

# How to debug using Synplify Pro and Identify Debugger in Libero

Created by: David Rubio G.

Blog post: <https://soceame.wordpress.com/2025/03/11/how-to-debug-using-synplify-pro-and-identify-debugger-in-libero/>

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

GitHub: <https://github.com/DRubioG>

Last modification date: 11/03/25

This is a little-known thing about Microchip FPGAs/SoCs, and that is that they can be debugged using other tools included in the installation. These tools are **Synplify Pro** (to select the debug signals) and **Identify Debugger** (to view the debugged signals), both tools developed by Synopsys.

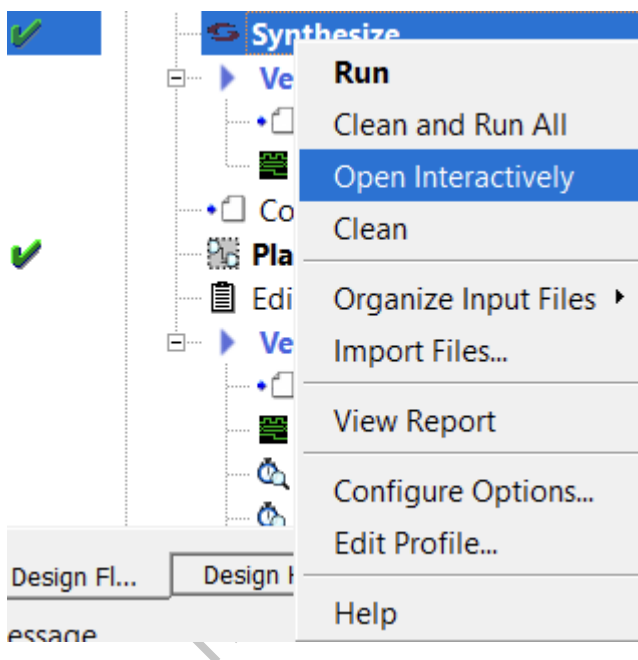
**NOTE:** these two tools do something similar to the **ILA** blocks from Xilinx or the **Signal Tap Logic Analyzer** from Altera. These tools can be opened directly from Libero or from the application itself.

*(Previous steps)*

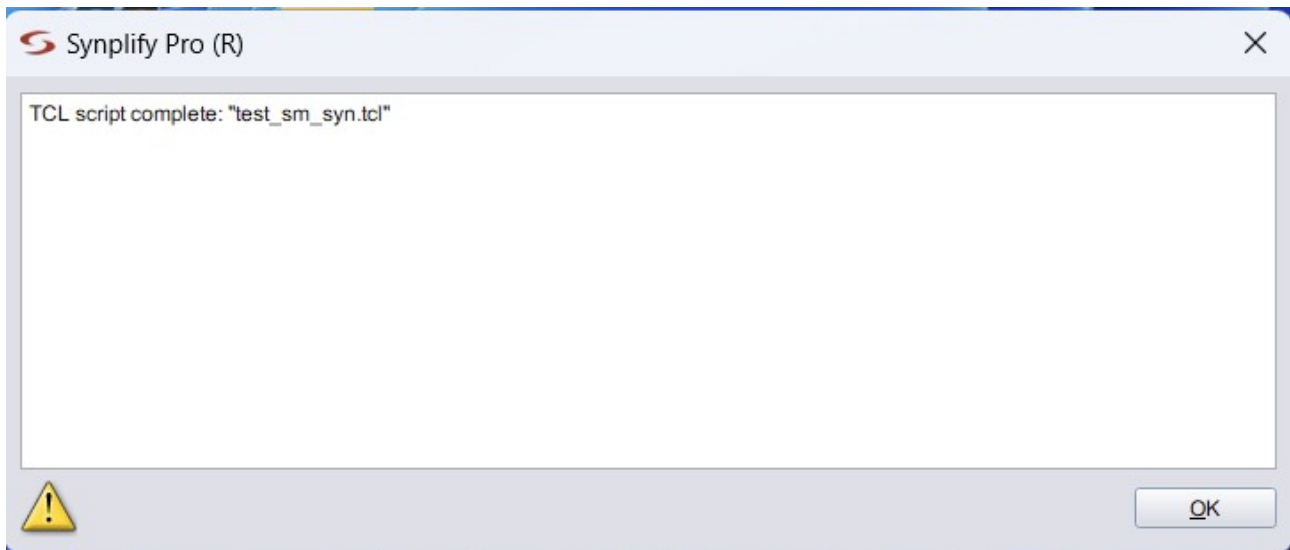
This debugging system first requires the user to have synthesized a project in Libero. To do this, click on the *Synthesize* option.

## Synplify Pro

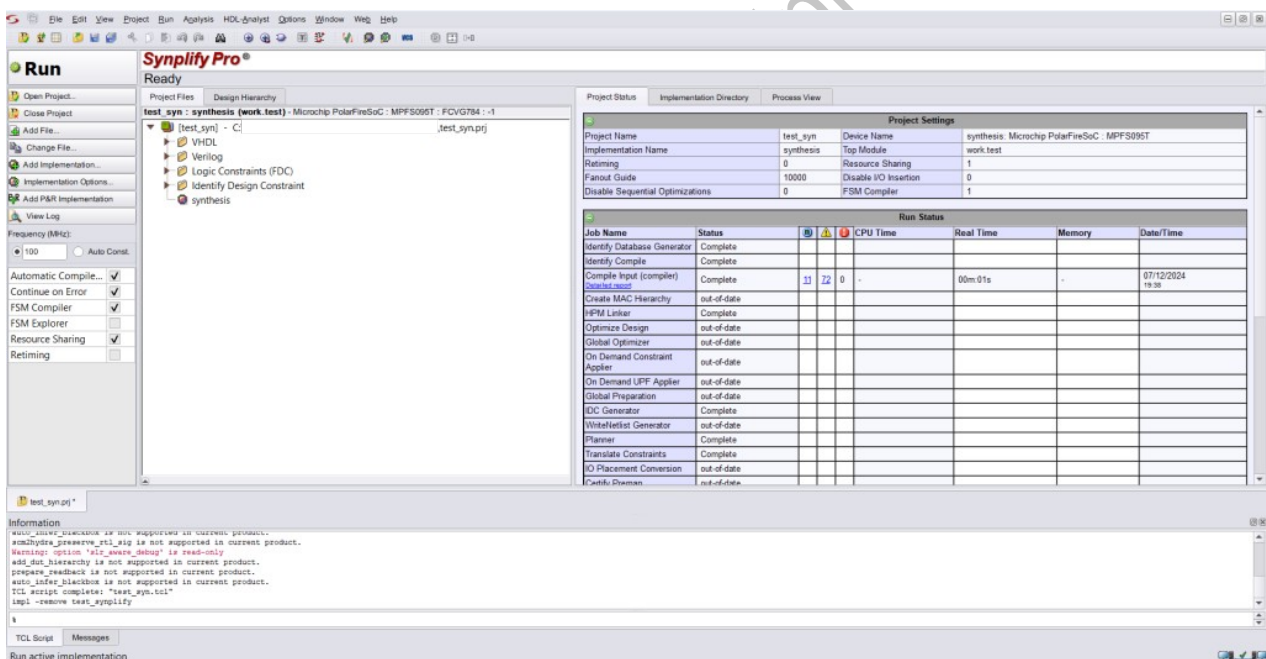
Once we have synthesized the project, we right-click on the **Synthesize** option and click on the *Open Interactively* option.



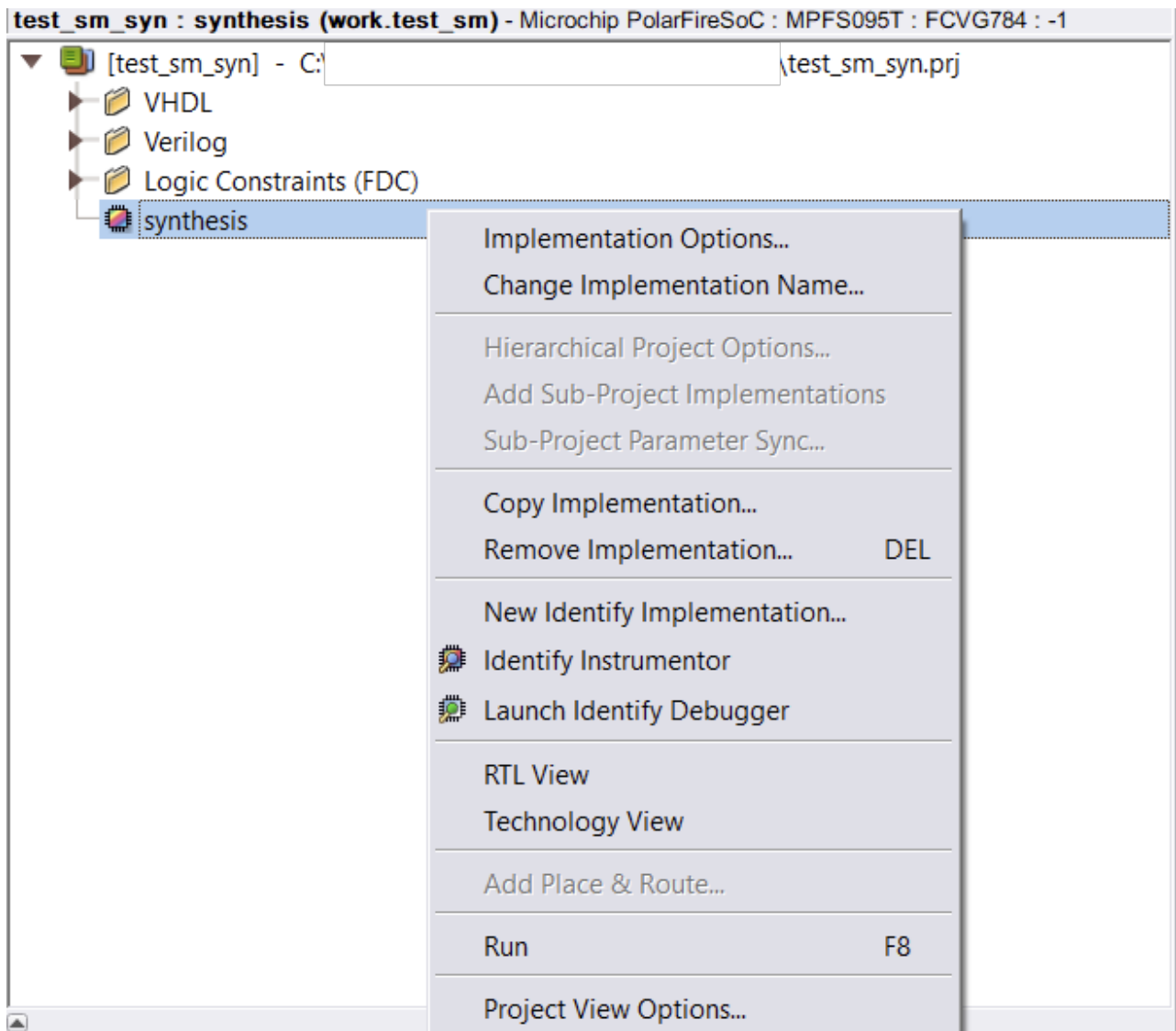
When we click on it, what it does is open **Synplify Pro**, and when it opens, it shows us this tab.



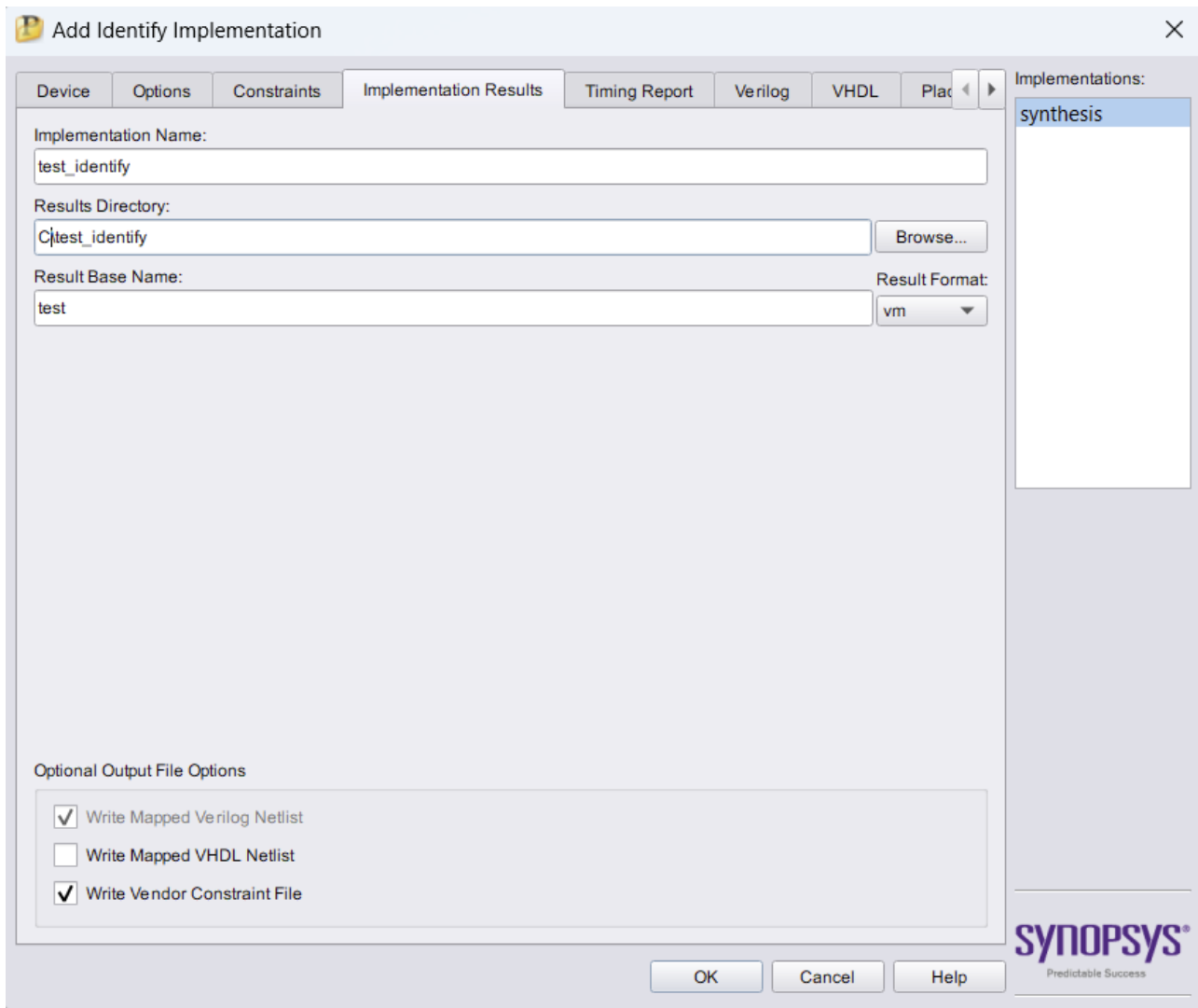
Then a tab opens where our project appears with the synthesis that Libero has done (called *synthesis*). Well, now what we have to do is create a new synthesis profile that allows us to add the logic analyzers.



To do this, right-click on the synthesis profile that comes from Libero, and create a new one in *New Identify Implementation*.



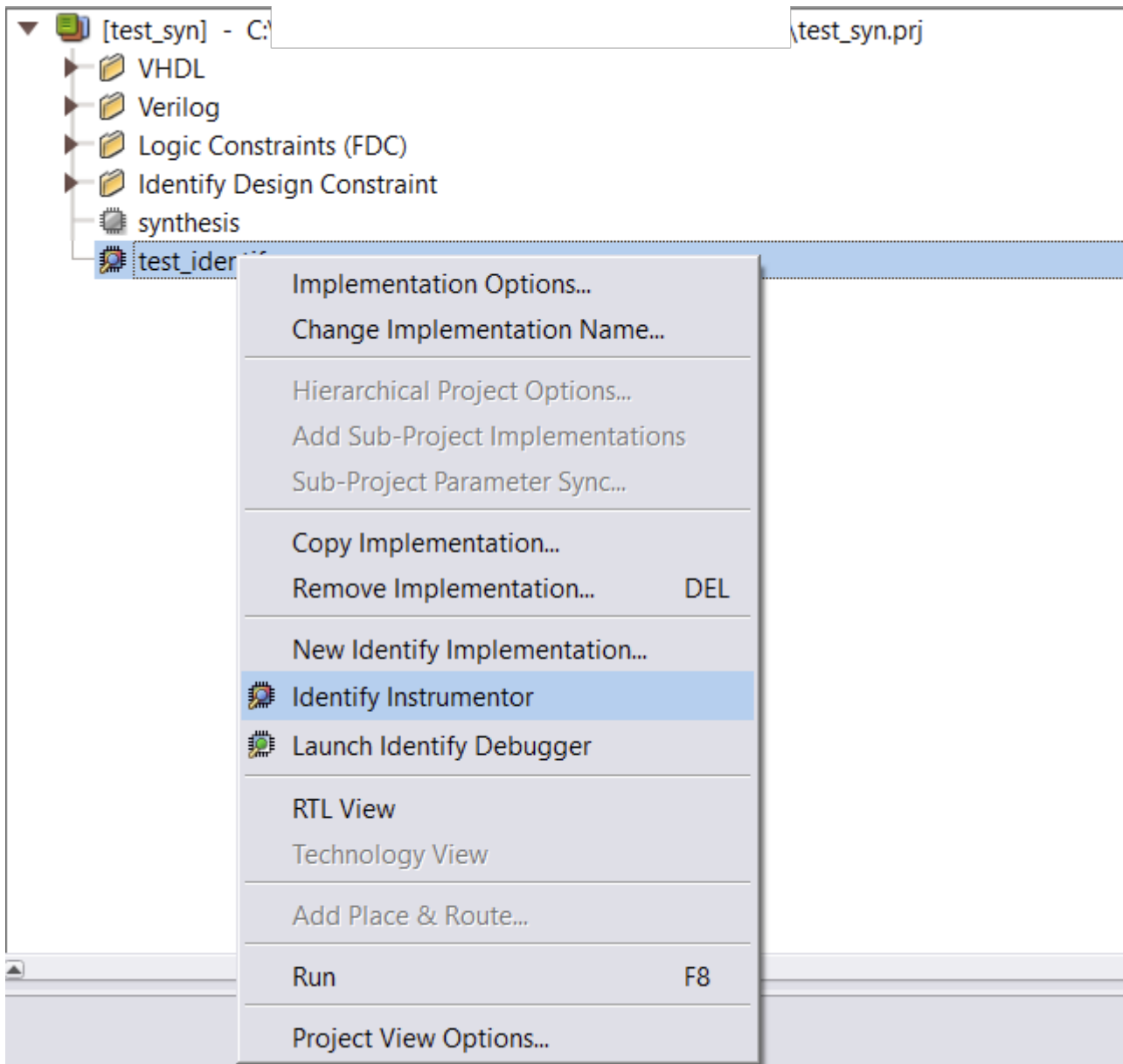
Then, a tab opens that asks us for the name of the new debugging profile and where it will be saved. The *Result Base Name* is the name of our top development file.



Once the new profile is created, we left-click on the profile, to indicate that we want to use that profile.

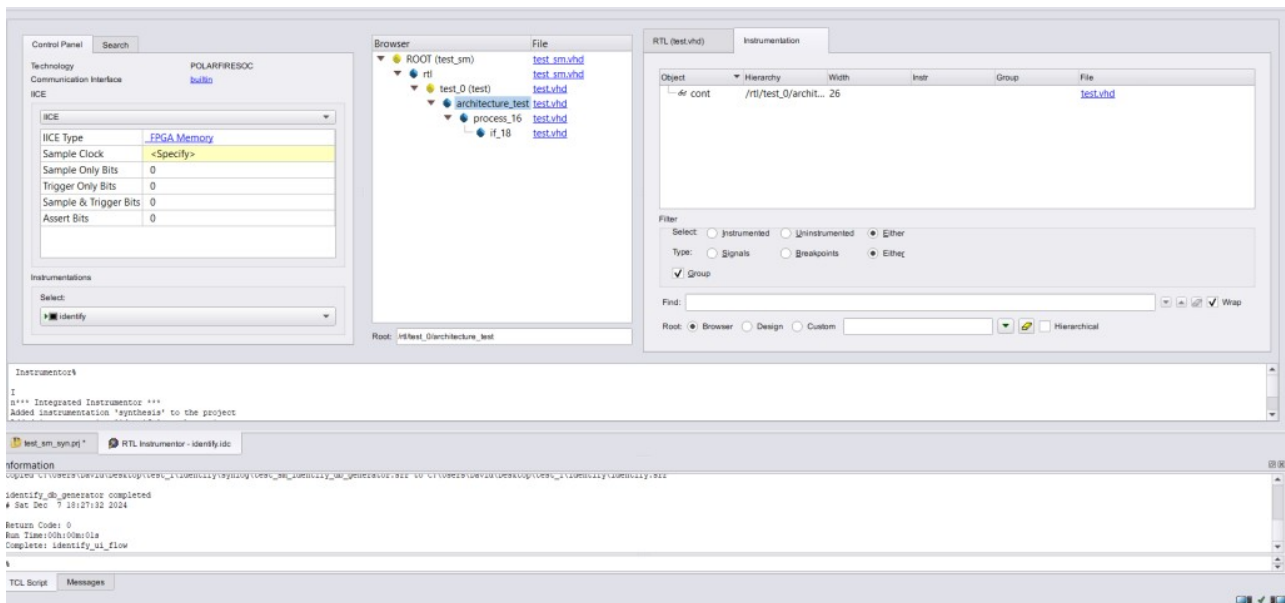


Click on *Identify Instrumentor*.



This opens a new tab within the program (so to return to the previous tab you have to go to the bottom).

This new tab is where you have to define the signals that you want to analyze.



Here, 3 sets of windows appear: the one on the **left** is the debug groups, which means that you can create different sets of signals to be analyzed in the same debug profile; the tab in the **center** is where the project structure appears so that the user can choose what to analyze; and the tab on the **right** is where you choose which signal or port you want to analyze. To do this, when selecting the file in the middle tab, the file appears here with icons for all the signals to be analyzed.

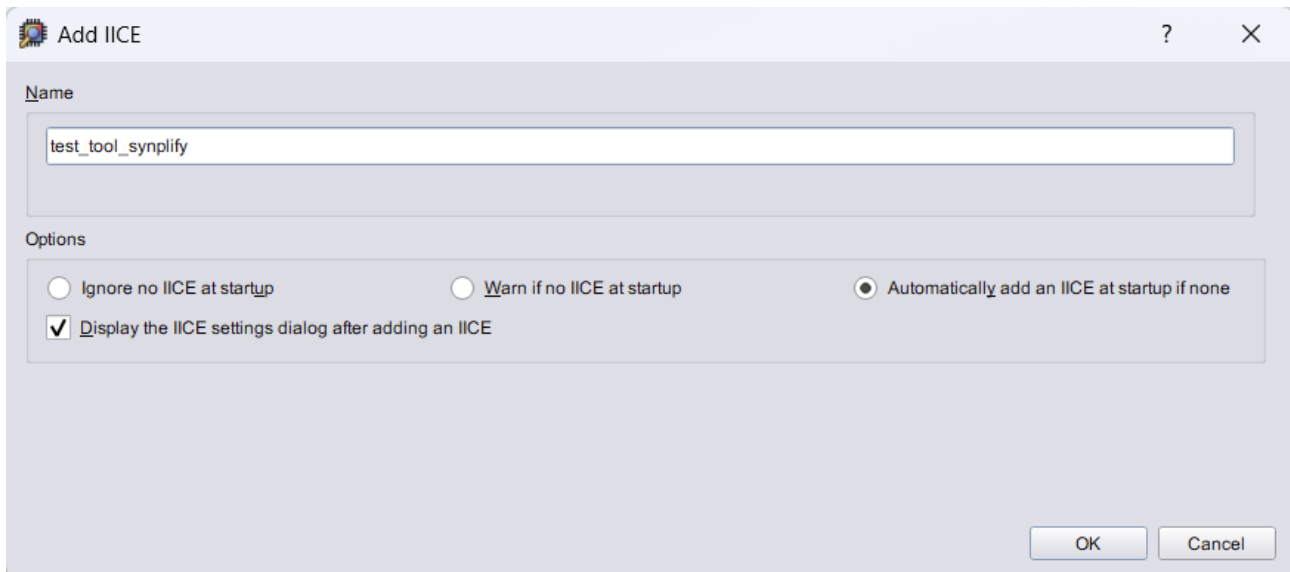
The first thing to do is to create a new signal analysis group (you can also use the default one, *IICE*).

**NOTE:** if you do not want to use the default one, it is recommended to delete it.

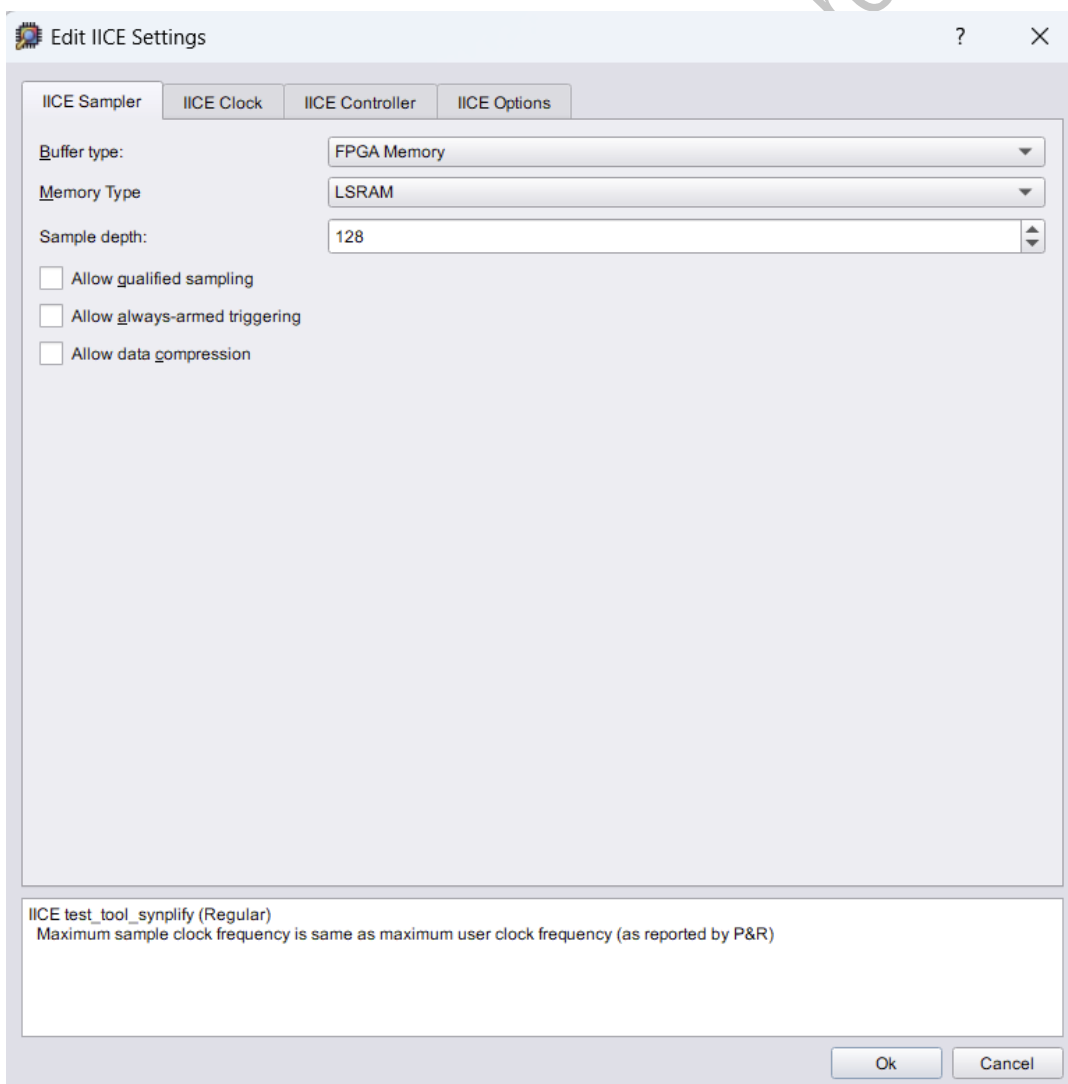


When you click on *Add IICE*, it asks you for the new set of signals.





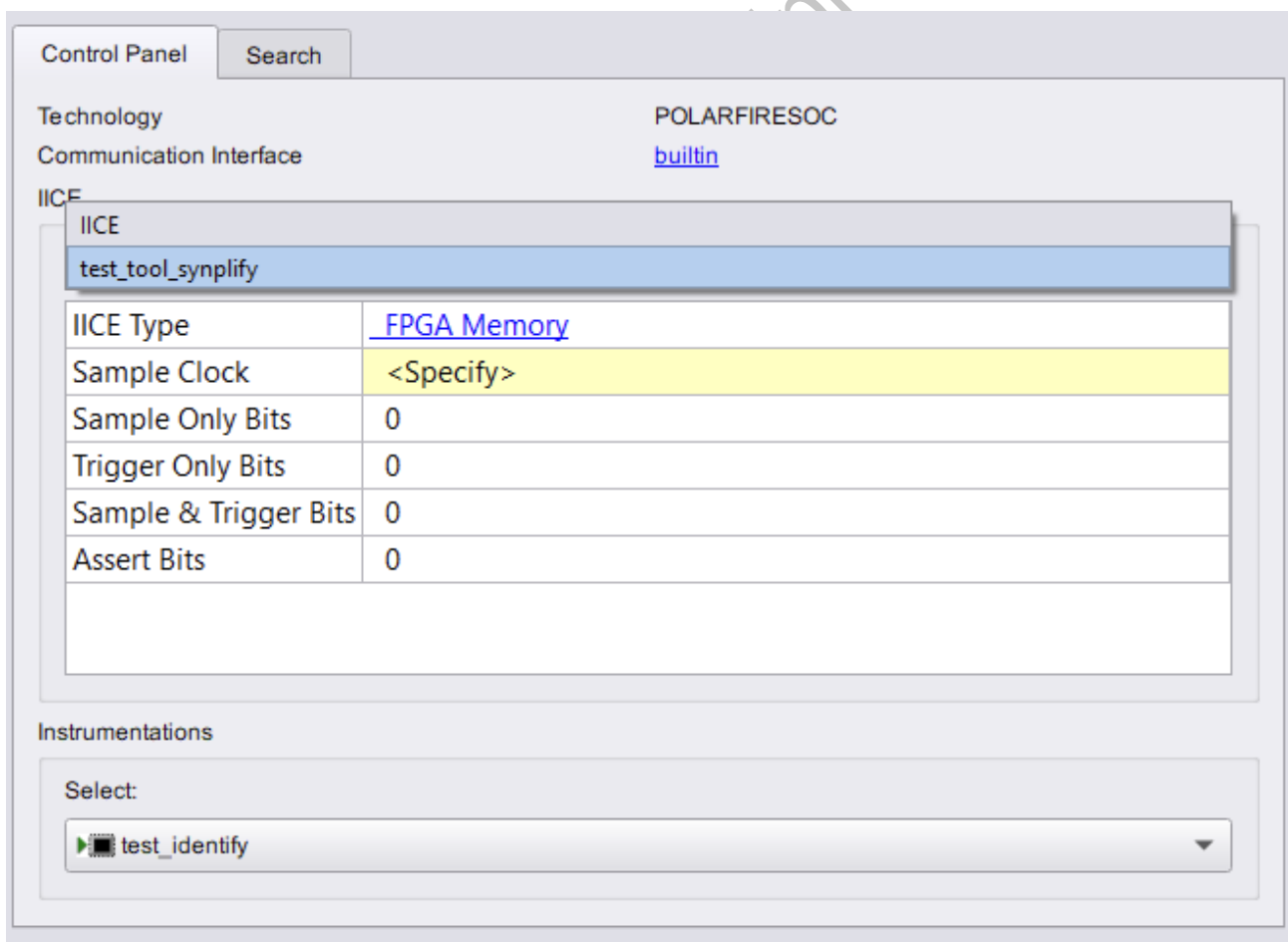
Then a tab opens where it asks you for the memory where the samples will be recorded, how many samples will be saved, the type of clock to be used.



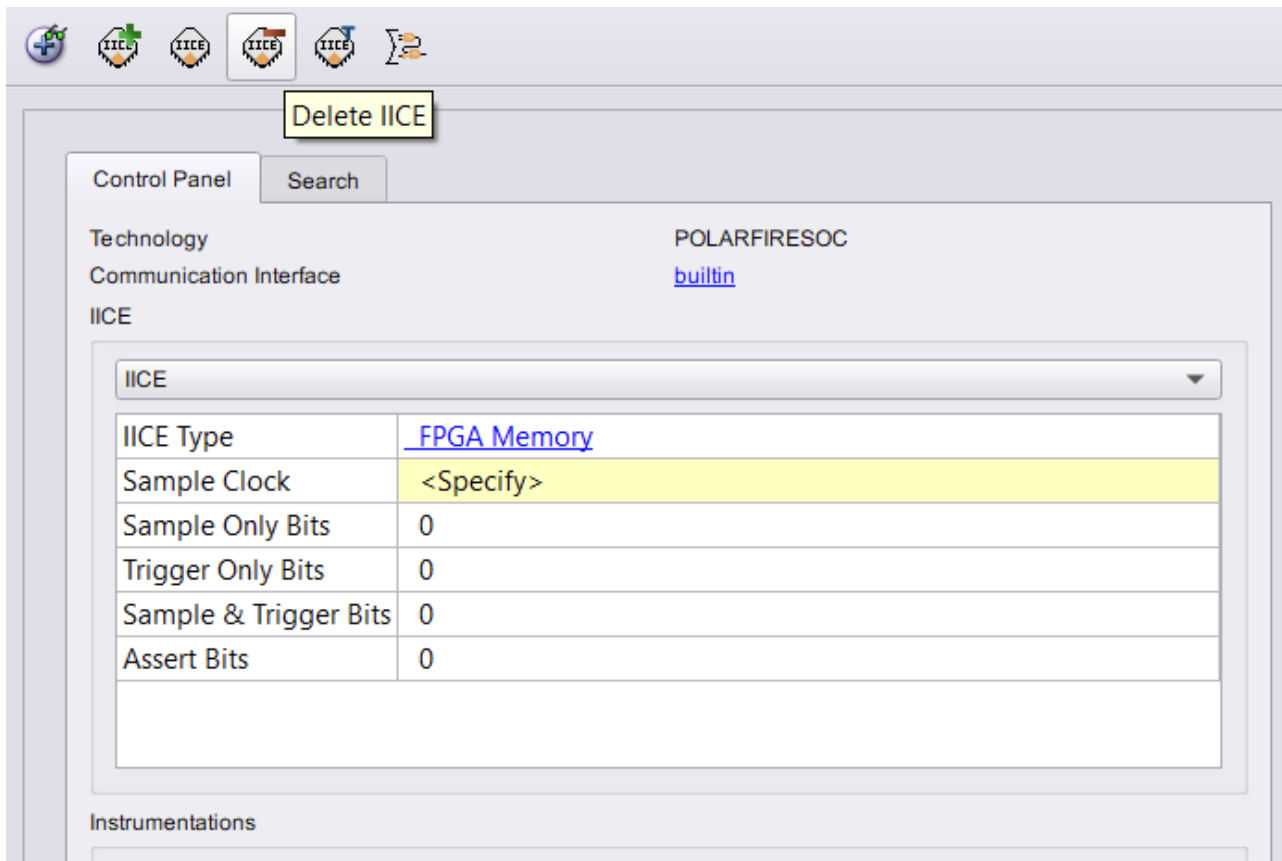
**NOTE:** it is possible that in *Memory Type*, if left as *AUTO*, an error message will appear.



On the left tab we can see the sample sets and the different debugging profiles (this is the *Instrumentations* option below)



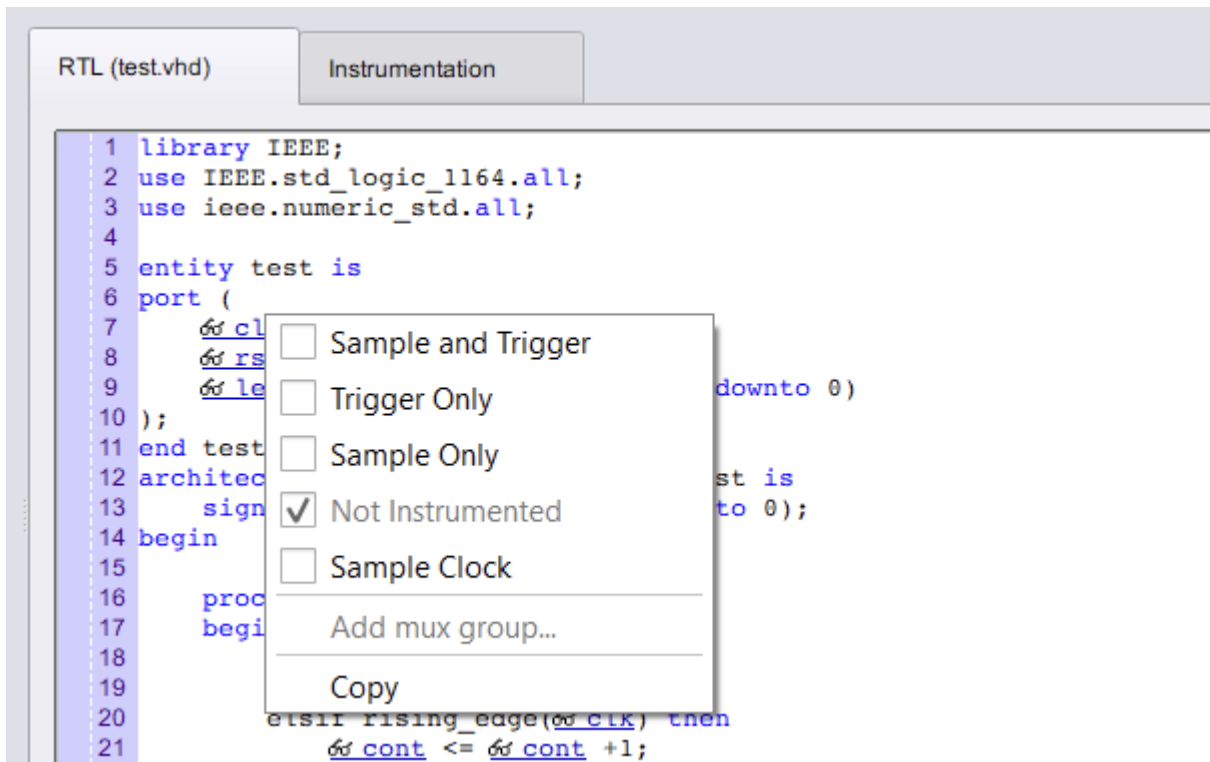
To delete, select the profile and click *Delete IICE*.



Once the container for the set of signals we want to create has been created, we move on to adding signals.

## Reference Clock

The first step for each set of signals in the debug profile is to select the reference clock for that group. To do this, go to the source code on the right (*after having selected the file in the center tab*) and click on the clock, and a drop-down menu will open, so all you have to do is select the *Sample Clock* option.



This will make the clock appear as a reference for the set to be debugged (in the left tab).

Control Panel

Search

Technology

POLARFIRESOC

Communication Interface

[builtin](#)

IICE

test\_tool\_synplify

IICE Type	<a href="#">FPGA Memory</a>
Sample Clock	/clk
Sample Only Bits	0
Trigger Only Bits	0
Sample & Trigger Bits	0
Assert Bits	0

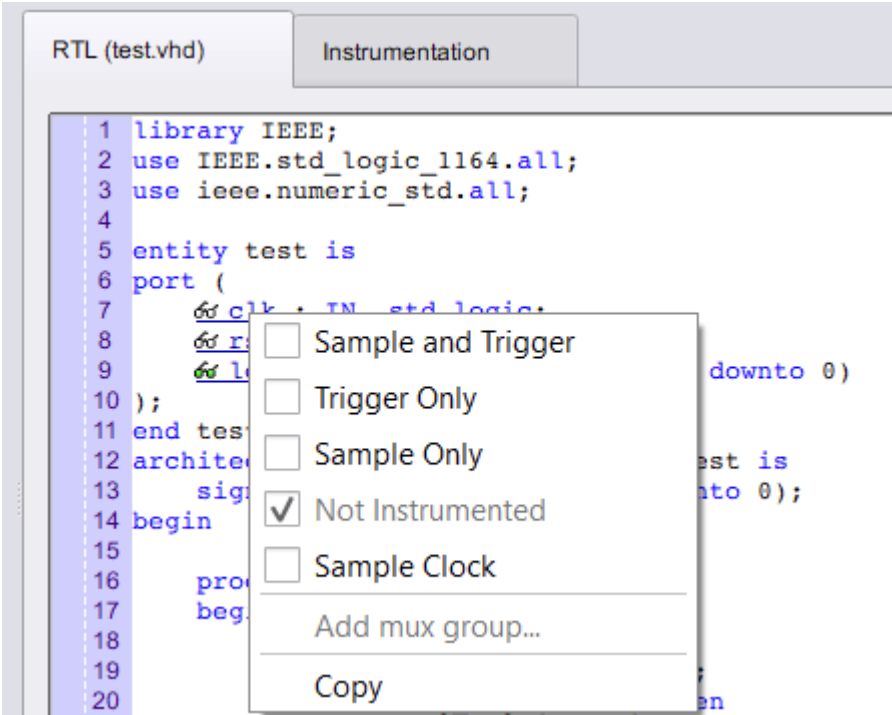
Instrumentations

Select:

▶ test\_identify

- **Selecting signals to be debugged**

The next step is to select the signals to be debugged. To do this, just like with the clock, we have 3 options: use the signal as *Trigger Only*, that is, as a signal that causes the samples to be read at that instant; as *Sample Only*, that is, as a signal that will be sampled when a change occurs in a signal with a trigger; and as *Sample and Trigger*, this is the combination of the other previous ones.



When you finish selecting signals, icons appear for the different signals that will be sampled.

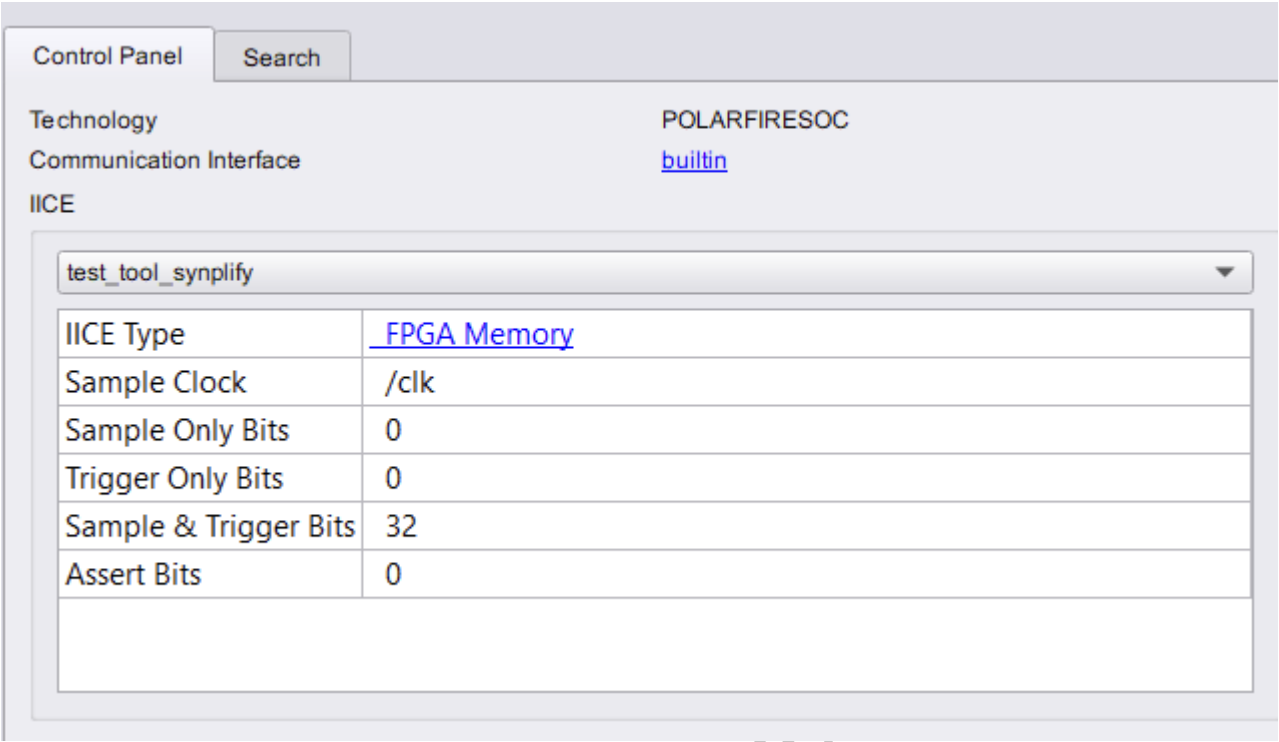
Signal Type	Icon
Sample Clock	Clock
Sample and trigger	Green glasses
Trigger Only	Red glasses
Sample Only	Blue glasses
<Nothing>	White glasses

```

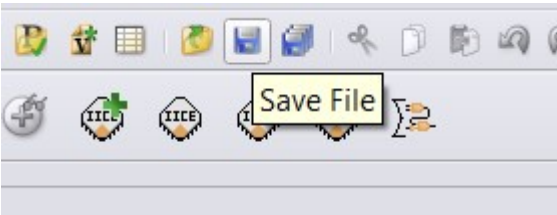
entity test is
port (
    clk : IN std_logic;
    rst_n : in std_logic;
    leds : OUT std_logic_vector(4 downto 0)
);

```

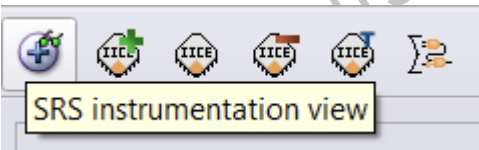
As the number of signals to be debugged increases, so does the *Sample & Trigger Bits* option.



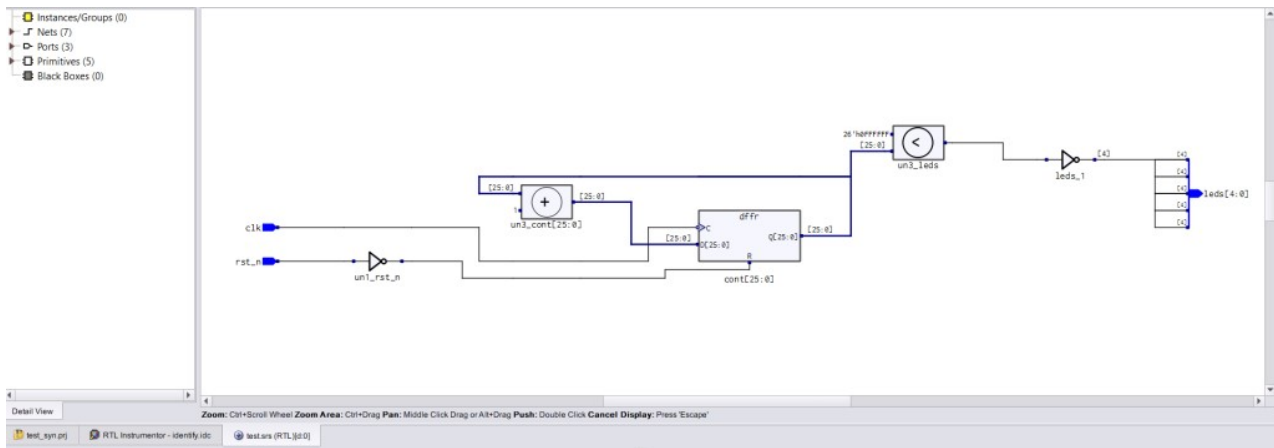
Once we have finished selecting the debug signals we have to save what we have done.



**NOTE:** The *SRS instrumentation view* option allows us to view the RTL model of the designed firmware.



The model would look like this.



The next step once we have created the set of all the signals we want to see is to return to the main tab

Project Name	test_syn	Device Name	test_identify	Microchip PolarFireSoC: MPFS095T
Implementation Name	test_identify	Top Module	work.test	
Retiming	0	Resource Sharing	1	
Fanout Guide	10000	Disable I/O Insertion	0	
Disable Sequential Optimizations	0	FSM Compiler	1	

Job Name	Status	CPU Time	Real Time	Memory	Date/Time
Compile Input	Complete				
Primap	out-of-date				
Map & Optimize	out-of-date				

Now we click on the **Run** option and this will analyze that everything we have done is correct.

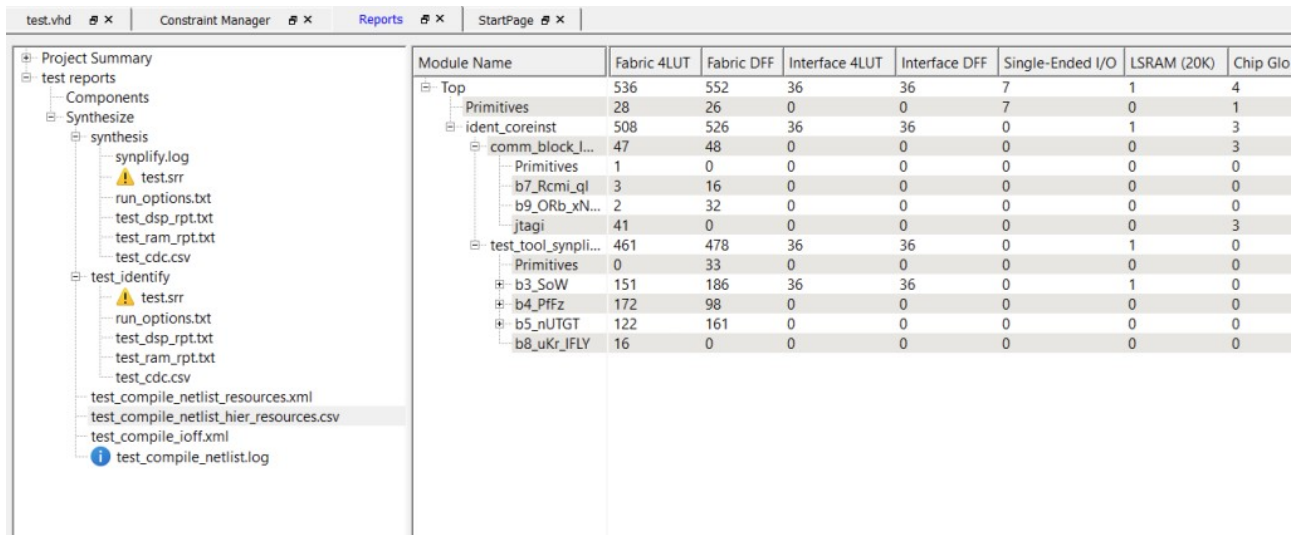
Job Name	Status	CPU Time	Real Time	Memory	Date/Time
Identify Database Generator (identify_db_generator)	Complete	2	0	0	08/12/2024 20:01
Identify Complete (identify_report)	Complete		00m:02s		08/12/2024 20:01
Compile Input (compiler) (compile_report)	Complete	15	22	0	08/12/2024 20:01
Primap (xmap) (primap_report)	out-of-date	33	5	0	08/12/2024 20:01
Map & Optimize (map_mapper) (map_report)	Complete	152	43	0	08/12/2024 20:01

Area Summary	
Carry Cells	42
DSP Blocks (dsp_used)	0
Global Clock Buffers	4
LUTs (total_luts)	508
Sequential Cells	552
I/O Cells	7
RAM1K20 (v_ram)	1



This will cause Libero to update the project behind the scenes but with our changes. Then, we can close *Synplicity Pro* and return to Libero.

If we now go to the synthesis report generated by Libero, to the option, *<project's name>\_compile\_netlist\_hier\_resources.csv*, a new field called *ident\_coreinst* appears, where all the consumption of the synthesis generated for debugging appears.



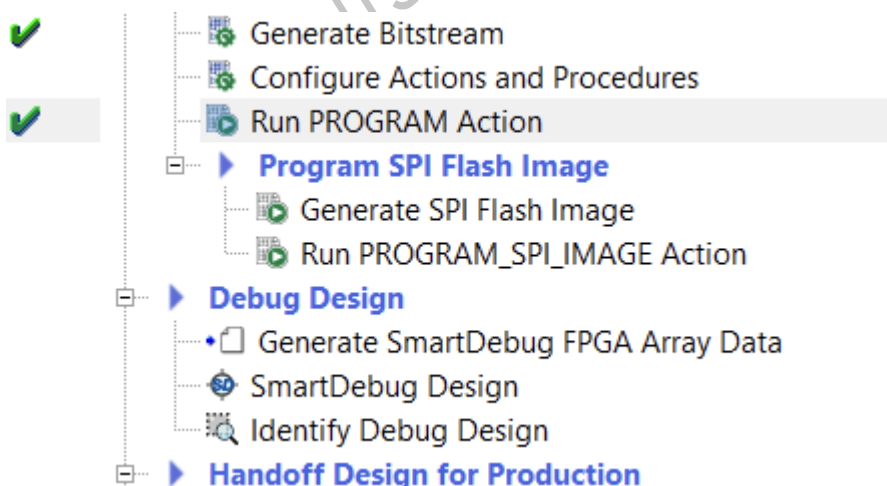
Module Name	Fabric 4LUT	Fabric DFF	Interface 4LUT	Interface DFF	Single-Ended I/O	LSRAM (20K)	Chip Glo
Top	536	552	36	36	7	1	4
Primitives	28	26	0	0	7	0	1
ident_coreinst	508	526	36	36	0	1	3
comm_block_I...	47	48	0	0	0	0	3
Primitives	1	0	0	0	0	0	0
b7_Rcmi_ql	3	16	0	0	0	0	0
b9_ORb_xN...	2	32	0	0	0	0	0
jtagi	41	0	0	0	0	0	3
test_tool_synpli...	461	478	36	36	0	1	0
Primitives	0	33	0	0	0	0	0
b3_SoW	151	186	36	36	0	1	0
b4_PfFz	172	98	0	0	0	0	0
b5_nUTGT	122	161	0	0	0	0	0
b8_uKr_IFLY	16	0	0	0	0	0	0

With all this, the next step is to generate a bitstream and upload it to the FPGA/SoC. Remember to generate a pin file to assign the ports to.

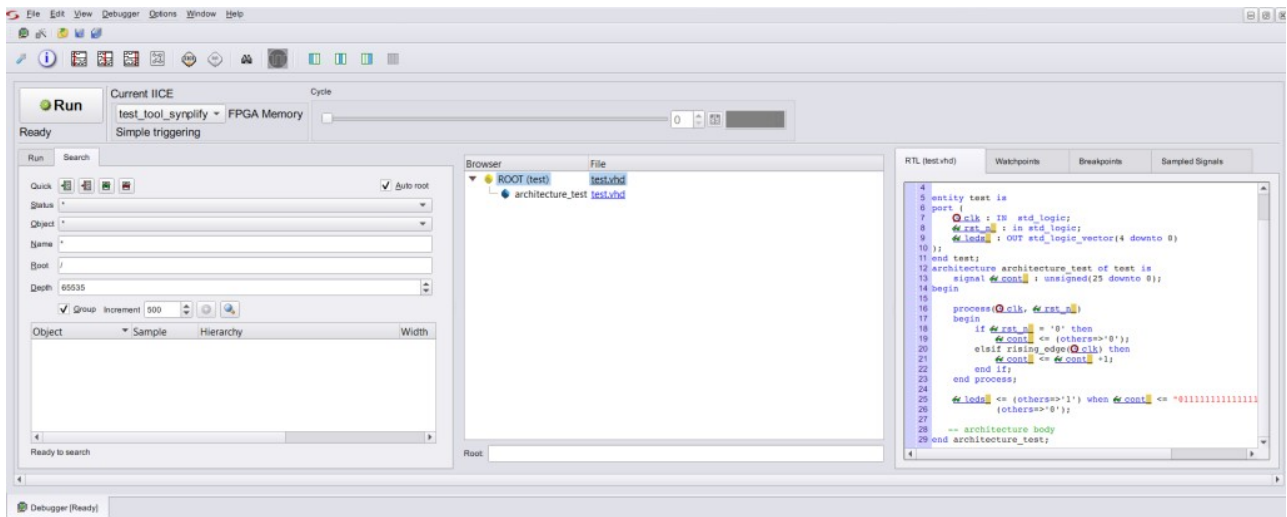
## Identify Debugger

With all the previous steps generated and a bitstream with the debugging configuration loaded, we move on to the *Identify Debugger*.

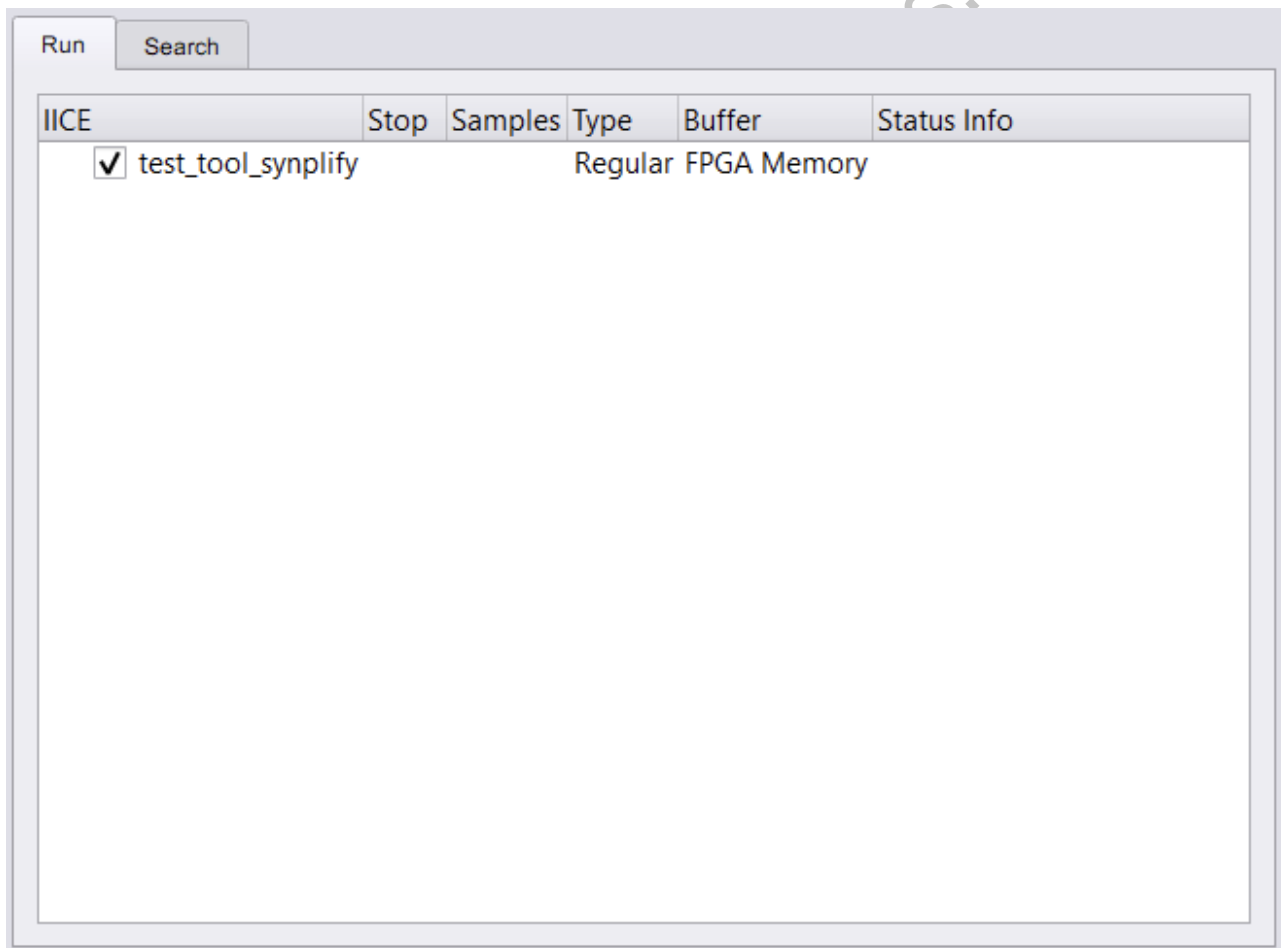
To do this we have to go to the debugging tools and select the one that says *Identify Debug Design*.



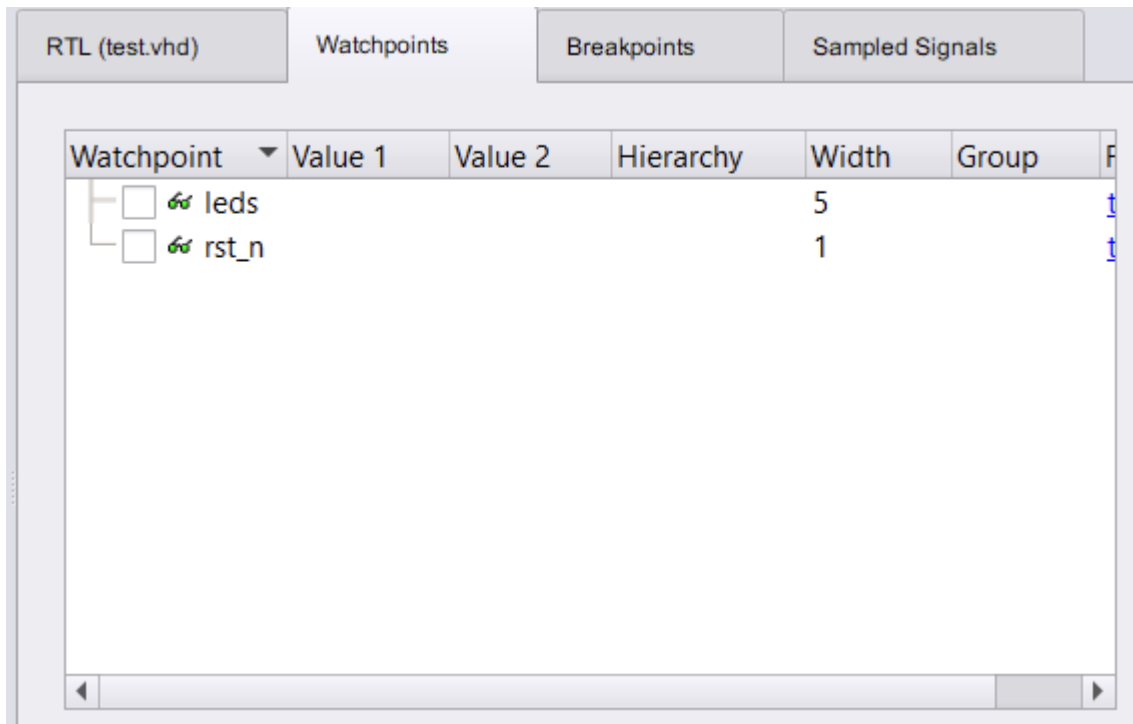
Now the *Identify Debugger* opens. This tool allows us to view both the signals we have to debug and the way to view them.



On the left we have the different sets of signals that we can view when launching the debugger.

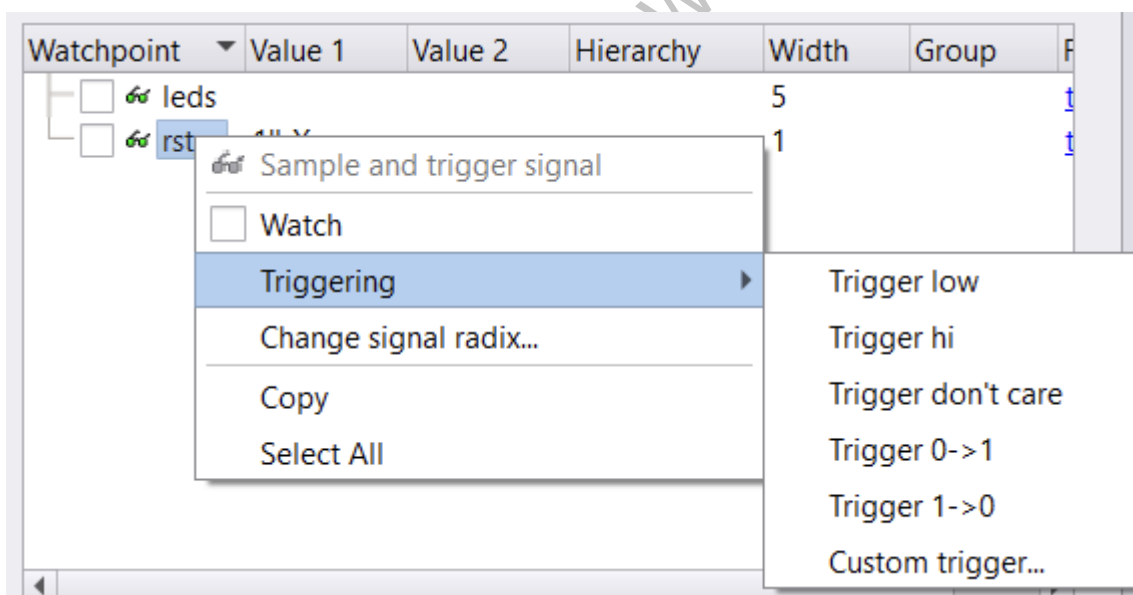


To start, we have to choose which condition the debugger will jump under. To do this, in the tab on the right, in the *Watchpoint* option, we have to choose which is the trigger and under what condition it is triggered.

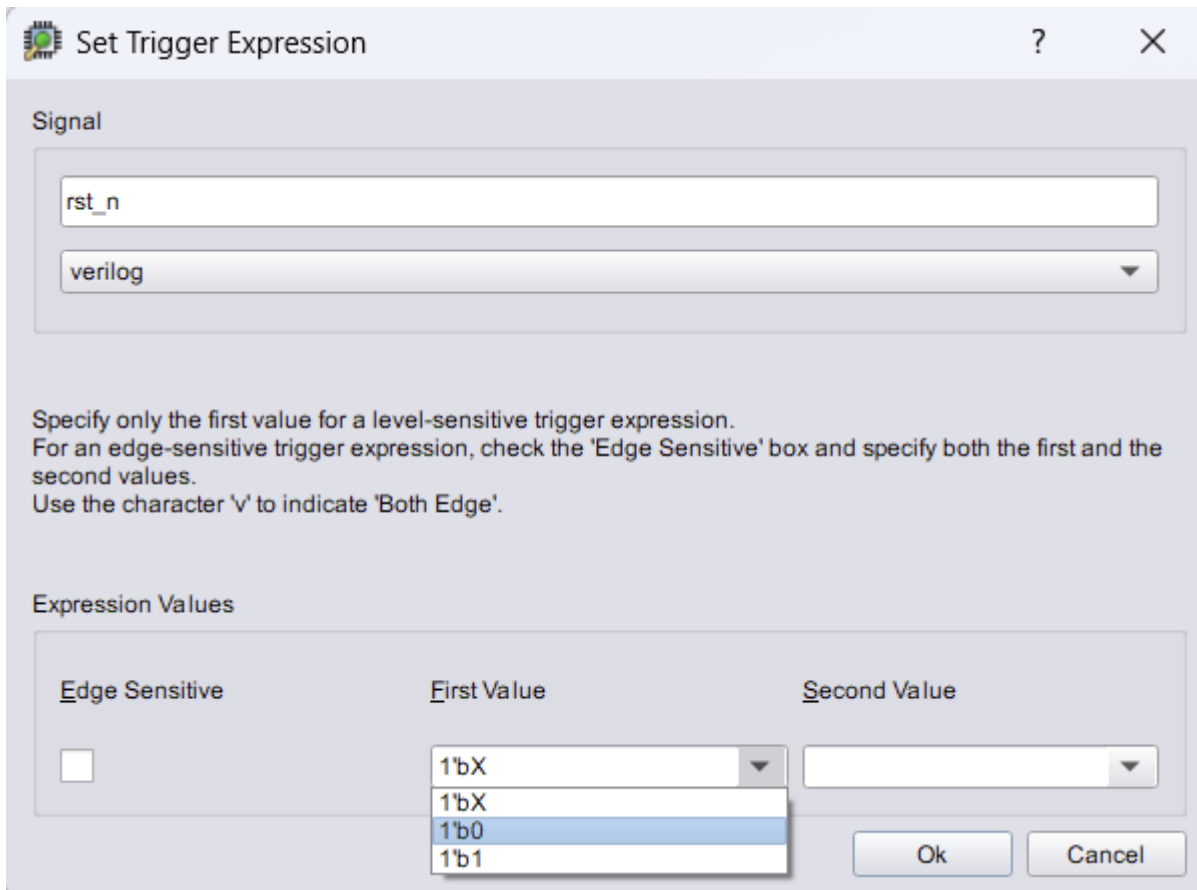


There are two ways to select the trigger form:

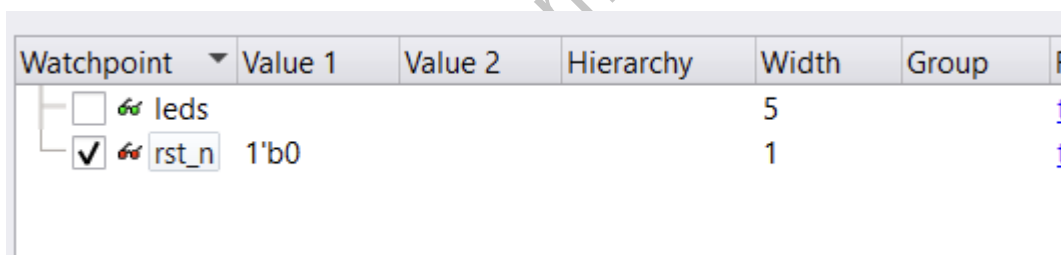
- Right click on the signal and on triggering, and select one of the options.



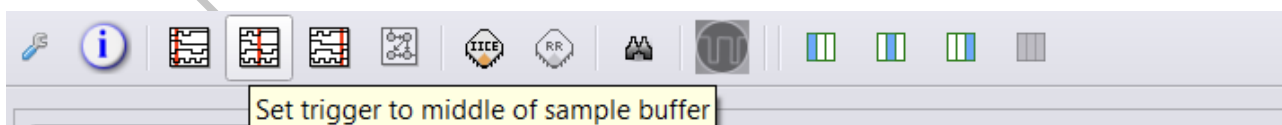
- Double click on the signal, this option is the same as if *Custom trigger* had been chosen in the previous option.  
In the row with the language, it is simply the format in which the trigger condition will be represented.



When you finish the selection, you can see the selected condition.

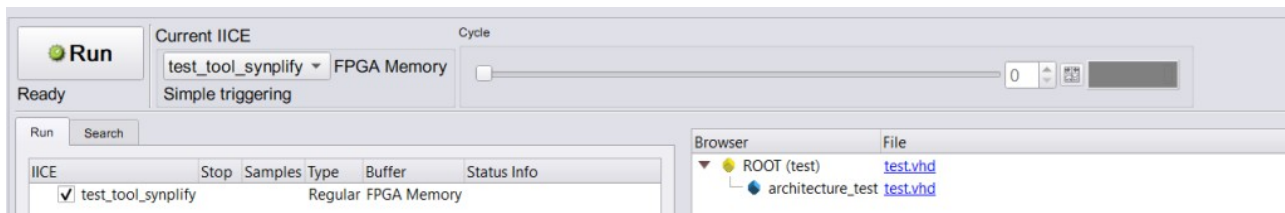


Before launching the debugging, there are several options for the representation of the sampled data, and that is to say that they are aligned to the left, to the right or to the center.



Once everything is selected, you press *Run*.

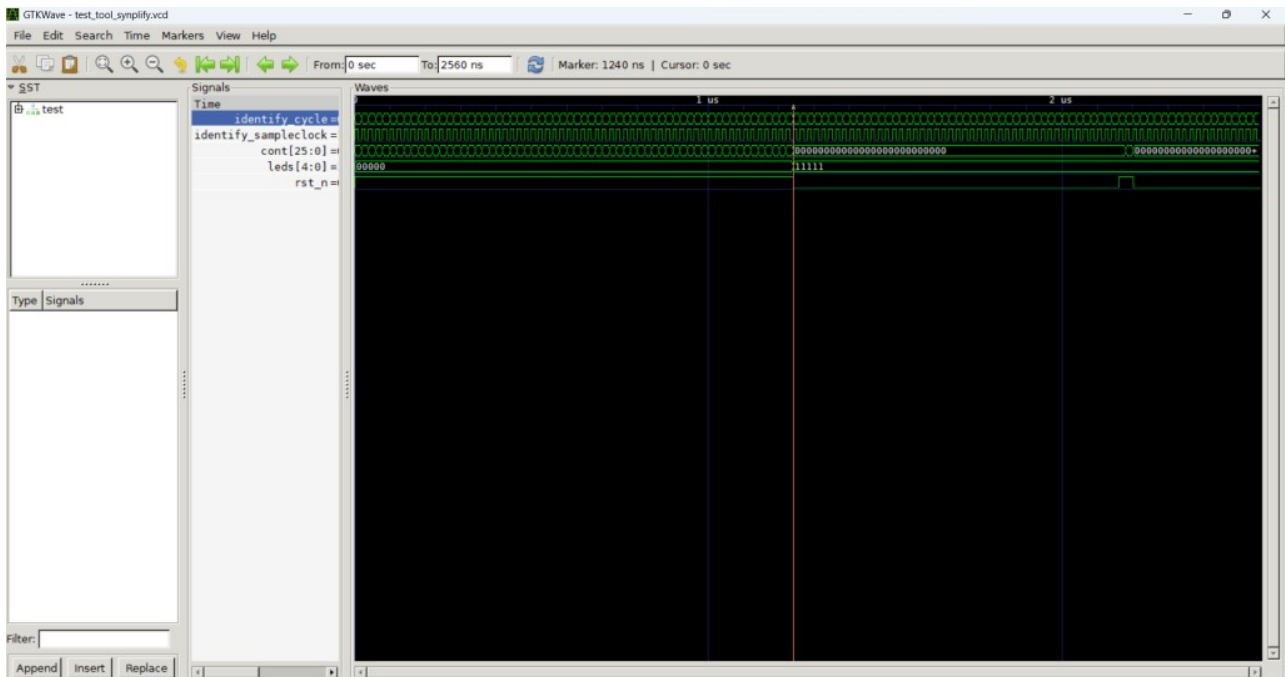
**NOTE:** as many tabs will open as sets of signals have been chosen.



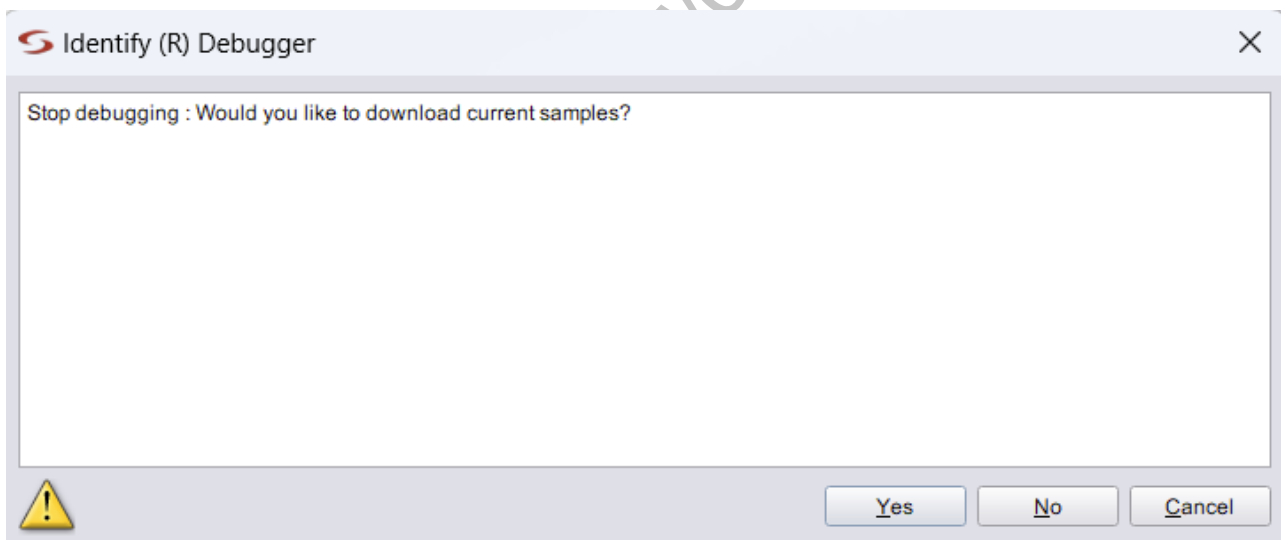
If everything has been done correctly, a tab will open in GTKWave where the signals chosen for debugging are represented.



If you look at the first row, it is the number of samples that have been selected in Synplify, but divided in half, so that you have -52 on the left and 65 on the right.



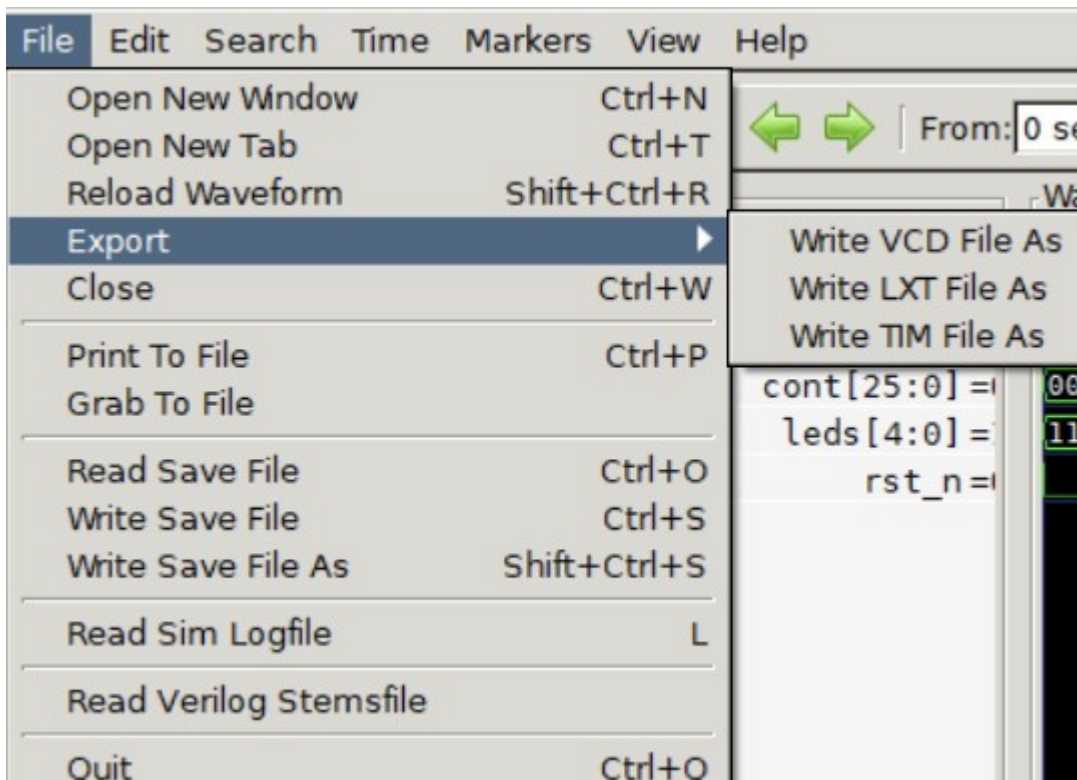
**NOTE:** If we decide to stop debugging, the debugger asks us if we want to see the last samples it has saved (*even if the trigger condition has not been met*). This opens a new GTKWave tab with the samples.



**NOTE:** as many tabs will be opened as times the debugging is launched.

The signals that are debugged can be saved for later analysis.





Finally, note that this bar is used to select which sample to place the trigger on to view the signals.



## References

<https://youtu.be/liV25OIBLrY>