

# **Cómo crear un MicroBlaze fácil**

Creador: David Rubio G.

Entrada: <https://soceame.wordpress.com/2024/07/13/como-crear-un-microblaze-facil/>

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

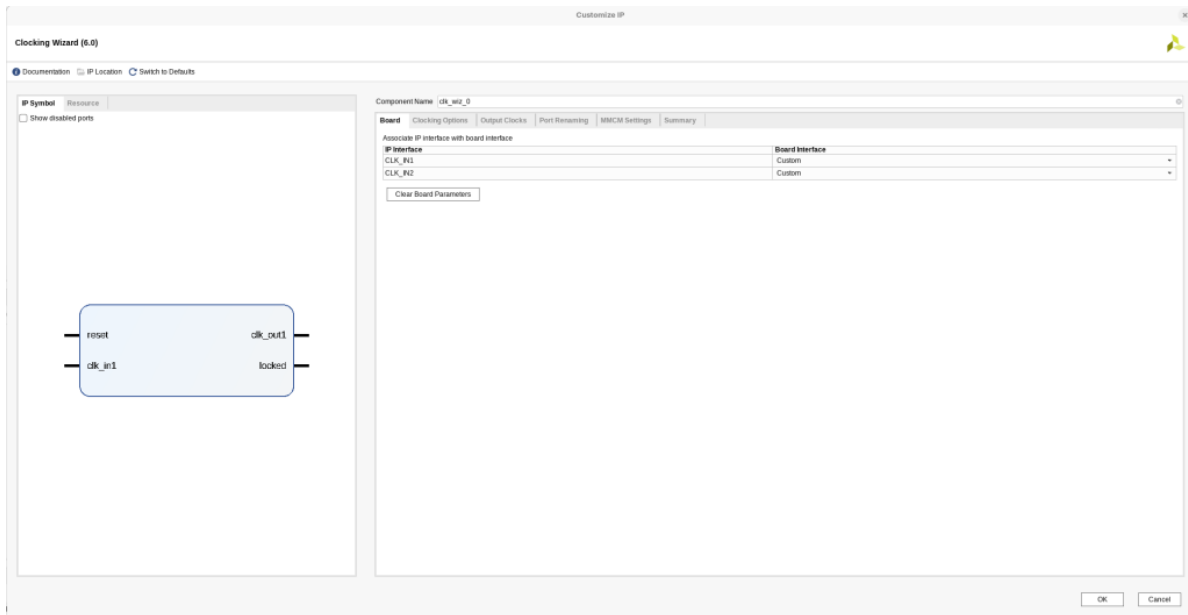
GitHub: <https://github.com/DRubioG>

Fecha última modificación: 23/02/2025

En internet hay muchas formas de crear un MicroBlaze, pero algunas son muy complejas u están obsoletas. Entonces, te explico la forma más fácil paso a paso, con algún detalle.

Lo primero es crear un *Diagram* para el MicroBlaze.

Para empezar el primer bloque que tienes que meter es el *Clocking Wizard* ajustado a la frecuencia de la FPGA que vas a utilizar, y con una frecuencia de salida de 100 o 200 MHz.

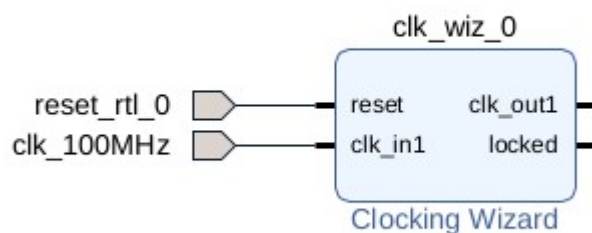


Ahora, le adjudicamos las entradas al *Clocking Wizard* como entradas exteriores.

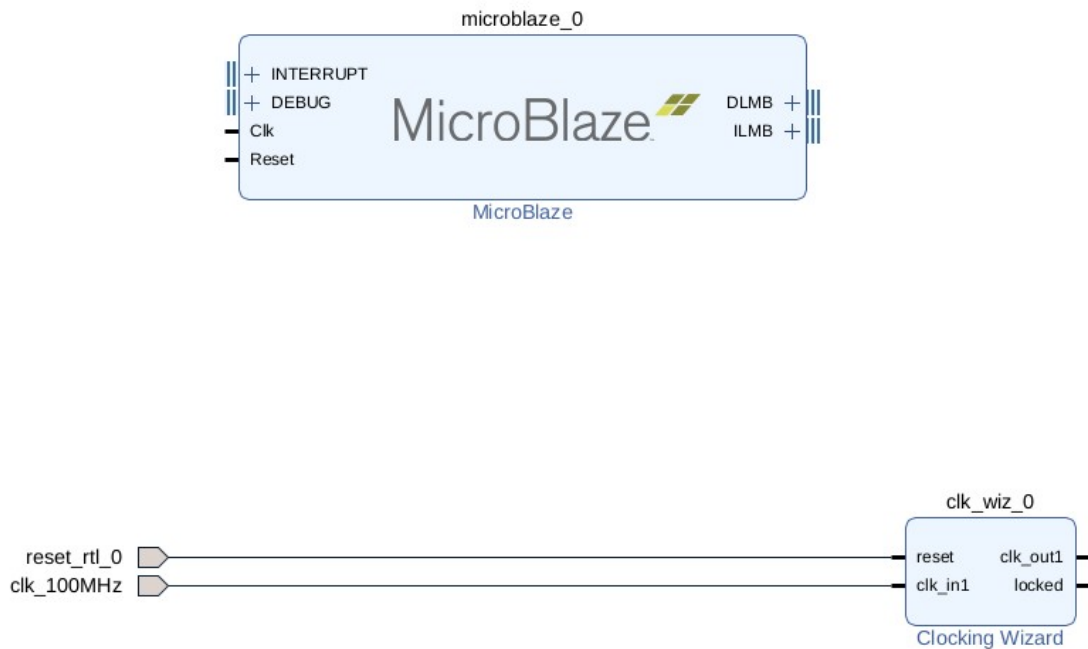
**NOTA:** si la FPGA que tienes tiene el reloj acoplado a un pin negativo (como en la imagen) de la FPGA, Vivado no te va a dejar crear un MicroBlaze.



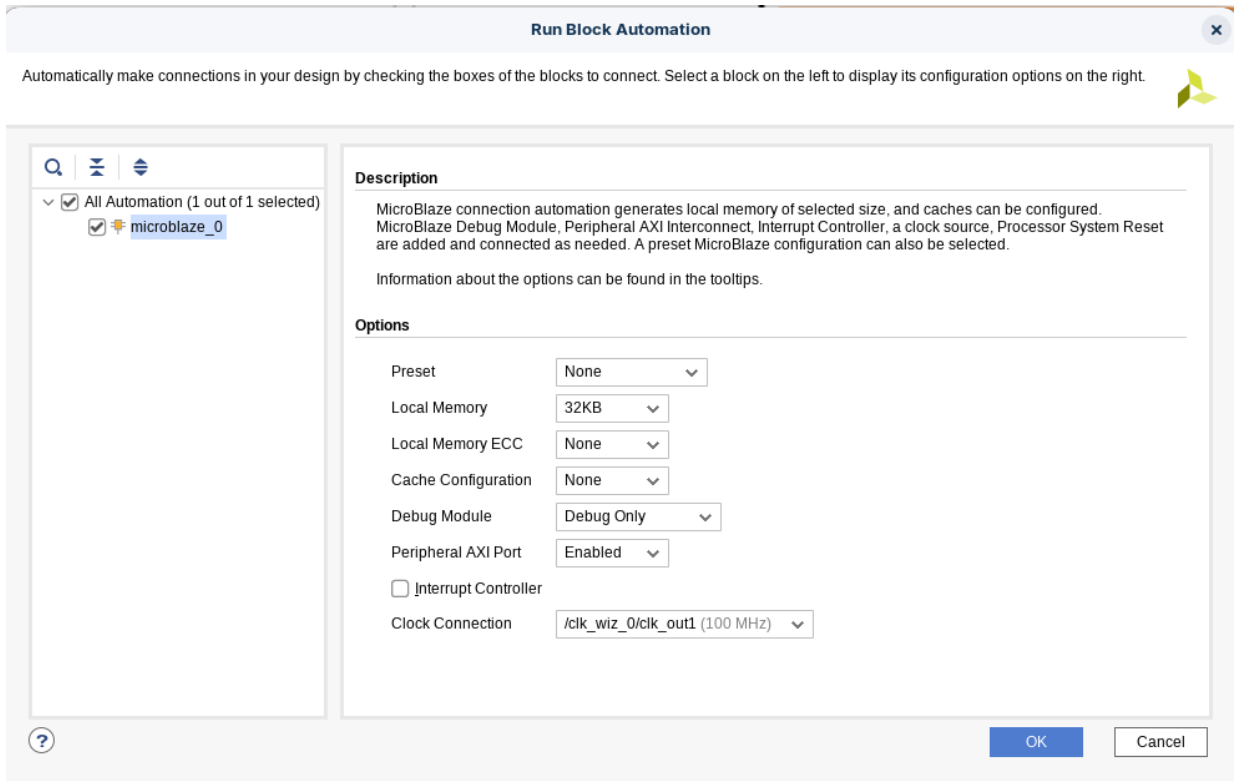
**NOTA 2:** si tu reloj es diferencial, te comento al final como trabajar con ese reloj



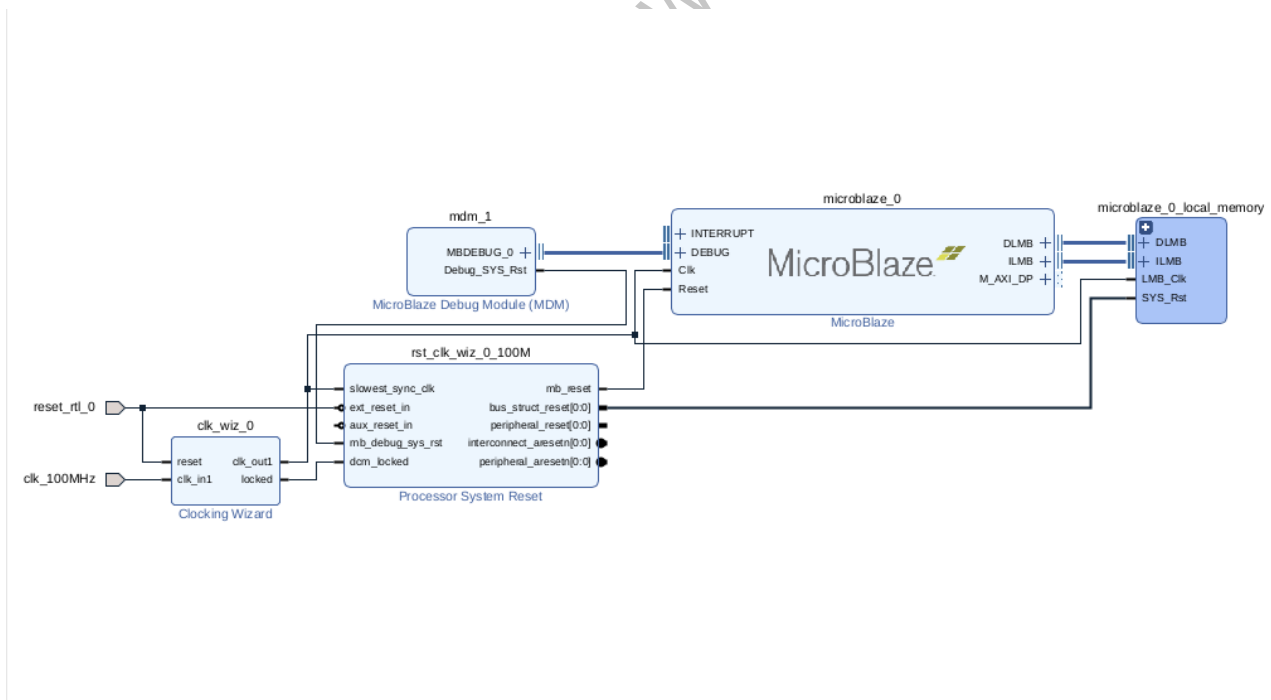
Lo siguiente es incluir el MicroBlaze, incluimos el MicroBlaze original (no el MicroBlaze MCS o el Debug Module).



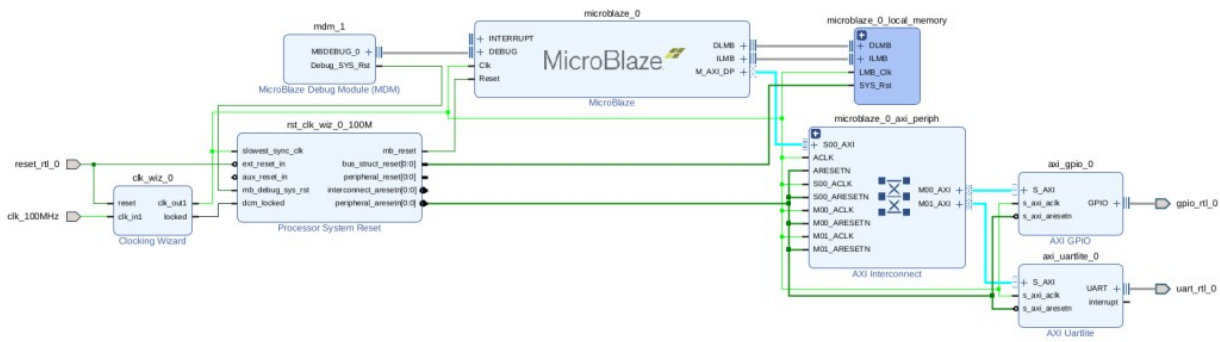
Ahora le damos al Run Block Automation para que nos conecte el Microblaze con el Clocking Wizard y otros bloques IP (en mi caso le cambio la memoria 32KB)



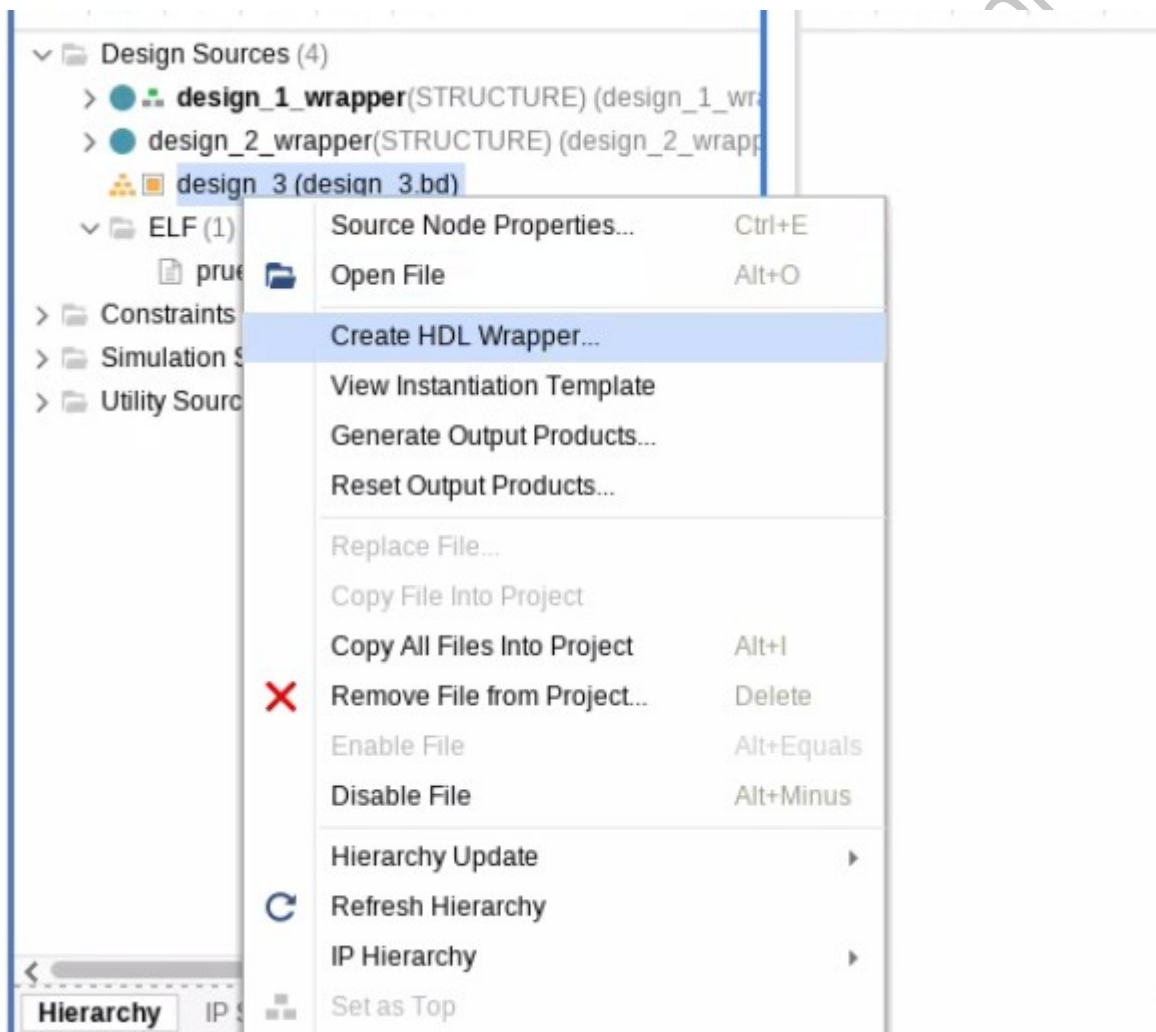
Al terminar deja un diagrama como el siguiente.



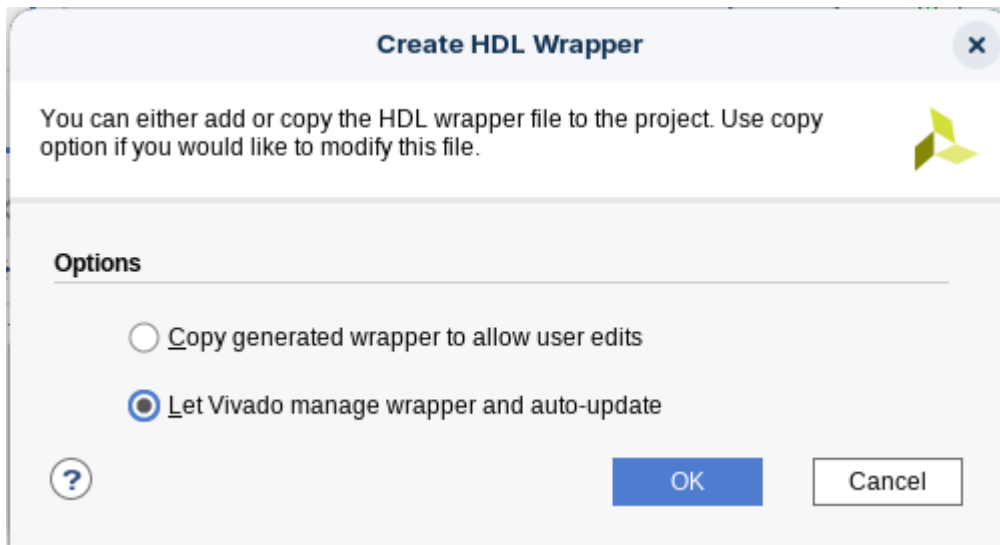
Ahora le añadimos los periféricos que queremos, solo tenemos que incluirlos y Vivado los autoconecta usando un *AXI Interconnect*.



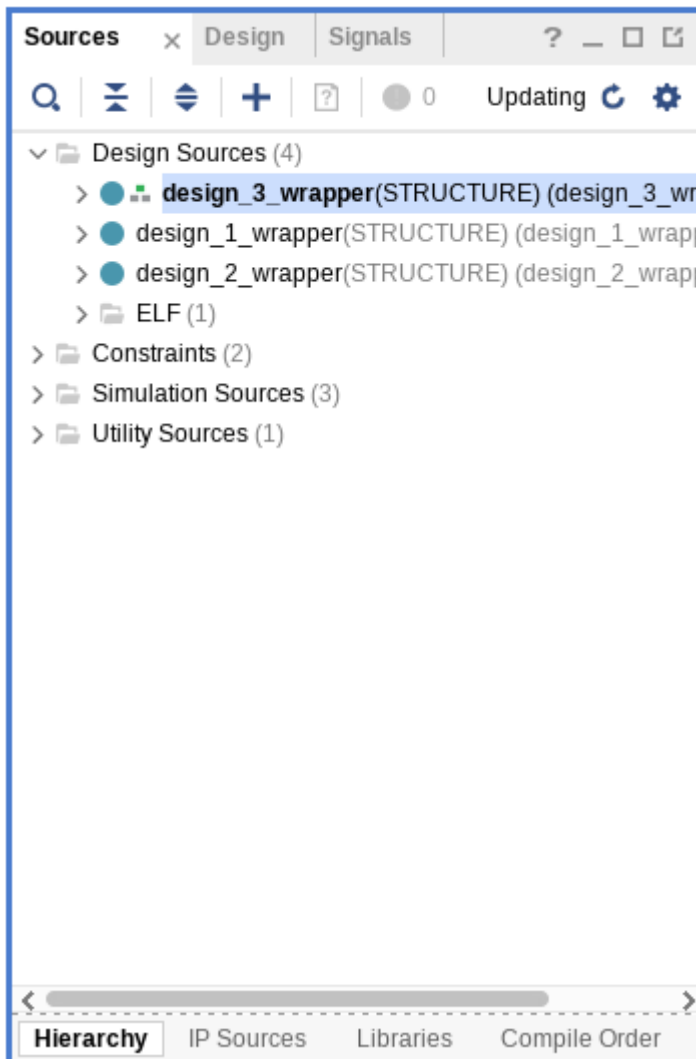
Ahora creamos un envoltorio con el MicroBlaze



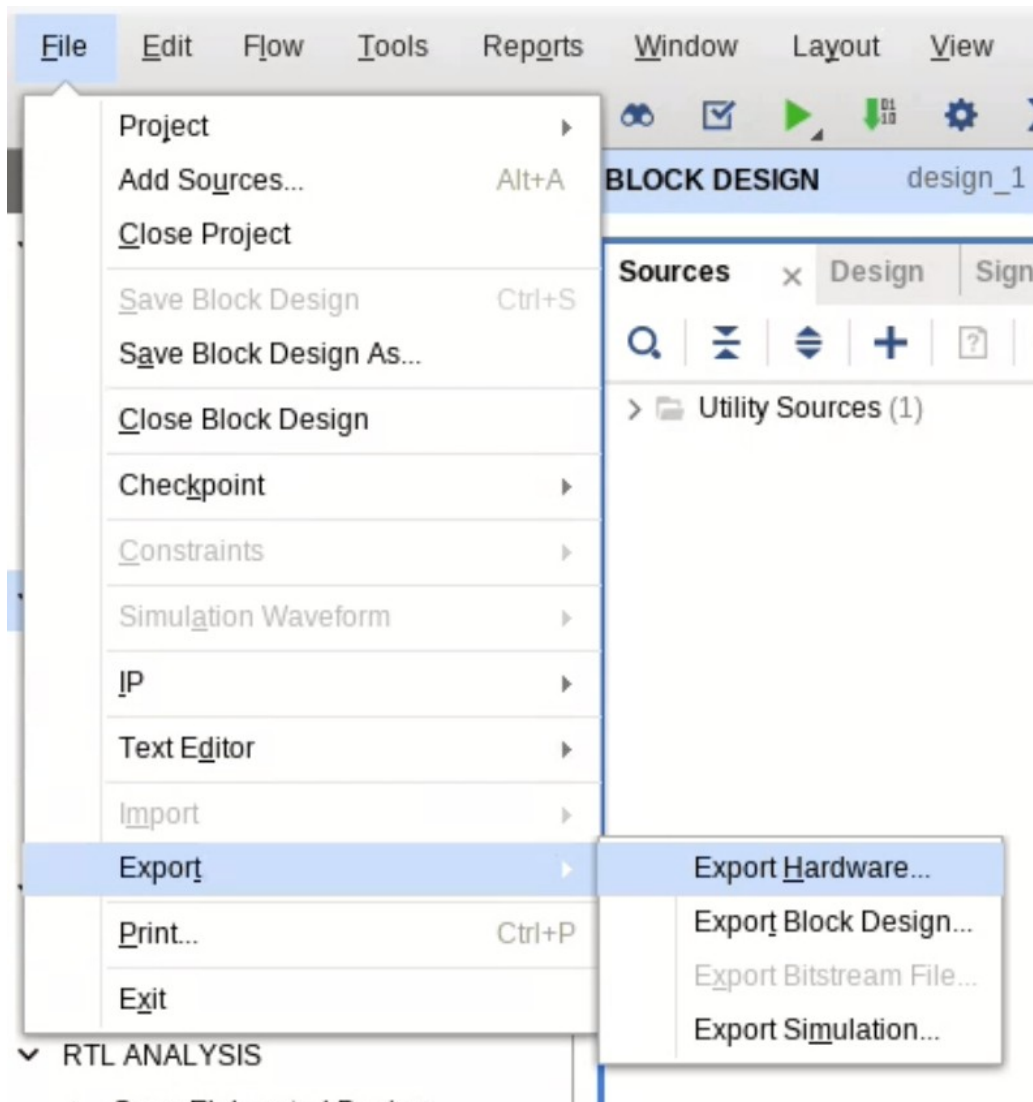
Y dejamos que Vivado lo envuelva.



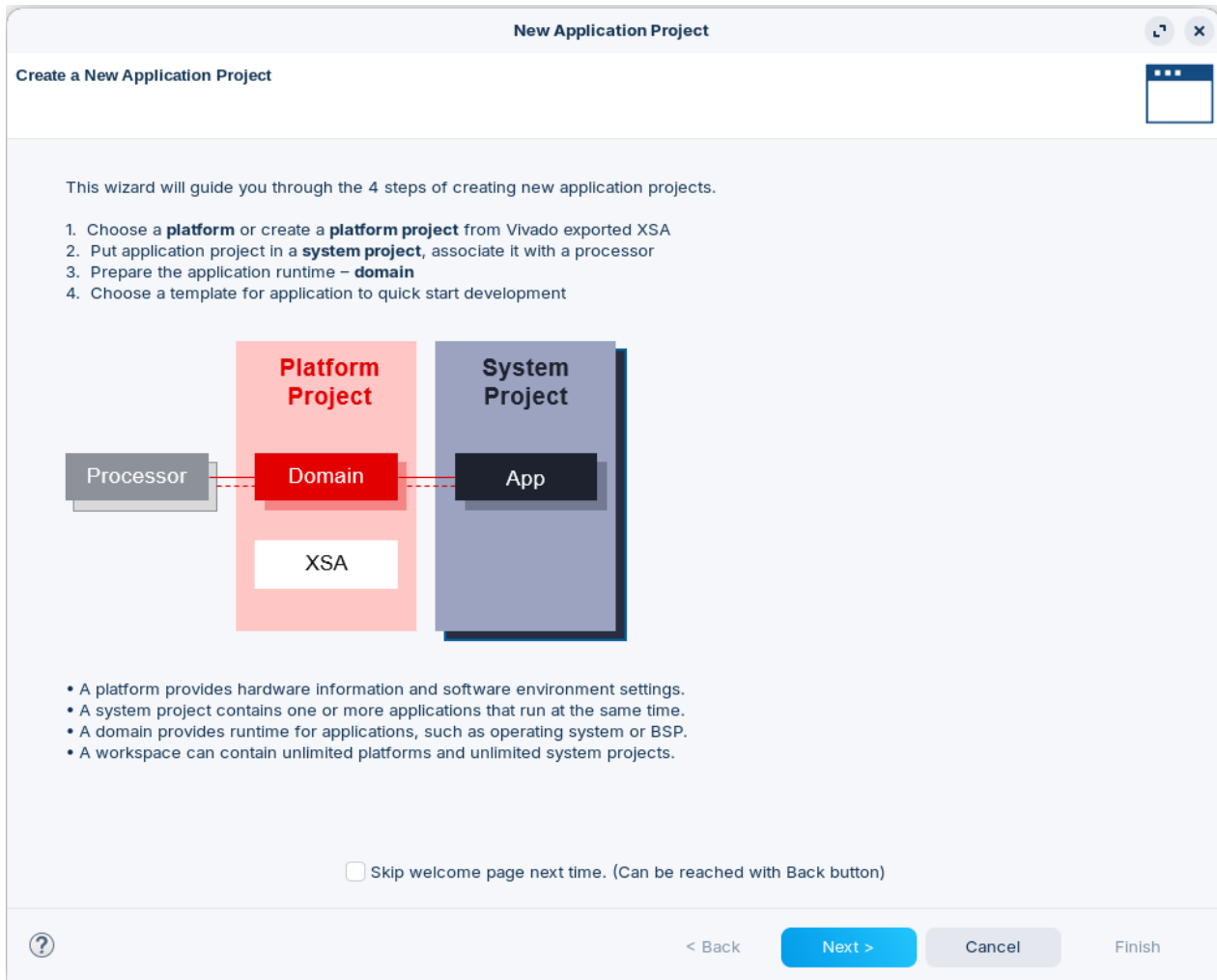
Cuando termina marcamos como Top File el wrapper que crea Vivado.



Después, solo generamos el bitstream, el bitstream tiene que tener los pines de la FPGA bien adjudicados en el XDC. Y exportamos el *Hardware*.



Después abrimos Vitis, y creamos un *Application Project*.



Para la plataforma creamos una nueva seleccionando el XSA que se ha creado al exportar el Hardware.



New Application Project

Platform

Please select a platform to create the project

Select a platform from repository Create a new platform from hardware (XSA)

Hardware Specification

XSA File: Provide your XSA file or use a pre-built board description Browse...

Platform name:

< Back Next > Cancel Finish

Después elegimos que la plataforma sea el MicroBlaze creado.

New Application Project

Application Project Details

Specify the application project name and its system project properties

Application project name: prueba\_microblaze

System Project

Create a new system project for the application or select an existing one from the workspace

Select a system project

prueba\_microblaze\_artty\_system

+ Create new...

System project details

System project name: prueba\_microblaze\_system

Target processor

Select target processor for the Application project.

Processor	Associated applications
microblaze_0	prueba_microblaze

Show all processors in the hardware specification

< Back

Next >

Cancel

Finish

El dominio lo dejamos como está

**New Application Project**

**Domain**

Select a domain for your project or create a new domain

Select the domain that the application would link to or create a new domain

Note: New domain created by this wizard will have all the requirements of the application template selected in the next step

Select a domain

standalone\_microblaze\_0

+ Create new...

**Domain details**

Name: standalone\_microblaze\_0

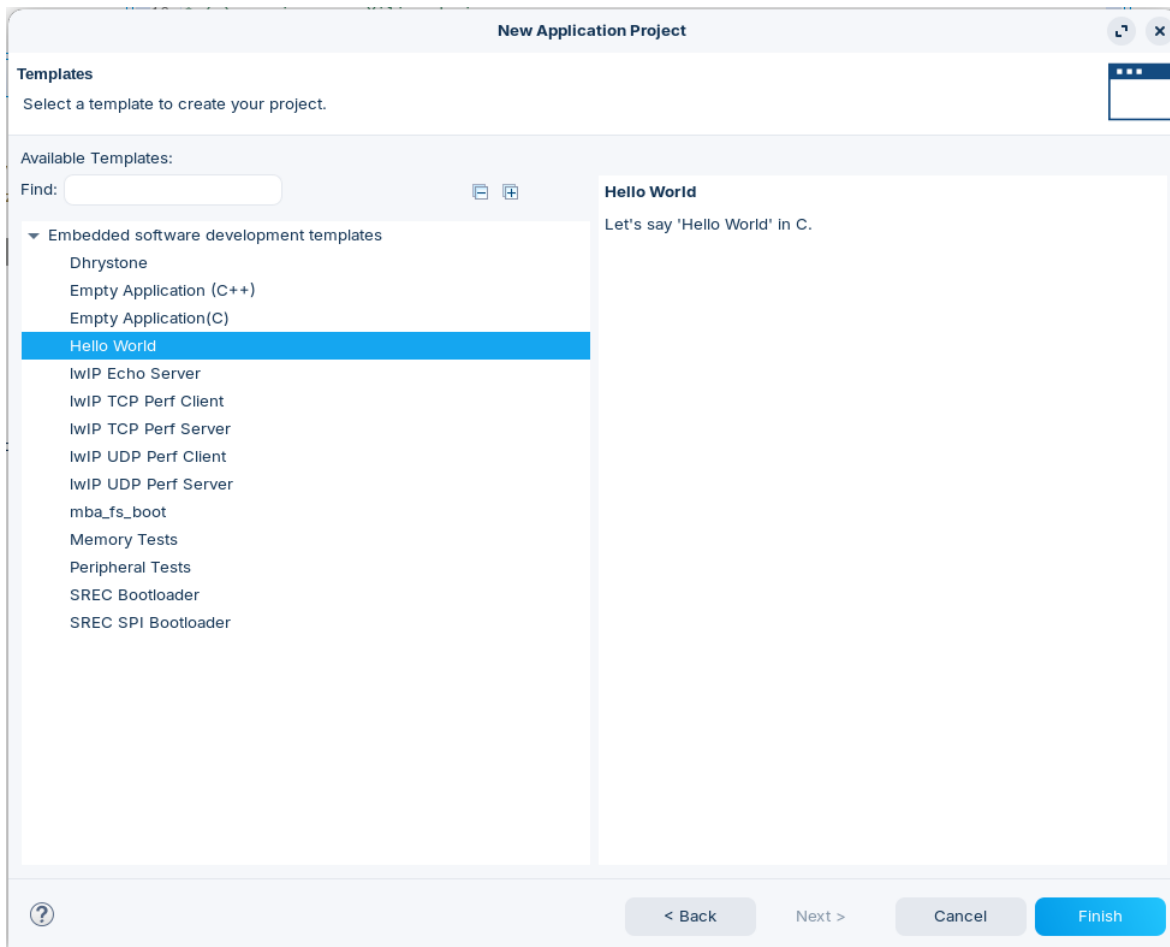
Display Name: standalone\_microblaze\_0

Operating System: standalone

Processor: microblaze\_0

? < Back Next > Cancel Finish

Y en el proyecto elegimos, por ejemplo, basarnos en un *Hello World*.



Si todo va bien y los pines de la UART están bien escogidos, se puede ejecutar la aplicación.

```
#include <stdio.h>
#include "platform.h"
#include "xil_printf.h"

int main()
{
    init_platform();

    print("Hello World\n\r");
    print("Successfully ran Hello World application");

    cleanup_platform();
    return 0;
}
```

Y en el terminal se puede ver el Hello World.

Hello World

Successfully ran Hello World application

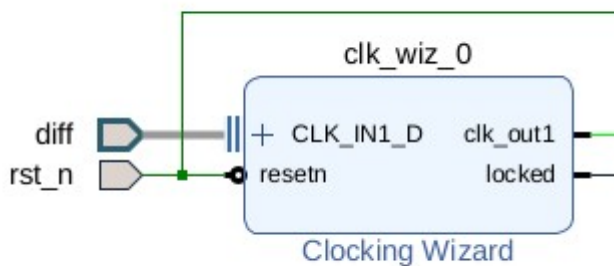
## Resumen

Esto es un resumen de los pasos anteriores.

1. **Añadir y configurar el Clocking Wizard**
2. **Añadir y configurar el MicroBlaze**
3. **Añadir periféricos y configurarlos**
4. **Generar Bitstream y llevarlo a Vitis**
5. **Crear aplicación en Vitis**

## Pines de reloj diferenciales

Las placas más potentes suelen llevar pines diferenciales de reloj, para ello el Clocking Wizard tiene que tener una entrada para pines diferenciales.



Esto se configura en la pestaña *Clocking Options*, en *Input Clock Information*, en la opción *Source*. Se selecciona la opción **Differential clock capable pin**.

Bien, y ahora para los pines del XDC se tiene que configurar de la siguiente forma.

Si el reloj yo le he llamado (en el Diagram) **diff**, en el XDC le voy a declarar como diferencial, entonces solo tengo que declarar el **pin positivo**, porque Vivado rellena el negativo.

Entonces, a este pin **diff** para el diferencial positivo de reloj le tengo que llamar **diff\_clk\_p** en el XDC.

**NOTA:** se le añade el sufijo «\_clk\_p»

```
create_clock -period 5.000 [get_ports diff_clk_p]
set_property PACKAGE_PIN AK17 [get_ports diff_clk_p]
set_property IOSTANDARD DIFF_SSTL12 [get_ports diff_clk_p]
```

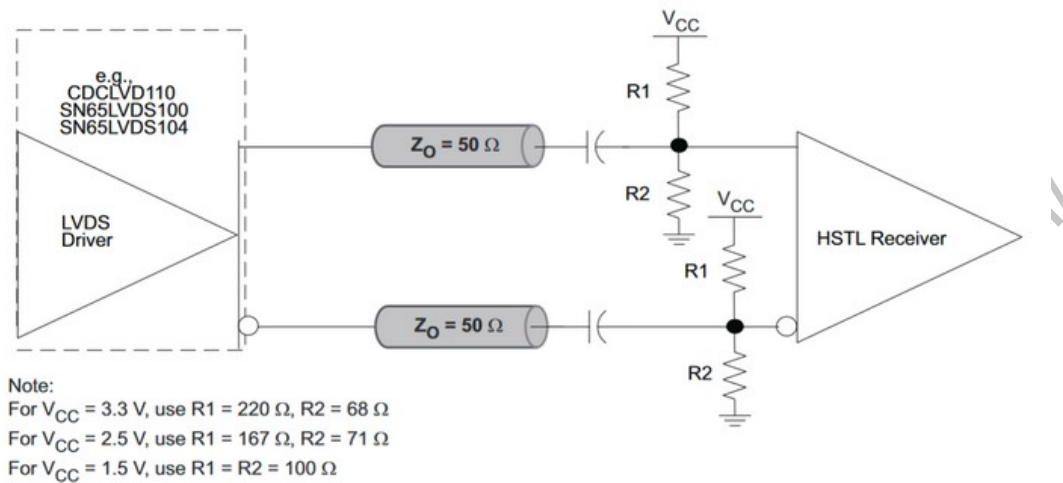
Si se comprueba como Vivado entiende los pines, se puede ver que Vivado autorrellena la columna **Neg Diff Pair** llamando a este pin «diff\_clk\_n».

Name	Direction	Neg Diff Pair	Package Pin	Fixed	Bank	I/O Std	Vcco	Vref	Drive Strength	Slew Type	Pull Type	Off-Chip Termination
▼ All ports (5)												
> RST_RST_N_54720 (1)	IN			✓	64	LVC MOS18	1.800				NONE	NONE
> uart_r0_54720 (2)	(Multiple)			✓	65	LVC MOS18	1.800				NONE	(Multiple)
▼ Scalar ports (2)												
diff_clk_p	IN	diff_clk_n	AK17	✓	45	DIFF_SSTL12					NONE	NONE

Eso es porque el pin positivo del reloj va al pin positivo **IO\_L12P**, y el negativo del reloj va al negativo **IO\_L12N**.

IO_L11P_T1U_N8_GC_45	AK16	PL_CLK0_N
IO_L12N_T1U_N11_GC_45	AK17	PL_CLK0_P
IO_L12P_T1U_N10_GC_45	AH17	

**NOTA:** el uso del *DIFF\_SSTL12* es por la tecnología con la que el fabricante de la PCB ha usado para acoplar el reloj con la FPGA, y el 12 es porque la VCC de la línea se ha usado 1,2V.



**Figure 15. LVDS to HSTL**

En mi caso, el ejemplo para utilizar el *DIFF\_SSTL12* es:

