

Cómo depurar utilizando Synplify Pro e Identify Debugger en Libero

Creador: David Rubio G.

Entrada: <https://soceame.wordpress.com/2024/12/08/como-depurar-utilizando-synplify-pro-e-identify-me-en-libero/>

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

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

Fecha última modificación: 23/02/2025

Esto es una cosa no muy conocida sobre las FPGAs/SoCs de Microchip y es que se pueden depurar utilizando otras herramientas incluidas en la instalación. Estas herramientas son el **Synplify Pro** (para seleccionar las señales de depuración) y el **Identify Debugger** (para visualizar la señales depuradas), ambas herramientas desarrolladas por Synopsys.

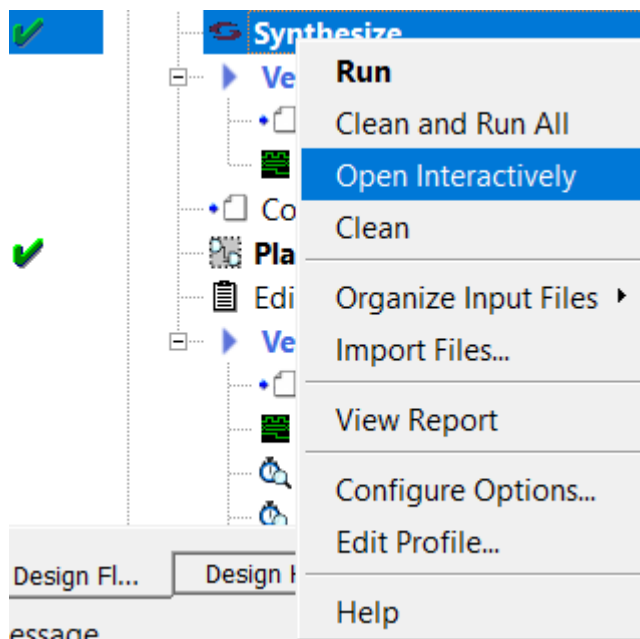
NOTA: estas dos herramientas hacen algo parecido a los bloques **ILA** de Xilinx o al **Signal Tap Logic Analyzer** de Altera. Estas herramientas se pueden abrir directamente desde Libero o desde la propia aplicación.

(Pasos previos)

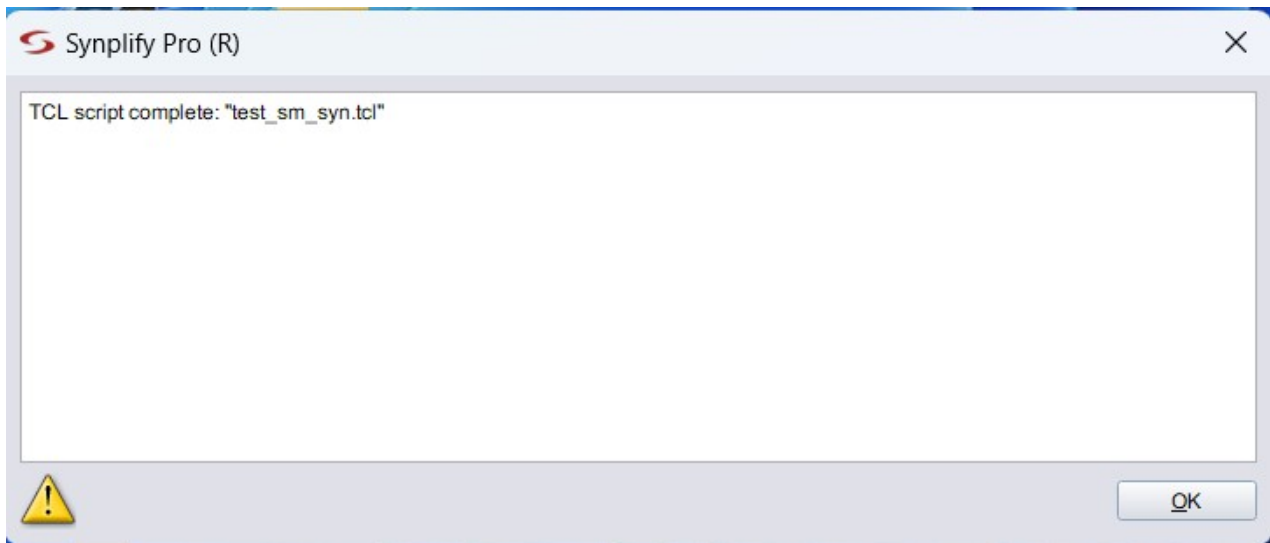
Este sistema de depuración primero necesita que el usuario haya sintetizado un proyecto en Libero. Para ello se pincha en la opción de *Synthesize*.

Synplify Pro

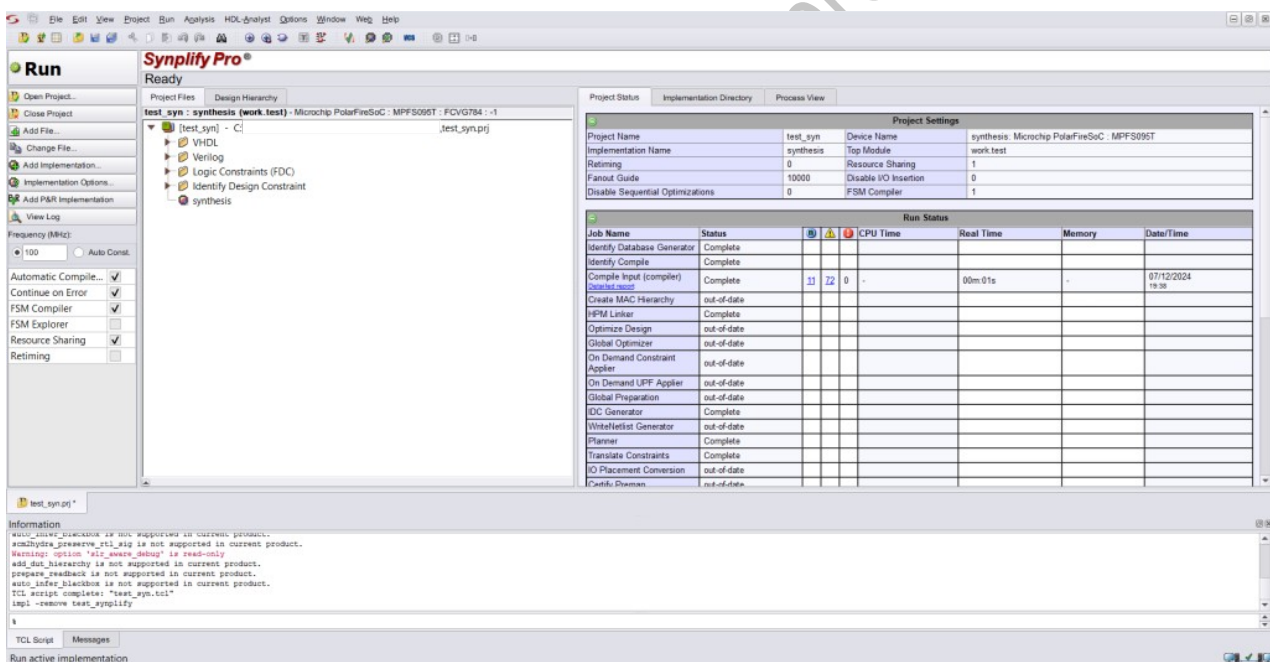
Una vez hayamos sintetizado el proyecto, le damos clic derecho a la opción de *Synthesize* y le damos a la opción de *Open Interactively*.



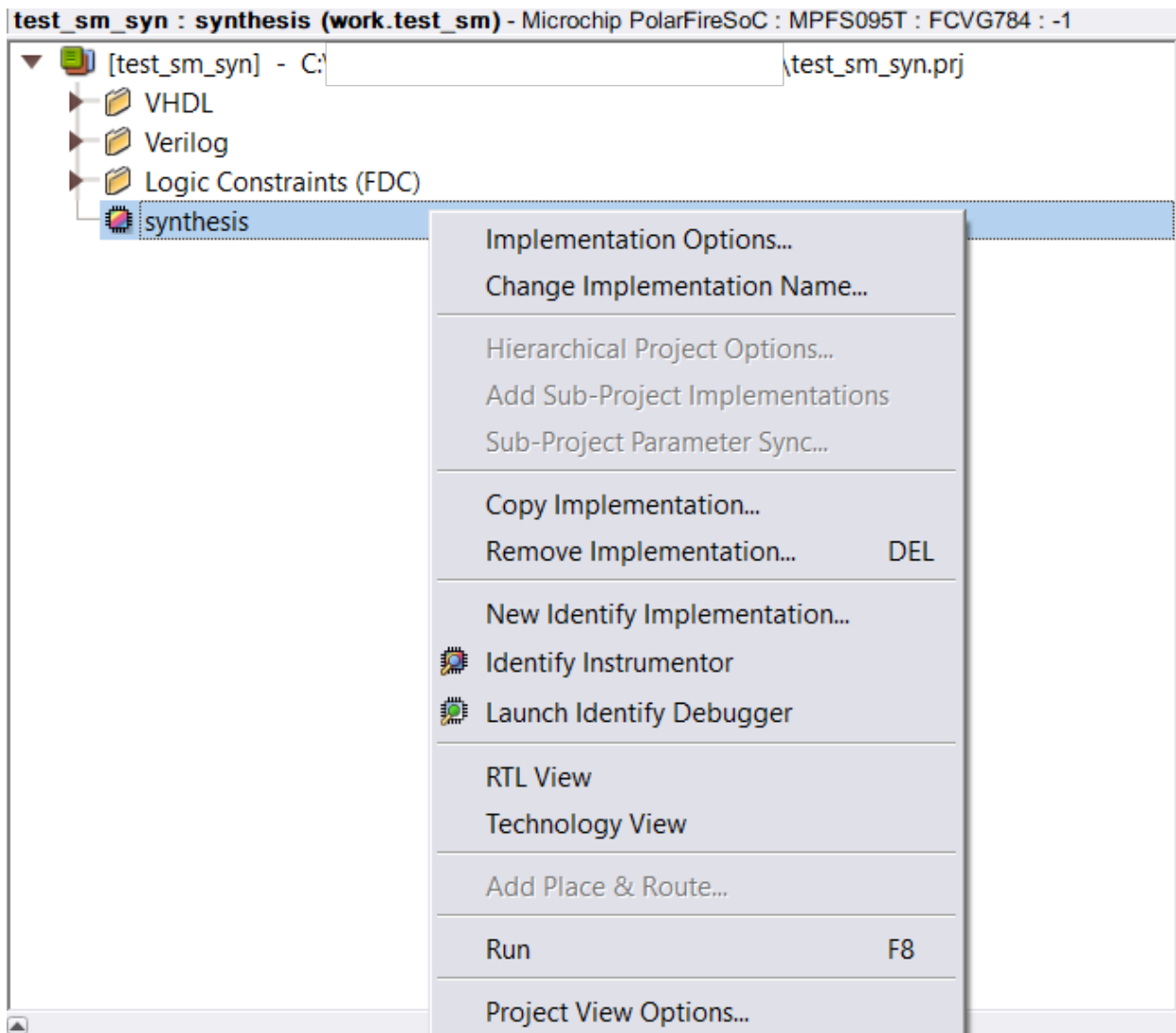
Al pulsarlo lo que hace es abrir el **Synplify Pro**, y al abrirse nos muestra esta pestaña.



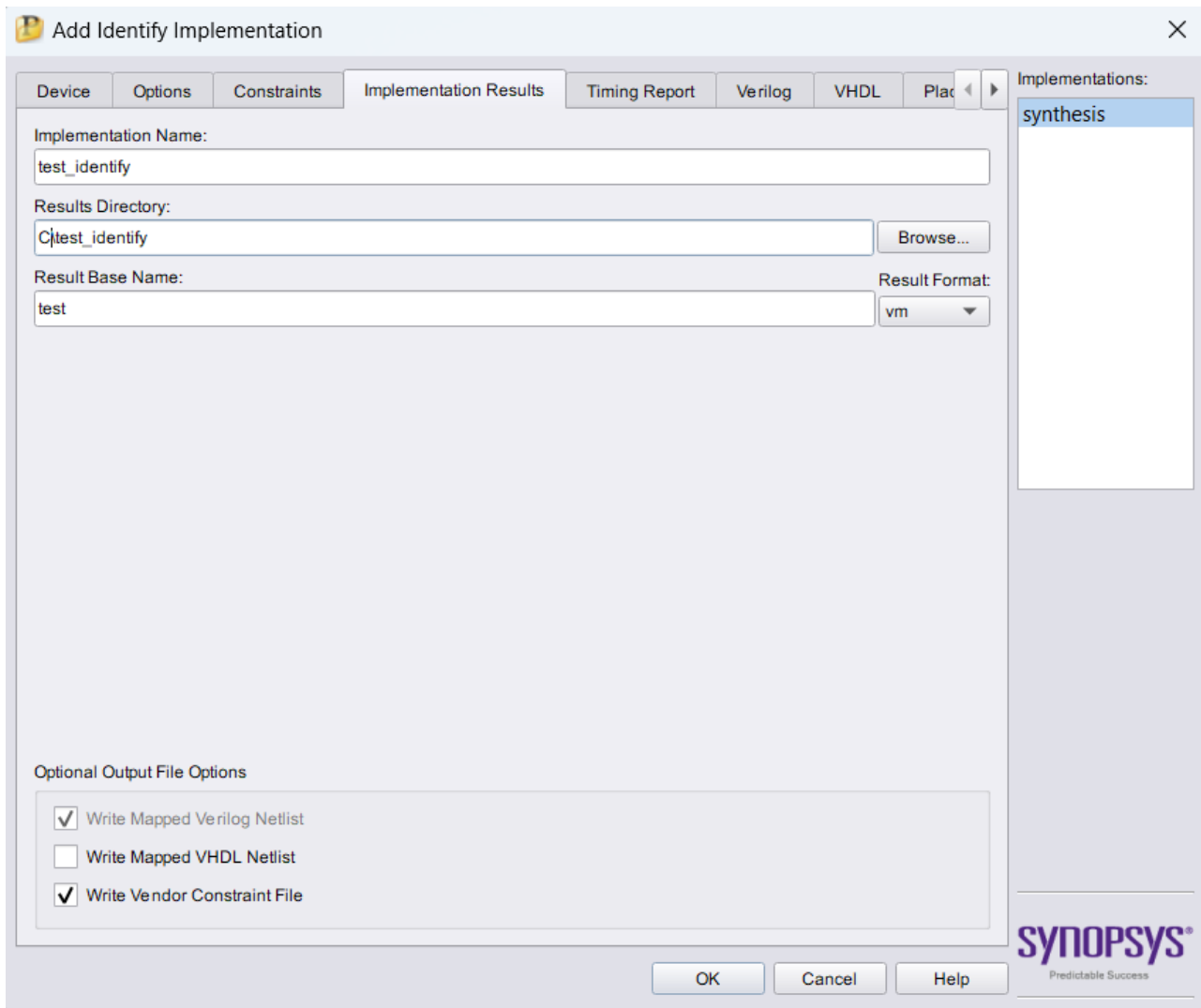
Después nos abre una pestaña que es dónde figura nuestro proyecto con la síntesis que ha hecho Libero (llamada *synthesis*). Bien, pues ahora lo que hay que hacer es crear un nuevo perfil de síntesis que nos permita añadir los analizadores lógicos.



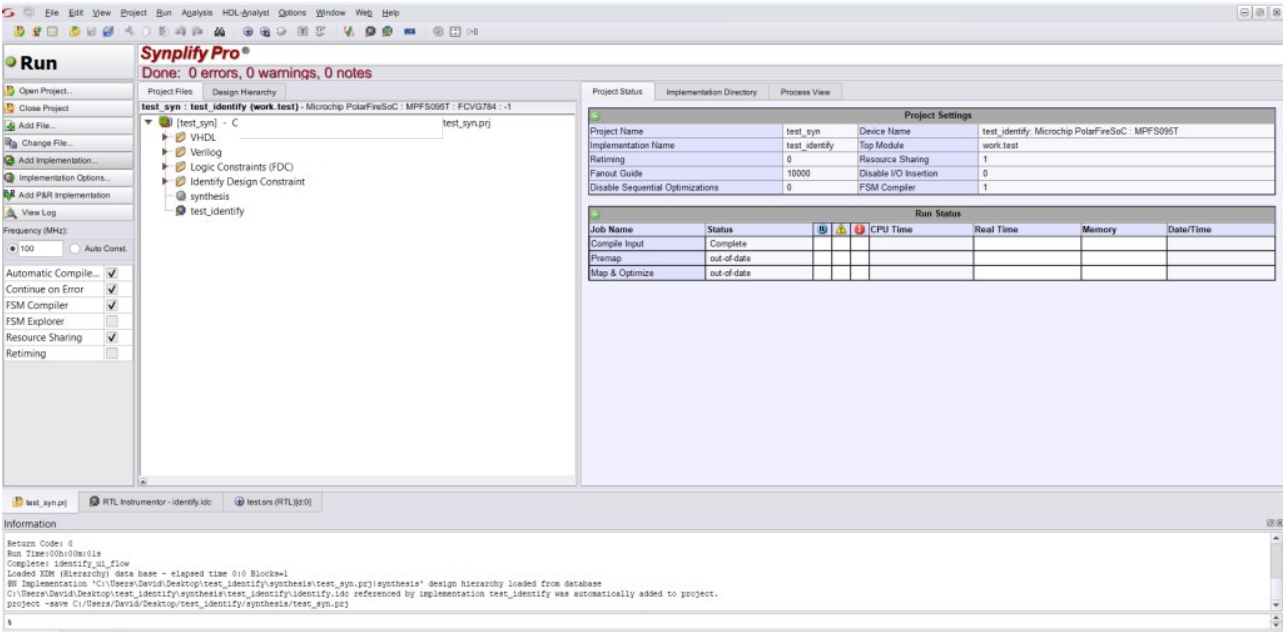
Para ello se le da clic derecho al perfil de síntesis que viene de Libero, y se crea uno nuevo en *New Identify Implementation*.



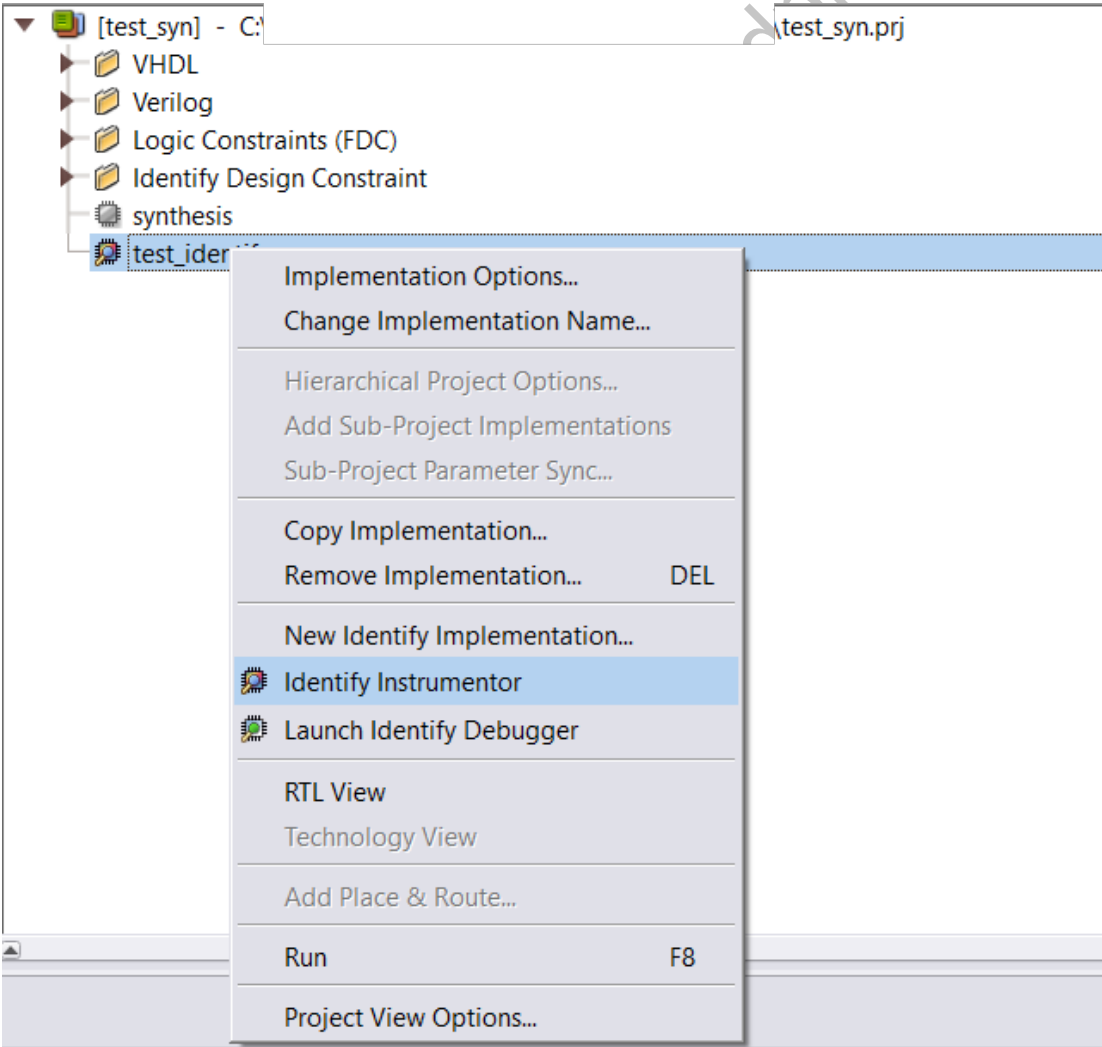
Entonces, se nos abre una pestaña que nos pregunta por el nombre del nuevo perfil de depuración y dónde se va a guardar. El *Result Base Name* es el nombre de nuestro fichero top de desarrollo.



Una vez creado el nuevo perfil, le damos clic izquierdo en el perfil, para indicar que queremos utilizar ese perfil.

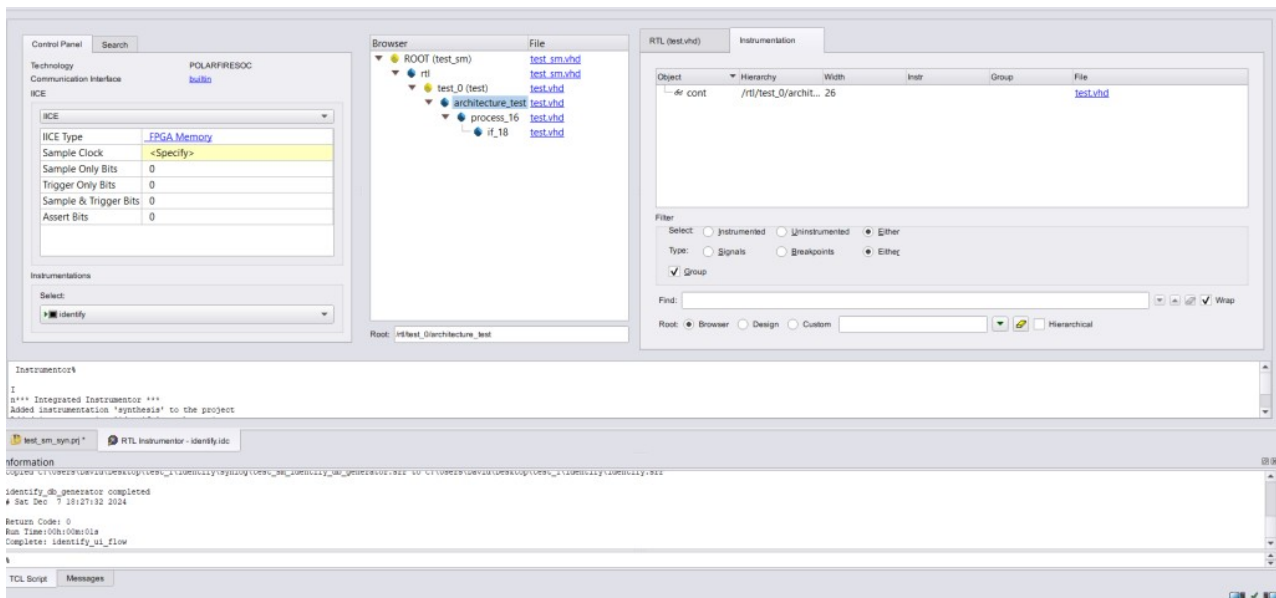


Después, le damos a *Identify Instrumentor*.



Esto nos abre una nueva pestaña dentro del programa (*por lo que para volver a la pestaña anterior hay que ir a la parte inferior*).

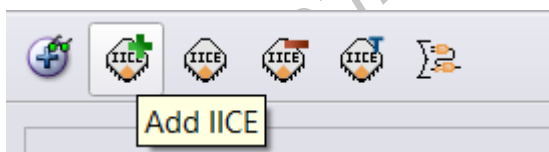
Esta nueva pestaña es donde hay que definir las señales que se quieren analizar.



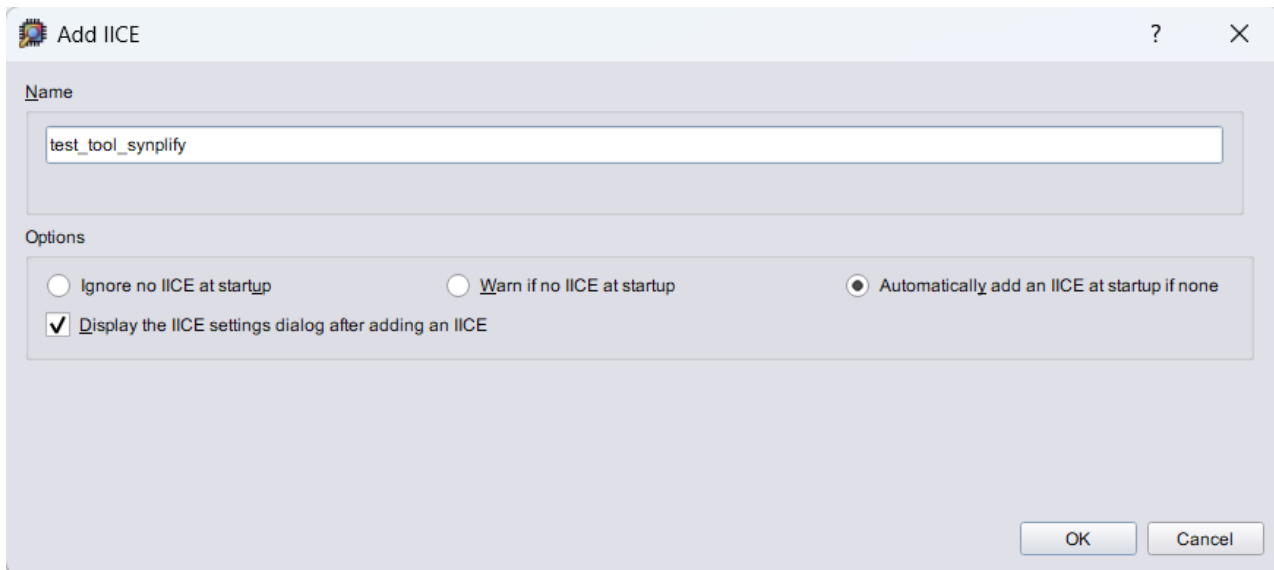
Aquí aparecen 3 conjuntos de ventanas: el de la **izquierda** son los grupos de depuración, esto quiere decir que puedes crear en un mismo perfil de depuración diferentes conjuntos de señales a analizar; la pestaña del **centro** que es donde figura la estructura del proyecto para que el usuario elija que quiere analizar; y la pestaña de la **derecha** que es en la que se elige se señal o puerto se quiere analizar, para ello en la selección del fichero de la pestaña del medio hace aparecer aquí el fichero con iconos en todas las señales a analizar.

Lo primero que hay que hacer es crear un nuevo grupo de análisis de señales (también se puede utilizar el que hay por defecto, **IICE**).

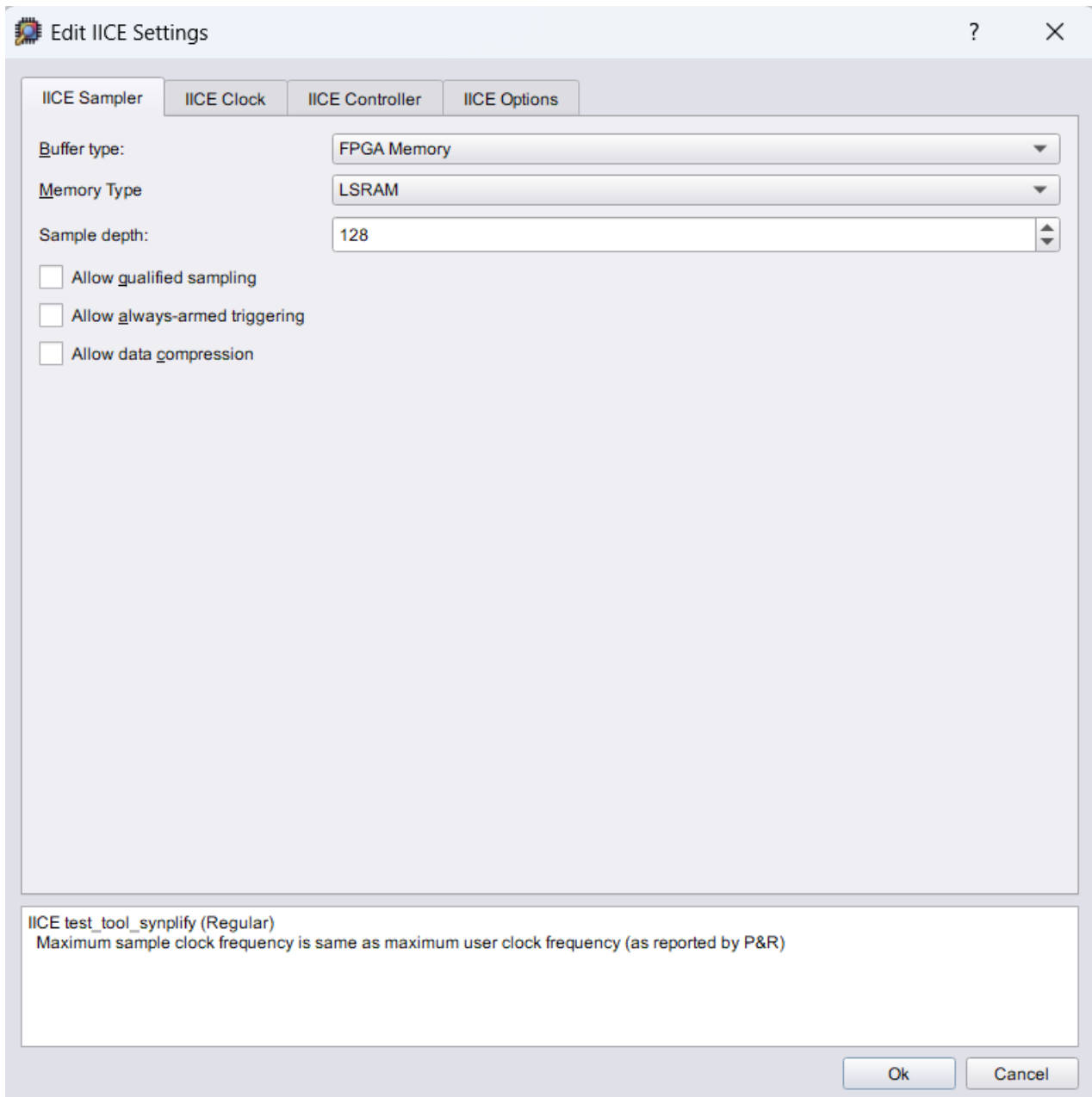
NOTA: si no se quiere utilizar el de por defecto, se recomienda borrarlo.



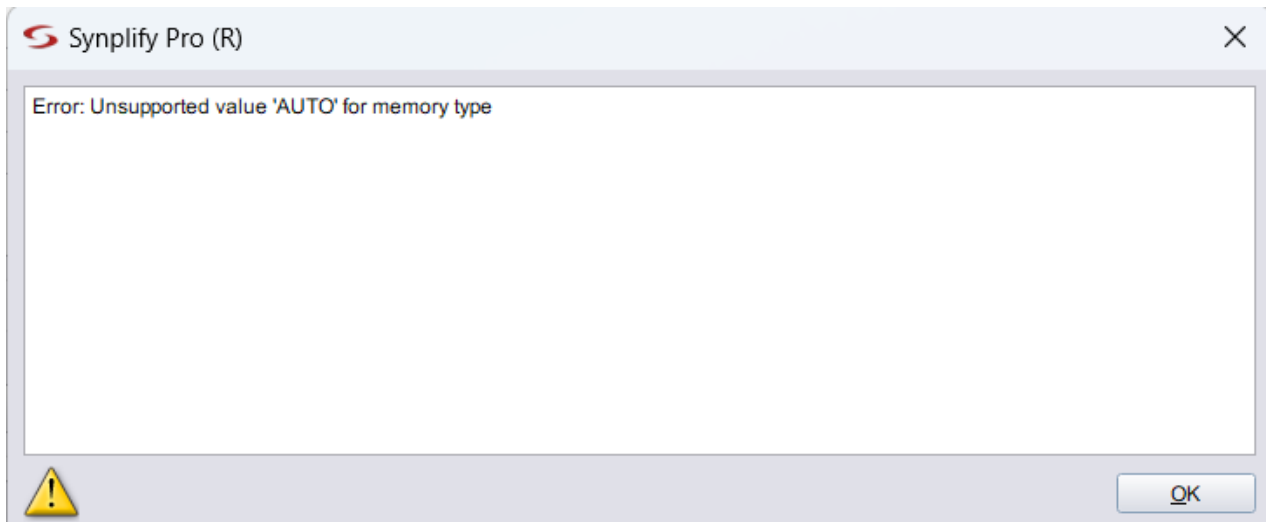
Al pulsar en **Add IICE**, nos pregunta por el nuevo conjunto de señales.



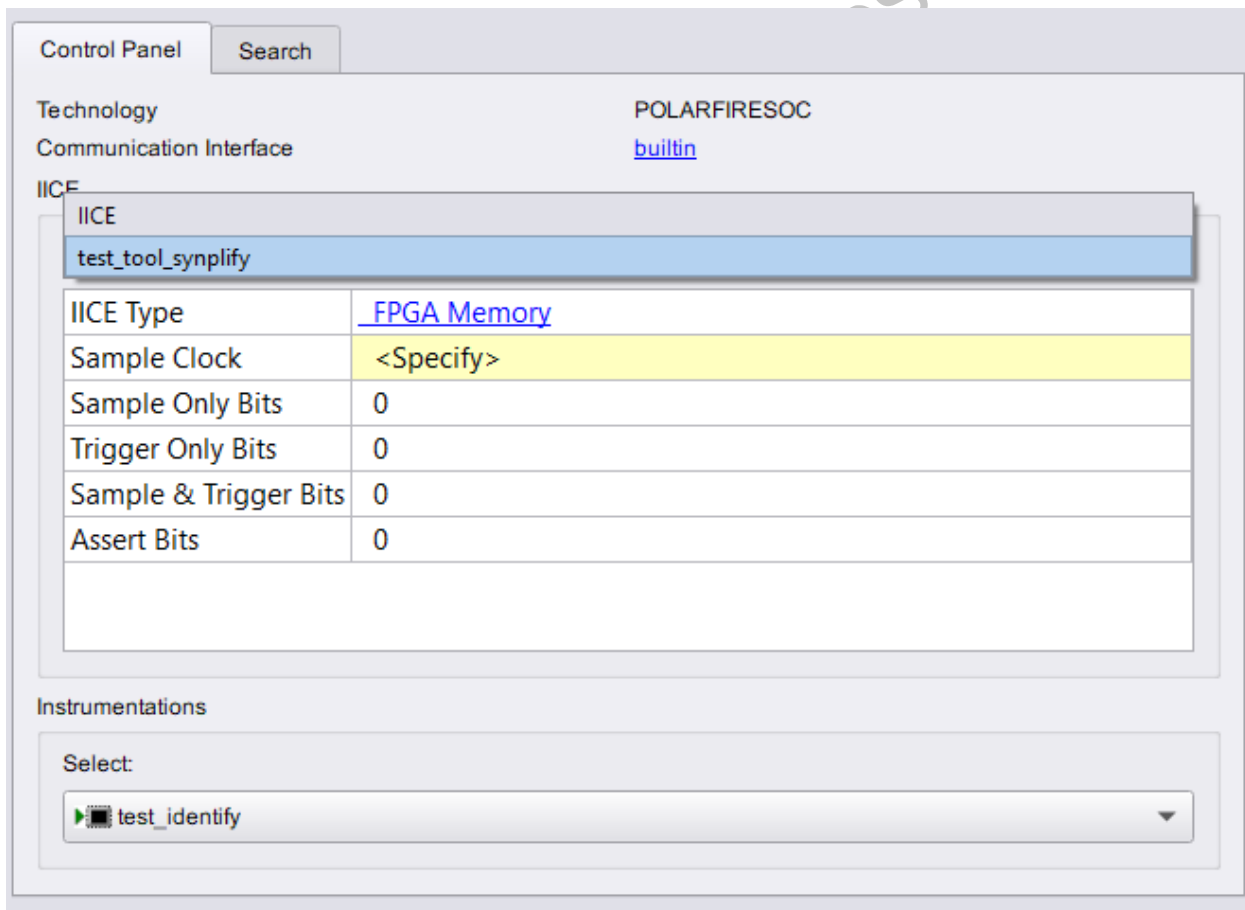
Luego se nos abre una pestaña dónde nos pregunta por la memoria dónde se grabarán las muestras, cuantas muestras se van a guardar, el tipo de reloj que se va a utilizar.



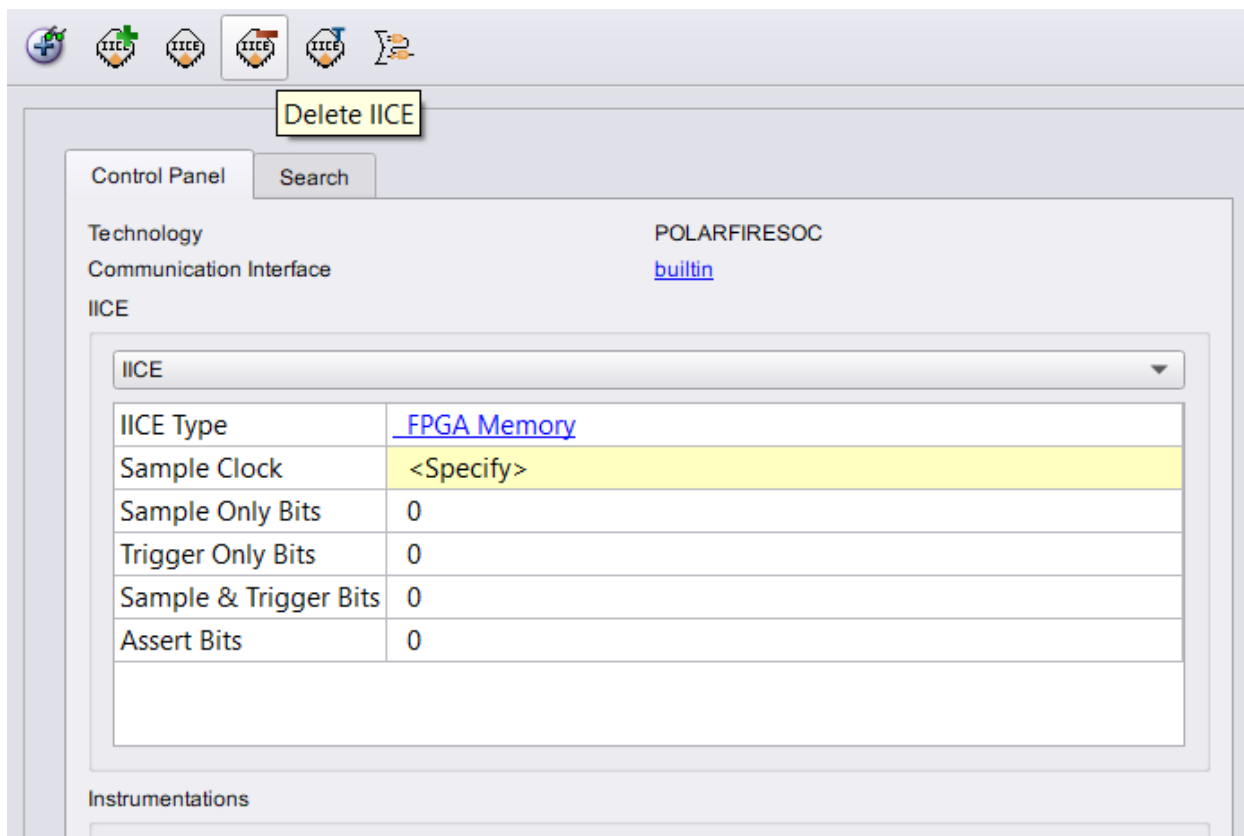
NOTA: es posible que en *Memory Type*, si se deja como *AUTO*, dé algún mensaje de error.



En la pestaña de la izquierda podemos ver los conjuntos de muestras y los diferentes perfiles de depuración (esto es la opción *Instrumentations* de abajo)



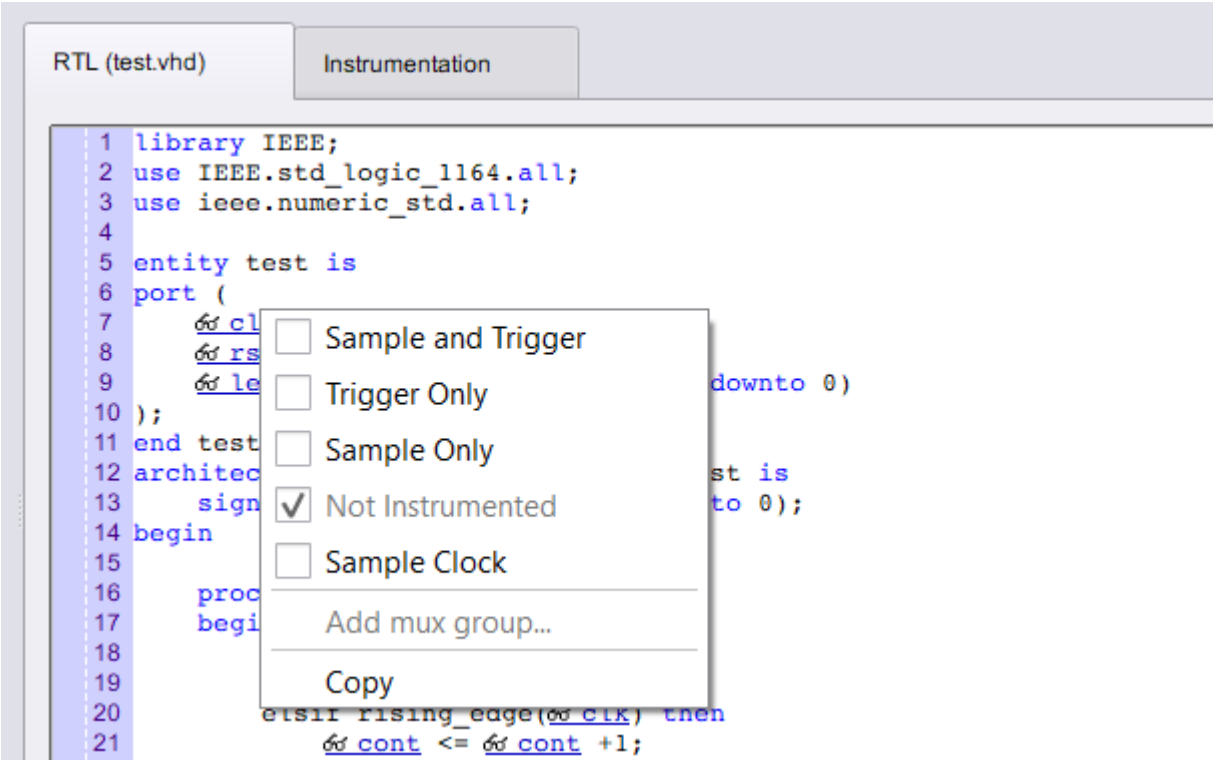
Para borrar, se selecciona el perfil y se le da a *Delete IICE*.



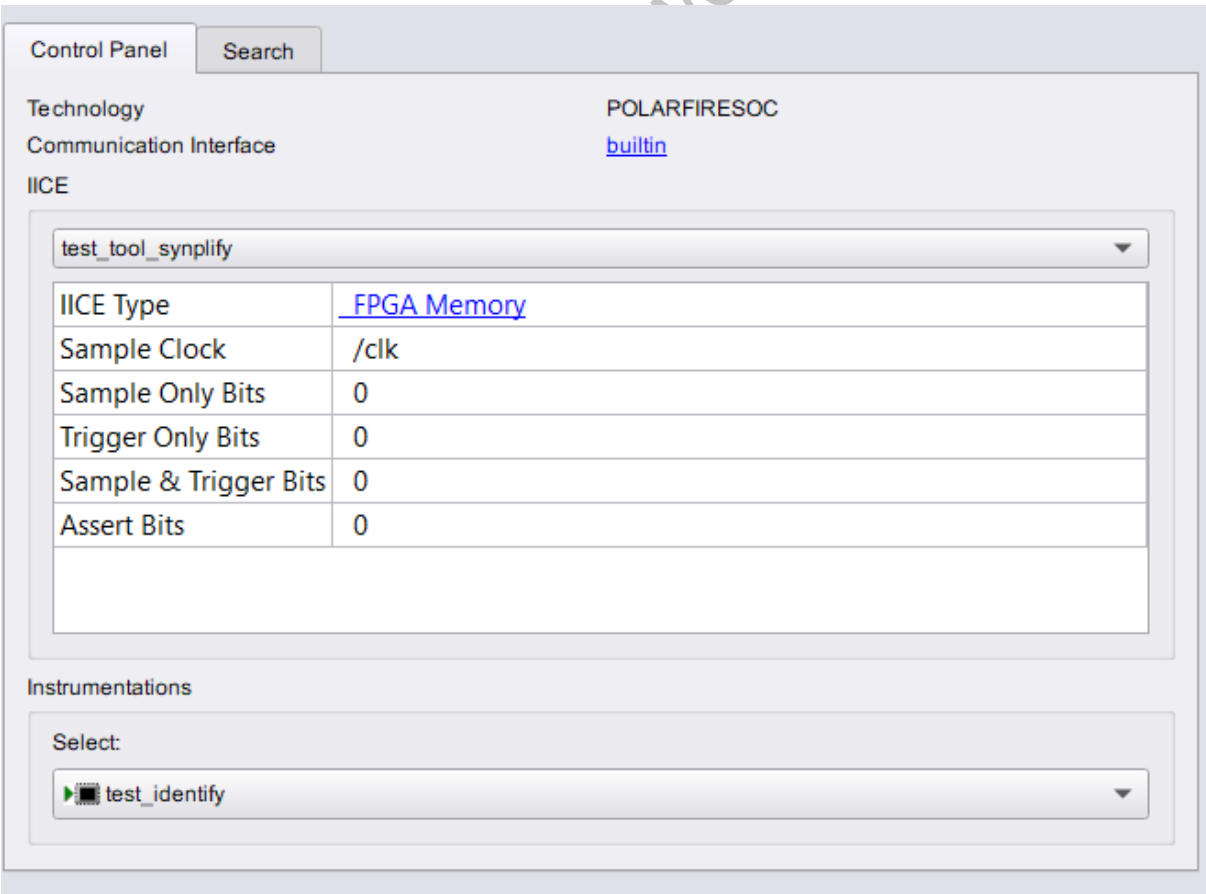
Una vez se tiene creado el contenedor del conjunto de señales que queremos crear, pasamos a añadir señales.

- **Reloj de referencia**

El primer paso para cada conjunto de señales del perfil de depuración es seleccionar el reloj de referencia para ese grupo. Para ello vamos al código fuente de la derecha (*después, de haber seleccionado el fichero en la pestaña del centro*) y pinchamos en el reloj, y se nos abre un desplegable, entonces, solo tenemos que marcar la opción de *Sample Clock*.

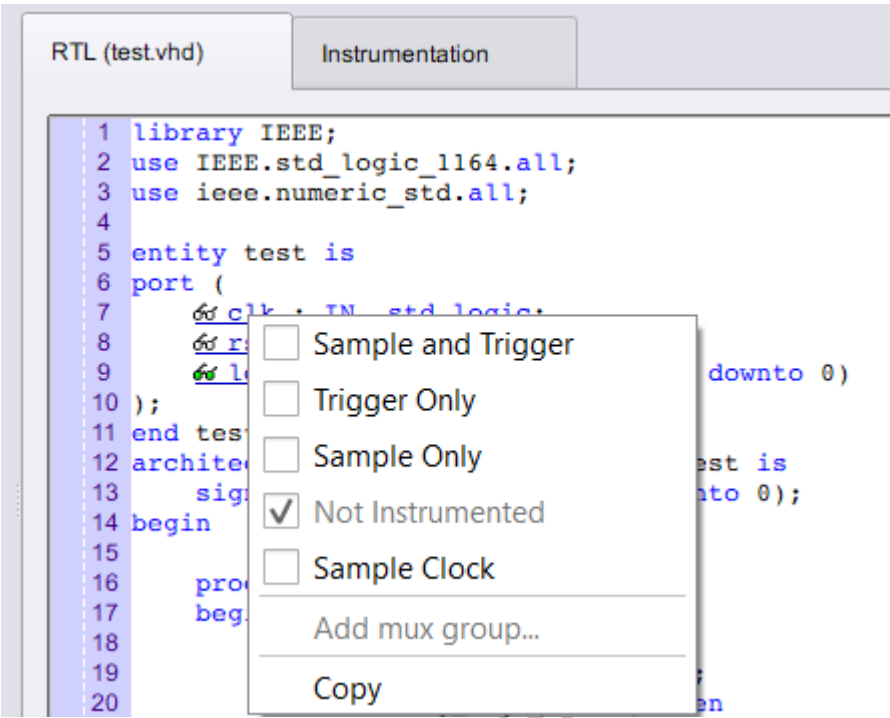


Esto hará que aparezca el reloj como referencia del conjunto a depurar (en la pestaña izquierda).



- **Selección de señales a depurar**

El siguiente paso es seleccionar las señales a depurar, para ello igual que con el reloj, nos aparecen 3 opciones: utilizar la señal como *Trigger Only*, o sea, como señal que hace que se lean las muestras en ese instante; como *Sample Only*, o sea, como señal que se va a muestrear cuando se produzca un cambio en una señal con trigger; y como *Sample and Trigger*, esta es la combinación de las otras anteriores.

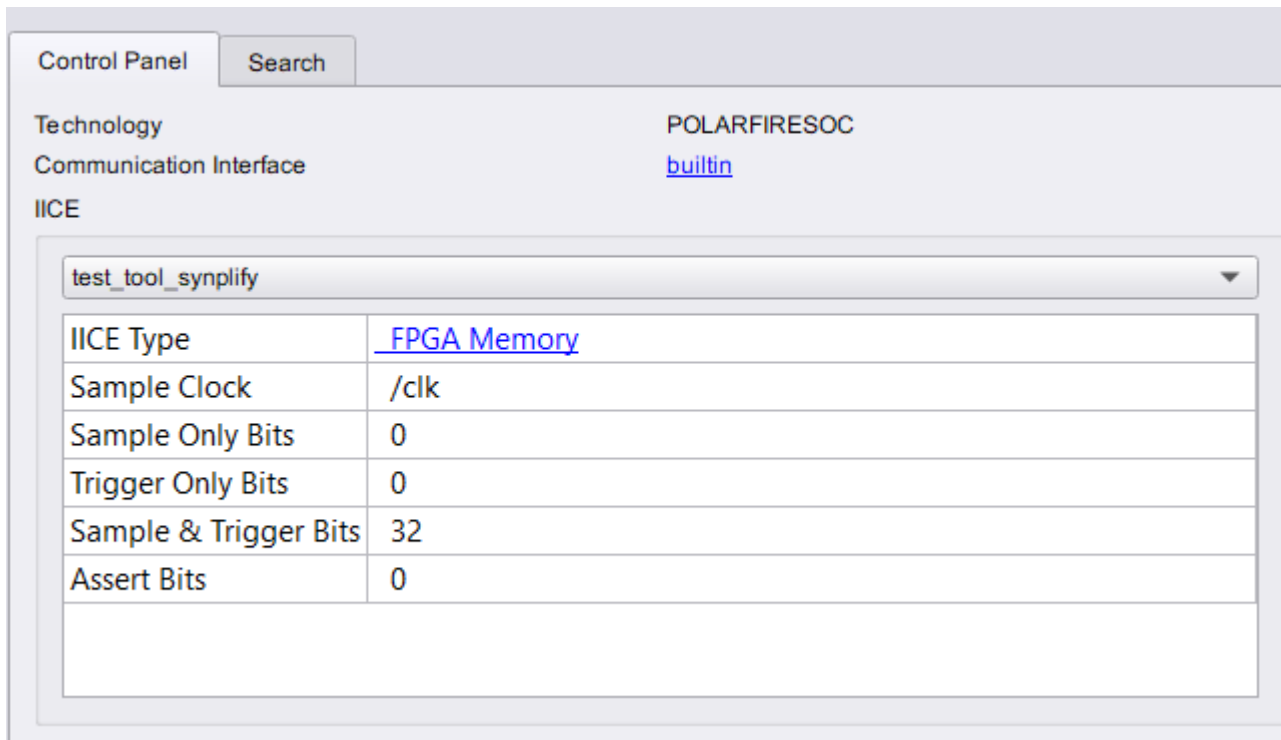


Al terminar de seleccionar señales a aparecen iconos en las diferentes señales que se van a muestrear.

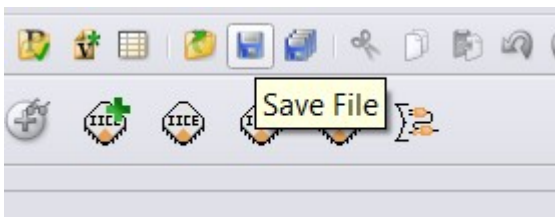
Tipo de señal	Icono
Sample Clock	Reloj
Sample and trigger	Gafas verdes
Trigger Only	Gafas rojas
Sample Only	Gafas azules
<Nada>	Gafas blancas

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

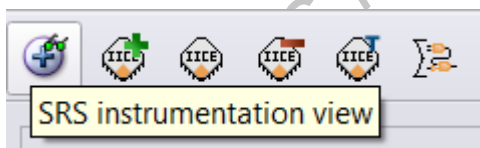
Según aumenta el número de señales que se quieren depurar, aumenta también la opción *Sample & Trigger Bits*.



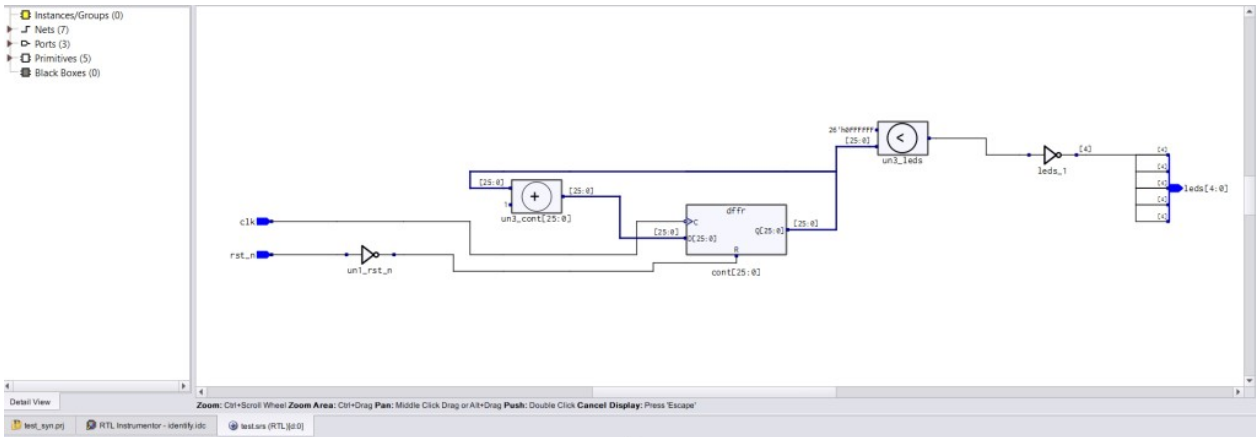
Una vez hayamos terminado de seleccionar la señales de depuración tenemos que guardar lo que hemos hecho.



NOTA: La opción *SRS instrumentation view* permite visualizar el modelo RTL del firmware diseñado.



El modelo se vería así.



El siguiente paso una vez hayamos creado el conjunto de todas las señales que queremos ver es volver a la pestaña principal

The screenshot shows the Synplify Pro Run window. The top section displays the project name `test_syn` and the device `Microchip PolarFireSoC : MPFS095T : FCVG784 : -1`. The left sidebar shows the project hierarchy with `test_syn` selected. The main area shows the `Run` status, which is `Complete`. The bottom section shows the `Run Status` table.

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

Ahora le damos a la opción de **Run** y esto nos analiza que todo lo que hayamos hecho esté correcto.

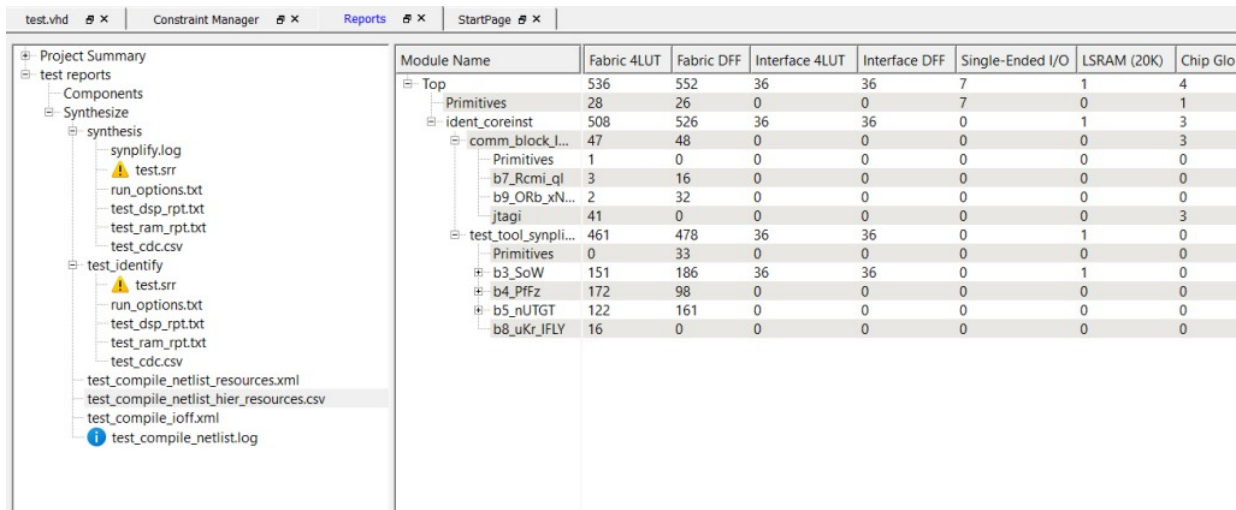
The screenshot shows the Synplify Pro Run window. The top section displays the project name `test_syn` and the device `Microchip PolarFireSoC : MPFS095T : FCVG784 : -1`. The left sidebar shows the project hierarchy with `test_syn` selected. The main area shows the `Run` status, which is `Complete`. The bottom section shows the `Run Status` table.

Job Name	Status	CPU Time	Real Time	Memory	Date/Time
Identify Database Generator (identify_db_generator)	Complete	2	0	0m 00s	09/12/2024 20:01
Identify Compile (identify_compile)	Complete	15	72	0	00m 02s
Compile Input (compiler)	Complete	15	72	0	00m 01s
Primap (xmap)	out-of-date	33	5	0m 01s	0m 05s
Map & Optimize (map_mapper)	Complete	162	43	0m 02s	0m 04s

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

Esto provocará que Libero por detrás actualice el proyecto pero con nuestros cambios. Entonces, ya podemos cerrar el *Synplicity Pro* y volver a Libero.

Si nos vamos ahora al informe de síntesis que genera Libero, a la opción, *<nombre del proyecto>_compile_netlist_hier_resources.csv*, ahora aparece un nuevo campo llamado *ident_coreinst*, donde figura todo el consumo que tiene la síntesis generada para depuración.



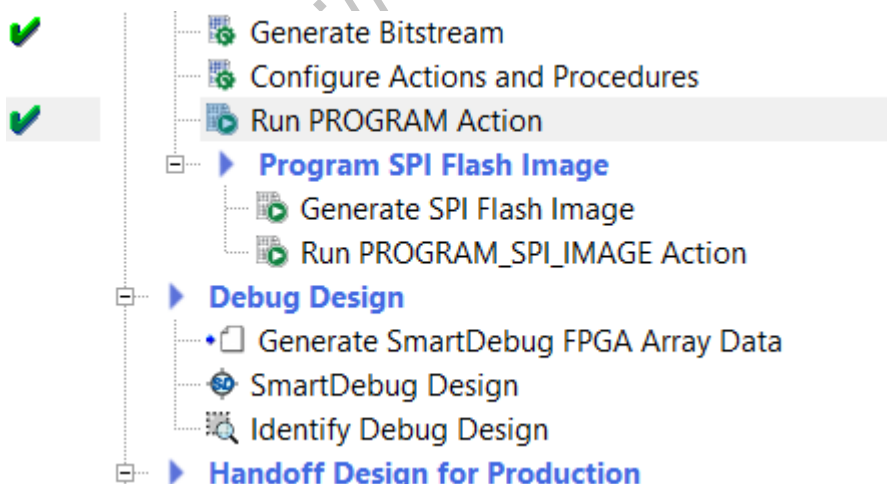
Module Name	Fabric 4LUT	Fabric DFF	Interface 4LUT	Interface DFF	Single-Ended I/O	LSRAM (20K)	Chip Glo
Top	536	552	36	36	7	1	4
Primitives	28	26	0	0	7	0	1
ident_coreinst	508	526	36	36	0	1	3
comm_block_1...	47	48	0	0	0	0	3
Primitives	1	0	0	0	0	0	0
b7_Rcmi_ql	3	16	0	0	0	0	0
b9_ORb_xN...	2	32	0	0	0	0	0
jtagi	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

Con todo esto, el siguiente paso es generar un bitstream y cargárselo a la FPGA/SoC. Recordad generar un fichero de pines al que asignar los puertos.

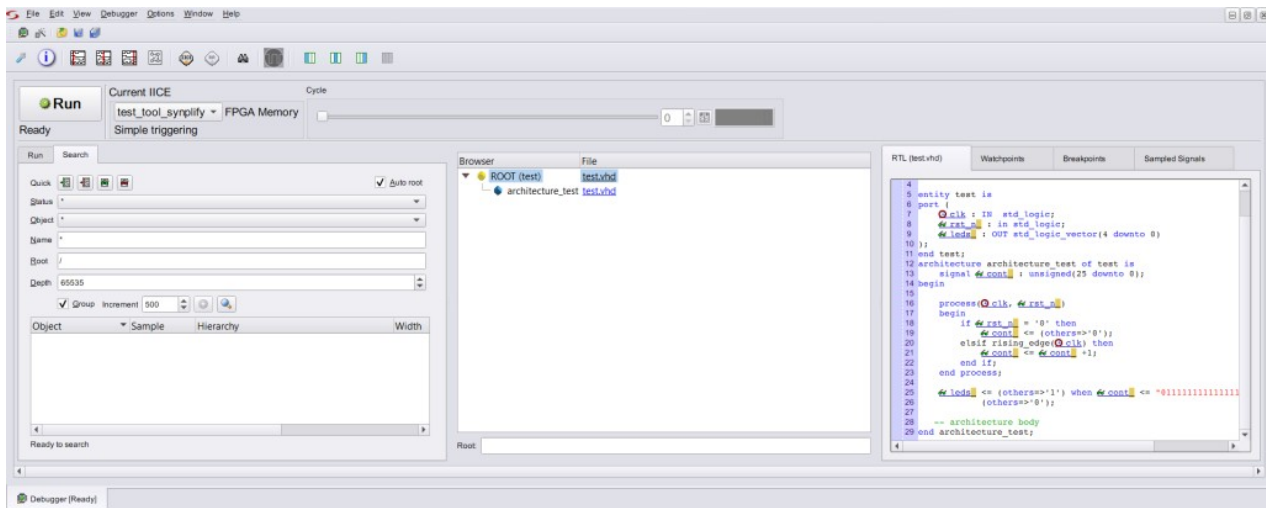
Identify Debugger

Con todos los pasos anteriores generados y un bitstream con la configuración de depuración cargada, pasamos al *Identify Debugger*.

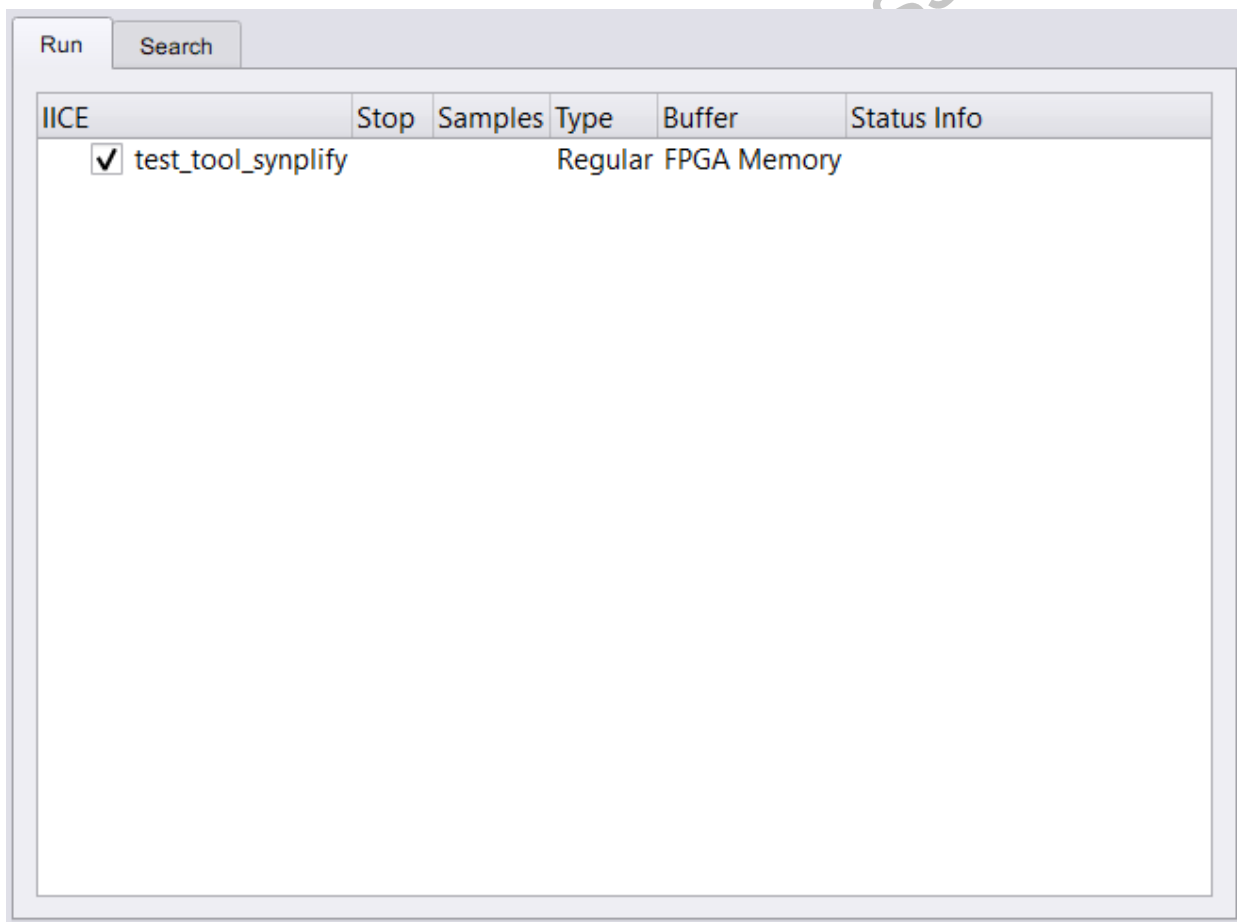
Para ello tenemos que irnos a las herramientas de depuración y seleccionar la que pone *Identify Debug Design*.



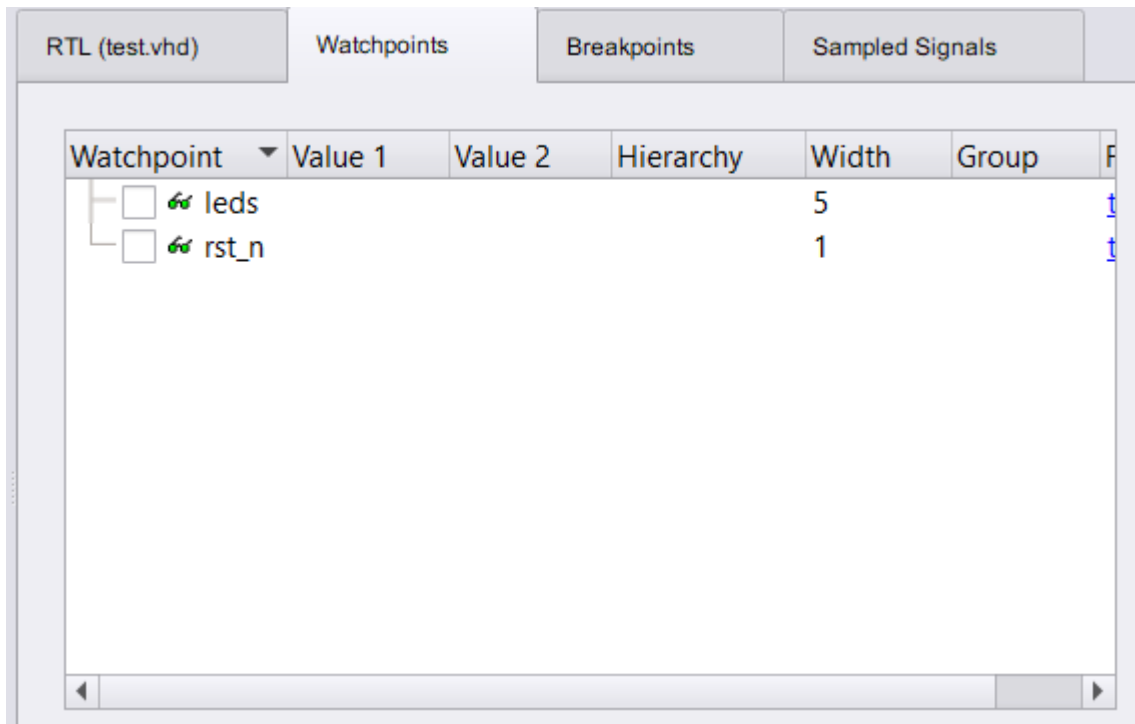
Ahora se nos abre el *Identify Debugger*. Esta herramienta lo que nos permite es visualizar tanto, aquellas señales que tenemos para depurar, como la forma de visualizarla



En la izquierda tenemos los diferentes conjuntos de señales que podemos visualizar al lanzar el depurador.

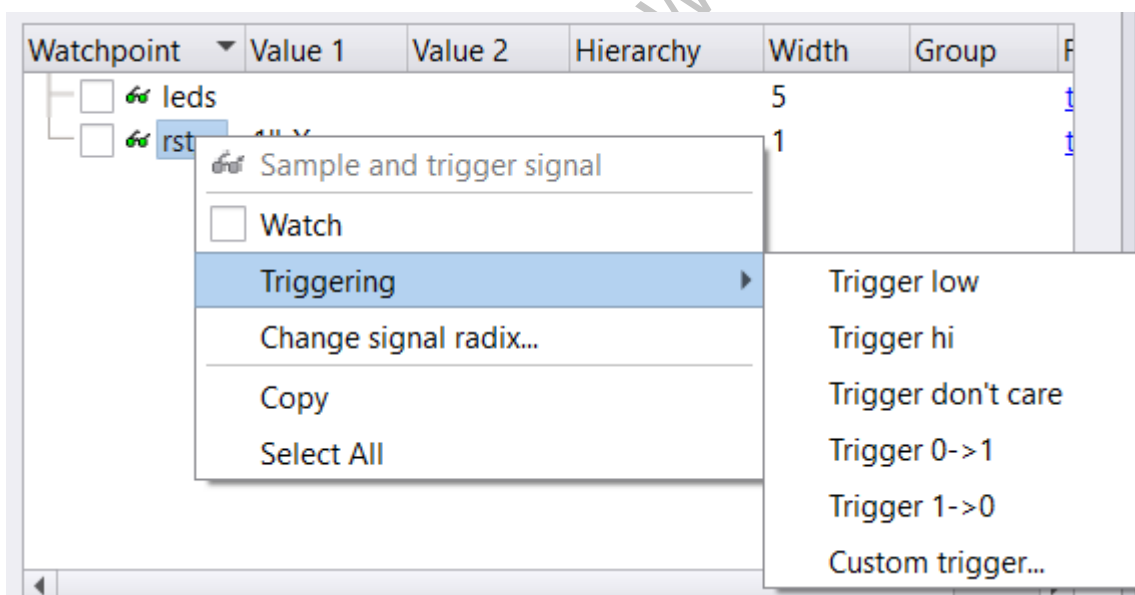


Para comenzar, tenemos que elegir cuál va a ser la condición en la que va a saltar el depurador, para ello en la pestaña de la derecha, en la opción *Watchpoint*, tenemos que elegir cuál es el disparador y con qué condición se dispara.

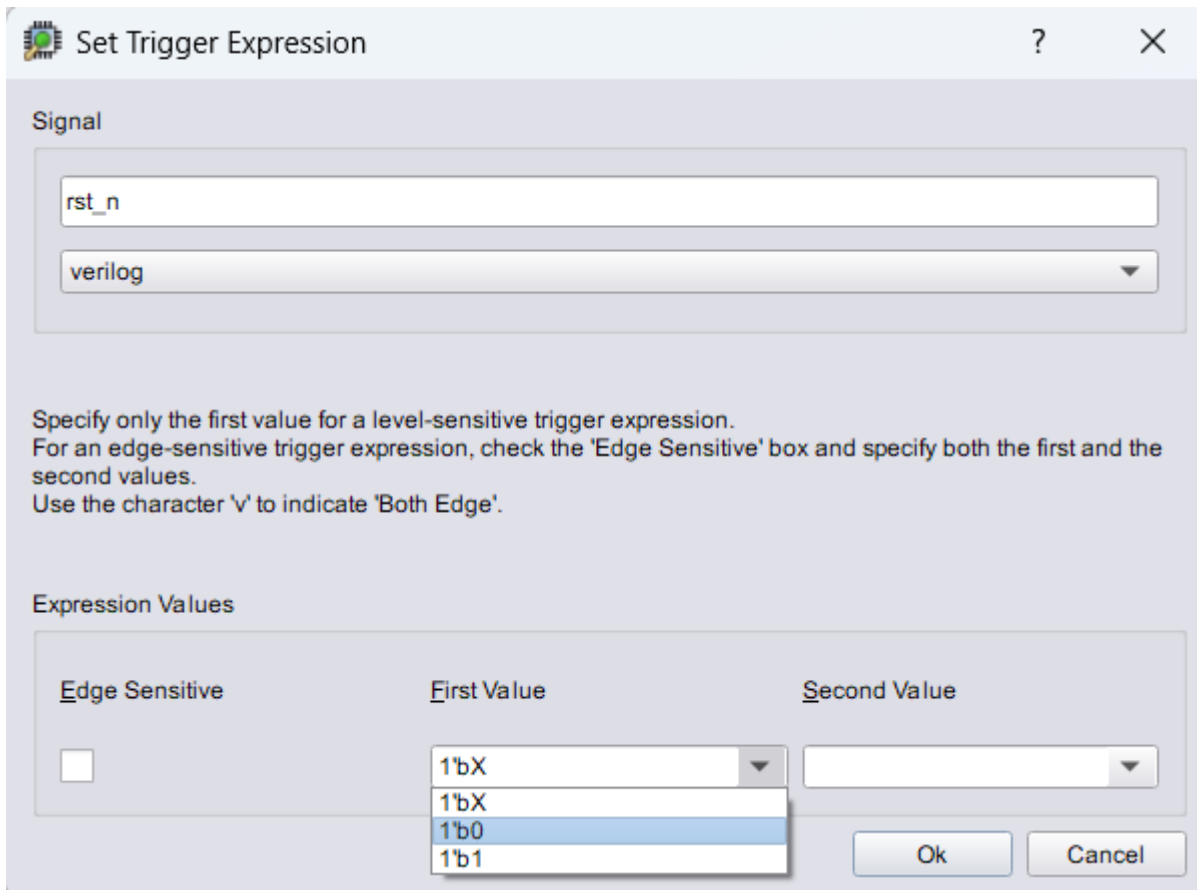


Hay dos formas de seleccionar la forma del disparo:

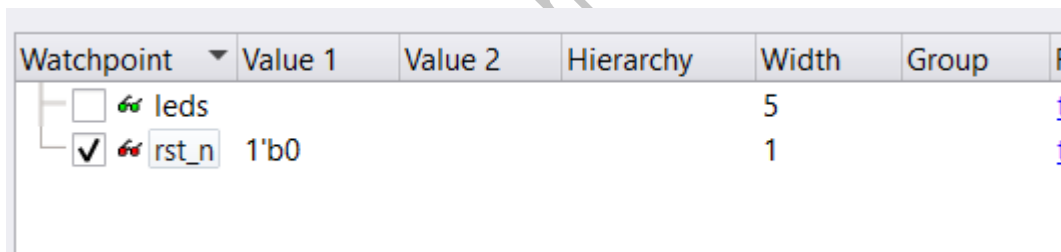
- Clic derecho en la señal y en triggering, y se selecciona una de las opciones



- Doble clic en la señal, esta opción es la misma que si en la anterior opción se hubiese elegido *Custom trigger*.
En el reglón con el lenguaje es simplemente el formato en el que se va a representar la condición de disparo.



Al terminar la selección se puede ver la condición seleccionada.

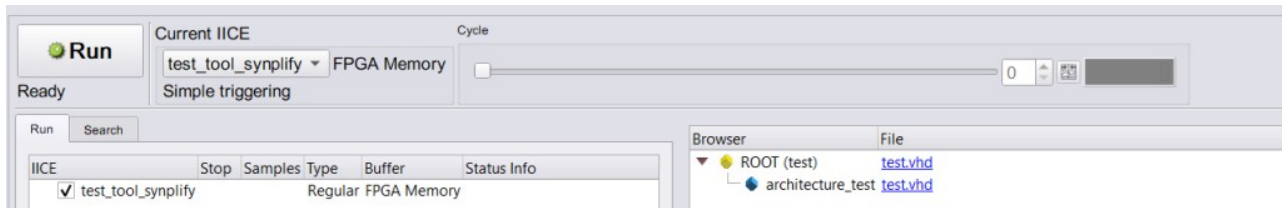


Antes de lanzar la depuración, hay varias opciones para la representación de los datos muestreados, y es que estén alineados a la izquierda, a la derecha o al centro.



Una vez seleccionado todo, se le da a *Run*.

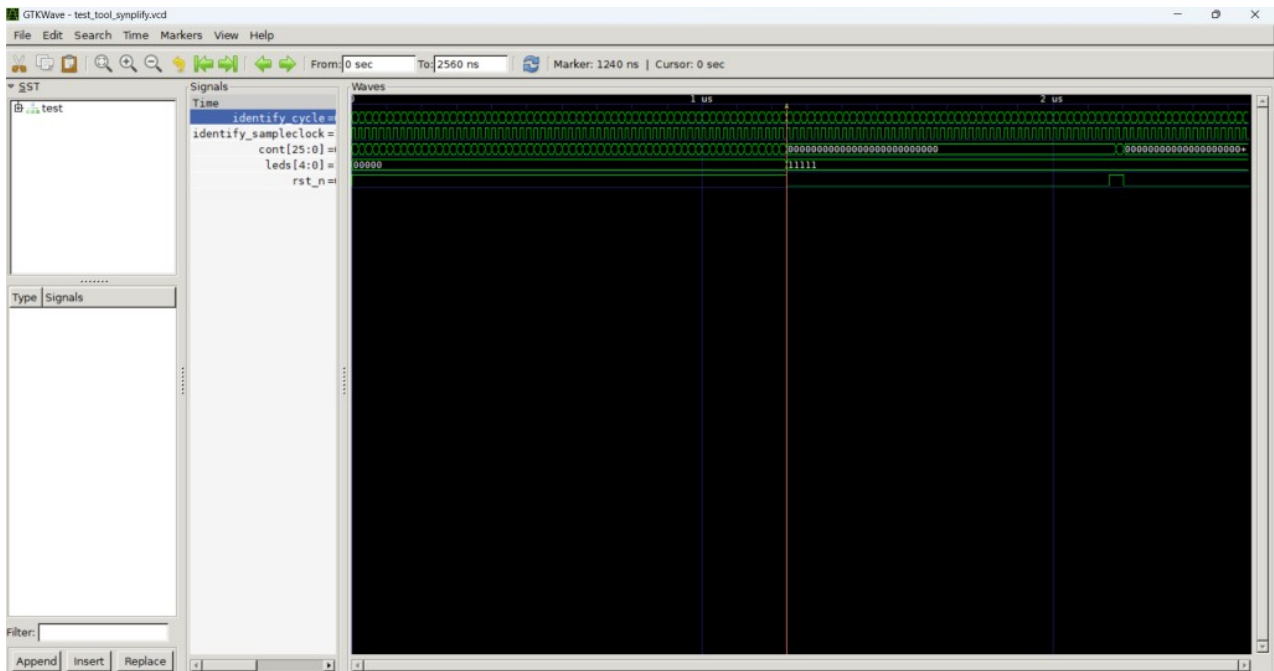
NOTA: se van a abrir tantas pestañas como conjuntos de señales se hayan elegido.



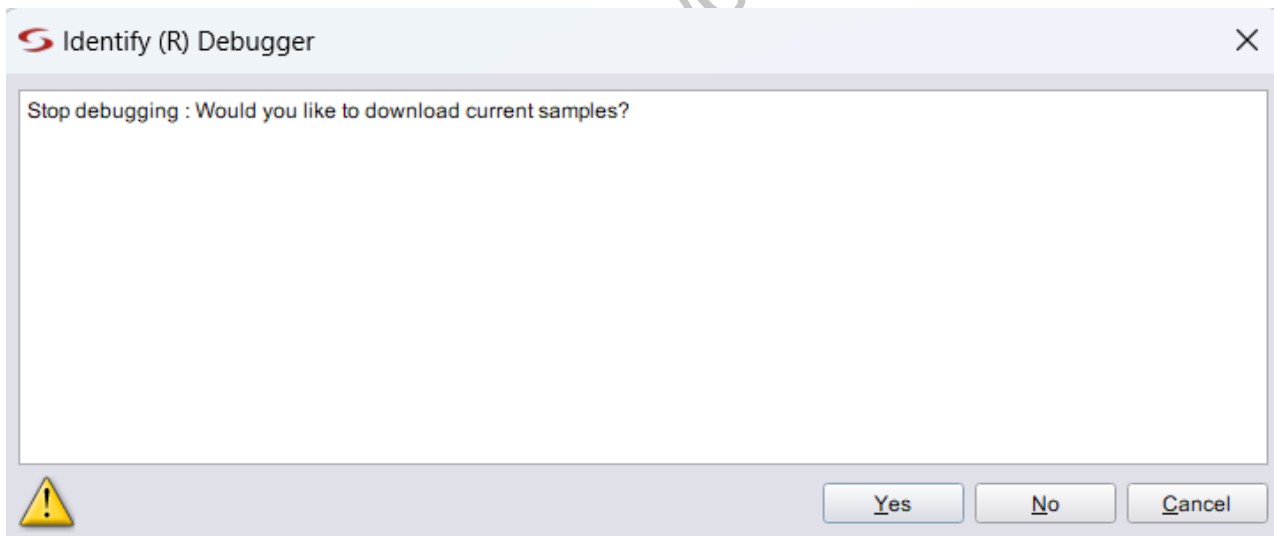
Si todo se ha hecho de forma correcta se abre una pestaña en el *GTKWave* dónde se representan las señales elegidas para depurar.



Si se mira la primera fila, es el número de muestras que se ha seleccionado en el Synplify, pero dividido a la mitad, de tal forma que se tienen -52 a la izquierda y 65 a la derecha

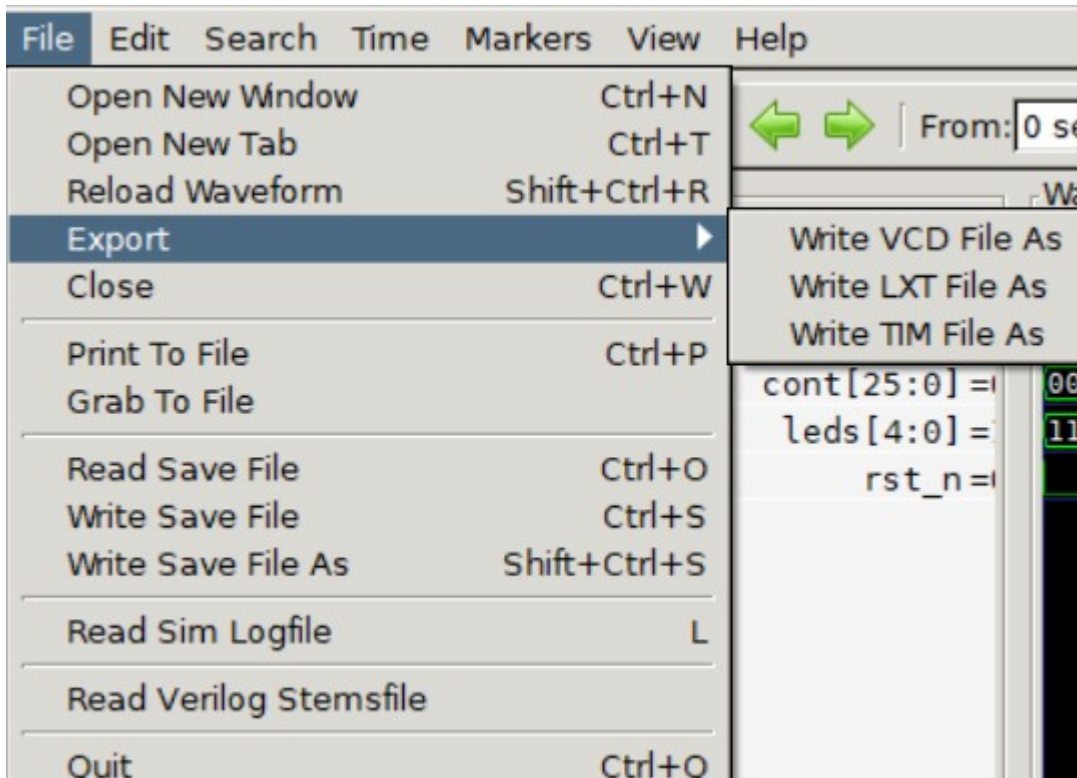


NOTA: Si decidimos parar la depuración, el debugger nos pregunta si queremos ver las últimas muestras que tiene guardadas (*aunque no se haya dado la condición de disparo*). Esto nos abre una nueva pestaña de GTKWave con las muestras.



NOTA: se van a abrir tantas pestañas como veces que se lance la depuración.

Las señales que se depuran se pueden guardar para análisis posteriores.



Por último indicar que esta barra es la que se utiliza para seleccionar en qué muestra colocar el disparador par visualizar las señales.



Referencias

<https://youtu.be/IiV25OIBLrY>