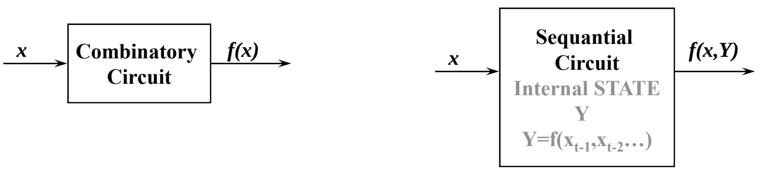
Hardware & Software for Information Technologies
Digital Sequential Circuits

NovaIMS, V.0.9 V.Lobo 2016

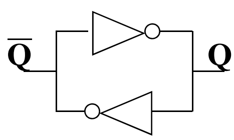
Sequential Circuits

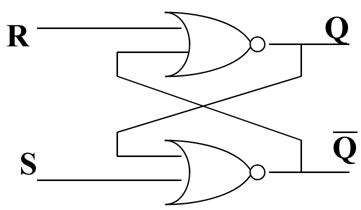
- **Combinatory Circuits**
  - The output (in each point in time) depend ONLY on the inputs present at that moment.
  - They have no memory
- **Sequential Circuits**
  - The outputs depend on the STATE of the circuit
  - The STATE of the circuit depends on the previous inputs
  - They have MEMORY



Basic memory element

- **How can we memorize a bit using only gates ?**
  - A pair of negations will maintain whatever state they have
  - It is not easy to change the state only with negations
- **S-R LATCH (Set-Reset)**
  - S (Set) input forces the output to 1
  - R (Reset) input forces the output to 0
  - 0





Can we describe a SR LATCH using thruth tables ?

Description of sequential circuits

Sistemas Lógicos (3)

- **Truth tables**
  - The PREVIOUS STATE is used as an input
  - We may simplify the truth table, specifying the output as a function of the previous state

S	R	Q	$Q_{(t+1)}$
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	x
1	1	1	x

S	R	$Q_{(t+1)}$
0	0	Q
0	1	0
1	0	1
1	1	x

Q \ S,R	00	01	11	10
0	0	0	x	1
1	1	0	x	1

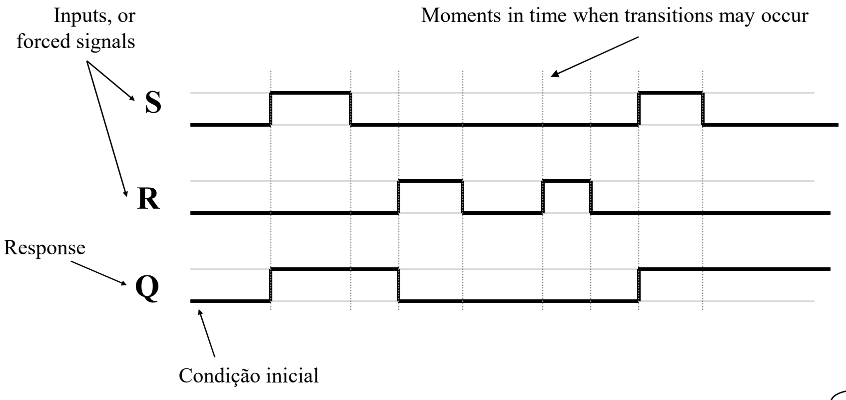
V.Lobo @ EN

3

Description of sequential circuits

Sistemas Lógicos (3)

- **Temporal diagrams**
  - Show the *behaviour in time* of the circuits when they receive a given input
  - They do not define completely the behavior of a circuit ! (just the behavior to aPARTICULAR input)



V.Lobo @ EN

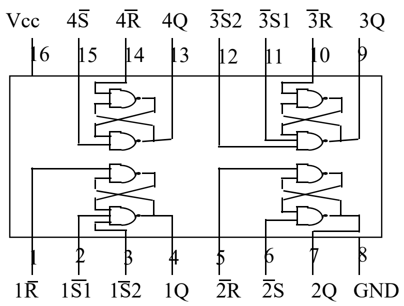
4

Descriptions of Sequential circuits

- **Asynchronous circuits**
  - **As soon as the inputs change, the outputs may change immediately**
  - **They can be very fast, but are not (yet) used much, because they are difficult to design and have problems with "critical runs"**
    - → Promissing future, as discussed in Proceedings of the IEEE, Febuary 1999
- **Synchronous circuits**
  - **There is synchronizing signal (called CLOCK) that regulates when transitions may occur**
  - **There ONLY are transisitons on the CLOCK EDGES**
  - **In temporal diagrams we only need to analyse what happens in clock edges.**
  - **The NEXT STATE is a function of what is present BEFORE the clock edge!**
  - **After a CLK edge, where changes may occur, there is a "relaxation time" where changes will take place internally, and after they all stabilize, a new clock edge may occur. This limits the maximum speed.**
  - **These are by far the most common circuits !**

SR Latches

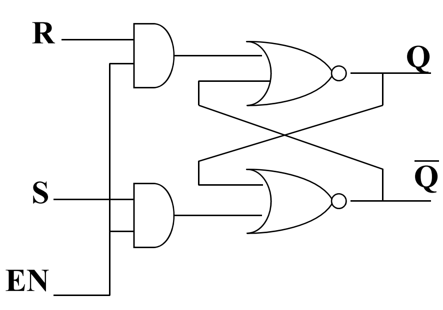
- **In the TTL family**
  - **74279**



74279

Nota: Todas as entradas são activas a 0

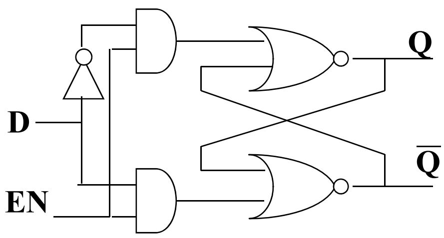
- **GATED LATCH**
  - **Have an ENABLE input**
  - **When ENABLE = 0, they keep the previous state. When ENABLE=1 it reacts to the R and S inputs.**

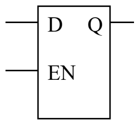


D type LATCH

- **They have only one signal imput D (from DELAY), and one enable signal.**
- **It is the basic MEMORY ELEMENT**
  - **The outut is always available.**
  - **EN=1 ⇒ Store internaly what is in the input**
  - **EN=0 ⇒ Keep what was stored previsously**

Is this synchronous?

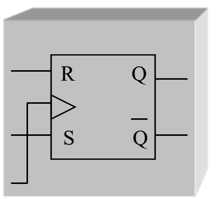




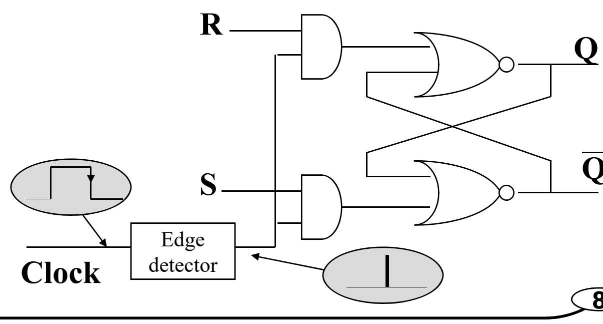
74373 has 8 Latches

FLIP-FLOPs

- **FLIP-FLOP**
  - **Only changes it's ouputs when a CLOCK edge occurs**
  - **Only one of the edges (RISING edge of the clock, or FALLING edge) leads to possible changes of the output.**
- **EDGE TRIGGERED FLIP-FLOP**
  - **Are sensitive to inputs in the moment just before the CLK edge**
  - **The "sensitivity window" is very narrow**

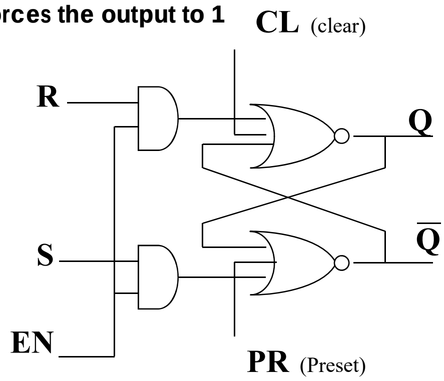


▷ means
Edge-Triggered



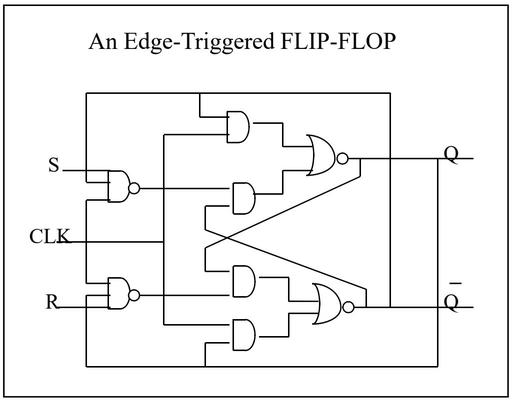
Edge-Triggered Flip-Flop

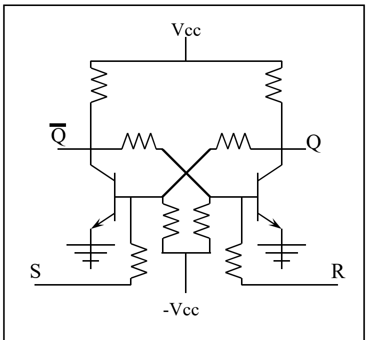
- **A flip-flop may have** *Asynchronous inputs*, **that act intependaly from the CLK**
  - *CLEAR* **- forces the output to 0**
  - *PRESET* **- forces the output to 1**



- **They are normally used to provoque RESETs or impose spectial conditions** (initial states, for example)

Other implementations





RS LATCH
(using only two transistors)

- **The way a circuit is physically implemented can vary considerably...**

Flip-flops according to inputs: D,RS,JK,T

- **D Flip-flops (delay)**
  - Memorizam e atrazam a entrada

D	Q
0	0
1	1

- **RS Flip-flops (Set-Reset)**
  - São "ligados" e "desligados" por S e R

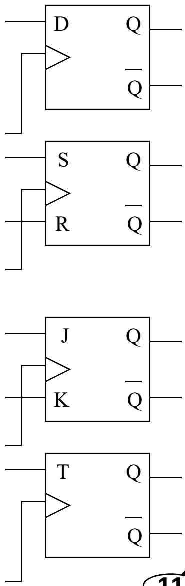
S	R	Q
0	0	Q
0	1	0
1	0	1
1	1	?

- **JK Flip-Flops**
  - Quase iguais aos RS, mas resolvem a ambiguidade de R=S=1, fazendo neste caso um *TOGGLE* (invertendo o estado anterior)
  - O *J* faz *S*et, e o *K* faz *R*eset
  - Os mais usados na prática

J	K	Q
0	0	Q
0	1	0
1	0	1
1	1	$\overline{Q}$

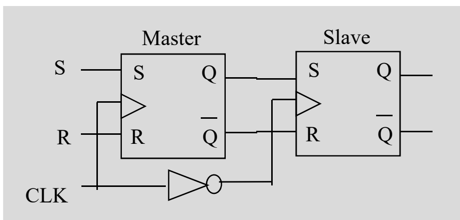
- **T Flip-Flops (Toggle)**
  - Invertem o estado quando a entrada é 1

T	Q
0	Q
1	$\overline{Q}$



Master-Slave FLIP-FLOP

- **Memoriza o que acontece quando o CLK está activo, mas só produz efeitos quando vem o flanco**
- **Tem uma concepção mais simples que os Edge-Triggered**
- **É composto por dois LATCHES em cadeia**
  - Quando um está "transparente", o outro está "fechado"
  - Nunca há um caminho directo entre a entrada e saída
  - O Primeiro (Mestre) está ligado à entrada, e fornece dados ao Segundo (o Escravo)
- **Tem " 1's catching "**
  - "Apanha" os picos



Simple Problems

Sistemas Lógicos (3)

- **In na automobile Num automóvel, um dos pisca-piscas tem uma frequência de 2Hz. Pretende-se diminuir essa frequência para 0,5Hz. Projecte o circuito diminui a frequência do pisca-pisca.**

- **Pretende-se desenhar um alarme contra ladrões, que tenha uns sensores (por hipótese uns feixes de laser infra-vermelho) que enviam um sinal 1 quando não está ninguém presente, e 0 quando está (quando está alguém presente o feixe é interrompido). Existem 4 desses sensores, e quando qualquer um deles é activado, o alarme deverá começar a tocar. O alarme só deverá parar quando alguém premir o botão de "reset" que se encontra na sua caixa. Projecte este sistema.**

- **Desenhe um sistema para actuar sobre semáforos que põe o semáforo encarnado quando recebe um 1 e verde quando recebe um 0. Nas transições de verde para vermelho, E VICE-VERSA, deverá 2 segundos no laranja.**

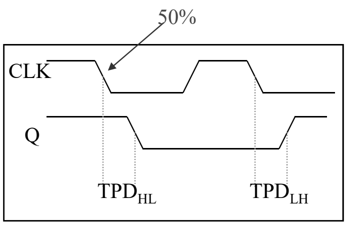
V.Lobo @ EN

13

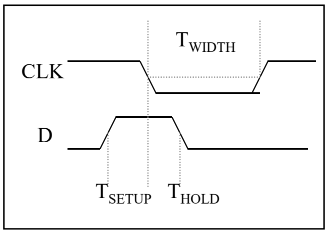
Flip-Flops – Time issues

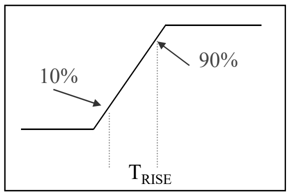
Sistemas Lógicos (3)

- **Os flip-flops podem ser sensíveis ao flanco *Ascendente* ou *Descendente*** (neste caso representados com uma bola no sinal de relógio, que significa que este é negado internamente)
- **Propagation Delay**
  - **Tempo entre o clock e a reacção do integrado**
  - **Pode ser diferente para transições H-L ou L-H**



- **Set-Up Time**
  - **Antecedência com que é necessário actuar nas entradas**
- **Hold Time**
- **Rise Time**
- **Pulse Width**





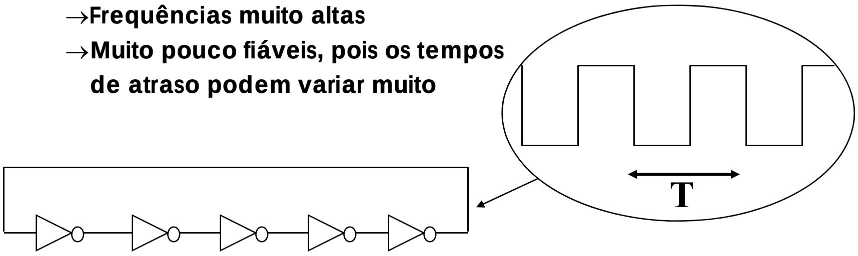
V.Lobo @ EN

14

Timers and clock generators

Sistemas Lógicos (3)

- **Geradores de Clock ou *Multivibradores Astáveis***
  - **Geram ondas quadradas com uma dada frequência**
- **Implementação com PORTAS LÓGICAS**
  - →**Frequências muito altas**
  - →**Muito pouco fiáveis, pois os tempos de atraso podem variar muito**



$$\text{Frequência} = \frac{1}{\text{Período}}$$

$$T \cong 2 \times n \times t_{pd}$$

$$f \cong 1/(2 \times n \times t_{pd})$$

5 gates com t_{pd}= 9ns ⇒ f = 11MHz

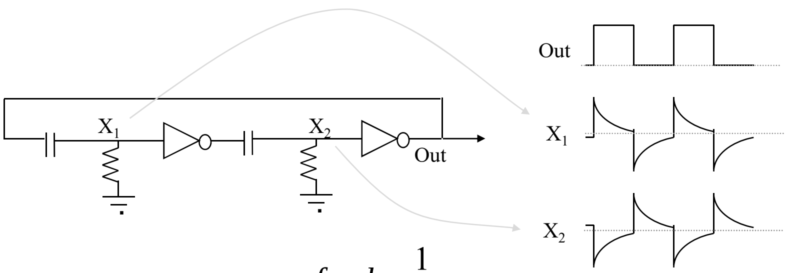
V.Lobo @ EN

15

Temporizadores

Sistemas Lógicos (3)

- **Com Circuitos RC e portas lógicas**
  - **Facilmente ajustáveis**
  - **Pouco precisos (dependentes das tolerâncias das resistências e condensadores)**



$$f = k\frac{1}{2RC}$$

- **Com Cristais**
  - **Muito precisos**
  - **Só para frequências razoavelmente altas**
  - **Normalmente usam-se integrados dedicados para gerar o clock a partir de um cristal de quartzo**

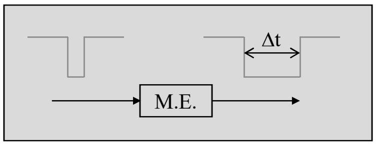
V.Lobo @ EN

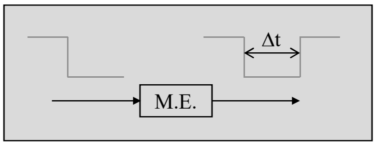
16

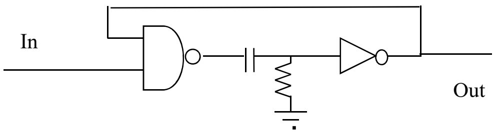
Temporizadores Monoestáveis

Sistemas Lógicos (3)

- **Mono-estáveis**
  - Também chamados "one-shot"
  - Geram pulsos de <u>largura fixa</u>
  - Podem ser usados para
    - →Gerar atrasos fixos
    - →"Rectificar" picos







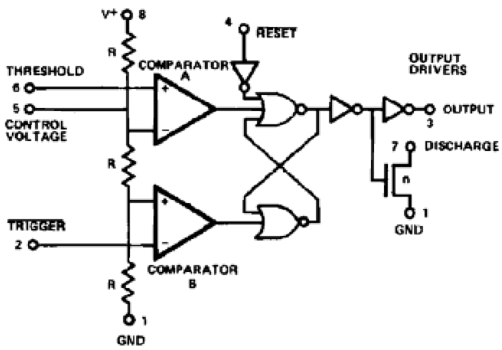
V.Lobo @ EN

17

Integrados para temporizadores dedicados

Sistemas Lógicos (3)

- **555**
  - Integrado muito flexível para circuitos temporizadores
  - Pode ser usado como monoestável ou gerador de clock
  - O "duty-cycle" pode ser regulado
  - É muito usado, e há bastantes " clones".
  - Esquema:
  - Pinout:



V.Lobo @ EN

18

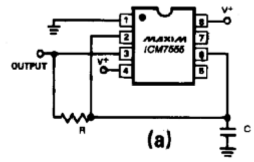
Integrated Circuits for clock generation

Sistemas Lógicos (3)

- Typical mountings for the 555

A) 50% duty-cycle Clock

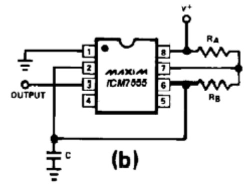
f = 1/(1,4 RC)



B) Variable duty-cycle Clock

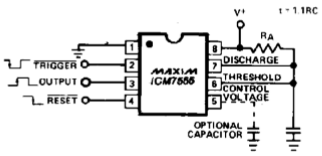
f = 1,46 / ((Ra+2Rb) C)

d = Rb / (Ra+2Rb)



C) Monostable

T_{pulso} =1,1 RC



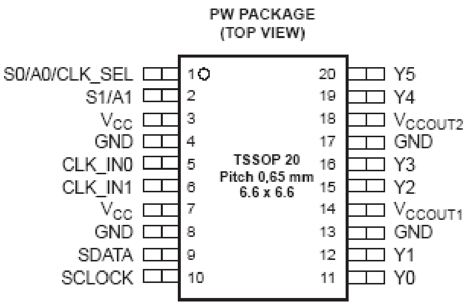
V.Lobo @ EN

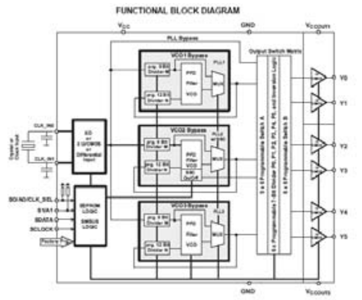
19

Other Clock Generators

Sistemas Lógicos (3)

- Texas Instruments CDCE706 and TICE906
  - Use only 1 crystal (or another clock signal)
  - Can generate 6 clock signals with different frequencies and duty-cycles
  - Programmable by software
  - Keeps programming in an EEPROM
  - (more information on http://focus.ti.com/docs/prod/folders/print/cdce706.html)



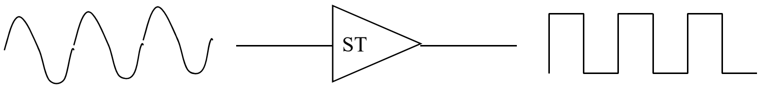


V.Lobo @ EN

20

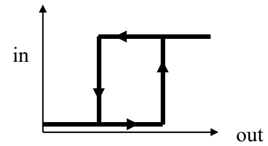
Schmitt-Triggers

- **May receive analog signals, but output digital**
  - **Output has only 2 levels**
  - **They are used to *regenerate* digital signals**



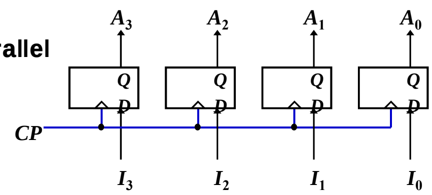
- **They have a hysteresis cycle:**
  - **To go from 0 to 1 the signal has to be significantly high, but to go from 1 to 0 the signal has to be significantly low**
  - **They eliminate "Cross-over noise"**

What types of problems can be solved by these circuits ?



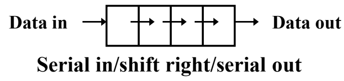
Registers

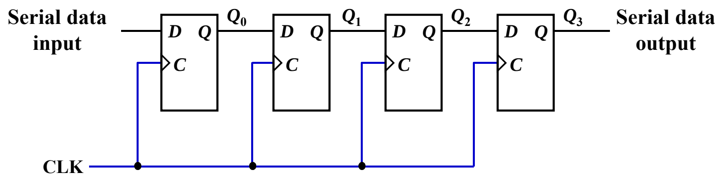
- **How can we store information that needs more than 1 bit?**
  - **Use a set of Flip-Flops**
  - **Some sort of ordering is necessary to organize the flip-flops (FF)**
- **Registers**
  - **Basically, a collection of Flip-flops**
  - **They differ in how data is LOADED into the flips-flops; how that data is available to be READ from the outside; and possibly how data is passed from one flip-flop to another.**
- **Parallel Load Registers**
  - **Data is input and output in parallel**
  - **Operations:**
    - **→Parallel-in/Parallel-out**



Shift Register

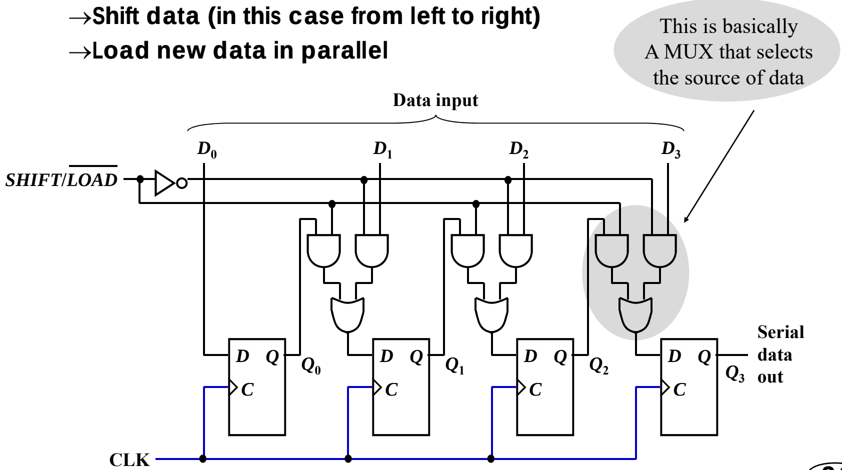
- **Shift Register** *(Registo de deslocamento)*
  - They are ***tapped delays***
  - The have a ***Serial-in*** input and a ***Serial-out*** output
  - Data (stored internally) is usually also available to be read in parallel
    - →May be used for series to parallel conversions
  - Some registers can do ***shift-right*** or ***shift-left*** operations
  - Application example
    - →Ring counter
    - →Detect a sequence of bits (for example morse code 00011000)





Multifunction Registers

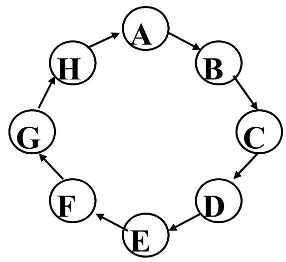
- **Parallel-in/serial-out registers**
  - May be used for Parallel/Serial conversions
  - They have a control input that selects which operation will be done:
    - →Shift data (in this case from left to right)
    - →Load new data in parallel



Counters

Sistemas Lógicos (3)

- **What is a counter ?**
  - A system that CYCLES through a series of STATES
  - Examples of sequences:
    - →0,1,2,3,,0,1,2,3,0,1,2,3…
    - →9,7,5,3,1,9,7,5,3…
  - Natural Binary Counter
    - →Corresponds to the traditional idea of counter
    - →Counts 0,1,2,3…(2^n-1)
  - Ring counter
    - →Counts 100,010,001...100,010….



- **Modulus of a counter**
  - Number of states before returning to the initial state

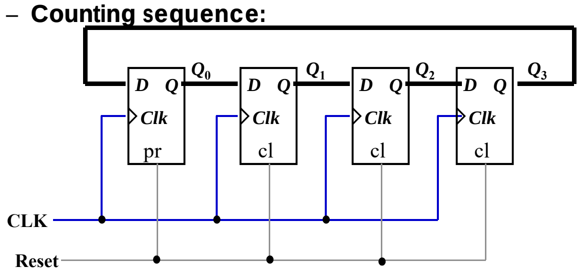
V.Lobo @ EN

25

Ring Counter

Sistemas Lógicos (3)

- *N* flip-flops, with only one set at 1 while others are 0
- The active flip-flop passes it's "1" onto the next when the clock edge arrives
  - It's a shift register with feedback from the last bit
- *N* flip-flops ⇒ *N* different states
- Example of a ring counter with 4 bits
  - Counting sequence:



reset

1000
0100
0010
0001
1000
0100
.
.
.

V.Lobo @ EN

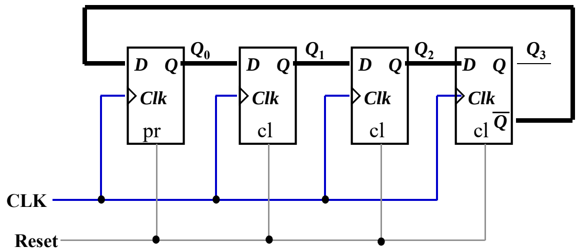
26

Johnson Counter

- Similar to the ring counter, but the negation of the output is fed back to the input (instead of the outut itself)
- *N* flip-flops ⇒ 2*N* diferente states
- Example of a 4 bit Johnson counter
  - Counting Sequence

reset

1000
1100
1110
1111
0111
0011
0001
0000
1000
1100
.
.



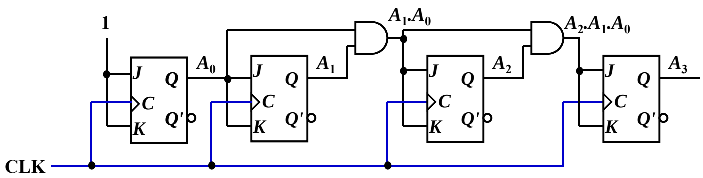
Q

Synchronous Binary Counter (mod 2^n)

- Counts in natural binary
  - The LSB (Least Significant Bit) always toggles
  - The more significant bits only toggle when all the previous bits are 1
- *N* flip-flops ⇒ 2^N different states
- Example of a 4 bit synchronous binary counter
  - Counting Sequence

$A_3A_2A_1A_0$

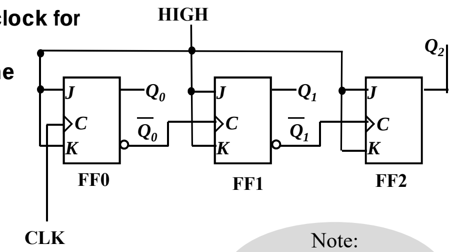
0000
0001
0010
0011
0100
0101
0110
0111
1000
1001
1010
1011
1100
1101
1110
1111
0000
0001
.



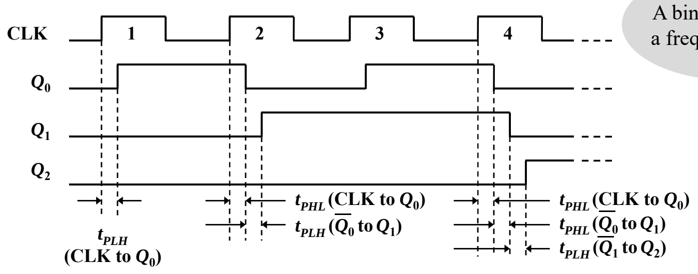
Asynchronous Binary Counter (ripple-mod 2^n)

Sistemas Lógicos (3)

- The clock is not commont to all FF
  - The output of one flip-flip acts as clock for the next one
  - There are slight delays between the Fli-flops
- They are easier to build
  - They do not need external gates



Note:
A binary counter is
a frequency divider



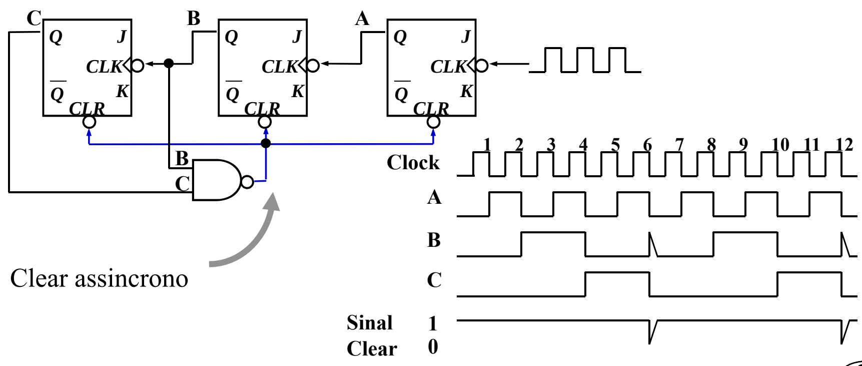
V.Lobo @ EN

29

Arbitrary modulus Binary Counter

Sistemas Lógicos (3)

- Key idea → Reset the counter before it finishes the natural cycle
  - Detect the first "unwanted" state, and force an asynchronous clear (reset)
    - →Use AND gates to detect the unwanted state
    - →The unwanted state will exist during a small peak



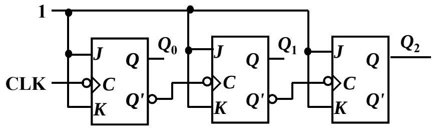
V.Lobo @ EN

30

UP/DOWN Counters

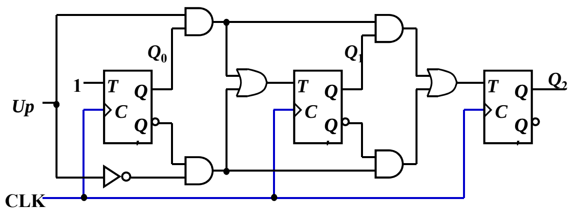
- **Down counters**
  - **Use NOT(Q) insted of Q**

Asynchronous 3 bit DOWN counter



111
110
101
100
011
010
001
000
111
110
101
100
011
.
.

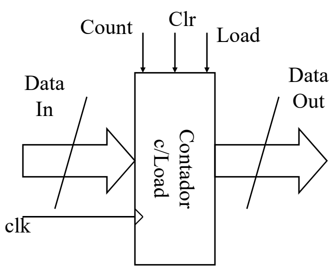
- **UP/DOWN counters**
  - **Can count UP (0,1,2...) or DOWN (9,8,7...)**
  - **The funcion (UP or DOWN) is normally selected by an extra contol input**

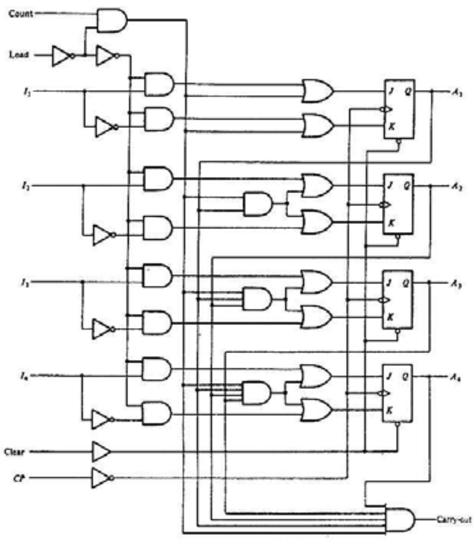


Contador UP/DOWN de 3 bits, síncrono

Counters with parallel load

- **Counters with parallel load**
  - **Instead of just a RESET input to force an initial 0 state, they may be reset to any given state (i.e., loaded with any data)**

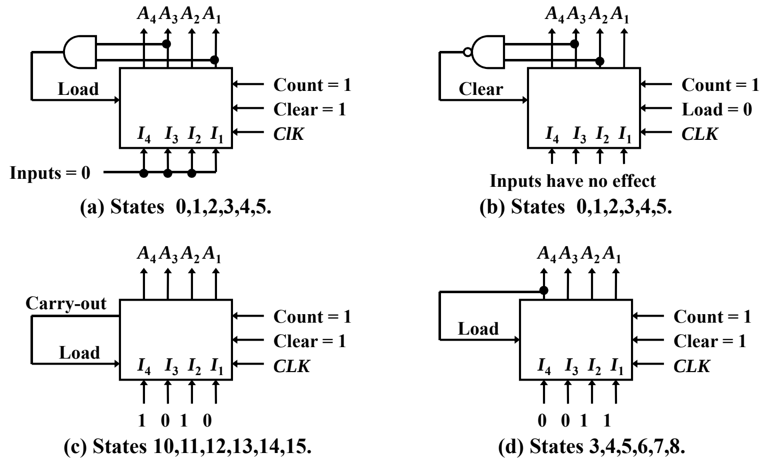




Examples modulus 6 counters

Sistemas Lógicos (3)

- **4 different ways of obtaining a modulus 6 counter with a modulus 16 binary counter with "parallel load" (synchronous) and clear (asynchronous)**

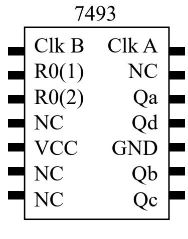


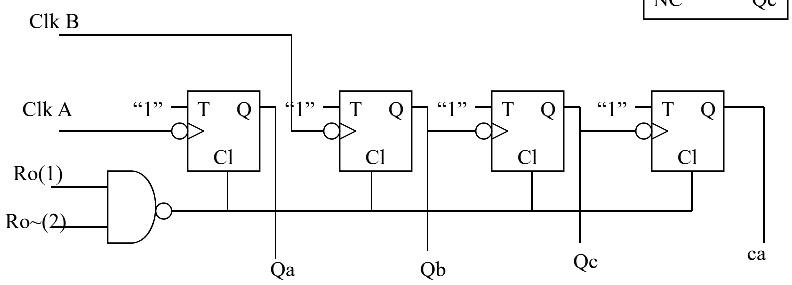
33

Counters – Available digital circuits

Sistemas Lógicos (3)

- **7493**
  - **Asynchronous Binary Counter (ripple-counter) with 4 bits**
  - **1st bit is independent from the others**
    - **→It must be connected externally**



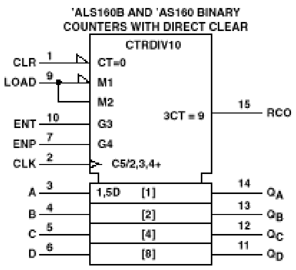


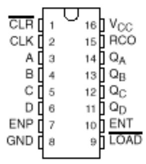
34

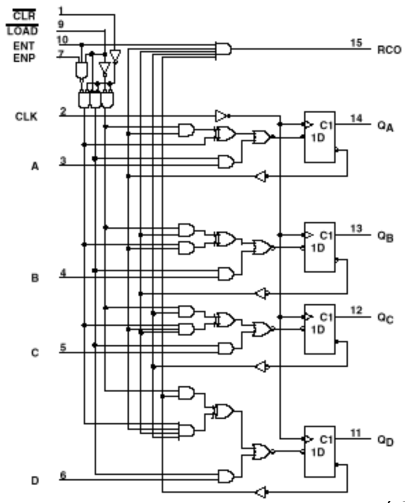
Contadores – Available devices

Sistemas Lógicos (3)

- **74163**
  - Synchronous 4 bit binary counter, with pre-load and enable
- **Others**
  - 74160 – Synchronous BCD counter
  - 74190 - Synchronous BCD up/down counter with pre-Load,







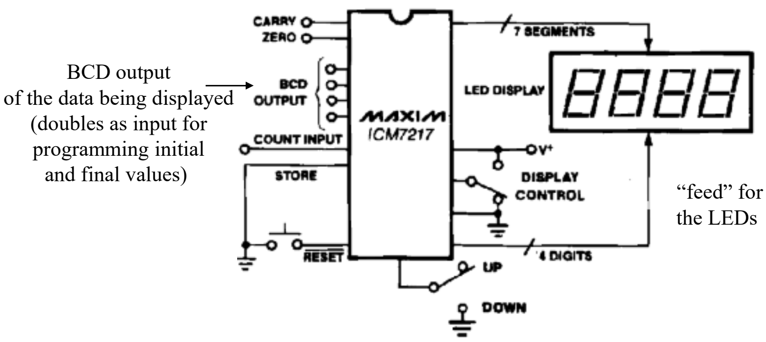
V.Lobo @ EN

35

Counters – More complex devices

Sistemas Lógicos (3)

- **Maxim 7217**
  - Up/down 4 digit counter
  - Direct outputs for 7 segment displays (common cathod or anod and variable frequency)
  - Initial and final value are programable
  - Output carry for cascading
  - Allows for very compact systems

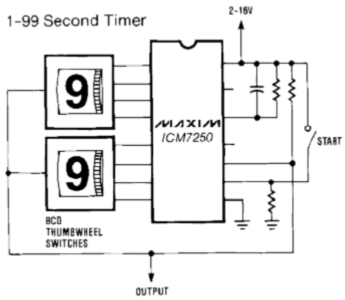


V.Lobo @ EN

36

Counters – More complex devices

- **7250**
  - **Digitally programable timer (0-99)**
  - **Unit delay defined by a RC circuit (thus, “tunable”)**
  - **May generate delays from micro-seconds to days**



Practical problem

For security reasons, we whish to prevent that more than 230 people get aboard one of boats that cross the Tagus river. In the entrance to the boats, there some rotating barriers that count how many people get in (sending a “1” pulse on the COUNT output each time someone passes).These barriers may be blocked by sending a “1” to the BLOCK input. The barrier should be blocked when 230 people have passed, and must only be unlocked when the boat leaves the pier.When it does, there is a sensor in the pier that sends a “1” pulse to it’s LEFT output.
Project this system