

# VHDL and Quartus prime introduction

MC. Martin González Pérez



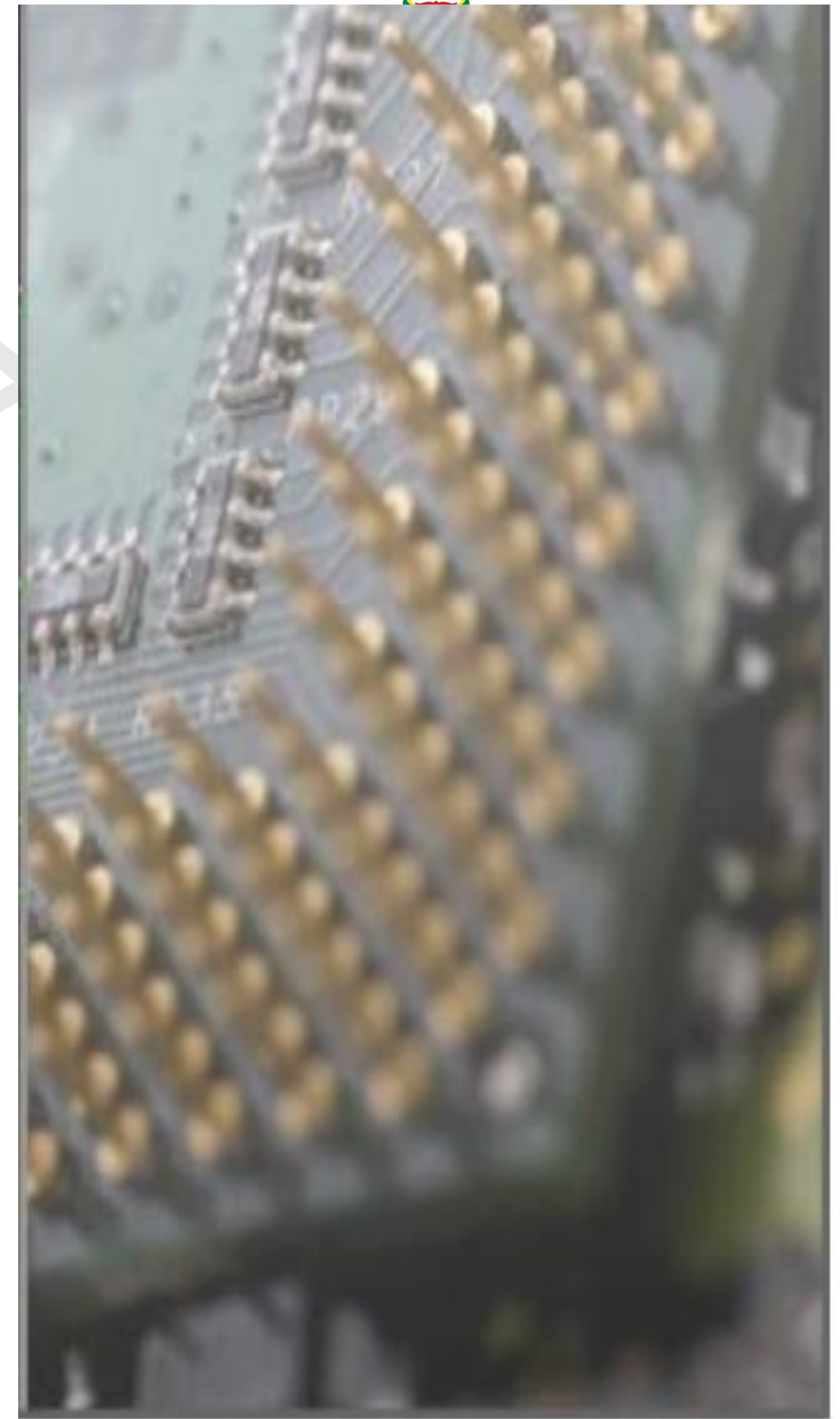
# Table of contents

- FPGA design flow.
- What is Verilog?
  - Modules
- An introduction to Quartus Prime Lite.
  - Creating a design.
  - Simulation.
  - Programing FPGA.

CONFIDENTIAL



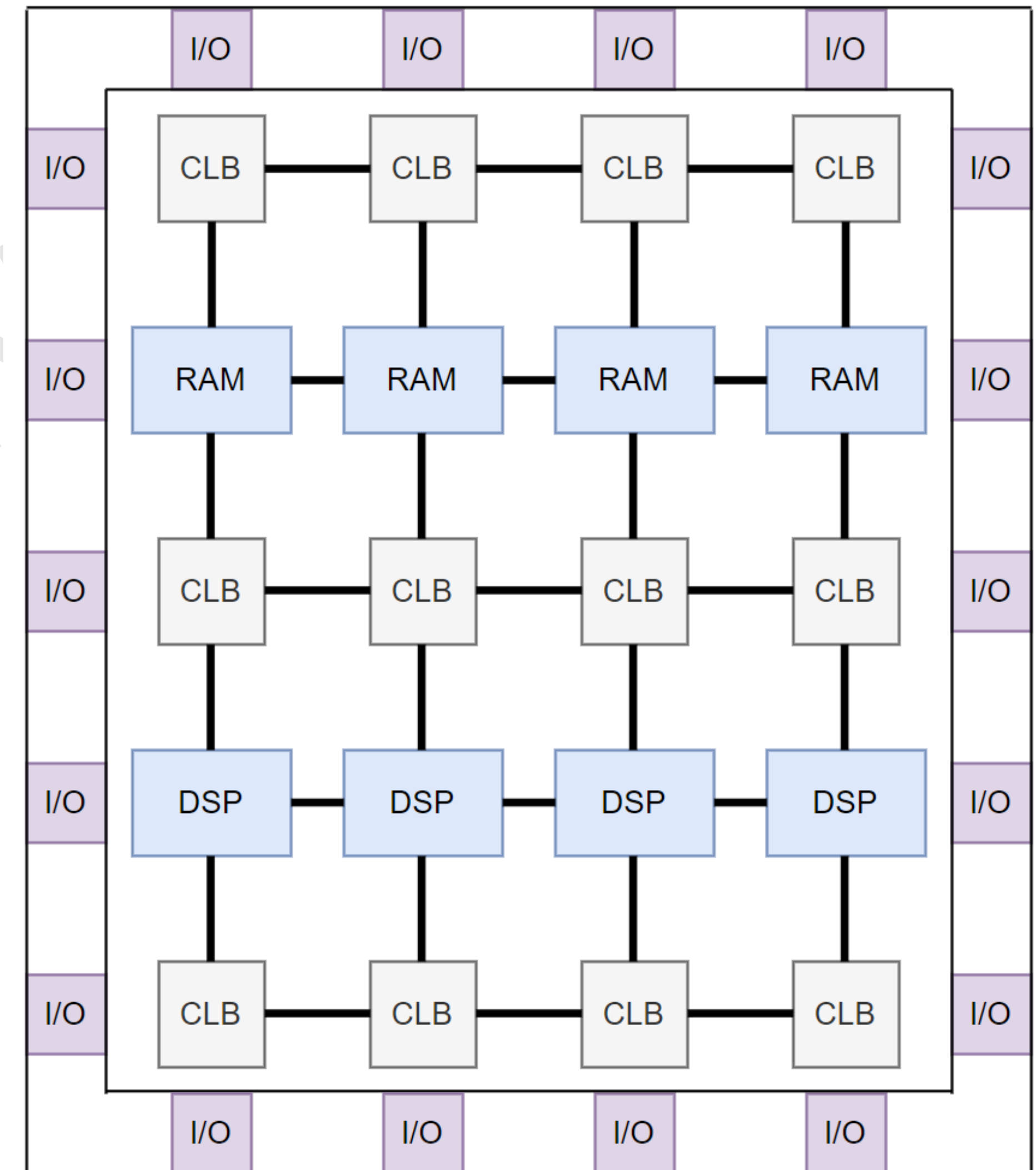
# FPGA design flow



# What is a FPGA?

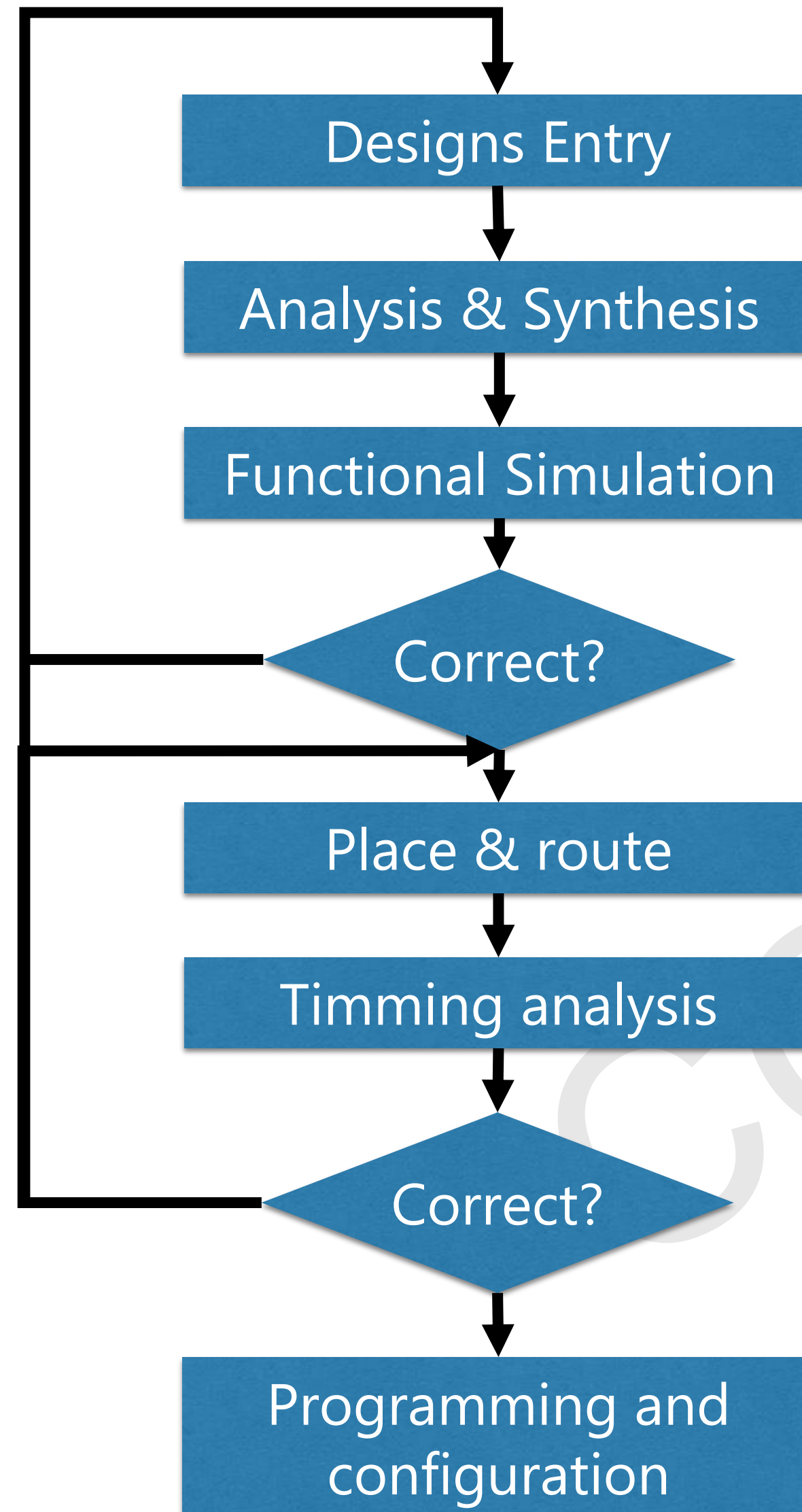
FPGAs are devices based on reconfigurable elements that are connected through programmable interconnections.

- Can implement any digital design, design space is limited by FPGA capacity.
- Modern devices also include custom blocks as DSP blocks, embedded memory and processors.





# FPGA design flow



**Design Entry:** Desired circuit is specified, in this case by means of a schematic diagram.

**Synthesis:** Entered design is mapped into a circuit that consist of logic elements (LEs).

**Functional Simulation:** Design is tested to verify its functional correctness.

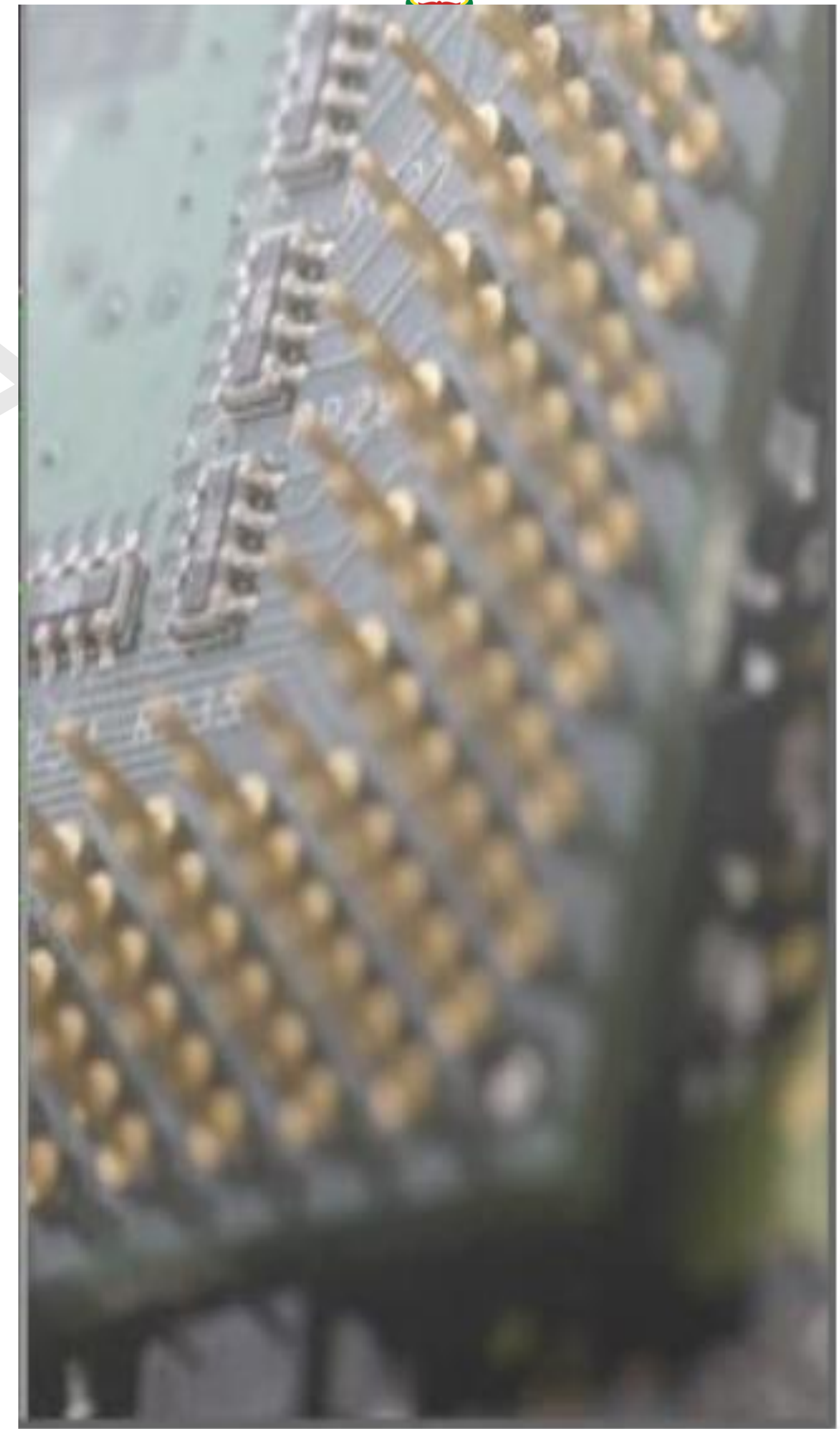
**Fitting:** CAD fitter tool determines the placement of the LEs defined in the netlist into the LEs in the actual FPGA chip.

**Timing Analysis:** Propagation delays along paths in the fitted circuit are analyzed to provide an estimate of the performance of the circuit.

**Programming and Configuration:** the designed circuit is implemented in a physical FPGA chip.



# What is Verilog HDL?

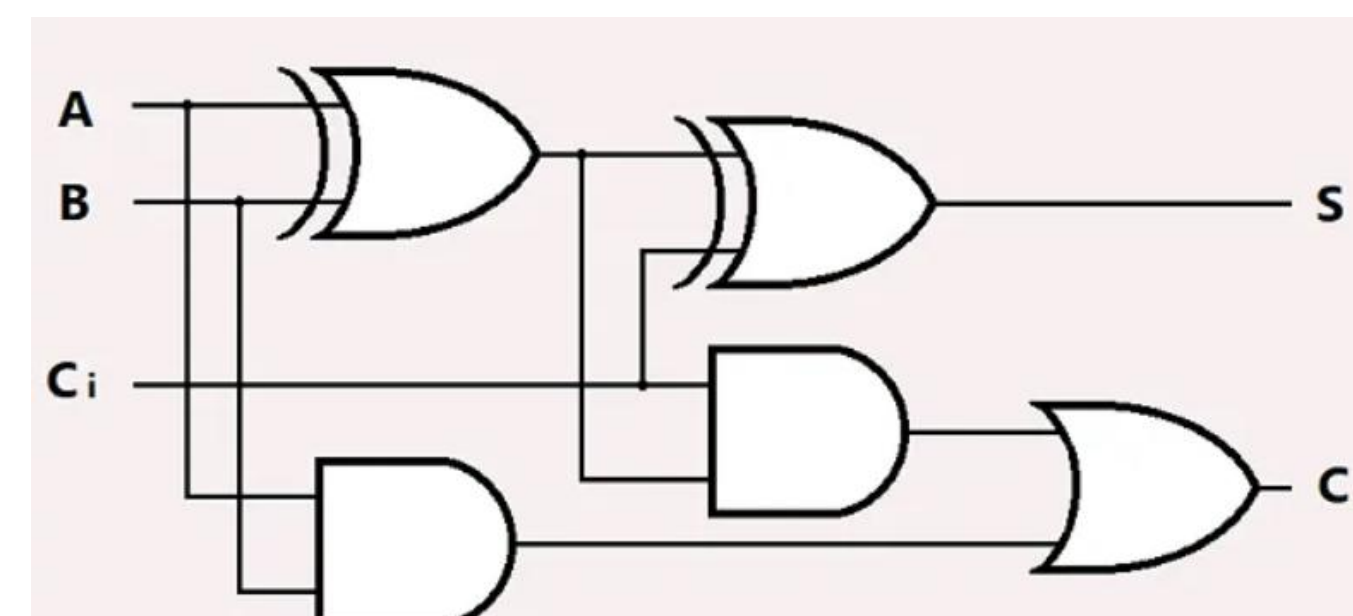
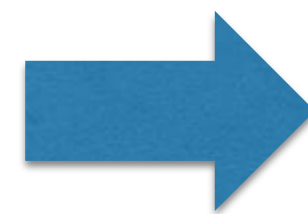
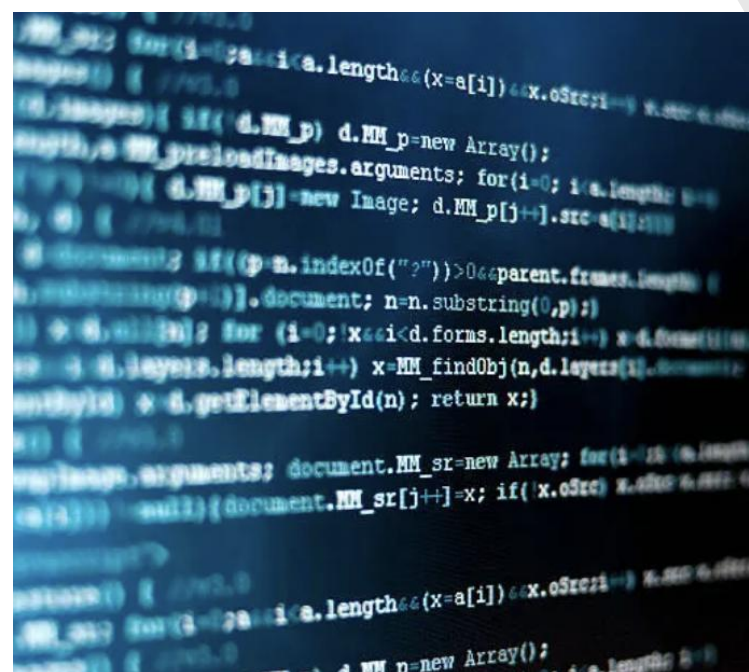




# What is Verilog?

Verilog is a hardware description language (HDL) used to specify the behavior and structure of digital circuits. Its syntax allows for modeling systems at the **gate, register, and behavioral levels**, enabling simulation, verification, and synthesis for implementation technologies such as FPGA and ASIC. It is widely used in electronic design to accurately represent the **logic and timing** of a digital system.

Verilog is not only used to describe hardware but also to verify it!!!



# HDL vs Programming Language

Verilog **IS NOT** a programming language. Verilog is designed for modeling digital circuits, **not to execute sequential instructions on a processor**. While programming languages describe algorithms that run in software, Verilog specifies hardware components and their interconnections, which **operate concurrently in parallel** rather than sequentially.



## Design tip:

For hardware design, it is recommended to create block diagrams of the hardware that meet the requirements, rather than using flowcharts or pseudocode as is done in software.



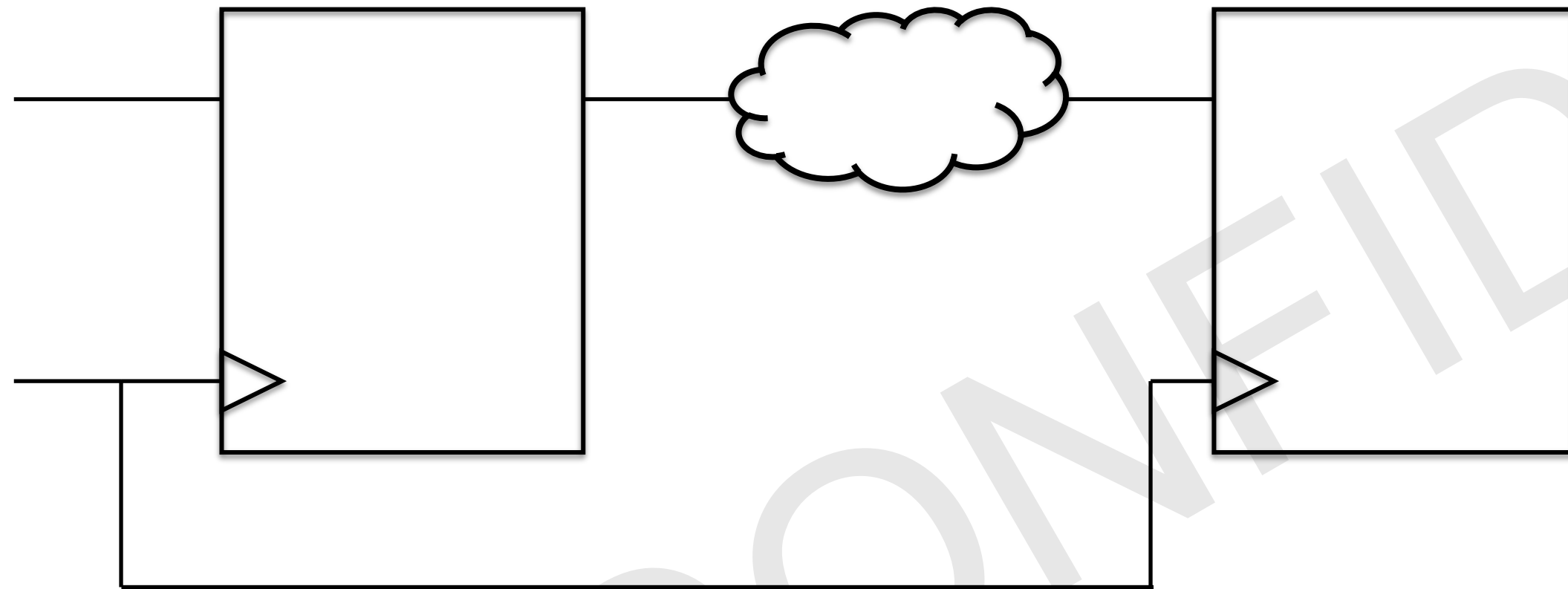
# Abstraction Levels

Verilog allows modeling systems at three abstraction levels:

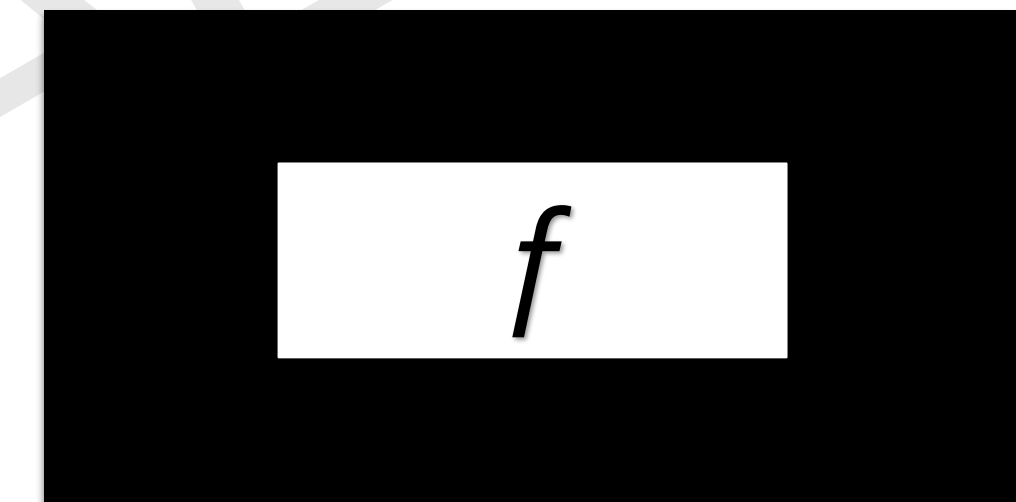
- **Behavioral:** Is used to create testbenches and verify functionality using mathematical equations while omitting timing, like software.
- **Register Transfer Level (RTL):** Is used to modeling combinational and sequential logic using **synthesizable** constructs. At this level, the timing is based on the cycles of the defined clocks.
- **Gate level (Structural):** At gate level, it is only possible to instantiate and interconnect predefined components (primitives). At this level, the timing consider the delay of gates and nets.

# Abstraction levels

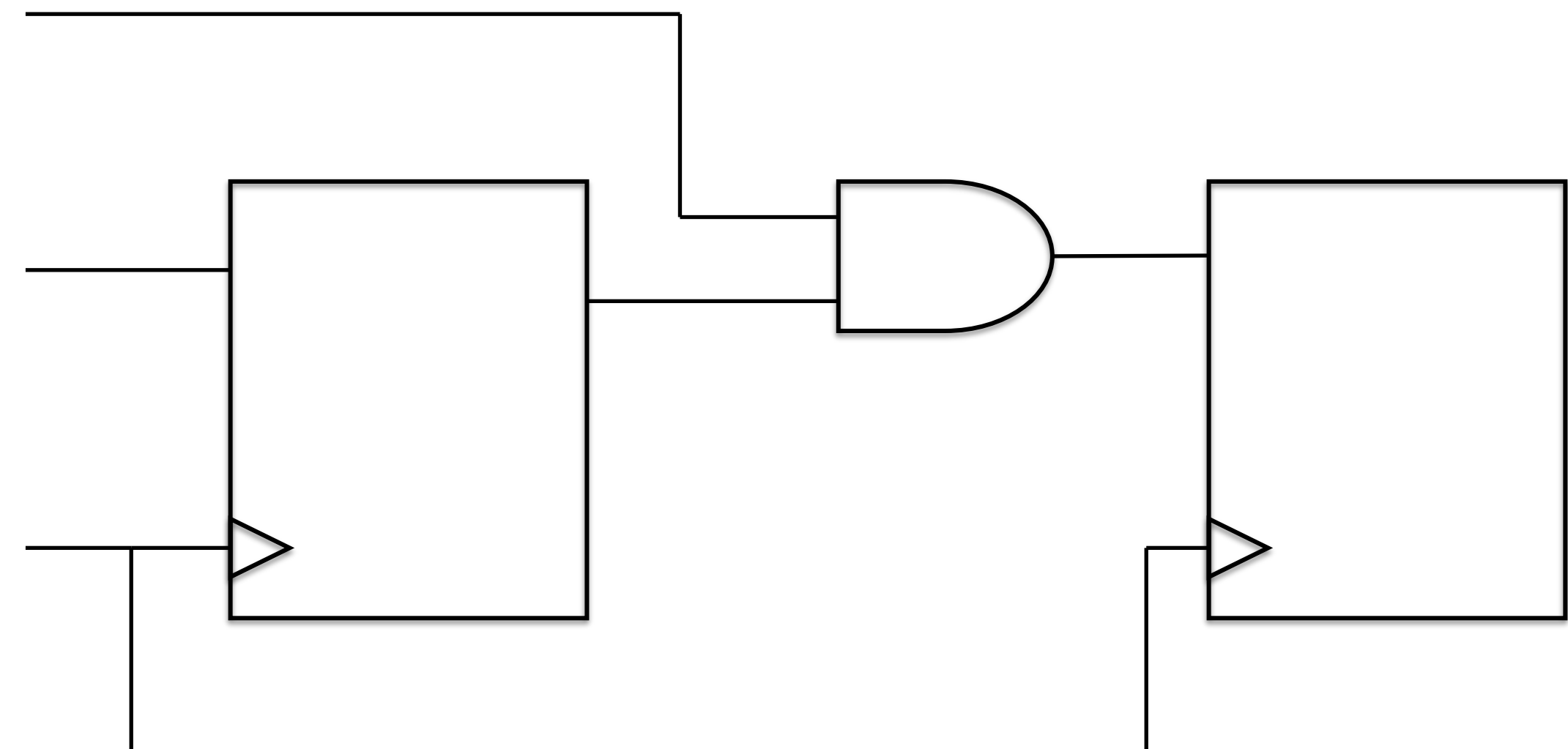
Register Transfer Level (RTL):  
Nets and registers



Behavioral:  
Algorithms



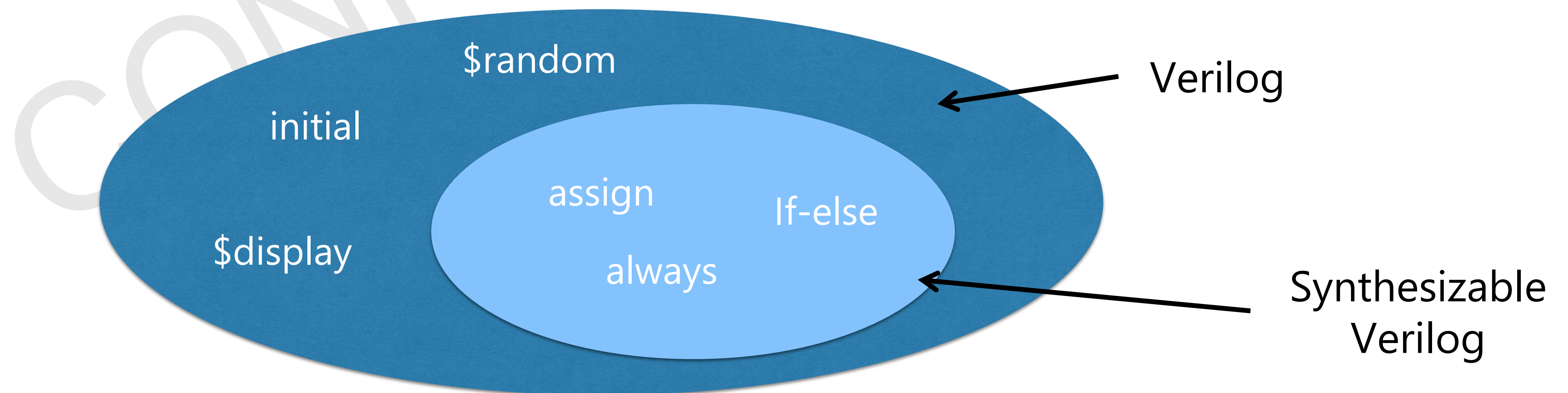
Gate Level:  
Primitives





# Synthesizable logic

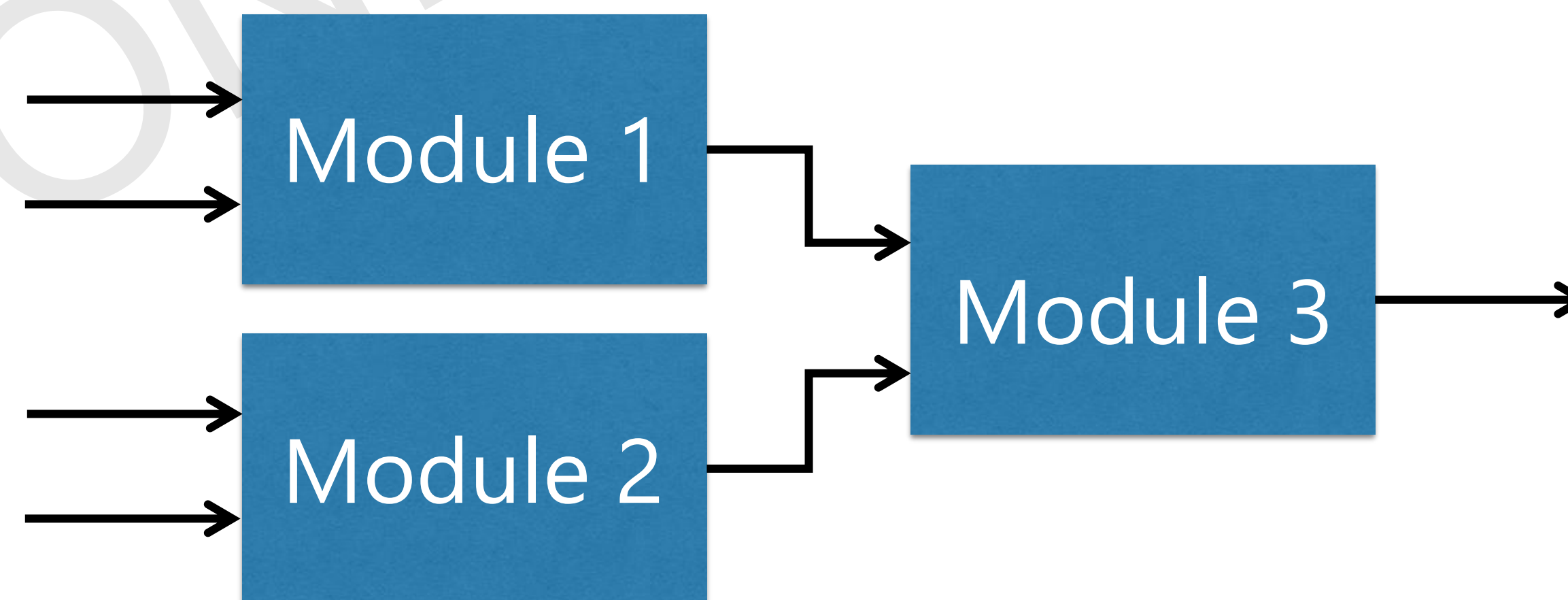
Synthesizable logic refers to the subset of language constructs and coding styles that **can be translated by synthesis tools into physical hardware implementations**, such as FPGA or ASIC circuits. This subset **excludes simulation-only features** and focuses on describing hardware structures and behavior that can be directly mapped to logic gates, flip-flops, and other digital components.



# Design Modules

The basic building block of designs is the module. A module can represent a complete system, a subblock or a basic cell.

- Modules “contain” logic internally and communicate with other modules through ports.
- A module may contain, in addition to logic, one or more instantiated modules within it.
- A module can be instantiated more than once.

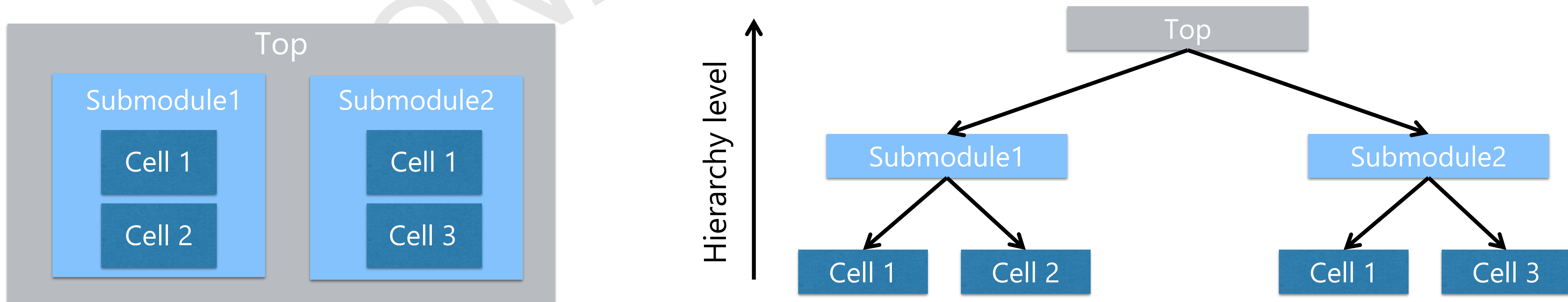




# Design Modules

The basic building block of designs is the module. A module can represent a complete system, a subblock or a basic cell.

- Modules “contain” logic internally and communicate with other modules through ports.
- A module may contain, in addition to logic, one or more instantiated modules within it.
- A module can be instantiated more than once.



# Creating a Module

Verilog 1995

```
module fulladder (a,b,carry_in,sum,carry_out);
input a,b,carry_in;
output sum, carry_out;
    assign sum = (a ^ b) ^ carry_in;
    assign carry_out = (a&&b) || (carry_in && (a||b));
endmodule
```

Verilog 2001

```
module fulladder (input a,b,carry_in, output sum,carry_out);
    assign sum = (a ^ b) ^ carry_in;
    assign carry_out = (a&&b) || (carry_in && (a||b));
endmodule
```

Ports

Statements  
(logic)





# Hierarchy instantiation

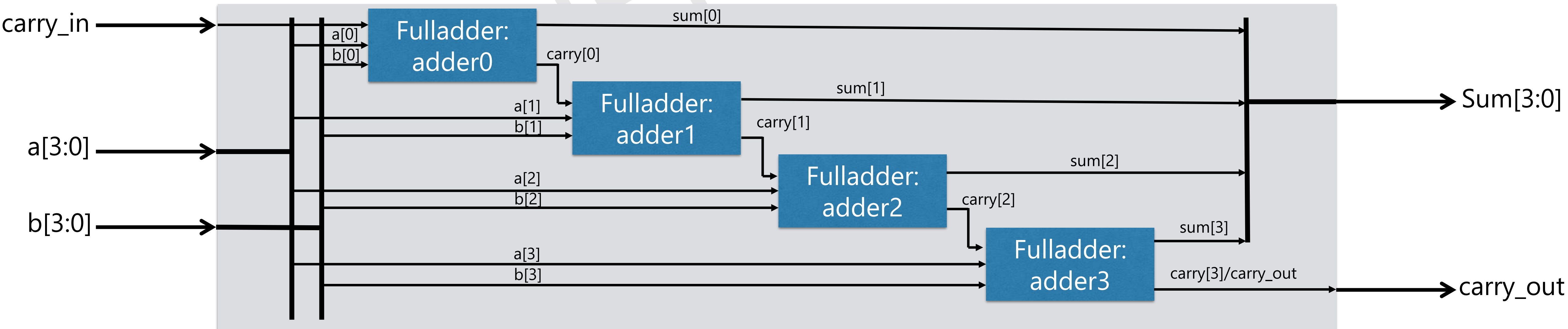
```

module adder_4bit (input [3:0]a, input [3:0]b, input carry_in, output [3:0]sum, output carry_out);
    wire [3:0]carry;
    fulladder adder0(.a(a[0]),.b(b[0]),.carry_in(carry_in),.sum(sum[0]),.carry_out(carry[0]));
    fulladder adder1(.a(a[1]),.b(b[1]),.carry_in(carry[0]),.sum(sum[1]),.carry_out(carry[1]));
    fulladder adder2(.a(a[2]),.b(b[2]),.carry_in(carry[1]),.sum(sum[2]),.carry_out(carry[2]));
    fulladder adder3(.a(a[3]),.b(b[3]),.carry_in(carry[2]),.sum(sum[3]),.carry_out(carry[3]));
    assign carry_out = carry[3];
endmodule

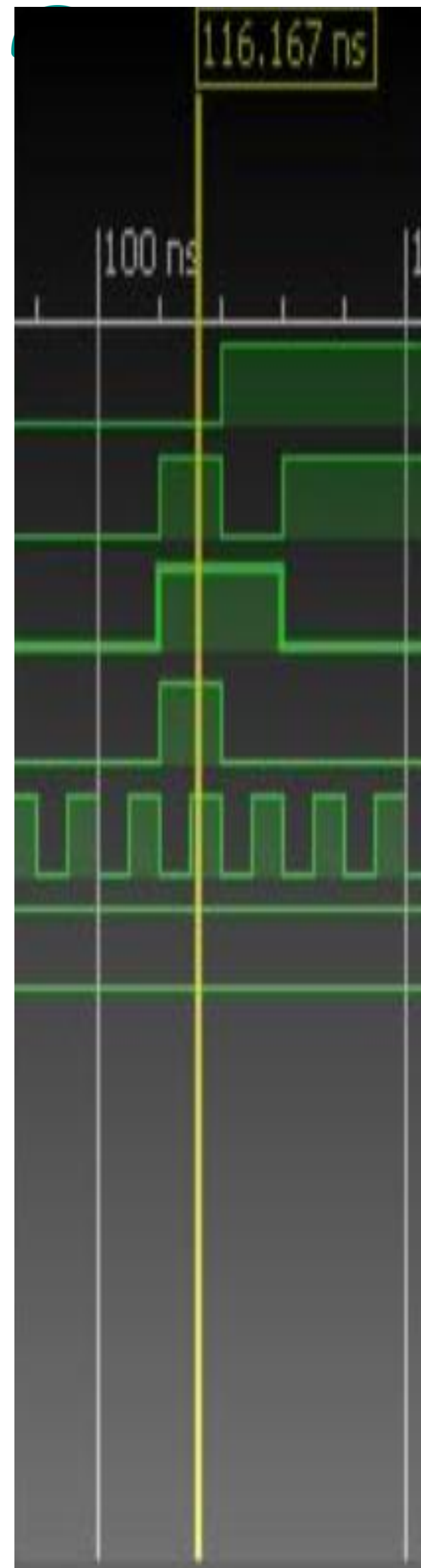
```

Module name

Instance name



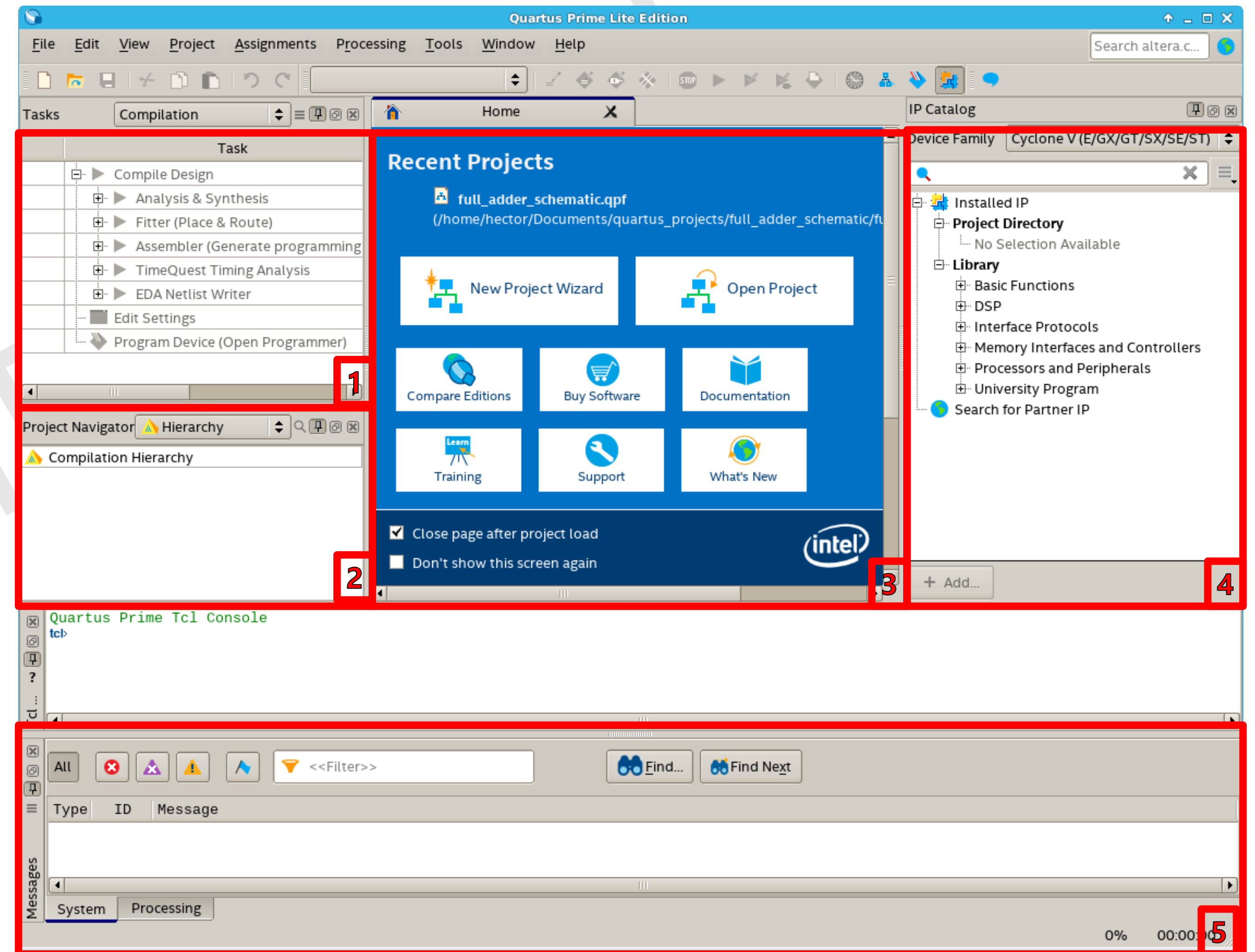
# An introduction to Quartus Prime Lite



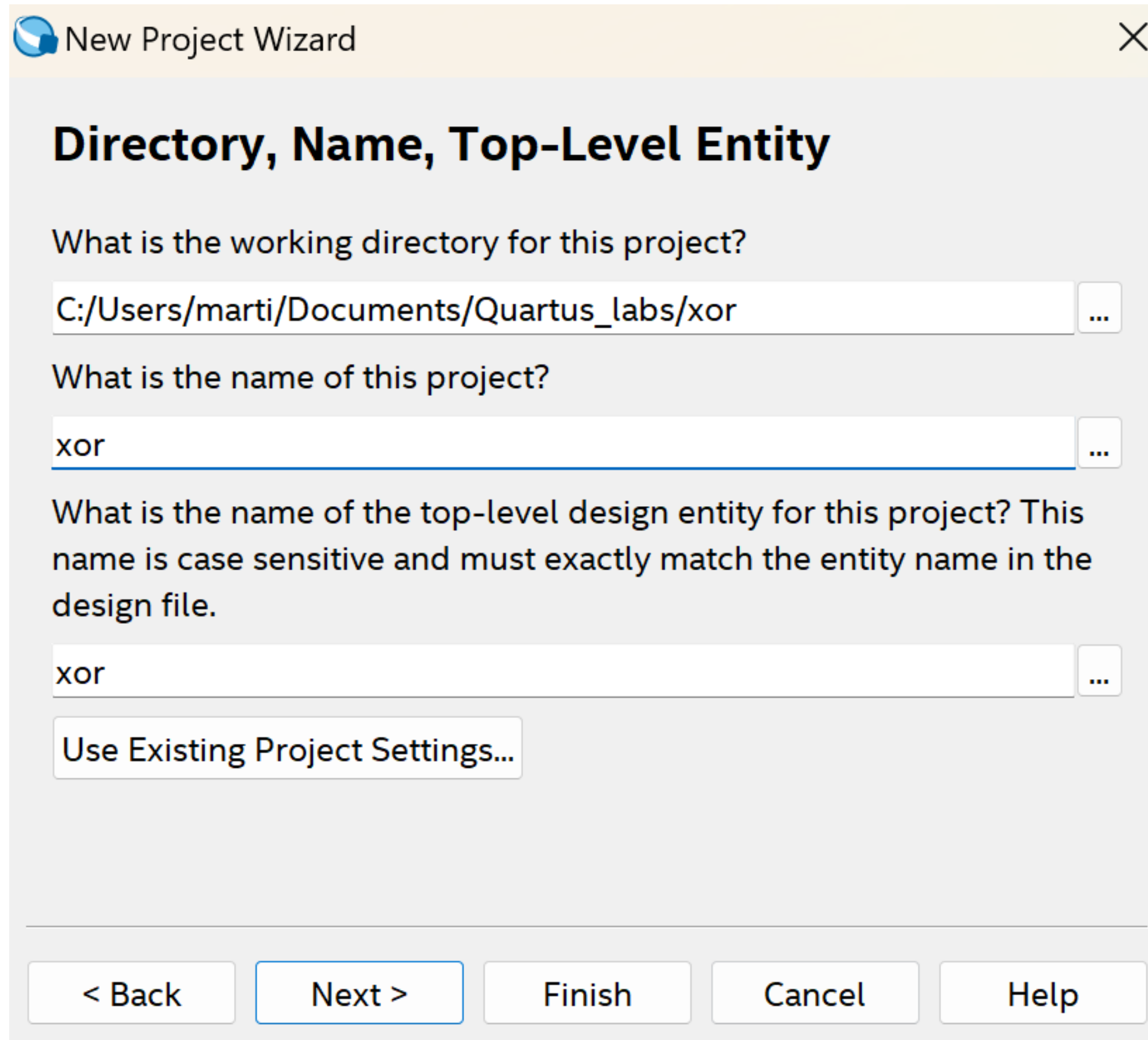


# Quartus interface

1. **Task:** design flow explorer, show the status of each process for the current design.
2. **Project navigator:** Visualize all files in the project.
3. **Work Area:** file editor windows are opened in this area.
4. **IP Catalog:** catalog of Ips and Altera macros.
5. **Message Window:** All messages produced through the design flow are displayed here.



# Getting started



**New Project Wizard**

**Directory, Name, Top-Level Entity**

What is the working directory for this project?

C:/Users/marti/Documents/Quartus\_labs/xor ...

What is the name of this project?

xor ...

What is the name of the top-level design entity for this project? This name is case sensitive and must exactly match the entity name in the design file.

xor ...

Use Existing Project Settings...

< Back   Next >   Finish   Cancel   Help

1. Go to file menu and select New Project Wizard ...
2. Select as working directory 'xor'.
3. Name your project 'xor'.
4. Set as top entity 'xor'. This box is case sensitive, and the entered name must exactly match the name of the top-level entity on the design.
5. On Project Type select 'Empty Project'.
6. On Add files press 'next'.

# Getting started

New Project Wizard

## Family, Device & Board Settings

Device | Board

Select the family and device you want to target for compilation.  
 You can install additional device support with the Install Devices command on the Tools menu.  
 To determine the version of the Quartus Prime software in which your target device is supported, refer to the [Device Support List](#) webpage.

Device family

Family: Cyclone IV E

Device: All

Target device

☐ Auto device selected by the Fitter

☒ Specific device selected in 'Available devices' list

☐ Other: n/a

Show in 'Available devices' list

Package: Any

Pin count: Any

Core speed grade: Any

Name filter:

☒ Show advanced devices

Available devices:

Name	Core Voltage	LEs	Total I/Os	GPIOs	Memory Bits	Embedded multiplier 9-bit ele
EP4CE115F29C7	1.2V	114480	529	529	3981312	532

< Back | Next > | Finish | Cancel | Help

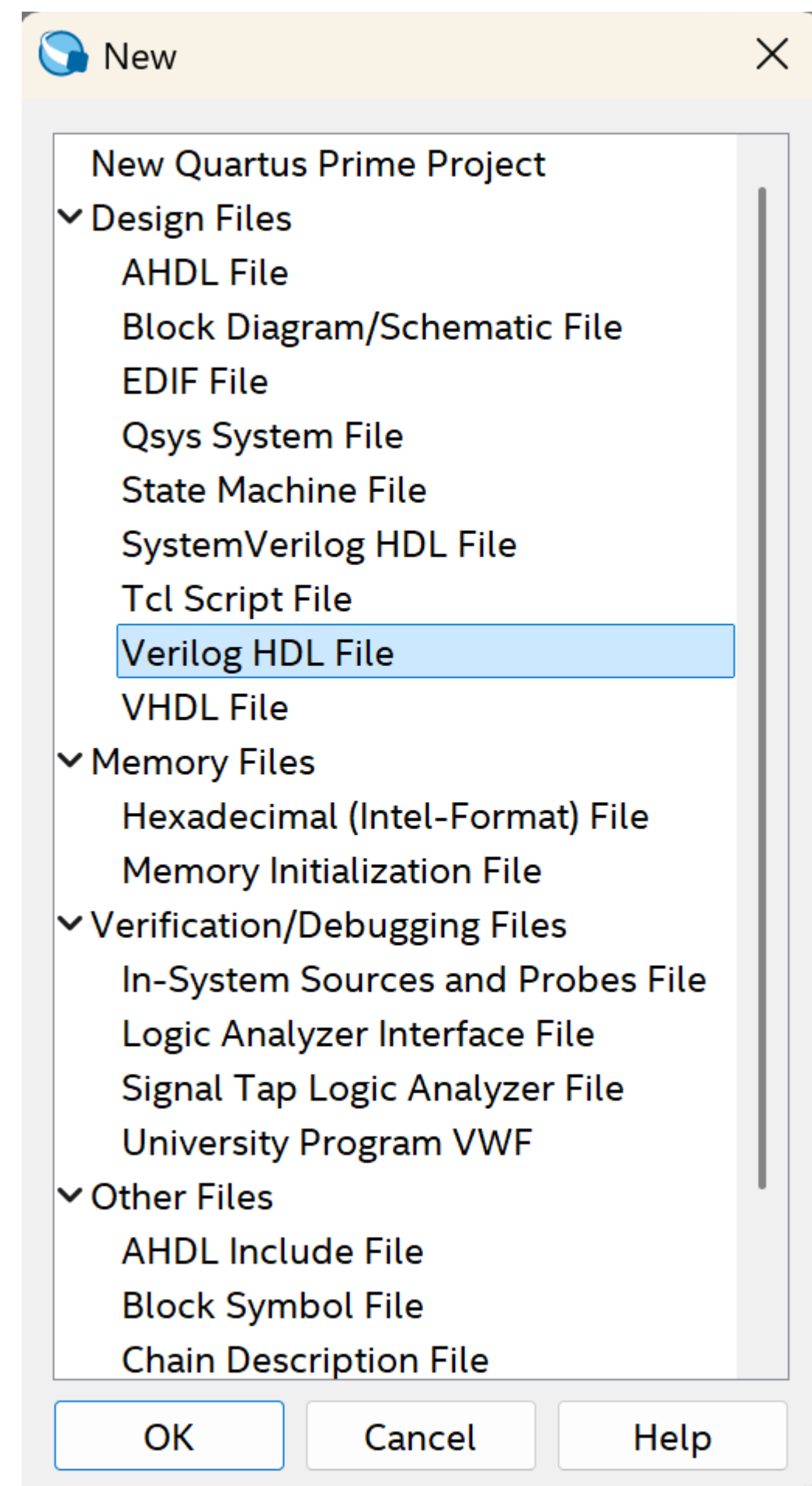
7. Select the device on your development board. If you're not sure consult the board manual.

8. On EDA Tool settings press 'Next'.

9. On summary check that all your selections are correct and press 'Finish'.



# Getting started



10. Go to file '*menu*' and select '*new*'.

11. On new window select '*Verilog HDL File*'.

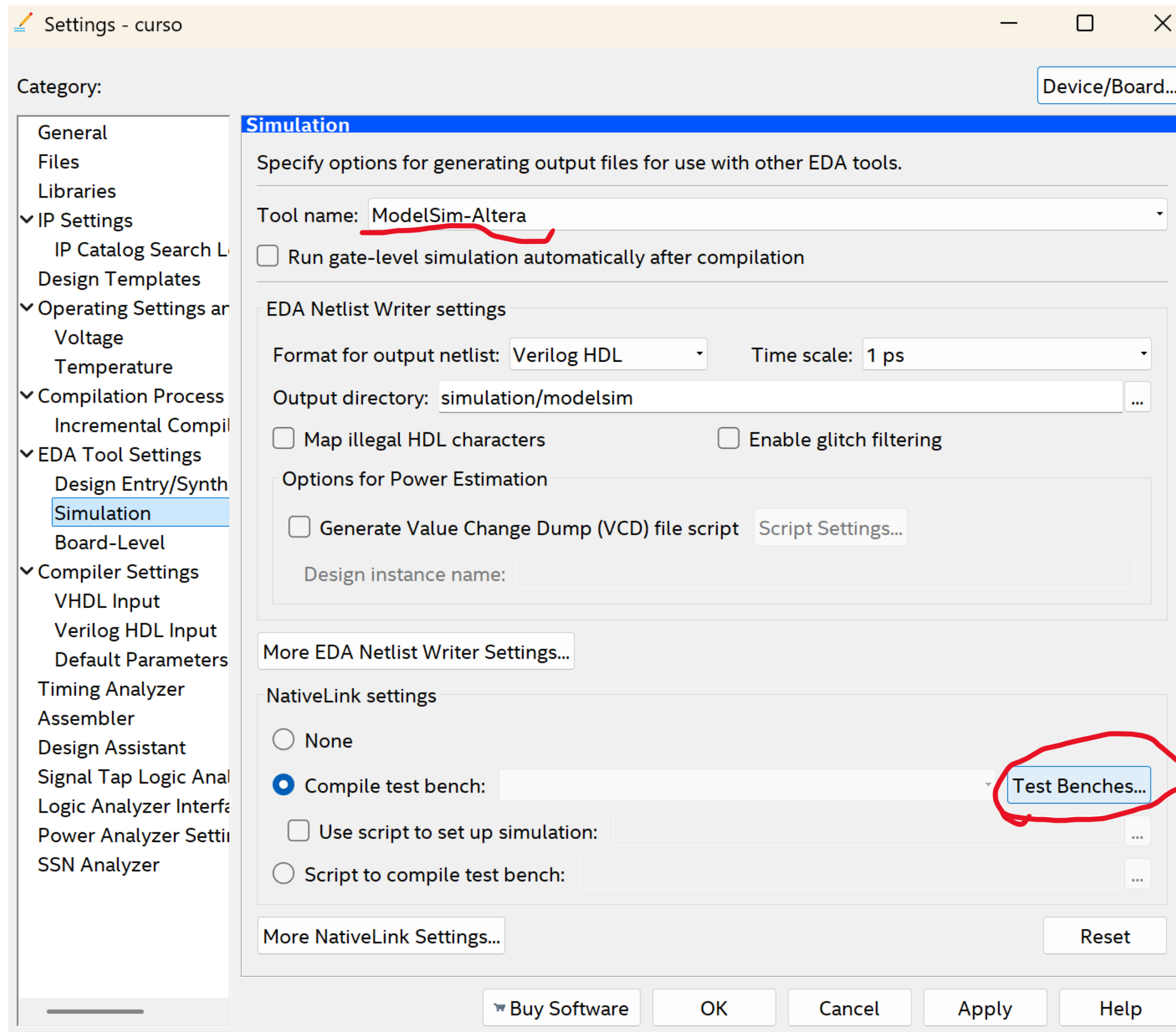
12. Write the Verilog code

# Simulation





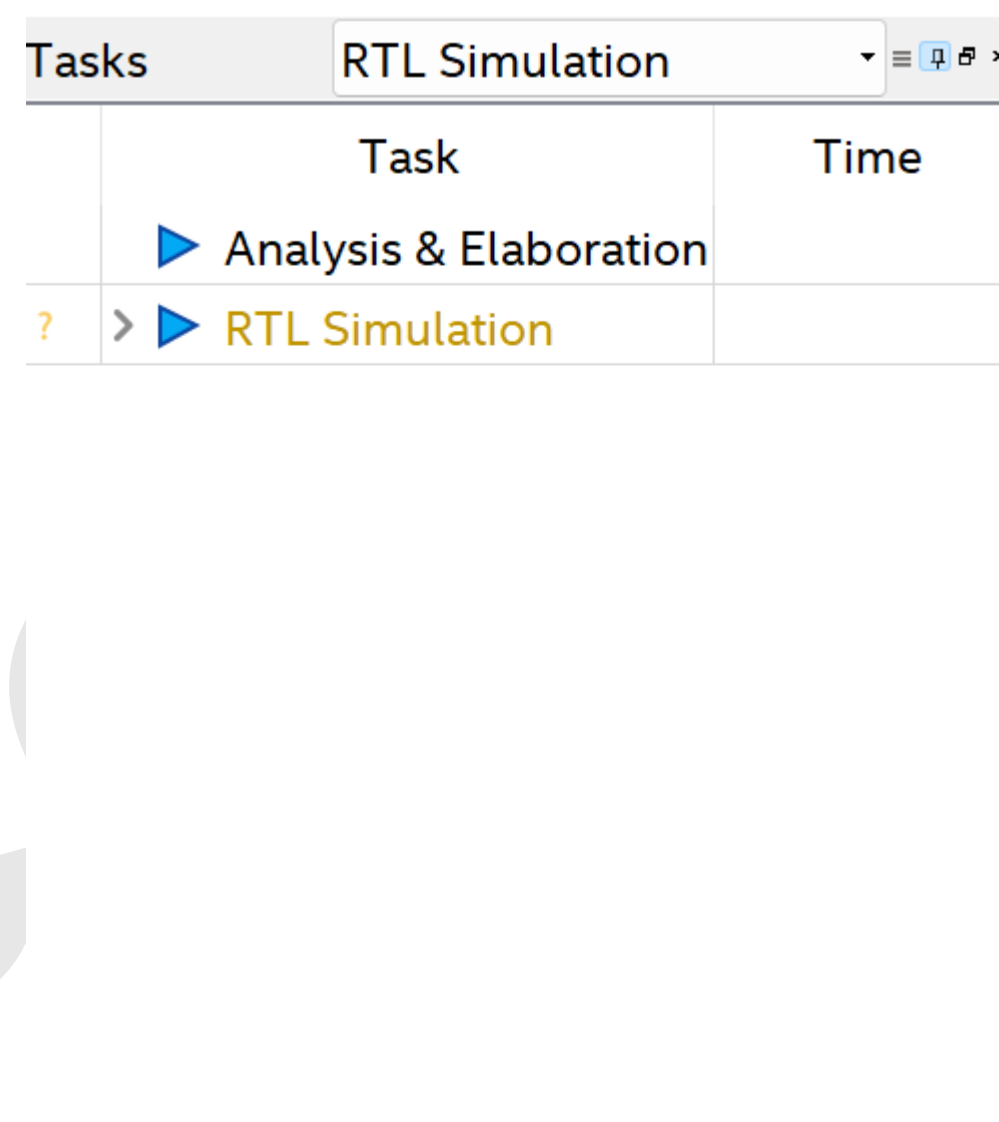
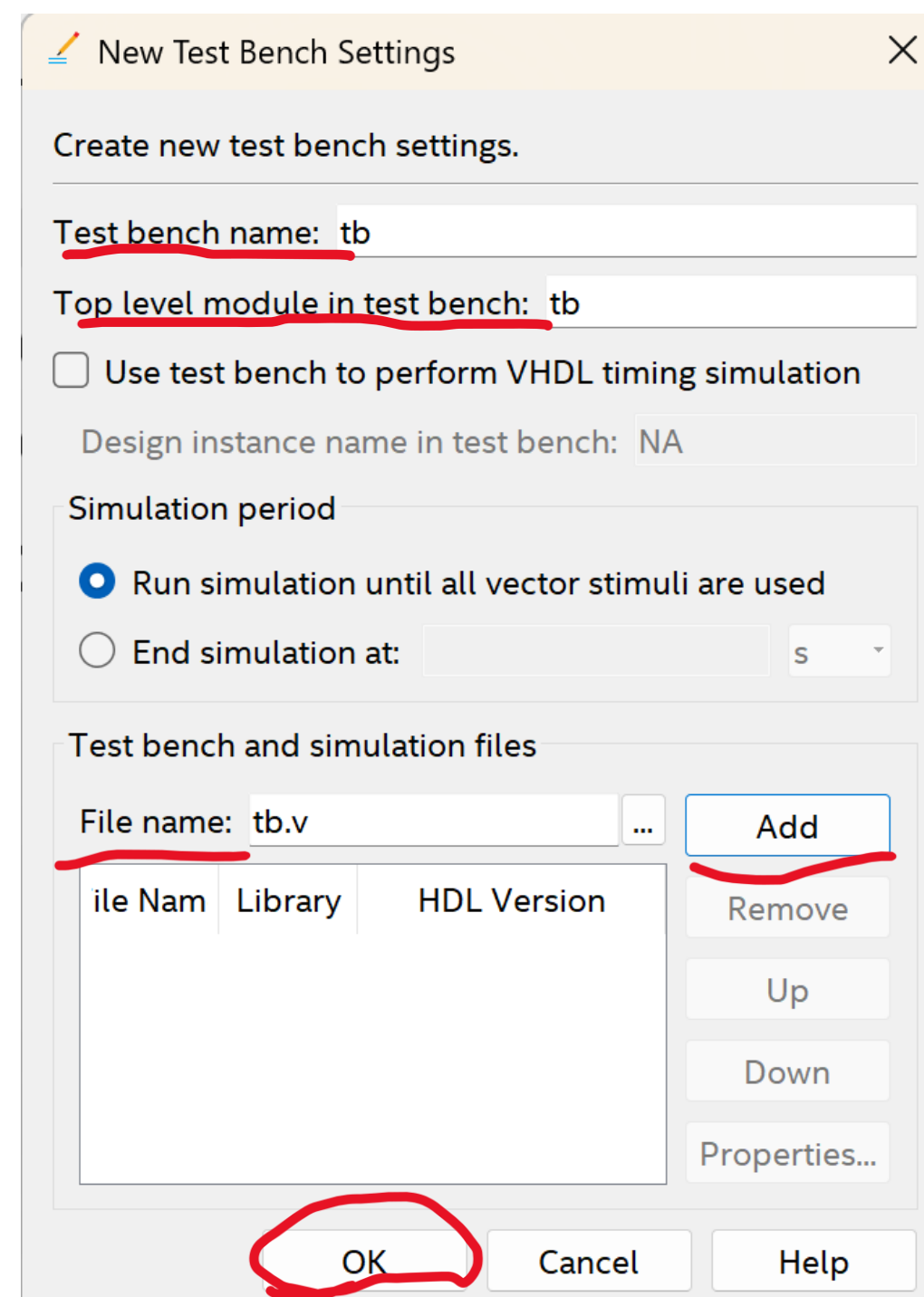
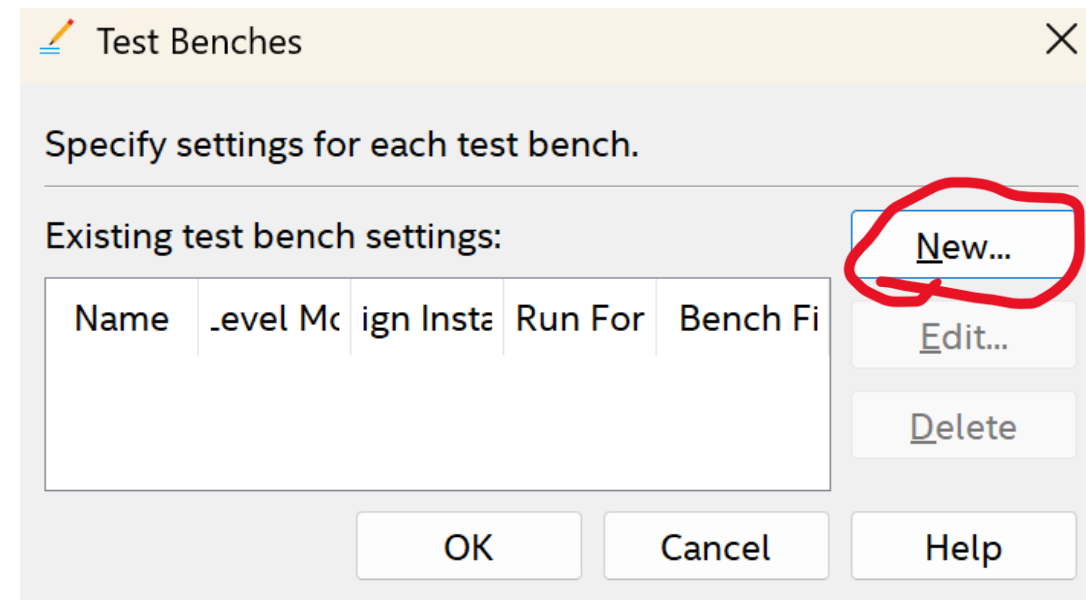
# Quartus testbench simulation



1. Click on 'file' menu and select 'New'.
2. On the 'New' window select 'Verilog HDL File'.
3. Write testbench code.
4. Click "assignment" menu and select "settings".
5. Go to "EDA Tool Settings > Simulation".
6. Select the simulation tool "ModelSim-Altera" and Click "Test Bench".

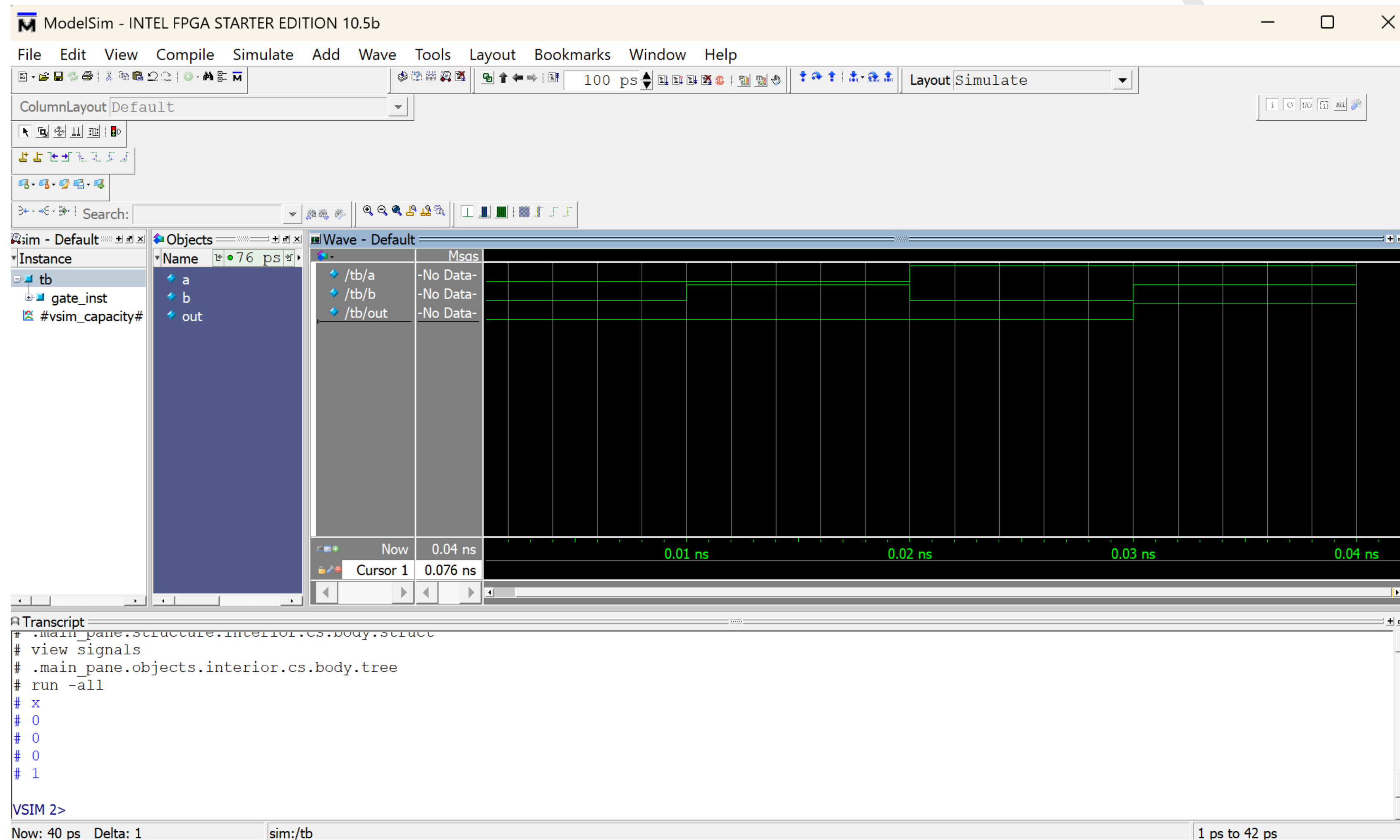


# Quartus testbench simulation

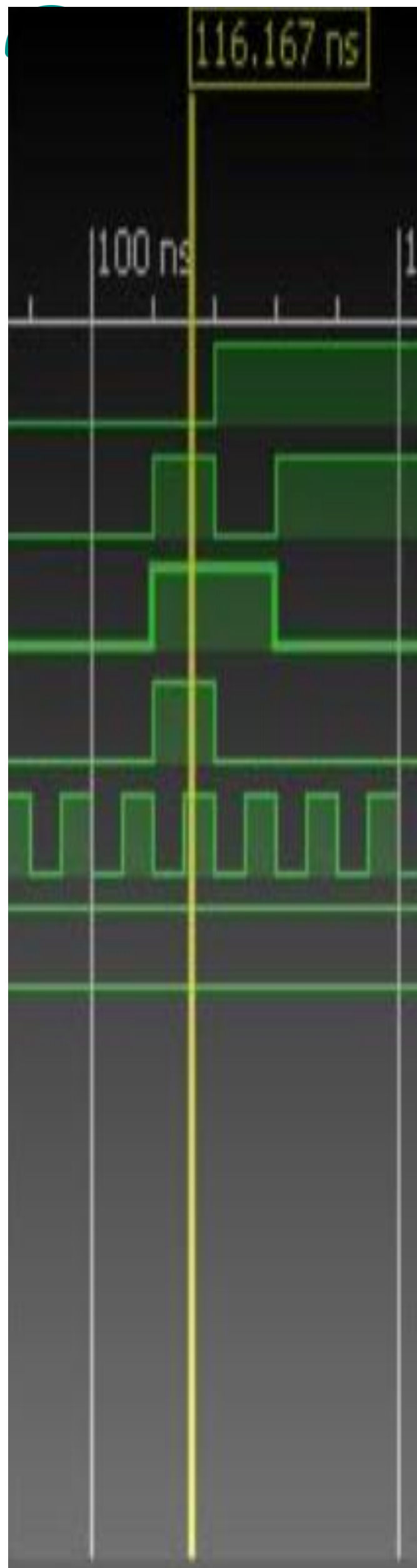


6. On the Test Bench window click "New"
7. On New Test Bench Settings window write the "Test bench name" and "Top level module in test bench".
8. Write or search the "Test bench File name", then click "Add".
9. Click "ok" on all windows.
10. On task window select RTL simulation.
11. Click twice RTL Simulation, a ModelSim window will prompt.

# Quartus testbench simulation

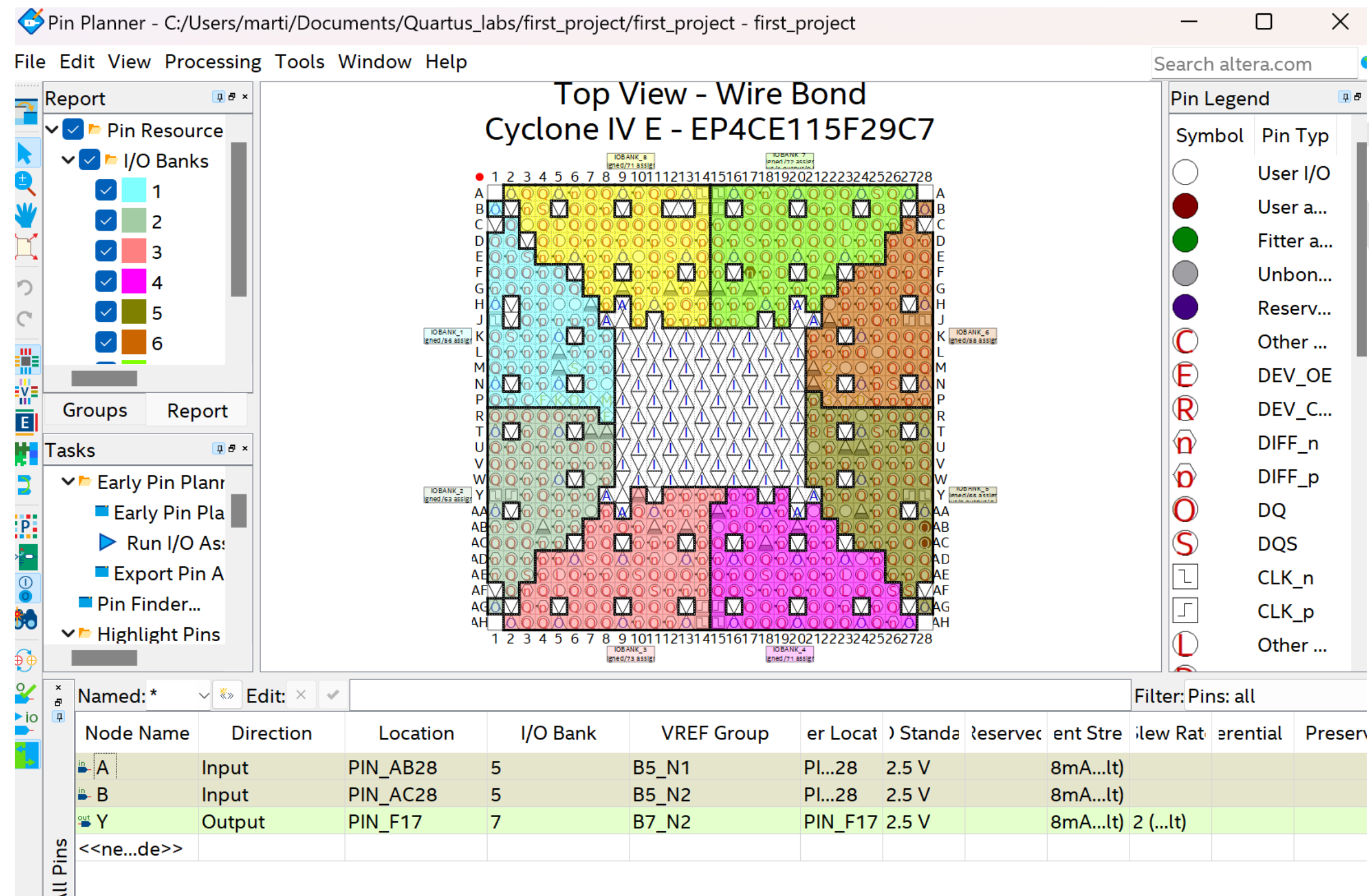


# Programming FPGA





# Pin Planner



**Top View - Wire Bond**  
Cyclone IV E - EP4CE115F29C7

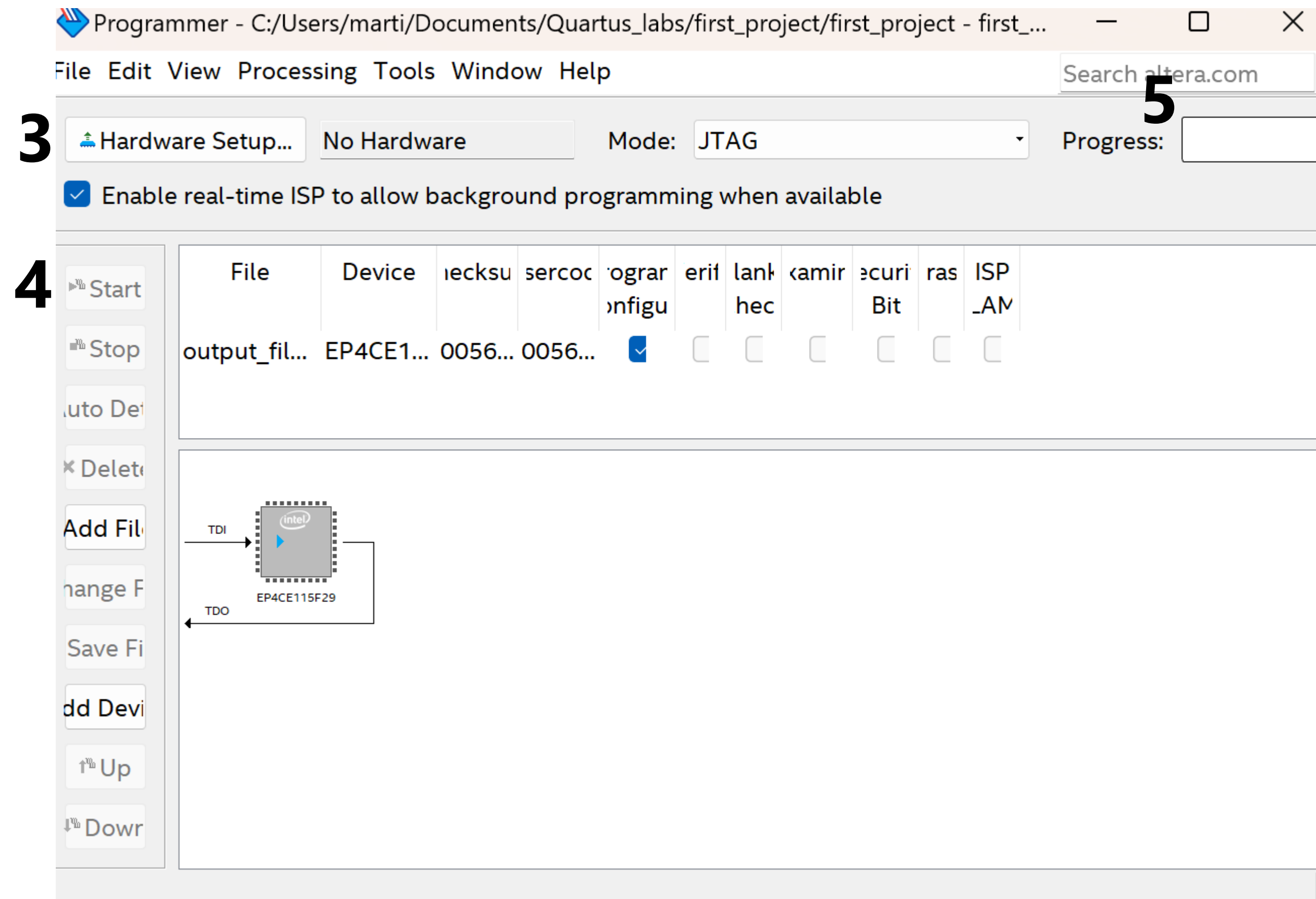
**Pin Legend**

Symbol	Pin Type
○	User I/O
●	User a...
●	Fitter a...
●	Unbon...
●	Reserv...
○	Other ...
○	DEV_OE
○	DEV_C...
○	DIFF_n
○	DIFF_p
○	DQ
○	DQS
○	CLK_n
○	CLK_p
○	Other ...

Node Name	Direction	Location	I/O Bank	VREF Group	er Locat	Stand	reserve	ent Stre	lew Rat	erential	Preserv
A	Input	PIN_AB28	5	B5_N1	PI...28	2.5 V		8mA...lt)			
B	Input	PIN_AC28	5	B5_N2	PI...28	2.5 V		8mA...lt)			
Y	Output	PIN_F17	7	B7_N2	PIN_F17	2.5 V		8mA...lt) 2 (...lt)			
<<ne...de>>											

1. Go to *'assignments'* menu and select *'Pin Planner'*.
2. In the table at the bottom of the window, enter the pin to be used in the *'Location'* column.

# Programming FPGA



1. Connect the FPGA to the Computer.
2. Go to '*tools*' menu and select '*Programmer*'.
3. *Click Hardware setup and select USB-Blaster (or the appropriate driver).*
4. Click start.
5. If everything es correct progress bar should be green.