Basic project with Libero

Created by: David Rubio G.

Blog post: https://soceame.wordpress.com/2025/03/09/basic-project-with-libero/

Blog: https://soceame.wordpress.com/

GitHub: https://github.com/DRubioG

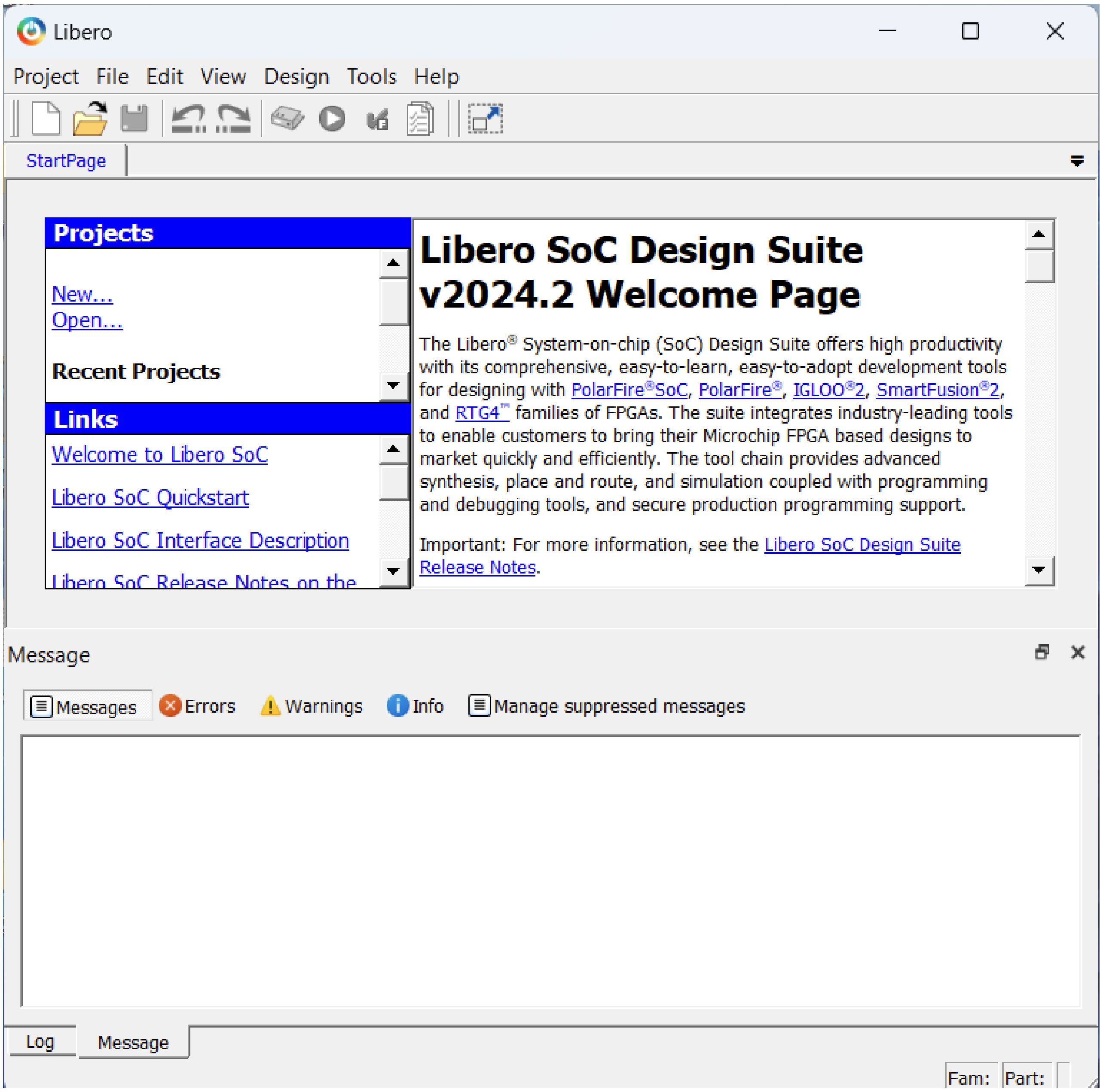
Last modification date: 09/03/25

https://soceame.wordpress.com/2025/03/09/basic-project-with-libero/

In this post I will discuss how to create a basic project with Libero.

Creating a project

The first step to working with Libero is to open the tool.

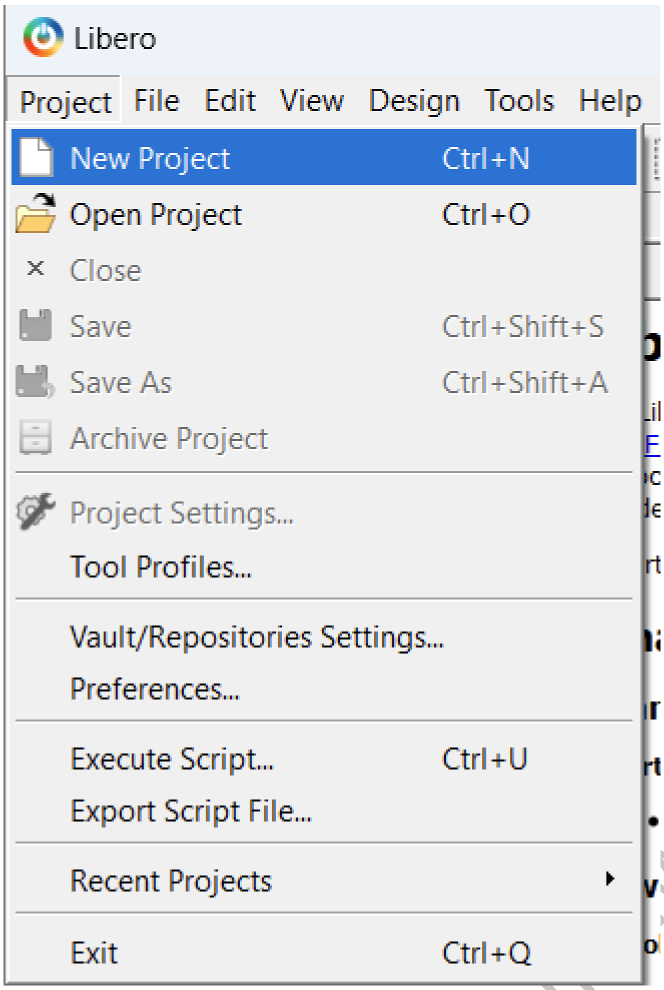

Libero
Project File Edit View Design Tools Help
StartPage
Projects
New...
Open...
Recent Projects
Links
Welcome to Libero SoC
Libero SoC Quickstart
Libero SoC Interface Description
Libero SoC Release Notes on the
Message
Messages Errors Warnings Info Manage suppressed messages
Log Message
Fam: Part:

Libero SoC Design Suite v2024.2 Welcome Page

The Libero® System-on-chip (SoC) Design Suite offers high productivity with its comprehensive, easy-to-learn, easy-to-adopt development tools for designing with PolarFire®SoC, PolarFire®, IGLOO®2, SmartFusion®2, and RTG4™ families of FPGAs. The suite integrates industry-leading tools to enable customers to bring their Microchip FPGA based designs to market quickly and efficiently. The tool chain provides advanced synthesis, place and route, and simulation coupled with programming and debugging tools, and secure production programming support.

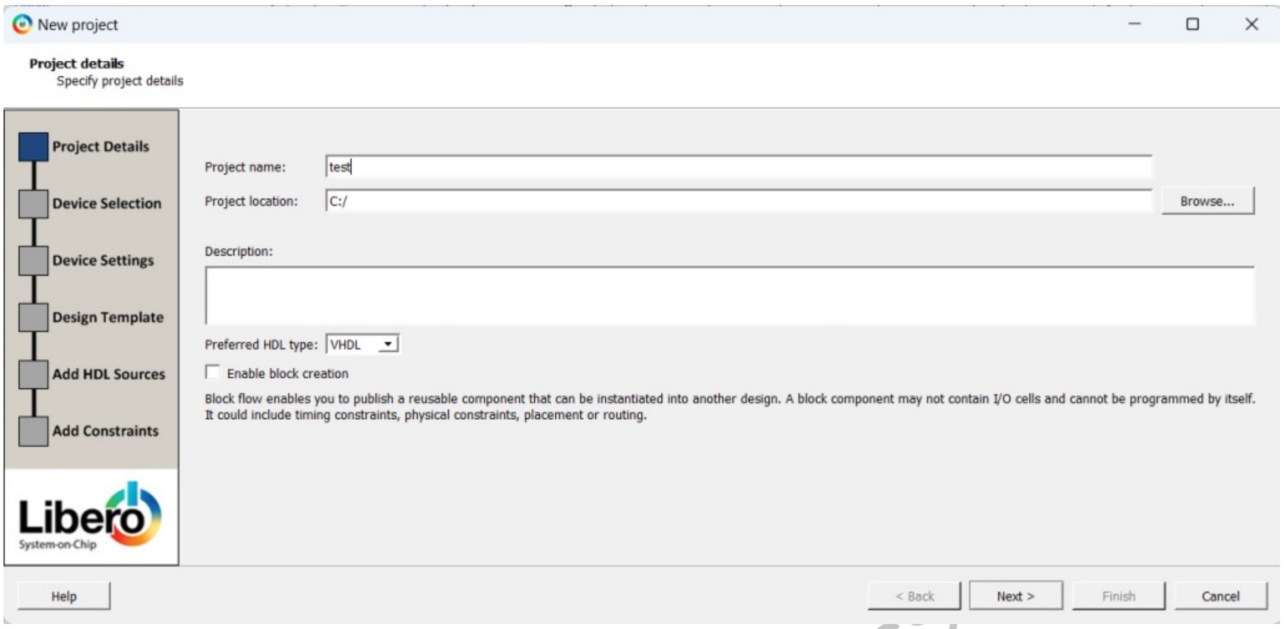
Important: For more information, see the Libero SoC Design Suite Release Notes.

Once opened, we tell it that we want to create a new project.



Then a tab for building the project opens. In the first part, it asks us what we want to call the project, where we are going to create it and the language we are going to use.

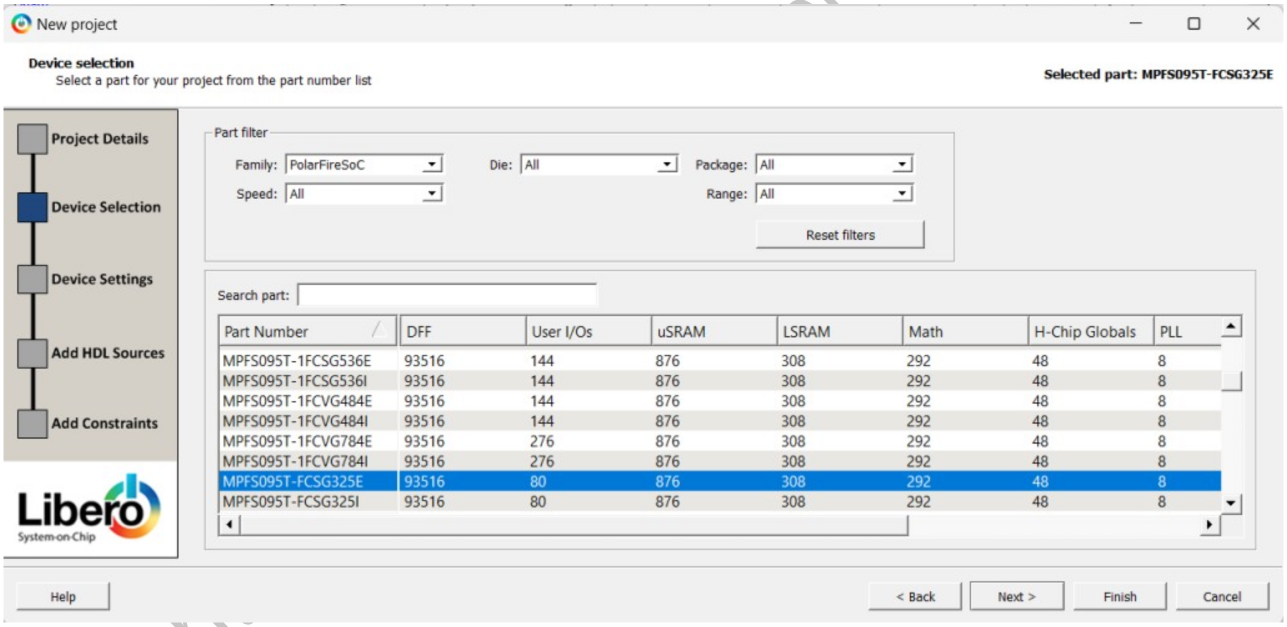
NOTE: it is worth remembering that the free Libero license does not allow the use of two languages in the same project.



Project details
Specify project details

Project name: test
Project location: C:/
Description:
Preferred HDL type: VHDL
☐ Enable block creation
Block flow enables you to publish a reusable component that can be instantiated into another design. A block component may not contain I/O cells and cannot be programmed by itself. It could include timing constraints, physical constraints, placement or routing.

The next tab asks us for the device we are going to program.



Device selection
Select a part for your project from the part number list

Selected part: MPFS095T-FCSG325E

Part filter
Family: PolarFireSoC Die: All Package: All
Speed: All Range: All

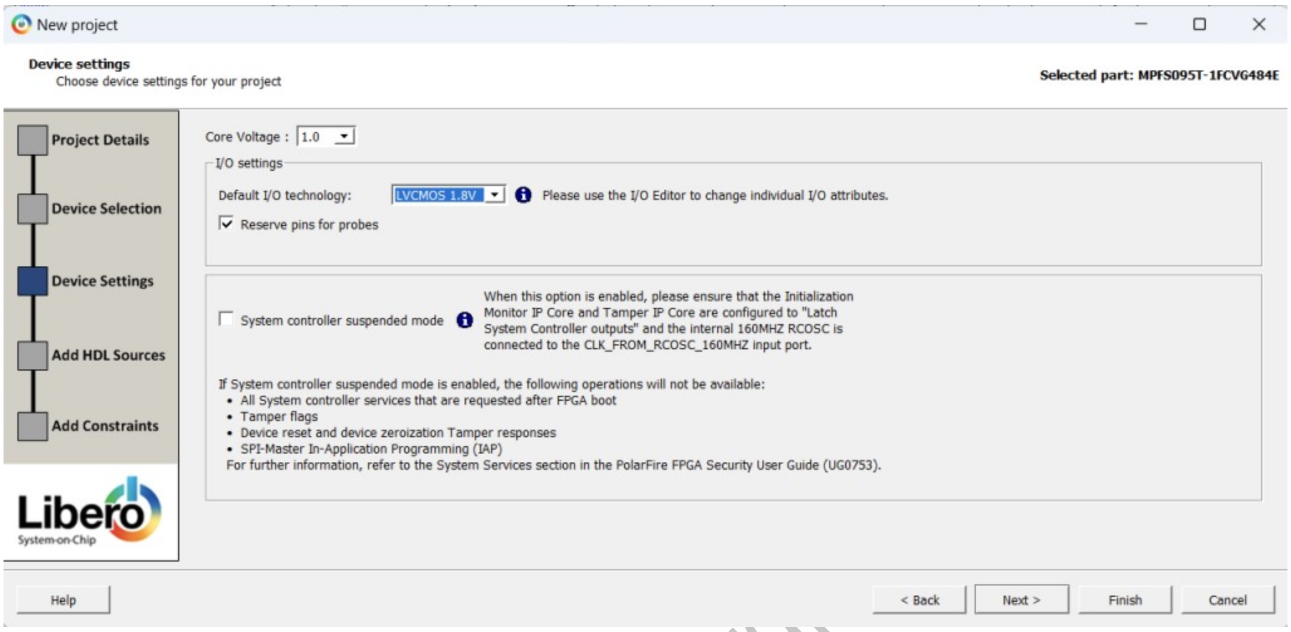
Search part:

Part Number	DFF	User I/Os	uSRAM	LSRAM	Math	H-Chip Globals	PLL
MPFS095T-1FCSG536E	93516	144	876	308	292	48	8
MPFS095T-1FCSG536I	93516	144	876	308	292	48	8
MPFS095T-1FCVG484E	93516	144	876	308	292	48	8
MPFS095T-1FCVG484I	93516	144	876	308	292	48	8
MPFS095T-1FCVG784E	93516	276	876	308	292	48	8
MPFS095T-1FCVG784I	93516	276	876	308	292	48	8
MPFS095T-FCSG325E	93516	80	876	308	292	48	8
MPFS095T-FCSG325I	93516	80	876	308	292	48	8

NOTE: if you haven't noticed, the Design Templates option has disappeared between the last two images. This is because you have selected to program a SoC that does not have a Cortex-M3 (the family that has the Cortex-M3 is the SmartFusion2). If you had chosen a SmartFusion2, this tab is the one that allows you to program the SoC (for the PolarFire SoC, it is done differently).

The next tab asks us for the core voltage and the default voltage of the pins. This influences when declaring the pins. It also asks us if we want to reserve pins for debugging. *I will explain the ways to debug with Libero in another post.*

NOTE: if you cannot change the voltage of the pins when declaring them, the problem is in the voltage defined by default. To access this option to change it once the project has been created, you have to go to the *Project* option, and inside look for *Project Settings*, and then *Device settings*.


New project

Device settings
Choose device settings for your project

Selected part: MPFS095T-1FCVG484E

Core Voltage : 1.0

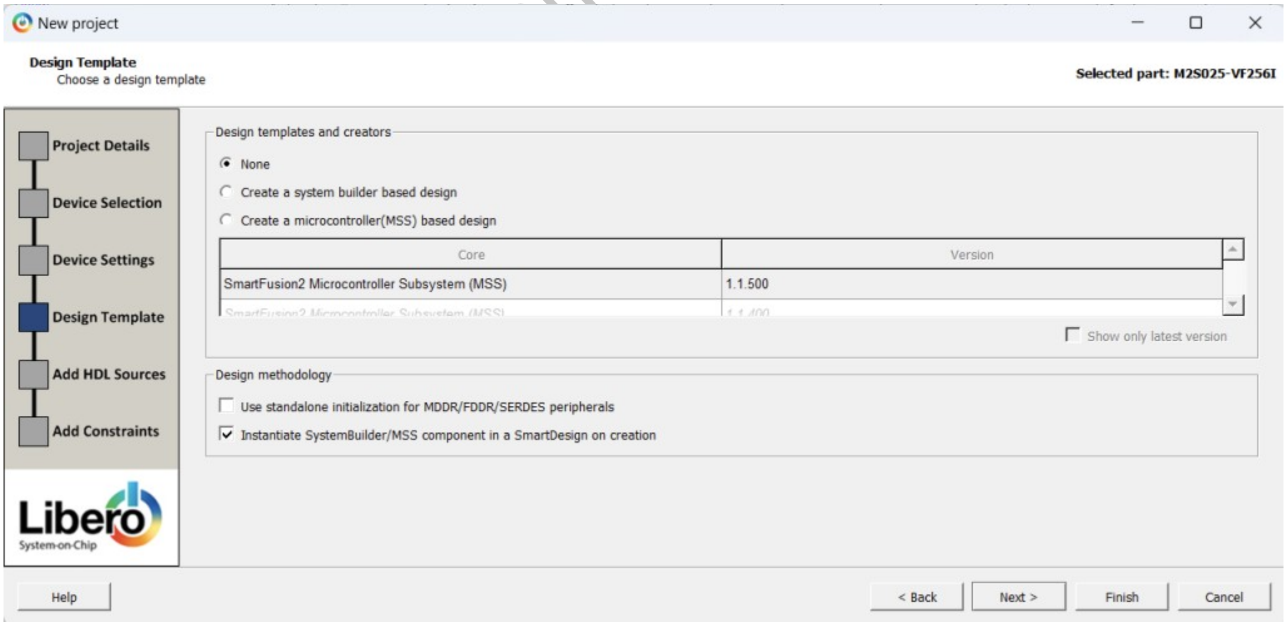
I/O settings
Default I/O technology: LVCMOS 1.8V — Please use the I/O Editor to change individual I/O attributes.
[✓] Reserve pins for probes

[ ] System controller suspended mode — When this option is enabled, please ensure that the Initialization Monitor IP Core and Tamper IP Core are configured to "Latch System Controller outputs" and the internal 160MHZ RCOSC is connected to the CLK_FROM_RCOSC_160MHZ input port.

If System controller suspended mode is enabled, the following operations will not be available:
- All System controller services that are requested after FPGA boot
- Tamper flags
- Device reset and device zeroization Tamper responses
- SPI-Master In-Application Programming (IAP)

For further information, refer to the System Services section in the PolarFire FPGA Security User Guide (UG0753).

If we had chosen to program a SoC from the PolarFire family, we have to go to the Project option, and inside look for Project Settings, and then Device settings. SmartFusion2, this tab would appear, where it allows us to choose to program the SmartFusion2 as FPGA (*None*) or as SoC, giving us two options, allowing us to preconfigure it or configure it later.


New project

Design Template
Choose a design template

Selected part: M2S025-VF256I

Design templates and creators
(●) None
( ) Create a system builder based design
( ) Create a microcontroller(MSS) based design

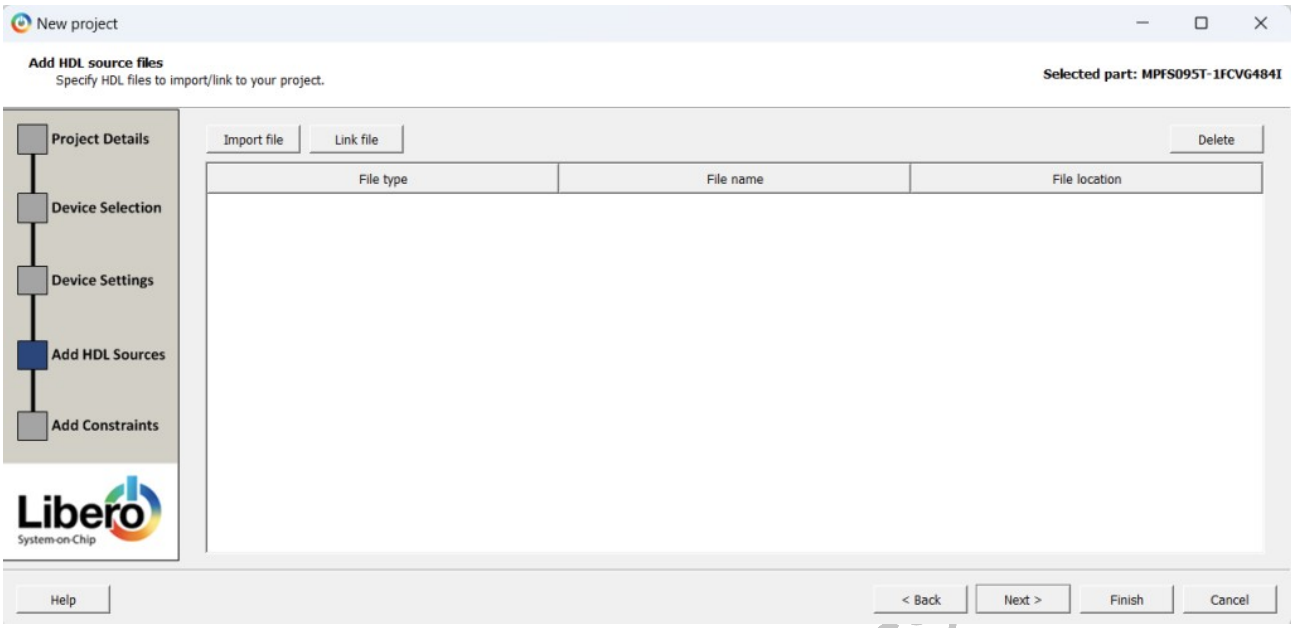
Core	Version
SmartFusion2 Microcontroller Subsystem (MSS)	1.1.500
SmartFusion2 Microcontroller Subsystem (MSS)	1.1.400

[ ] Show only latest version

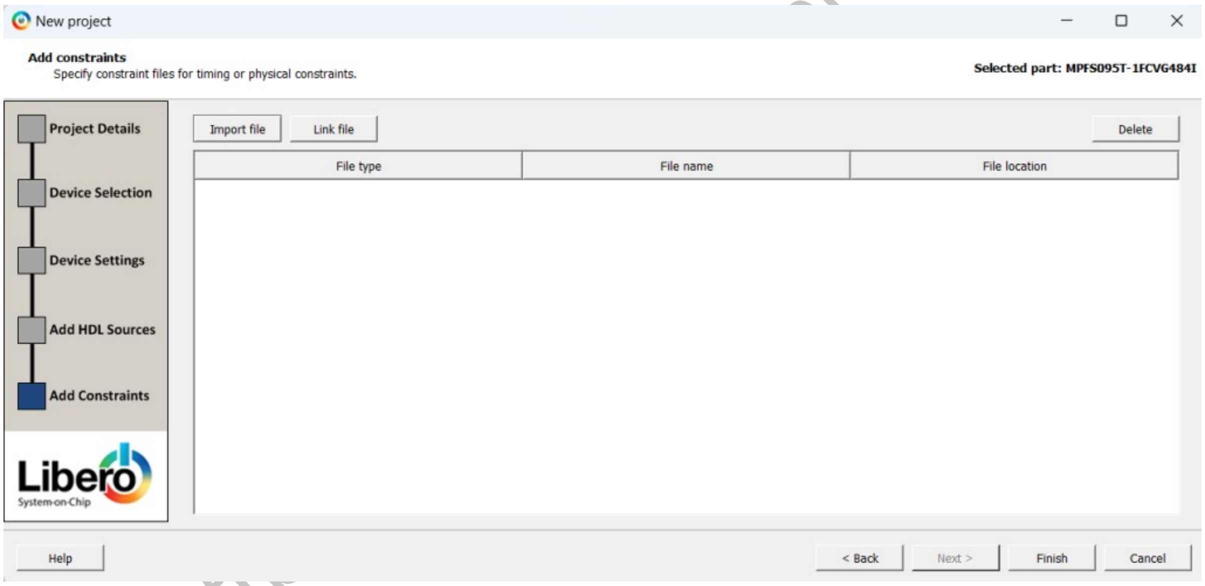
Design methodology
[ ] Use standalone initialization for MDDR/FDDR/SERDES peripherals
[✓] Instantiate SystemBuilder/MSS component in a SmartDesign on creation

The penultimate tab asks us if we want to add or link a folder with source code.

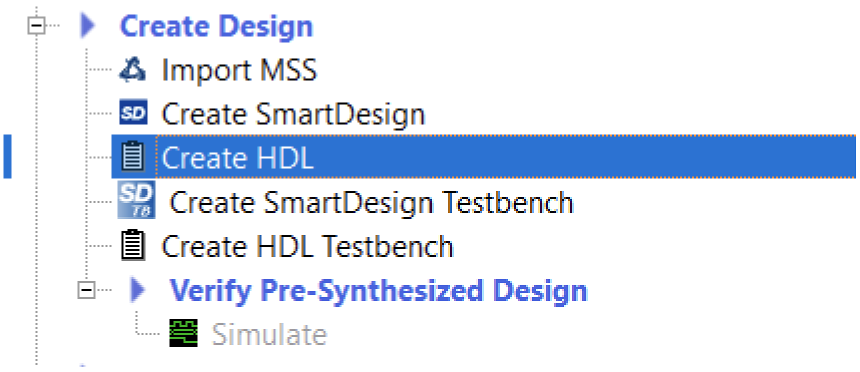
https://soceame.wordpress.com/2025/03/09/basic-project-with-libero/



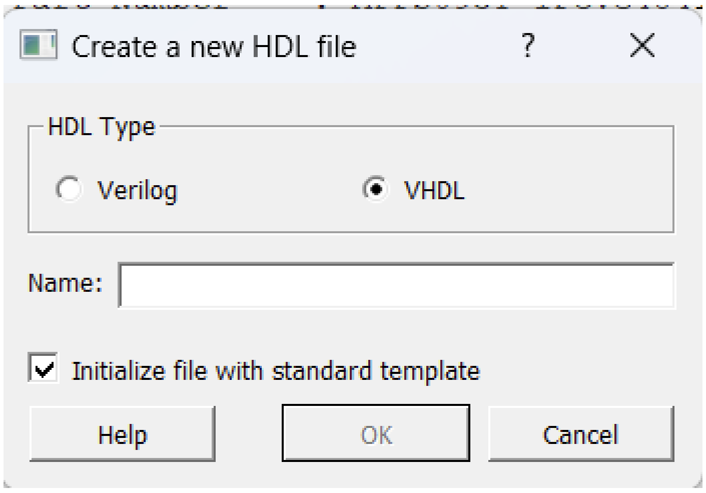
The last tab asks us if we want to add or link a file with restrictions.



Once the project is created, the first step is to create an HDL file, to do this we click on the Create HDL option.



The next thing it asks us is the name of the file to be created and the language used.



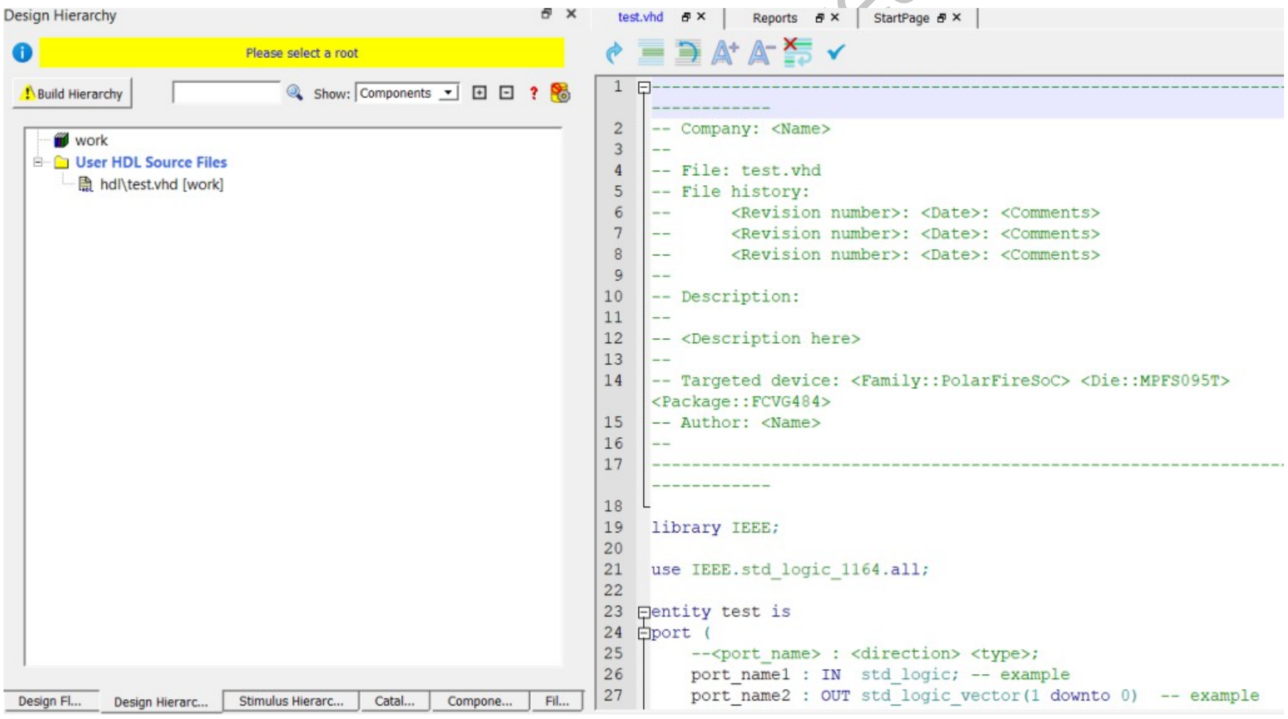
Create a new HDL file

HDL Type: Verilog [✓ VHDL]

Name:

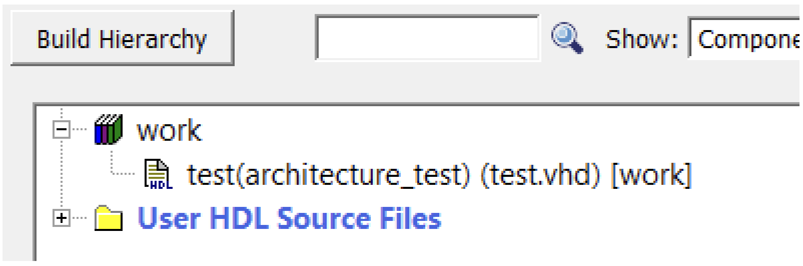
[✓] Initialize file with standard template

Once created, in the *Design Hierarchy* tab, the created file appears, but a message appears that says *Please select a root*, this is because we have not selected any file as the main one.

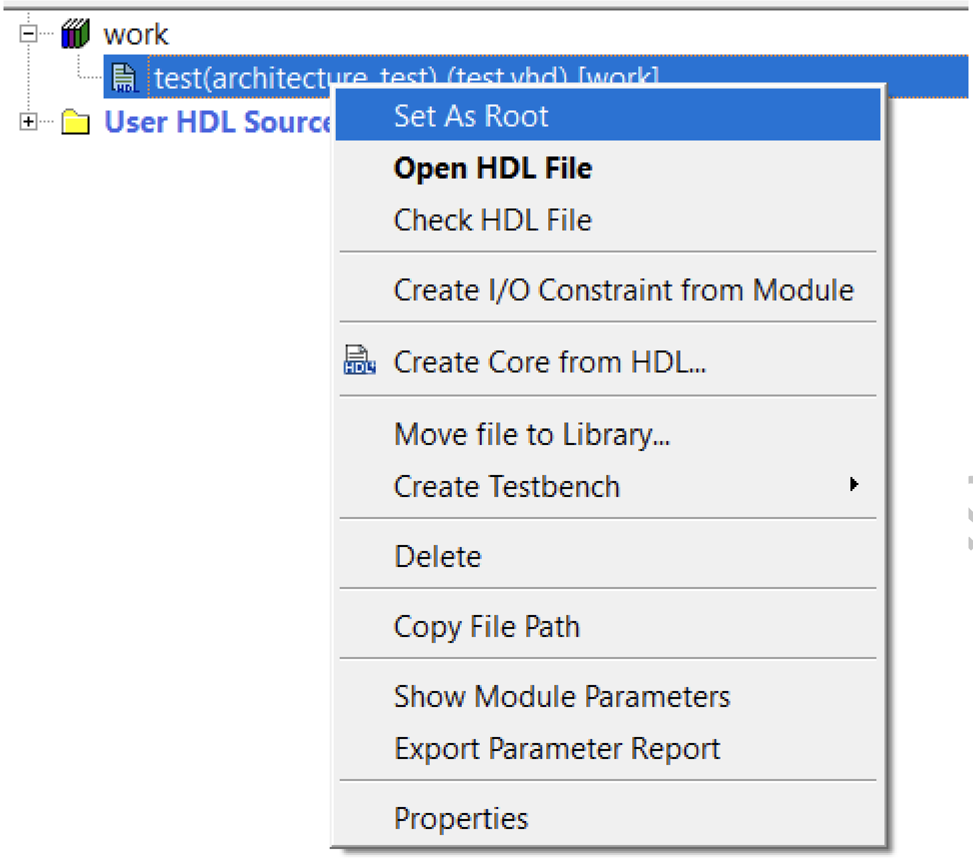


```
---------------------------------------------------------------------------------
-- Company: <Name>
--
-- File: test.vhd
-- File history:
--      <Revision number>: <Date>: <Comments>
--      <Revision number>: <Date>: <Comments>
--      <Revision number>: <Date>: <Comments>
--
-- Description:
--
-- <Description here>
--
-- Targeted device: <Family::PolarFireSoC> <Die::MPFS095T> <Package::FCVG484>
-- Author: <Name>
--
---------------------------------------------------------------------------------

library IEEE;

use IEEE.std_logic_1164.all;

entity test is
port (
    --<port_name> : <direction> <type>;
    port_name1 : IN  std_logic; -- example
    port_name2 : OUT std_logic_vector(1 downto 0)  -- example
```

From the point where we are right now, we first have to click on the *Build Hierarchy* button, when we click it, the HDL file that we have created is uploaded to the work library.



Afterwards, we right-click *Set As Root*, and the message disappears.

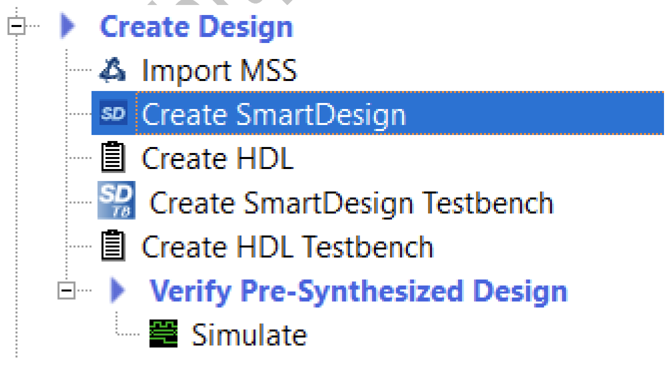


From now on we can develop the FW. Each new file that is added requires pressing the *Build Hierarchy* button again.

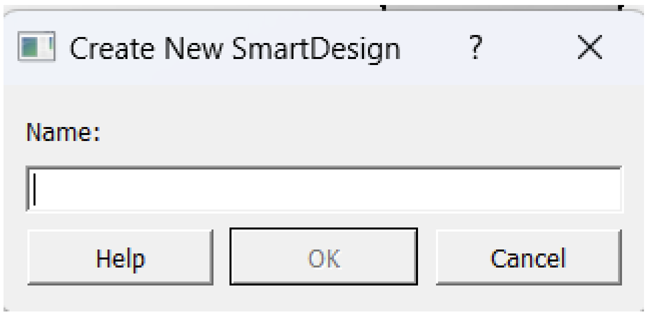
Working with blocks

In Libero there is another working option when working with FW, and that is the use of graphic blocks.

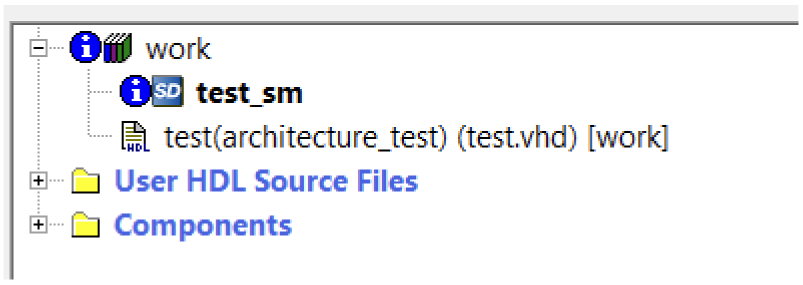
To work with them you first have to create a frame, Libero calls it SmartDesign.



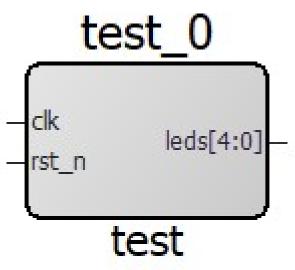
When creating a SmartDesign it asks us for a name.



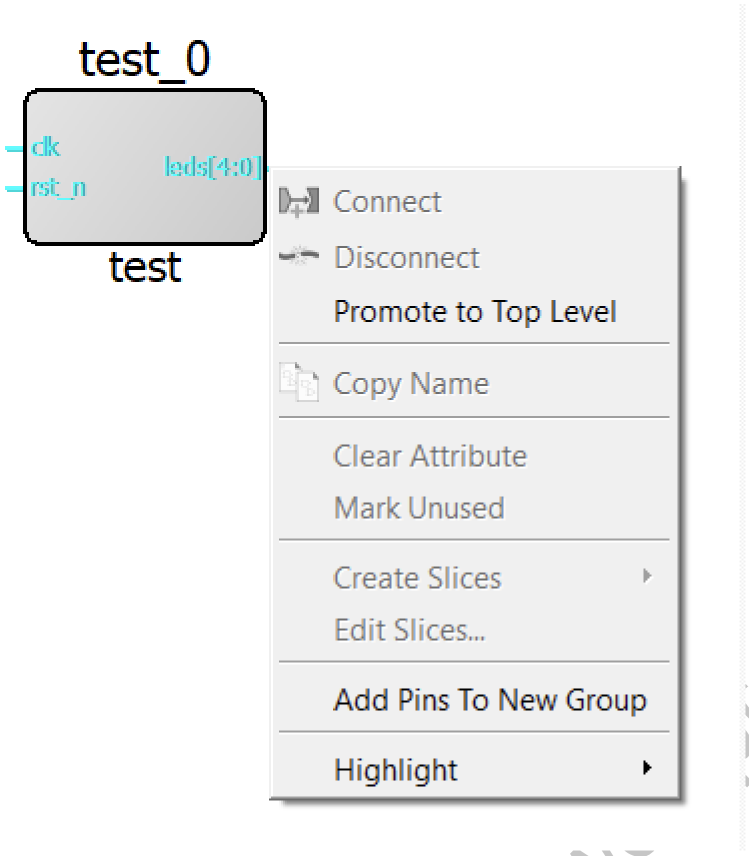
When creating the SmartDesign, the SmartDesign appears at the same level as the HDL code.



First you have to mark the SmartDesign as root and then you take the FW file and drag it to the SmartDesign. And the block appears, but without connections to the outside.



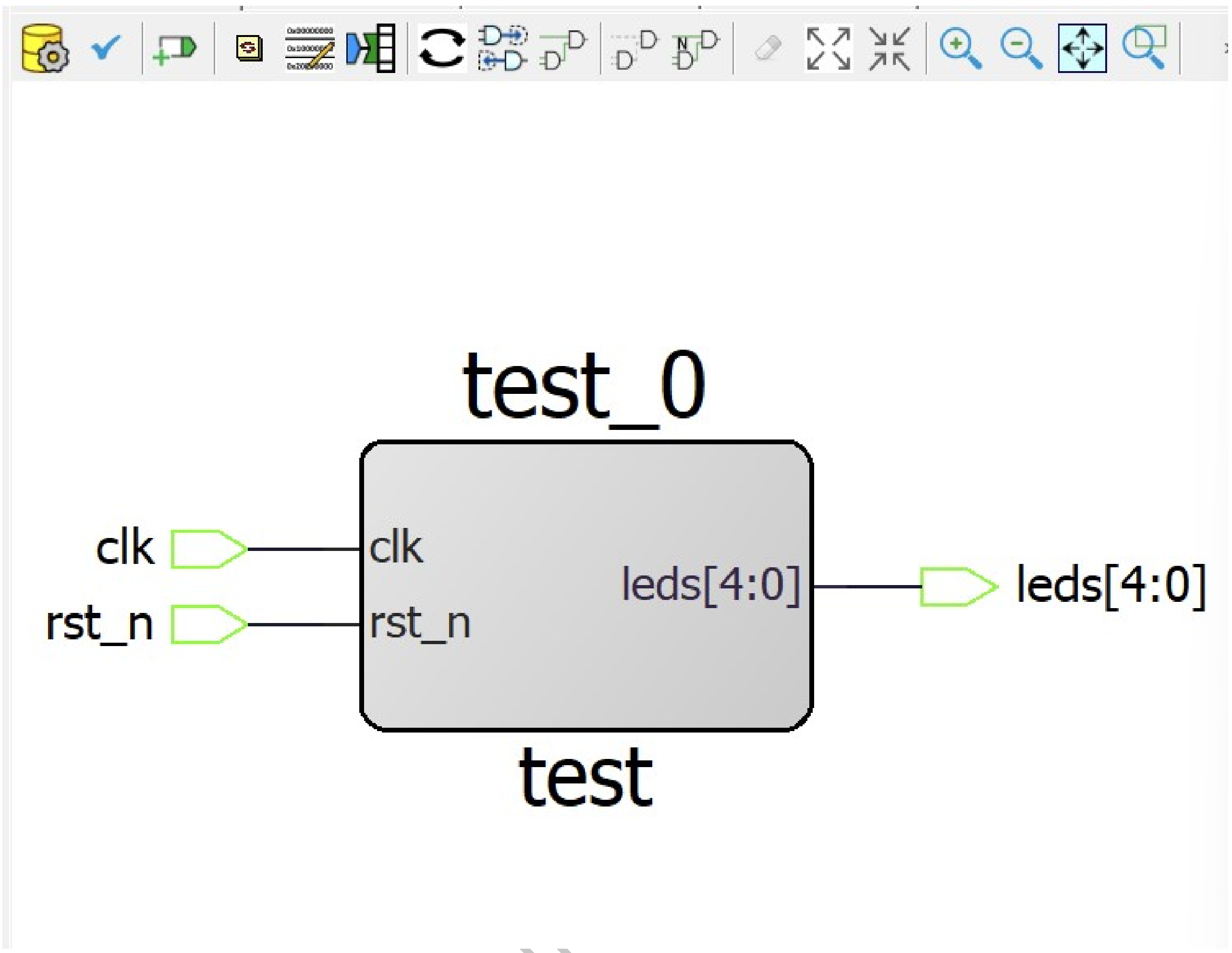
But clicking on the pins shows us an option called *Promote to Top Level*.



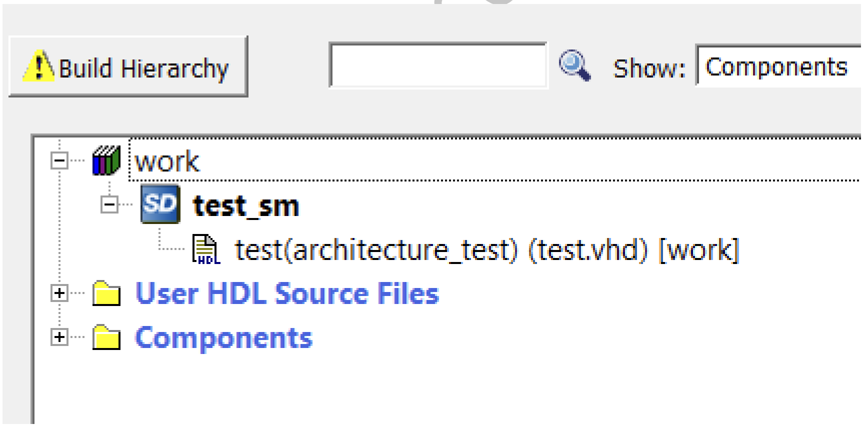
Once the pins have been created, we have to click on the yellow icon with the gear at the top left.
Once the pins have been created, we have to click on the yellow icon with the gear at the top left.

If any problem occurs when clicking on it, it will inform us below of the errors that have occurred, mostly, these errors are due to unconnected inputs, so we have to specify the value they will have.

https://soceame.wordpress.com/2025/03/09/basic-project-with-libero/



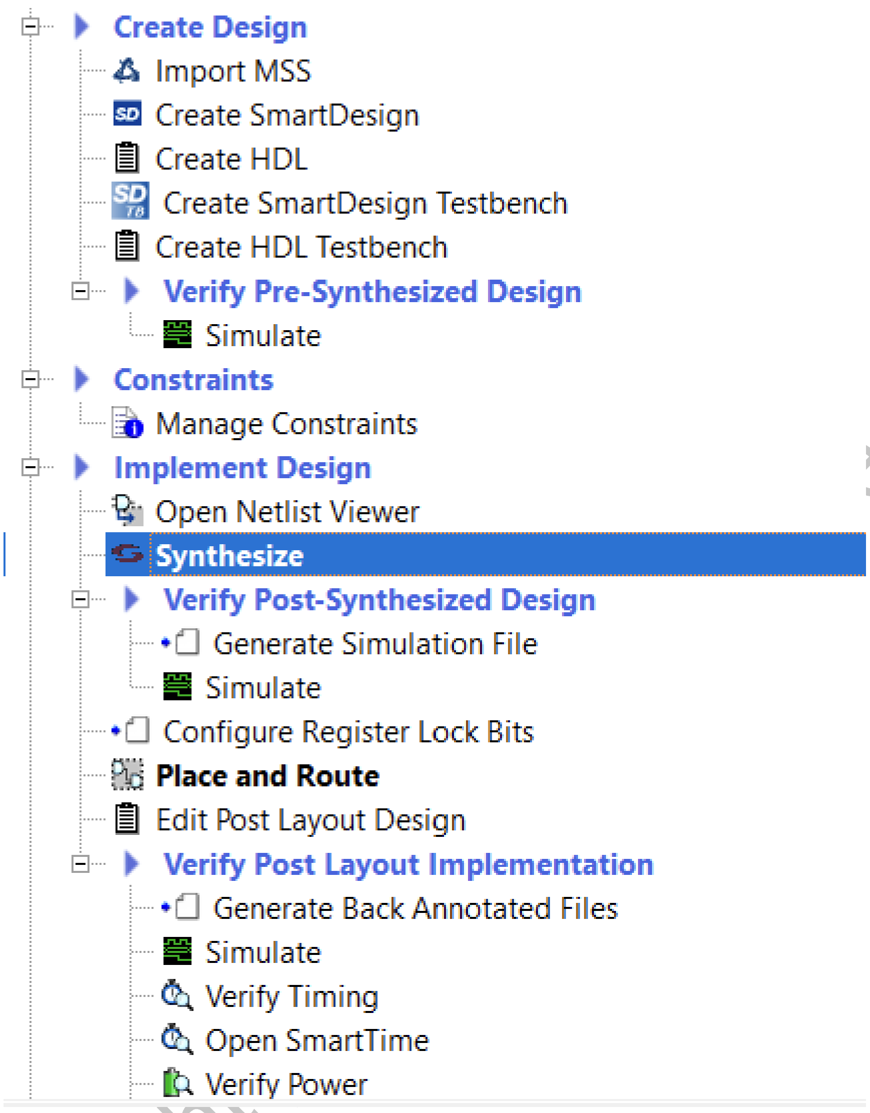
This icon makes the project structured.



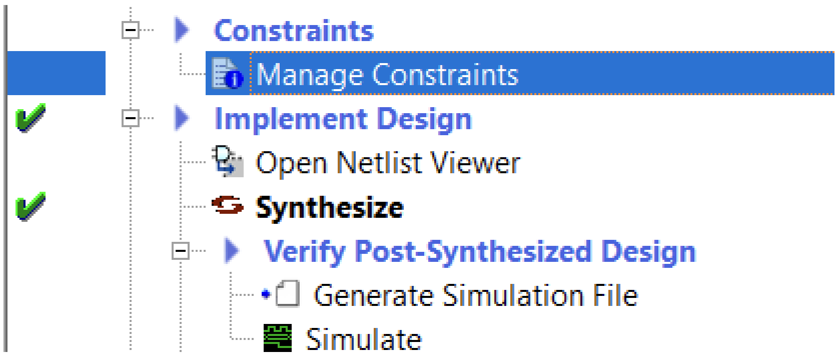
Now we just need to click on the *Build Hierarchy* button so that we can synthesize the project.

Generate the bitstream

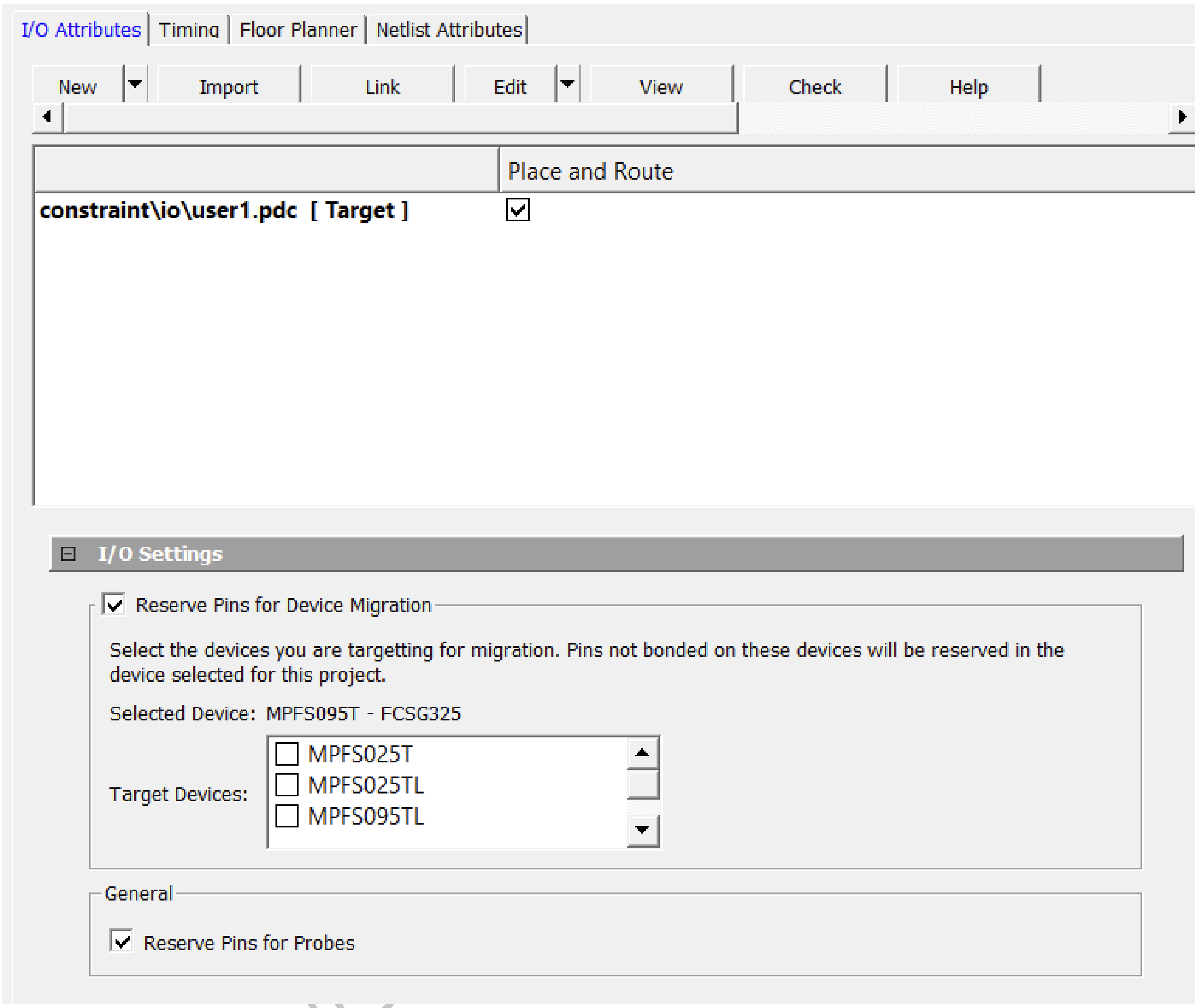
Once we have it developed, returning to *Design Flow,* we can synthesize the project. Synthesizing the project also allows us to unlock the selection of FPGA pins, because the pins are collected from the synthesis.



Once it tells us that the synthesis is correct, we open the *Manage Constraints* option.



This option allows us to manage the different restrictions of the project, both pins and times, etc. Pin constraints generate a PDC file, while timing constraints generate an SDC file.



constraint\io\user1.pdc [Target] ✓

I/O Settings

[✓] Reserve Pins for Device Migration

Select the devices you are targetting for migration. Pins not bonded on these devices will be reserved in the device selected for this project.

Selected Device: MPFS095T - FCSG325

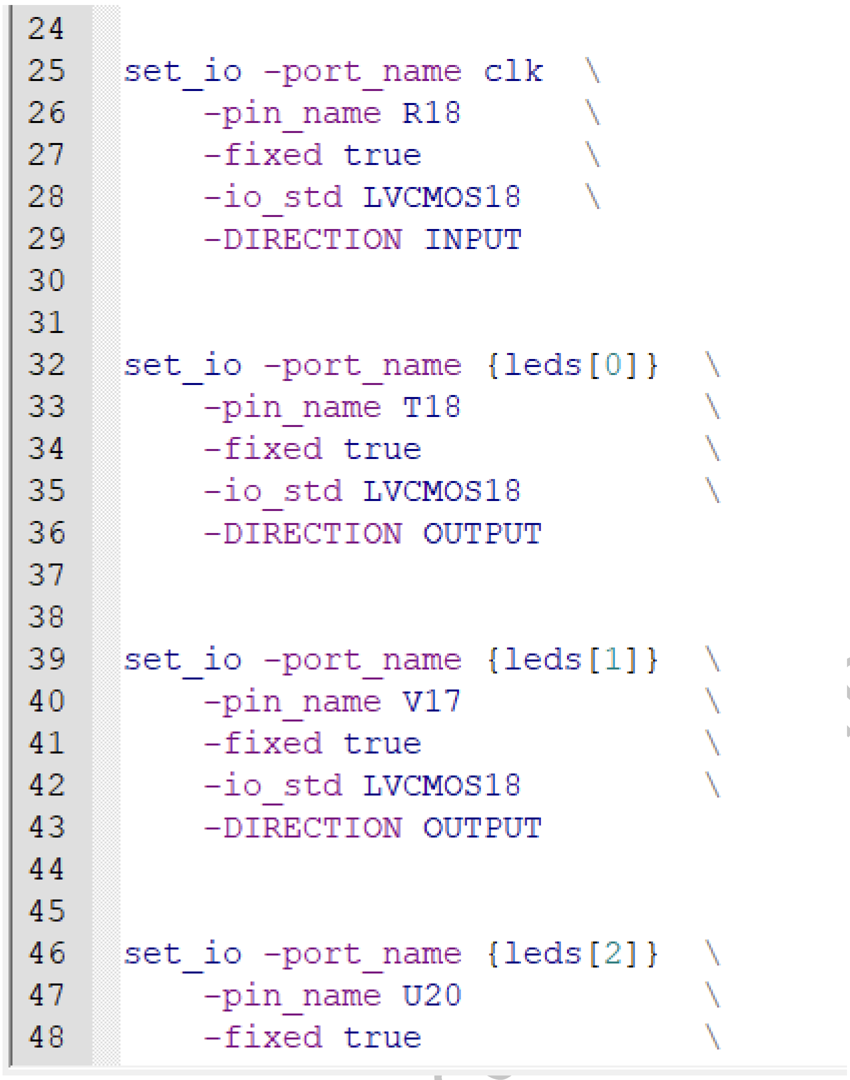
Target Devices: [ ] MPFS025T, [ ] MPFS025TL, [ ] MPFS095TL

General

[✓] Reserve Pins for Probes

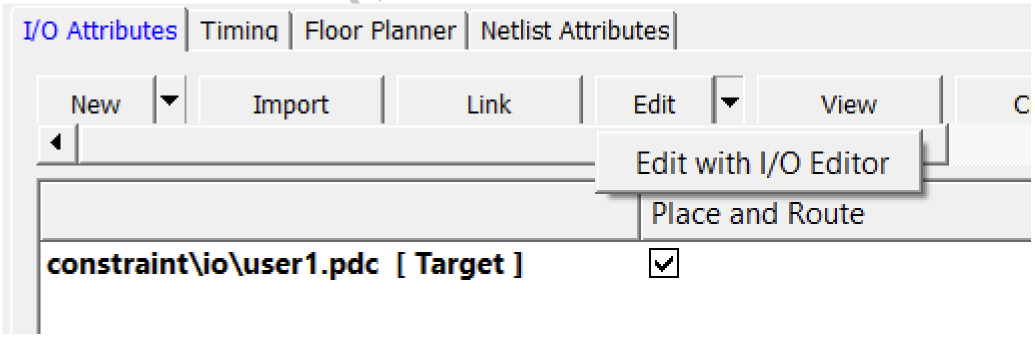
To create a pin constraint we have two options:

- Make the constraints by hand, which can be quite complex if you don't know the structure of the PDC file.



```

set_io -port_name clk  \
    -pin_name R18      \
    -fixed true        \
    -io_std LVCMOS18   \
    -DIRECTION INPUT


set_io -port_name {leds[0]}  \
    -pin_name T18            \
    -fixed true              \
    -io_std LVCMOS18         \
    -DIRECTION OUTPUT


set_io -port_name {leds[1]}  \
    -pin_name V17            \
    -fixed true              \
    -io_std LVCMOS18         \
    -DIRECTION OUTPUT


set_io -port_name {leds[2]}  \
    -pin_name U20            \
    -fixed true              \
```

- The other option is to use the interactive tool provided by Libero. To access it, open the *Edit* drop-down menu and click on the *Edit with I/O Editor* option.



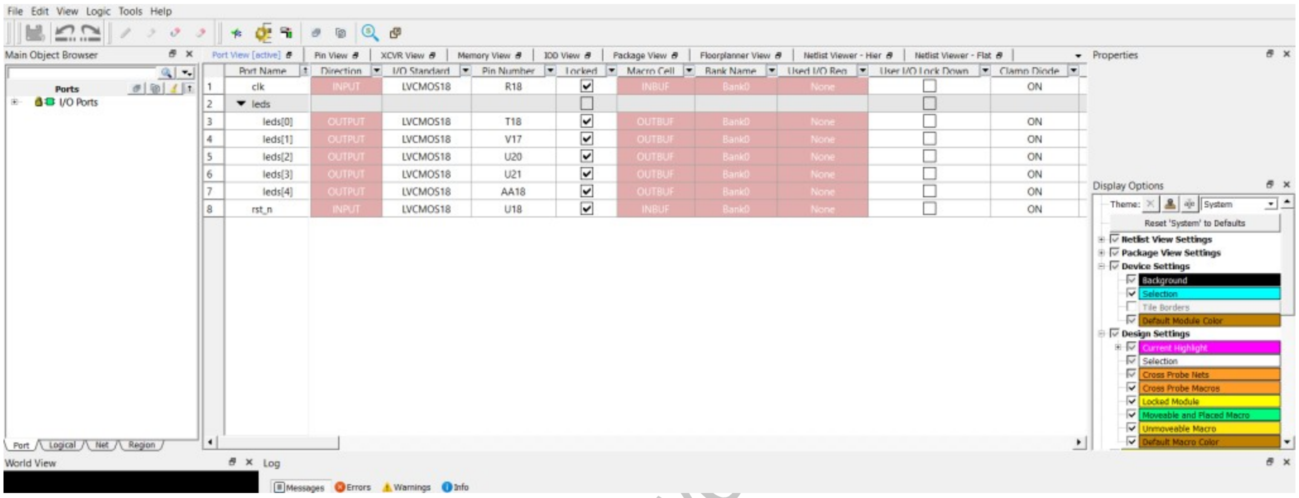
constraint\io\user1.pdc [Target] ✓

When you open this option, an interactive tool opens where the user can select the pin they want for the FW port.

NOTE: this tool works by banks, so that if you select a pin at a specific voltage, it sets the entire bank to that same voltage. Also, if you select a wrong pin and assign it to a different one, you can leave the entire bank marked at that voltage, and the pin selector will only let you use pins from that bank. It is advisable to be careful.

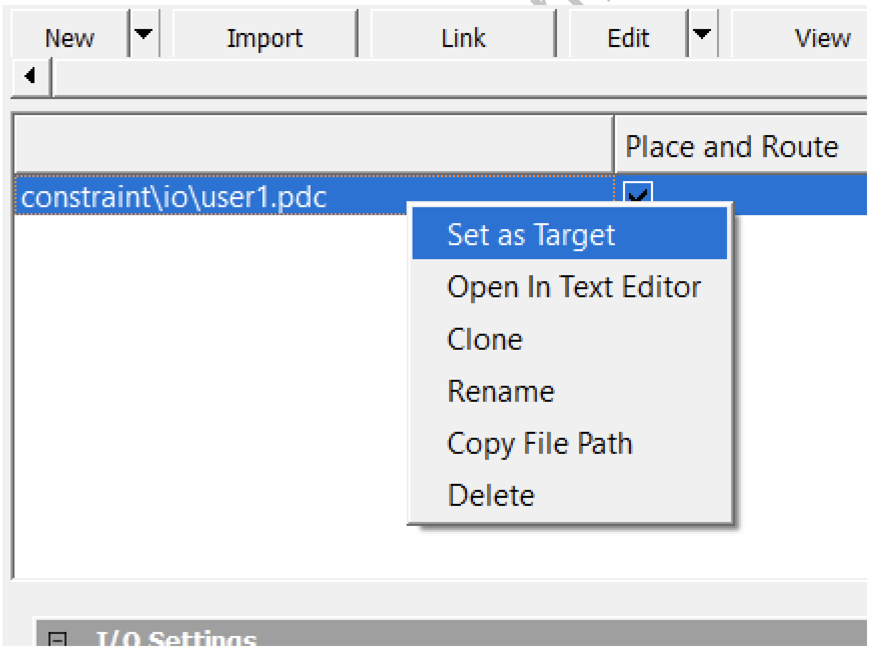
NOTE 2: to solve the problem of preselecting the pins, you have to manually delete the pin and press enter, so that it stops marking that pin. Many of the problems with this tool are solved by manually deleting the pin.

NOTE 3: if you cannot change the voltage of the pins that appears by default, you have to go to the *Project* option, and inside look for *Project Settings*, and then *Device settings*.

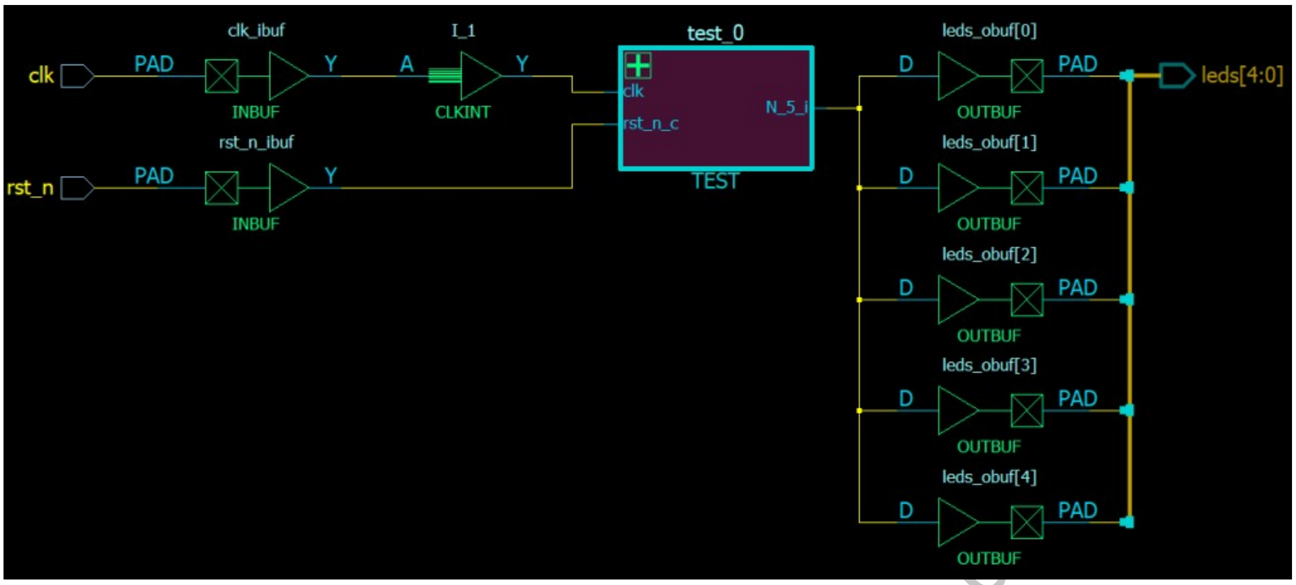


	Port Name	Direction	I/O Standard	Pin Number	Locked	Macro Cell	Bank Name	Used I/O Reg	User I/O Lock Down	Clamp Diode
1	clk	INPUT	LVCMOS18	R18	✓	INBUF	Bank0	None	☐	ON
2	leds				☐				☐	
3	leds[0]	OUTPUT	LVCMOS18	T18	✓	OUTBUF	Bank0	None	☐	ON
4	leds[1]	OUTPUT	LVCMOS18	V17	✓	OUTBUF	Bank0	None	☐	ON
5	leds[2]	OUTPUT	LVCMOS18	U20	✓	OUTBUF	Bank0	None	☐	ON
6	leds[3]	OUTPUT	LVCMOS18	U21	✓	OUTBUF	Bank0	None	☐	ON
7	leds[4]	OUTPUT	LVCMOS18	AA18	✓	OUTBUF	Bank0	None	☐	ON
8	rst_n	INPUT	LVCMOS18	U18	✓	INBUF	Bank0	None	☐	ON

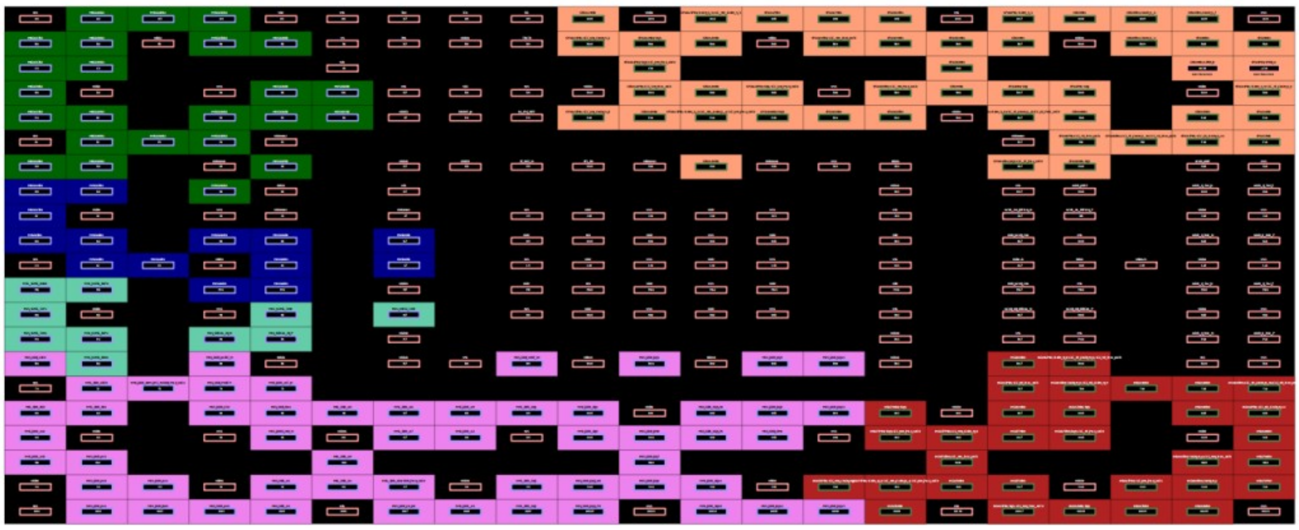
Once finished, you have to click on the save icon, and then create the pin restriction file for the project. The next thing to do is to mark the pin file as **Target**.



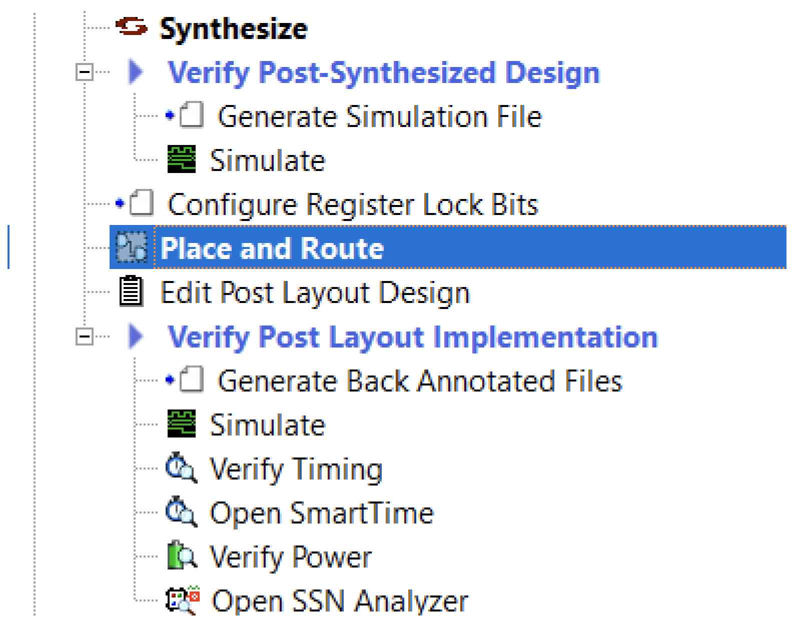
NOTE: This editor also allows us to see the synthesized model that Libero has generated for us, it is in the Netlist Viewer tab (if a blue message appears, just click on it to make it appear).



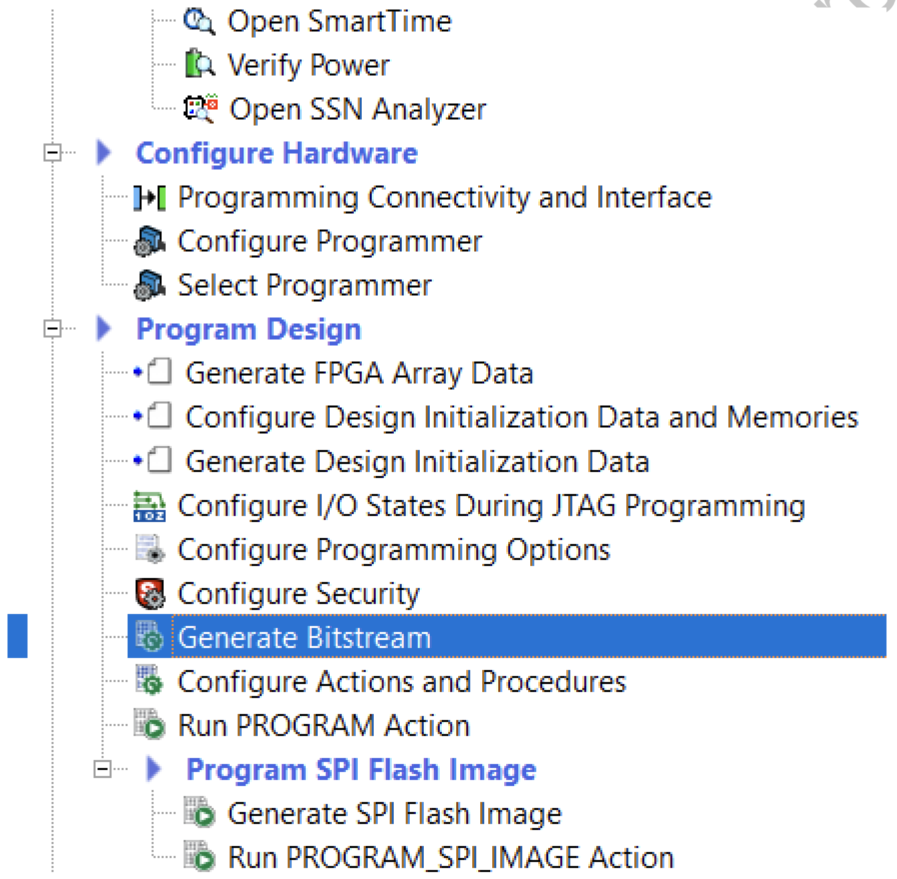
It also allows us to see other things, such as the location of the chip's pins or the type of selectable pins on the chip.



Once the pins have been selected, the implementation is carried out, to do this, click on the Place and Route option.

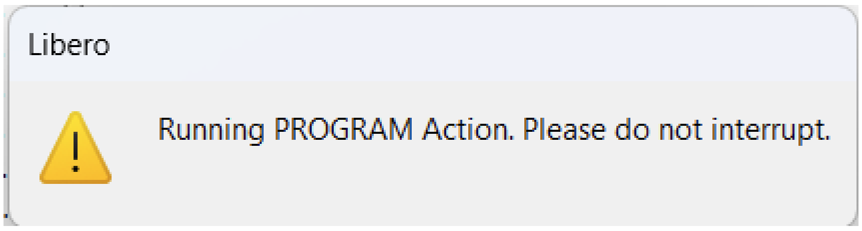


Once finished, the bitstream can be generated. To do this, select the *Generate Bitstream* option.


Open SmartTime
Verify Power
Open SSN Analyzer
Configure Hardware
Programming Connectivity and Interface
Configure Programmer
Select Programmer
Program Design
Generate FPGA Array Data
Configure Design Initialization Data and Memories
Generate Design Initialization Data
Configure I/O States During JTAG Programming
Configure Programming Options
Configure Security
Generate Bitstream
Configure Actions and Procedures
Run PROGRAM Action
Program SPI Flash Image
Generate SPI Flash Image
Run PROGRAM_SPI_IMAGE Action

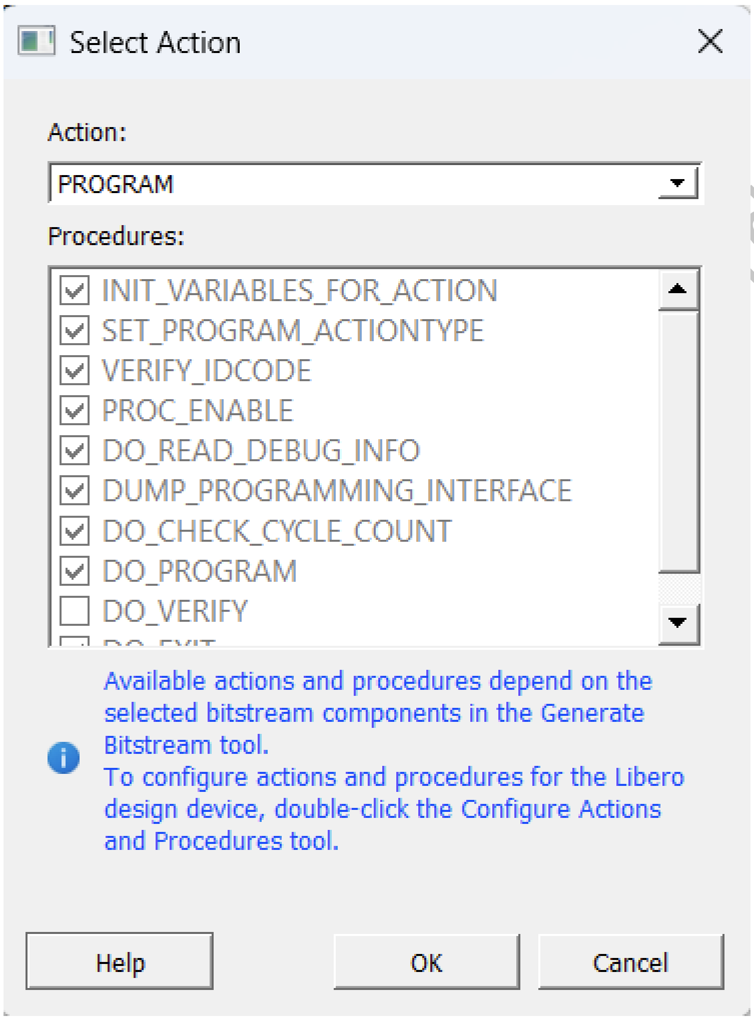
Once generated, we already have the project's bitstream generated, we just need to save it to the FPGA or export it.

- To save it to the FPGA, you have to connect the FPGA to the computer and click on the *Run PROGRAM Action* option. While it is being saved, this tab appears and below in Libero a bar appears with the bitstream loading status.


Libero

Running PROGRAM Action. Please do not interrupt.

The *Run PROGRAM Action* option also has the option to perform more actions, to do this, right click on *Run PROGRAM Action* and click on *Select Action*. The PROGRAM action is to record the FPGA.


Select Action

Action: PROGRAM

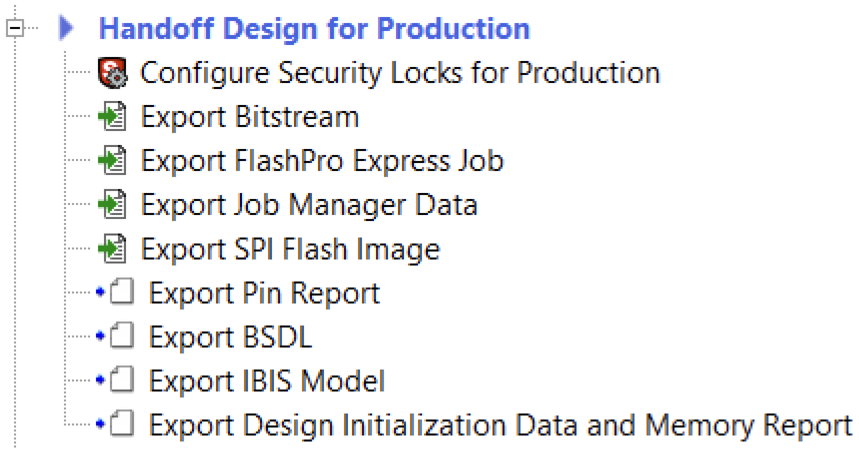
Procedures:

- [✓] INIT_VARIABLES_FOR_ACTION
- [✓] SET_PROGRAM_ACTIONTYPE
- [✓] VERIFY_IDCODE
- [✓] PROC_ENABLE
- [✓] DO_READ_DEBUG_INFO
- [✓] DUMP_PROGRAMMING_INTERFACE
- [✓] DO_CHECK_CYCLE_COUNT
- [✓] DO_PROGRAM
- [ ] DO_VERIFY

Available actions and procedures depend on the selected bitstream components in the Generate Bitstream tool.
To configure actions and procedures for the Libero design device, double-click the Configure Actions and Procedures tool.

- To export it you have to go to the Libero export options. Here several options appear, the main ones are to export the bitstream and to export to generate a project in *FlashPro Express*.

https://soceame.wordpress.com/2025/03/09/basic-project-with-libero/


Handoff Design for Production
Configure Security Locks for Production
Export Bitstream
Export FlashPro Express Job
Export Job Manager Data
Export SPI Flash Image
Export Pin Report
Export BSDL
Export IBIS Model
Export Design Initialization Data and Memory Report

The two export options are described at the end of this previous entry.

https://soceame.wordpress.com/2025/03/09/flashpro-express-tutorial-installing-and-running/