

# Circuitos Lógicos

Aula 11

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

# Na última aula

- Latch
- Flip-flop
  - SR



# Hoje

- Flip-flops
  - SR
  - D
  - JK
  - T

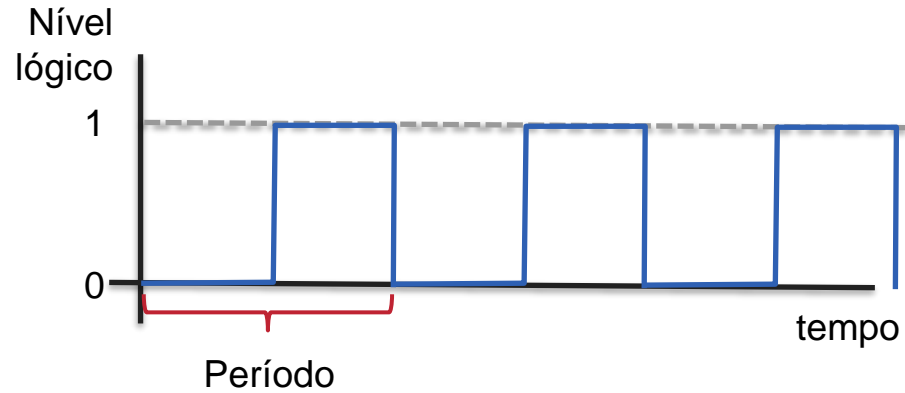


# Clock



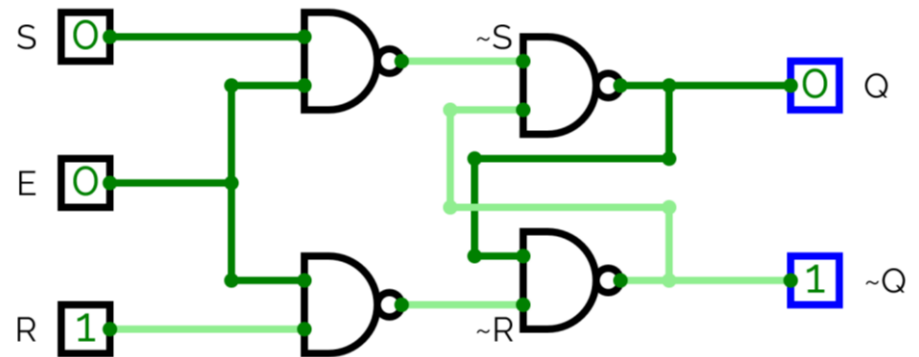
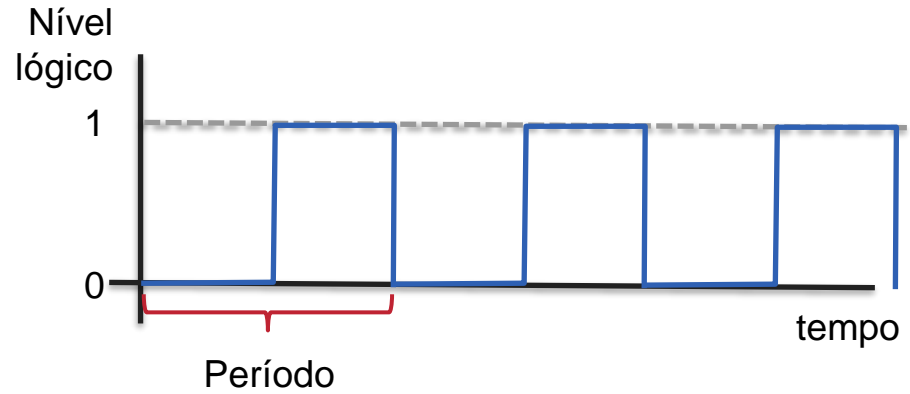
# Clock

- 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)



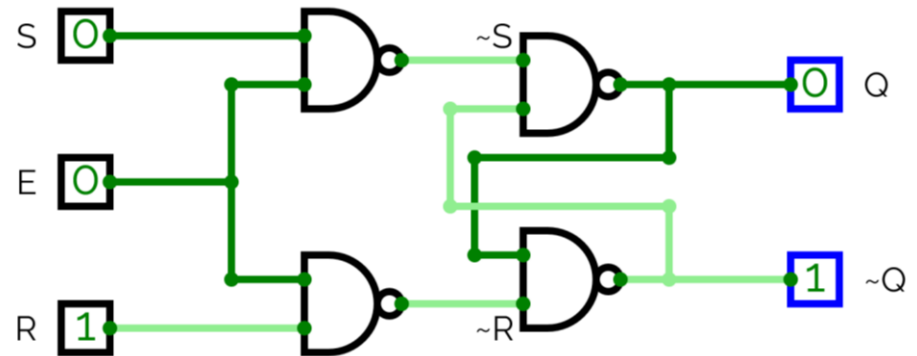
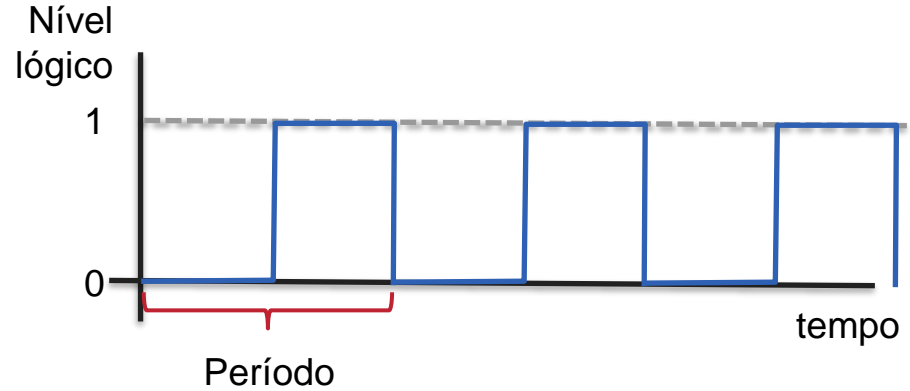
# Clock

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



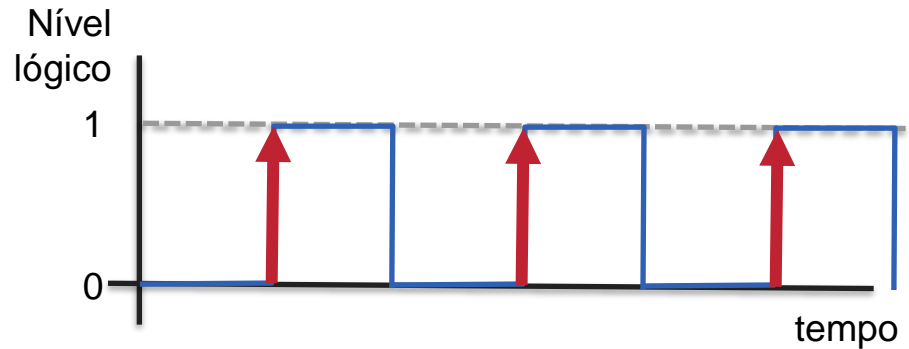
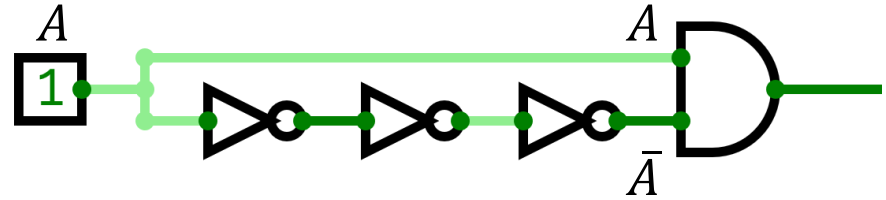
# Problema

- 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



# 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





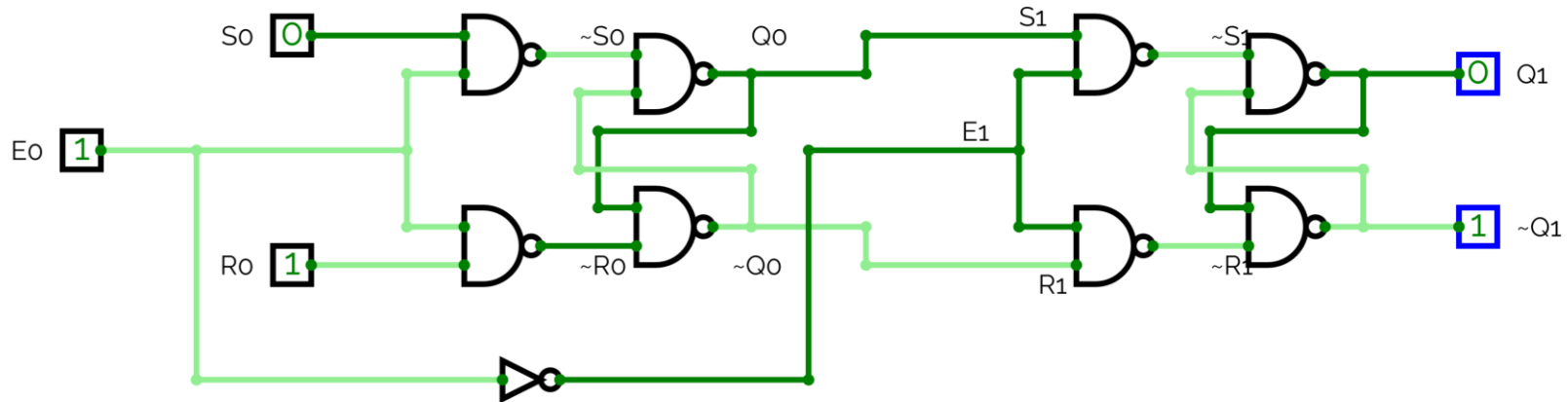
# Detecção de descida de clock

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



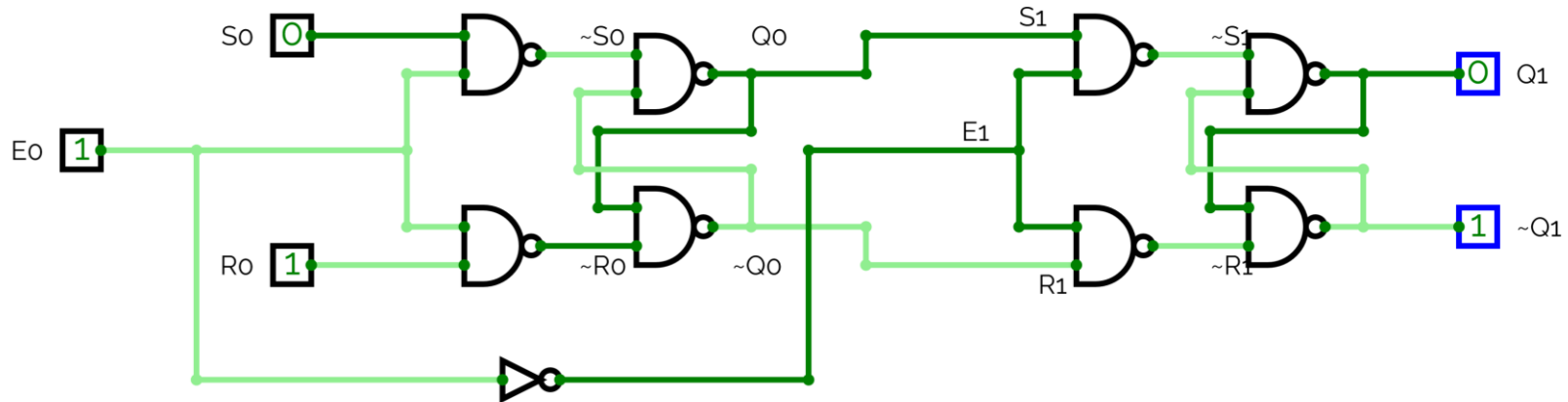
# Detecção de descida com *latches*

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



# Detecção de descida com *latches*

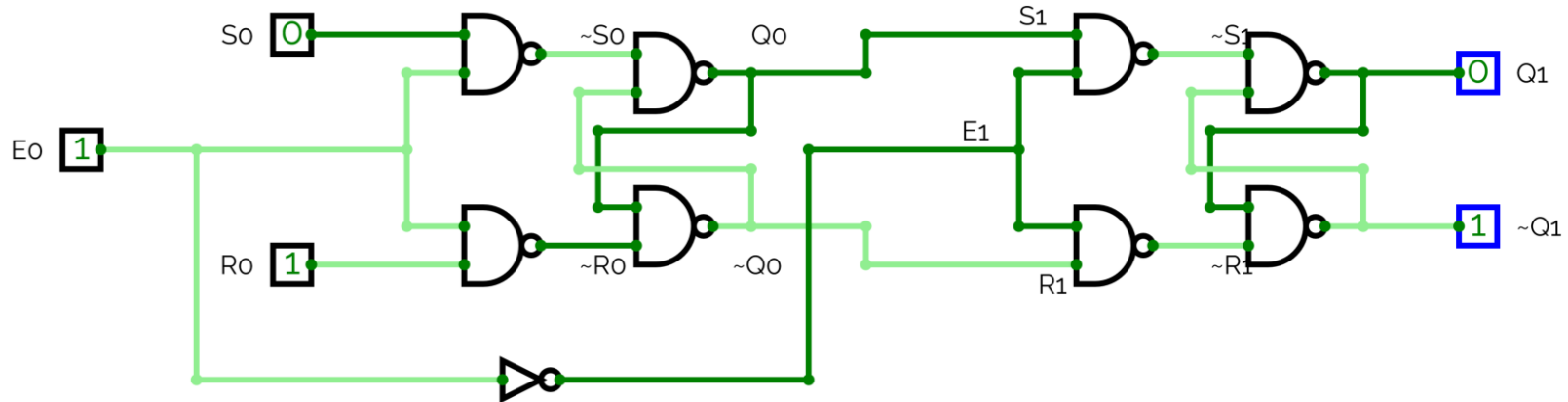
- Enable = 1
  - Valor é “carregado” no primeiro
- Enable = 0
  - Valor é carregado no segundo
  - Primeiro não pode mudar



# Detecção de descida com *latches*

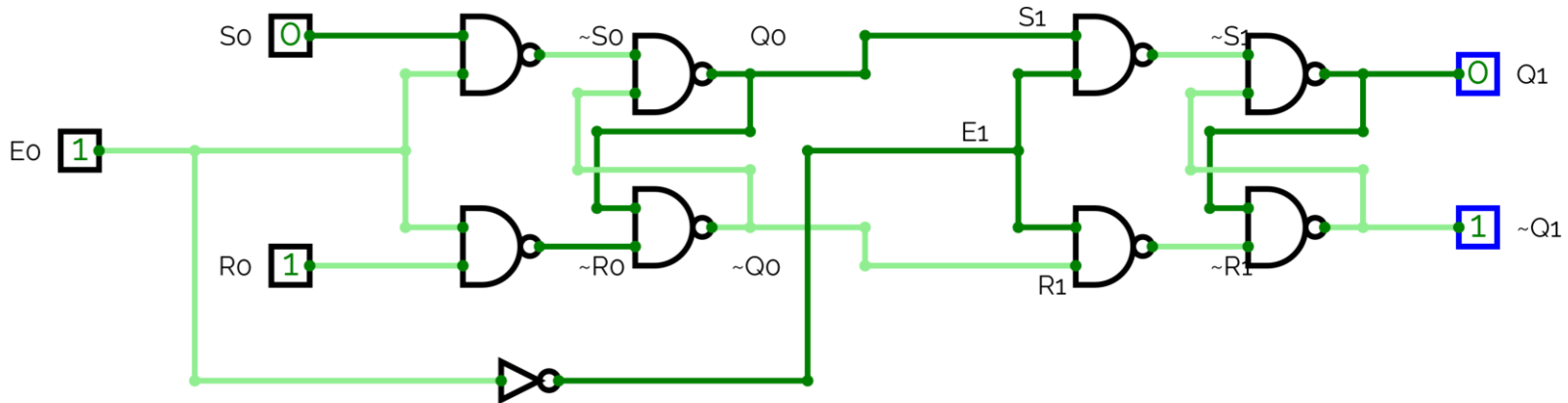
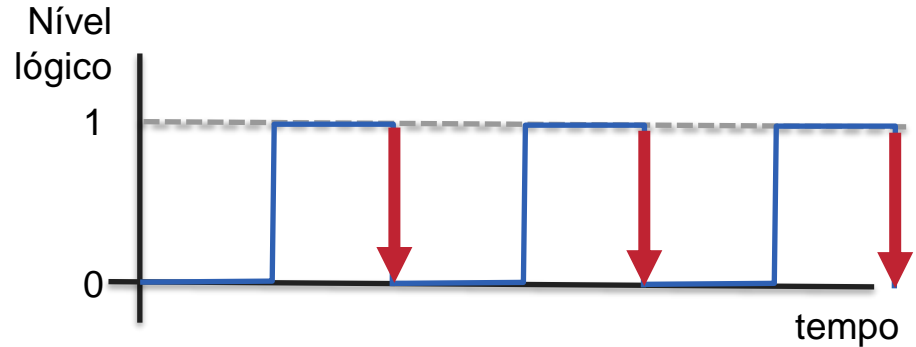
- Enable = 1
  - Valor é “carregado” no primeiro
- Enable = 0
  - Valor é carregado no segundo
  - Primeiro não pode mudar

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



# Detecção de descida com *latches*

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

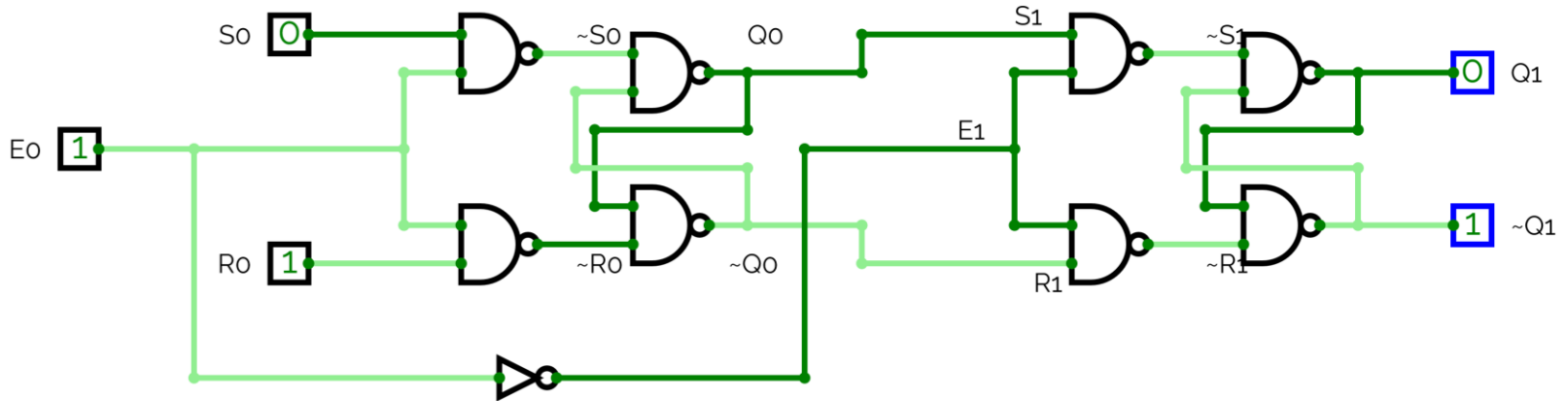


# Flip-flop SR



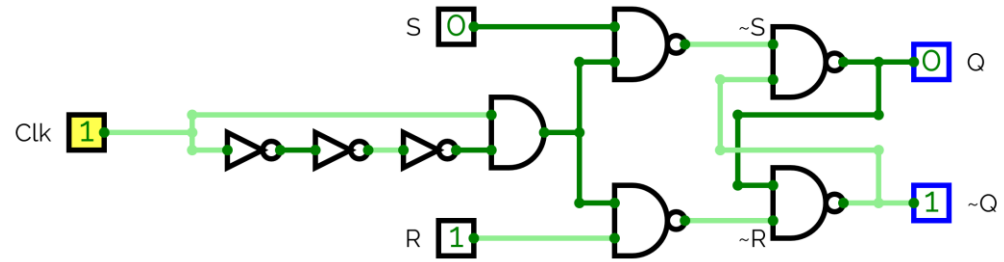
# Latch com clock

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



# Flip-flop SR

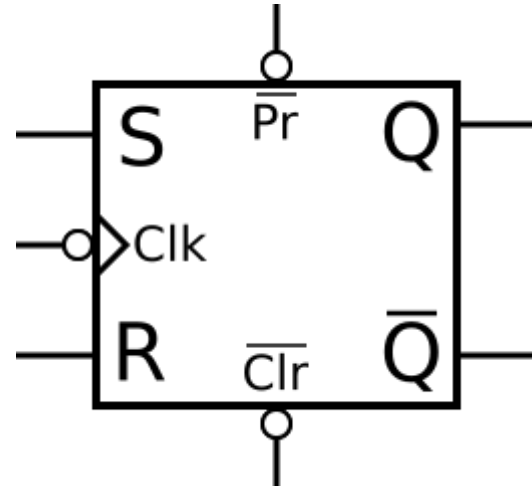
$Clk$	$\overline{Pr}$	$\overline{Clr}$	$R$	$S$	$Q$
Subida	1	1	x	x	$Q_{anterior}$
Descida	1	1	1	0	0
Descida	1	1	0	1	1
Descida	1	1	0	0	$Q_{anterior}$
x	0	1	x	x	1
x	1	0	x	x	0
x	x	x	1	1	Proibido





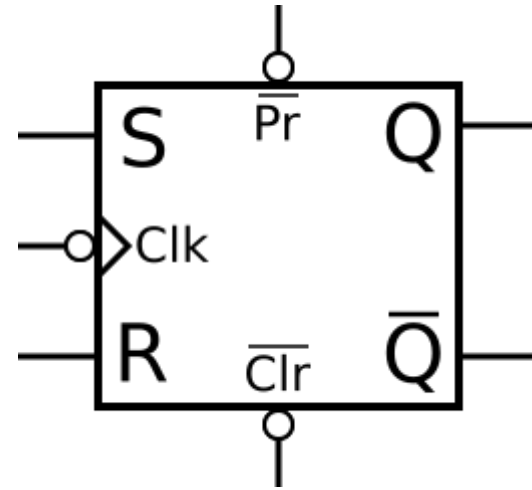
# Módulo flip-flop SR

- Encapsula um FF SR
  - Entrada  $S$
  - Entrada  $R$
  - Entrada *clock*
    - De subida ou descida
  - Entrada *clr*
    - “Clear” ou “reset assíncrono”
  - Entrada *pr*
    - “Preset” ou “set assíncrono”
  - Saída  $Q$
  - Saída  $\bar{Q}$



# Módulo flip-flop SR

$Clk$	$\overline{Pr}$	$\overline{Clr}$	$R$	$S$	$Q$
Subida/0/1	1	1	x	x	$Q_{anterior}$
Descida	1	1	1	0	0
Descida	1	1	0	1	1
Descida	1	1	0	0	$Q_{anterior}$
x	0	1	x	x	1
x	1	0	x	x	0
x	x	x	1	1	Proibido

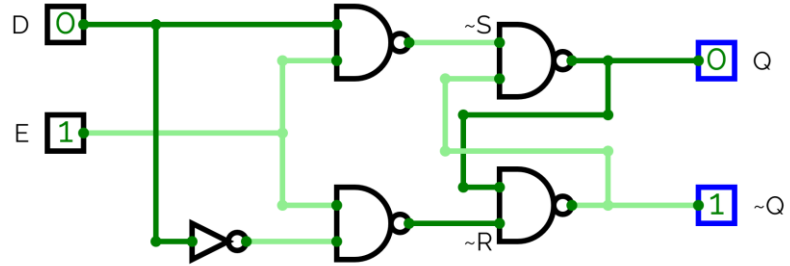


# Flip-flop D



# Latch D

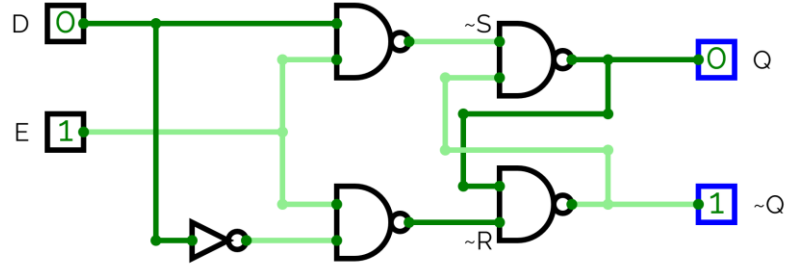
- Latch RS possui 3 entradas
  - Desnecessário quando só queremos guardar uma informação
  - Possui uma configuração “proibida”
- Latch D
  - Guarda informação que chega na entrada D
    - Se *enable* está ativo



# Latch D

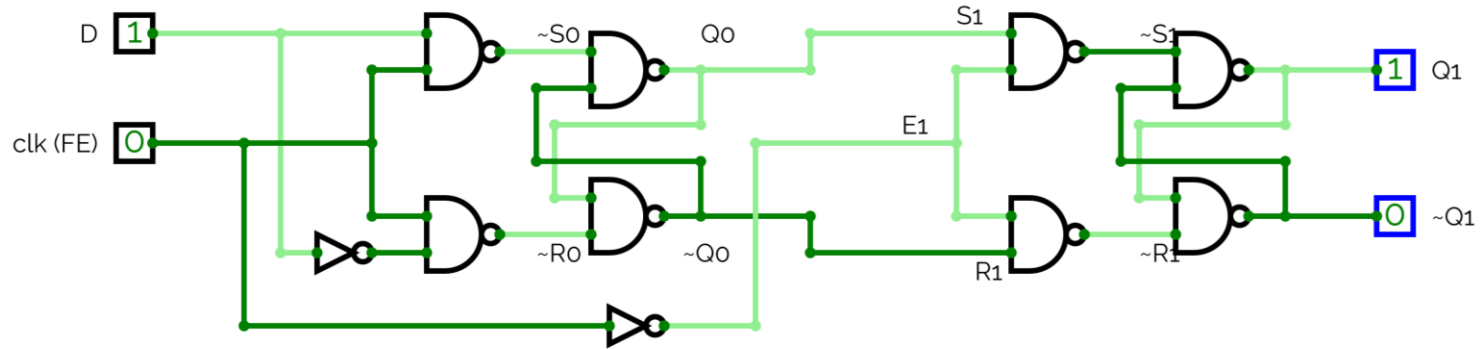
- *Latch D*
  - Guarda informação que chega na entrada D
    - Se *enable* está ativo

<i>E</i>	<i>D</i>	<i>Q</i>
0	x	$Q_{\text{anterior}}$
1	0	0
1	1	1



# Flip-flop D

- Latch D excitado por clock
  - Subida ou descida



# Módulo flip-flop D

- Encapsula um FF D
  - Entrada D
  - Entrada *clock*
    - De subida ou descida
  - Entrada R
    - Reset assíncrono
  - Entrada S
    - Set assíncrono
  - Saída  $Q$
  - Saída  $\bar{Q}$ 
    - Frequentemente omitida

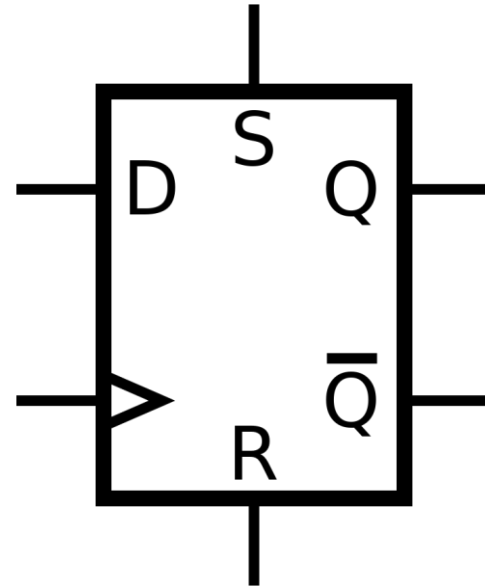


Imagem retirada da wikipedia



# Módulo flip-flop D

<i>Clk</i>	<i>S</i>	<i>R</i>	<i>D</i>	<i>Q</i>
Descida/0/1	0	0	x	$Q_{\text{anterior}}$
Subida	0	0	0	0
Subida	0	0	1	1
x	1	0	x	1
x	0	1	x	0

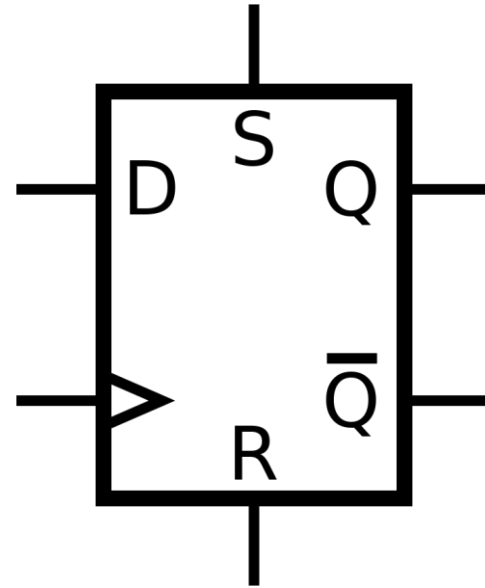


Imagem retirada da wikipedia





# Aplicação: registrador de deslocamento

- FFs D cascadeados
- Entrada síncrona de dados
  - *Clock* pulsa a cada novo bit
- FF 0 recebe da linha de dados
- FF n envia dado ao FF n+1

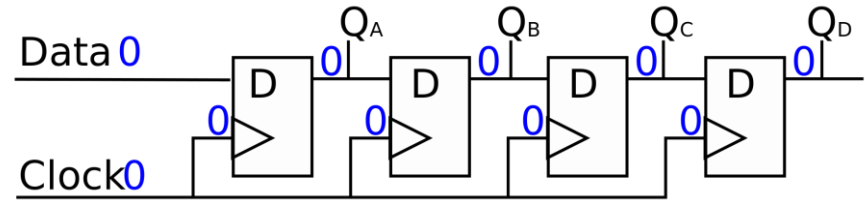


Imagem retirada da wikipedia



# Conversor serial/paralelo

- Dados chegam de maneira serial
- Dados são utilizados de maneira paralela
- Registrador de deslocamento recebe dados seriais
  - “Lembra” de dados anteriores

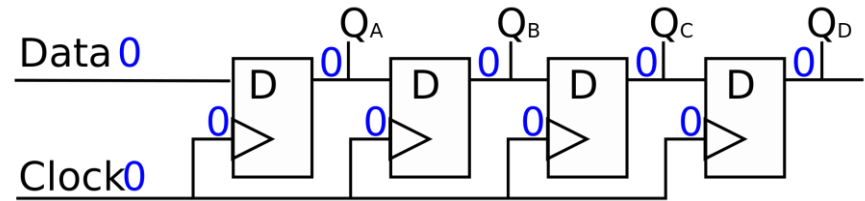


Imagem retirada da wikipedia

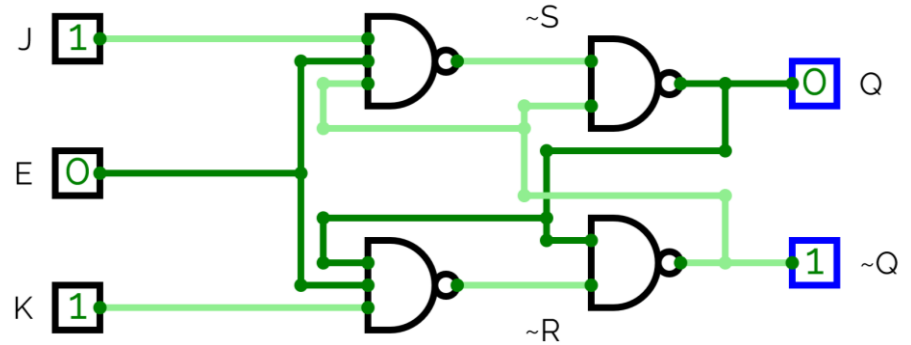


# Flip-flop JK



# Flip-flop JK

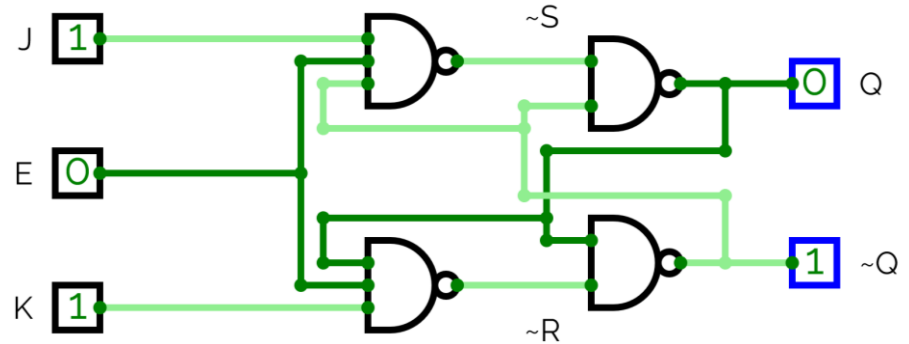
- Latch SR realimentado por  $Q$  e  $\bar{Q}$



# Flip-flop JK

- Latch SR realimentado por  $Q$  e  $\bar{Q}$ 
  - Realimentação no *enable*

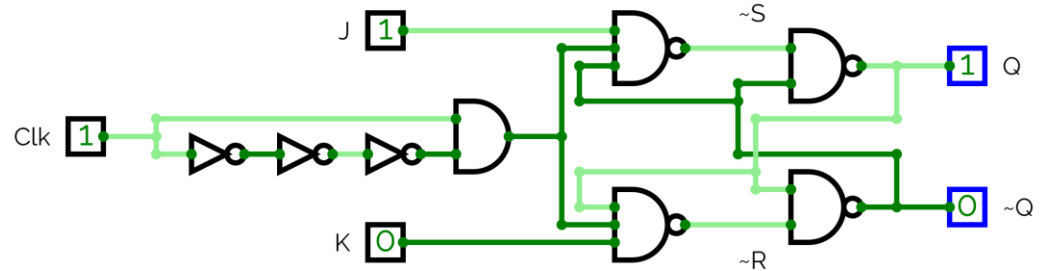
$E$	$J$	$K$	$Q$
0	x	x	$Q_{anterior}$
1	1	0	1
1	0	1	0
1	0	0	$Q_{anterior}$
1	1	1	$\overline{Q_{anterior}}$



# Flip-flop JK

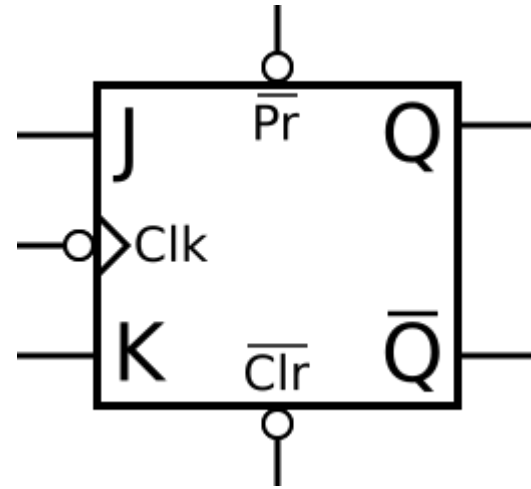
- Estado inverte se  $J=K=1$ 
  - Circuito pode oscilar
- *Clock* de subida ou descida
  - Força apenas uma inversão

<i>Clk</i>	<i>J</i>	<i>K</i>	<i>Q</i>
0	x	x	$Q_{anterior}$
Subida	1	0	1
Subida	0	1	0
Subida	0	0	$Q_{anterior}$
Subida	1	1	$\overline{Q_{anterior}}$



# Módulo flip-flop JK

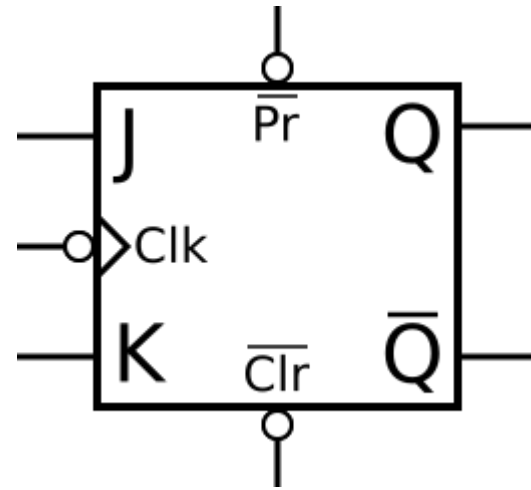
- Encapsula as funcionalidades do FF JK
  - Entrada J
  - Entrada K
  - Entrada *clock*
    - De subida ou descida
  - Entrada clr
    - “Clear” ou “reset assíncrono”
  - Entrada *pr*
    - “Preset” ou “set assíncrono”
  - Saída  $Q$
  - Saída  $\bar{Q}$



# Módulo flip-flop JK

- Encapsula todas as funcionalidades do FF JK

<i>Clk</i>	$\overline{Pr}$	$\overline{Clr}$	<i>J</i>	<i>K</i>	<i>Q</i>
Subida/0/1	1	1	x	x	$Q_{anterior}$
Descida	1	1	1	0	1
Descida	1	1	0	1	0
Descida	1	1	0	0	$Q_{anterior}$
Descida	1	1	1	1	$\overline{Q_{anterior}}$
x	0	1	x	x	1
x	1	0	x	x	0





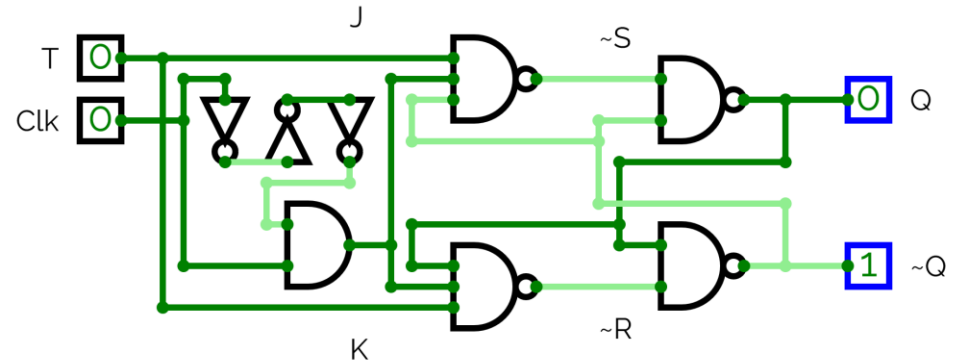
# Flip-flop T



# Flip-flop T (*toggle*\*)

- FF JK cujas entradas são conectadas

<i>Clk</i>	<i>T</i>	<i>Q</i>
0	x	$Q_{anterior}$
Subida	0	$Q_{anterior}$
Subida	1	$\overline{Q_{anterior}}$



“Toggle” significa “alternar”

# Módulo Flip-flop T

- FF JK cujas entradas são conectadas

<i>Clk</i>	<i>T</i>	<i>Q</i>
0	x	$Q_{anterior}$
Subida	0	$Q_{anterior}$
Subida	1	$\overline{Q_{anterior}}$

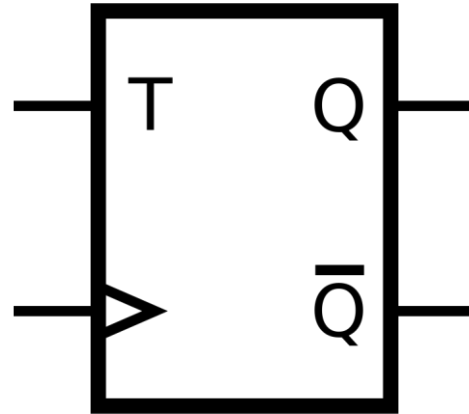


Imagem retirada da wikipedia



# Conclusões

- Lembram um bit
  - Setar (bit = 1)
  - Resetar (bit = 0)
- Trabalham de maneira síncrona
  - Entrada *clock* decide quando valor pode ser alterado
- Possuem entradas assíncronas
  - *Preset*
  - *Reset/clear*

Lembrança permite nos construir circuitos sequenciais

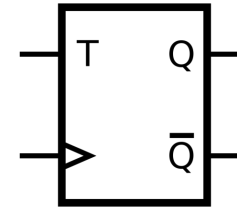
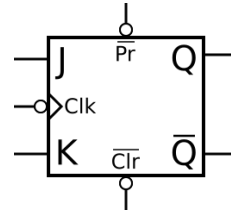
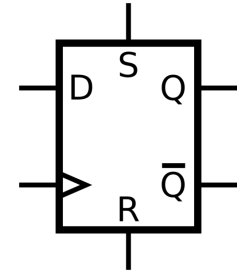
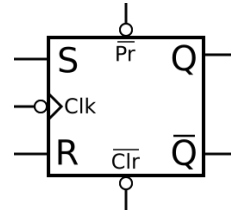


Imagem dos FFs D e T retiradas da wikipedia

# Próxima aula

- Divisor de frequência
- Contador
- Estado de um circuito sequencial
- Máquina de estados





**GTA / UFRJ**

GRUPO DE TELEINFORMÁTICA E AUTOMAÇÃO

[www.gta.ufrj.br](http://www.gta.ufrj.br)