



Electronics Systems (938II)

Lecture 1.2

Modern Electronic Systems (intro) – SystemVerilog (intro) and simulation

Intro to HDLs

- HDLs

→ Parallel ⇒ Different sub-processes working in parallel

- Concurrent language (not sequential !!!)
 - Description of HW circuits

- VHDL → .vhd
- Verilog → .v
- SystemVerilog (extension of Verilog) → .sv

→ extension of Verilog that extends the verification

Intro to HDLs

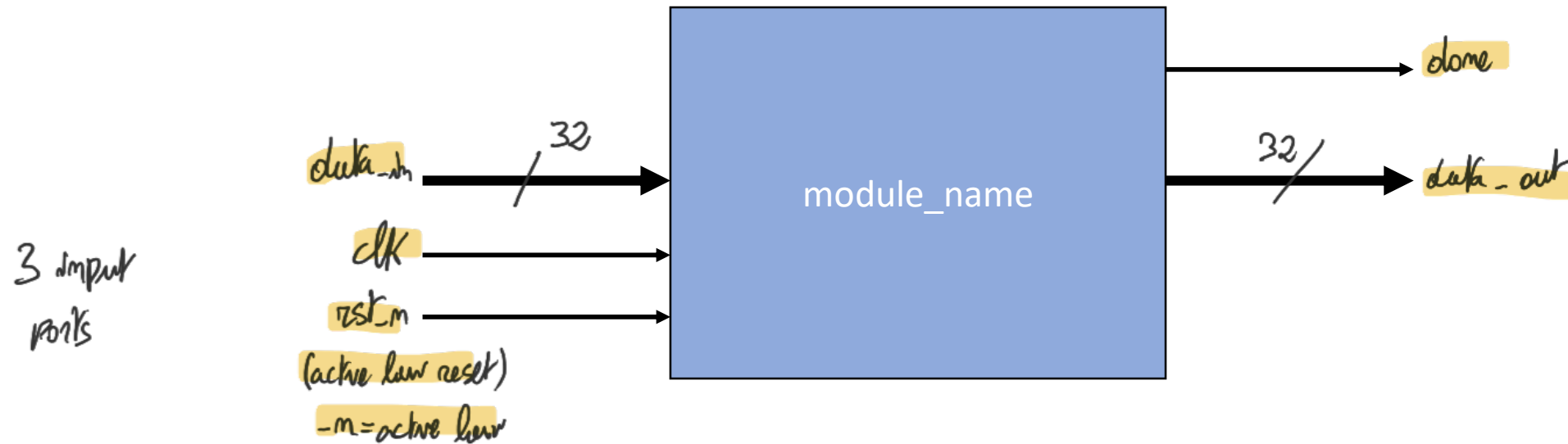
- HDLs
 - Concurrent language (not sequential !!!)
 - Description of HW circuits
 - VHDL → .vhd
 - Verilog → .v
 - **SystemVerilog** (extension of Verilog) → .sv

Intro to circuit design in SystemVerilog

- The **main building block** in SystemVerilog (SV) is the **module** *circuit is a module*
 - Like Verilog
 - Represents an electronic circuit or a subpart thereof
 - Defined by **ports** (input and output)

Intro to circuit design in SystemVerilog

- Example of SV module



Intro to circuit design in SystemVerilog

- Example of SV module

```

module module_name (
    input clk
    ,input rst_n
    ,input [31:0] data_in
    ,output done
    ,output reg [31:0] data_out
);
    // Logic description
endmodule
  
```

Handwritten annotations:
 - **keyword module** (pointing to `module`)
 - **name of the module** (pointing to `module_name`)
 - **one word no spaces** (next to `module_name`)
 - **delimiter** (pointing to the opening parenthesis `(`)
 - **list of ports** (bracketed next to the port declarations)
 - **final delimiter => almost all statements in verilog** (pointing to the closing parenthesis `)` and semicolon `;`)
 - `// Logic description` (in green)

Intro to circuit design in SystemVerilog

- Example of SV module

```
module module_name (  
    input clk  
    ,input rst_n  
    ,input [31:0] data_in  
    ,output done  
    ,output reg [31:0] data_out  
);  
  
    // Logic description  
  
endmodule
```

Intro to circuit design in SystemVerilog

- Example of SV module

```
module module_name (  
    input clk  
    ,input rst_n  
    ,input [31:0] data_in  
    ,output done  
    ,output reg [31:0] data_out  
);  
  
    // Logic description  
  
endmodule
```


Intro to circuit design in SystemVerilog

- Example of SV module

- List of ports

- input or output (also must be declared)

Syntax: **<polarity>** [**<type>**] [**<bit width and range>**] **<name>**

$\rightarrow \begin{cases} \text{Reg} \\ \text{Wire} \end{cases} \Rightarrow \text{two main types, Wire is the default type. The type declaration is optional. Reg can be used for sequential logic. Inputs are always wire.}$

$\xrightarrow{\text{bit range}} [\text{initial index: final index}] \Rightarrow \text{bit width} = |\text{final} - \text{initial}| + 1$
 $1 = \text{default}$

- Ports must be separated by a **comma (,)**

* For indexes every natural number (0) and either ascending or descending order. $[31:0] \rightarrow \text{width} = 32 \text{ bits}$

Intro to circuit design in SystemVerilog

- Example of SV module
 - List of ports

- Syntax: <polarity> [<type>] [<bit width and range>] <name>

↑ arbitrary word, no spaces

- Ports must be separated by a **comma** (,)

Intro to circuit design in SystemVerilog

- Example of SV module
 - List of ports
 - Syntax: <polarity> [<type>] [<bit width and range>] <name>
 - Ports must be separated by a **comma** (,)

Intro to circuit design in SystemVerilog

- Example of SV module
 - List of ports
 - Syntax: <polarity> [<type>] [<bit width and range>] <name>
 - Ports must be separated by a **comma** (,)

Intro to circuit design in SystemVerilog

- Example of SV module
 - List of ports
 - Syntax: <polarity> [<type>] [<bit width and range>] <name>
 - Ports must be separated by a **comma** (,)

Intro to circuit design in SystemVerilog

- Example of SV module
 - List of ports
 - Syntax: <polarity> [<type>] [<bit width and range>] <name>
 - Ports must be separated by a **comma** (,)

Intro to circuit design in SystemVerilog

- Example of SV module
 - List of ports
 - Syntax: <polarity> [<type>] [<bit width and range>] <name>
 - Ports must be separated by a **comma** (,)

Intro to circuit design in SystemVerilog

- Example of SV module
 - List of ports
 - Syntax: <polarity> [<type>] [<bit width and range>] <name>
 - Ports must be separated by a **comma** (,)

Intro to circuit design in SystemVerilog

- Example of SV module
 - List of ports
 - Syntax: <polarity> [<type>] [<bit width and range>] <name>
 - Ports must be separated by a **comma** (,)

Intro to circuit design in SystemVerilog

- Example of SV module

```
module module_name (  
    input clk  
    ,input rst_n  
    ,input [31:0] data_in  
    ,output done  
    ,output reg [31:0] data_out  
);
```

// Logic description

endmodule

} body of the module.

↓
Comments like C

↓
Close the envelope of the module

Intro to circuit design in SystemVerilog

- Example of SV module

```
module module_name (  
    input clk  
    ,input rst_n  
    ,input [31:0] data_in  
    ,output done  
    ,output reg [31:0] data_out  
);  
  
    // Logic description  
  
endmodule
```

Intro to circuit design in SystemVerilog

- Modules (or sub-modules) in SV can be simulated
 - Debug and (functional) verification
 - RTL simulation *(first representation, abstract. Doesn't contain info about physical properties)*
 - Zero-delay: no information about timing of logic gates (propagation delays, ...)
 - Dedicated tools
 - Example: Modelsim = used in the industry
- We are going to use Modelsim, ...

Intro to circuit design in SystemVerilog

..., so you can download it, for free, from the Intel FPGA Download Center

<https://www.intel.com/content/www/us/en/collections/products/fpga/software/downloads.html?q=modelsim&s=Relevancy>

Filter by

Title
 ModelSim-Intel® FPGAs Standard Edition Software Version 20.1.1

Downloads

Linux Software

Windows Software

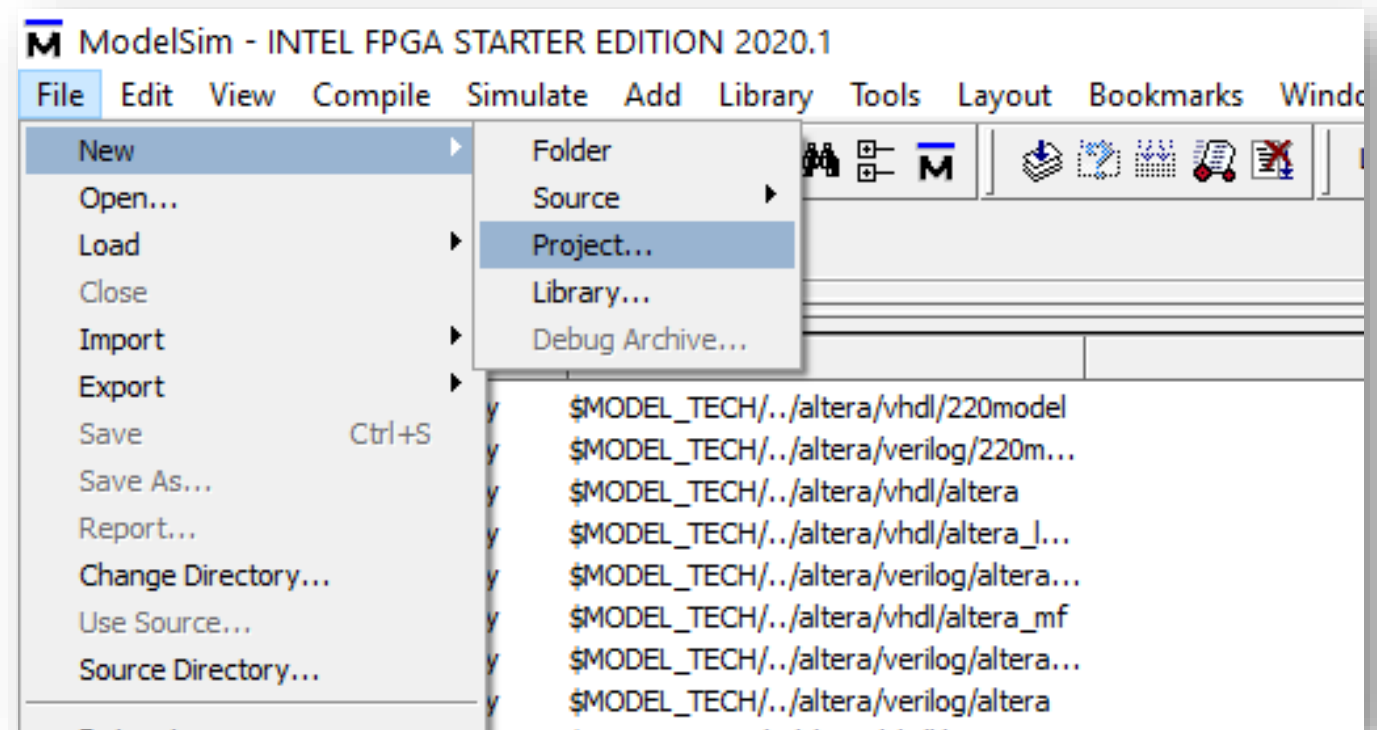
ModelSim Software

ModelSim-Intel® FPGA Edition (includes Starter Edition)

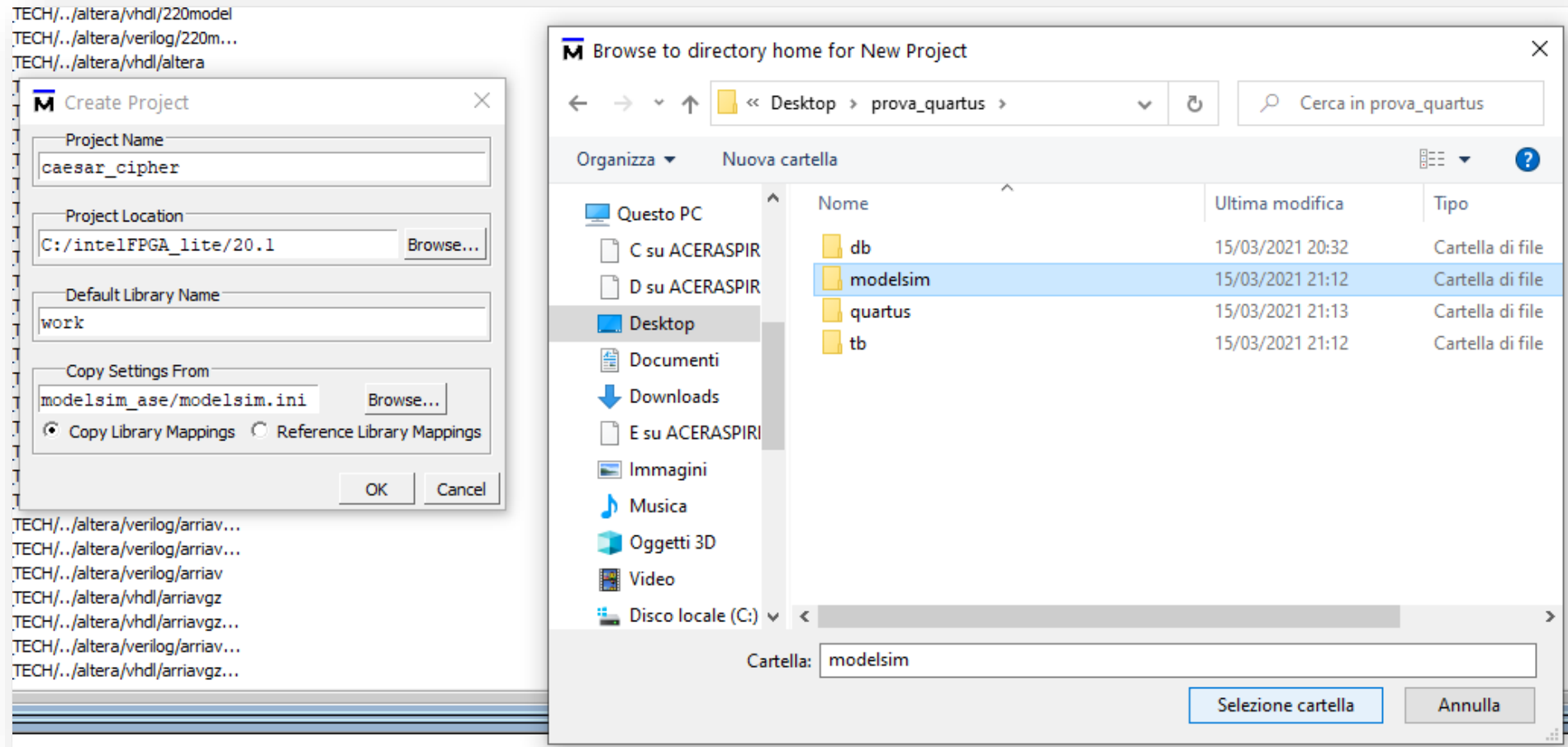
Download
ModelSimSetup-20.1.1.720-linux.run

Quick tutorial on Modelsim

- Run Modelsim and create a new project
 - File > New > Project...

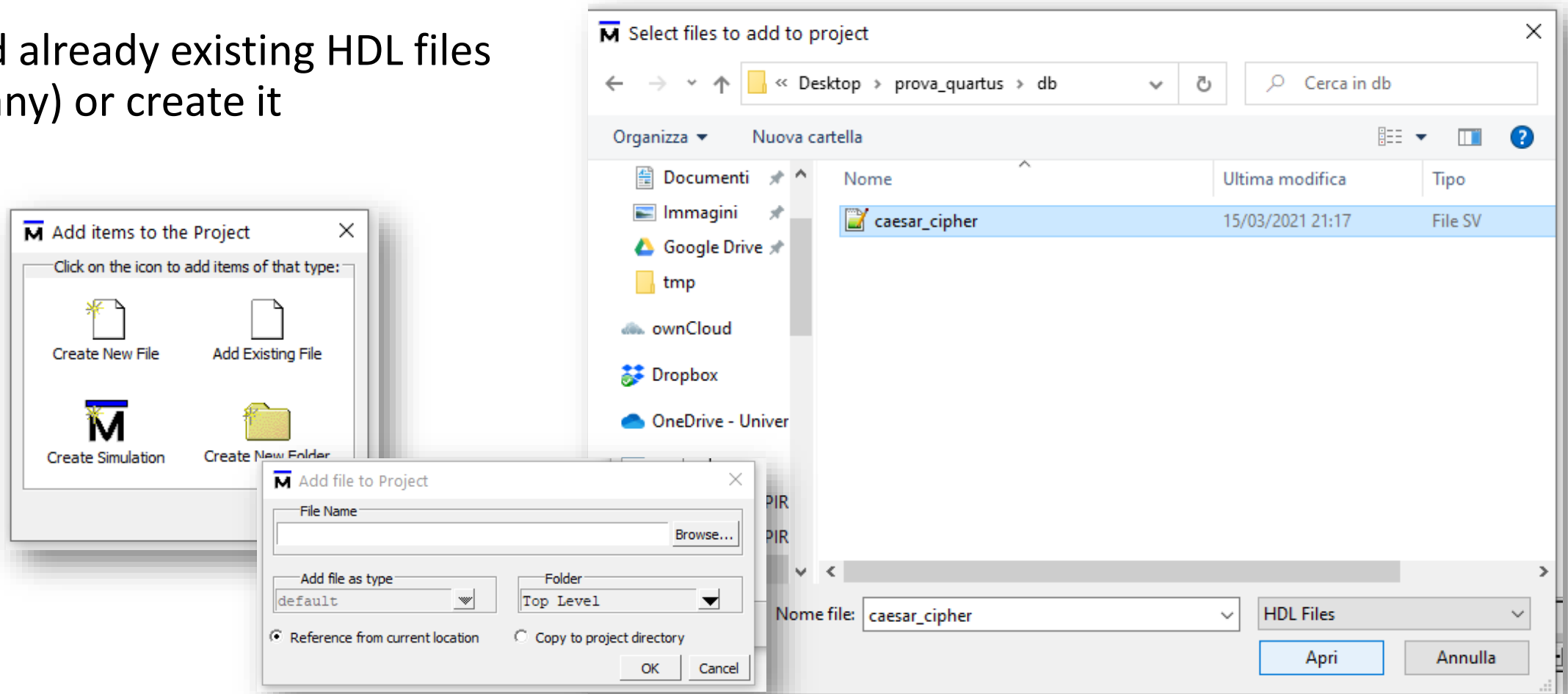


Quick tutorial on Modelsim

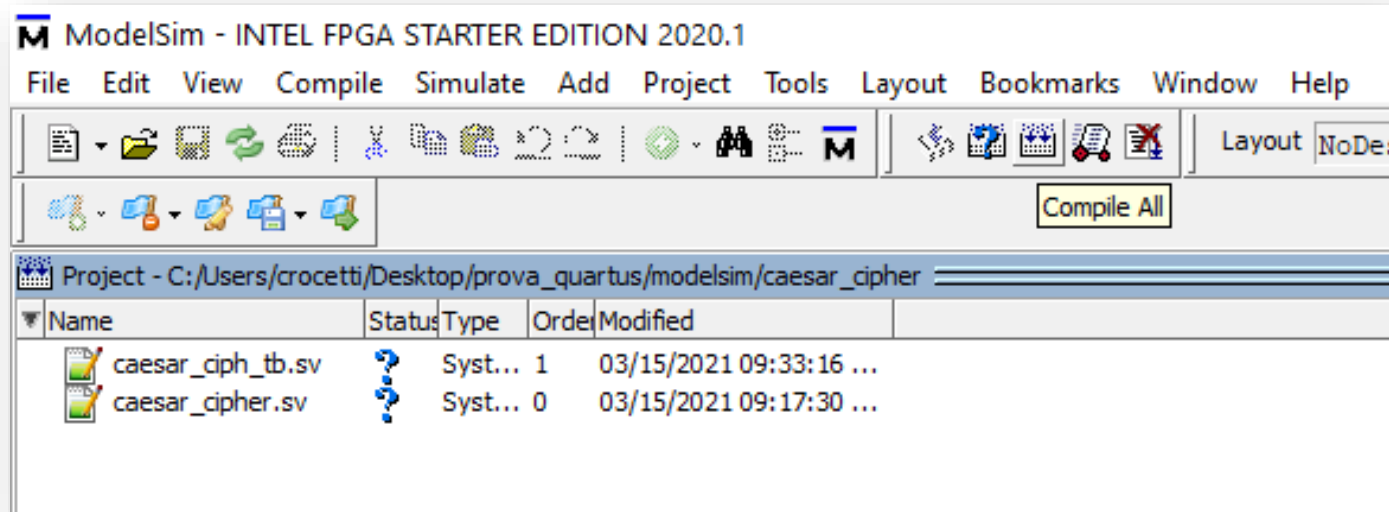


Quick tutorial on Modelsim

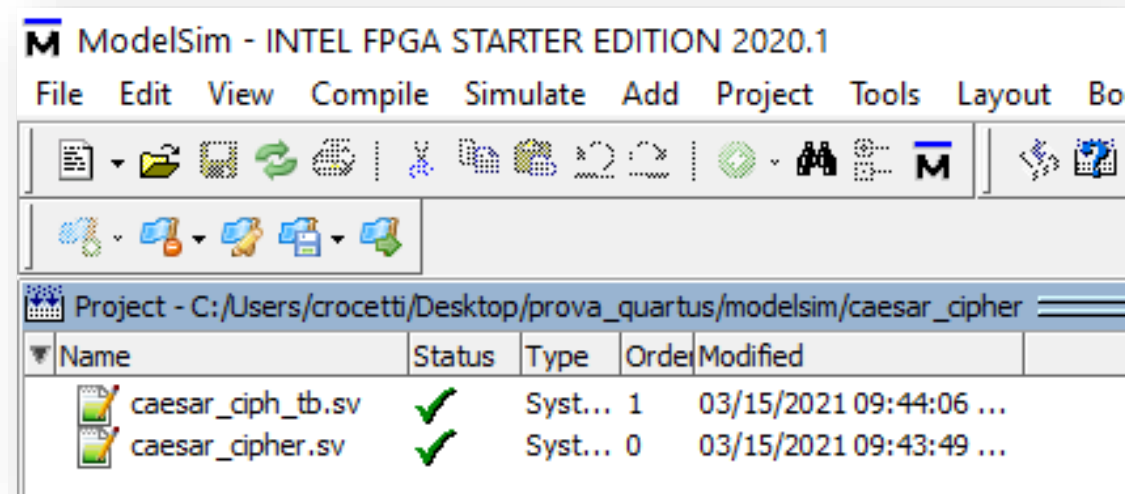
- Add already existing HDL files (if any) or create it



Quick tutorial on Modelsim

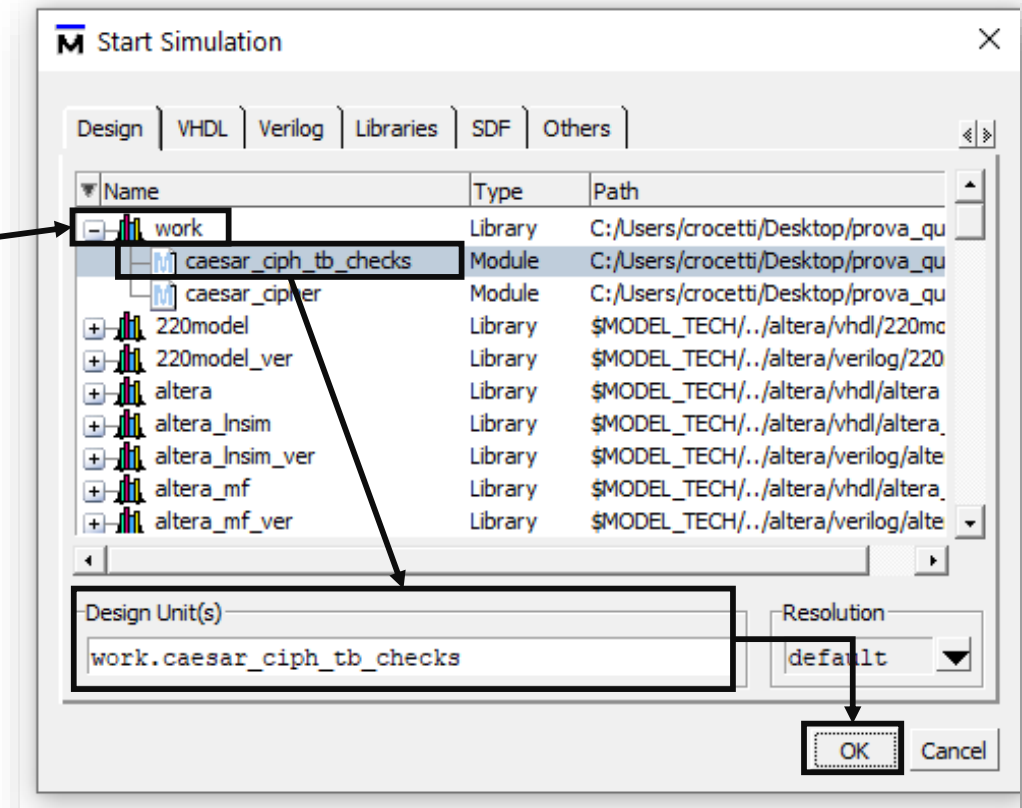
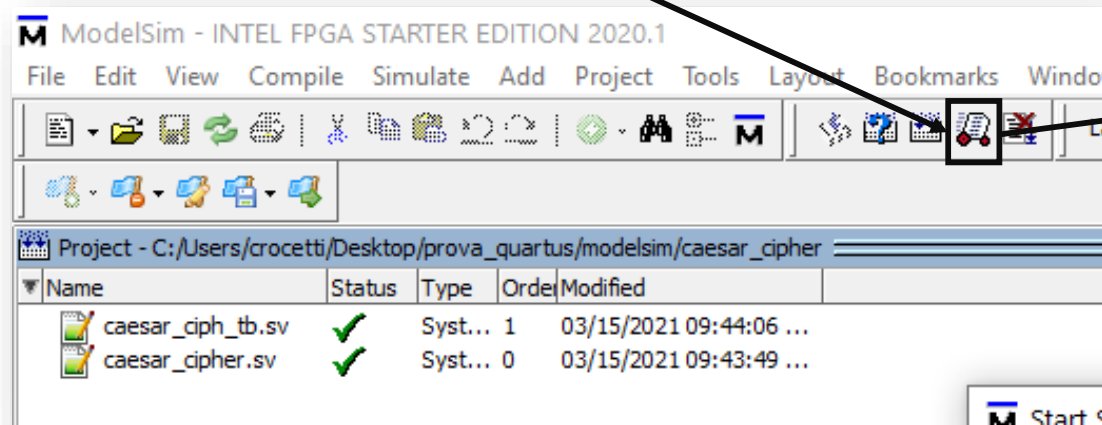


Quick tutorial on Modelsim

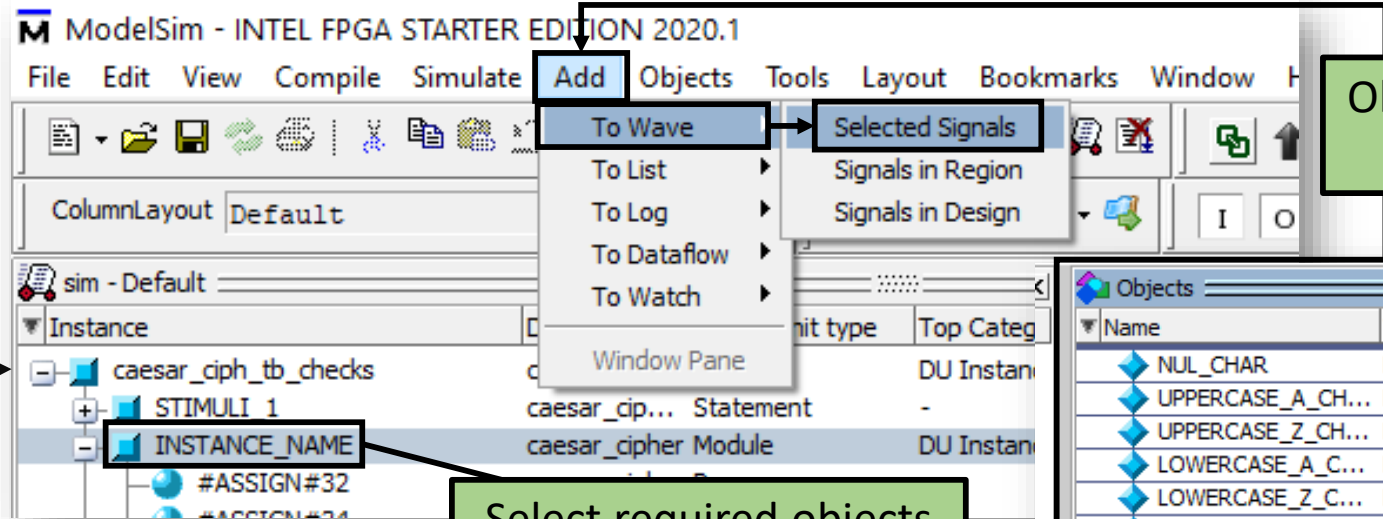


Quick tutorial on Modelsim

Launch the simulation



Quick tutorial on Modelsim



Objects can be added to a Waveform

Select required objects

Hierarchy of the selected unit for simulation

Objects

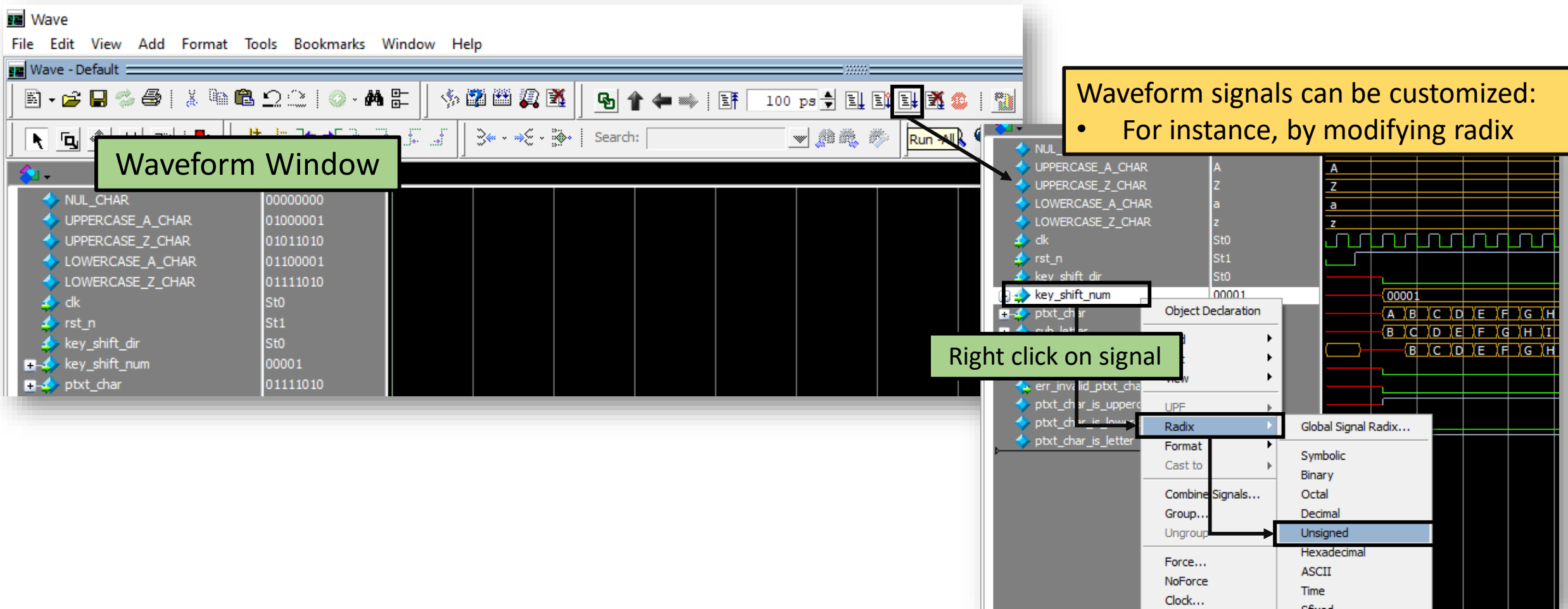
Name	Value	Kind	Mode
NUL_CHAR	Not L...	Para...	Internal
UPPERCASE_A_CH...	Not L...	Para...	Internal
UPPERCASE_Z_CH...	Not L...	Para...	Internal
LOWERCASE_A_C...	Not L...	Para...	Internal
LOWERCASE_Z_C...	Not L...	Para...	Internal
clk	Not L...	Net	In
rst_n	Not L...	Net	In
key_shift_dir	Not L...	Net	In
key_shift_num	Not L...	Net	In
ptxt_char	Not L...	Net	In
ctxt_char	Not L...	Pack...	Out
err_invalid_key_shi...	Not L...	Net	Out
err_invalid_ptxt_ch...	Not L...	Net	Out
ptxt_char_is_uppe...	Not L...	Net	Internal
ptxt_char_is_lower...	Not L...	Net	Internal
ntxt_char is letter	Not L...	Net	Internal

Processes (Active)

Name	Type (filtered)	State
------	-----------------	-------

Handwritten note: force a 0, force b 0. Run himeStamp [ms,ms,us,ns] ^{1 femtosecond}

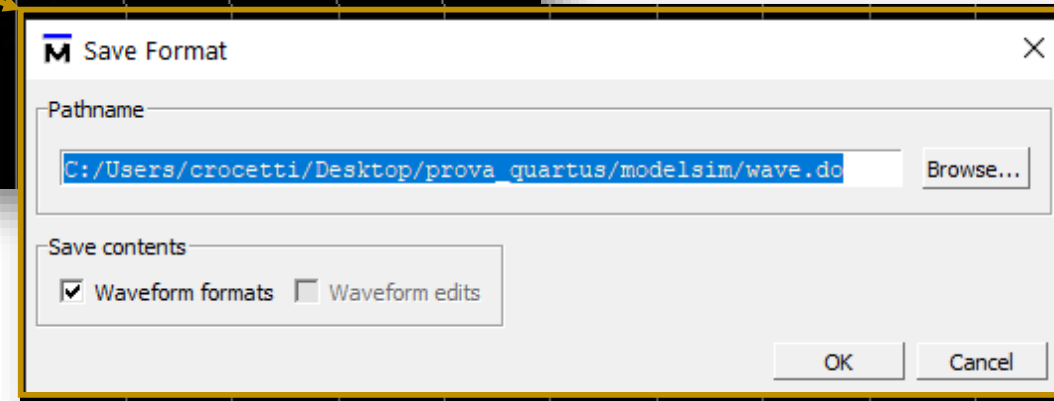
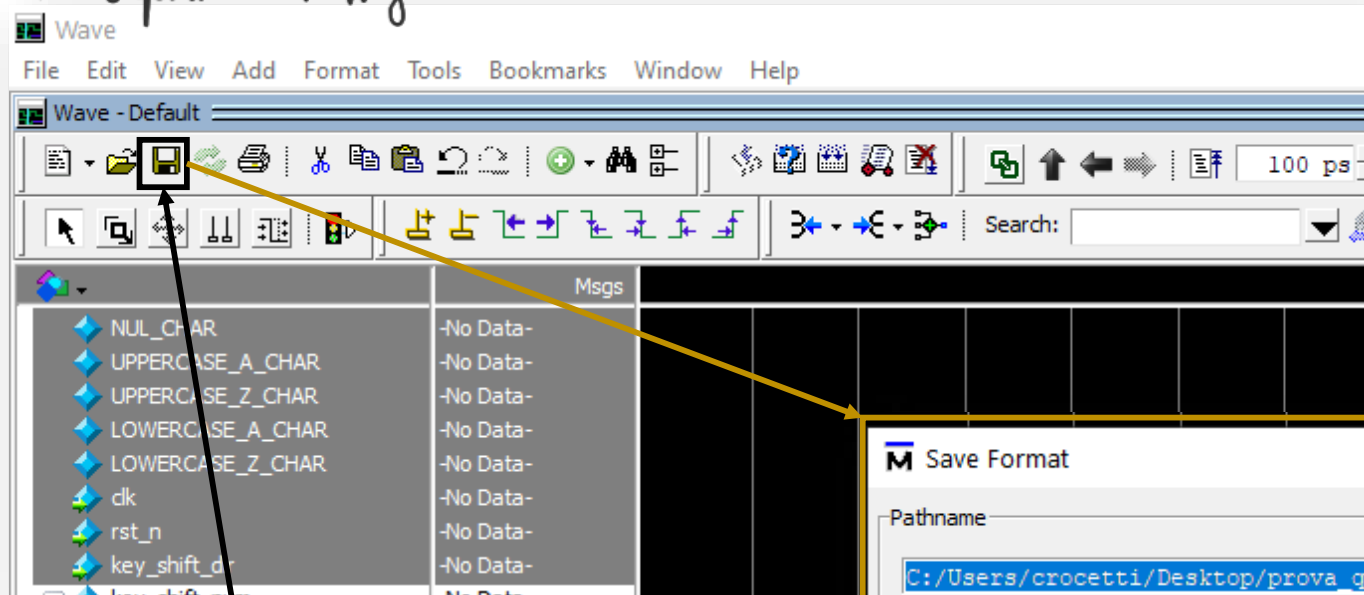
Quick tutorial on Modelsim



The screenshot shows the Modelsim Waveform Window. The left pane lists signals: NUL_CHAR, UPPERCASE_A_CHAR, UPPERCASE_Z_CHAR, LOWERCASE_A_CHAR, LOWERCASE_Z_CHAR, clk, rst_n, key_shift_dir, key_shift_num, and ptxt_char. The right pane shows a waveform for these signals. A green box labeled "Waveform Window" points to the left pane. A yellow box contains the text: "Waveform signals can be customized: • For instance, by modifying radix". A green box labeled "Right click on signal" points to a right-click context menu for the "key_shift_num" signal. The menu includes options like "Object Declaration", "Format", "Cast to", "Combine Signals...", "Group...", "Ungroup", "Force...", "NoForce", and "Clock...". The "Radix" option is highlighted, and a sub-menu shows "Global Signal Radix...", "Symbolic", "Binary", "Octal", "Decimal", "Unsigned" (highlighted), "Hexadecimal", "ASCII", and "Time".

Open waveform only after
stopping simulation before running

Quick tutorial on Modelsim

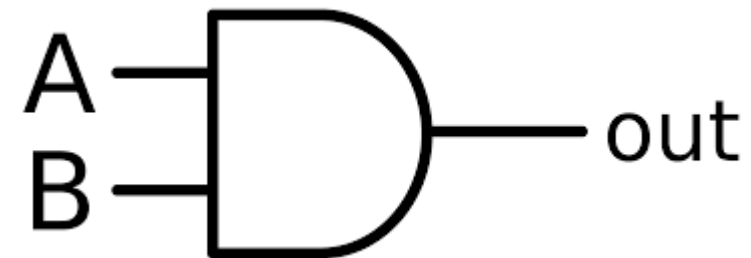


Waveform can be saved, to be restored
later (customizations, ...)

Quick tutorial on Modelsim

- Exercise

```
module and_gate (  
    input  a  
    ,input  b  
    ,output y  
);  
  
    assign y = a & b;  
  
endmodule
```



Quick tutorial on Modelsim

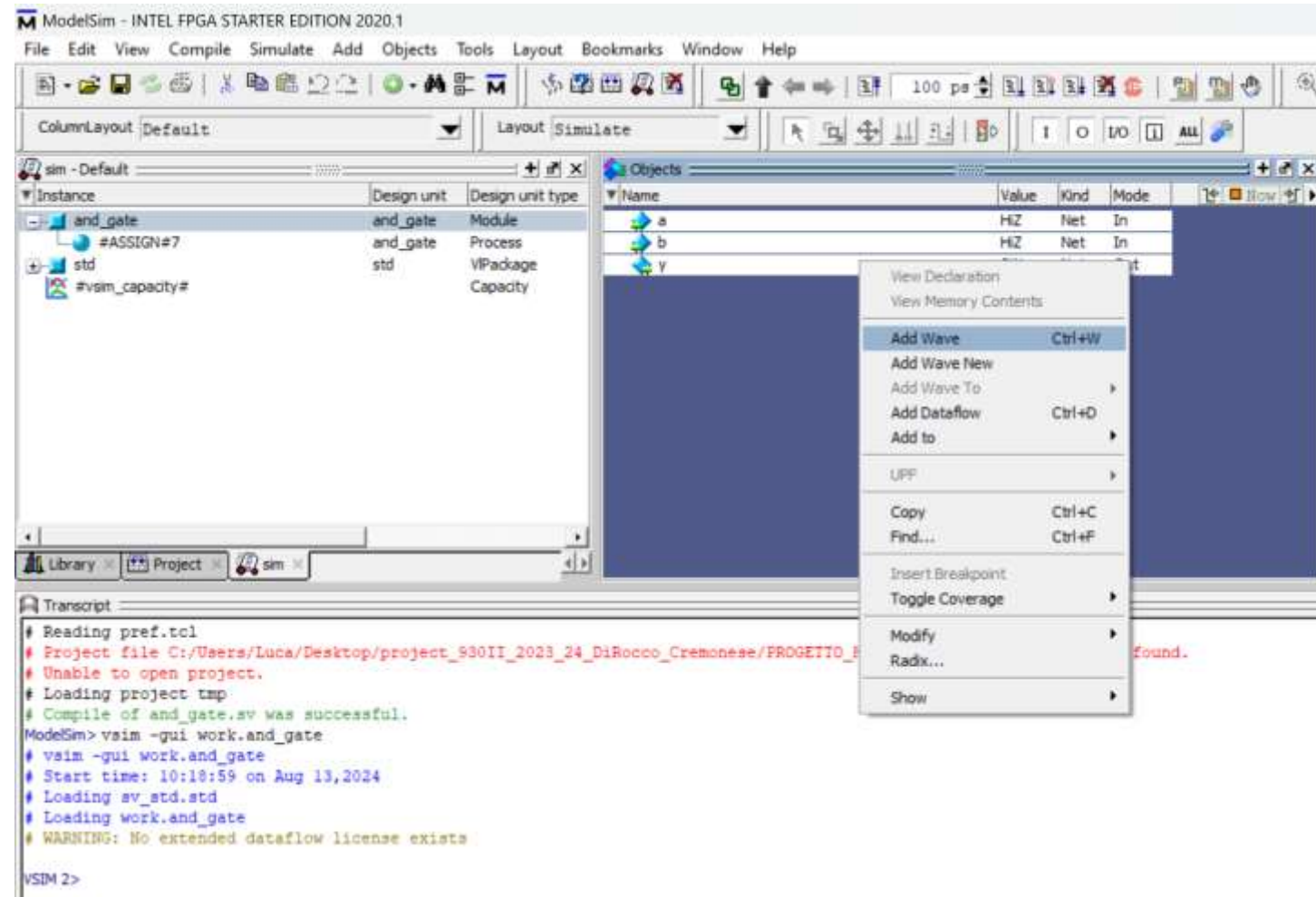
- Exercise
 1. Create the SV file (and_gate.sv) and fill with the previous example
 2. Create a Modelsim project adding the SV file at the previous step
 3. Compile and launch the simulation selecting the **and_gate** unit/module
 4. Now run manually the simulation using the **force** and **run** commands in the **Transcript** Tab, as indicated in the next slide

Quick tutorial on Modelsim

- Simulation (manual)
 - Using the Transcript Tab (\approx terminal/command line interface)
 - The **force** command apply logic values to ports (and internal signals) of module(s)
 - We are going to use it to apply stimuli on the input ports of the simulate unit/module
 - Syntax: **force** <input port name> <logic value>
 - The **run** command advances the simulation by the specified number of timesteps
 - When launching the simulation, the initial simulation time is 0 (seconds)
 - Syntax: **run** <timestep>
 - We are going to see the outputs using the **waveform** (at the same time)

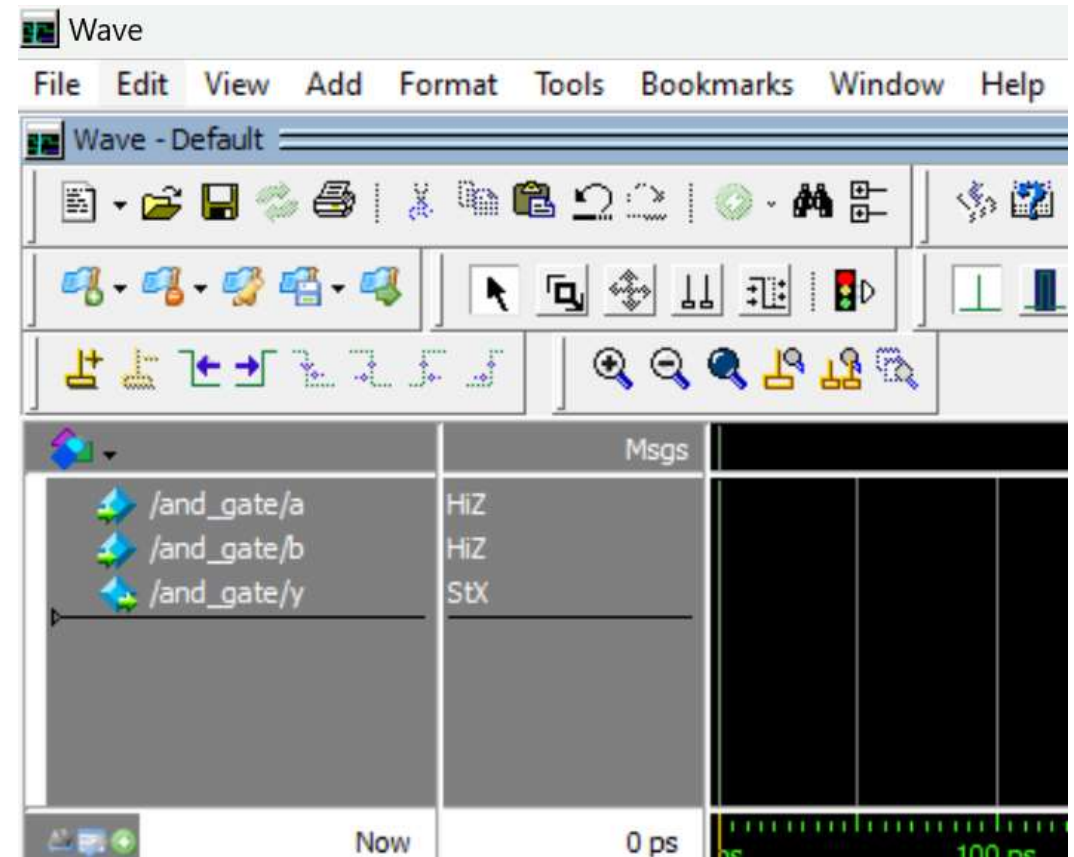
Quick tutorial on Modelsim

- Simulation (manual)
 1. Add the and_gate unit signals to the waveform



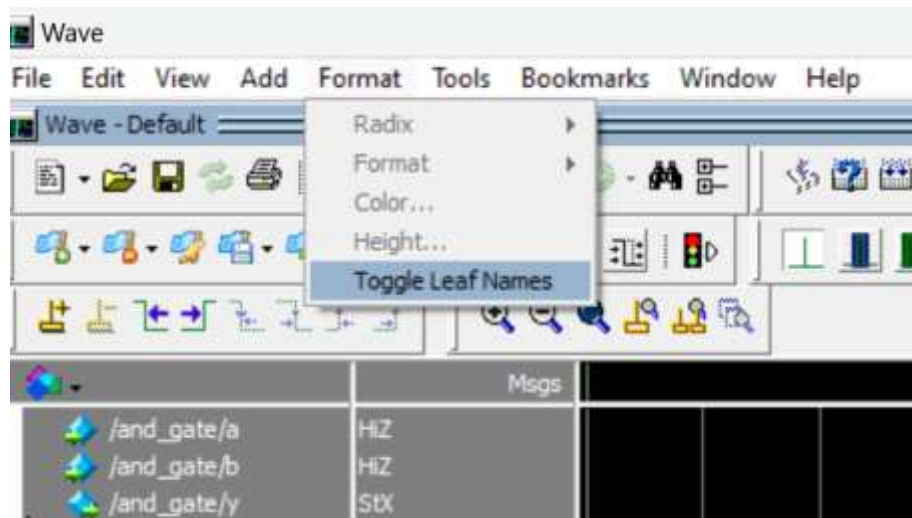
Quick tutorial on Modelsim

- Simulation (manual)
 1. Add the and_gate unit signals to the waveform
 - You will get something like this
 - Note the simulation time
 - 0 ps = zero picoseconds



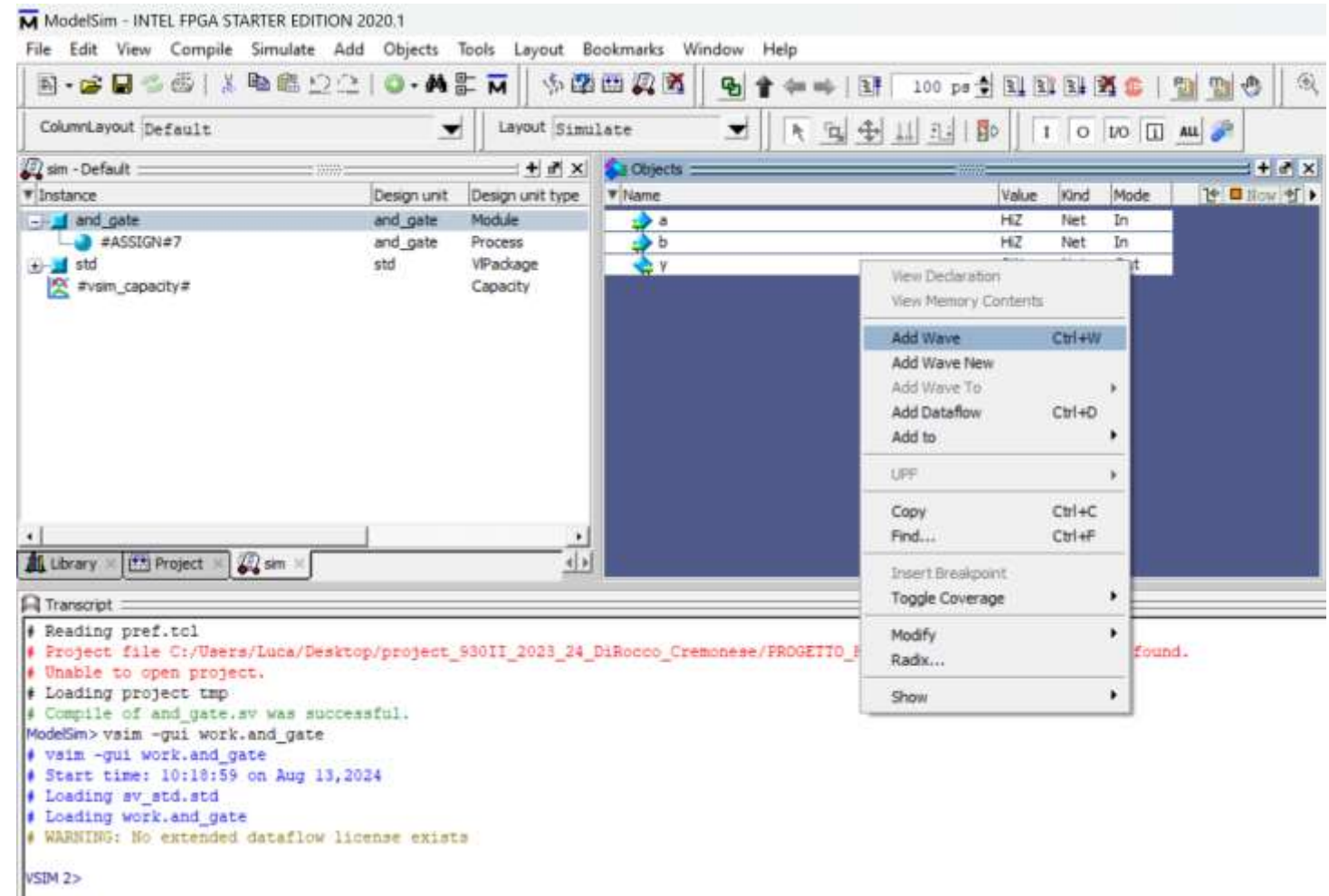
Quick tutorial on Modelsim

- Simulation (manual)
 1. Add the and_gate unit signals to the waveform
 - You can remove the hierarchical path from the displayed signals
 - More readable
 - Format > Toggle Leaf Names



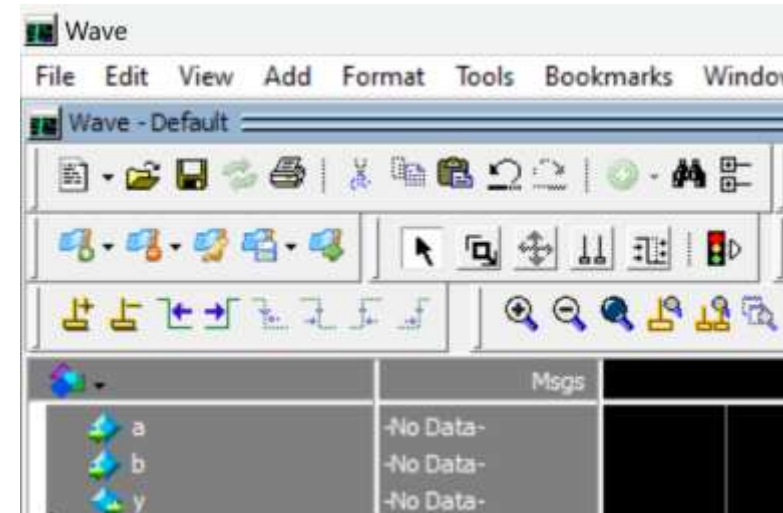
Quick tutorial on Modelsim

- Simulation (manual)
 2. Move to the **Transcript** Tab and execute the **force** and **run** commands to simulate the unit/module



Quick tutorial on Modelsim

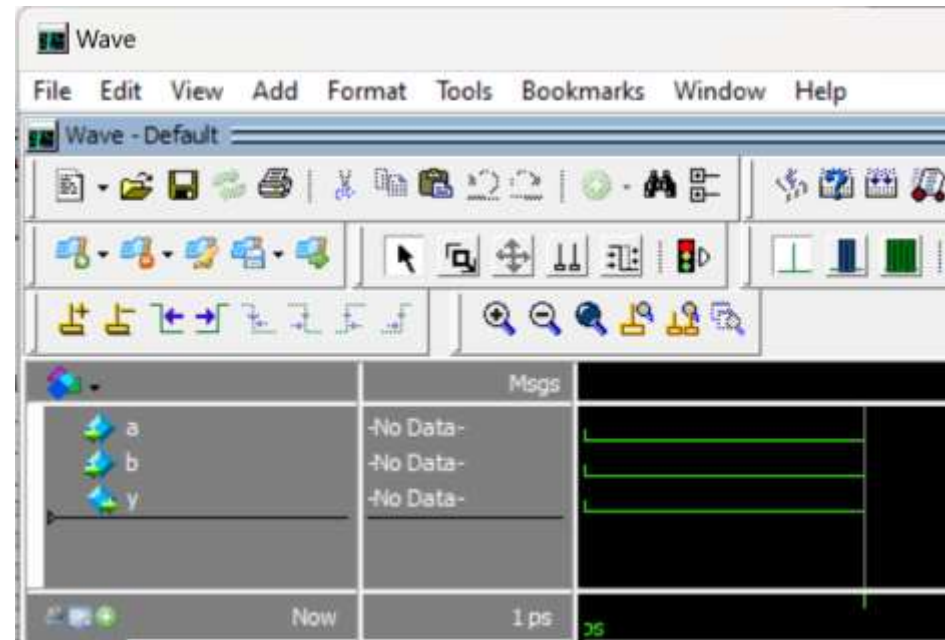
- Simulation (manual)
 2. Move to the **Transcript** Tab and execute the **force** and **run** commands to simulate the unit/module
 - Example:



- force a 0 → set input port 'a' to 0 (otherwise it is unknown, X, by default)
- force b 0 → set input port 'b' to 0 (otherwise it is unknown, X, by default)

Quick tutorial on Modelsim

- Simulation (manual)
 2. Move to the **Transcript** Tab and execute the **force** and **run** commands to simulate the unit/module
 - Example:



- run 1 → advance the simulation by 1 time unit (ps or ns, by default: depends on the tool settings)
- Note: 'y' is determined by the unit/module (thanks to the description specified in the SV code)

Quick tutorial on Modelsim

- Simulation (manual)
 - Try different combinations of 'a' and 'b' and for each combination run the simulation for some timestep
 - Debug/verification by checking the truth table of the AND gate

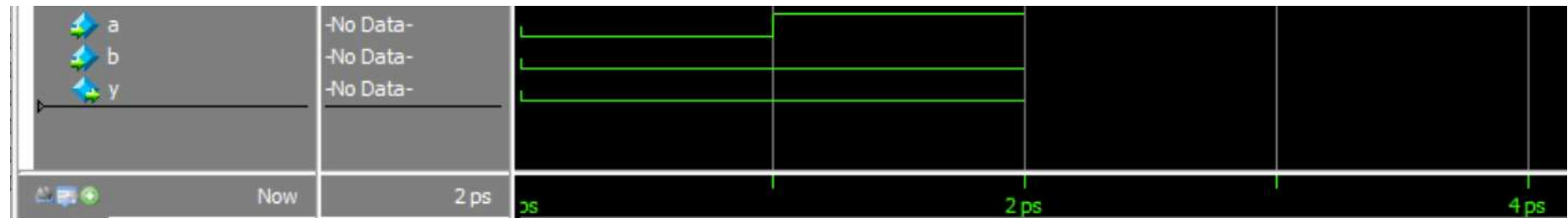


Quick tutorial on Modelsim

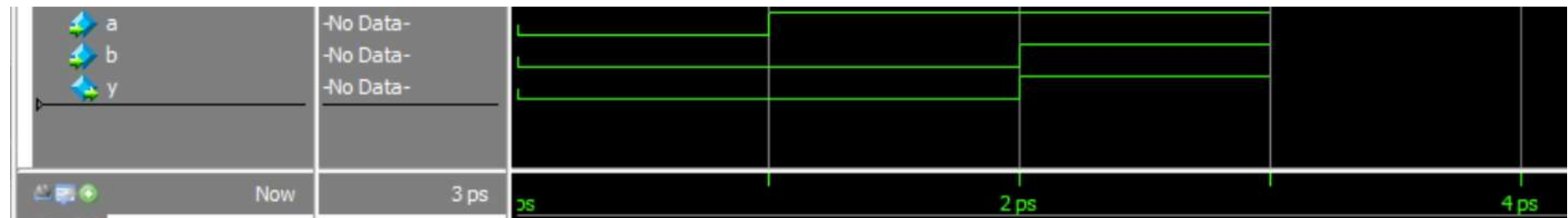
```
VSIM 3> force a 0
VSIM 4> force b 0
VSIM 5> run 1
```



```
VSIM 13> force a 1
VSIM 14> run 1
```



```
VSIM 15> force b 1
VSIM 16> run 1
```



Quick tutorial on Modelsim

- You can find all the files about this exercise in the dedicated folder on the Team of the course
 - File > Electronics Systems module > Crocetti > Exercises > 1.2
 - Please, start reading the README.txt file
- I also included two files (.do files) to automate the waveform and the simulation
 - wave.do, sim.do
 - Refer to README.txt file
 - However, you can run them using again the Transcript Tab and the **do** command
 - After having launched the simulation

```
VSIM 24> do wave.do  
VSIM 25> do sim.do
```



Thank you for your attention

Luca Crocetti
(luca.crocetti@unipi.it)