

Sistemas Reconfiguráveis

Eng. de Computação

Profs. Francisco Garcia e Antônio Hamilton Magalhães

Unidade 0 – Revisão de conceitos básicos em Sistemas Digitais

Revisão de conceitos básicos

Sistema de numeração decimal

- Posicional
- Algarismos de 0 a 9
- Base 10

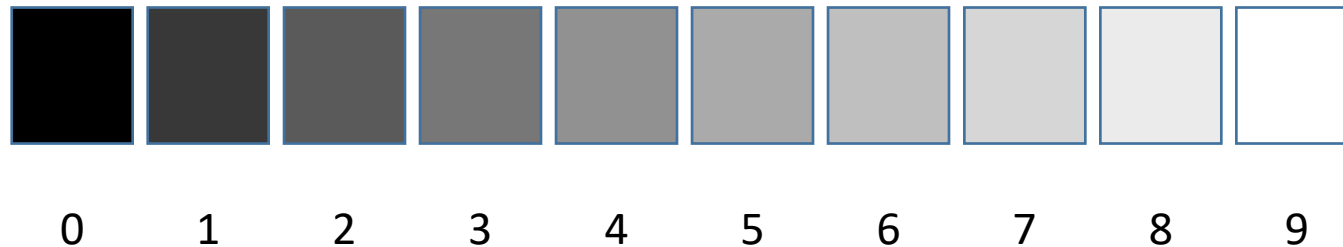
3 3 3 , 3

$3 = 3 \times 10^0$

$30 = 3 \times 10^1$

$300 = 3 \times 10^2$

$0,3 = 3 \times 10^{-1}$



Revisão de conceitos básicos

Sistema de numeração binário

- Posicional
- Algarismos 0 e 1
- Base 2

$$\begin{array}{c} 101,01_2 \\ \hline = 1 \times 2^0 + 0 \times 2^1 + 1 \times 2^2 + 0 \times 2^{-1} + 1 \times 2^{-2} = 5 + 0,25 \\ = 5,25_{10} \end{array}$$



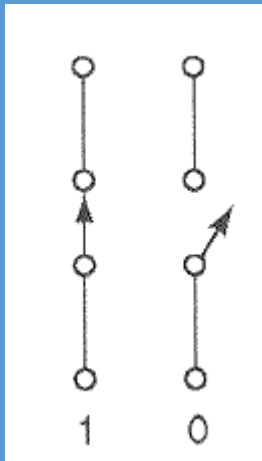
0

1

Revisão de conceitos básicos

Estados lógicos em sistemas digitais

Chave

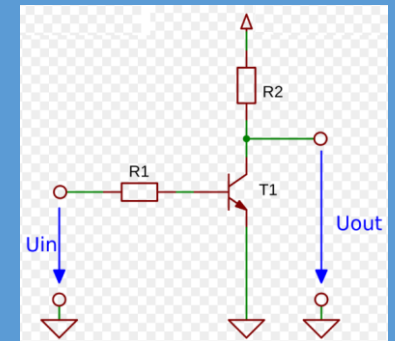


Lâmpada



Acesa = 1
Apagada=0

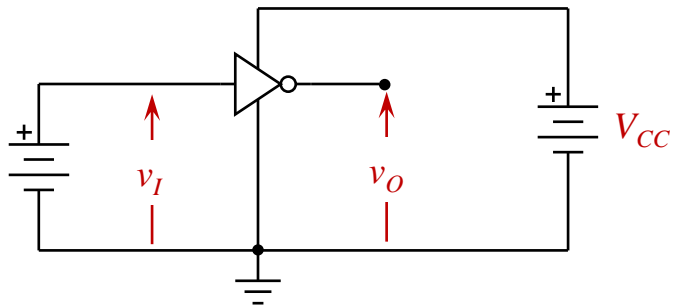
Transistor



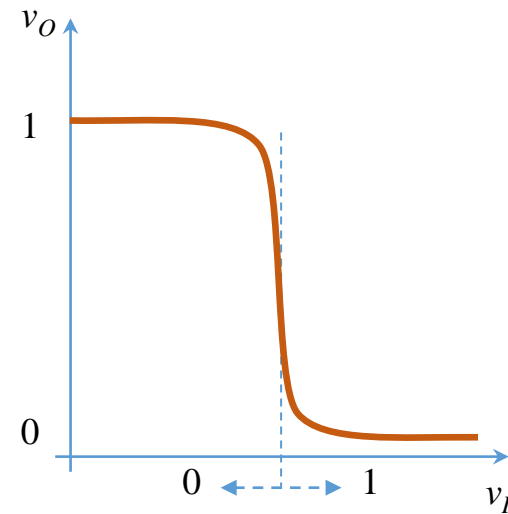
Saturado = 0
Cortado = 1

Revisão de conceitos básicos

Experimento com um inversor lógico



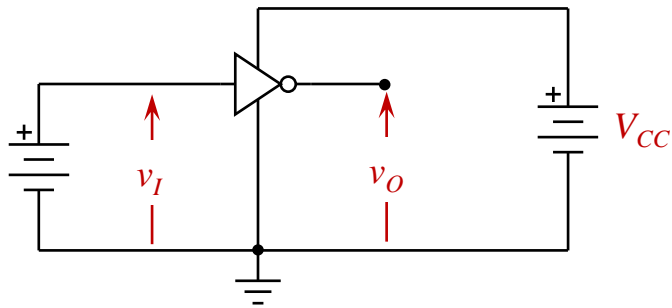
Circuito



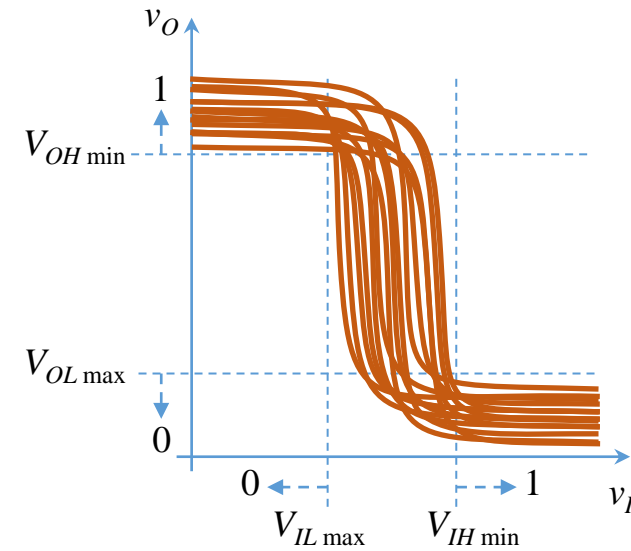
Resultado do experimento

Revisão de conceitos básicos

Experimento com um inversor lógico



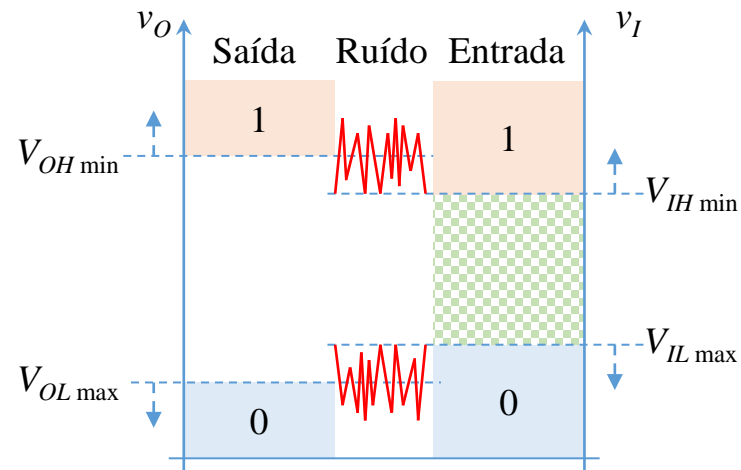
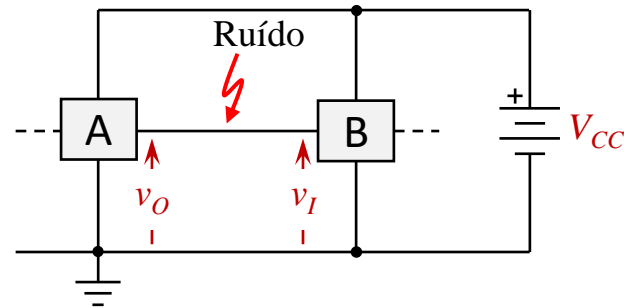
Circuito



Resultado do experimento em várias situações

Revisão de conceitos básicos

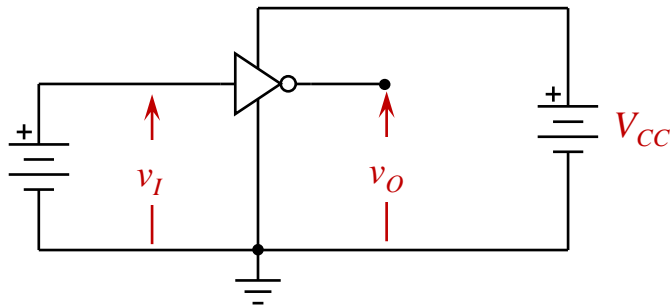
Níveis lógicos e margem de ruído



Padrão	V_{CC}	$V_{IL\max}$	$V_{IH\min}$	$V_{OL\max}$	$V_{OH\min}$
TTL	5 V	0,8 V	2,0 V	0,4 V	2,5 V
CMOS	5 a 15 V	$0,3 \cdot V_{CC}$	$0,7 \cdot V_{CC}$	0,05 V	$V_{CC} - 0,05 \text{ V}$
LVTTL	3,3 V	0,8 V	1,7 V	0,45 V	2,4 V
LVC MOS	3,3 V	0,8 V	1,7 V	0,2 V	3,1 V

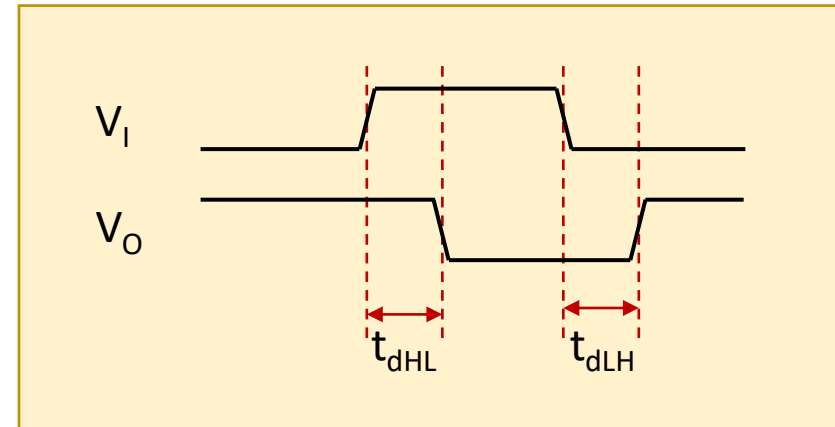
Revisão de conceitos básicos

Experimento com um inversor lógico – atraso de propagação



Todo circuito eletrônico apresenta atraso de propagação.

Dependendo da tecnologia de fabricação, esse atraso pode ser de alguns poucos nanosegundos ou menos de 1 ns



t_{dHL} = tempo de atraso quando a saída vai de alto (H) para baixo (L)

t_{dLH} = tempo de atraso quando a saída vai de baixo (L) para alto (H)

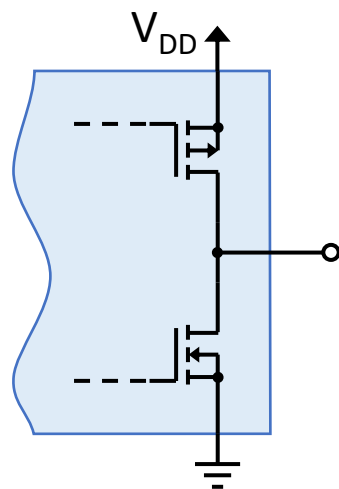
Em alguns circuitos, $t_{dHL} = t_{dLH}$. Nesse caso, o atraso é chamado apenas de t_d .

Às vezes é usada a letra **p** (*propagation*) ao invés de **d** (*delay*)

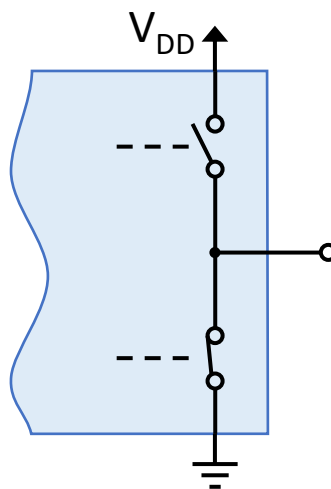
Revisão – *tri-state*

Lógica de dois estados

Estágio de saída típico de um circuito digital

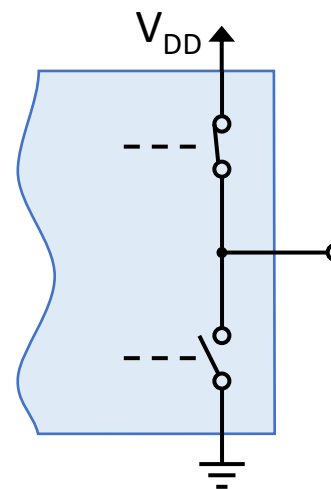


Circuito equivalente para nível lógico '0'



Tensão de saída próxima de 0V

Circuito equivalente para nível lógico '1'



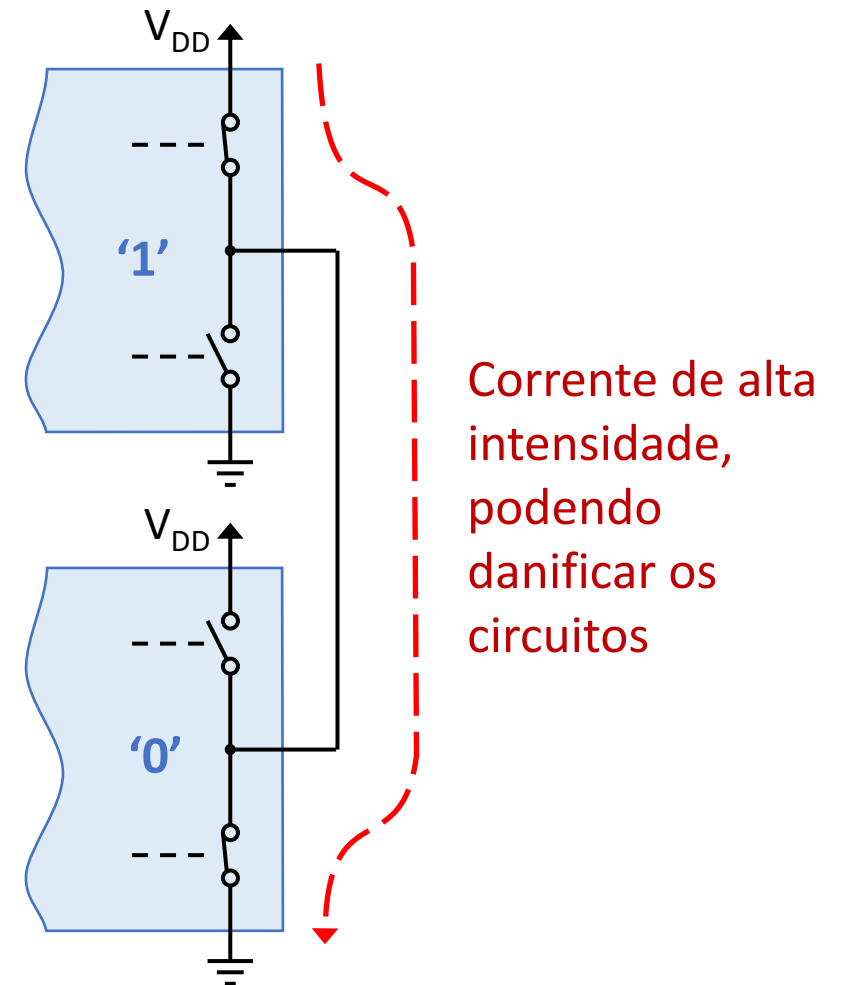
Tensão de saída próxima de V_{DD}

V_{DD} é o nome que representa, usualmente, a fonte de alimentação de circuitos digitais com tecnologia C-MOS. Pode ser +5V, +3,3V ou, em alguns casos, valores mais baixos, como +2,5V, +1,8V ou +1,2V. Não há um padrão.

Revisão – *tri-state*

Lógica de dois estados

Em circuitos com lógica de dois estados, não é permitida a ligação da saída de um circuito à saída de outro, pois isso leva a um conflito, que pode danificar os circuitos.

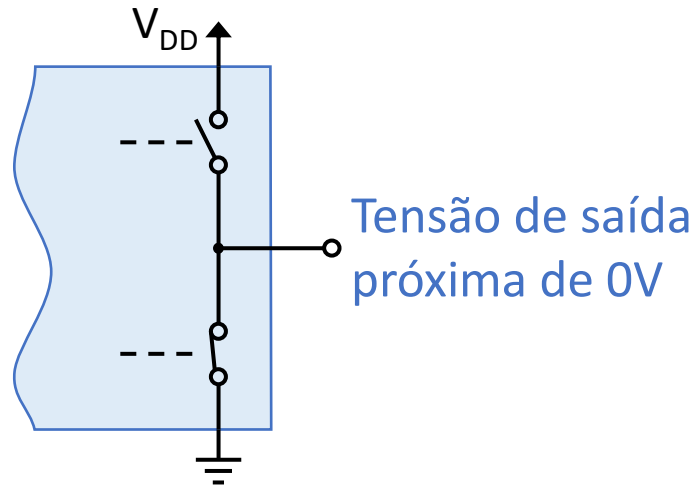


Revisão – *tri-state*

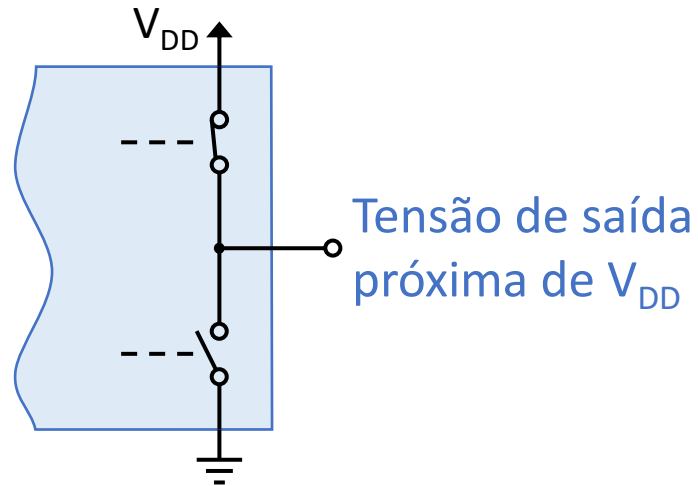
Lógica de três estados (*tri-state*)

Alguns circuitos trabalham com três estados possíveis na saída. Além do '0' e '1', existe um terceiro estado de alta impedância, chamado, em inglês, de *high-Z* ou, abreviadamente, 'Z'

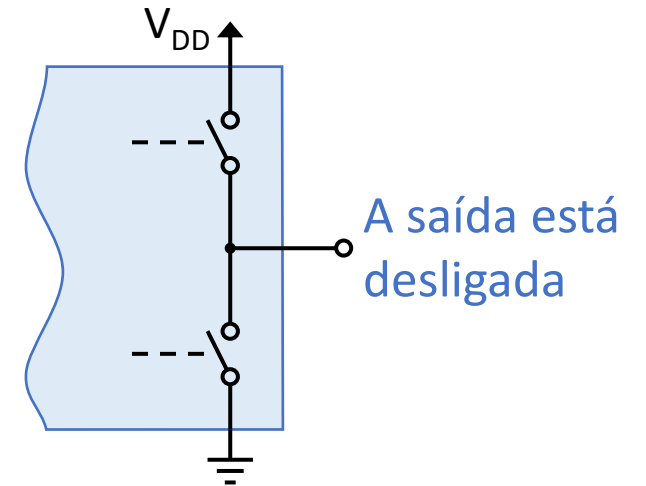
Circuito equivalente
para '0'



Circuito equivalente
para '1'



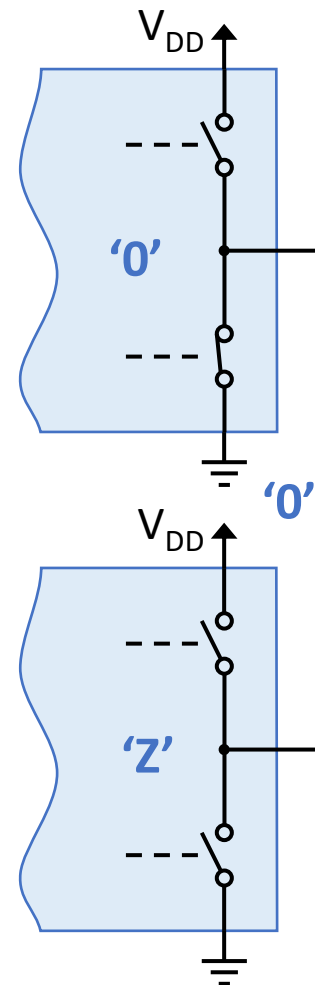
Circuito equivalente
para 'Z'



Revisão – *tri-state*

Lógica de três estados (*tri-state*)

No estado de alta impedância ('Z'), a tensão na saída do pino fica indefinida. O nível de tensão nessa condição pode ser definido por outra saída ativa ('0' ou '1') conectada ao mesmo do ponto do circuito ou com uso de resistores de *pull-up* e *pull-down* conectados nesse ponto.

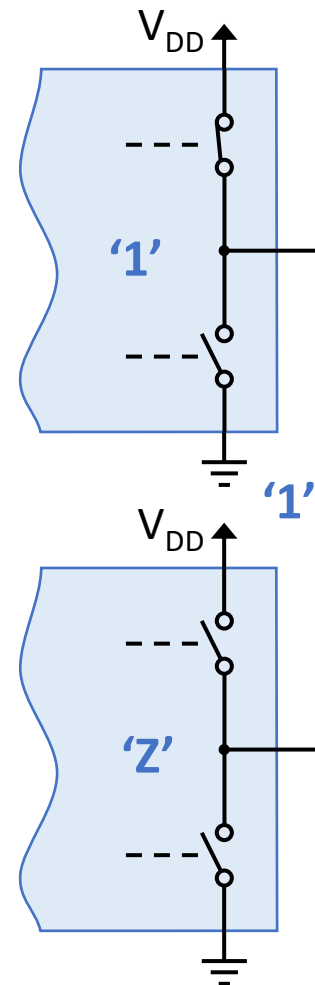


Quando uma saída em 'Z' está conectada a uma saída em nível '0', o barramento fica em '0'

Revisão – *tri-state*

Lógica de três estados (*tri-state*)

No estado de alta impedância ('Z'), a tensão na saída do pino fica indefinida. O nível de tensão nessa condição pode ser definido por outra saída ativa ('0' ou '1') conectada ao mesmo do ponto do circuito ou com uso de resistores de *pull-up* e *pull-down* conectados nesse ponto.

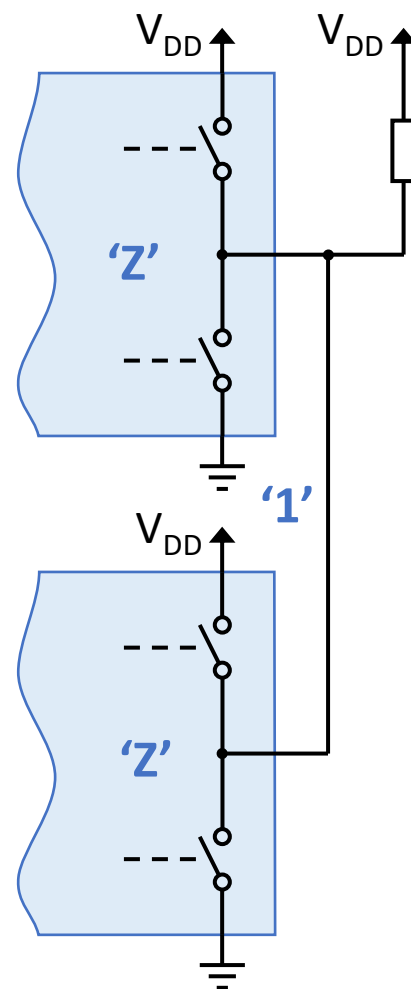


Quando uma saída em 'Z' está conectada a uma saída em nível '1', o barramento fica em '1'

Revisão – *tri-state*

Lógica de três estados (*tri-state*)

No estado de alta impedância ('Z'), a tensão na saída do pino fica indefinida. O nível de tensão nessa condição pode ser definido por outra saída ativa ('0' ou '1') conectada ao mesmo do ponto do circuito ou com uso de resistores de *pull-up* e *pull-down* conectados nesse ponto.



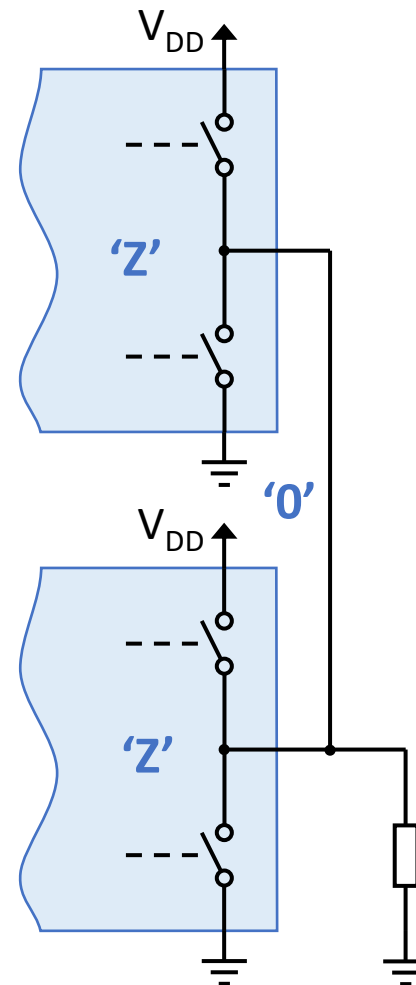
Um resistor ligado à alimentação V_{DD} (chamado resistor de *pull-up*) faz com a tensão no barramento fique alta ('1'), quando todas as saídas conectadas ao barramento estão em 'Z'.

No entanto, se há uma saída habilitada ('0' ou '1'), prevalece o nível dessa saída.

Revisão – *tri-state*

Lógica de três estados (*tri-state*)


No estado de alta impedância ('Z'), a tensão na saída do pino fica indefinida. O nível de tensão nessa condição pode ser definido por outra saída ativa ('0' ou '1') conectada ao mesmo do ponto do circuito ou com uso de resistores de *pull-up* e *pull-down* conectados nesse ponto.



Um resistor ligado à terra (chamado resistor de ***pull-down***) faz com a tensão no barramento fique baixa ('0'), quando todas as saídas conectadas ao barramento estão em 'Z'.

No entanto, se há uma saída habilitada ('0' ou '1'), prevalece o nível dessa saída.

Revisão – *tri-state*



Lógica de três estados (*tri-state*)

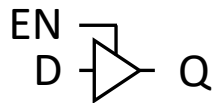
Para controlar o terceiro estado, existe um sinal de habilitação (*enable*), que habilita a saída ('0' ou '1') ou desabilita ('Z').

A seguir são mostrados os símbolos e as tabelas verdade de alguns circuitos que trabalham com lógica *tri-state*. As entradas desses circuitos continuam funcionando com lógica de dois estados. Apenas a saída é *tri-state*.

Revisão – *tri-state*

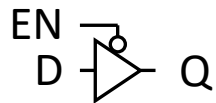
Lógica de três estados (*tri-state*)

Buffer tri-state
com *enable* ativo
em nível alto



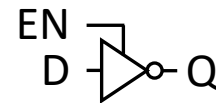
Entradas		Saída
EN	D	Q
0	X	Z
1	0	0
1	1	1

Buffer tri-state
com *enable* ativo
em nível baixo



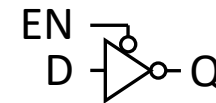
Entradas		Saída
EN	D	Q
1	X	Z
0	0	0
0	1	1

Inversor tri-state
com *enable* ativo
em nível alto



Entradas		Saída
EN	D	Q
0	X	Z
1	0	1
1	1	0

Inversor tri-state
com *enable* ativo
em nível baixo

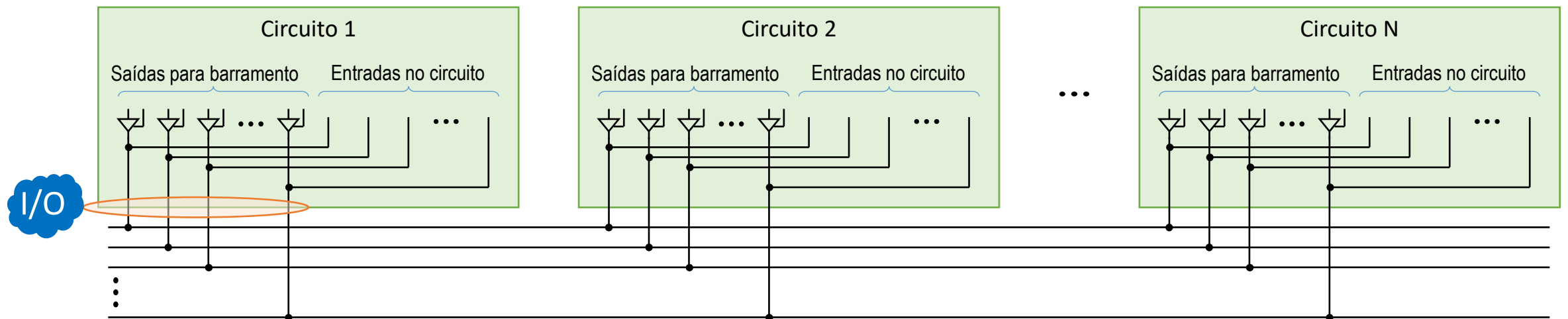


Entradas		Saída
EN	D	Q
1	X	Z
0	0	1
0	1	0

Revisão – *tri-state*

Lógica de três estados (*tri-state*) - Barramentos

O principal uso de circuitos *tri-state* é em **barramentos** (*bus*), onde vários circuitos compartilham o mesmo conjunto de ligações.

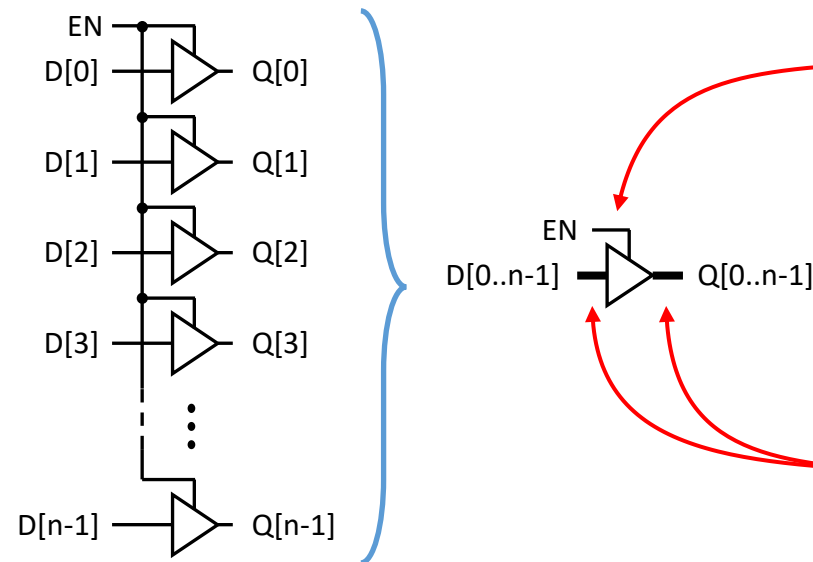


Essa ligação não funciona com lógica de dois estados, pois várias saídas estão ligadas entre si. Para funcionar adequadamente, apenas uma saída pode estar habilitada a cada instante.

Revisão – *tri-state*

Lógica de três estados (*tri-state*) - Barramentos

Um conjunto de n buffers tri-state pode ser desenhado de uma forma mais simplificada:



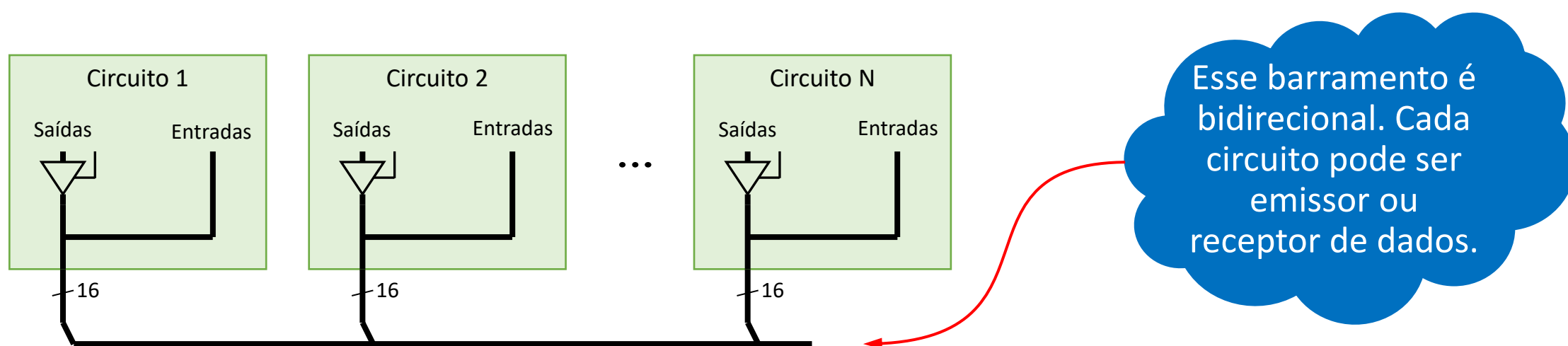
Como o *enable* é comum a todos os *buffers*, é representado com um traço fino, pois trata-se de um único sinal

O traço grosso indica um conjunto de ligações (barramento).

Revisão – *tri-state*

Lógica de três estados (*tri-state*) - Barramentos

O diagrama esquemático anterior pode ser simplificado como mostrado abaixo.

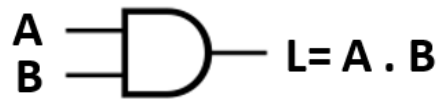


O tracinho e número próximo ao barramento indicam o número de ligações (vias)
Nesse exemplo, 16 vias

Revisão – portas lógicas

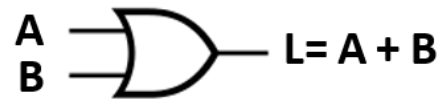
Portas lógicas

Porta AND



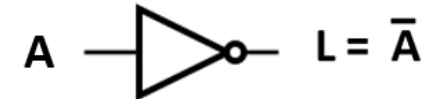
Entradas		Saída
A	B	L
0	0	0
0	1	0
1	0	0
1	1	1

Porta OR



Entradas		Saída
A	B	L
0	0	0
0	1	1
1	0	1
1	1	1

Porta NOT
(inversor)



Entrada	Saída
A	L
0	1
1	0

Obs.:

Uma tabela verdade relaciona os valores das variáveis de entrada e os correspondentes valores das variáveis de saída.

Revisão – portas lógicas

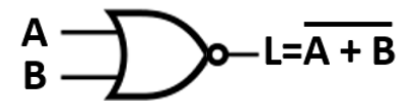
Portas lógicas

Porta NAND



Entradas		Saída
A	B	L
0	0	1
0	1	1
1	0	1
1	1	0

Porta NOR

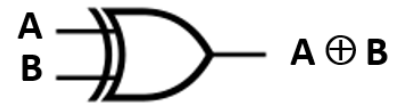


Entradas		Saída
A	B	L
0	0	1
0	1	0
1	0	0
1	1	0

Revisão – portas lógicas

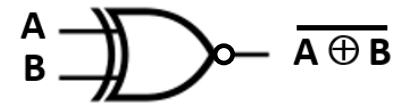
Portas lógicas

Porta XOR



Entradas		Saída
A	B	L
0	0	0
0	1	1
1	0	1
1	1	0

Porta XNOR

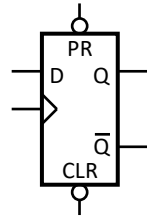


Entradas		Saída
A	B	L
0	0	1
0	1	0
1	0	0
1	1	1

Revisão – flip-flops

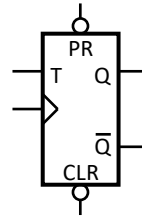
Flip-flops

Flip-flop tipo D



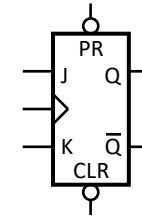
Entradas				Saídas	
$\overline{\text{PR}}$	$\overline{\text{CLR}}$	D	CLK	Q	$\overline{\text{Q}}$
0	0	X	X	*	*
0	1	X	X	1	0
1	0	X	X	0	1
1	1	0	\uparrow	0	1
1	1	1	\uparrow	1	0
1	1	X	0, 1, \downarrow	Qa	$\overline{\text{Qa}}$

Flip-flop tipo T



Entradas				Saídas	
$\overline{\text{PR}}$	$\overline{\text{CLR}}$	T	CLK	Q	$\overline{\text{Q}}$
0	0	X	X	*	*
0	1	X	X	1	0
1	0	X	X	0	1
1	1	0	\uparrow	Qa	$\overline{\text{Qa}}$
1	1	1	\uparrow	$\overline{\text{Qa}}$	Qa
1	1	X	0, 1, \downarrow	Qa	$\overline{\text{Qa}}$

Flip-flop tipo JK

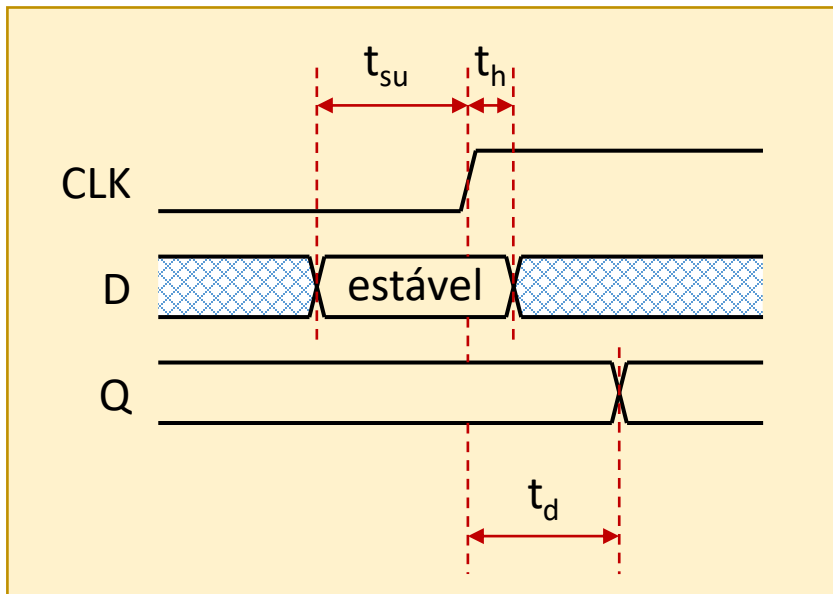
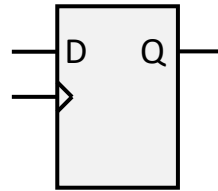


Entradas					Saídas	
$\overline{\text{PR}}$	$\overline{\text{CLR}}$	J	K	CLK	Q	$\overline{\text{Q}}$
0	0	X	X	X	*	*
0	1	X	X	X	1	0
1	0	X	X	X	0	1
1	1	0	0	\uparrow	Qa	$\overline{\text{Qa}}$
1	1	0	1	\uparrow	0	1
1	1	1	0	\uparrow	1	0
1	1	1	1	\uparrow	$\overline{\text{Qa}}$	Qa
1	1	X	X	0, 1, \downarrow	Qa	$\overline{\text{Qa}}$

Obs.: A condição * normalmente não é usada.

Requisitos e características temporais em FF

Flip-flop tipo D:



Aplica-se também para outros tipos de FF (T e JK)

Requisitos:

- t_{su} (**setup time** = tempo de preparação): Tempo que o sinal deverá estar estável **antes** da transição de *clock*
- t_h (**hold time** = tempo de manutenção): Tempo que o sinal deverá ser mantido estável **depois** da transição de *clock*

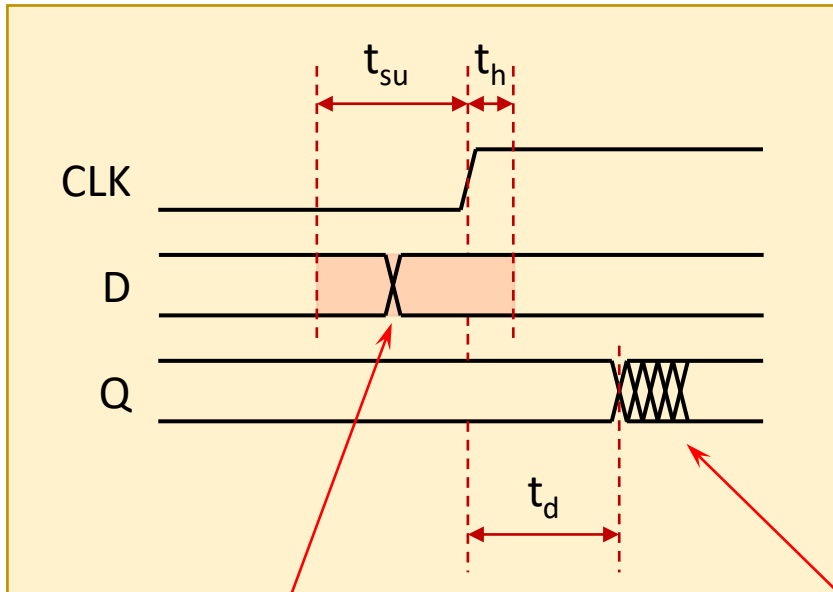
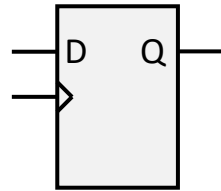
Característica:

- t_d (**delay time** = tempo de atraso), às vezes chamado de t_p (*propagation time* = tempo de propagação): tempo que a saída demora para mudar de estado **depois** da transição de **clock**

Em todos os FF, o **tempo de atraso** é maior que o **tempo de manutenção**

Requisitos e características temporais em FF

Flip-flop tipo D:



Violação dos
requisitos

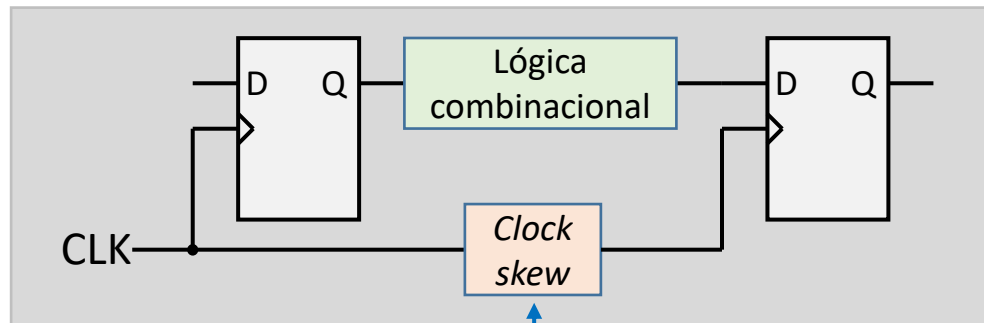
Requisitos:

- Caso os requisitos temporais não sejam satisfeitos, ocorrerá um mau funcionamento do flip-flop chamado **metaestabilidade**.
- Essa metaestabilidade caracteriza-se pela demora da saída em atingir um valor de tensão estável, que pode ser o valor correto ou o valor contrário ao correto.
- De qualquer forma, isso pode causar erros no funcionamento do sistema.

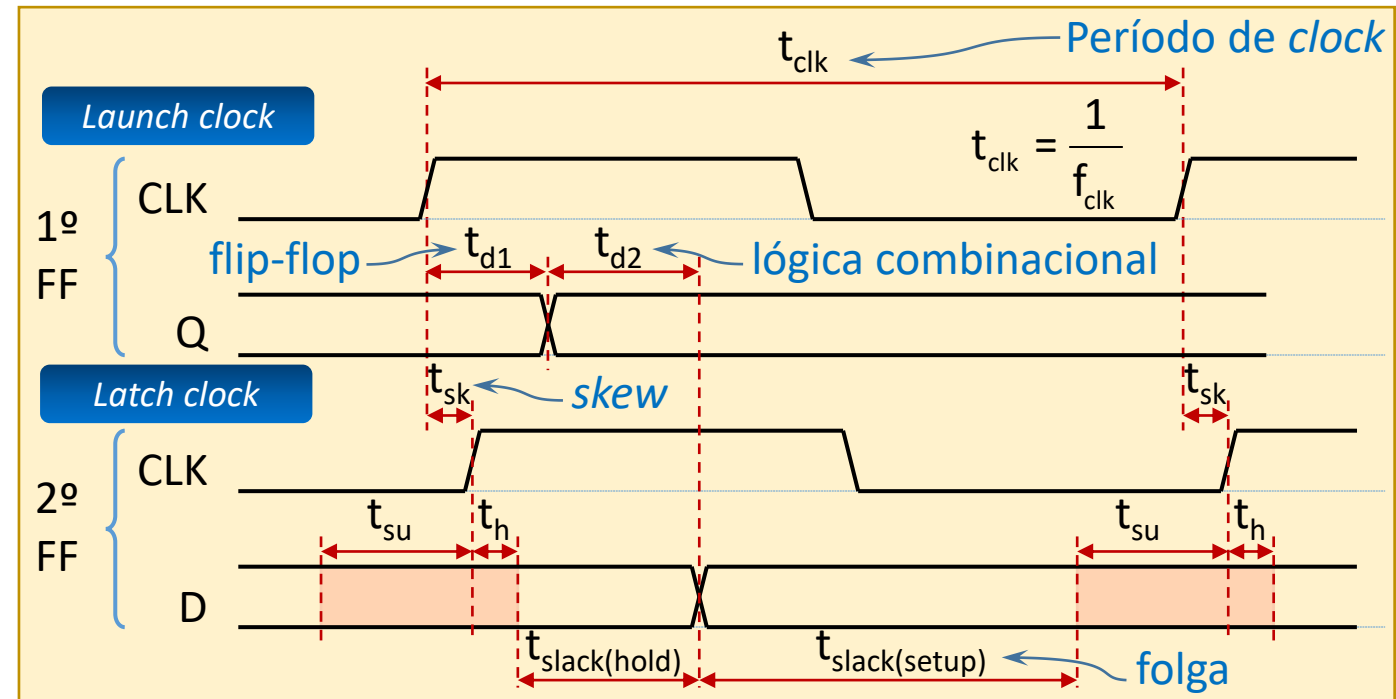
Metaestabilidade

Requisitos e características temporais em FF

Analisar o funcionamento do circuito síncrono abaixo:



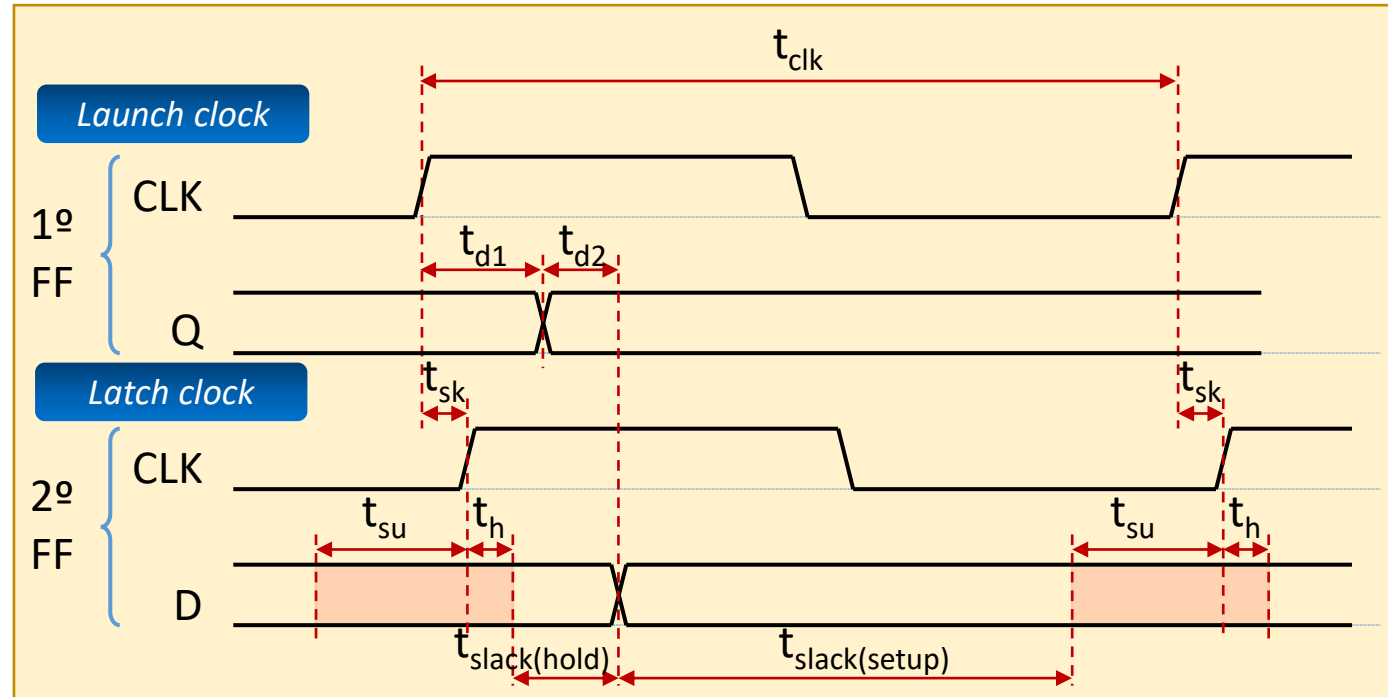
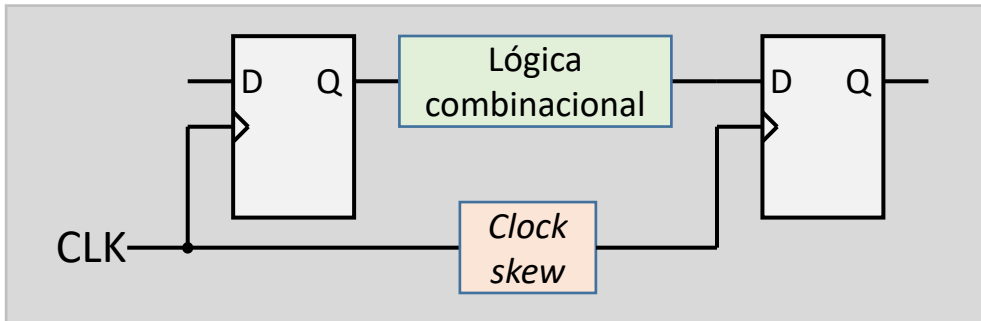
Apesar do circuito ser **síncrono**, pode haver um atraso na interligação de **CLK**



- Nesse exemplo, os requisitos de *setup* e *hold* foram **atendidos** com folga (*slack*), pois não houve transições na entrada **D** durante o período que deveria ficar estável (marcado com)

Requisitos e características temporais em FF

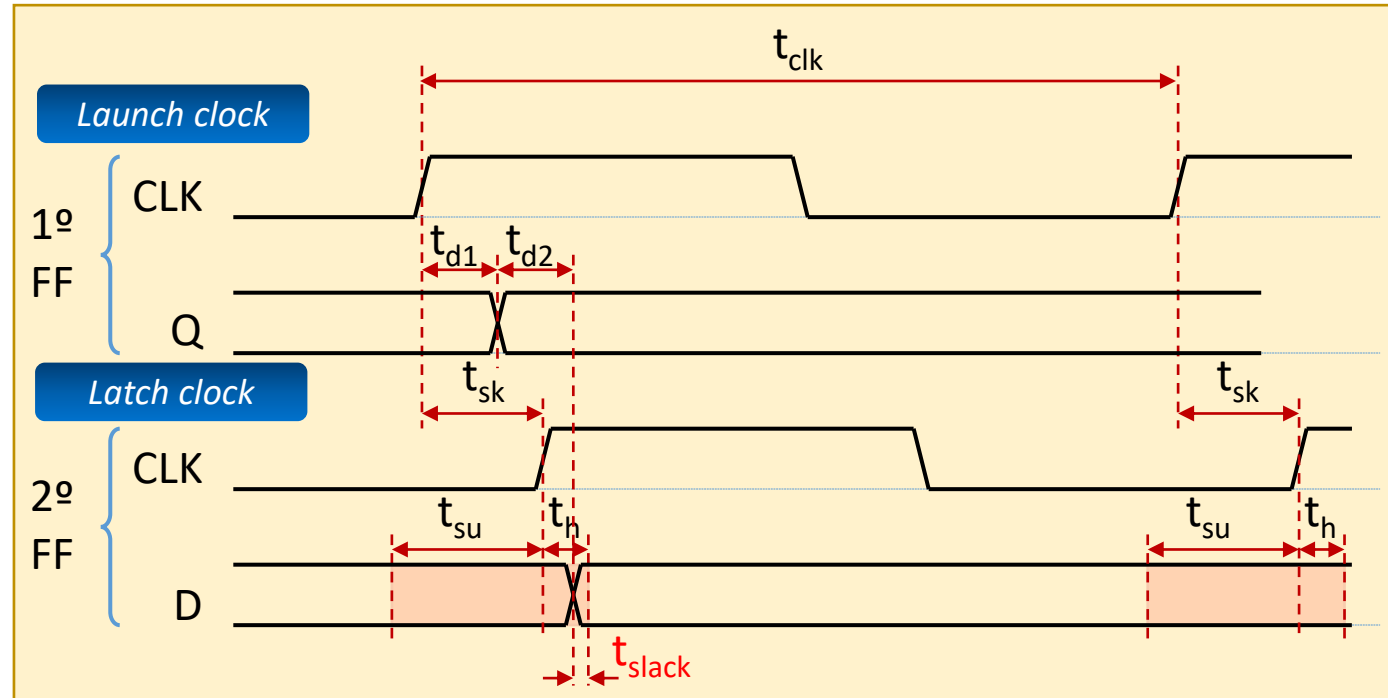
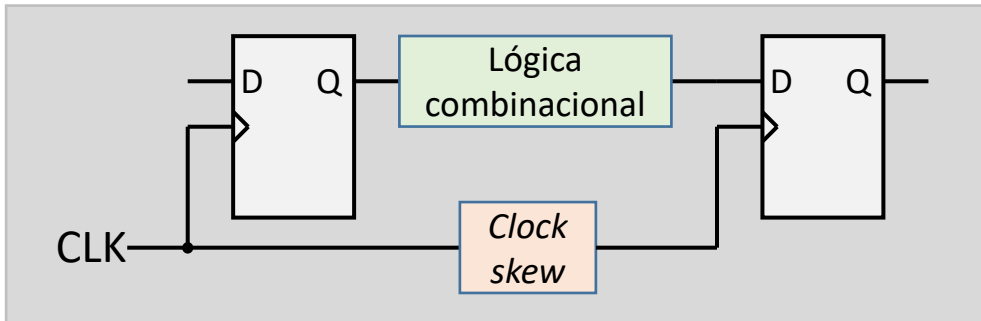
Analisar o funcionamento do circuito síncrono abaixo:



- Se o atraso da lógica combinacional (t_{d2}) diminuir, a folga para *hold* diminuir e a folga para *setup* aumentar.

Requisitos e características temporais em FF

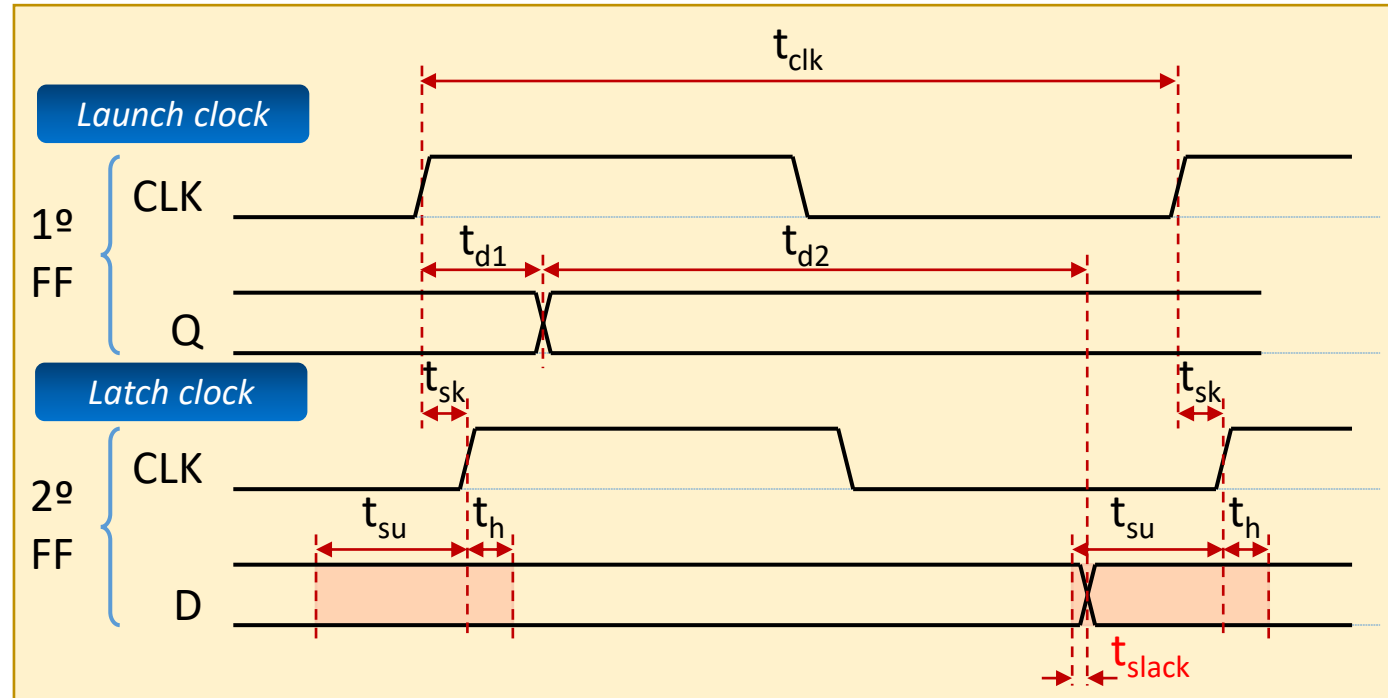
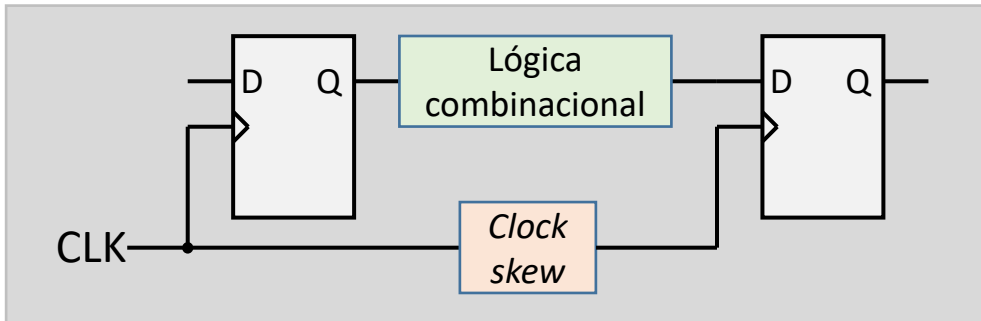
Analisar o funcionamento do circuito síncrono abaixo:



- O que aconteceria caso t_{d1} e t_{d2} fossem muito pequenos e t_{sk} fosse grande?
- Nesse caso, o requisito de *hold* seria **violado** (a folga é negativa)

Requisitos e características temporais em FF

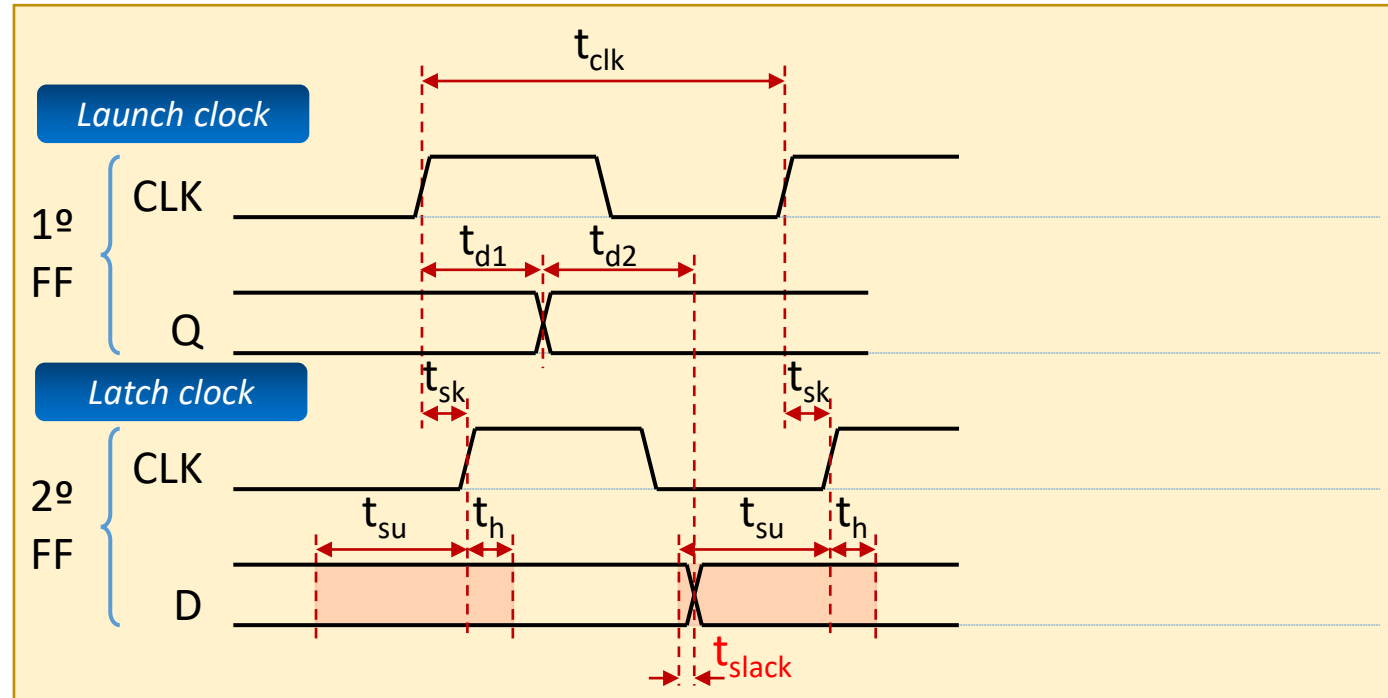
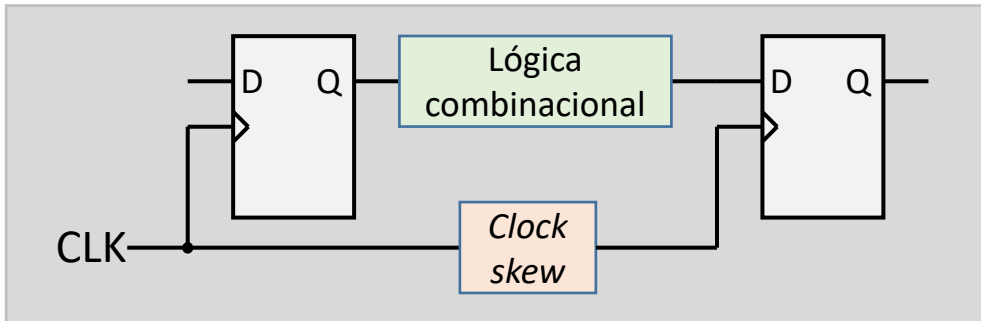
Analisar o funcionamento do circuito síncrono abaixo:



- O que aconteceria caso a lógica combinacional fosse muito complexa $\Rightarrow t_{d2}$ grande?
- Nesse caso, o requisito de *setup* seria **violado** (a folga é negativa)

Requisitos e características temporais em FF

Analisar o funcionamento do circuito síncrono abaixo:



- O que aconteceria caso a frequência do *clock* fosse alta $\Rightarrow t_{clk}$ pequeno?
- Nesse caso, o requisito de *setup* seria **violado** (a folga é negativa)

Requisitos e características temporais em FF



Em todo circuito síncrono, a máxima frequência do *clock* depende dos atrasos de propagação, do *clock skew* e dos requisitos de *setup* dos flip-flops

$$t_{\text{slack}}(\text{setup}) = t_{\text{clk}} + t_{\text{sk}} - (t_{\text{d1}} + t_{\text{d2}}) - t_{\text{su}}$$

No limiar (folga zero):

$$t_{\text{clk}(\text{min})} = t_{\text{d1}} + t_{\text{d2}} + t_{\text{su}} - t_{\text{sk}}$$

$$f_{\text{clk}(\text{max})} = 1 / t_{\text{clk}(\text{min})}$$

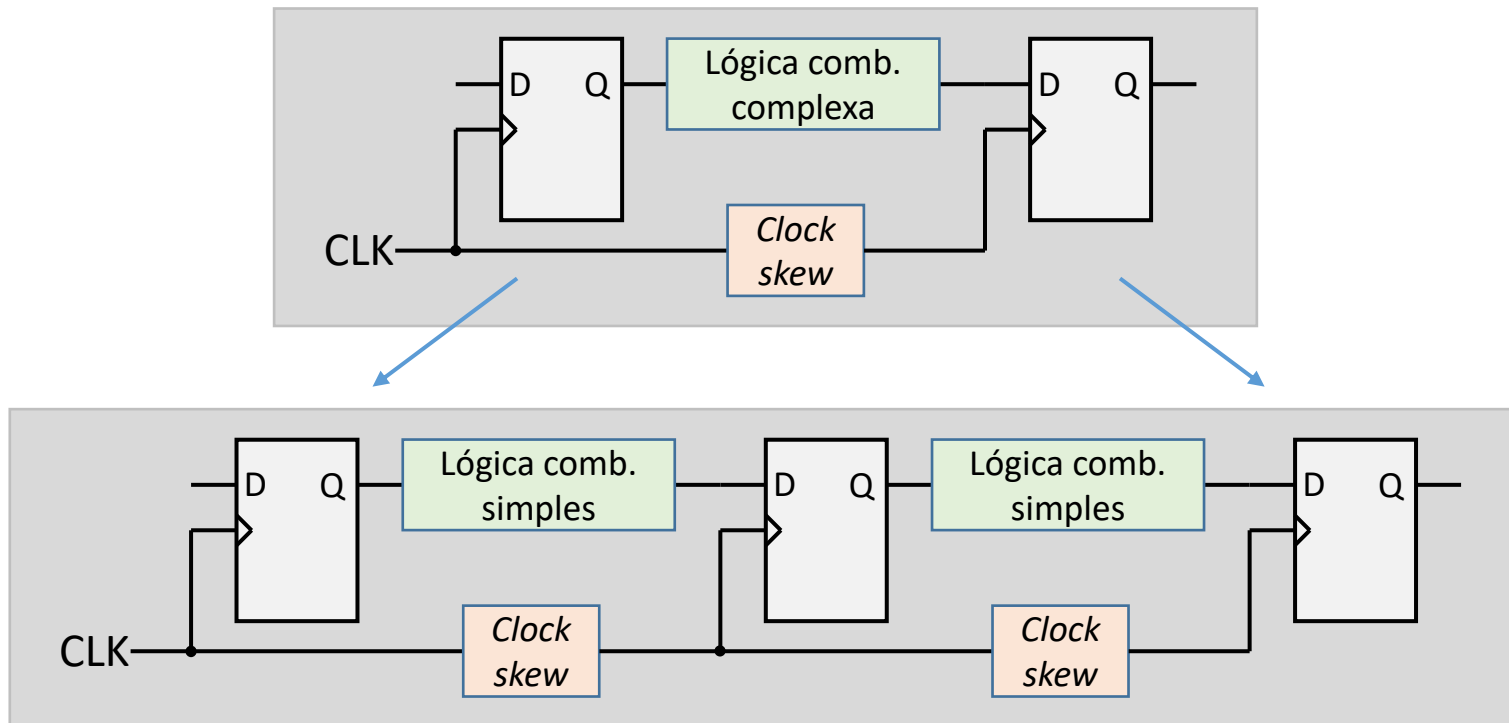
- Lógica simples \Rightarrow tempo de atraso pequeno \Rightarrow frequência máxima grande
- Lógica muito complexa \Rightarrow tempo de atraso muito grande \Rightarrow frequência máxima pequena

O que fazer se o projeto exige uma lógica complexa e frequência de trabalho alta?

Requisitos e características temporais em FF

O que fazer se o projeto exige uma lógica complexa e frequência de trabalho alta?

Solução: partir a lógica complexa em duas ou mais lógicas simples, armazenando os resultados intermediários (isso é chamado de **pipeline**)



O uso de **pipeline** aumenta a frequência máxima de trabalho de um circuito.

Entretanto, aumenta a **latência**, que é o número de ciclos de *clock* que um dado demora para ir da entrada à saída do circuito.



Fim

Até a próxima aula!