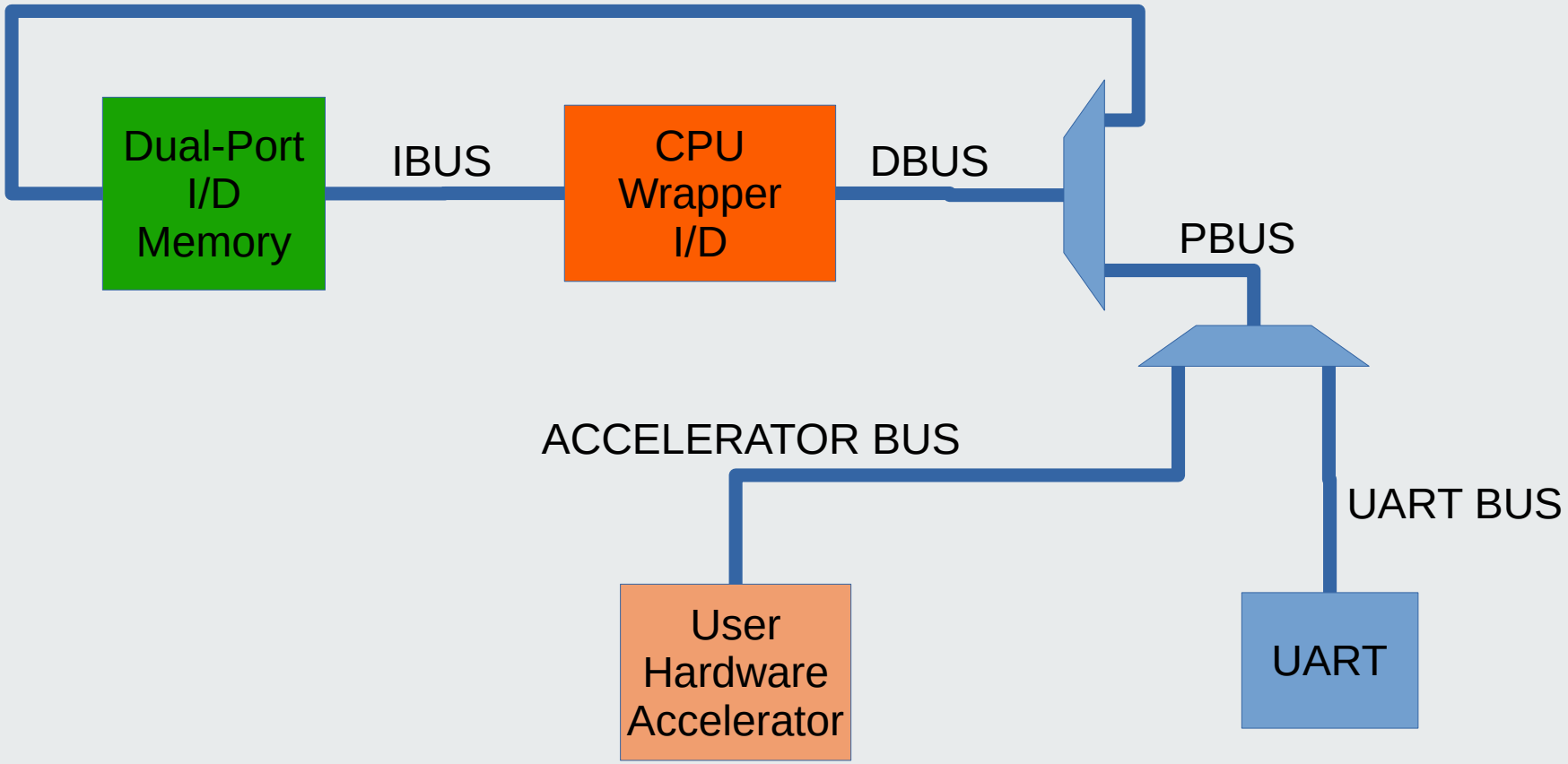
TÉCNICO LISBOA

Electronic Systems of Computers

Lecture 4: IOb-SoC Tour (continued) and first Verilog notions

IOb-SoC: recap of last lesson & more



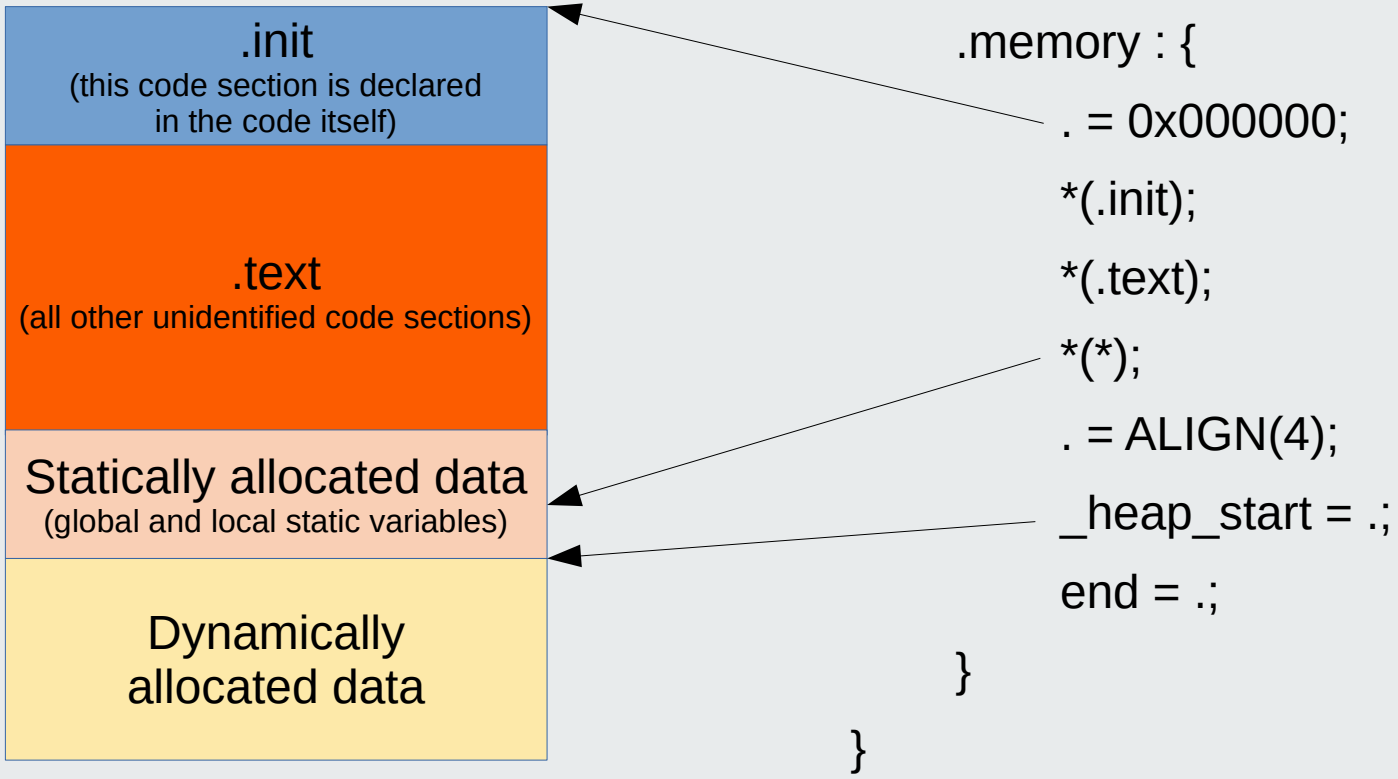
IOb-SoC: directories

- **software**: C/C++ software programs + python scripts for automation
- **hardware**: verilog hardware descriptions
- **document**: latex documents
- **submodules**: other repositories used by IOb-SoC, uses *git submodule* technology

IOb-SoC: the software directory

- Directories
  - firmware: the program to run on IOb-SoC (modify to create your own program)
  - bootloader: the resident bootloader program (do not edit)
- Files: (not intended to be edited)
  - software.mk: makefile segment for software makefiles
  - system.h: system parameters extracted from configurations entered in .mk files
  - template.lds: linker script

IOb-SoC: linker script



IOb-SoC: the firmware directory

- Files (visit & understand)
  - Makefile: firmware top build script
    - Includes software.mk
  - firmware.S: the assembly start file
    - Includes system.h
  - firmware.c: the firmware C program
    - Includes system.h

IOb-SoC: bootloader and console

- Bootloader
  - Same structure as firmware
    - it is a normal program but used to boot the system
- Console (submodules/LIB/software/python/console)
  - Communicates with the target where IPb-SoC is running
    - Board: USB/RS232 (UART)
    - Simulator: files cnsl2soc and soc2cnsl
  - May send the firmware to target repeatedly for debugging without hardware recompilation
  - Acts like a server and the target is the client asking it to print messages, send or receive files and terminate execution

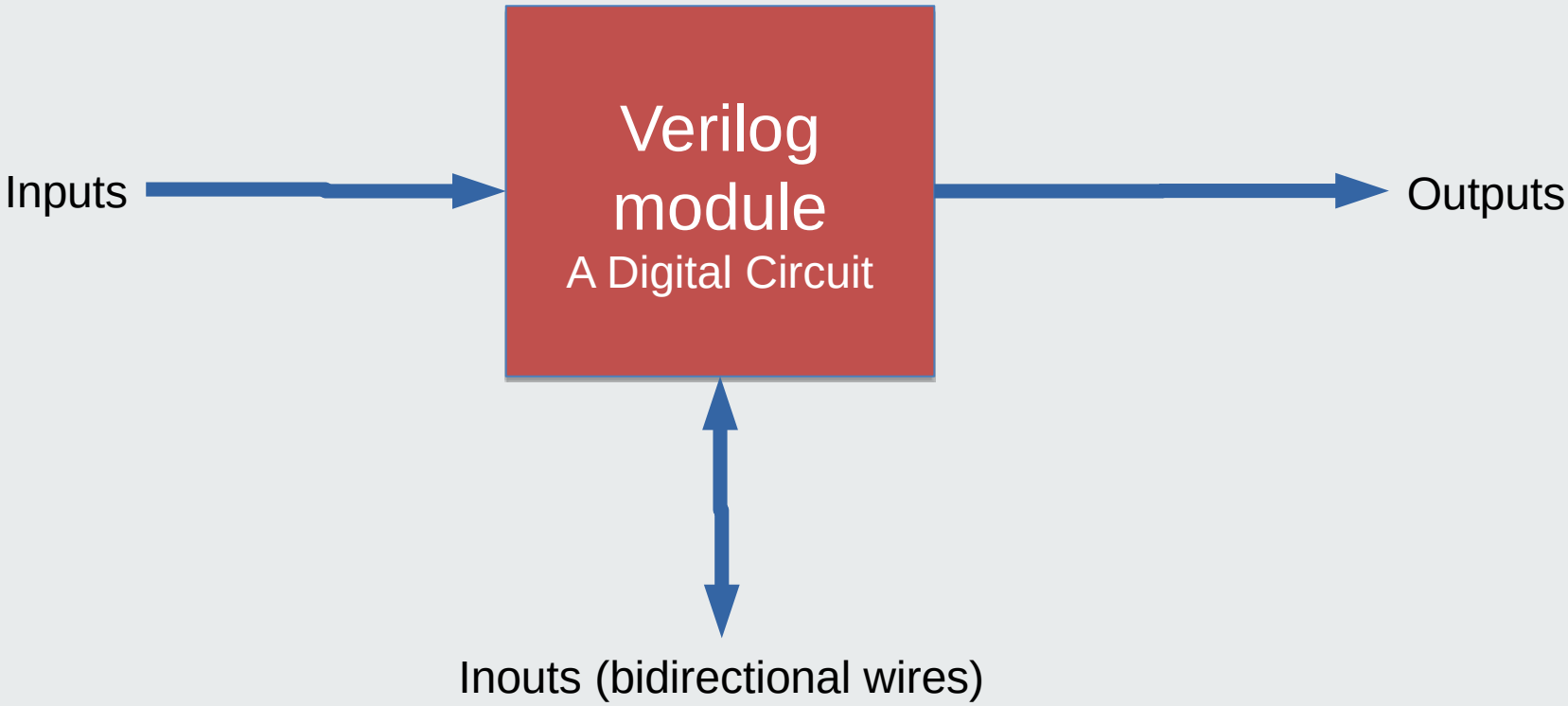
IOb-SoC: the hardware directory

- Directories
  - src: the basic system verilog files
  - testbench: Verilog testbench
  - include: the Verilog header files (shared by many files)
  - simulation: build and run directories
  - fpga: build and run directories
  - asic: build and run directories
- Files:
  - hardware.mk: makefile segment for hardware makefiles (simulation, fpga and asic)

IOb-SoC: the hardware/src directory

- Contains Verilog modules
- Verilog is a hardware DESCRIPTION language
- Verilog is not a programming language
- We are describing circuits not procedures
- We use procedures to describe circuits

Verilog module (a digital circuit)



Verilog module (a real circuit)

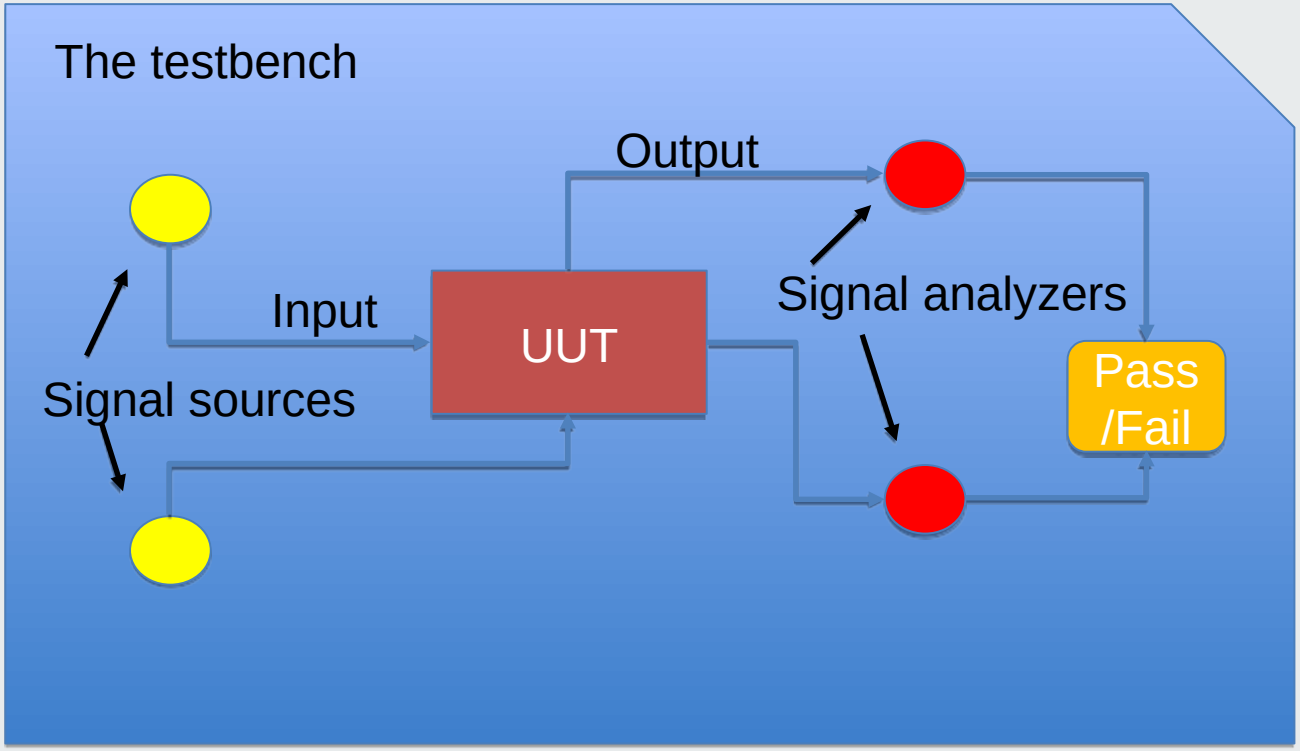
```
module my_peripheral
  (
   input                        clk,
   input                        rst,

   // native bus
   input                                    valid,
   input    [`SRAM_ADDR_W-3:0] addr,
   input    [`DATA_W-1:0]           wdata, //used for booting
   input    [`DATA_W/8-1:0]         wstrb, //used for booting
   output   [`DATA_W-1:0]           rdata,
   output                                   ready
);

//description of module goes here

endmodule
```

The testbench directory



Verilog testbench (not a real circuit!)

Used in Verilog simulation!

```
module my_testbench;

//description of test hardware

//instance of the Unit Under Test

endmodule
```

Verilog header files

```
`include “my_header_file.vh”
// this like the C language #include

module my_testbench;

//description of test hardware

//instance of the Unit Under Test

endmodule
```

IOb-SoC: the simulation directory

- Contains simulation build and run directories, one directory per simulator
  - Only the icarus simulator will be used
    - The icarus directory only has a simulation makefile (visit)
- Files:
  - the simulation.mk makefile segment (visit)
  - The waves.gtkw input file for the gtkwave program

IOb-SoC: the document directory

- Contains document build directories, one directory per document type
  - pb: product brief
    - Makefile
    - .tex files
    - .pdf file for easy viewing
    - Private figures directory
  - presentation: slide presentation
- Contains the figures directory
  - Figures that are shared among different documents

IOb-SoC: the submodules directory

- System submodules
  - CPU
  - CACHE
  - UART
- A git submodule is a pointer to another git repository
  - Allows for modular and distributed development
  - Easy to add with the git submodule add <url> path/to/the/submodule
  - Easy to remove: git rm path/to/the/submodule
- The way the system collects hardware and software components from the submodules and assembles them in the system is taught in the next class.