GTA / UFRJ
GRUPO DE TELEINFORMÁTICA E AUTOMAÇÃO

Politécnica UFRJ

COPPE UFRJ

Circuitos Lógicos

Aula 12

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

- Flip-flop
  - SR
  - D
  - JK
  - T

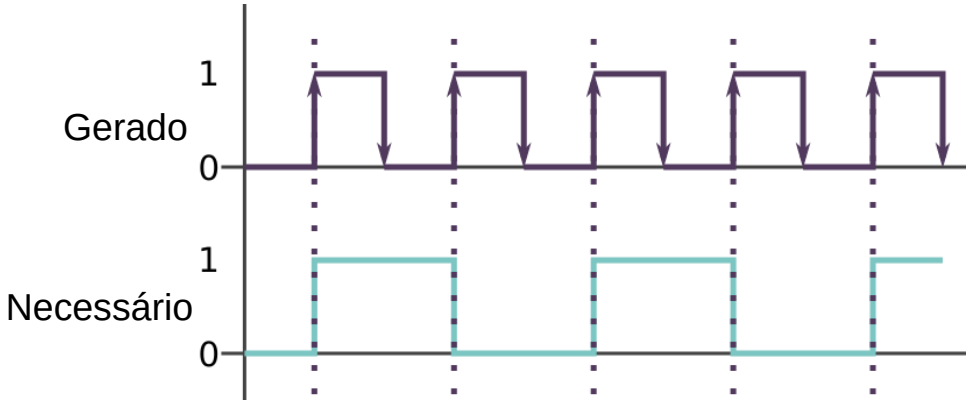
Hoje

- Introdução a circuitos sequenciais
  - Divisor de frequência
  - Contador assíncrono
- Sincronismo
  - Contador síncrono

Divisor de frequência

Problema da vida real*

- Um gerador de pulsos gera um sinal de 10kHz
  - Compartilhado por muitos dispositivos
- Um dispositivo de comunicação precisa de um clock de 5kHz
- Necessário dividir a frequência
  - Dobrar o período



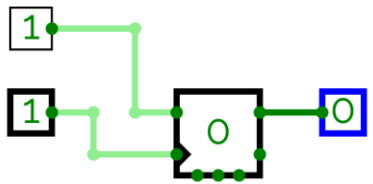
*Não é tão real assim. Foi inventada para fins pedagógicos

Saída do FF T comparada com o Clock

- Entrada ligada em 1
  - Inversão na subida de *clock*
- A cada duas subidas de *clock*
  - Q sobe uma
  - Q desce na outra

Clk	T	Q
0	x	$Q_{anterior}$
Subida	0	$Q_{anterior}$
Subida	1	$\overline{Q_{anterior}}$

Clk	T	Q
0	x	$Q_{anterior}$
Subida	0	$Q_{anterior}$
Subida	1	$\overline{Q_{anterior}}$

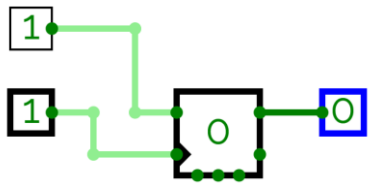


Saída do FF T comparada com o Clock

- Entrada ligada em 1
  - Inversão na subida de *clock*
- A cada duas subidas de *clock*
  - Q sobe uma
  - Q desce na outra
- **Divide o *clk* por dois**

Clk	T	Q
0	x	$Q_{anterior}$
Subida	0	$Q_{anterior}$
Subida	1	$\overline{Q_{anterior}}$

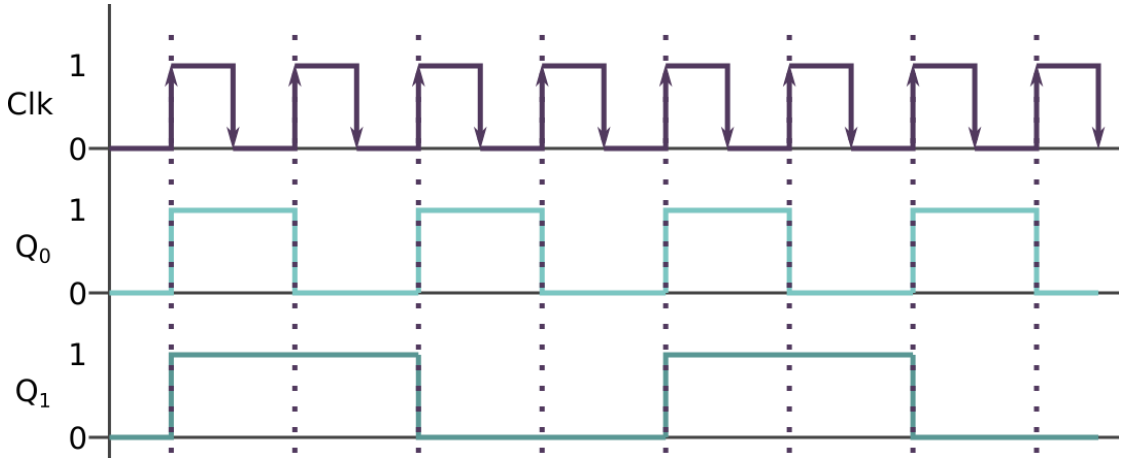
Clk	T	Q
0	x	$Q_{anterior}$
Subida	0	$Q_{anterior}$
Subida	1	$\overline{Q_{anterior}}$

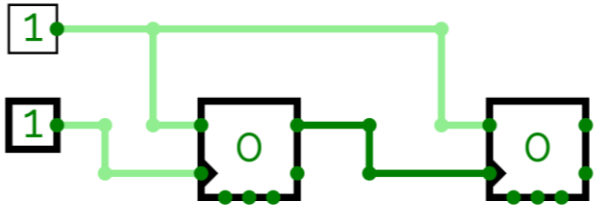


Saída do FF T ligada ao *clock* de outro T

- Clock sobe duas vezes
  - Q_0 sobe uma
  - Q_0 desce na outra
- Q_0 sobe duas vezes
  - Q_1 sobe uma
  - Q_1 desce na outra

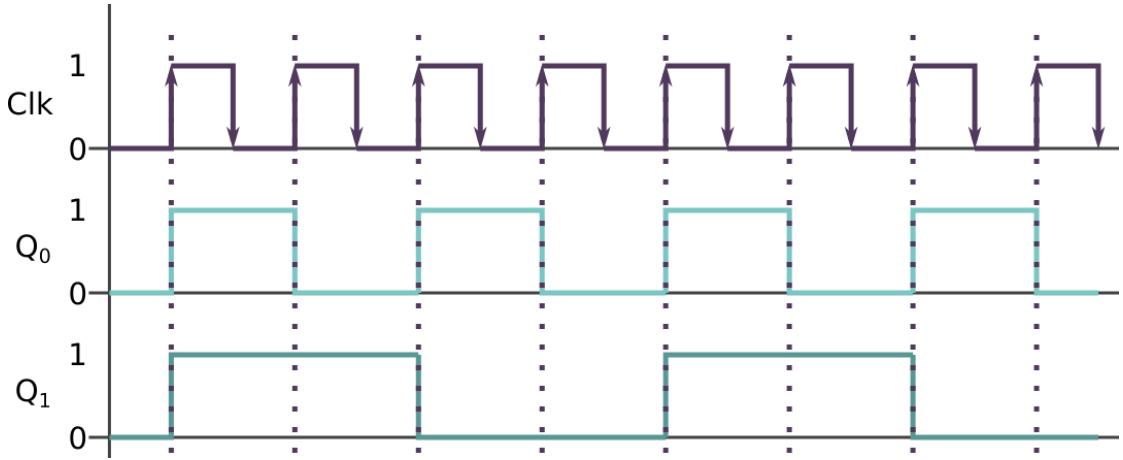
Clk	T	Q
0	x	$Q_{anterior}$
Subida	0	$Q_{anterior}$
Subida	1	$\overline{Q_{anterior}}$

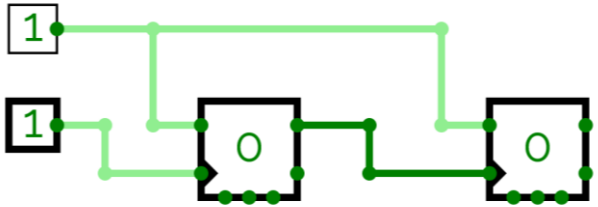




Saída do FF T ligada ao *clock* de outro T

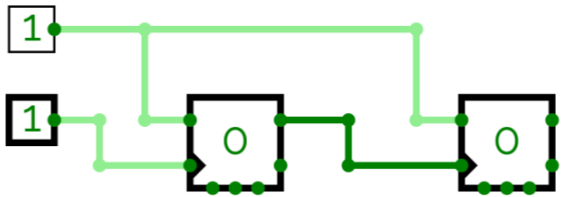
- Q_0 pulsa uma vez a cada 2 clocks
  - Divide o *clk* por dois
- Q_1 pulsa uma vez a cada 4 clocks
  - Divide o *clk* por quatro





Saída do FF T ligada ao *clock* de outro T

- Clock sobe duas vezes
  - Q_0 sobe uma
  - Q_0 desce na outra
- Q_0 sobe duas vezes
  - Q_1 sobe uma
  - Q_1 desce na outra



Clk	T	Q
0	x	$Q_{anterior}$
Subida	0	$Q_{anterior}$
Subida	1	$\overline{Q_{anterior}}$

Clk	Q_0	Q_1
inicial	0	0
subida	1	1
subida	0	1
subida	1	0
subida	0	0
subida	1	1
subida	0	1
subida	1	0
subida	0	0

E se eu colocasse ainda mais FFs ?

- Cada FF divide o *clk* mais duas vezes
  - Divide o *clk* por dois em Q_0
  - Divide o *clk* por quatro em Q_1
  - Divide o *clk* por oito em Q_2
  - Divide o *clk* por 16 em Q_3

E se eu colocasse ainda mais FFs ?

- Cada FF divide o *clk* mais duas vezes
  - Divide o *clk* por dois em Q_0
  - Divide o *clk* por quatro em Q_1
  - Divide o *clk* por oito em Q_2
  - Divide o *clk* por 16 em Q_3

Clk	Q_0	Q_1	Q_2	Q_3
inicial	0	0	0	0
subida	1	1	1	1
subida	0	1	1	1
subida	1	0	1	1
subida	0	0	1	1
subida	1	1	0	1
subida	0	1	0	1
subida	1	0	0	1
subida	0	0	0	1
subida	1	1	1	0

Olhando os Q_n em outra ordem

- Cada FF divide o *clk* mais duas vezes
  - Divide o *clk* por dois em Q_0
  - Divide o *clk* por quatro em Q_1
  - Divide o *clk* por oito em Q_2
  - Divide o *clk* por 16 em Q_3

Q_3	Q_2	Q_1	Q_0	Clk
0	0	0	0	inicial
1	1	1	1	subida
1	1	1	0	subida
1	1	0	1	subida
1	1	0	0	subida
1	0	1	1	subida
1	0	1	0	subida
1	0	0	1	subida
1	0	0	0	subida
0	1	1	1	subida

Interpretação decimal

- Cada FF divide o *clk* mais duas vezes
  - Divide o *clk* por dois em Q_0
  - Divide o *clk* por quatro em Q_1
  - Divide o *clk* por oito em Q_2
  - Divide o *clk* por 16 em Q_3

Decimal	Q_3	Q_2	Q_1	Q_0	Clk
0	0	0	0	0	inicial
15	1	1	1	1	subida
14	1	1	1	0	subida
13	1	1	0	1	subida
12	1	1	0	0	subida
11	1	0	1	1	subida
10	1	0	1	0	subida
9	1	0	0	1	subida
8	1	0	0	0	subida
7	0	1	1	1	subida

Interpretação decimal

- Cada FF divide o *clk* mais duas vezes
  - Divide o *clk* por dois em Q_0
  - Divide o *clk* por quatro em Q_1
  - Divide o *clk* por oito em Q_2
  - Divide o *clk* por 16 em Q_3

Mesmo mecanismo da contagem binária!

Porém, em ordem inversa !

Decimal	Q_3	Q_2	Q_1	Q_0	Clk
0	0	0	0	0	inicial
15	1	1	1	1	subida
14	1	1	1	0	subida
13	1	1	0	1	subida
12	1	1	0	0	subida
11	1	0	1	1	subida
10	1	0	1	0	subida
9	1	0	0	1	subida
8	1	0	0	0	subida
7	0	1	1	1	subida

Contador assíncrono

Problema da vida real*

- Máquina de vendas
  - Cliente seleciona quantas unidades de um determinado produto apertando um botão de +1 repetidas vezes
  - Máquina entrega n produtos
- Caso geral
  - Sensor/transdutor emite um pulso quando um evento de interesse acontece

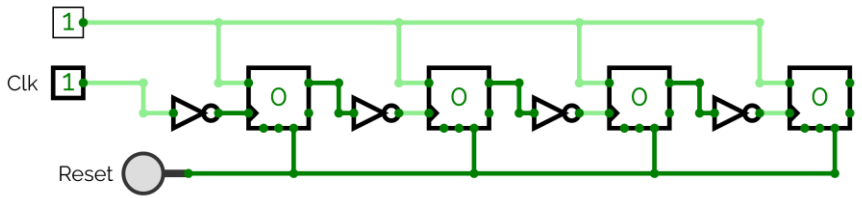
- Caso geral
  - Sensor/transdutor emite um pulso quando um evento de interesse acontece
  - Máquina deve executar uma determinada ação repetidas vezes, de acordo com o número de pulsos

Devemos **contar** o número de pulsos

*Problema da vida real é mais da vida real que problemas reais

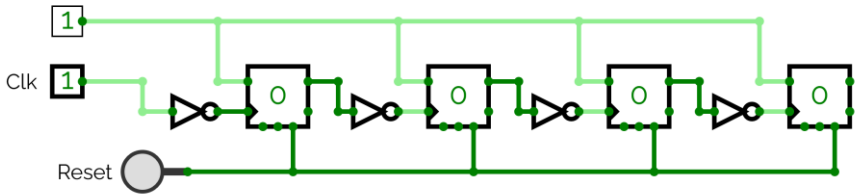
Contador assíncrono

- Divisor de frequência, mas Q_n é negado antes de ser *clk* de Q_{n+1}



Contador assíncrono

- Divisor de frequência, mas Q_n é negado antes de ser *clk* de Q_{n+1}

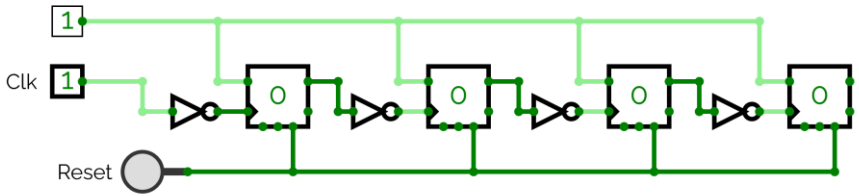


Decimal	Q_3	Q_2	Q_1	Q_0	Clk
0	0	0	0	0	inicial
1	0	0	0	1	descida
2	0	0	1	0	descida
3	0	0	1	1	descida
4	0	1	0	0	descida
5	0	1	0	1	descida
6	0	1	1	0	descida
7	0	1	1	1	descida
8	1	0	0	0	descida
9	1	0	0	1	descida

Contador assíncrono

- Divisor de frequência, mas Q_n é negado antes de ser *clk* de Q_{n+1}

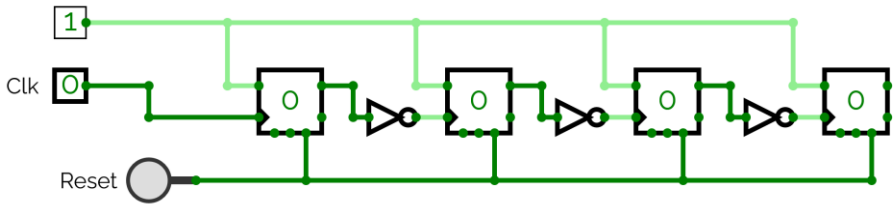
E se eu preciso de um contador com clock de subida?



Decimal	Q_3	Q_2	Q_1	Q_0	Clk
0	0	0	0	0	inicial
1	0	0	0	1	descida
2	0	0	1	0	descida
3	0	0	1	1	descida
4	0	1	0	0	descida
5	0	1	0	1	descida
6	0	1	1	0	descida
7	0	1	1	1	descida
8	1	0	0	0	descida
9	1	0	0	1	descida

Contador assíncrono RE

- Divisor de frequência, mas Q_n é negado antes de ser *clk* de Q_{n+1}
- Não negar o *clk* de Q_0

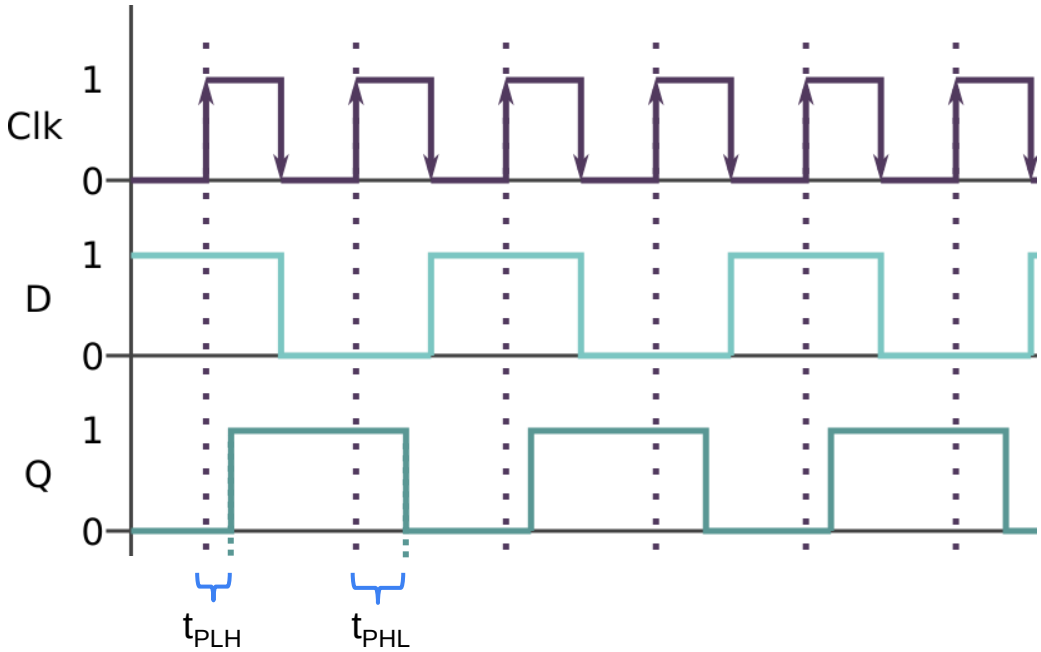


Decimal	Q_3	Q_2	Q_1	Q_0	Clk
0	0	0	0	0	inicial
1	0	0	0	1	subida
2	0	0	1	0	subida
3	0	0	1	1	subida
4	0	1	0	0	subida
5	0	1	0	1	subida
6	0	1	1	0	subida
7	0	1	1	1	subida
8	1	0	0	0	subida
9	1	0	0	1	subida

Atrasos dos FFs

Atrasos de FFs

- "*Clock-to-output delay*"
  - Atraso de propagação
  - Atraso desde a ocorrência do clock até o novo valor aparecer na saída do FF
- Tempo de subida pode ser diferente do tempo de descida



Efeitos de comparação

- Clock intel i7
  - 4GHz
    - Pulso de 0,25ns
- Atraso 7473 (2 FFs JK)
  - t_{PHL}: 40ns
  - t_{PLH}: 25ns

Entre 100 e 160 vezes!!!!

- Naturalmente, FFs do i7 são muito mais rápidos que 7473

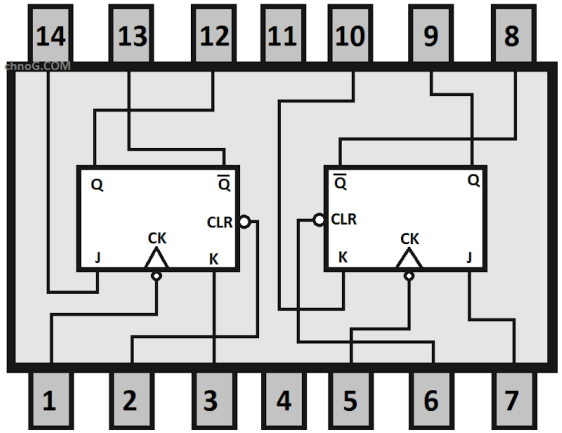
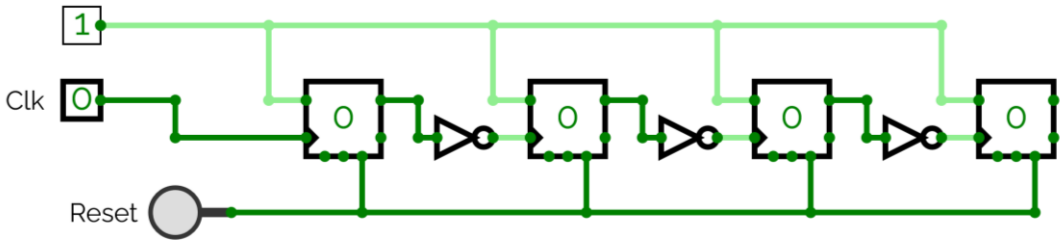


Imagem retirada de https://www.etechnog.com

Contador assíncrono

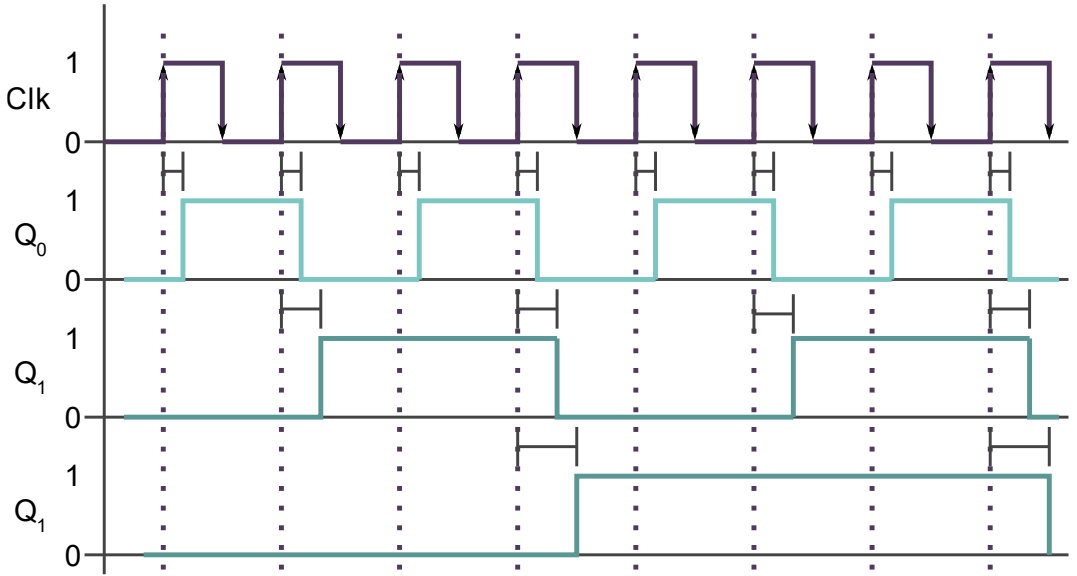
- FFs tipo T cascateados
  - Descida do FF_{n-1} excita o *clk* do FF_n

- Atraso de subida do FF_{n-1} aparece no clk do FF_n
  - Clk do FF_n possui atraso com relação ao clk do FF_n
    - Se for subida, t_{PLH}; se for descida, t_{PHL}



Contador assíncrono

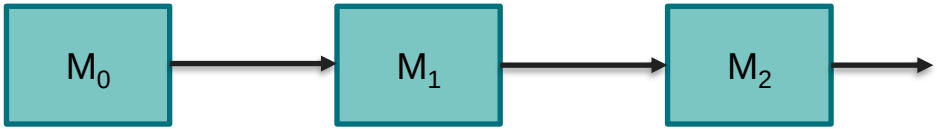
- Atrasos propagados podem afetar a leitura
  - Bit mais significativo demora n atrasos para estar correto
- Contadores com muitos bits podem ter erros
  - No próximo pulso de clock, bit mais significativo ainda não está pronto



Sincronismo

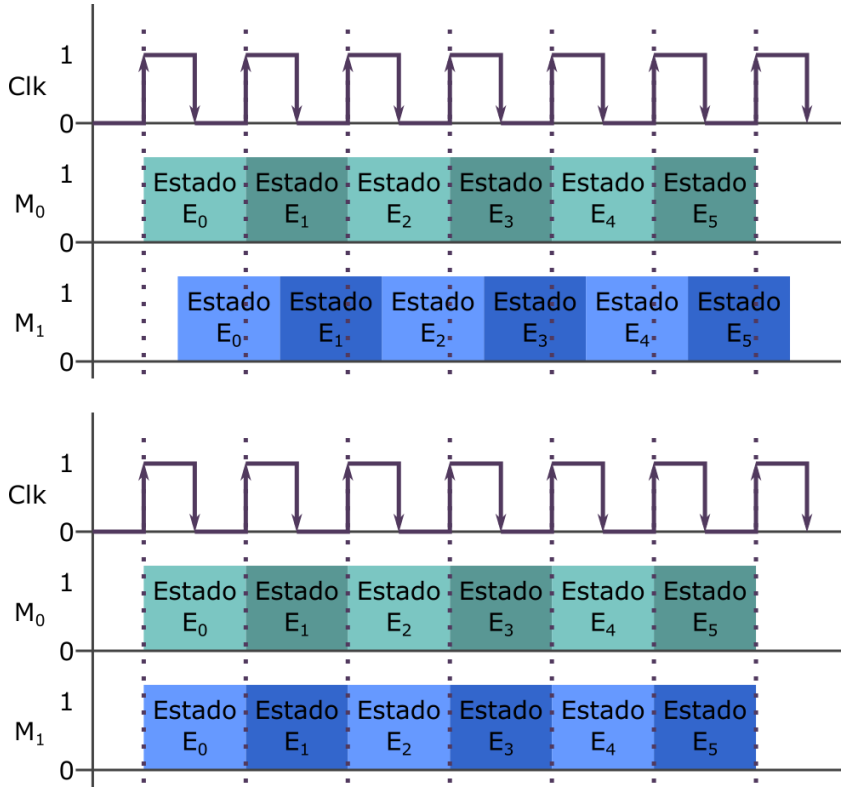
Modularização de sistemas

- Sistemas são projetados em módulos
  - “Dividir para conquistar”
- Módulos precisam se comunicar
  - Implementar sistema completo



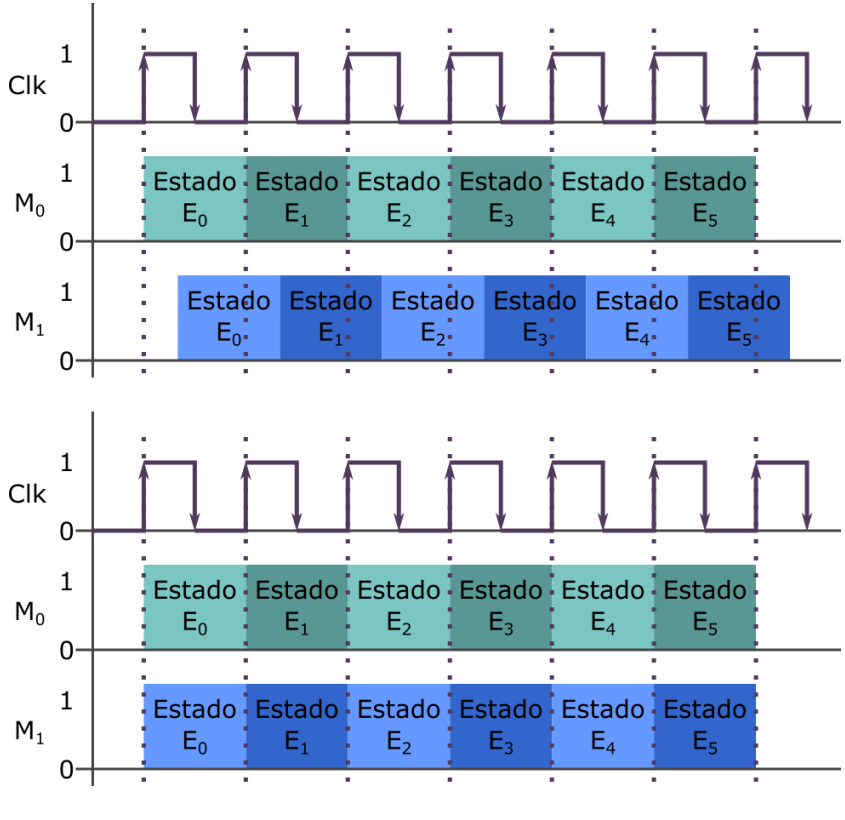
Sistemas sequenciais

- Assíncronos
  - Acoplamento temporal fraco
    - "Antes" e "agora" não necessariamente são os mesmos para todos os módulos
- Síncronos
  - Acoplamento temporal forte
    - "Antes" e "agora" são bem definidos para todos os módulos



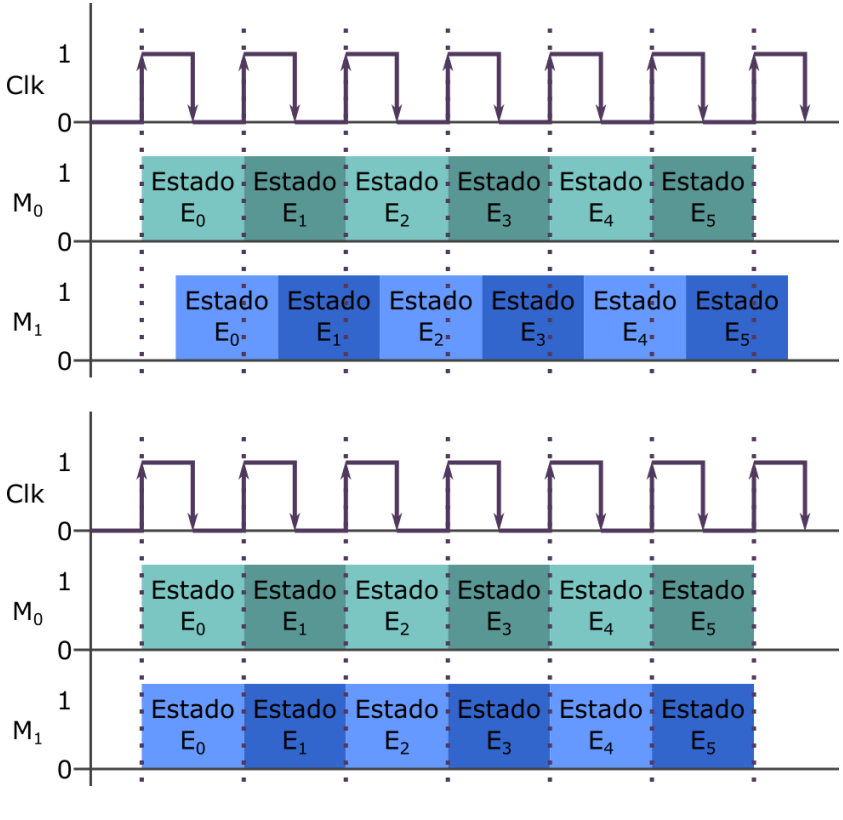
Vantagens da abordagem assíncrona

- Desacoplamento dá liberdade a cada módulo
  - Liberdade de projeto
  - Liberdade de ação



Vantagens da abordagem síncrona

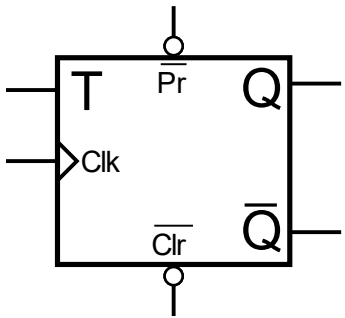
- Acoplamento garante que comunicação se dá entre módulos que estão no mesmo estado



Contador síncrono

Contador assíncrono – problema

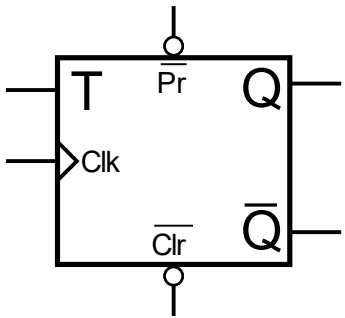
- FFs acionados com mesmo clock
  - Entrada T deve definir se FF "mantém" ou "inverte
- Quando manter?
- Quando inverter?



Decimal	Q_3	Q_2	Q_1	Q_0
0	0	0	0	0
1	0	0	0	1
2	0	0	1	0
3	0	0	1	1
4	0	1	0	0
5	0	1	0	1
6	0	1	1	0
7	0	1	1	1
8	1	0	0	0
9	1	0	0	1

Contador assíncrono – problema

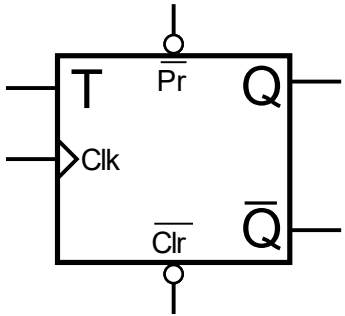
- FFs acionados com mesmo clock
  - Entrada T deve definir se FF "mantém" ou "inverte



Decimal	Q_3	Q_2	Q_1	Q_0
0	0	0	0	0
1	0	0	0	1
2	0	0	1	0
3	0	0	1	1
4	0	1	0	0
5	0	1	0	1
6	0	1	1	0
7	0	1	1	1
8	1	0	0	0
9	1	0	0	1

Contador assíncrono – problema

- Quando manter?
- Quando inverter?



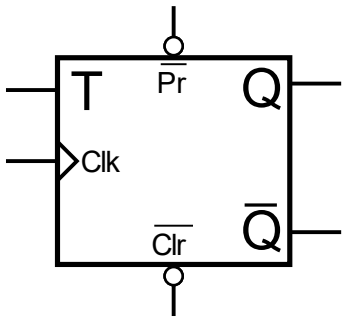
Decimal	Q_3	Q_2	Q_1	Q_0
0	0	0	0	0
1	0	0	0	1
2	0	0	1	0
3	0	0	1	1
4	0	1	0	0
5	0	1	0	1
6	0	1	1	0
7	0	1	1	1
8	1	0	0	0
9	1	0	0	1

Contador assíncrono – problema

- Quando manter?
- Quando inverter?

FFs invertem quando todos os menos significativos estão em 1

FF menos significativo sempre inverte



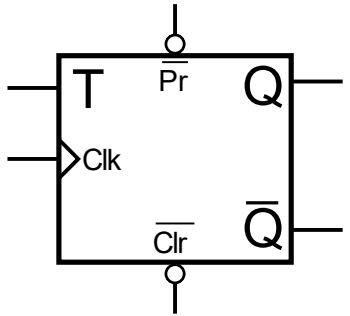
Decimal	Q_3	Q_2	Q_1	Q_0
0	0	0	0	0
1	0	0	0	1
2	0	0	1	0
3	0	0	1	1
4	0	1	0	0
5	0	1	0	1
6	0	1	1	0
7	0	1	1	1
8	1	0	0	0
9	1	0	0	1

Contador assíncrono – problema

- Lembrando que o FF T
  - Inverte quando T = 1
  - Mantém quando T = 0

$$T_0 = 1$$
$$T_1 = (T_0)$$
$$T_2 = (T_1 \cdot T_0)$$

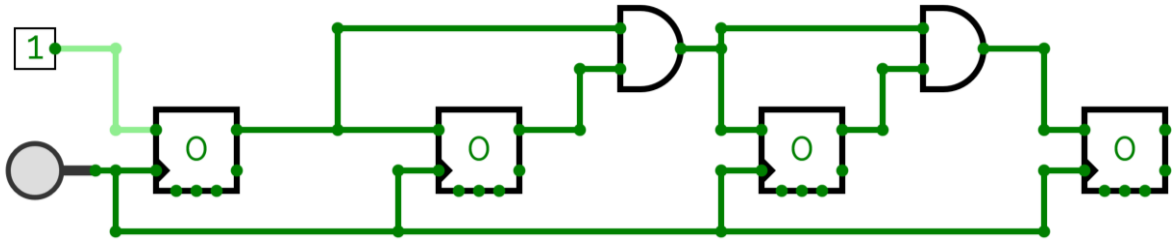
$$T_n = (T_{n-1} \cdot T_{n-2} \cdot T_{n-3} \ \ldots)$$



Contador assíncrono – problema

- Lembrando que o FF T
  - Inverte quando T = 1
  - Mantém quando T = 0

$$T_0 = 1$$
$$T_1 = (T_0)$$
$$T_2 = (T_1 \cdot T_0)$$



$$T_n = (T_{n-1} \cdot T_{n-2} \cdot T_{n-3} \ \ldots)$$

Conclusões

- Flip-flops
  - JK
    - Não tem estado proibido
    - Sabe inverter estado
  - T
    - Inverte ou não inverte
  - Divisor de frequência
    - Cada novo FF divide a frequência por 2
  - Contador
    - A cada pulso de *clock*, sofre um incremento
- Sincronismo
  - Circuitos podem ou não trocar de estado juntos

Próxima aula

- Máquinas de estado
- Projeto de circuitos sequenciais

GTA / UFRJ

GRUPO DE TELEINFORMÁTICA E AUTOMAÇÃO

www.gta.ufrj.br