TÉCNICO LISBOA

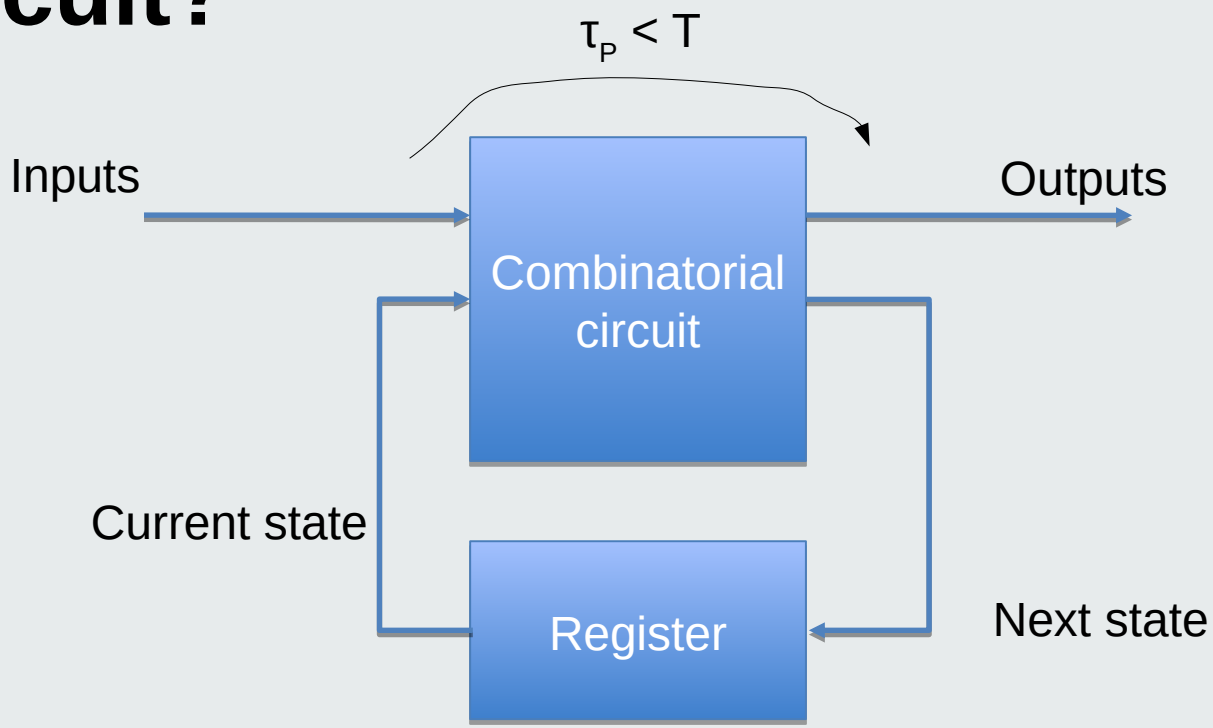
# Computer Electronics

## Lecture 6: Digital Circuits and Verilog

# Recap: Digital circuits

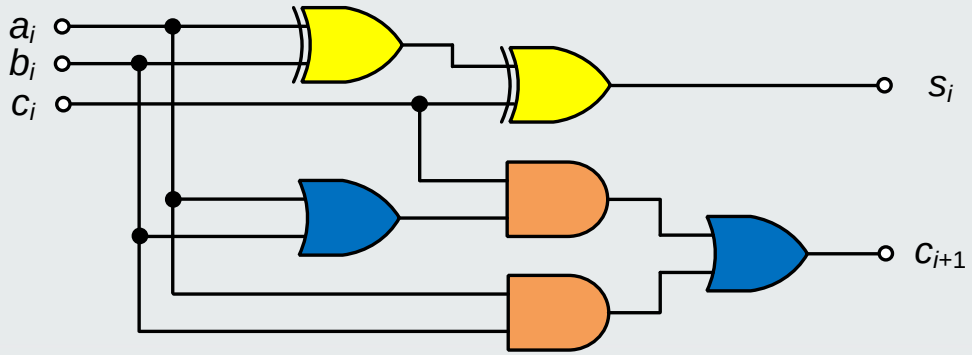
- Combinational (no feedback paths)
- Sequential (with feedback paths – remembers the past)
  - Asynchronous (no clock signal)
  - Synchronous (with clock signal)
- Asynchronous circuits
  - Very difficult to design: need to account for physical propagation delays and logic glitches
  - Only small designs: latches and flip-flops
- Synchronous circuits
  - Easy to design: propagation delays and glitches are forced out of the equation
    - During the clock period we wait for them to go way!
  - Really large designs: CPUs, GPUs, FPGAs, custom

# Recap: what is inside of a digital circuit?



# Recap: what is a combinatorial circuit?

- Full adder circuit
- $S_i = S_i(a_i, b_i, c_i)$;
- $C_i+1=C_i+1(a_i, b_i, c_i)$



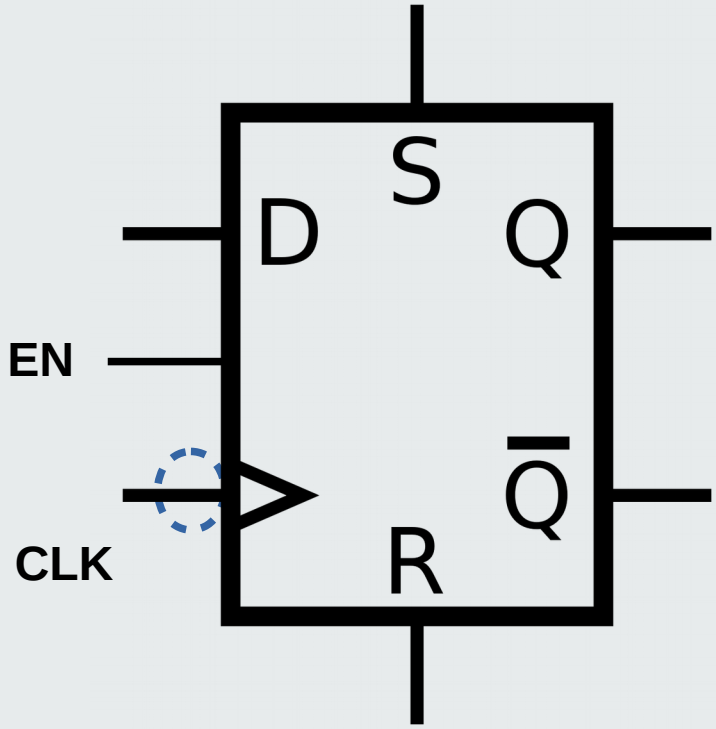

Propagation delays!

Glitches!

# Recap: what is a register

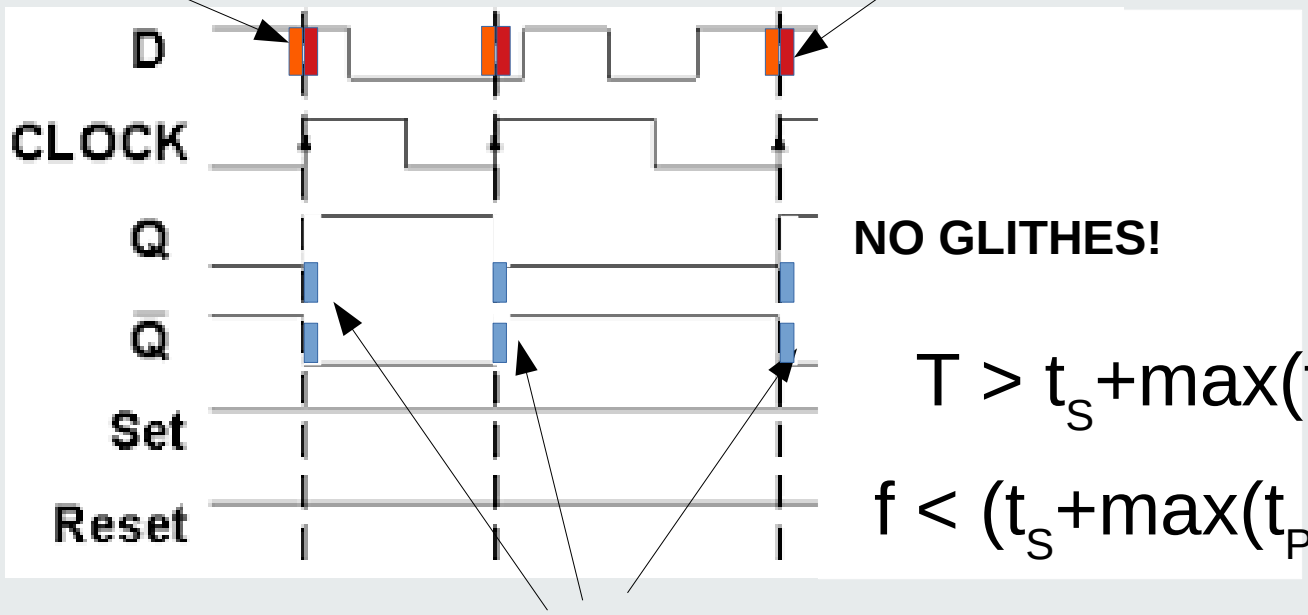
- It is a set of memory elements (flip-flops or latches)
- **Latches** are *transparent*, the output changes with the input if enabled: **RARELY USED**
- **Flip-flops** are *not transparent*, the output only changes at on of the active clock signal edge: **THEY ARE EVERYWHERE**

# The FLIP-FLOP!




- Only D type is studied
- Verilog description is directly mapped to a library component
- Set (S) and Reset (R) are optional
- It is rare that S and R are simultaneously supported: no library component
- Responds to the rising (positive) edge of the clock (CLK)
- Responds to the falling (negative) edge of the clock if an inverter is placed at the CLK input (DANGER ZONE!)
- The optional Enable signal (EN) determines whether the flip-flop will respond or not to the clock edge

# The FLIP-FLOP Timing Diagram



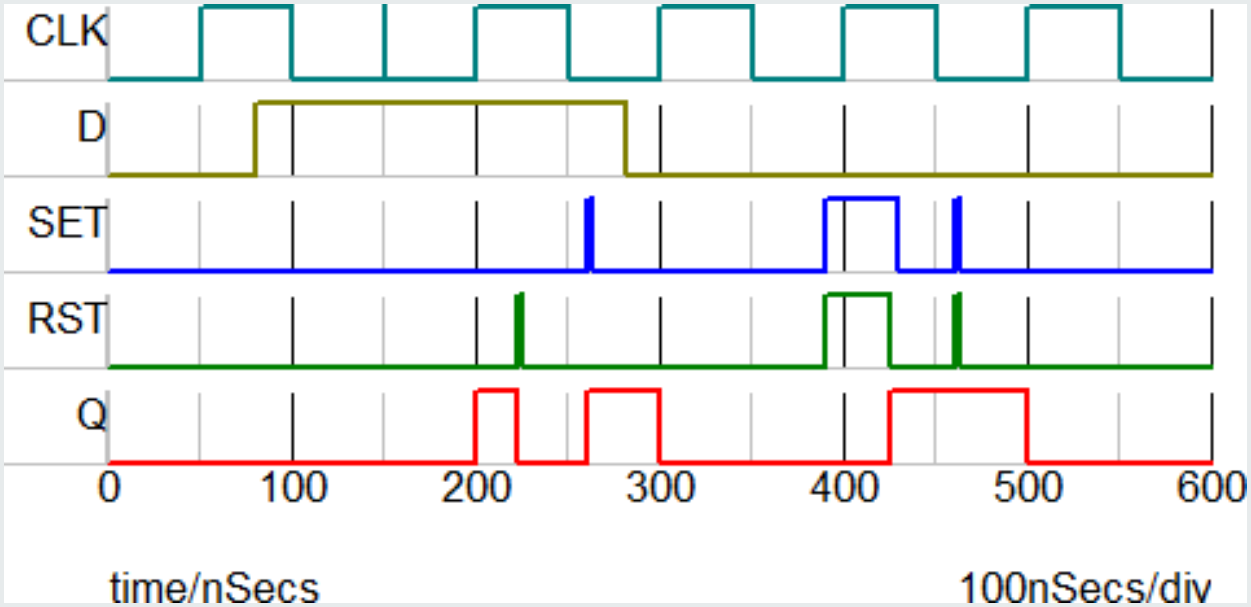
# Behaviour summary

- The output Q takes the value of the input D at the active clock edge after time $t_P$
- The input D must be stable for at least a time $t_S$ before the clock edge
- The input D must be stable for at least a time $t_H$ after the clock edge (automatically guaranteed by propagation time)
- Violations of $t_S$ and $t_H$ result in malfunction!

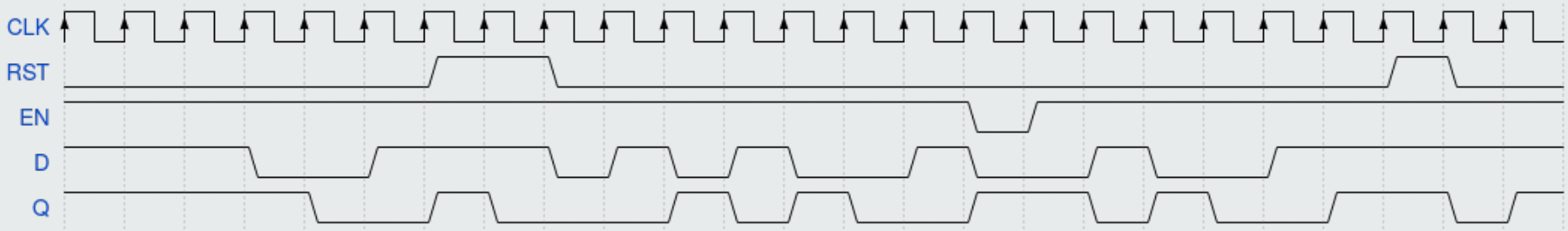
# The Flip-flop Set/Reset types

- Asynchronous
  - Output Q responds IMMEDIATELY (after small $t_P$) to *assertions* of S and R
- Synchronous
  - Output Q responds to S/R assertions present before the clock edge (before $t_S$)

# The FLIP-FLOP with asynchronus reset



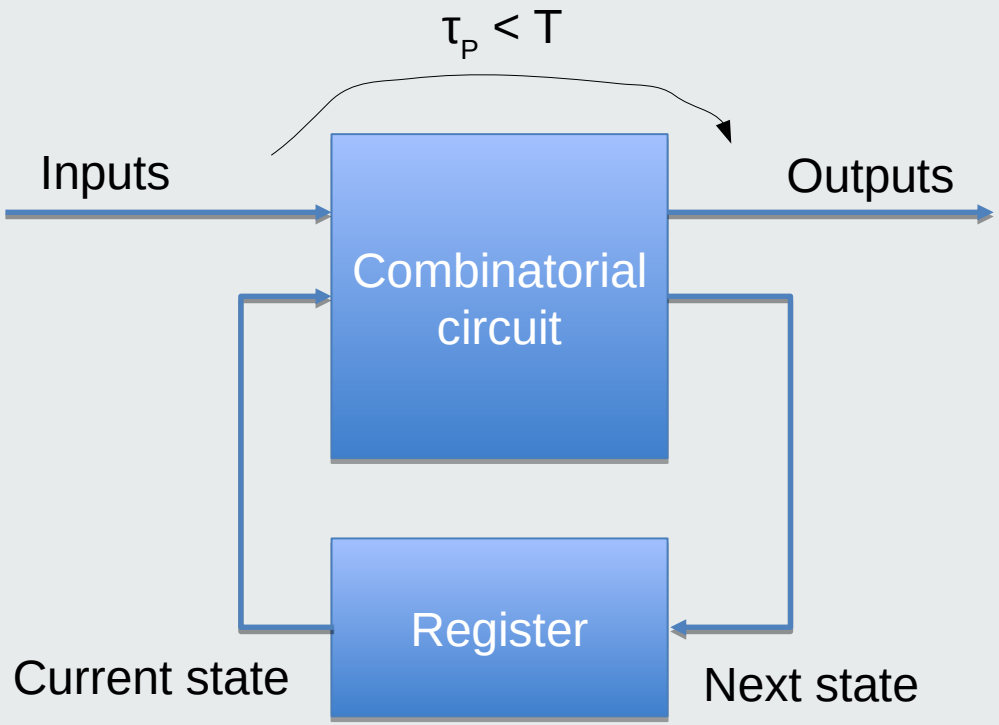
# The FLIP-FLOP with synchronous reset and enable



# So let’s learn Verilog! Or NOT!

- The **iob-lib** repo has gone public as of today!
- The **iob-lib** repo contains
  - Useful Verilog **macros** to avoid writing the same Verilog description over and over again
  - Useful **components** (modules) to avoid writing the same Verilog description over and over again
- Enjoy!

# Just learn 2 things!




- If <u>any</u> digital circuit is just this then you just need to learn to describe:
  - combinatorial circuits
  - registers
- Not bad, is that?

# Signal Wires

- Verilog describes circuits
- Circuits are blocks connected by wires that carry digital signals (0 or 1)
- So we need to declare and use wires
- Verilog describes two types of wires: *wire* and *reg*
- And the mess begins:
  - A *wire* is a wire but it cannot be used in all situations
  - A *reg* is a wire (not a register as the name suggests) but it cannot be used in all situations
- What a start!!!

# *wire* versus *reg*

- So when do we use wire?
  - A component output must be a wire
  - The left hand side of an *assign* statement must be a wire
  - A *resolved* signal must be a wire
- So when do we use reg?
  - In the left hand side of expressions inside a *process*
- So what is assign, resolved signal, process?
- It's a long story to be made short with **iob-lib!**