Debugging methods in Libero

Created by: David Rubio G.

Blog post: https://soceame.wordpress.com/2025/03/10/debugging-methods-in-libero/

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

GitHub: https://github.com/DRubioG

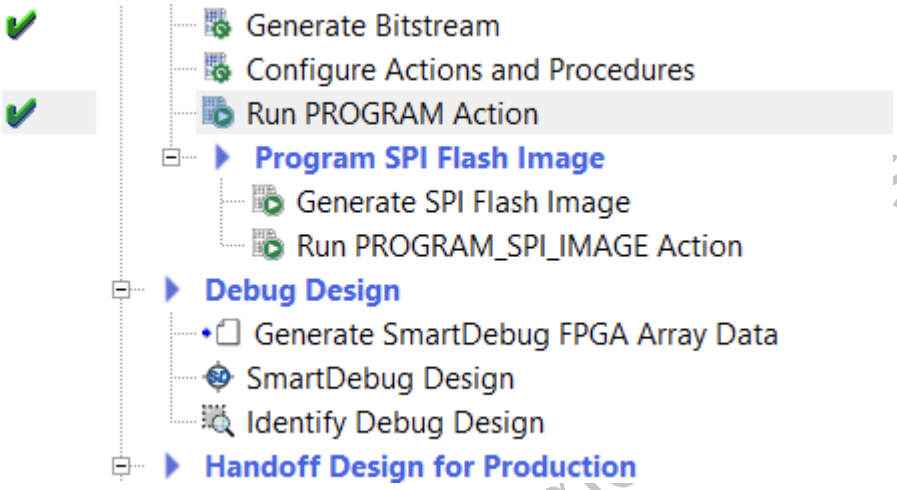
Last modification date: 10/03/25

In this post I will discuss the different ways to debug a Microchip FPGA/SoC using Libero. Some of these ways are quite interesting.

The difference between debugging a Microchip FPGA and a Xilinx FPGA is that Xilinx requires you to tell it which signal you want to debug, while Microchip allows you to debug any internal signal of the FPGA, without having to define which one you want.

Accessing debugging

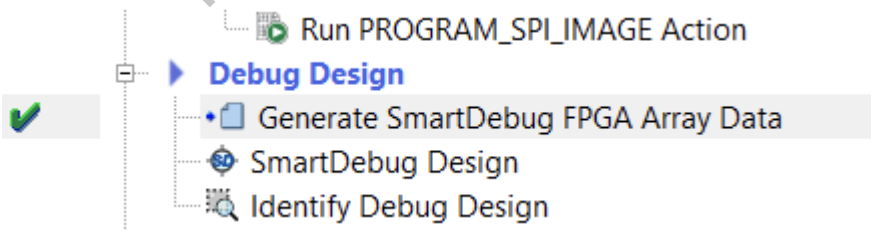
To access debugging with Libero we have to use the *Debug Design* options.



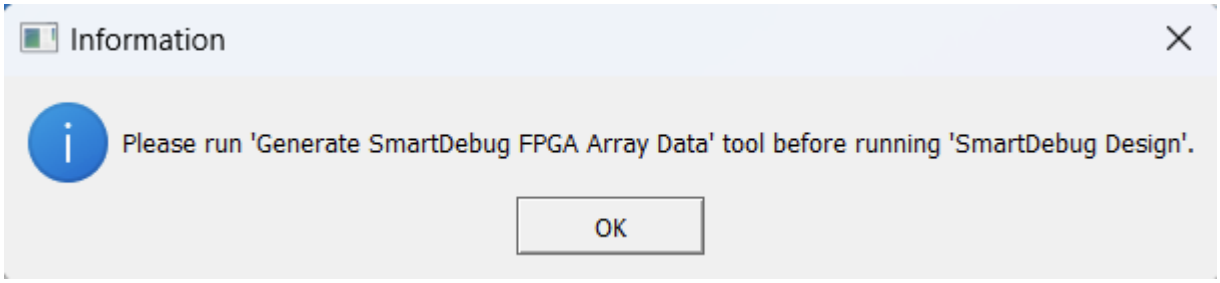
Three options appear here:

- *Generate SmartDebug FPGA Array Data*: this option is the initialization of the Libero debugging system.
- *SmartDebug Design*: this option is the one that takes us to the debugging tools.
- *Identify Debug Design*: this option opens the *Synopsys Identify* program for debugging. This will be explained exclusively in another post.

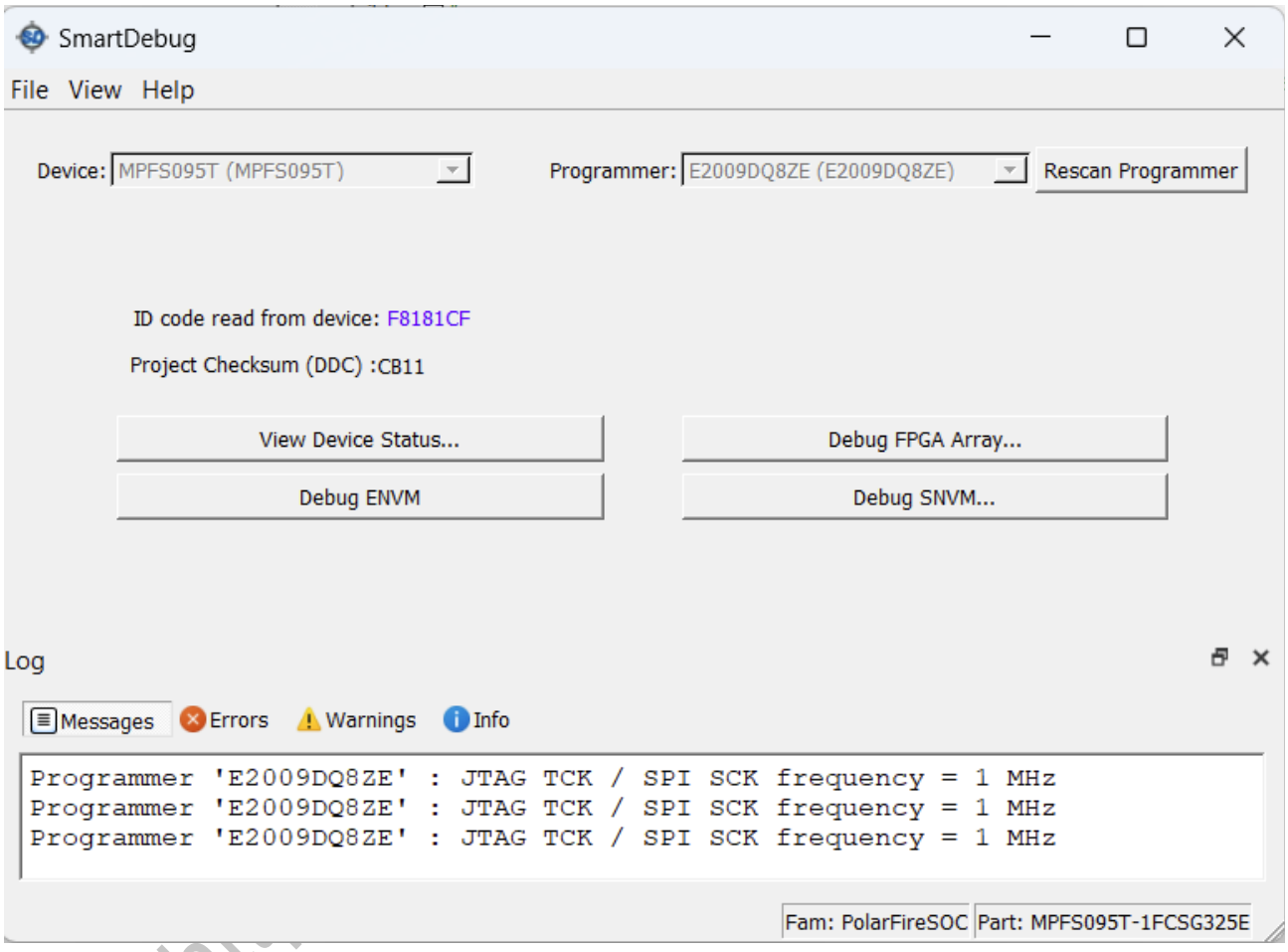
To start, we first have a bitstream recorded in the FPGA/SoC and then we have to click on *Generate SmartDebug FPGA Array*, which generates the debug profile.



NOTE: if we don't click on it first, this tab will pop up



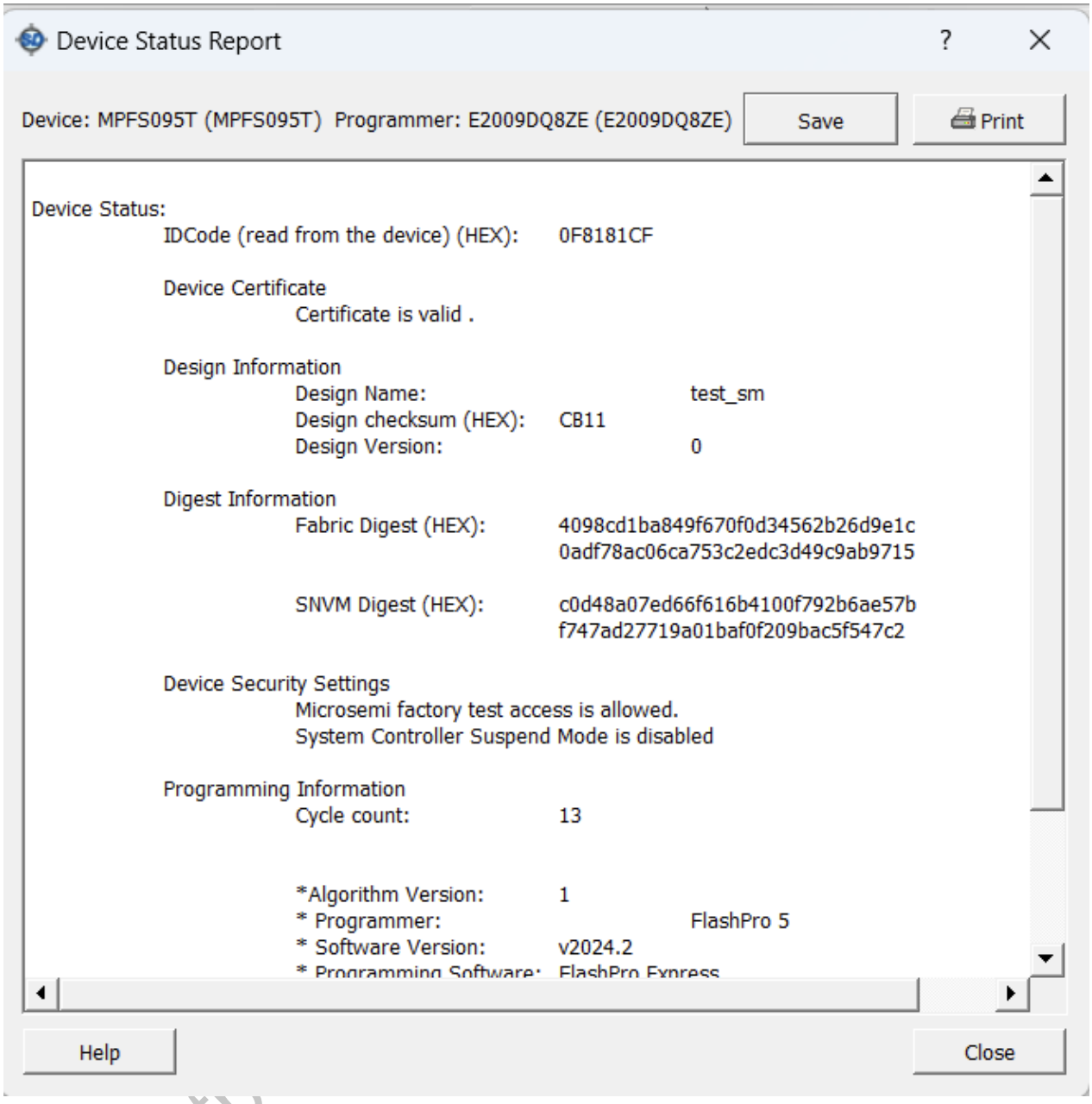
Then we can click on the *SmartDebug Design* option, which will open a tab for debugging.



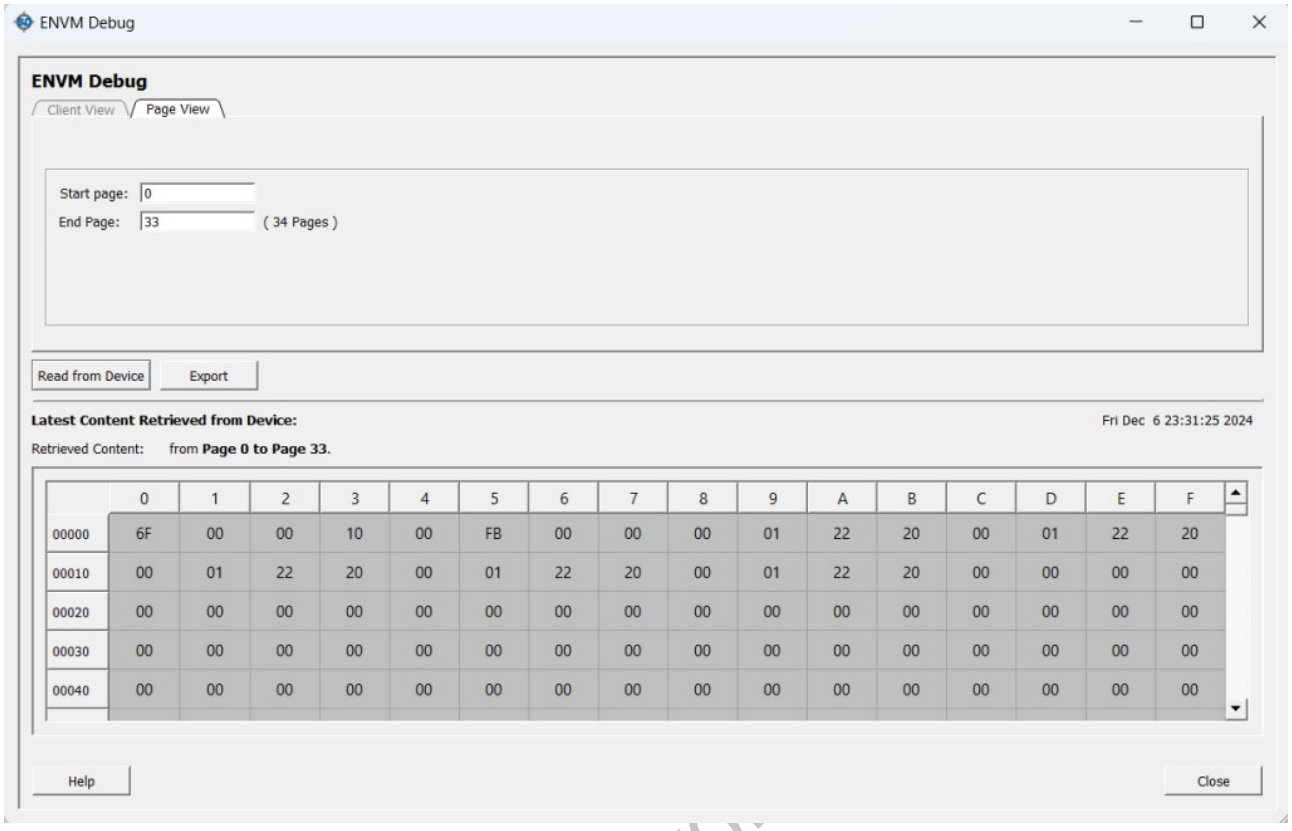
Debugging tools

The tool that opens in the previous step has different tools inside.

- **View Device Status**: this tool gives us information about the device we are going to debug.

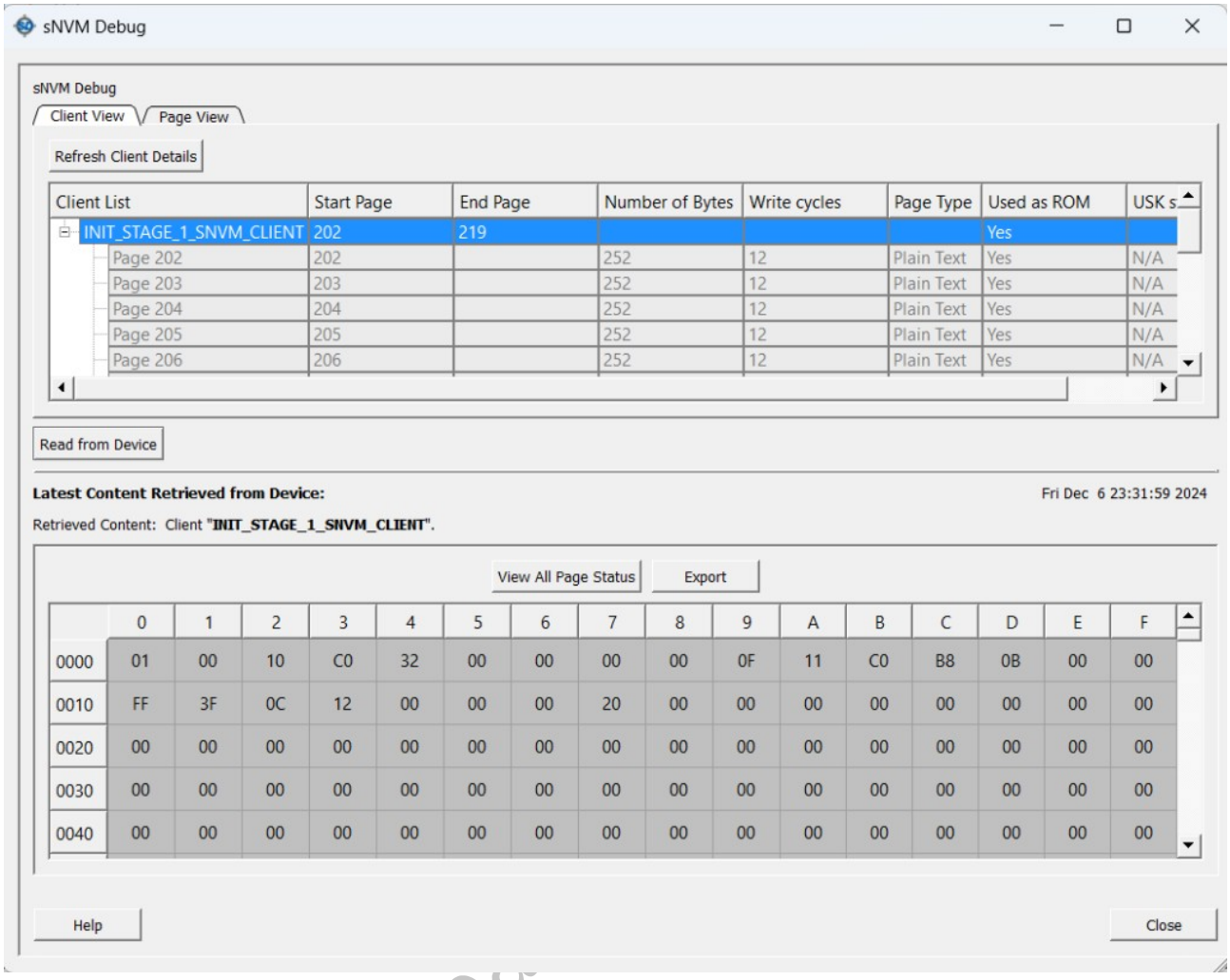


- **Debug ENVM**: this tool allows us to debug the internal non-volatile memory of Microchip FPGAs/SoCs. We only need to indicate a start memory address and an end memory address. It also has a button that allows us to read the memory at the exact moment of pressing it *(Read from Device)*.

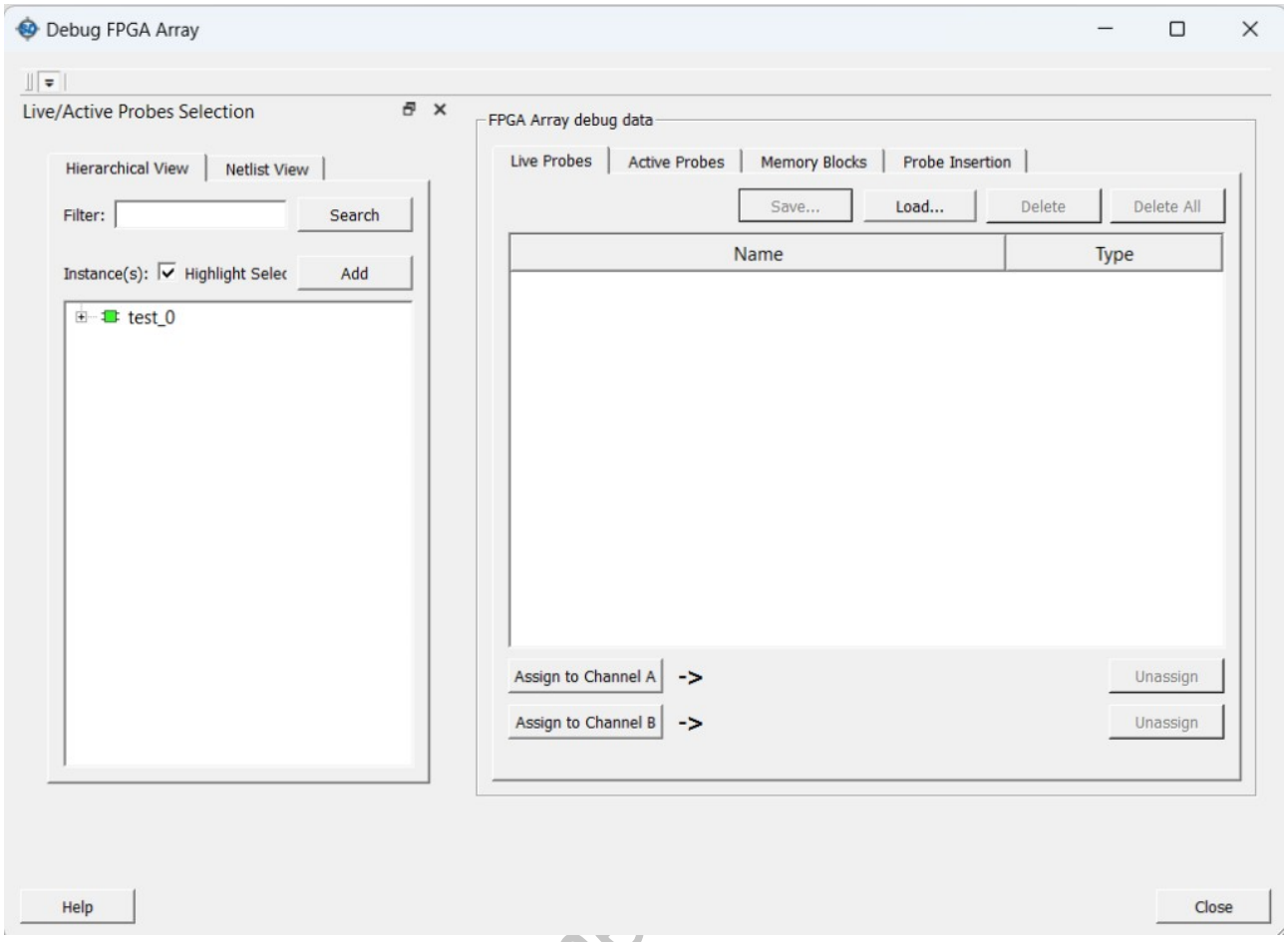


	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
00000	6F	00	00	10	00	FB	00	00	00	01	22	20	00	01	22	20
00010	00	01	22	20	00	01	22	20	00	01	22	20	00	00	00	00
00020	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
00030	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
00040	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00

- **Debug SNVM**: this tool allows us to debug the SRAM memory of the SoCs, provided there is a memory profile recorded in it.



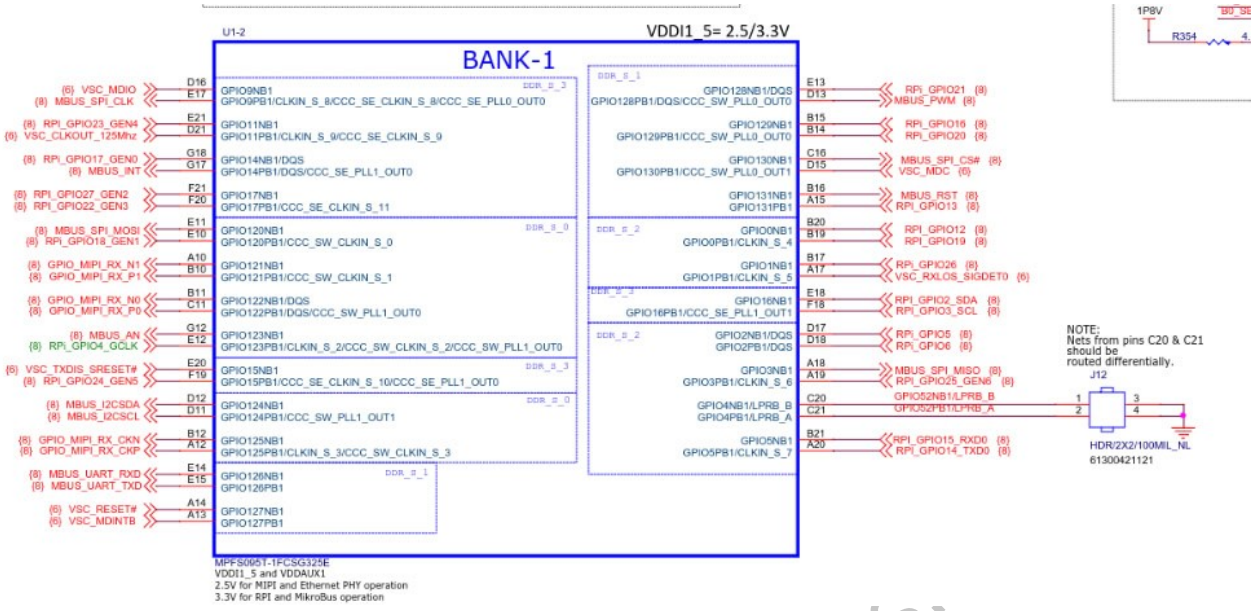
- **Debug FPGA Array**: this tool is the one that allows us to debug the FPGAs with different debugging options.



Debug FPGA Array

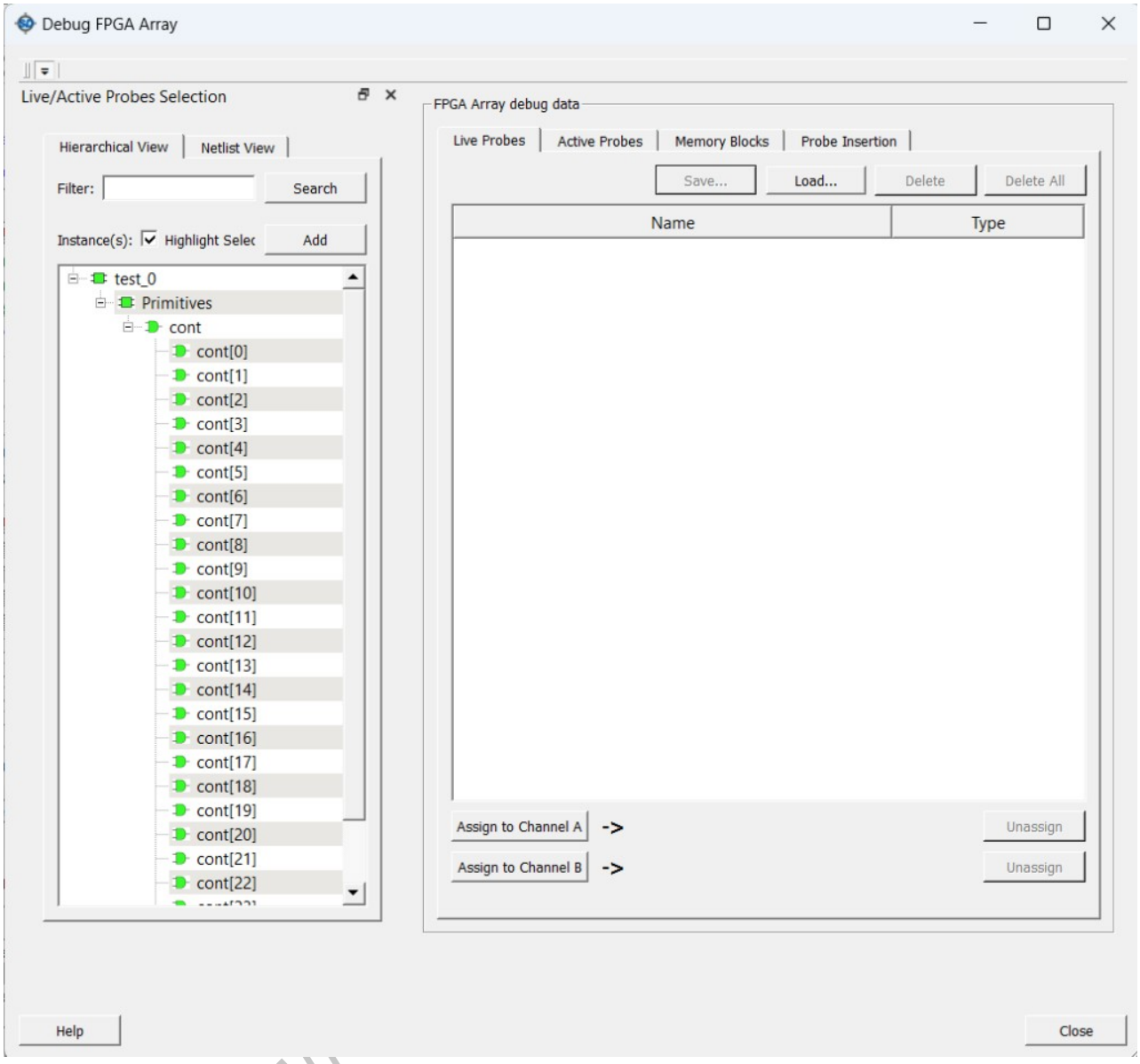
This tool has very interesting options to debug an FPGA.

- **Live Probes**: this option allows one of the internal signals of the FPGA (**LPRB_A** and **LPRB_B**) to be output through one of the two specific debugging pins that the FPGA has.

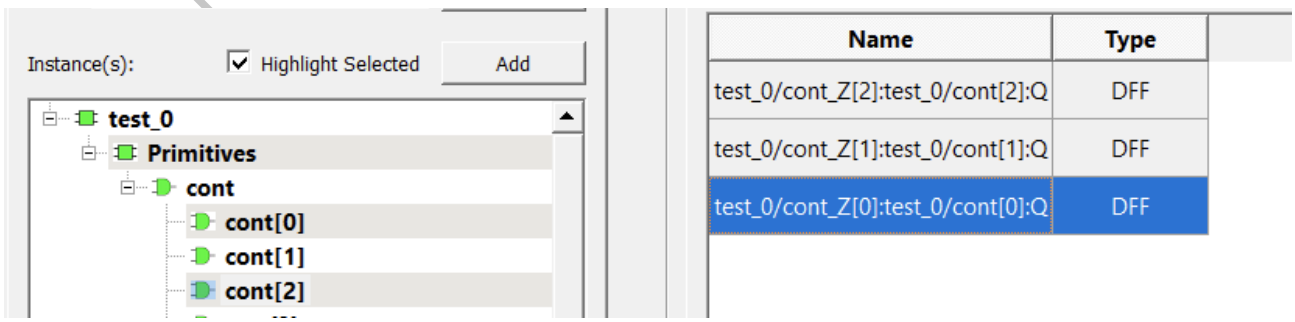


So if I want to know what the status of an internal signal is in real time, I can output that signal through a pin and view it on an oscilloscope. This tool is more focused on signals that vary in time *(for discrete signals there is another tool)*

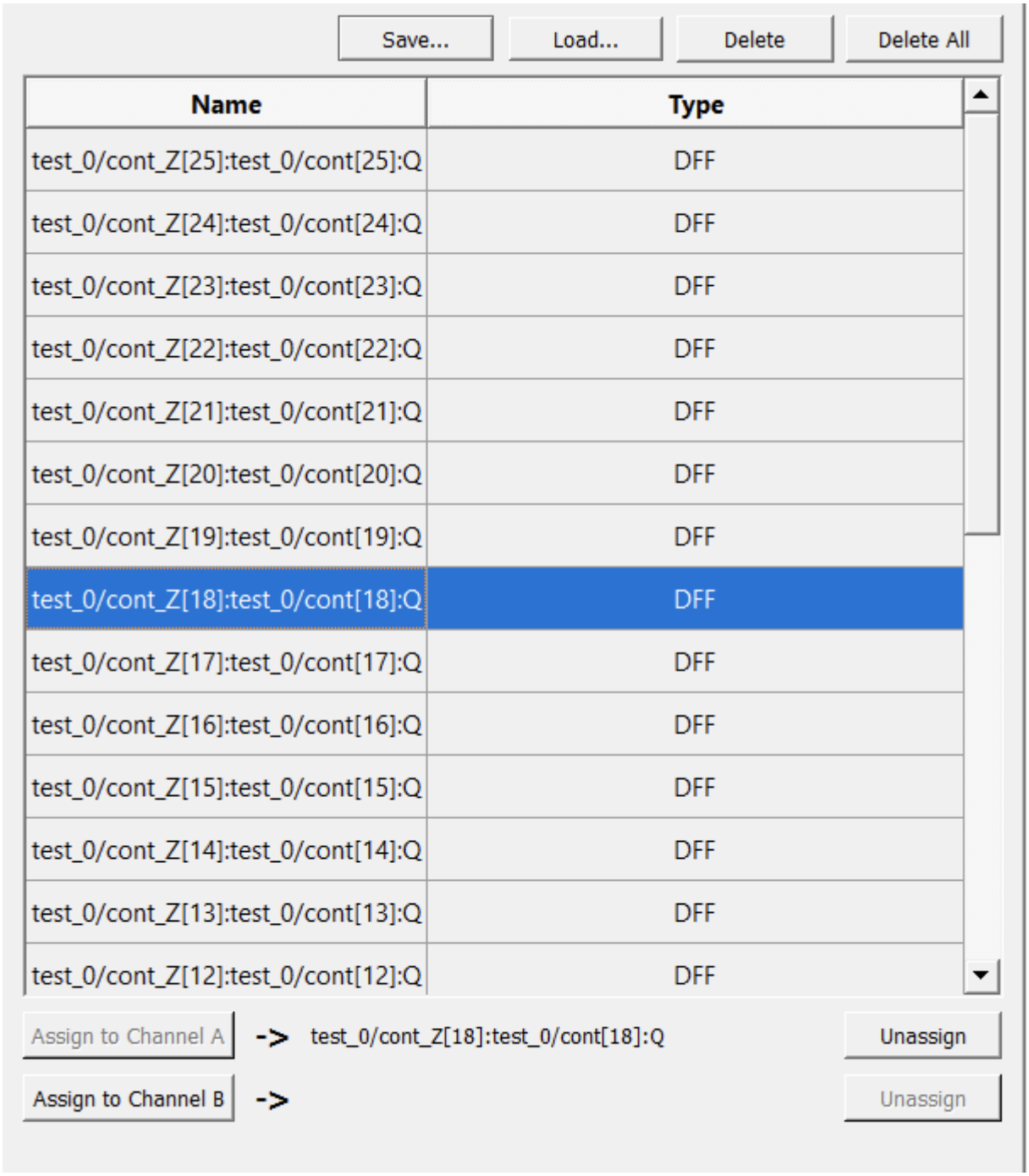
NOTE: this tool is automatic, so when selecting a signal it automatically outputs through the pins.



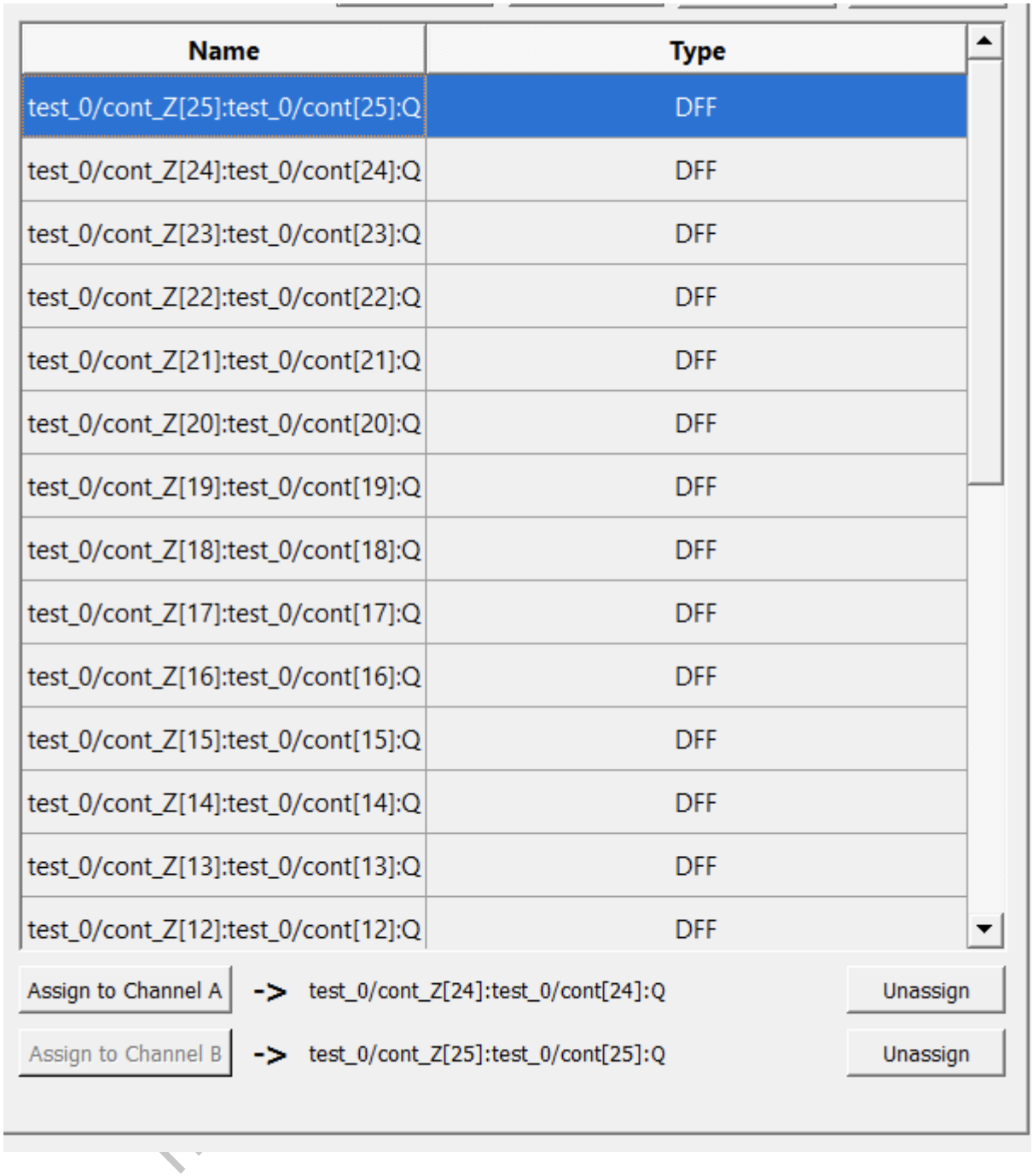
To do this I have two options, choose signal by signal which ones I want to debug (double click on each signal you want) or select all the signals at once (double click on the signals container). But you can only debug two at a time.



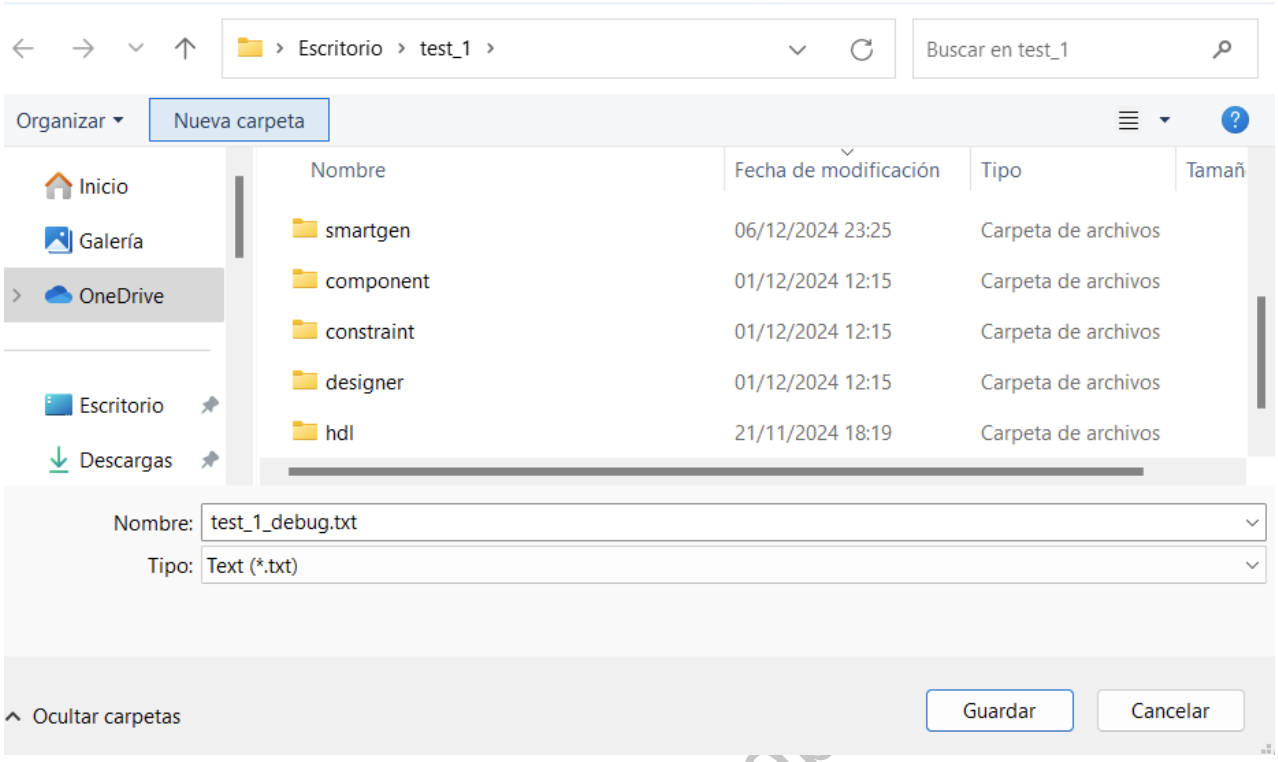
To debug signals, you select them in the interface and click on the buttons below *(Assign to Channel…)*.



You can select up to two signals, which you can debug instantly when you click the button.



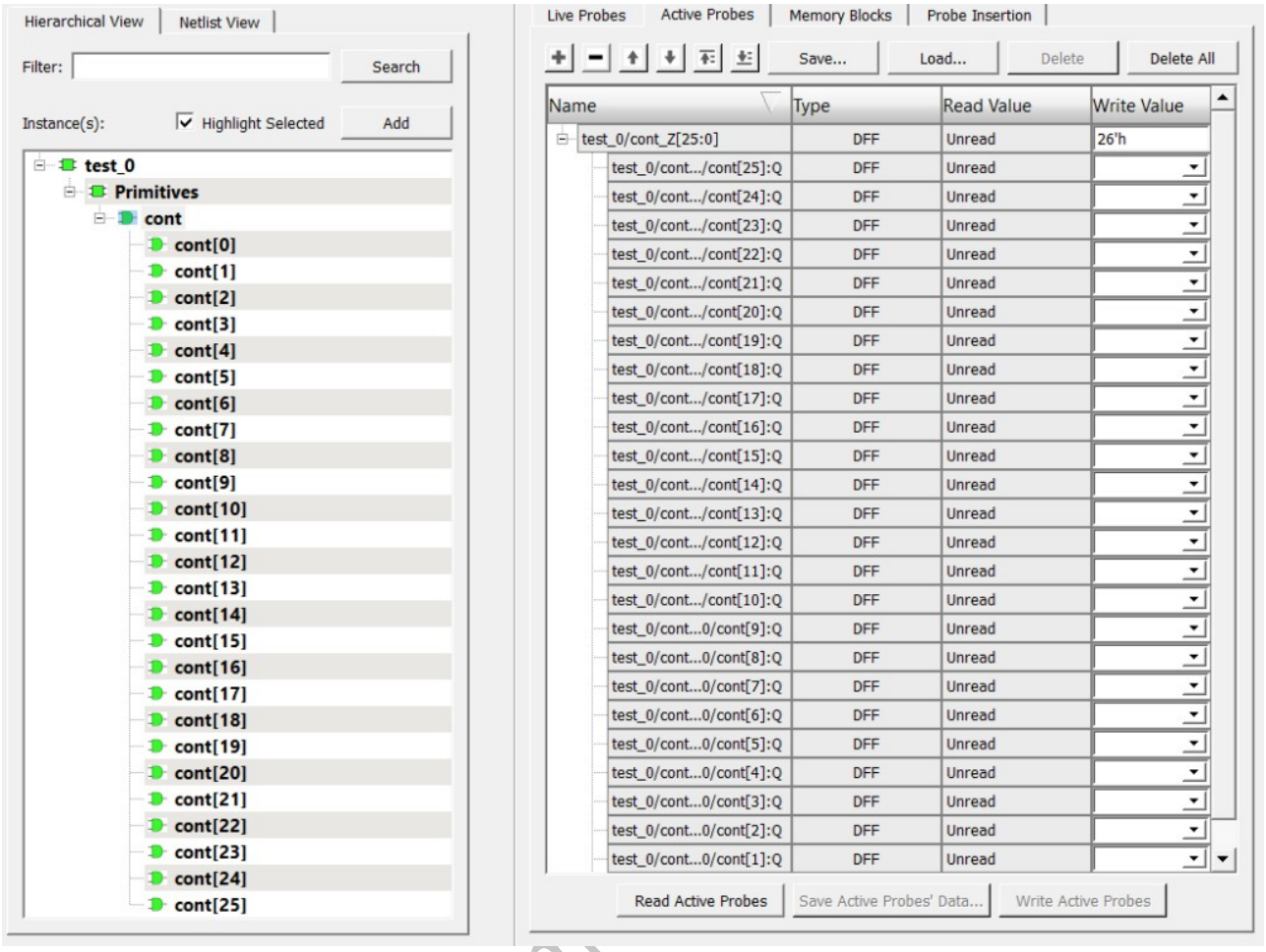
To facilitate later debugging, there is the option to save the debug profile, for this you click on *Save…*



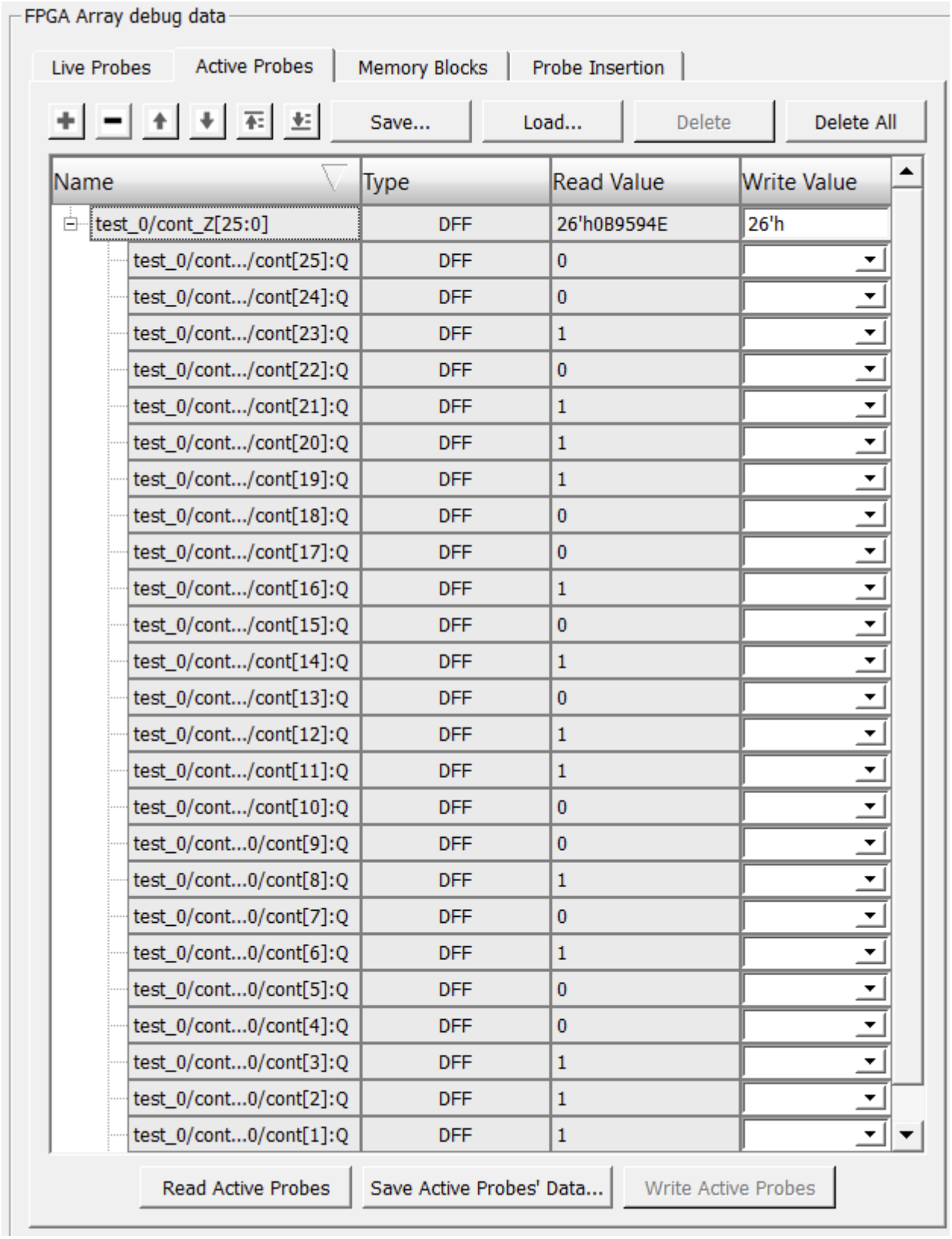
And it creates a file like the following one.

```
Bus test_0/cont_Z[25:0] display_format hex
test_0/cont_Z[25]:test_0/cont[25]:Q
test_0/cont_Z[24]:test_0/cont[24]:Q
test_0/cont_Z[23]:test_0/cont[23]:Q
test_0/cont_Z[22]:test_0/cont[22]:Q
test_0/cont_Z[21]:test_0/cont[21]:Q
test_0/cont_Z[20]:test_0/cont[20]:Q
test_0/cont_Z[19]:test_0/cont[19]:Q
test_0/cont_Z[18]:test_0/cont[18]:Q
test_0/cont_Z[17]:test_0/cont[17]:Q
test_0/cont_Z[16]:test_0/cont[16]:Q
test_0/cont_Z[15]:test_0/cont[15]:Q
test_0/cont_Z[14]:test_0/cont[14]:Q
test_0/cont_Z[13]:test_0/cont[13]:Q
test_0/cont_Z[12]:test_0/cont[12]:Q
test_0/cont_Z[11]:test_0/cont[11]:Q
test_0/cont_Z[10]:test_0/cont[10]:Q
test_0/cont_Z[9]:test_0/cont[9]:Q
test_0/cont_Z[8]:test_0/cont[8]:Q
test_0/cont_Z[7]:test_0/cont[7]:Q
test_0/cont_Z[6]:test_0/cont[6]:Q
test_0/cont_Z[5]:test_0/cont[5]:Q
test_0/cont_Z[4]:test_0/cont[4]:Q
test_0/cont_Z[3]:test_0/cont[3]:Q
test_0/cont_Z[2]:test_0/cont[2]:Q
test_0/cont_Z[1]:test_0/cont[1]:Q
test_0/cont_Z[0]:test_0/cont[0]:Q
Bus end
```

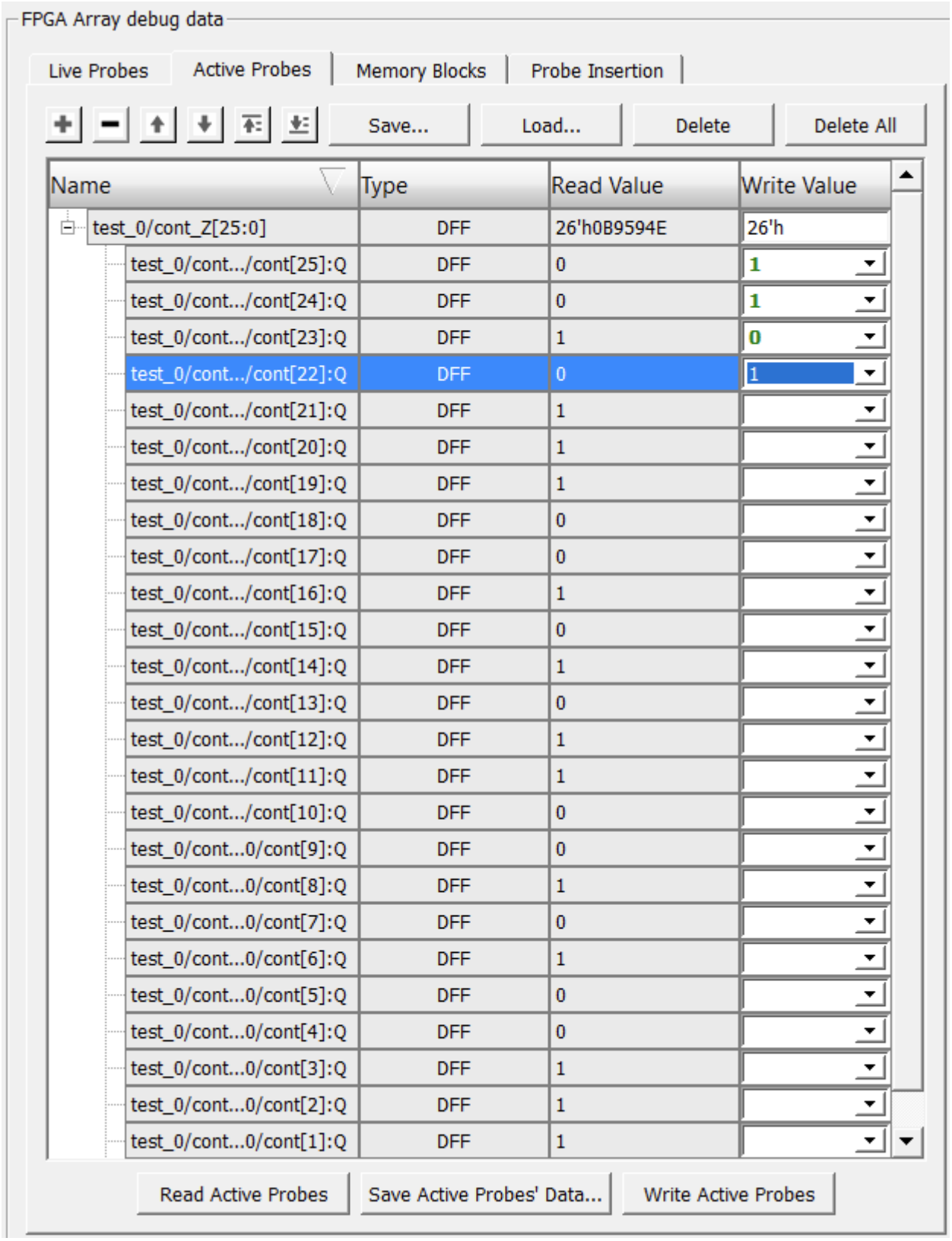
- **Active Probes:** this option is made to debug all the internal discrete values of the FPGA. This tool can read all the internal signal values and can also write the desired value to them.



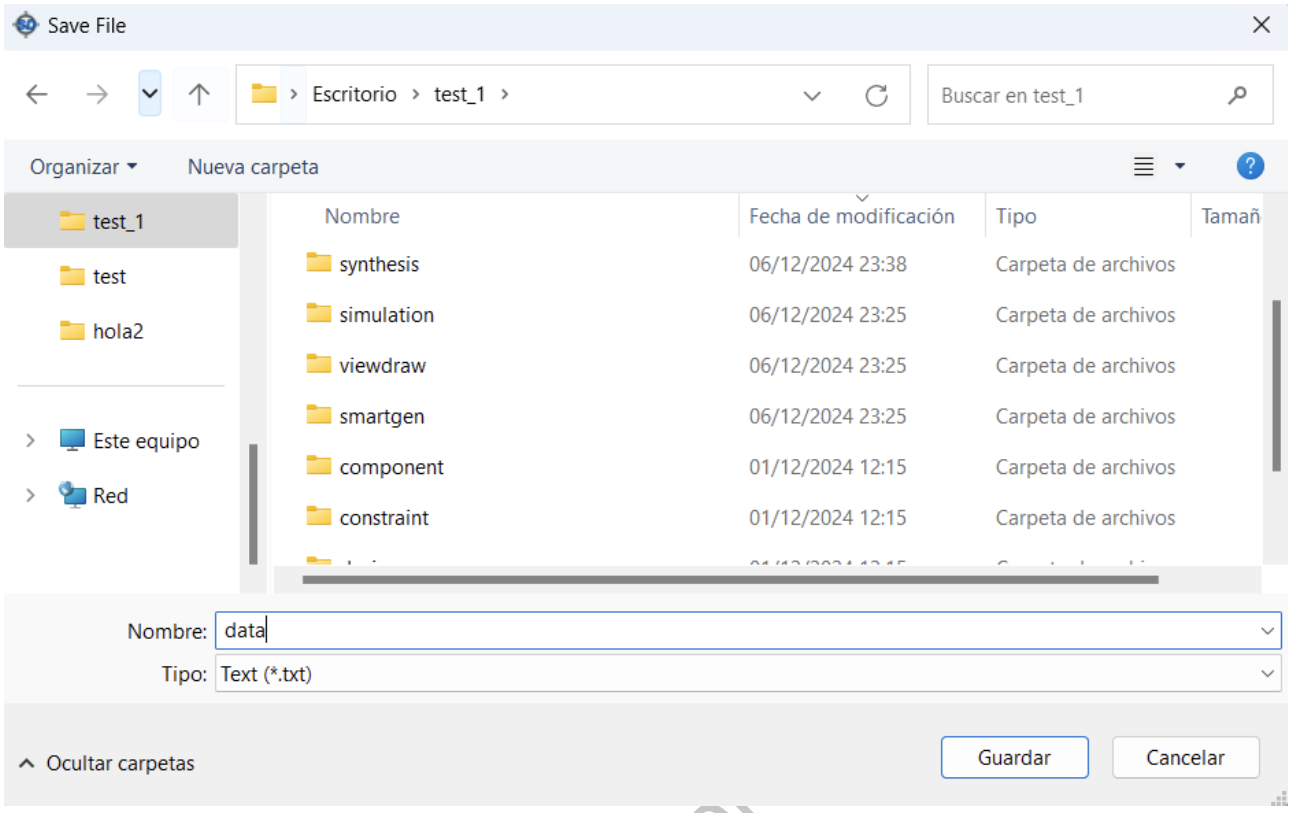
To do this, click on the *Read Active Probes* button, then you can see all the internal values of the signals you have selected.



To write an internal value you have to change the desired signal value and then click on the *Write Active Probes* button.



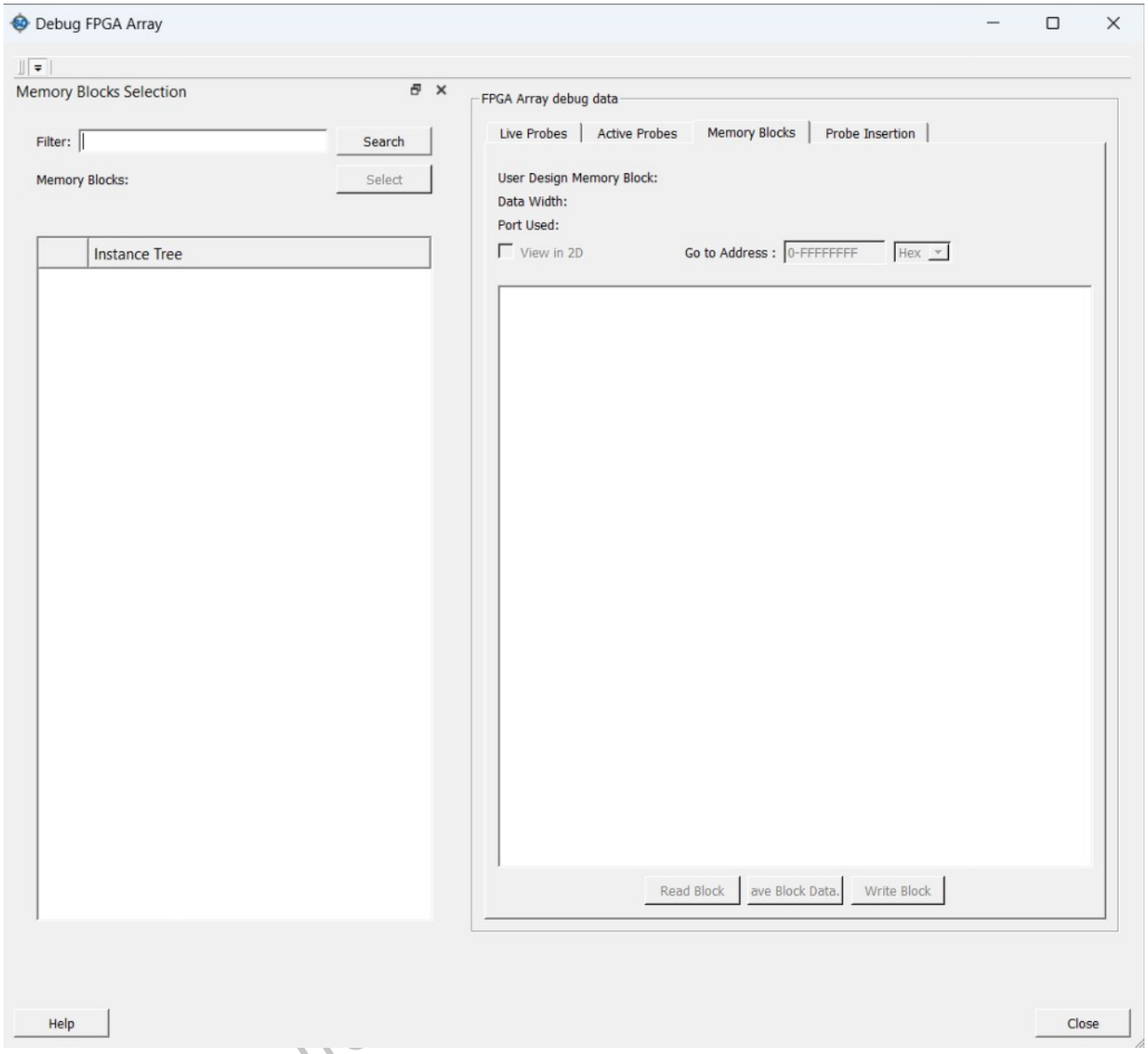
If you want to save the read values in a file, you only need to click on the *Save Active Probes' Data* option.



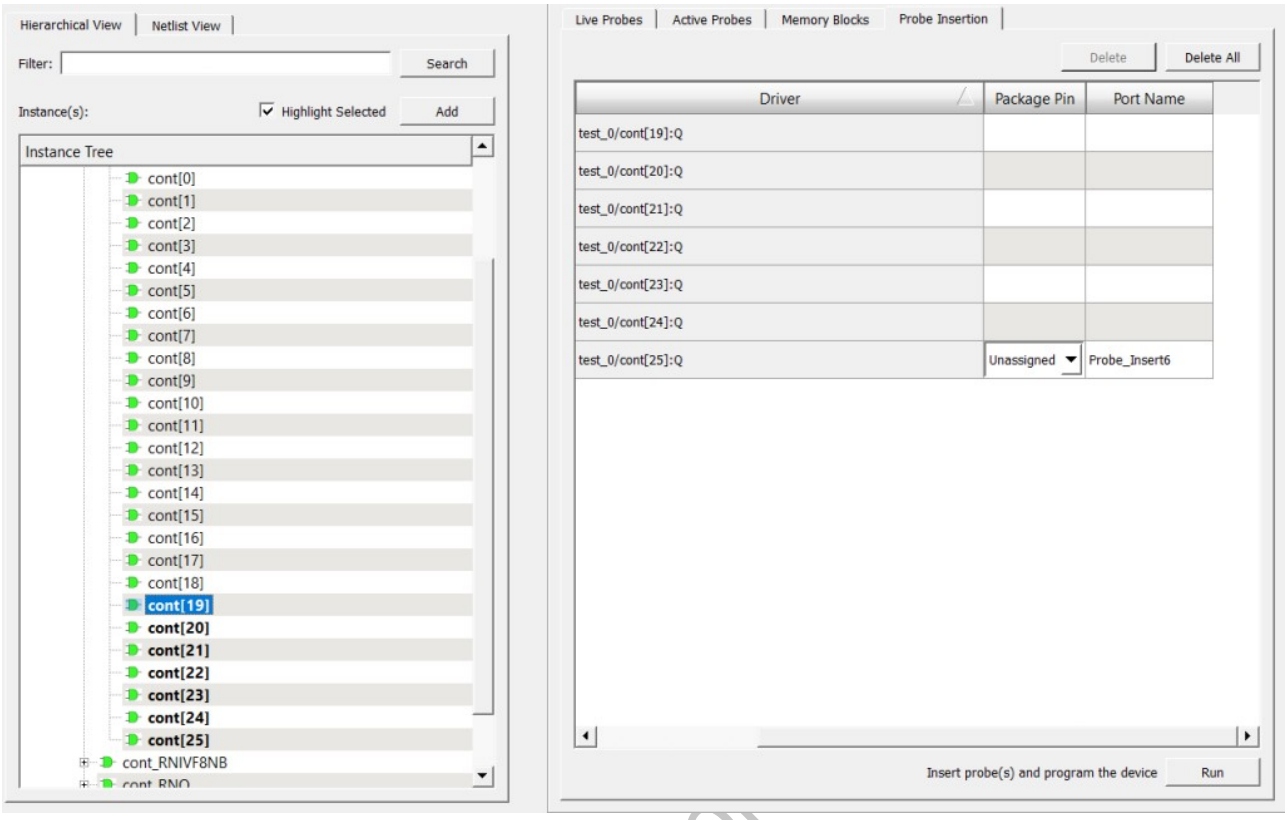
This generates a text file with the following form.

```
Active probes and values for .dprj

Saved at: vi. dic. 6 23:48:27 2024

+-----------------------------------------+------+-------------+
| Name                                    | Type | Read Value  |
+-----------------------------------------+------+-------------+
|                                         |      |             |
| test_0/cont_Z[25:0]                     | DFF  | 26'h31FDA74 |
|     test_0/cont_Z[25]:test_0/cont[25]:Q | DFF  | 1           |
|     test_0/cont_Z[24]:test_0/cont[24]:Q | DFF  | 1           |
|     test_0/cont_Z[23]:test_0/cont[23]:Q | DFF  | 0           |
|     test_0/cont_Z[22]:test_0/cont[22]:Q | DFF  | 0           |
|     test_0/cont_Z[21]:test_0/cont[21]:Q | DFF  | 0           |
|     test_0/cont_Z[20]:test_0/cont[20]:Q | DFF  | 1           |
|     test_0/cont_Z[19]:test_0/cont[19]:Q | DFF  | 1           |
|     test_0/cont_Z[18]:test_0/cont[18]:Q | DFF  | 1           |
|     test_0/cont_Z[17]:test_0/cont[17]:Q | DFF  | 1           |
|     test_0/cont_Z[16]:test_0/cont[16]:Q | DFF  | 1           |
|     test_0/cont_Z[15]:test_0/cont[15]:Q | DFF  | 1           |
|     test_0/cont_Z[14]:test_0/cont[14]:Q | DFF  | 1           |
|     test_0/cont_Z[13]:test_0/cont[13]:Q | DFF  | 0           |
|     test_0/cont_Z[12]:test_0/cont[12]:Q | DFF  | 1           |
|     test_0/cont_Z[11]:test_0/cont[11]:Q | DFF  | 1           |
|     test_0/cont_Z[10]:test_0/cont[10]:Q | DFF  | 0           |
|     test_0/cont_Z[9]:test_0/cont[9]:Q   | DFF  | 1           |
|     test_0/cont_Z[8]:test_0/cont[8]:Q   | DFF  | 0           |
|     test_0/cont_Z[7]:test_0/cont[7]:Q   | DFF  | 0           |
|     test_0/cont_Z[6]:test_0/cont[6]:Q   | DFF  | 1           |
|     test_0/cont_Z[5]:test_0/cont[5]:Q   | DFF  | 1           |
|     test_0/cont_Z[4]:test_0/cont[4]:Q   | DFF  | 1           |
|     test_0/cont_Z[3]:test_0/cont[3]:Q   | DFF  | 0           |
|     test_0/cont_Z[2]:test_0/cont[2]:Q   | DFF  | 1           |
|     test_0/cont_Z[1]:test_0/cont[1]:Q   | DFF  | 0           |
|     test_0/cont_Z[0]:test_0/cont[0]:Q   | DFF  | 0           |
|                                         |      |             |
+-----------------------------------------+------+-------------+
```

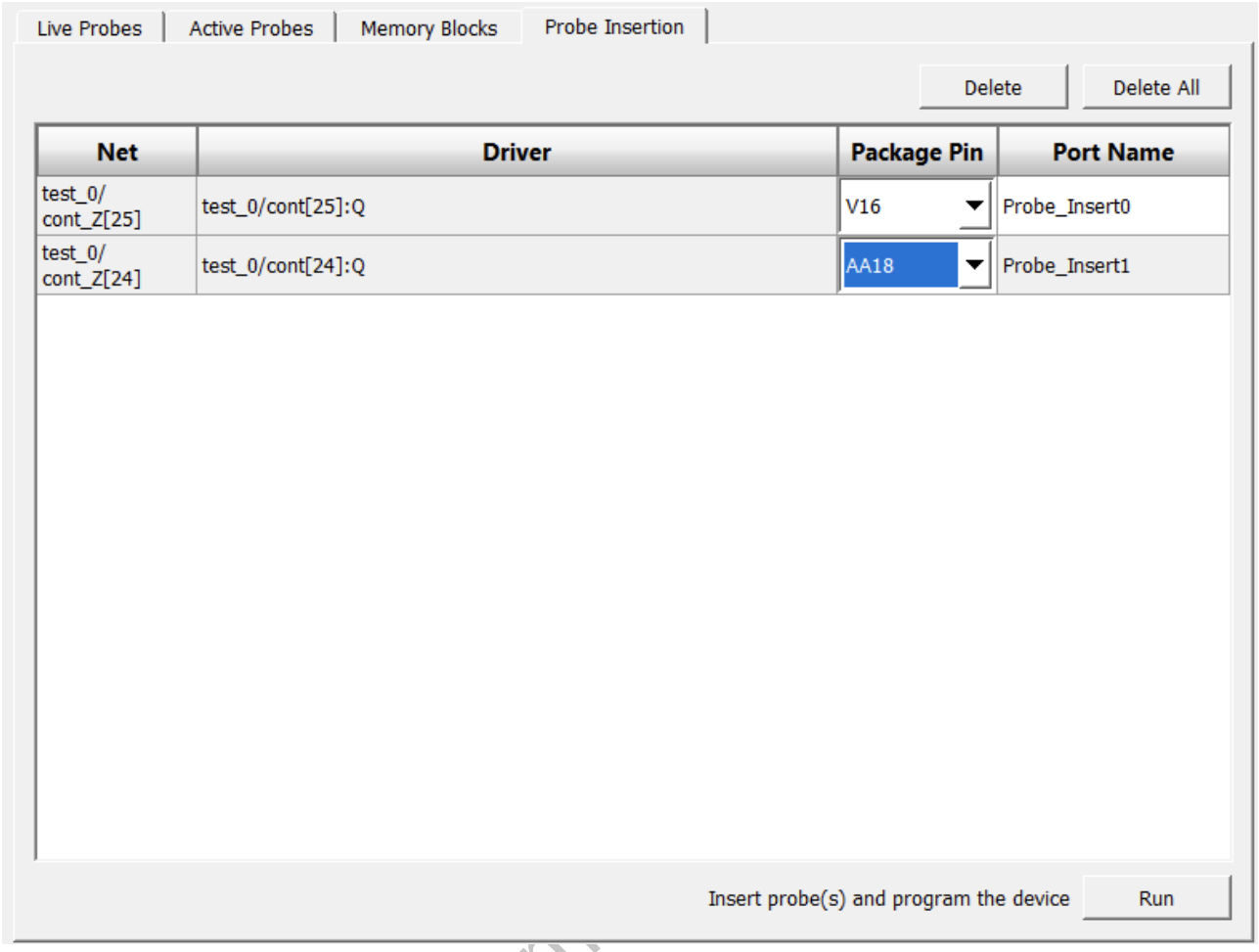
- **Memory Block**: This is an option to read the internal memory block by block. This option is only enabled if there is a predefined memory block for an FPGA/SoC.



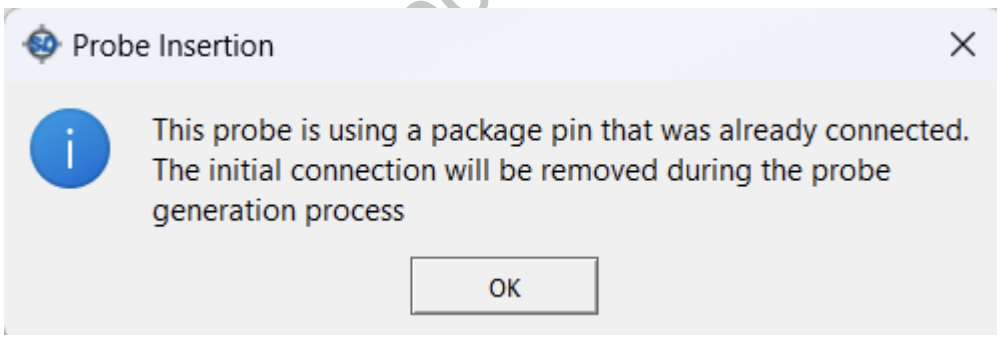
- **Probe Insertion**: This option does the same as *Live Probes*, but instead of outputting the signal through two specific pins of the FPGA, you can choose any pin of the package to output the signal. This option is created to output signals through free pins of the FPGA. To do this, first you select the signals you want to output.



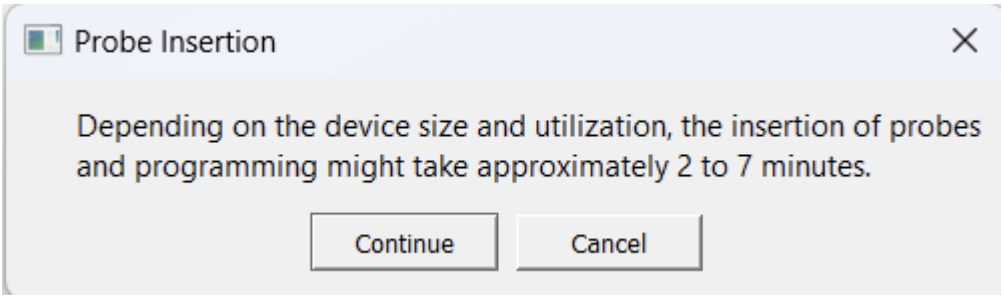
And then you select the pin through which you want to output the signal.



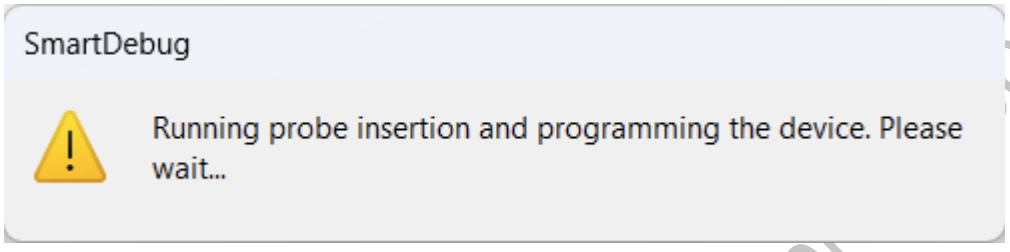
NOTE: *you can output from any pin, even those that have been selected to generate the bitstream. If you do this, skip this tab.*



Once you have chosen the signals you want to output along with the pins, click Run. Then, a tab like the following one appears. It says that it can take between 2 to 7 minutes for it to take effect, that is because what it does internally is generate a new bitstream for debugging, to do this it takes advantage of the synthesis already done and makes a new *Place and Route*, and then loads a new bitstream.



When it loads the new bitstream, the FPGA resets and starts from scratch again, so it does not load the bitstream dynamically. It is important to keep this in mind. **This does not happen with *Live Probes***.

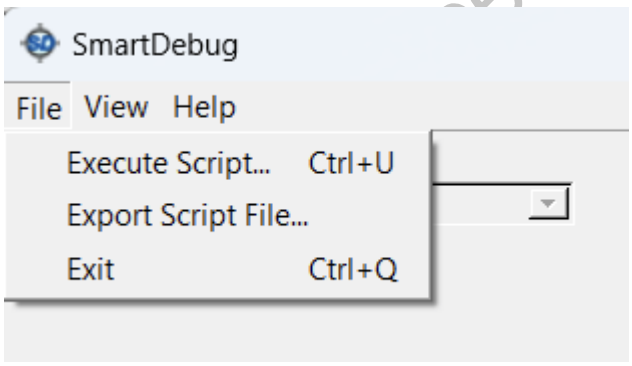


Once finished, output the new signals from the desired pins.

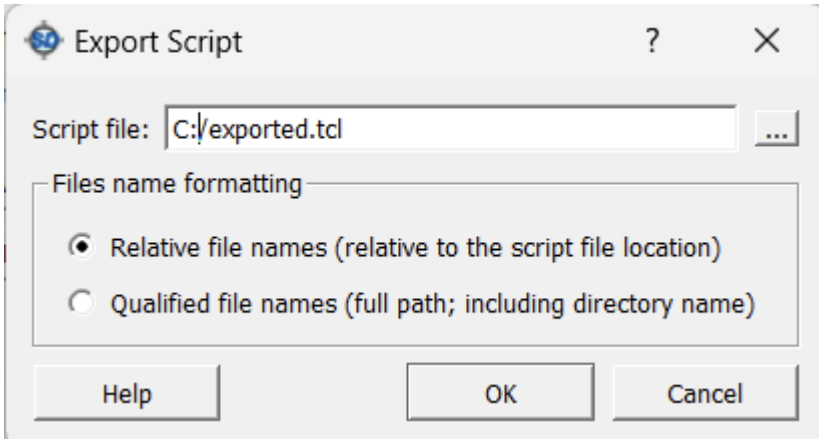
Final note

If you want to save the debug profiles to facilitate later debugging, what you can do is export a TCL file with the settings used.

To do this, in File, click *Export Script File…*



When you click on it, it asks you to tell it the path in which to save the file.



When you save it, it generates a file like the following one.

```
# Microsemi Tcl Script
# sdbg
# Date: Sat Dec  7 00:11:33 2024
# Directory C:\Users\David\Desktop\test_1
# File C:\..\exported.tcl


export_script -file {./exported.tcl} -relative_path 1
select_active_probe \
         -name {test_0/cont_Z[0]:test_0/cont[0]:Q} \
         -name {test_0/cont_Z[10]:test_0/cont[10]:Q} \
         -name {test_0/cont_Z[11]:test_0/cont[11]:Q} \
         -name {test_0/cont_Z[12]:test_0/cont[12]:Q} \
         -name {test_0/cont_Z[13]:test_0/cont[13]:Q} \
         -name {test_0/cont_Z[14]:test_0/cont[14]:Q} \
         -name {test_0/cont_Z[15]:test_0/cont[15]:Q} \
         -name {test_0/cont_Z[16]:test_0/cont[16]:Q} \
         -name {test_0/cont_Z[17]:test_0/cont[17]:Q} \
         -name {test_0/cont_Z[18]:test_0/cont[18]:Q} \
         -name {test_0/cont_Z[19]:test_0/cont[19]:Q} \
         -name {test_0/cont_Z[1]:test_0/cont[1]:Q} \
         -name {test_0/cont_Z[20]:test_0/cont[20]:Q} \
         -name {test_0/cont_Z[21]:test_0/cont[21]:Q} \
         -name {test_0/cont_Z[22]:test_0/cont[22]:Q} \
         -name {test_0/cont_Z[23]:test_0/cont[23]:Q} \
         -name {test_0/cont_Z[24]:test_0/cont[24]:Q} \
         -name {test_0/cont_Z[25]:test_0/cont[25]:Q} \
         -name {test_0/cont_Z[2]:test_0/cont[2]:Q} \
         -name {test_0/cont_Z[3]:test_0/cont[3]:Q} \
         -name {test_0/cont_Z[4]:test_0/cont[4]:Q} \
         -name {test_0/cont_Z[5]:test_0/cont[5]:Q} \
         -name {test_0/cont_Z[6]:test_0/cont[6]:Q} \
         -name {test_0/cont_Z[7]:test_0/cont[7]:Q} \
         -name {test_0/cont_Z[8]:test_0/cont[8]:Q} \
         -name {test_0/cont_Z[9]:test_0/cont[9]:Q} \
         -reset 0
read_active_probe
export_script -file {./exported.tcl} -relative_path 1
remove_probe_insertion_point -net {test_0/cont_Z[25]} -driver {test_0/cont[25]:Q}
program_probe_insertion -all 0
```

To import it, click *Execute Script*…

References

For more information, you can see the following video.

https://youtu.be/7-hmSy-TdPI