

# ¿Qué es el Dynamic Function eXchange?

Creador: David Rubio G.

Entrada: <https://soceame.wordpress.com/2024/10/20/que-es-el-dynamic-function-exchange/>

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

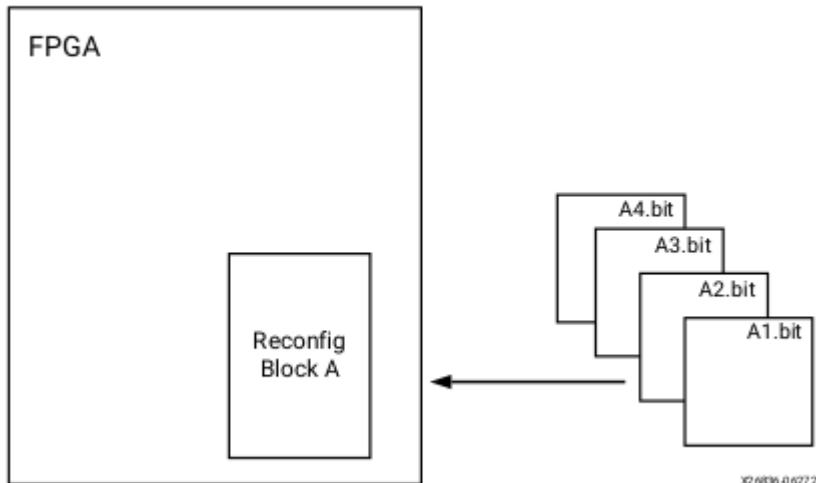
GitHub: <https://github.com/DRubioG>

Fecha última modificación: 23/02/2025

El Dynamic Function eXchange (DFX) es una cosa muy poco conocida de las FPGAs de Xilinx, que básicamente lo que permite es grabar un bitstream en la FPGA mientras otro bitstream está cargado y en funcionamiento, y que este nuevo bitstream no altere al bitstream ya en funcionamiento.

Entonces, permite crear bitstreams parciales, además, de los bitstreams totales.

**Figure 1: Basic Premise of Dynamic Function eXchange**



Se utiliza sobre todo para sustituir bitstreams que estén en funcionamiento dentro de la FPGA por otros, para poder acelerar el desarrollo.

El proceso consta de 4 pasos

- Paso 1: Crear una partición
- Paso 2: Crear una partición con un módulo distinto a los declarados en el código base
- Paso 3: Seleccionar las zonas donde se generarán los bitstream
- Paso 4: Generar los bitstreams

## ¿Cómo se hace?

La mejor forma de explicar cómo se hace funcionar el Dynamic Function eXchange es con un ejemplo.

Este ejemplo son 6 leds, de los que se van a dividir en dos bloques de 3 leds cada uno. Y se van a cargar dos ficheros vhdl, uno para que los leds vayan a derechas y otro para que vayan a izquierdas. Ambos sobre el fichero top\_led.

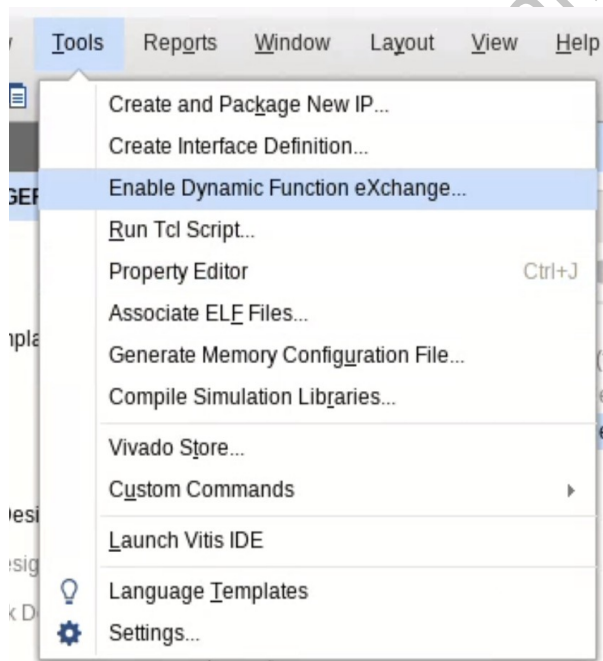
Para ello primero creamos el proyecto y utilizamos el mismo fichero de funcionamiento para los leds. Este primer fichero sólo instancia los módulos que hacen que los leds vayan hacia la derecha, por lo que por defecto los leds van hacia la derecha.



```
1  library IEEE;
2  use IEEE.STD_LOGIC_1164.ALL;
3
4  entity top_led is
5      Port (
6          clk : in std_logic;
7          rst_n : in std_logic;
8          led_1 : out std_logic_vector(2 downto 0);
9          led_2 : out std_logic_vector(2 downto 0)
10     );
11 end top_led;
12
13 architecture Behavioral of top_led is
14     component leds is
15         Port (
16             clk : in std_logic;
17             rst_n : in std_logic;
18             led : out std_logic_vector(2 downto 0)
19         );
20     end component;
21
22     begin
23
24
25     part_a : leds
26         Port map (
27             clk => clk,
28             rst_n => rst_n,
29             led => led_1
30         );
31
32     part_b : leds
33         Port map (
34             clk => clk,
35             rst_n => rst_n,
36             led => led_2
37         );
38
39 end Behavioral;
```

## PASO 1: Crear una partición

Una tenemos el proyecto ya diseñado, nos vamos a la pestaña **Tools** y pulsamos en la opción «*Enable Dynamic Function eXchange*».

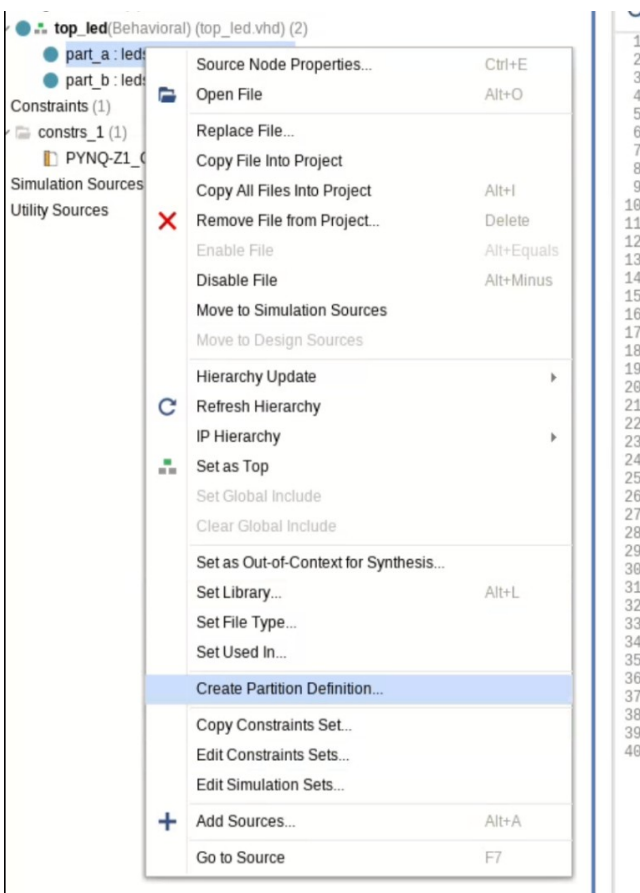


Al pulsarla salta este recuadro, hay que marcar la opción de *Convert*.



(Ahora en Tools ha cambiado el nombre de la opción y ahora se llama *Dynamic Function eXchange Wizard* [que no va a funcionar hasta el siguiente paso])

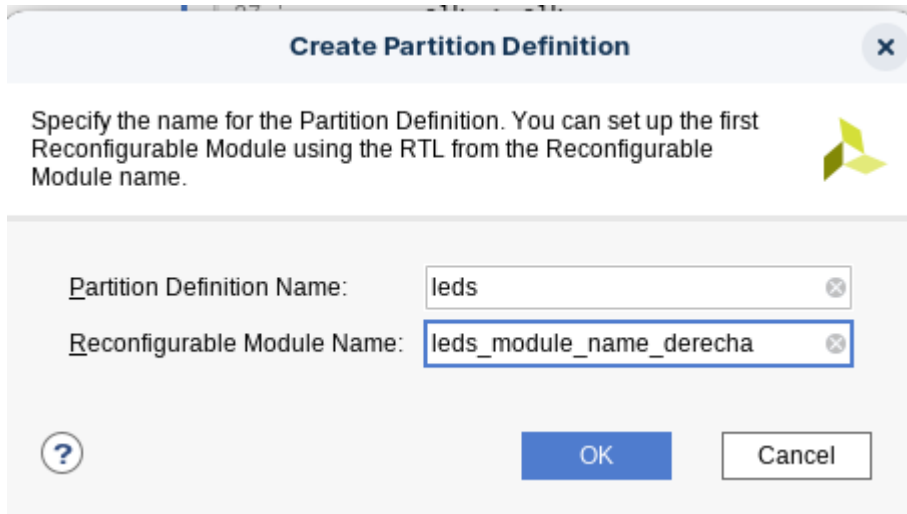
El siguiente paso es hacer clic derecho sobre el fichero hijo del top. Ahora aparece la opción «*Create Partition Definition*»



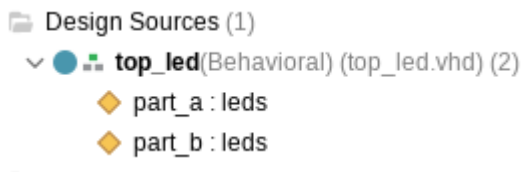
Al pulsarle aparece una pestaña, en ella

- el **Partition Definition Name** es el *nombre de la partición* que se quiere crear
- y el **Reconfigurable Module Name** es el *nombre que va a tener el fichero* que has seleccionado dentro de la partición creada.

**NOTA:** antes de pulsar piensa bien si tienes todos los ficheros ya desarrollados y correctos porque desde este momento es muy posible que no lo puedas seguir desarrollando desde este proyecto (y lo tengas que desarrollar desde fuera o generar este proyecto desde cero).



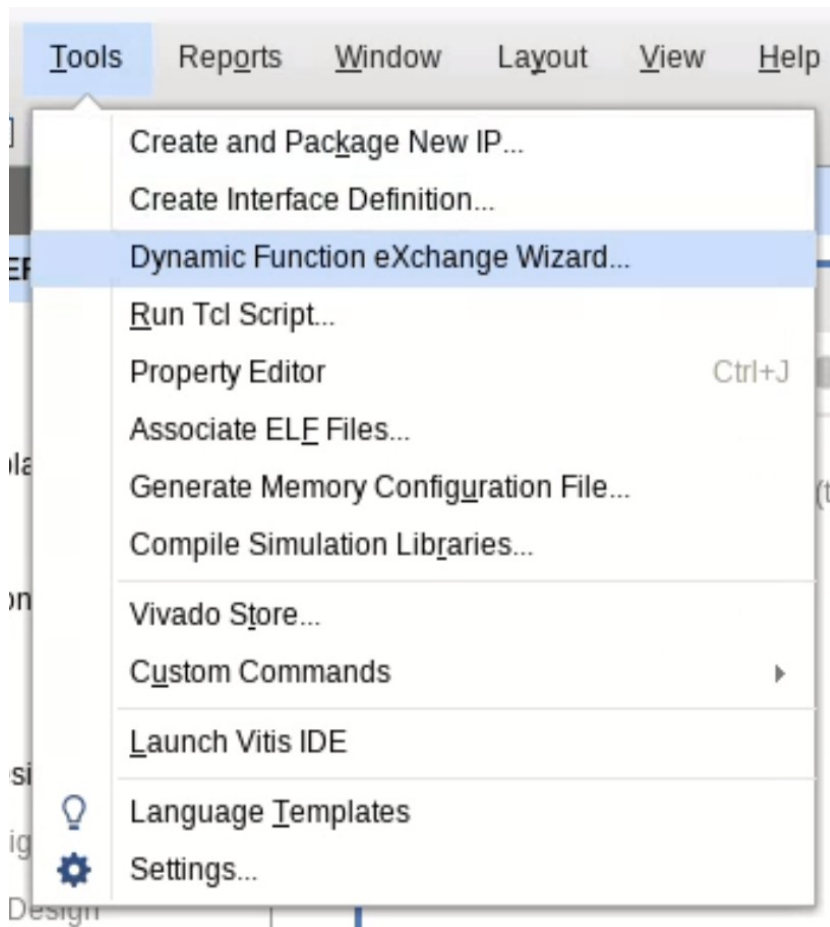
Ahora el fichero ahora ha cambiado y aparece con un cuadro naranja (que no permite volver acceder al fichero)



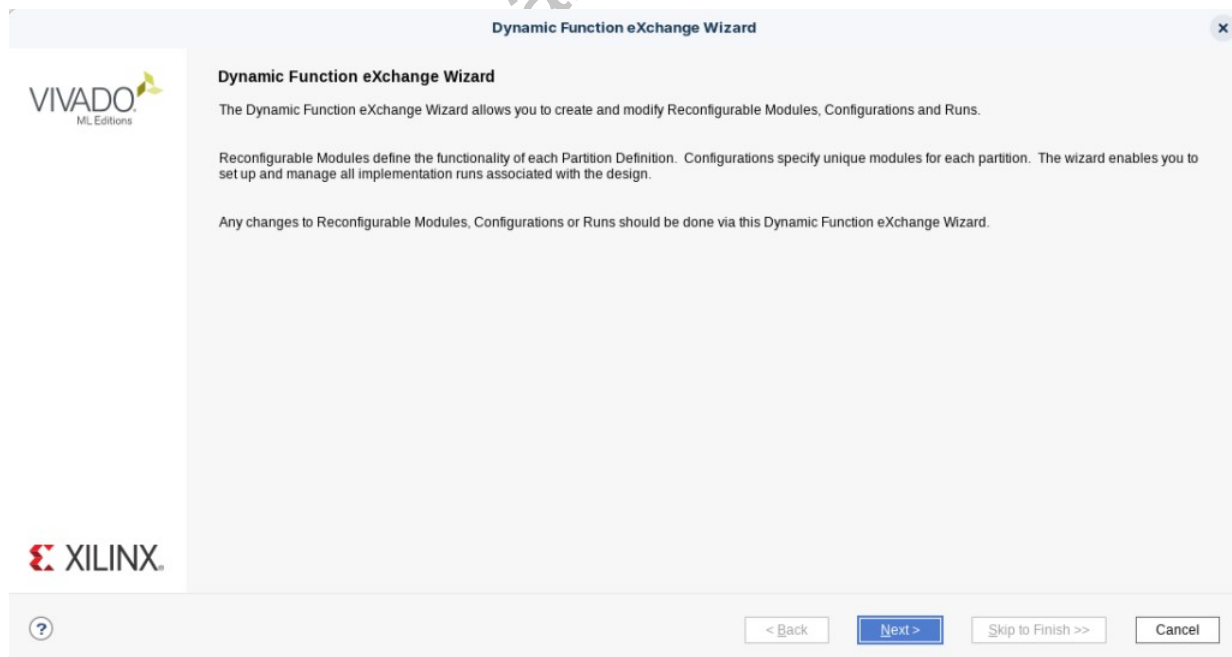
## PASO 2: Crear una partición con un módulo distinto

Con esto ya tenemos la primera parte, que es tener una partición que hace que los leds se desplacen a la derecha, ahora vamos a crear la partición que hace que los leds vayan a la izquierda. Esto significa que se va a modificar las referencias del fichero top, sin modificar el fichero top.

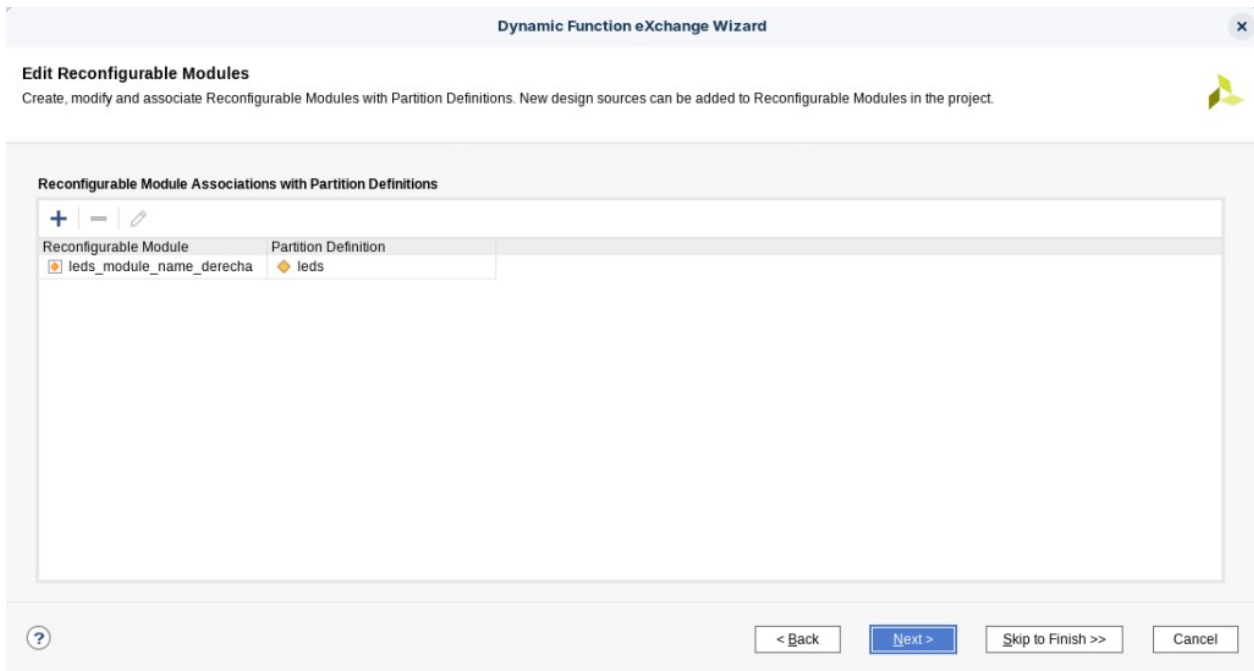
Para ello, otra vez en **Tools**, pulsamos en **Dynamic Function eXchange Wizard**.



Ahora se abre la pestaña.



En la primera pestaña aparece la partición primera que hemos creado antes.



Ahora creamos la nueva partición para que vaya a izquierdas.

Para ello pulsamos en el '+' y se nos abre una pestaña, donde nos permite elegir *la partición sobre la que queremos trabajar*, un *nuevo top module name*, que sirve para nombrar el fichero top al que la partición (en este caso leds) solo se le aplique esta nueva funcionalidad (esto último se entenderá mejor cuando lleguemos al final). Y luego lo que nos pide es un nombre del *Reconfigurable Module Name* y un fichero.

**Add Reconfigurable Module**

Add sources to a Reconfigurable Module. Choose which Partition Definition the module should be associated with and specify a name for the module. A Top Module name is required if the module hierarchy is ambiguous within the set of source files.

Partition Definition Name: **leds**

Top Module Name:

Reconfigurable Module Name:

☐ Sources are already synthesized

Use Add Files or Add Directories buttons below

Add Files

Add Directories

☒ Scan and add RTL include files into project

☐ Copy sources into project

☒ Add sources from subdirectories

?

OK

Cancel

Debido a que el fichero anterior lo he llamado *leds\_module\_name\_derecha*, pues ahora lo llamaré *leds\_module\_name\_izquierda* y le doy el módulo en VHDL que hace que los leds se desplacen a izquierdas.



**Add Reconfigurable Module**

Add sources to a Reconfigurable Module. Choose which Partition Definition the module should be associated with and specify a name for the module. A Top Module name is required if the module hierarchy is ambiguous within the set of source files.

Partition Definition Name:  Top Module Name:

Reconfigurable Module Name:  ☐ Sources are already synthesized

	Index	Name	Library
<input checked="" type="radio"/>	1	leds_izda.vhd	xil_defaultlib

☐ Scan and add RTL include files into project  
☐ Copy sources into project  
☒ Add sources from subdirectories

Ahora cuando le damos a OK, nos devuelve a la pestaña anterior, que queda de la siguiente forma.

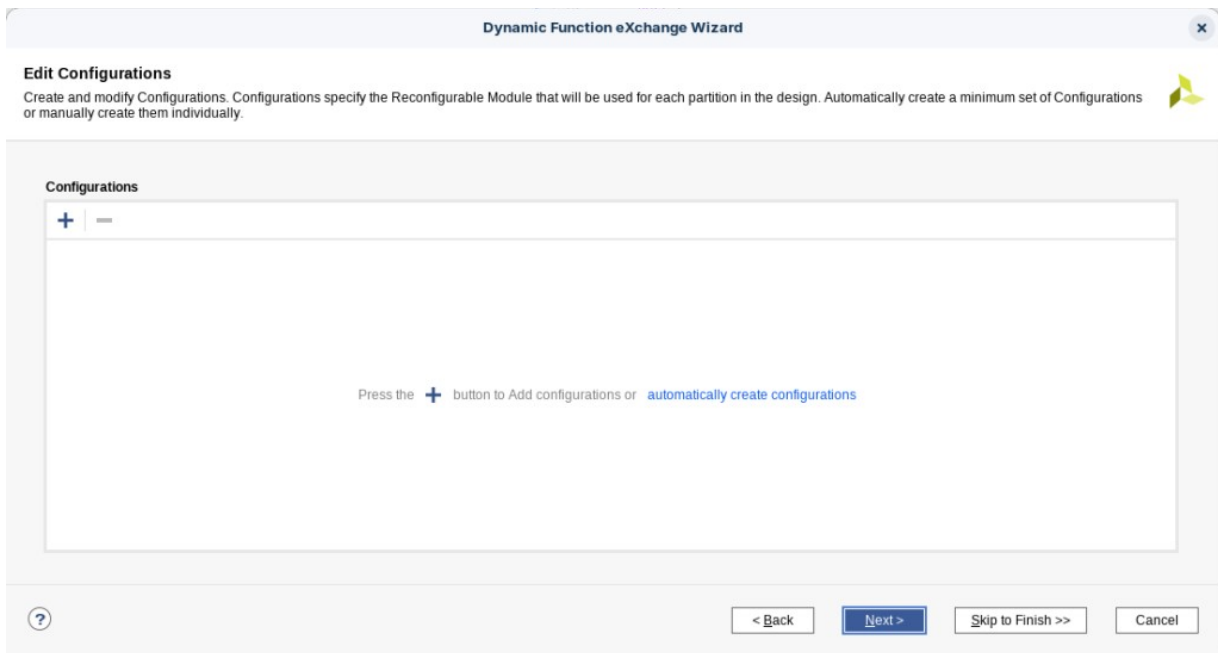
**Dynamic Function eXchange Wizard**

**Edit Reconfigurable Modules**  
Create, modify and associate Reconfigurable Modules with Partition Definitions. New design sources can be added to Reconfigurable Modules in the project.

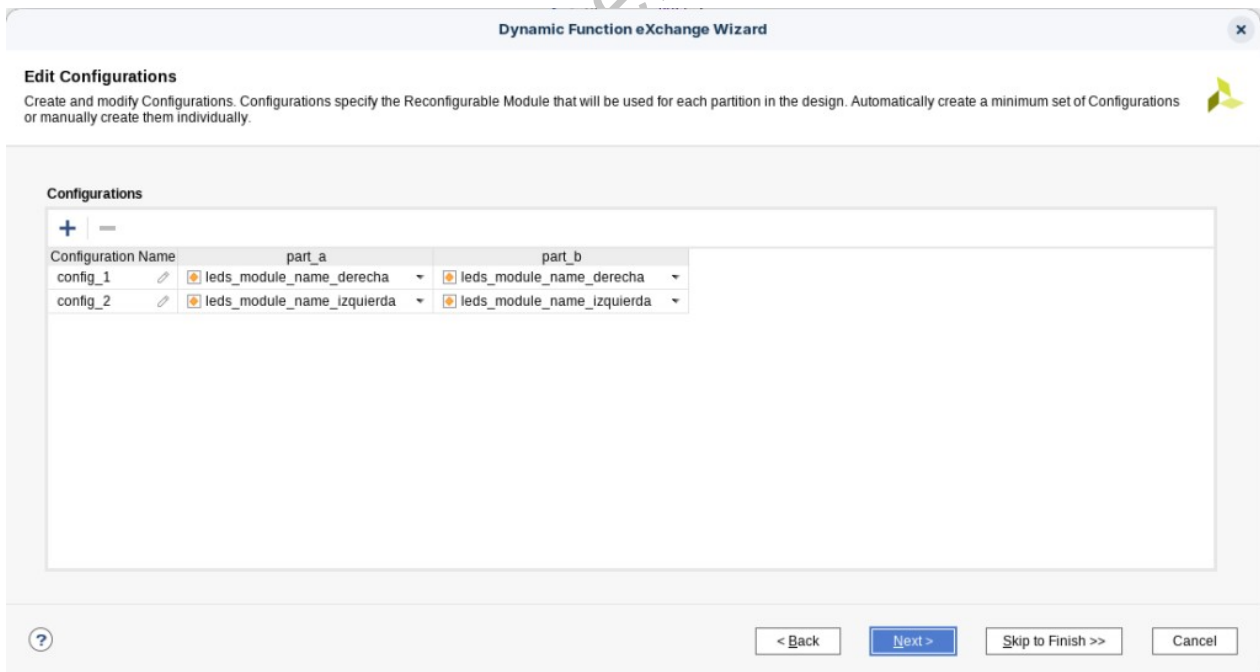
**Reconfigurable Module Associations with Partition Definitions**

Reconfigurable Module	Partition Definition
<input checked="" type="checkbox"/> leds_module_name_derecha	<input checked="" type="checkbox"/> leds
<input checked="" type="checkbox"/> leds_module_name_izquierda	<input checked="" type="checkbox"/> leds

Pasamos a la siguiente pestaña, que aparece de primeras vacía. Esta pestaña es para crear configuraciones.



Para facilitar el trabajo le damos a la opción *automatically create configurations* que nos crea dos configuraciones, una por cada partición con las dos opciones que hay. Esto sirve para que cuando se genere el bitstream completo lleve las particiones en el orden elegido por el usuario.



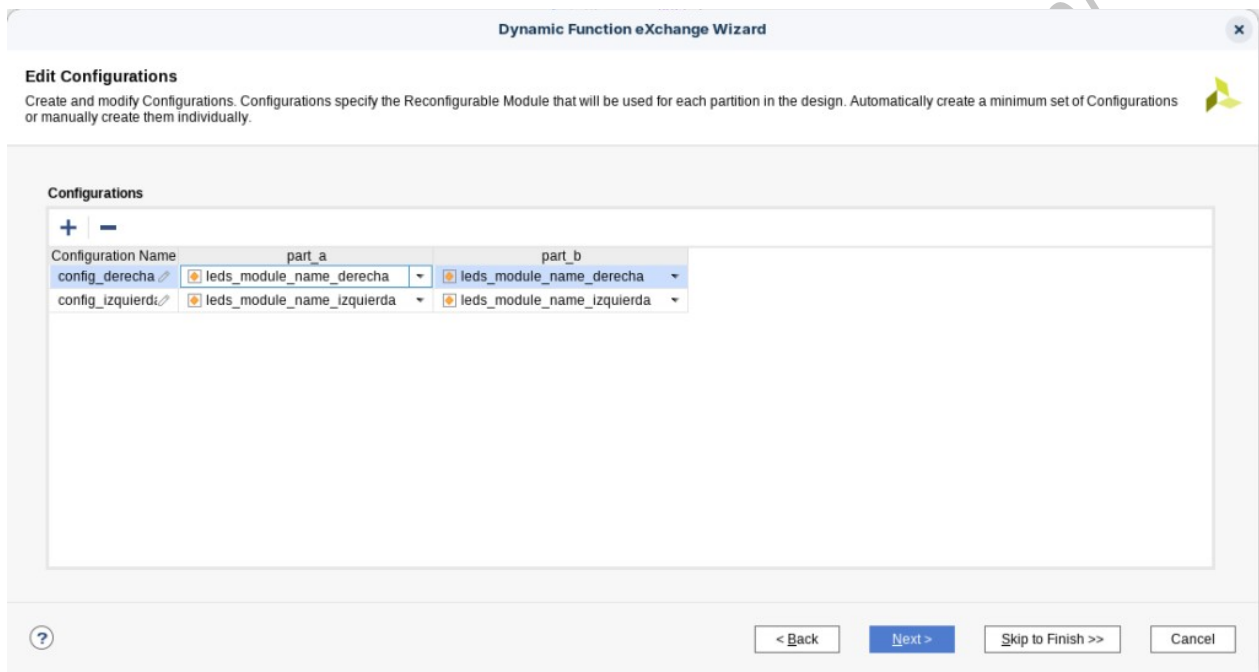
Para que nos entendamos, la pestaña anterior significa que va a crear dos bitstream para el fichero top, uno donde los puertos *led\_1* y *led\_2* van a ir a derechas, y otro bitstream en el que los puertos

*led\_1* y *led\_2* van a ir a izquierdas. Entonces, es aquí donde se puede elegir la configuración sobre el fichero top.

**NOTA:** que se pueda elegir la configuración de las particiones del fichero top, no significa que no existan bitstreams parciales que se pueden cargar sobre cualquiera de los bitstreams de los ficheros top. (*más adelante se entenderá esto mejor*)

**NOTA:** todos los módulos que se quieran añadir a la FPGA tienen que estar en un *config*. Esto es debido a que si hay alguno que no está en un *config*, no lo crea como bitstream parcial. **Solo crea un bitstream parcial de aquellos que estén en un *config*.**

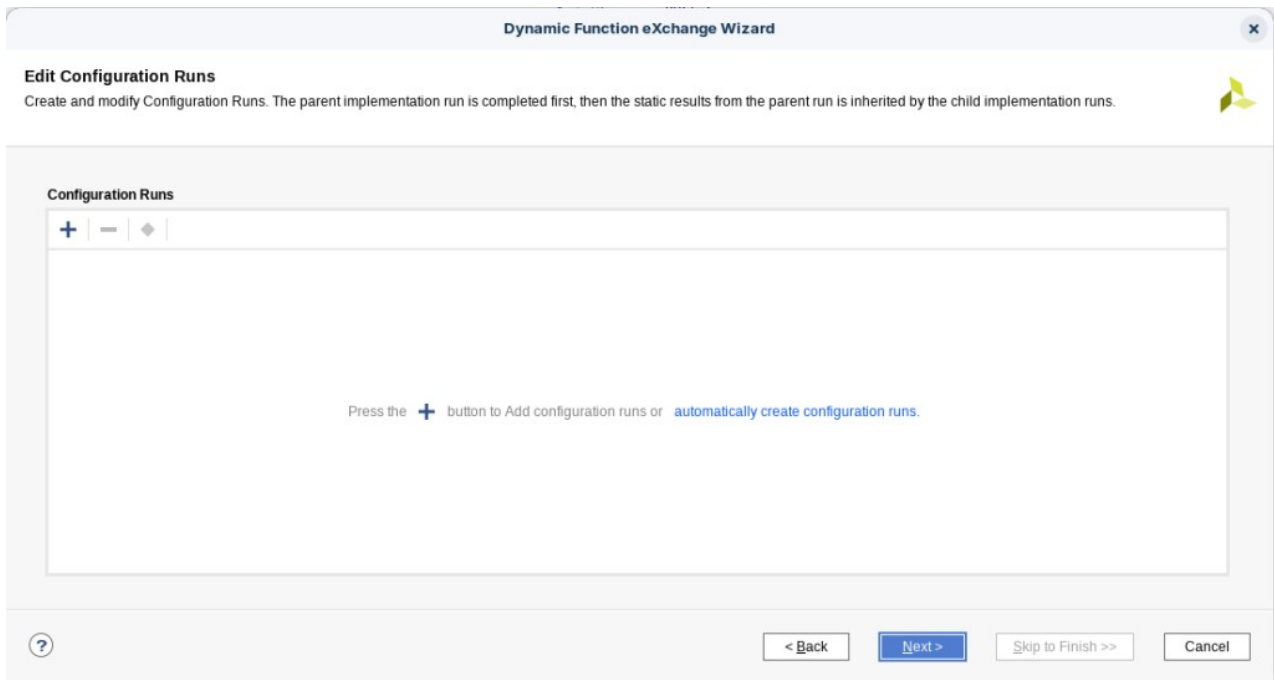
La cambiamos el nombre de las configuraciones para dejarlo así



**NOTA:** hay una opción de crear una configuración vacía, que es utilizando la opción *<greybox>*, esto permite crear un bitstream vacío para que el usuario lo rellene con los bitstreams parciales.



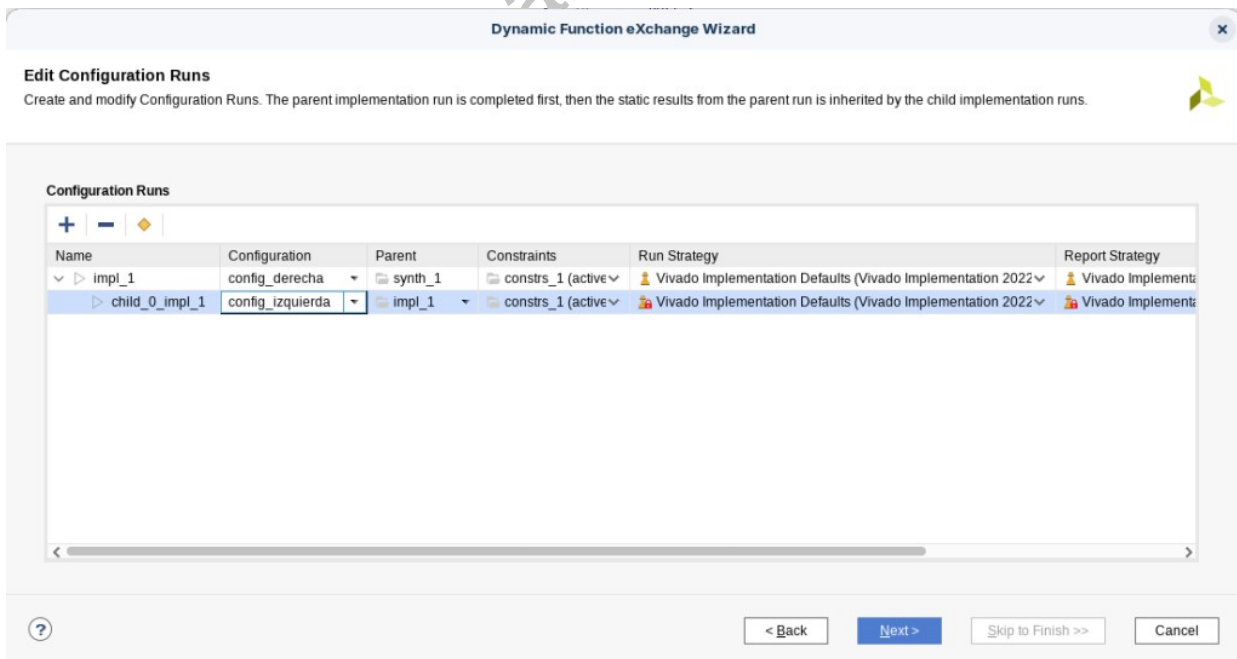
En la siguiente pestaña *Edit Configuration Runs* se configuran los perfiles de quien es la configuración de bitstream padre y cual(es) las hijas.



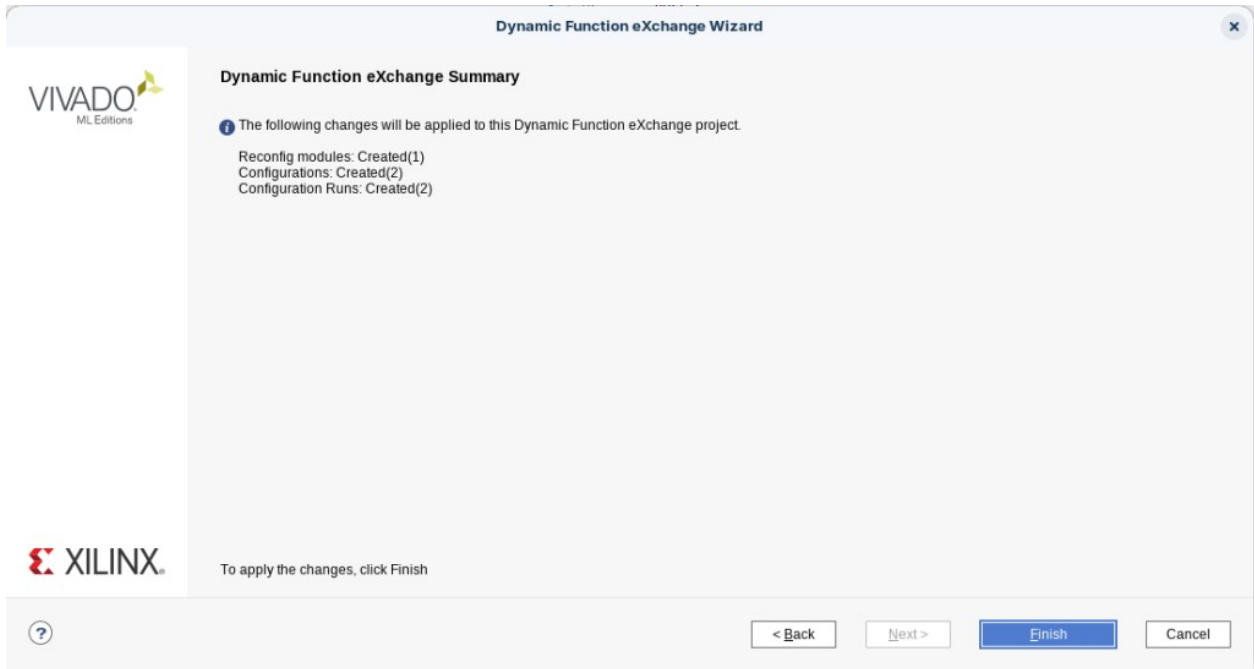
Hacemos igual que antes y pulsamos *automatically create configuration runs* y nos dice cuales son los perfiles.

Se puede ver que el fichero bitstream padre va a ser *config\_derecha* y el bitstream hijo va a ser el *config\_izquierda*. Se pueden crear tantos child como *config* haya.

**NOTA:** si solo has creado un *config* es posible que no se generen los bitstreams parciales, porque no tienen un child donde crearse.



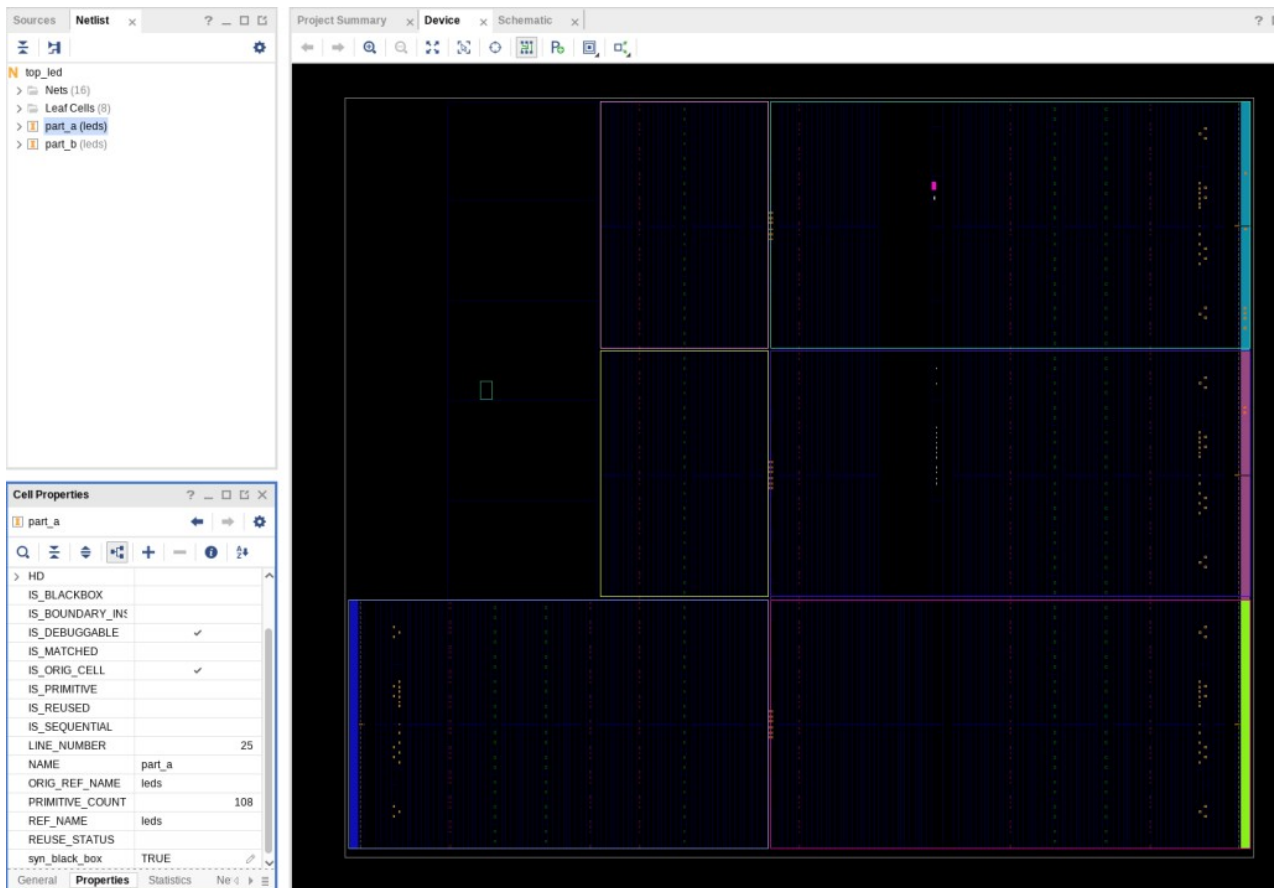
La última pestaña es un resumen de lo creado.



Una vez tenemos todos listo, pasamos al siguiente paso.

### **PASO 3: Seleccionar las zonas donde se generarán los bitstream**

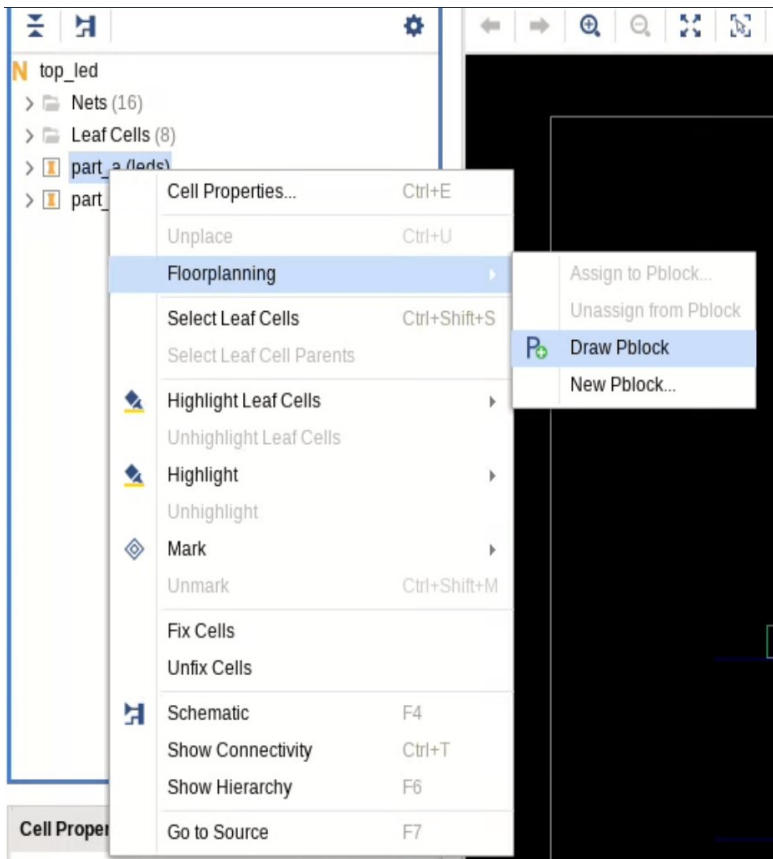
En este apartado lo que tenemos que hacer es sintetizar el proyecto. Y una vez este sintetizado tenemos que abrir la síntesis y abrir la pestaña Device, para ver las puertas lógicas de la FPGA.



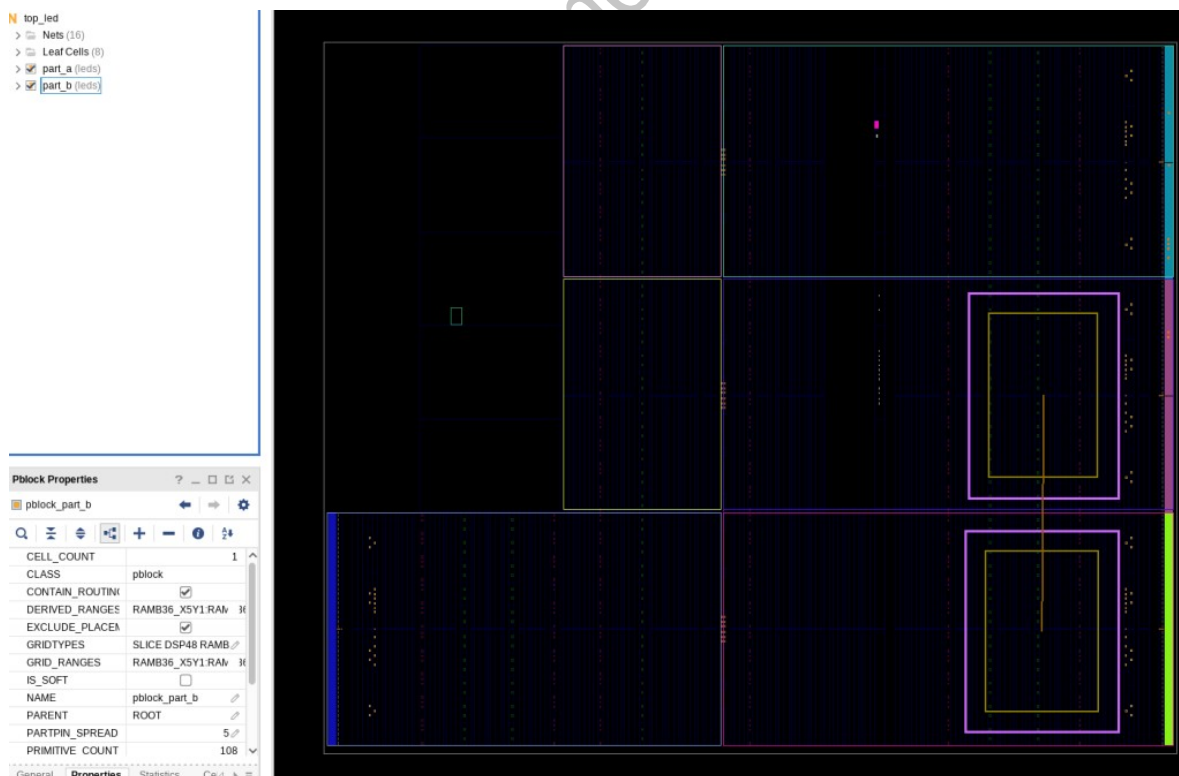
Una vez abierto se puede ver que a la izquierda en netlist tenemos dos celdas, una para parte de los leds(*part\_a* y *part\_b*).

Ahora lo que se tiene que hacer es asignar una zona de la FPGA para cada celda. Para ello se hace clic derecho sobre la celda y en la opción *Floorplaning* podemos elegir, si queremos crear una nueva zona para las celdas o asignar una que ya exista.

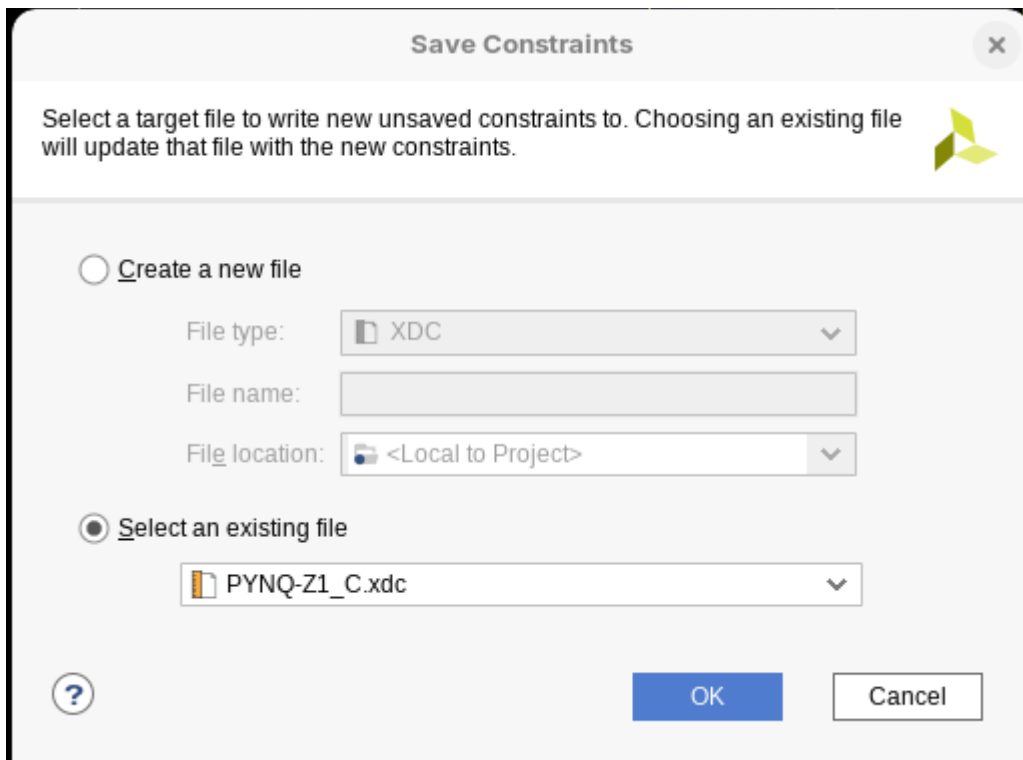
**NOTA:** Estas zonas se crean utilizando la opción *Draw Pblock*. Si ya existe, puedes asignársela a la celda.



Y creamos dos zonas separadas, una para cada celda.



Una vez tengamos creadas las zonas le damos a **guardar el proyecto**, y nos pregunta, si queremos crear un nuevo XDC con las restricciones creadas o actualizar uno que ya tengamos.

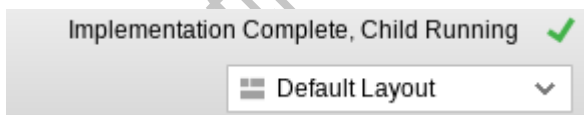


Con esto ya lo tenemos todo listo para la última parte.

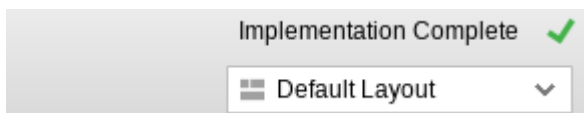
## PASO 4: Generar los bitstreams

Para generar los bitstreams se hace igual que siempre. Se marca a la opción **Generate Bitstream**, y si todo está bien configurado generará los bitstreams.

**NOTA:** Si lo que quieres es generar la implementación para ver como se han implementado las zonas, no hay que fiarse de las marcas que pone en la esquina superior. Porque la marca siguiente **NO** significa que la implementación esté terminada, porque todavía le falta las hijas.



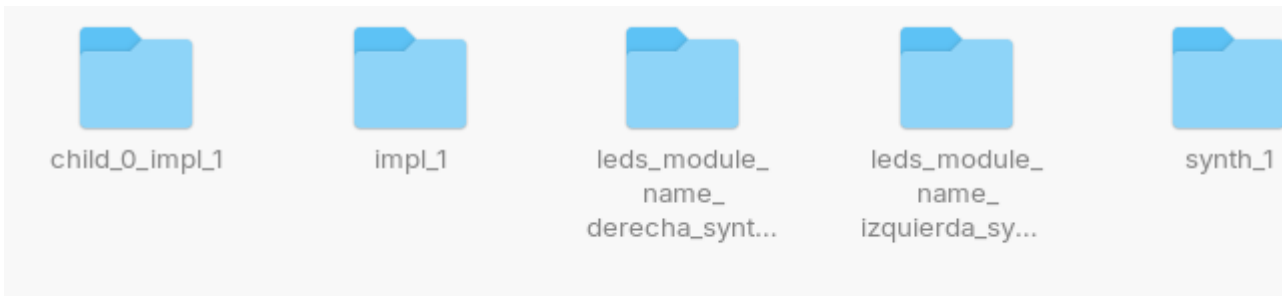
La finalización de la implementación es la de siempre



Una vez haya terminado de generar los bitstreams, nos vamos a la carpeta `<nombre_proyecto>.runs` y los buscamos.



Podemos ver que tenemos la carpeta *impl\_1* y una nueva carpeta llamada *child\_0\_impl\_1*. Además, tenemos dos carpetas de las síntesis, una de la izquierda y otra de la derecha.



Si abrimos la carpeta del bitstream del fichero top general (*impl\_1*), que es la que hemos dicho que tanto *part\_a* y *part\_b* tengan el bitstream de derecha(*config\_derecha*). Podemos ver que hay tres bitstreams, el del fichero top, y los dos parciales que van a derechas.

```
01 top_led.bit
10
01 part_a_leds_module_name_derecha_partial.bit
10
01 part_b_leds_module_name_derecha_partial.bit
10
```

Y si abrimos la carpeta hija(*child*), vemos lo mismo pero este fichero *top\_led*, hace que los leds vayan a izquierdas, y los parciales son los de la izquierda.

Entonces, se puede ver que todos los bitstreams existen, ahora hay que demostrar qué y cómo funcionan.

## Demostración

Aquí tienes un video de minuto y medio con la demostración de funcionamiento.

**NOTA:** los nombres de los bitstreams están al revés (el que dice izquierda va hacia la derecha, y el que dice derecha va a la izquierda)

<https://www.youtube.com/watch?v=6uP0OMx2xsE>

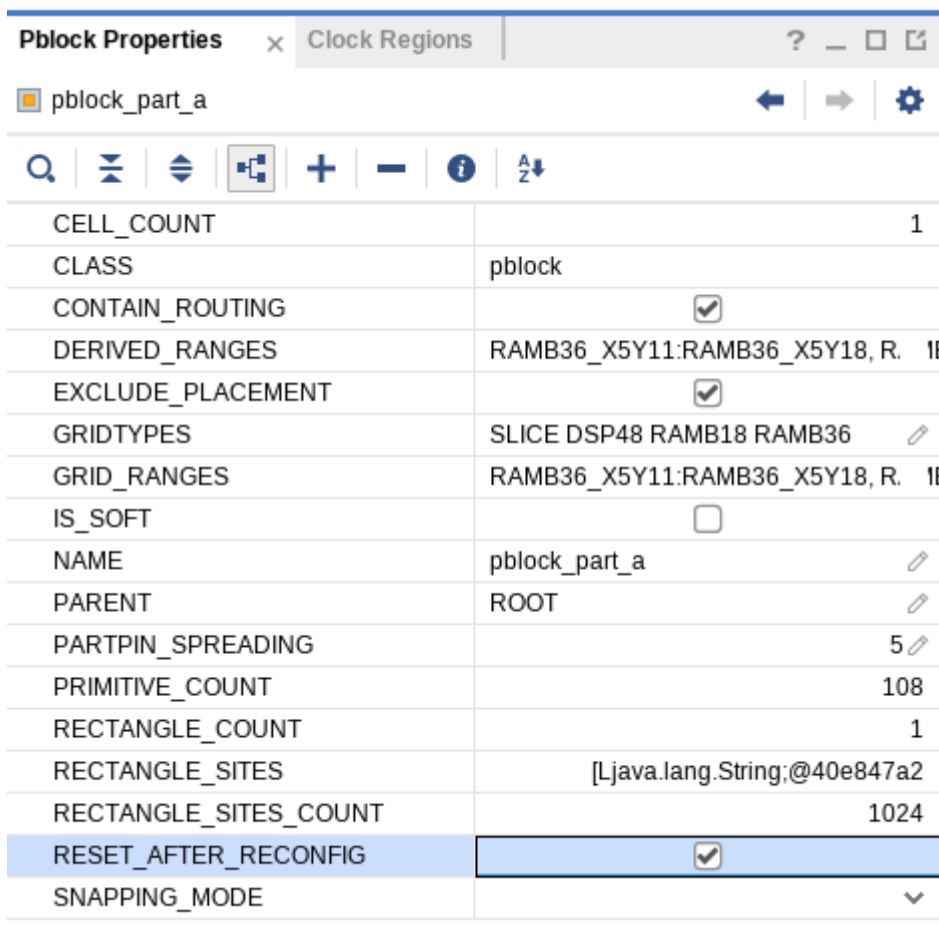
## Últimas notas

**NOTA 1: NO** se puede grabar de primeras un bitstream parcial. Primero se tiene que grabar un bitstream completo y después, se pueden grabar los bitstreams parciales.

Si lo que quieres es dejar sin funcionalidad el primer bitstream y grabar parciales paso a paso, puedes utilizar la opción de crear un bitstream total vacío (utilizando el `<greybox>`).

**NOTA 2:** Al crear los Pblocks se pueden configurar si quieres que se resetee toda la lógica interna durante la reconfiguración, esto es debido a que durante la reconfiguración, el HW que está dentro de los PBlocks pasa a un estado intermedio, este reseteo lo evita. Para ello se marca la opción

**RESET\_AFTER\_RECONFIG**

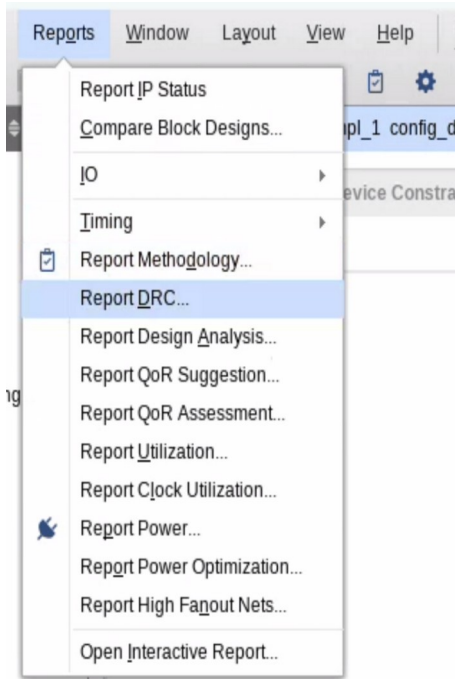


CELL_COUNT	1
CLASS	pblock
CONTAIN_ROUTING	<input checked="" type="checkbox"/>
DERIVED_RANGES	RAMB36_X5Y11:RAMB36_X5Y18, R. 1E
EXCLUDE_PLACEMENT	<input checked="" type="checkbox"/>
GRIDTYPES	SLICE DSP48 RAMB18 RAMB36
GRID_RANGES	RAMB36_X5Y11:RAMB36_X5Y18, R. 1E
IS_SOFT	<input type="checkbox"/>
NAME	pblock_part_a
PARENT	ROOT
PARTPIN_SPREADING	5
PRIMITIVE_COUNT	108
RECTANGLE_COUNT	1
RECTANGLE_SITES	[Ljava.lang.String;@40e847a2
RECTANGLE_SITES_COUNT	1024
RESET_AFTER_RECONFIG	<input checked="" type="checkbox"/>
SNAPPING_MODE	▼

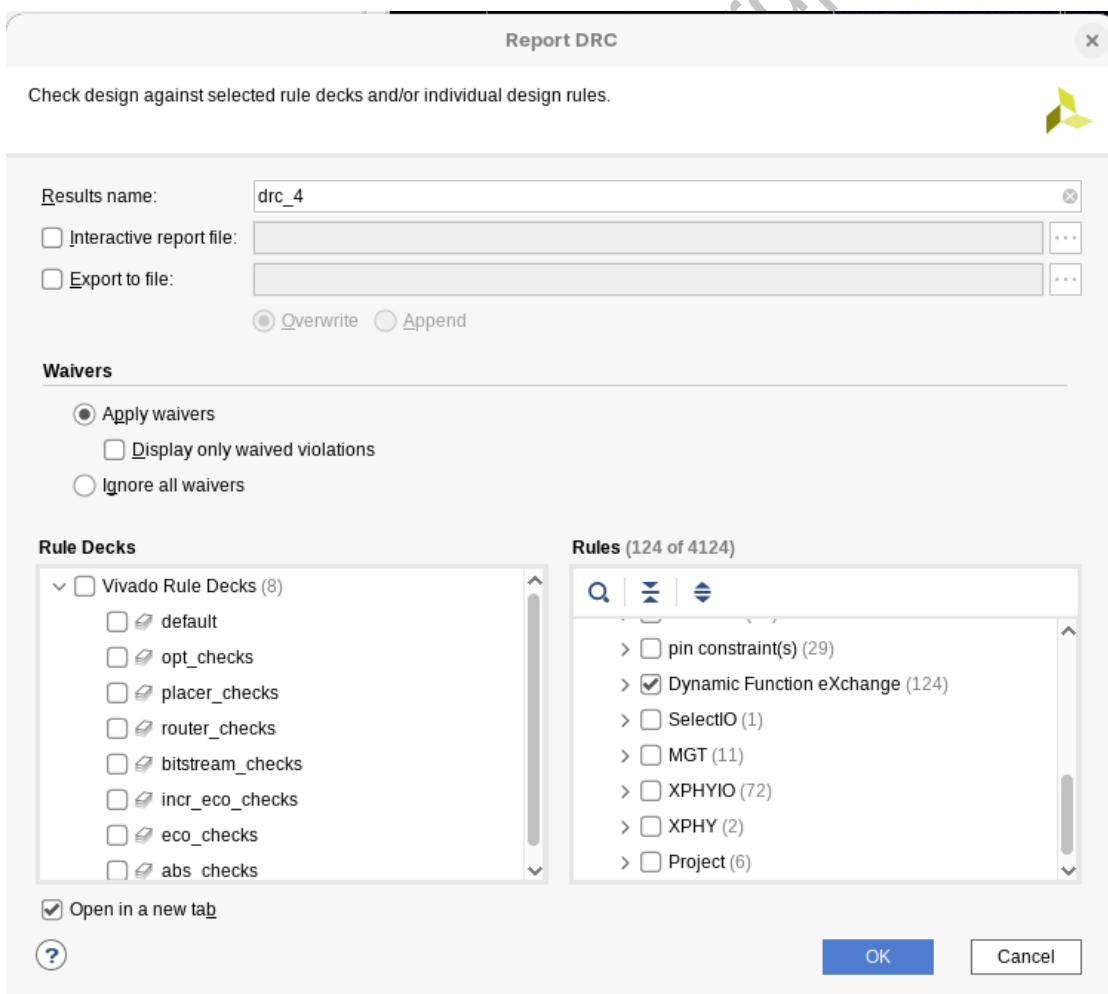
También la opción **SNAPPING\_MODE** permite ampliar o recortar los bloques PBlocks en caso de estos corten alguna CLB por el medio al ser seleccionados.

**NOTA 3:** Y una última cosa a tener en cuenta es que se puede hacer un DRC de los PBlocks, para comprobar que todo esté bien.

Para ello, en la pestaña *Reports*, en la opción *Report DRC*.



Deseleccionamos todas las casillas y nos quedamos con la casilla *Dynamic Function eXchange*.



<https://soceame.wordpress.com/2024/10/20/que-es-el-dynamic-function-exchange/>

Al realizarlo nos dirá los posible errores que hay.

## **Bibliografía**

Para más información:

- UG909 <https://docs.amd.com/r/en-US/ug909-vivado-partial-reconfiguration>

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