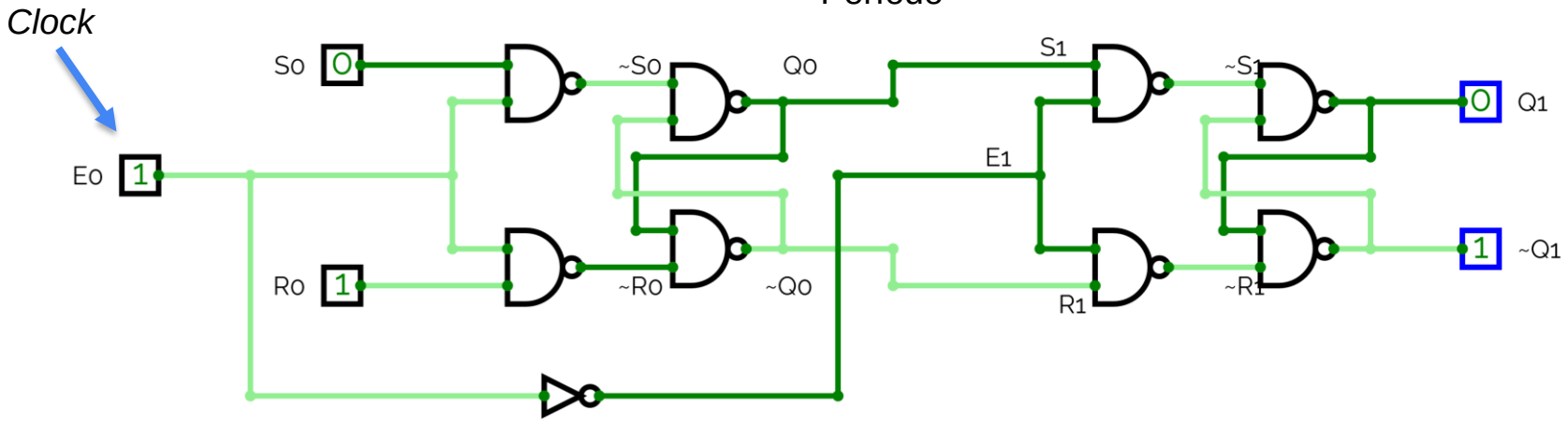
A variação do *enable* de 1 -> 0 permitiu a carga do valor no segundo *latch* (e o aparecimento do valor na saída Q1)



Clock - desambiguação

- Sinal de sincronismo entre módulos
- Entrada de um módulo que permite o sincronismo com os outros módulos

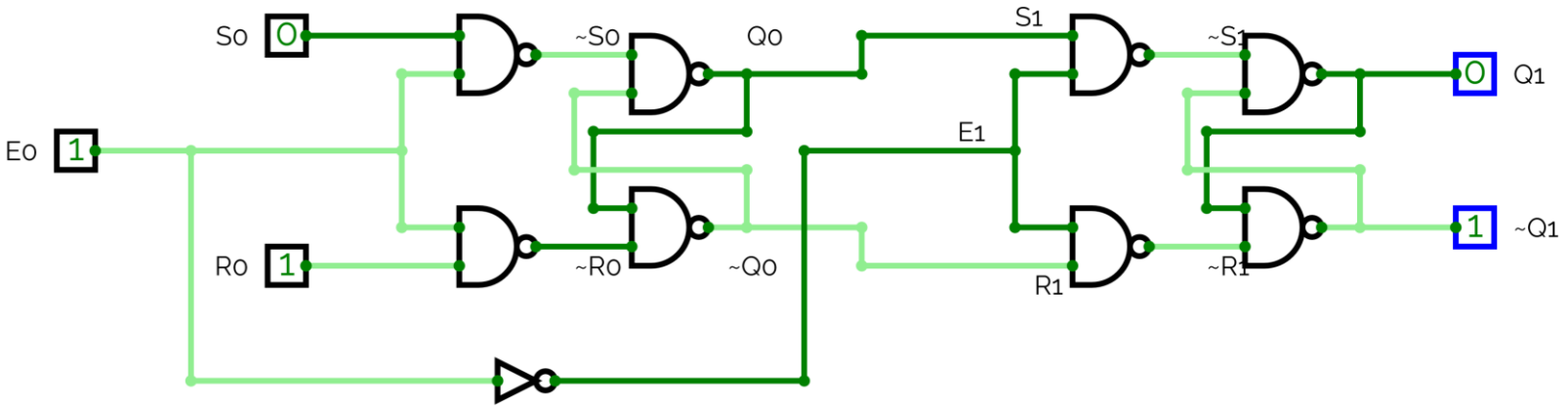




Flip-flops

Latch com clock

- Quando um *latch* é excitável somente na borda de clock (subida ou descida), ele é chamado de *flip-flop*



Conclusões

- Circuitos *latch* dependem de seu estado anterior
  - Uma forma de memória
- Se existe estado anterior, pode haver uma delimitação temporal
  - Antes
  - Agora
- Circuitos *latch* podem ser sincronizados por *clock*
  - Flip-flop

Próxima aula

- Tipos de flip-flop
  - SR
  - JK
  - T
  - D

GTA / UFRJ

GRUPO DE TELEINFORMÁTICA E AUTOMAÇÃO

www.gta.ufrj.br