# ESIstream

The **E**fficient **S**erial **I**nterface

## ESIstream 62B64B

## MICROCHIP FPGA MPF300T-1FCG115E2

## USER GUIDE

*ESIstream is a license-free high efficiency serial interface protocol based on 14b/16b encoding.*

*Its main benefits are low overhead and ease of hardware implementation.*

# INTRODUCTION

The package ESIstream allows to generate VHDL design examples to get started with ESIstream 62B64B High-speed serial interface.

This package provides VHDL sources, constraint files, TCL scripts for project generation, VHDL testbench files for simulation.

For technical support, please get the team involved and contact us using ESIstream contact web page or at GRE-HOTLINE-BDC@Teledyne.com

# TERMINOLOGY

| | |
|---|---|
| ADC | Analog to Digital Converter |
| ASIC | Application-Specific Integrated Circuit |
| BE | Bit Error |
| CB | Clock Bit |
| CDR | Clock and Data Recovery |
| DAC | Digital to Analog Converter |
| ESIstream | the Efficient Serial Interface |
| ESS | ESIstream Synchronization Sequence |
| FAS | Frame Alignment Sequence |
| FPGA | Field Programmable Gate Array |
| GT | Gigabit Transceiver |
| HSSL | High Speed Serial Lane |
| ILA | Integrated Logic Analyzer (a Vivado feature) |
| LFSR | Linear Feedback Shift Register |
| PAS | PRBS Alignment sequence |
| PL | Programmable Logic |
| PLL | Phase Locked Loop |
| PRBS | Pseudo-Random Binary Sequence |
| RX | Receiver |
| SSO | Slow Synchronization Output. GT reference clock. |
| TX | Transmitter |
| UART | Universal Asynchronous Receiver Transmitter |
| UI | Unit Interval: time to send a bit through the serial interface. |
| Xcvr | Transceiver |

## DISCLAIMER

This is free and unencumbered software released into the public domain.

Anyone is free to copy, modify, publish, use, compile, sell, or distribute this software, either in source code form or as a compiled bitstream, for any purpose, commercial or non-commercial, and by any means.

In jurisdictions that recognize copyright laws, the author or authors of this software dedicate any and all copyright interest in the software to the public domain. We make this dedication for the benefit of the public at large and to the detriment of our heirs and successors. We intend this dedication to be an overt act of relinquishment in perpetuity of all present and future rights to this software under copyright law.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

THIS DISCLAIMER MUST BE RETAINED AS PART OF THIS FILE AT ALL TIMES.

ESI stream

TABLE OF CONTENTS

## FIGURES

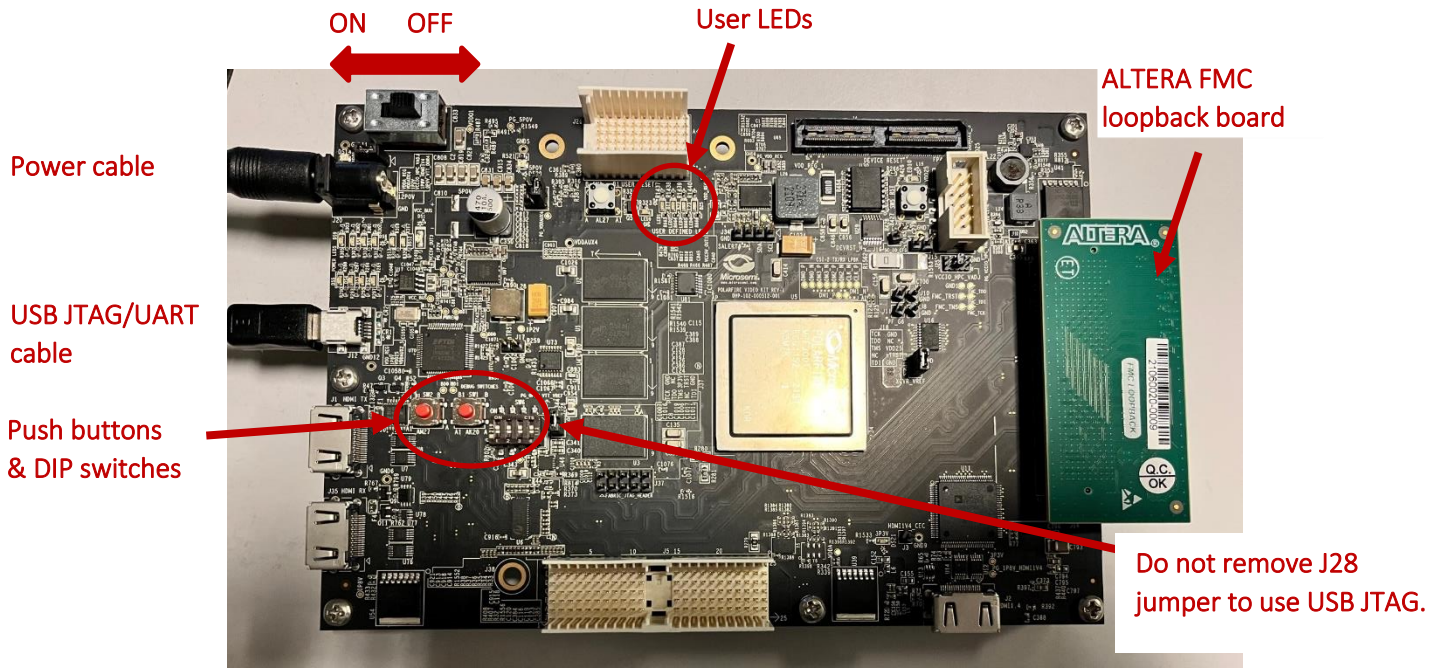
# 1 HARDWARE

## 1.1 MATERIAL LIST

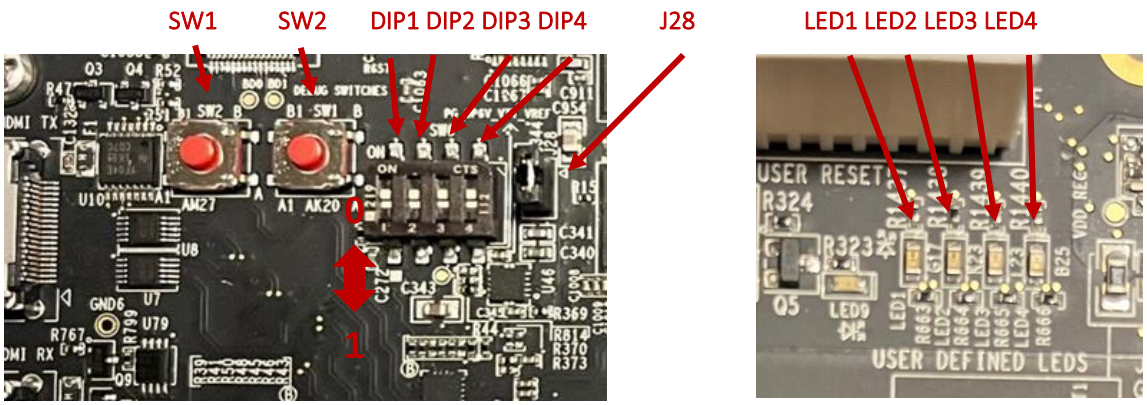
- MPF300-VIDEO-KIT-NS
  - ✓ Product web page: https://www.microsemi.com/existing-parts/parts/150804
- XM107 FMC loopback board
  - ✓ Product web page: https://www.whizzsystems.com/loopback-card/
  - ✓ Schematic: https://www.xilinx.com/content/dam/xilinx/support/documents/boards_and_kits/xtp090.pdf
  - ✓ User guide: https://docs.xilinx.com/v/u/en-US/ug539
- ALTERA FMC loopback board
  - ✓ Mouser web page: https://eu.mouser.com/ProductDetail/Terasic-Technologies/S0485?qs=81r%252BiQLm7BT%2FhHjxmUuTyA%3D%3D
  - ✓ Schematic: https://www.intel.com/content/www/us/en/support/programmable/articles/000086949.html

## 1.2 USER INTERFACE

- Connect Power cable
- Connect USB JTAG/UART cable
- Connect ALTERA FMC loopback board



## 1.3 USER LEDS, DIP SWITCHES AND PUSH BUTTONS




| User interface | Description |
|---|---|
| SW1 | Reset, 1s min |
| SW2 | SYNC, 1s min |
| DIP1 | d_ctrl(0). Default: 1 for ramp test pattern. (*) |
| DIP2 | ESIstream clock bit (CB) enable, for synchronization monitoring. Default: 1 |
| DIP3 | ESIstream RX and TX scambling enable. Default: 1 |
| DIP4 | ESIstream RX disparity processing enable. Default:1 |
| LED1 | ESIstream RX and TX modules are ready when ON. |
| LED2 | ESIstream RX lanes are ready when ON. |
| LED3 | Receiver frames Checking module valid status |
| LED4 | Clock bit and ESIstream data bits error status.<br>When there is one bit error detected on one ESIstream frame transmitted then LED is turned ON. Push reset button to clear the error. |

(*) If d_ctrl = “00” then all zeroes test pattern.
(*) If d_ctrl = “01” or “10” then ramp test pattern.
(*) If d_ctrl = “11” then all one’s test pattern.

# 3 LIBERO PROJECTS

The package contains two projects, a 32-bit and a 64-bit.

32-bit project will generate a design implementation using Gigabit Transceiver (GT) with a serialization factor of 64-bit and a deserialization factor of 32-bit.

64-bit project will generate a design implementation using Gigabit Transceiver (GT) with a serialization factor of 64-bit and a deserialization factor of 64-bit.

It means that the raw data logic vector at Gigabit Transceiver (GT) outputs for receivers (RX) is configured with a size of 32-bit or 64-bit.

32-bit or 64-bit implementation selection is a trade-off between minimum link latency, minimum logic resources and frames frequency.

Receiver 32-bit implementation reduces link latency, uses less logic resources but it increases frame frequency.

64-bit implementation increases link latency, uses more logic resources but it reduces frame frequency, it can help to relax FPGA design timing constraints.

$$f_{rx\_frame\_32\text{-}bit} = 2*f_{rx\_frame\_64\text{-}bit}$$

This project offers a VHDL design example to test the ESIstream serial link using both ESIstream TX and RX modules and a loopback board connecting FPGA HSSLs TX outputs on FPGA HSSLs RX inputs.

DIP switches, push-buttons, LEDs allow to quickly synchronize the high-speed serial link, to start data transfer of a known ramp pattern and to check that there is no communication error.

Synchronization signal, errors status, received frames, or other signals can be directly mapped to Libero Identify Debug to be analyzed.

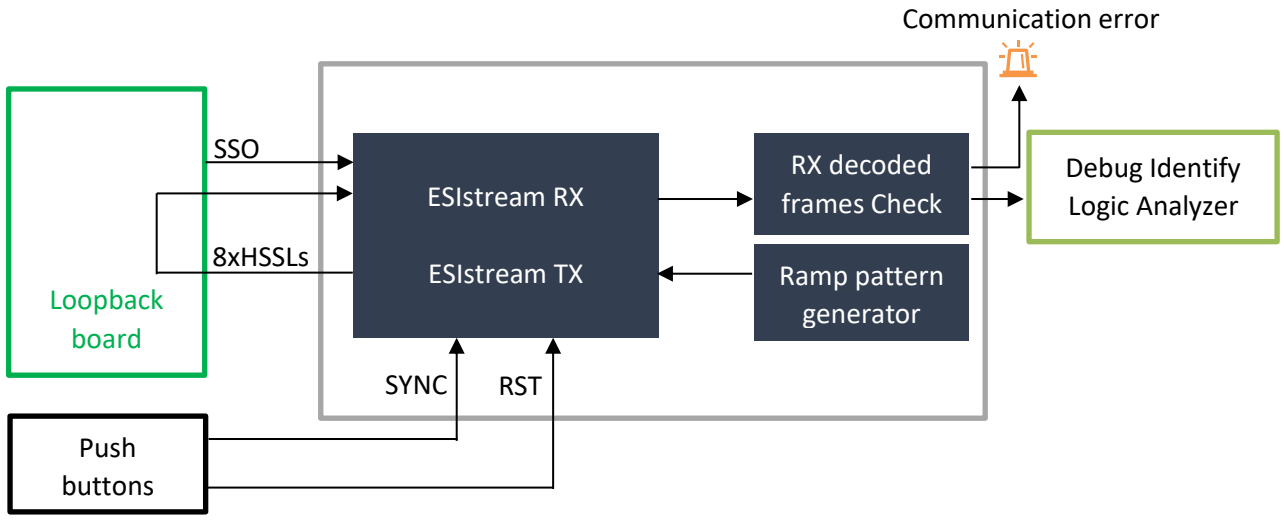
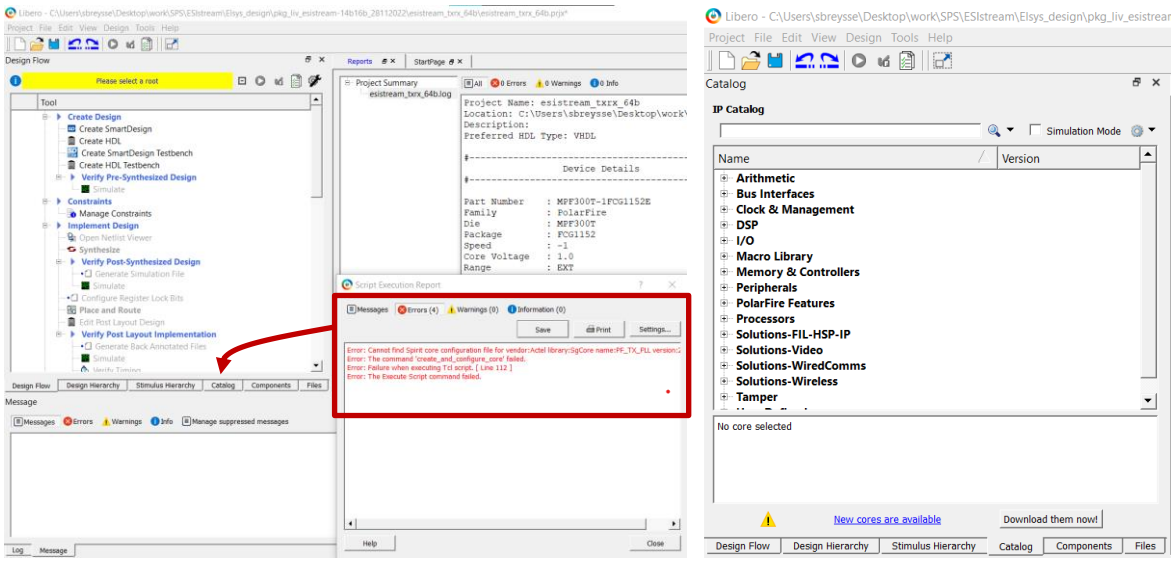



*Figure 1: ESIstream TXRX with loopback board project overview.*

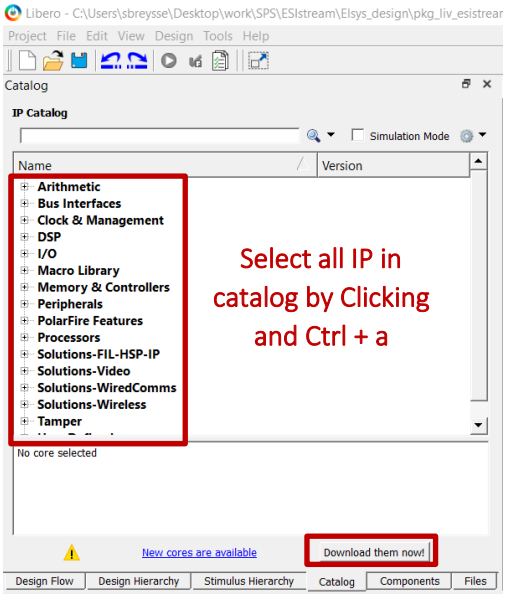
## 3.1 How to download and to install Libero IPs from catalog?

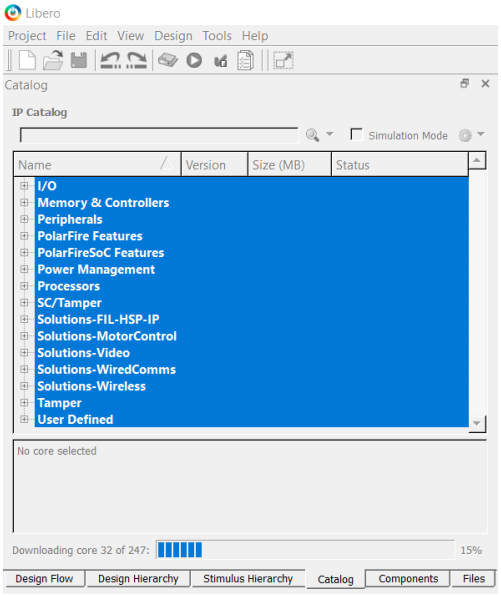
If this is the first time after Libero install, then this error appears.

- Click on Catalog tab
- Click on Download them now! To download new available cores.




- In Catalog tab, select all IP categories and click on Download them now!



## 3.2 How to download ESIstream package?

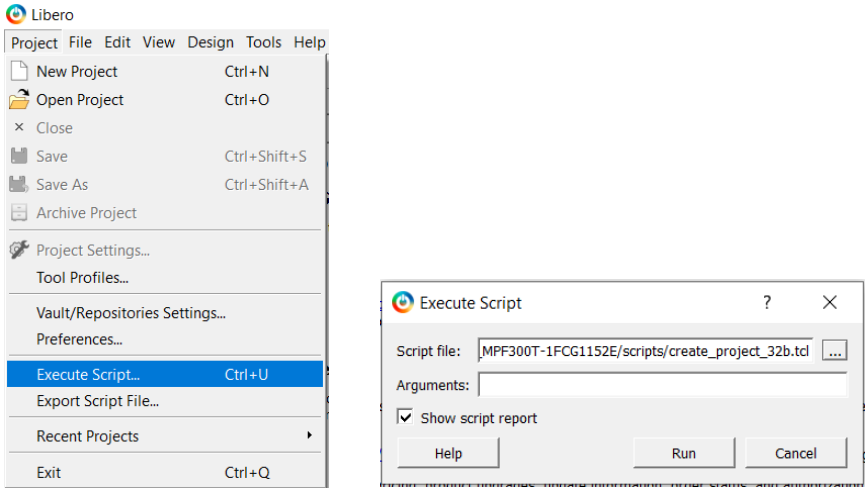
- Go to ESIstream web page: https://www.esistream.com/package/esistream-62b64b-package
- Click on download button.


## 3.3 How to generate Libero project?

- Open Libero 2022.1:

| Package name | Version to use |
|---|---|
| Package_ESIstream62b64b_Microchip_MPF300T-1FCG1152E | Libero 2022.1 |

- Project > Execute Script…




- Select the tcl script that fits the requirements of the design and click on Run
  - /scripts/create_project_32b.tcl or /scripts/create_project_64b.tcl

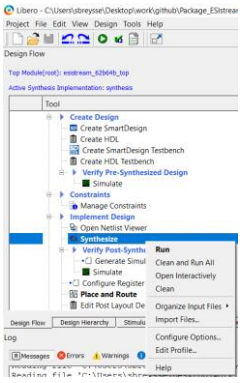
- When the project is built, a Libero project file (.prjx) is available in the corresponding directory (~/esistream_62b64b_[32b/64b]).

- Project > Open Project

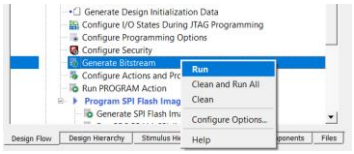
It is now possible to compile, simulate or modify the example design using Microchip Libero toolchain.

## 3.4 How to generate bitstream?

- Click on synthesize. Once the project is synthetized, open “manage constraints”,

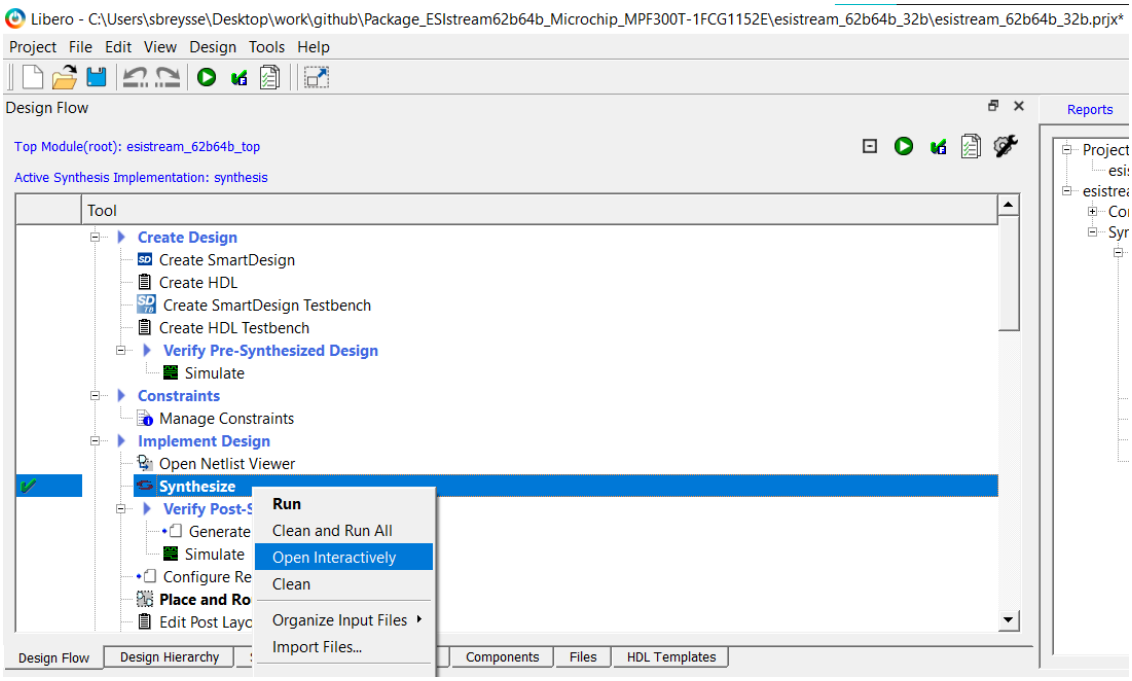


- Right-click on generate bitstream and on Run.

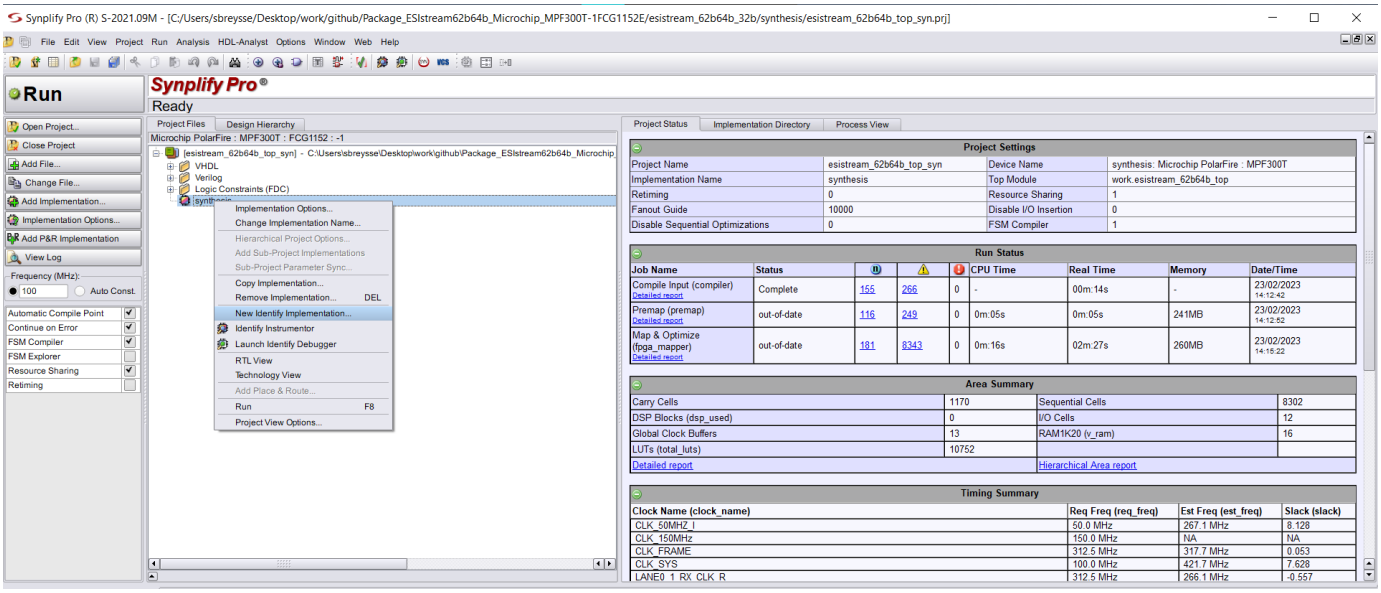


## 3.5 How generate bitstream and program FPGA with Identify debugger?

- Open the project.
- Click on synthesize.
- Right-click on “synthesize “and select “open interactively” to launch the synthesis tool Synplify Pro:



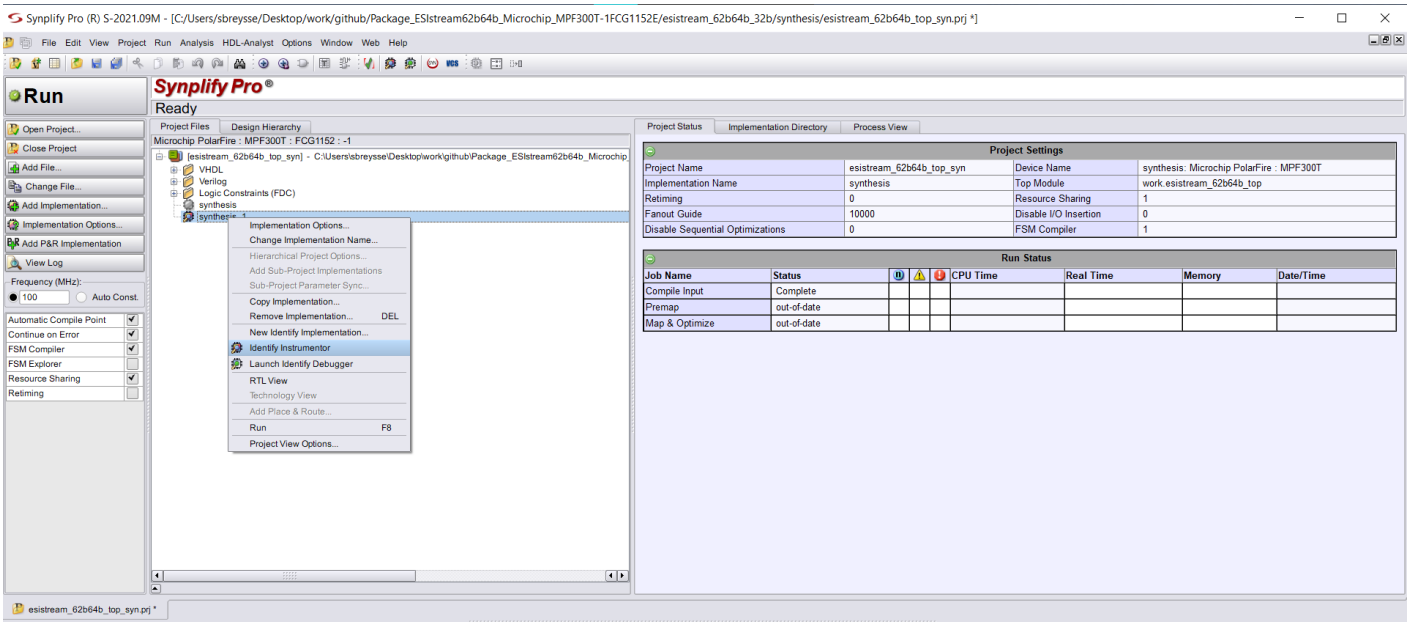
- Create a New Identify Implementation (see the figure below)



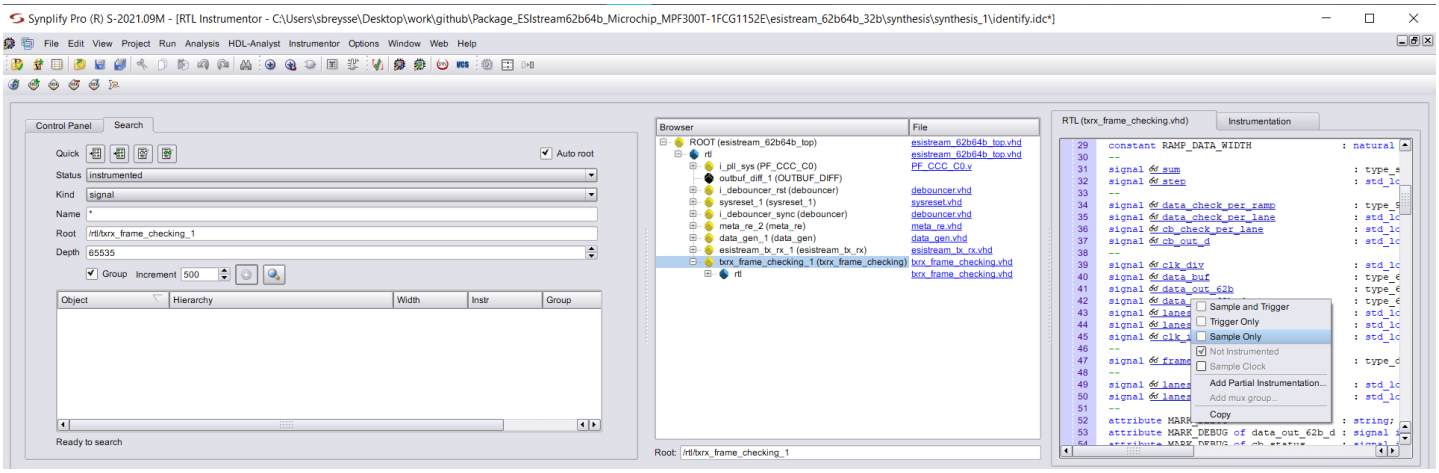

SynplifyPro creates a new Identify implementation with the default location at \synthesis\synthesis_1.

You can edit the implementation name in the Implementation Results tab. Do not change the path of the Results Directory.

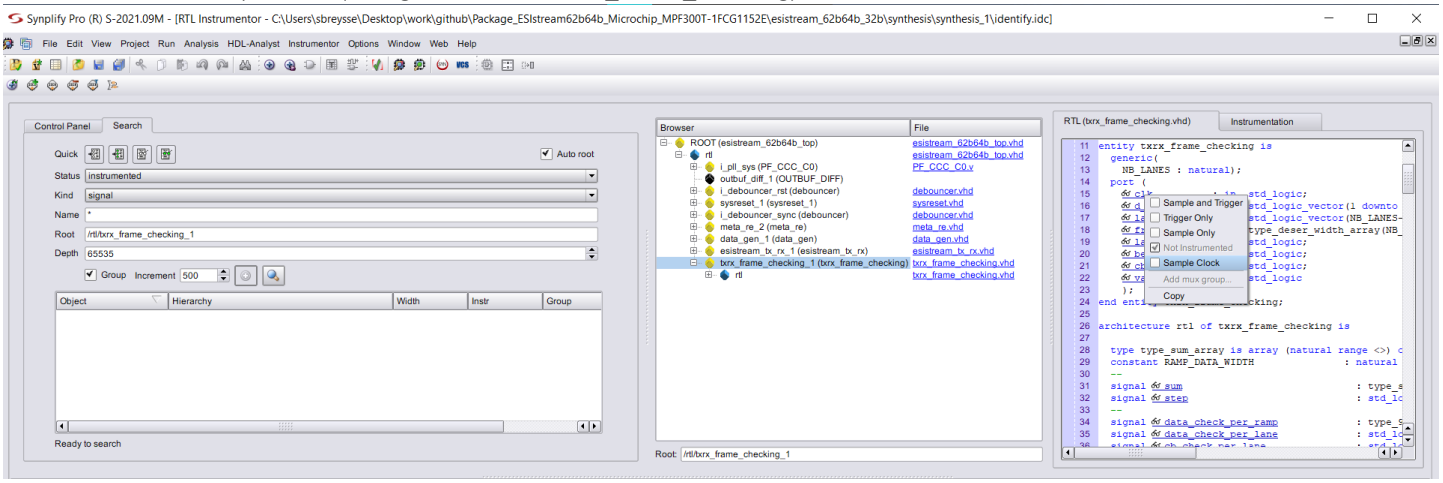
- Launch “Identify Instrumentator” (Right-click on synthesis_1)



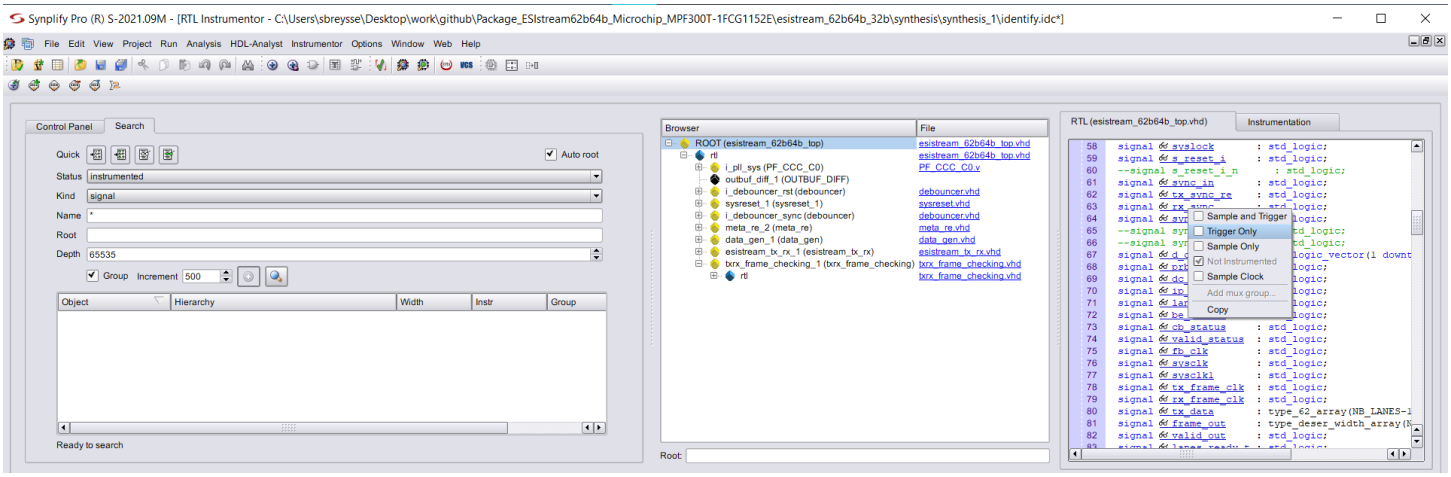

- Select debug signals and a sampling clock.
  - ✓ Sample only (data_out_62b_d signal of module txrx_frame_checking)



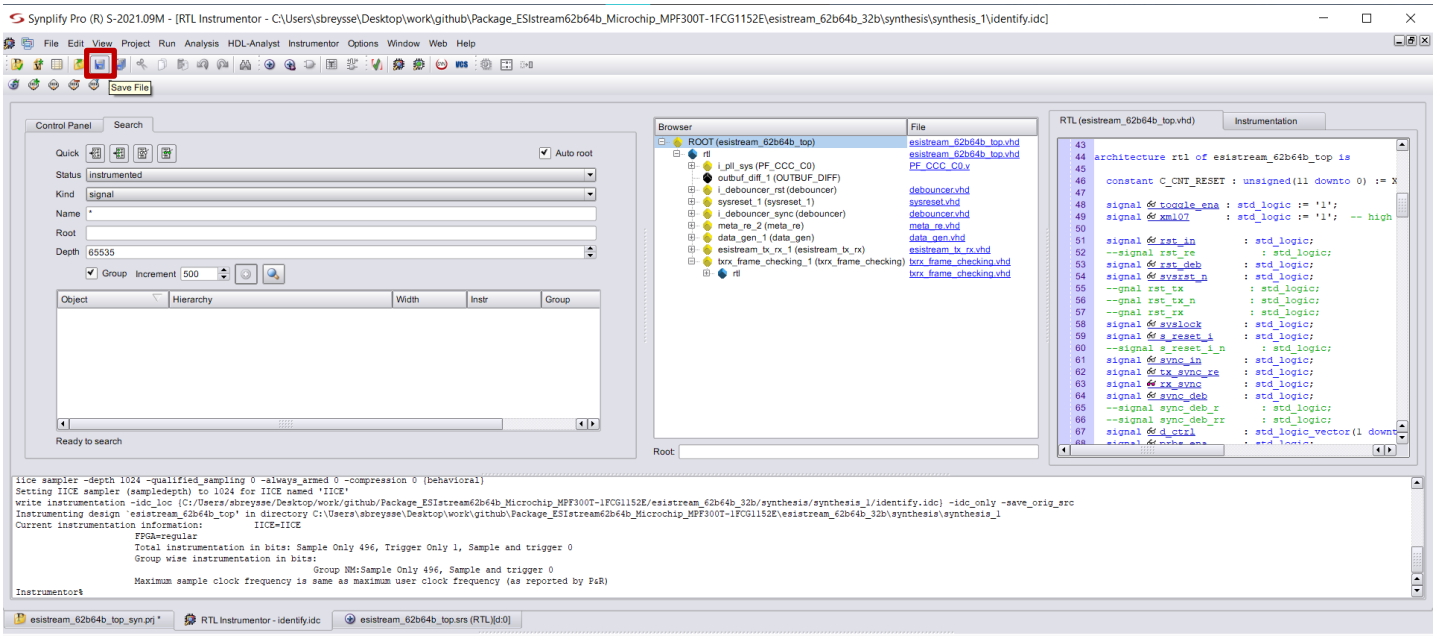

✓ Sample clock (clk signal of module txrx_frame_checking)

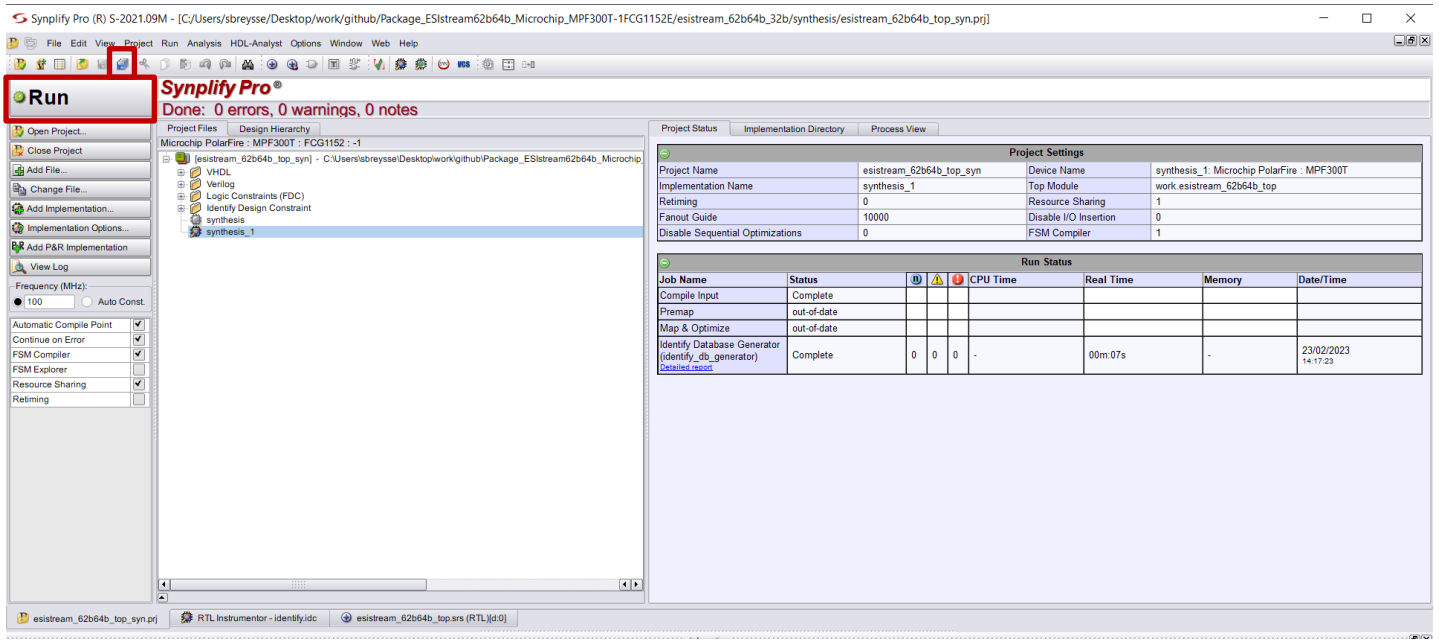



✓ Trigger (rx_sync signal of top module esistream_62b64b)

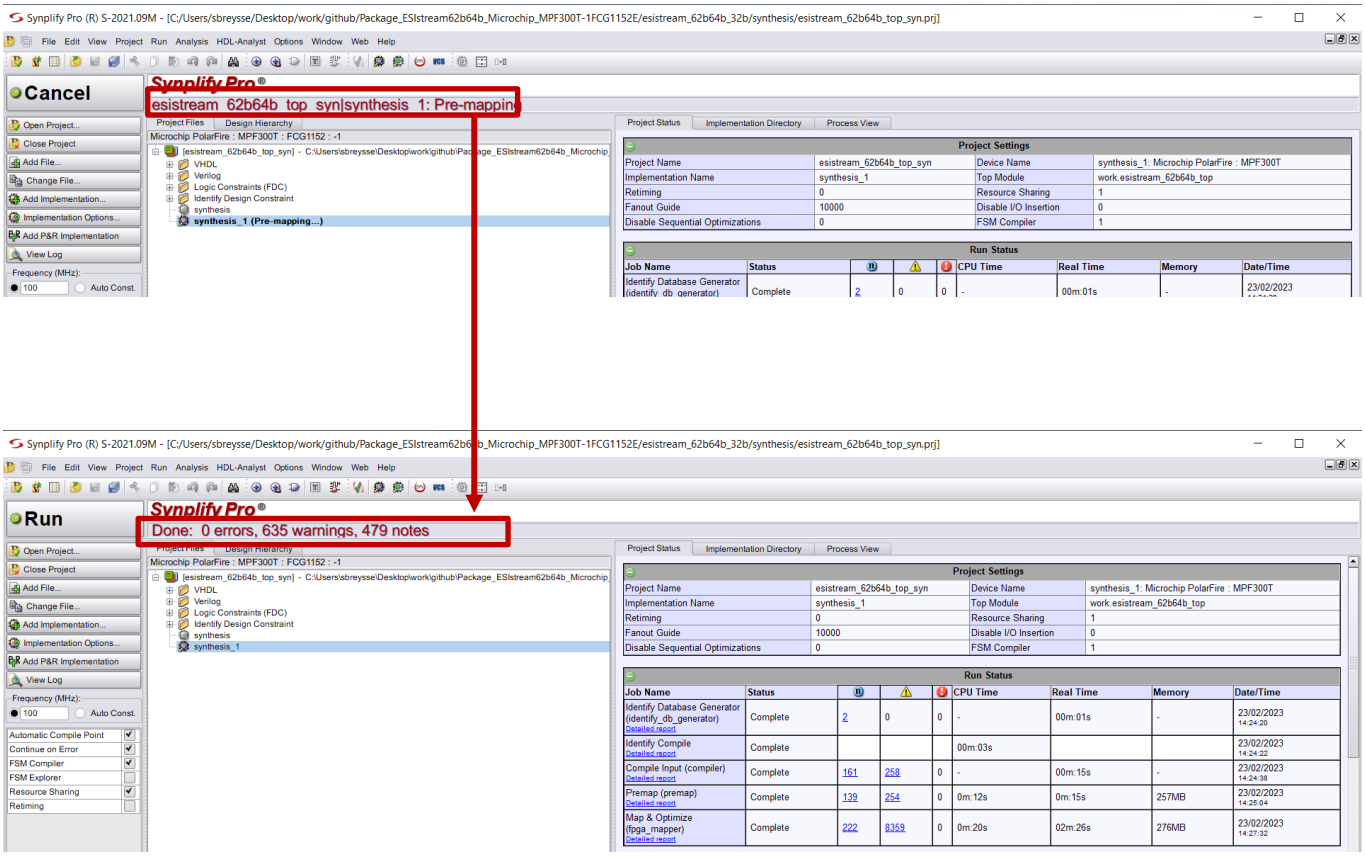


- Save the project then click on run.



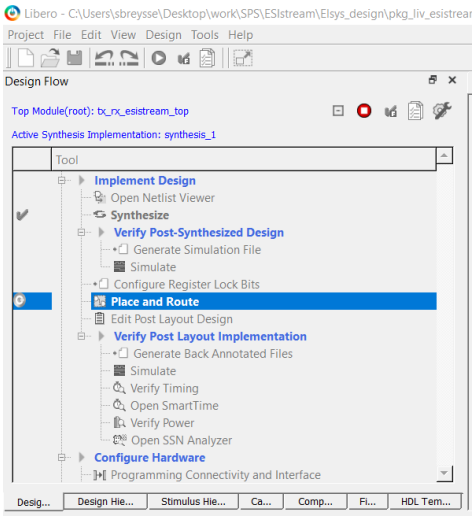



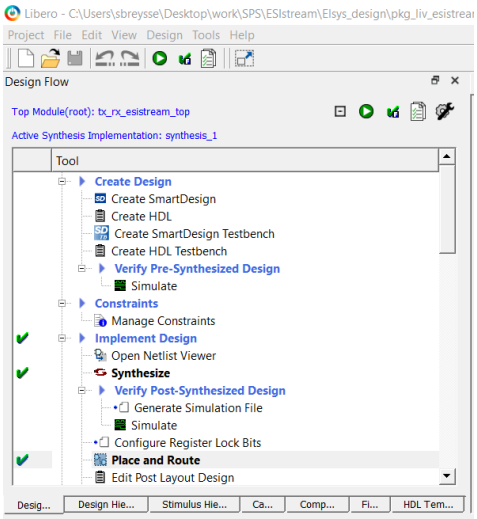

- Wait for end of mapping…



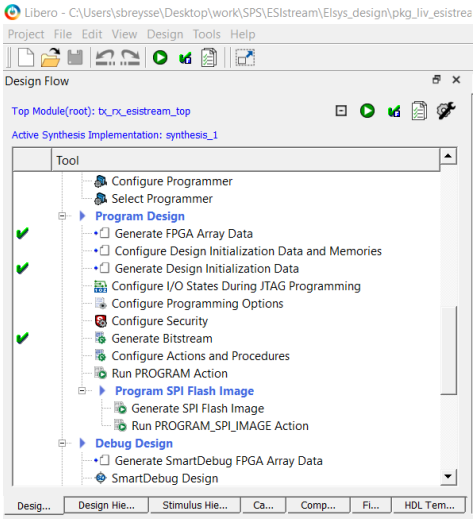

- Close Symplify Pro.

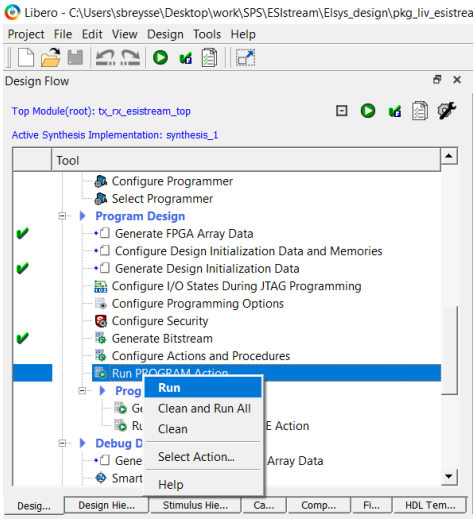
- Run Place and Route



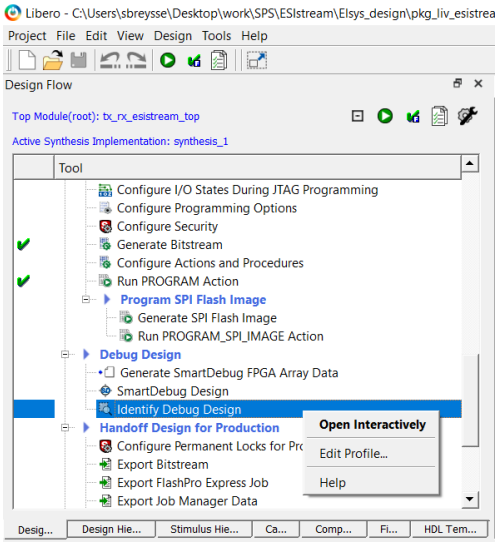




- Generate Bitstream and Run Program Action

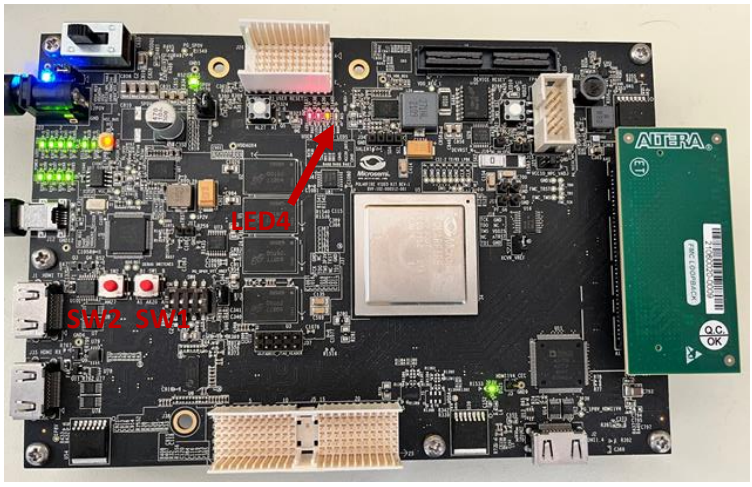


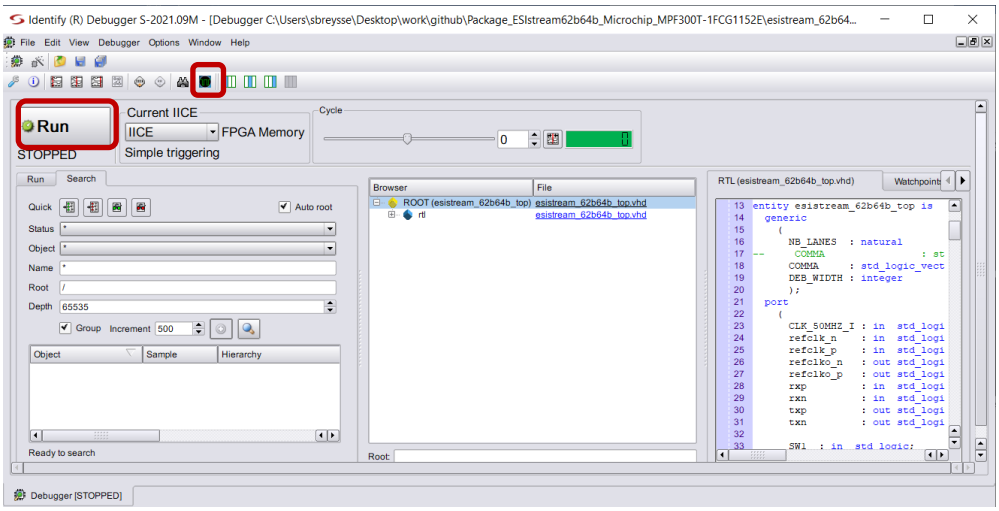

- Wait for end of programming…
- All debug signals can be analyzed by launching the tool Identify Debug Design.

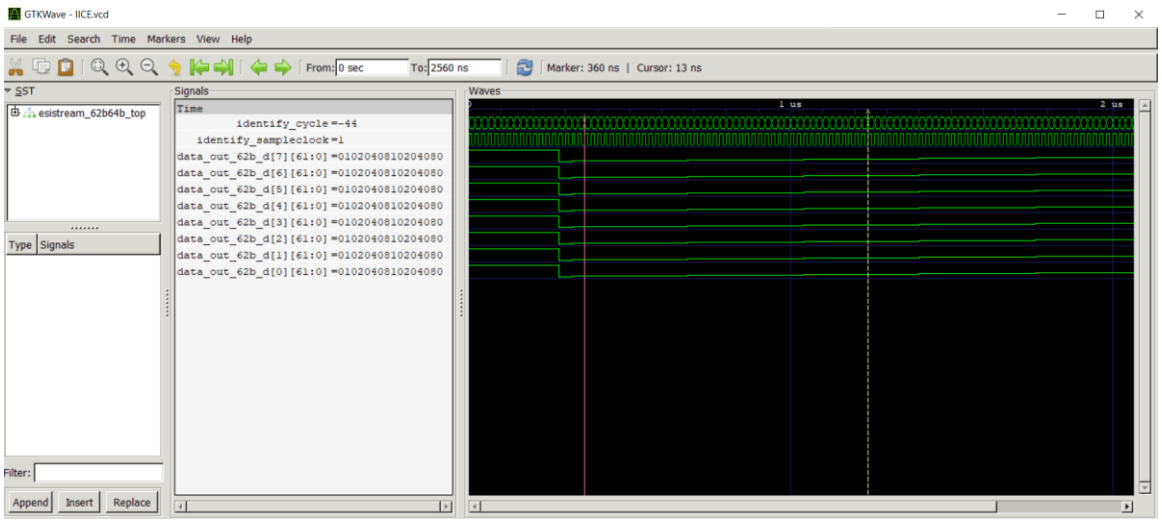


- Push Reset push-button (SW1) and then push SYNC push-button (SW2).
  - ✓ LED4 is OFF indicating there is no communication error.



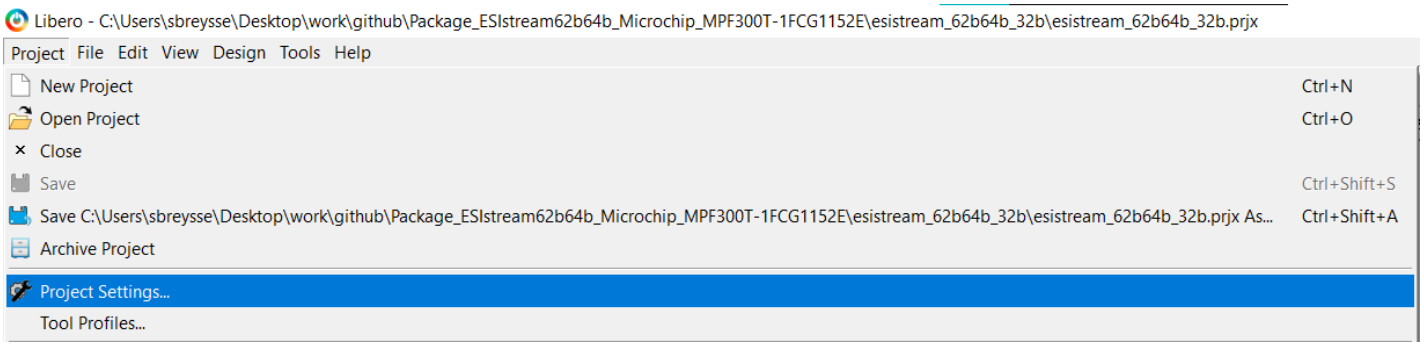

- Click on Run and open Waveform viewer to display ramp data for each serial lane [0:7].



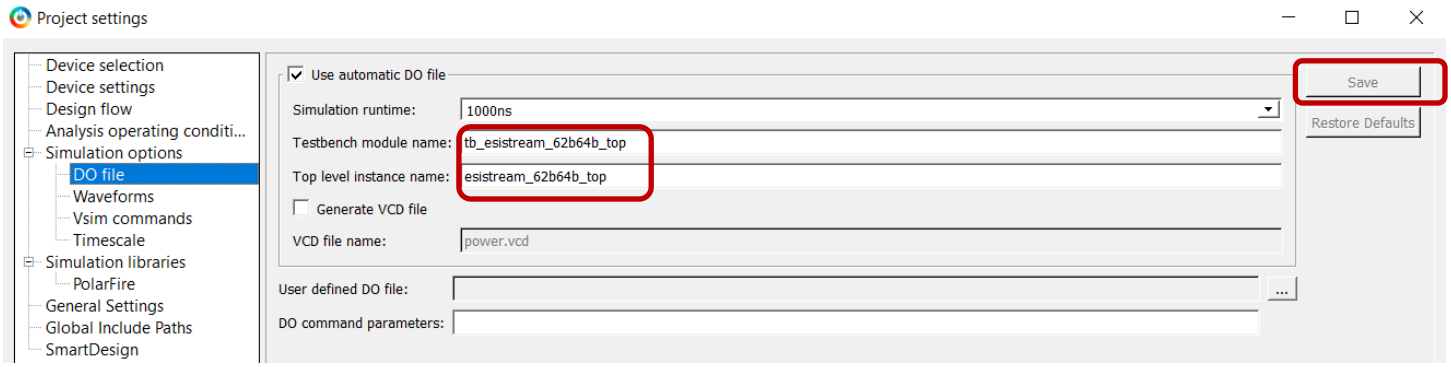



## 3.6 How to simulate FPGA design example?

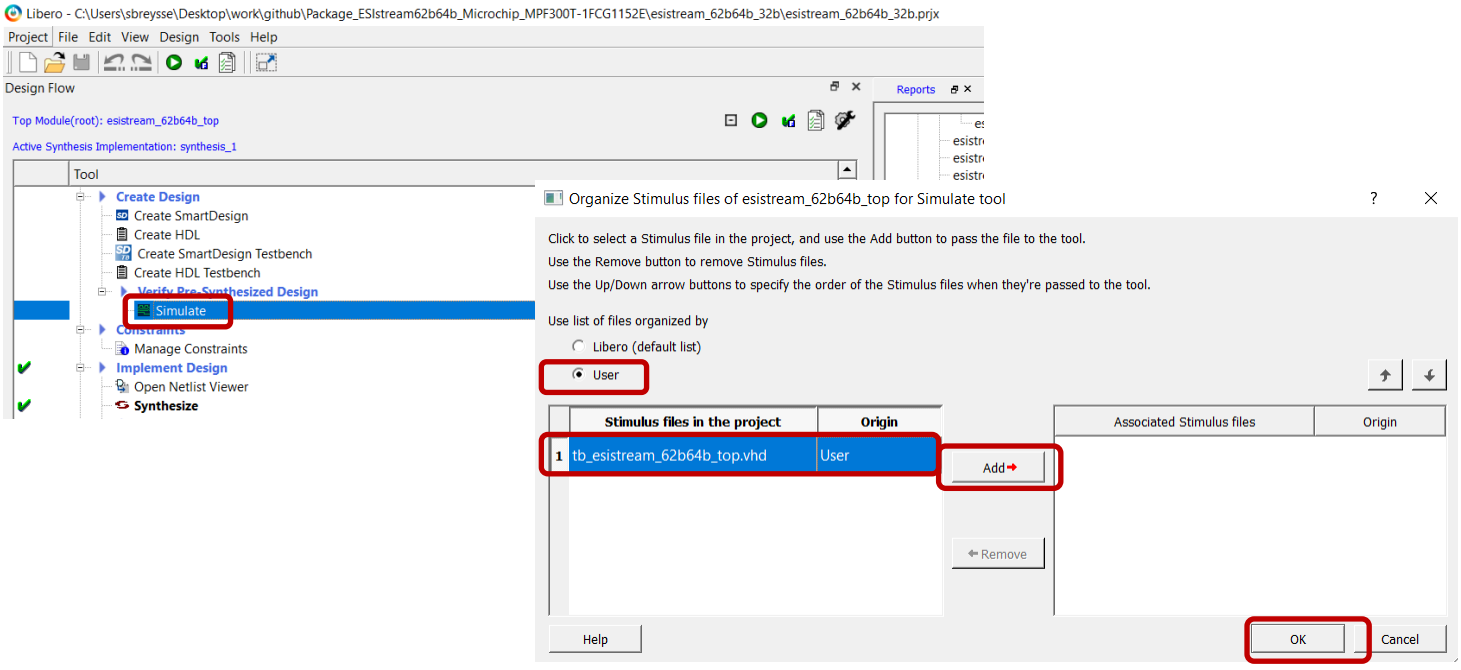
- Click on Project and Project Settings…




- Click on DO file
  - Fill Testbench module name and Top level instance name fields and click on Save button.




- Launch simulation and add user stimulus:




- Run simulation for 40 µs.