

**ALL PROGRAMMABLE**

**ANY MEDIA**

**5G**

**4K/8K**

**ANY STANDARD**

**ANY MACHINE**

**ANY NETWORK**

5G Wireless • Embedded Vision • Industrial IoT • Cloud Computing



Ciclo de Diseño con Vivado

# Objetivos

## ➤ Al finalizar el módulo, los alumnos podrán:

- Conocer las características principales del entorno de desarrollo Vivado
- Conocer las distintas opciones de análisis aplicadas en el desarrollo de sistemas con FPGA
- Implementar un ciclo de desarrollo de un sistema basado en FPGA en el entorno Vivado
- Adquirir nociones básicas del uso de scripts en el ciclo de desarrollo

# Contenido

- **Características del Entorno de Desarrollo Vivado**
- Introducción a las distintas herramientas del Entorno
- Ciclo de Desarrollo
- Resumen

# Entorno Vivado - Características

## ➤ Diseño y análisis interactivo

- Análisis de temporización, de conectividad, utilización de recursos, y restricciones de temporización.

## ➤ Desarrollo a nivel RTL

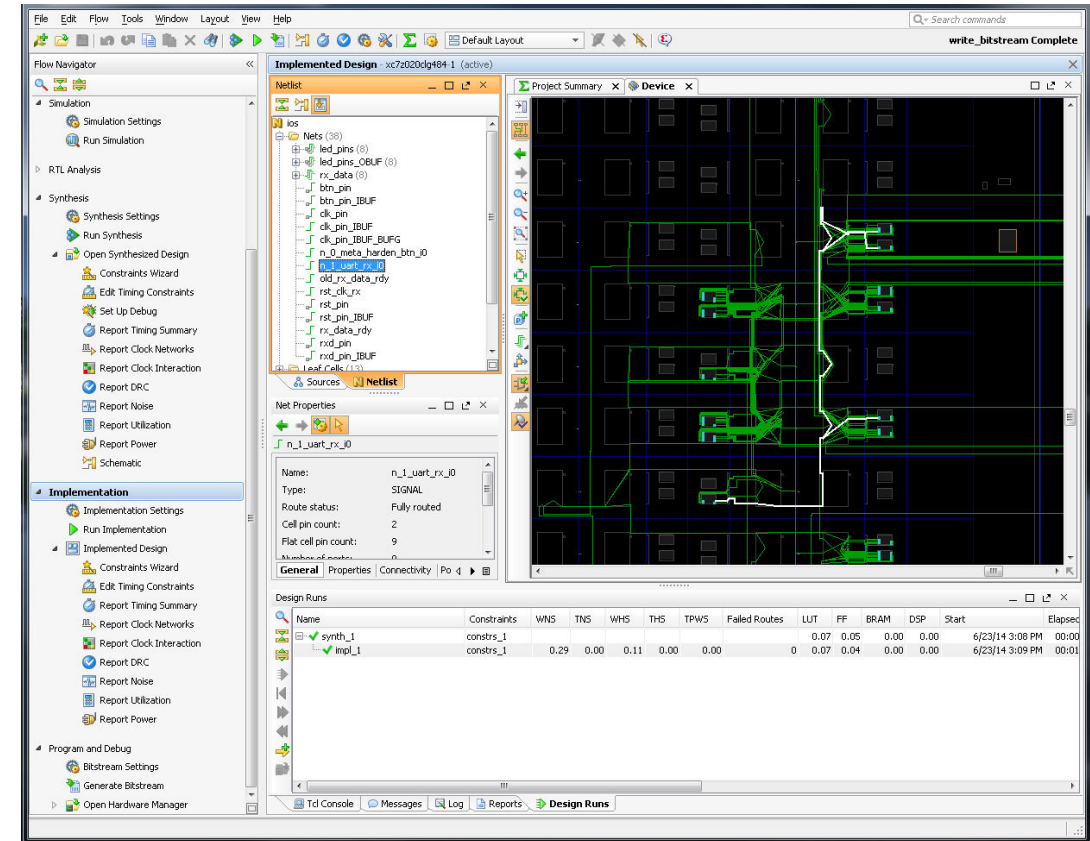
- Generación de código HDL
- Esquemas jerarquicos
- Generación de diagramas esquemáticos

## ➤ Integración con el simulador XSIM

## ➤ Procesos de síntesis e implementación

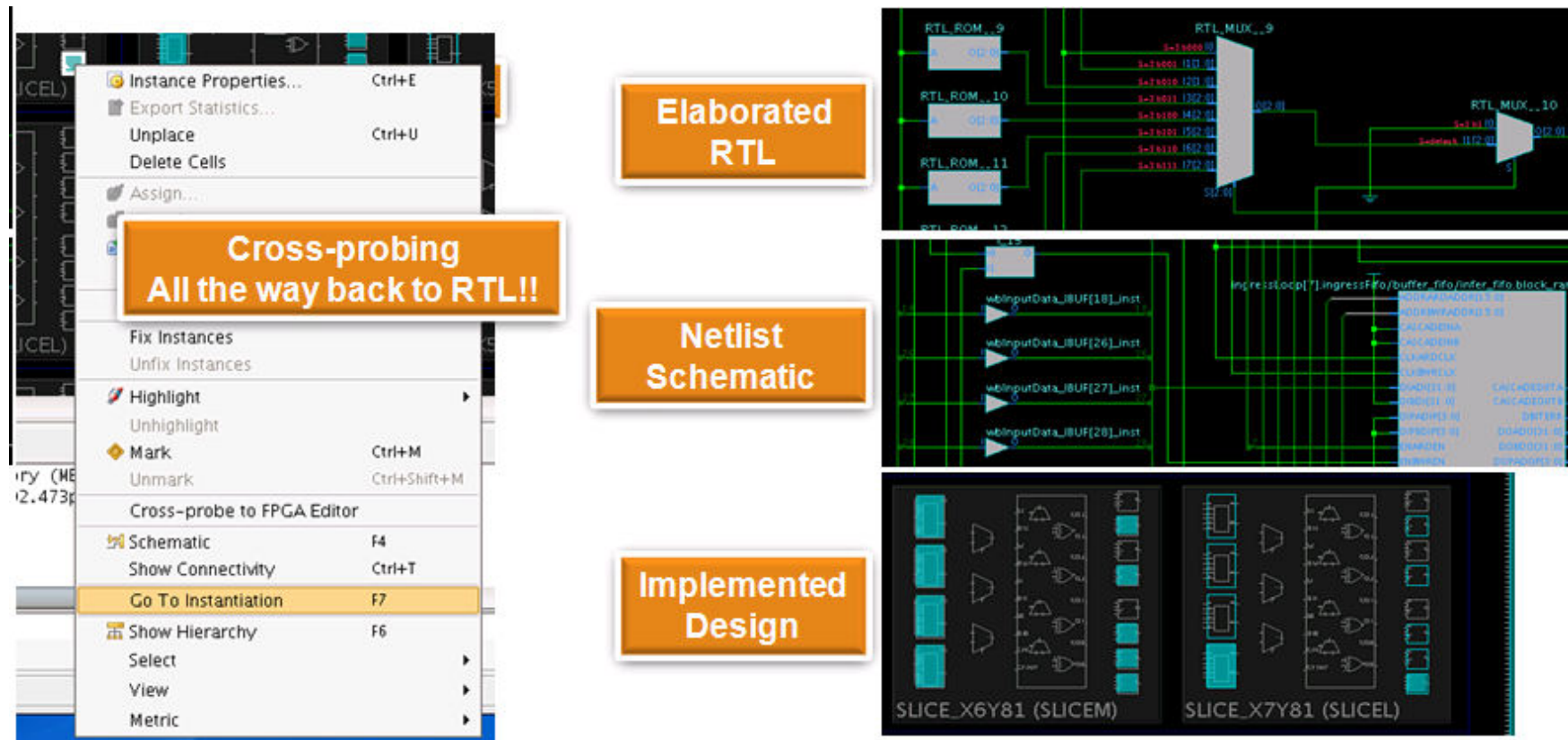
## ➤ Gestión de ubicación de I/O

- Asignación de I/O basada en reglas (interfase a memorias)



# Opciones de visualización del diseño

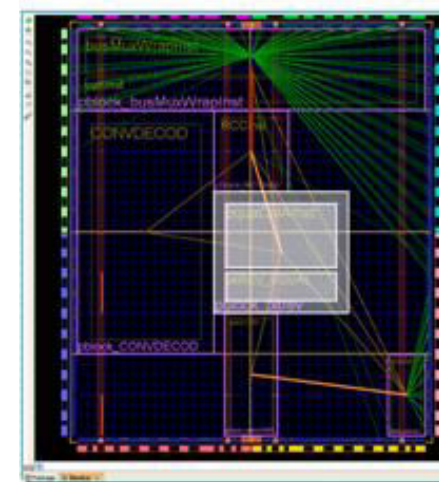
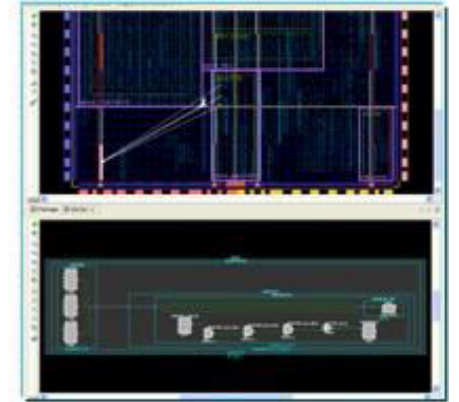
- **Se puede generar una vista visual en las distintas etapas del ciclo de desarrollo**
  - Análisis en vistas netlist/diagrama/RTL





# Herramientas para alcanzar restricciones de temporización

- **Se pueden analizar distintas implementaciones**
  - Resaltado de los caminos que no cumplen restricciones en un análisis post-ruteo
  - Identificación de los caminos críticos para la temporización
- **Planificación de ubicación física**
  - Ubicación y ruteo guiados
- **Utilizacion de estimaciones**
  - Estimación de recursos
  - Estimación de área y tiempos
- **Conectividad**
  - I/Os utilizados, redes representativas de los circuitos lógicos, dominios de reloj, etc



# Consola y Lenguaje Tcl

- TCL (Tool Command Language) es un lenguaje de scripting interpretado. Su característica principal es que todas sus facilidades (estructuras de control, asignación de variables, etc) tiende a la forma de un comando (command Arg0 Arg1 ... ArgN)
- La consola Tcl permite interactuar con las herramientas del Entorno
- El uso de scripts se puede aplicar tanto a desarrollos centrados en un proyecto, como a desarrollos independientes.
- Los archivos de Journal y Logs se pueden utilizar como parte de los scripts

# Desarrollos basados en Proyectos y desarrollos independientes

## ➤ Las herramientas de Vivado pueden trabajar de 2 modos

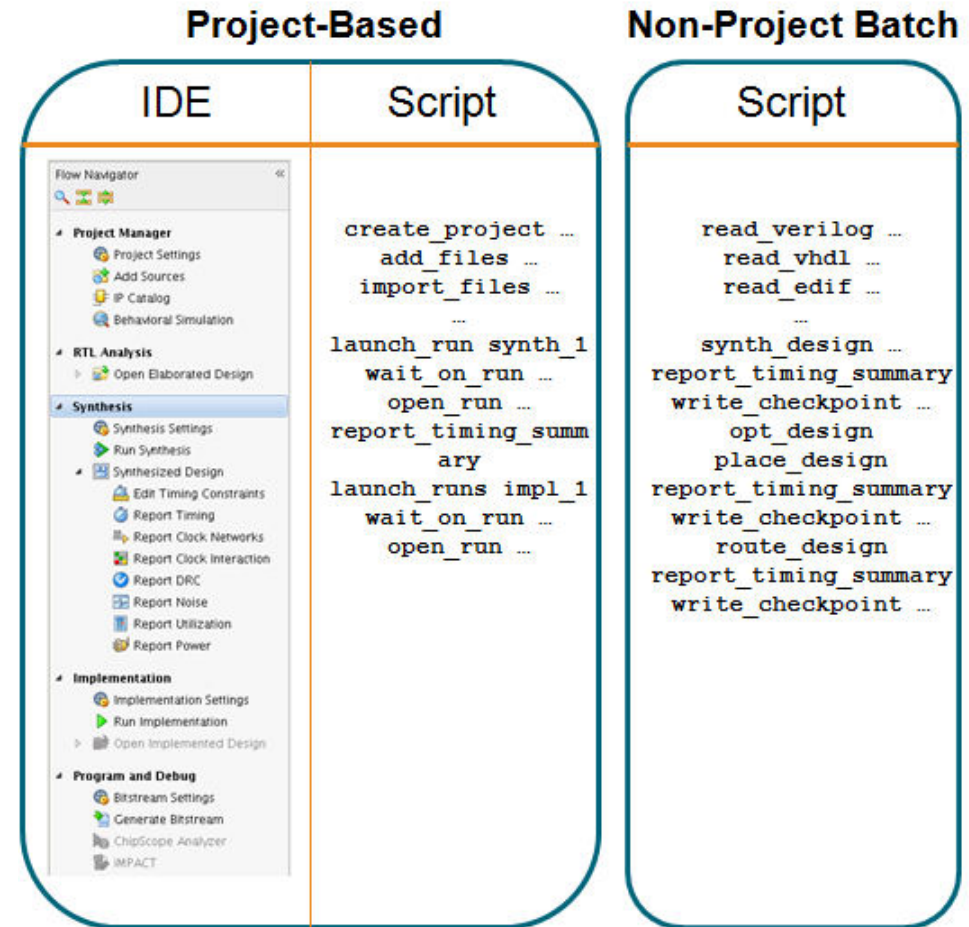
- Desarrollos basados en Proyectos
- Desarrollos independientes

## ➤ Desarrollos independientes

- No tiene estructura de proyecto
- Basados en scripts Tcl
- Se puede usar interface gráfica (comando start\_gui)
- Los reportes y puntos de chequeo deben generarse manualmente mediante comandos TCL
- Útil para tests de regresión

## ➤ Desarrollos basados en Proyectos

- Estructura de proyecto (archivo \*.XPR)
- Los Reportes/Estados/Iteraciones disponibles en cada punto del ciclo de desarrollo
- Gestión mediante GUI o scripts Tcl
- Útil para nuevos desarrollos



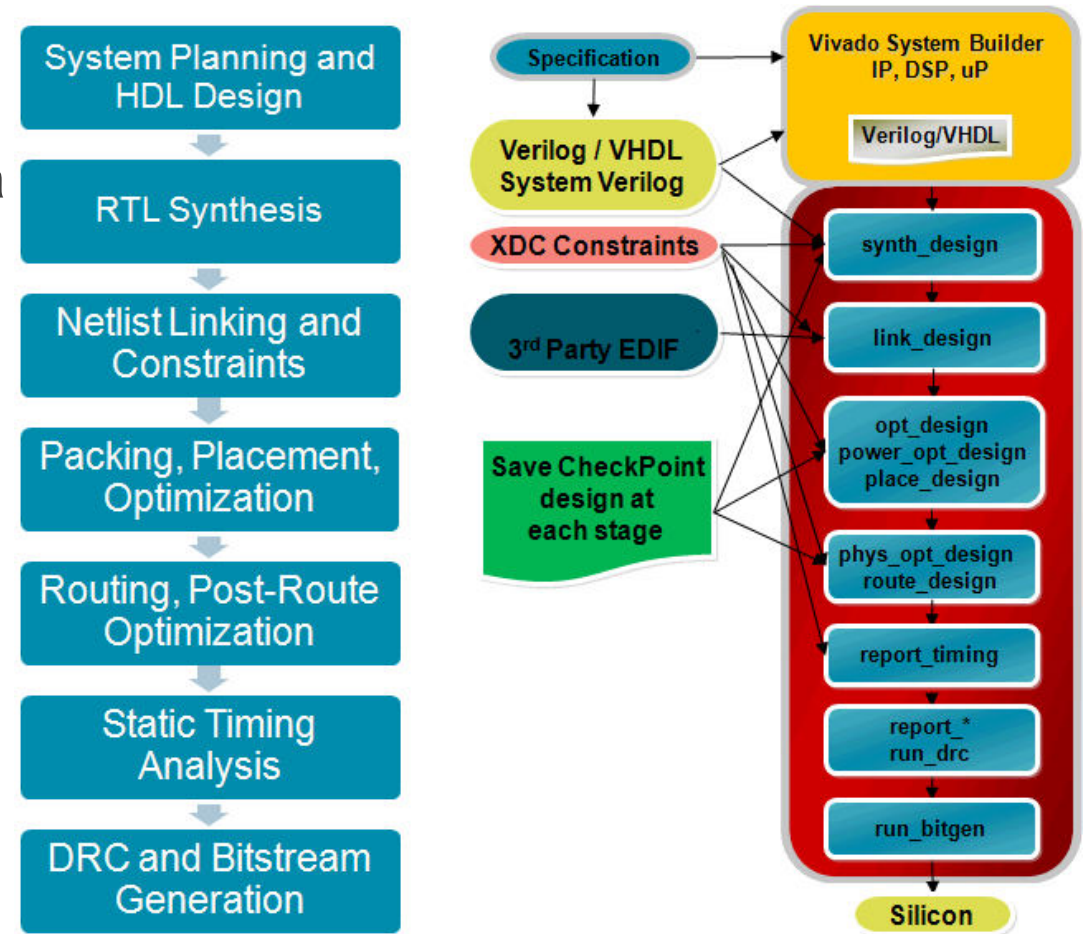


# Contenido

- Características del Entorno de Desarrollo Vivado
- **Introducción a las distintas herramientas del Entorno**
- Ciclo de Desarrollo
- Resumen

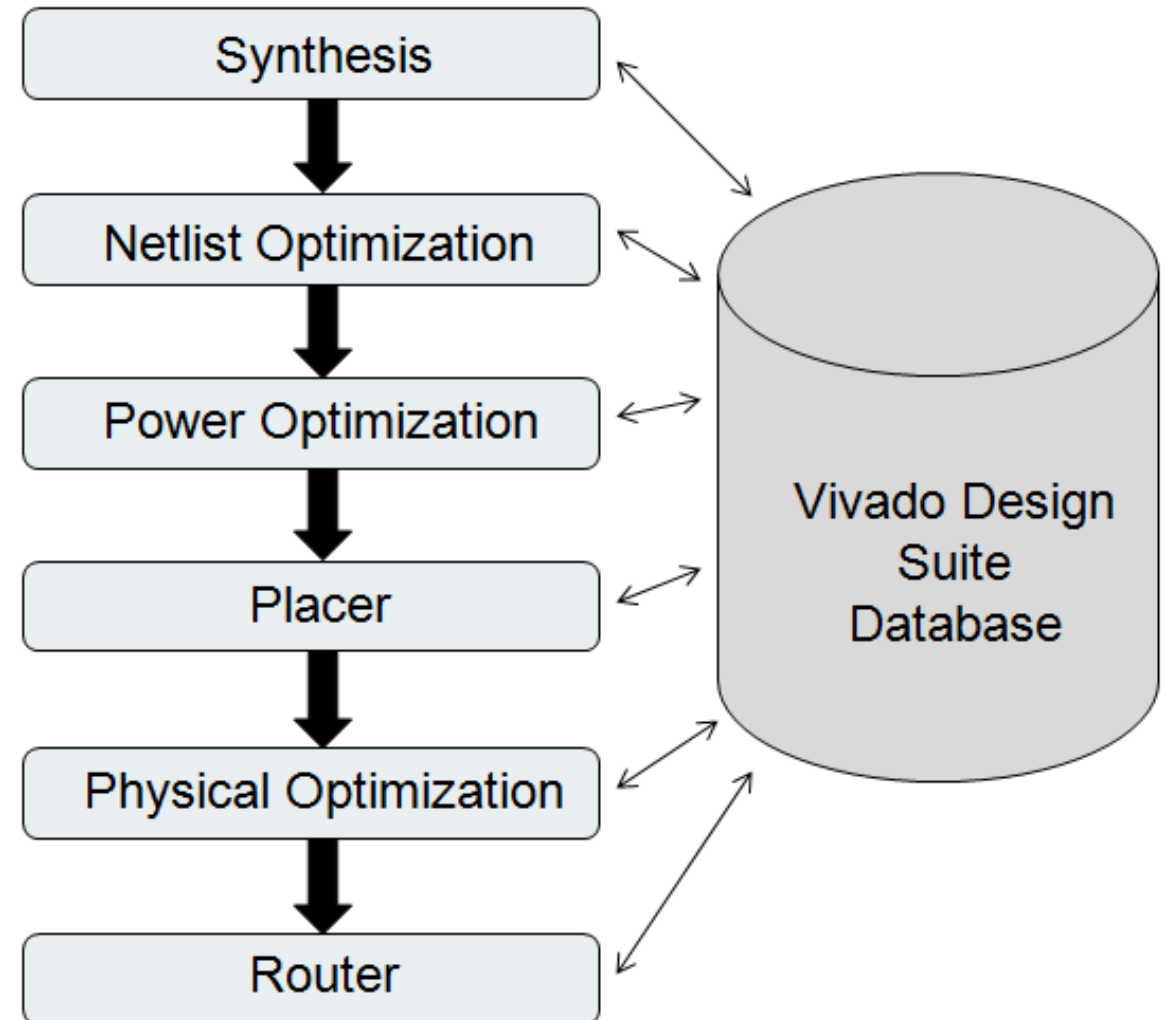
# Ciclo de diseño con Vivado

- **Agregado de IP en forma interactiva**
  - AXI4, IP\_XACT, etc
- **Un único lenguaje de restricciones aplicable a todas las etapas del ciclo de desarrollo (XDC)**
  - Permite aplicar restricciones en forma dinámica en cualquier etapa
- **Reportes en cualquier etapa del desarrollo**
  - A través de la API de reportes de Tcl
- **Un modelo de datos único a través de todo el ciclo de desarrollo**
- **Se pueden salvar puntos de chequeo en cualquier etapa**
  - Se mantienen los resultados de Netlist, restricciones, ubicación y ruteo



# Diseño centrado en una base de datos

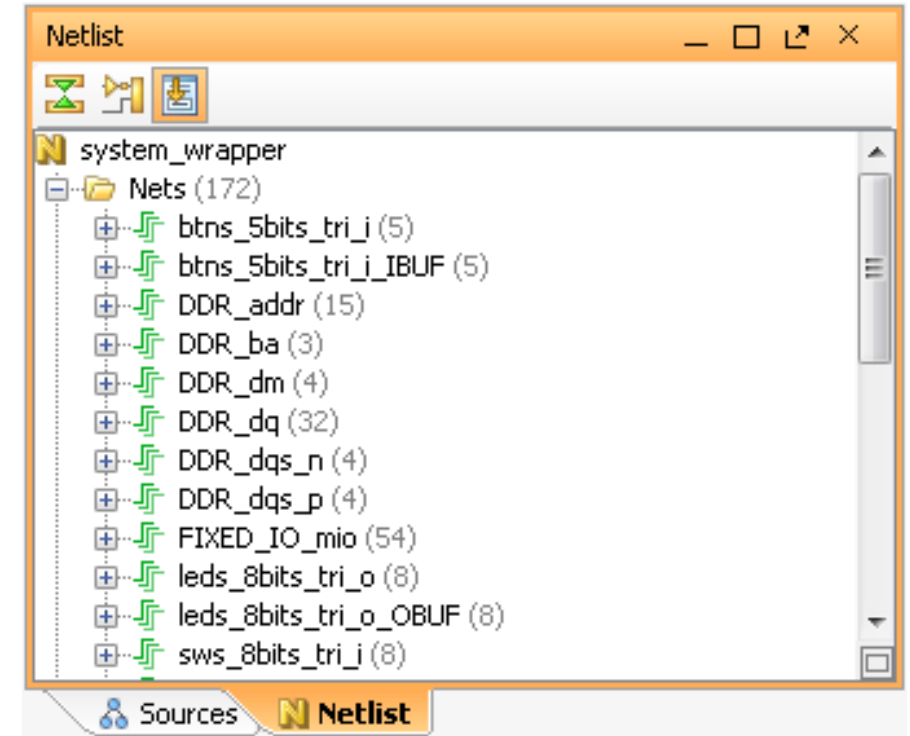
- **Todas las herramientas acceden a la misma información del proyecto**
  - Cada herramienta opera sobre una netlist y la modifica o crea una nueva
- **La netlist pasa por diferentes estados durante el proceso de diseño**
  - Elaborated
  - Synthesized
  - Implemented



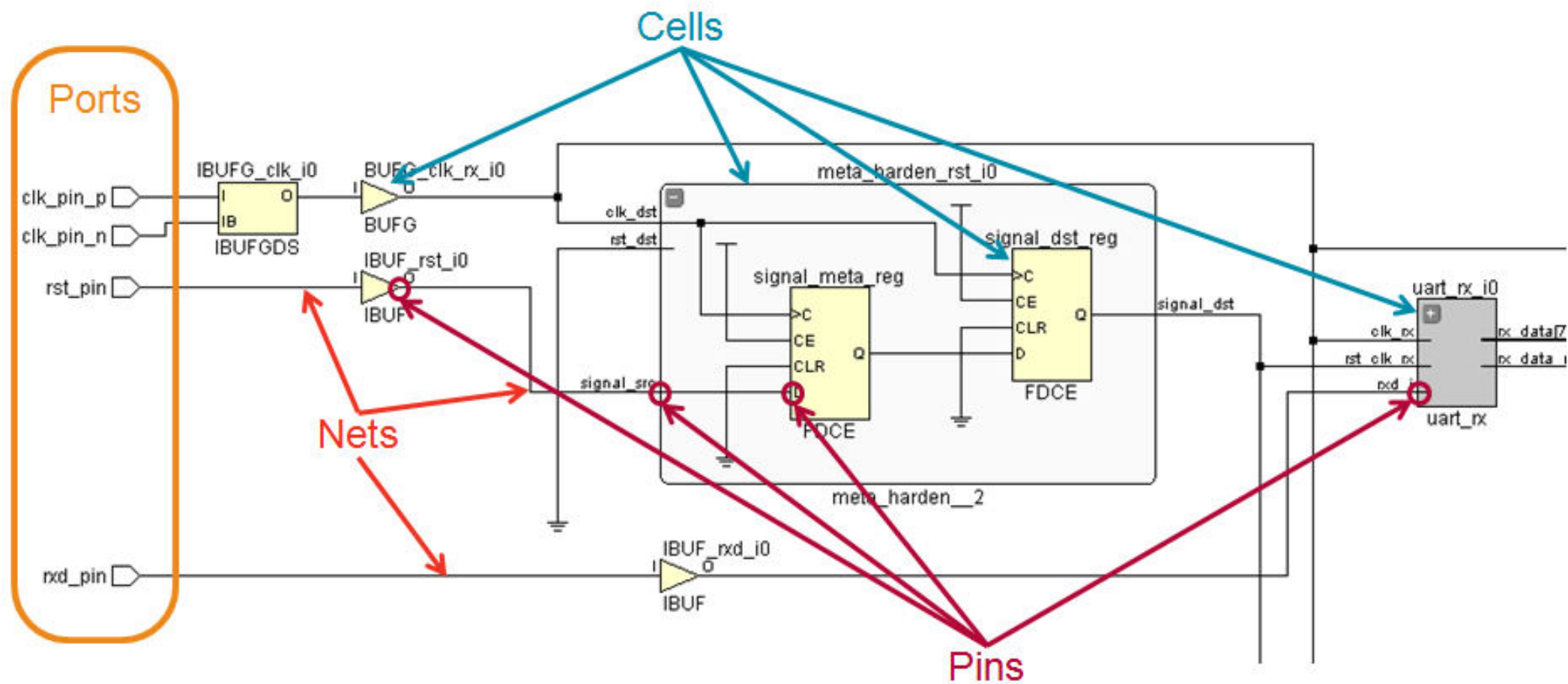
# Netlist:

## ➤ Una Netlist es una descripción del sistema que se está diseñando

- Consiste en celdas (cells), pines (pins), puertos (ports) y redes (nets) que los interconectan
- Las **celdas** son los objetos básicos del diseño
  - Instancias de módulos (verilog) o entidades (vhdl) creadas por el usuario
  - Instancias de elementos de librerías
    - LUTs, FF, RAMs, bloques DSP, etc...
  - Representaciones de funciones de hardware
  - Cajas negras
- Los **pines** son puntos de conexiones de las celdas.
- Los **puertos** son la interfaz de entrada y salida del diseño.
- Las **redes** son conexiones entre los pines y entre los pines y los puertos



# Objetos de la Netlist

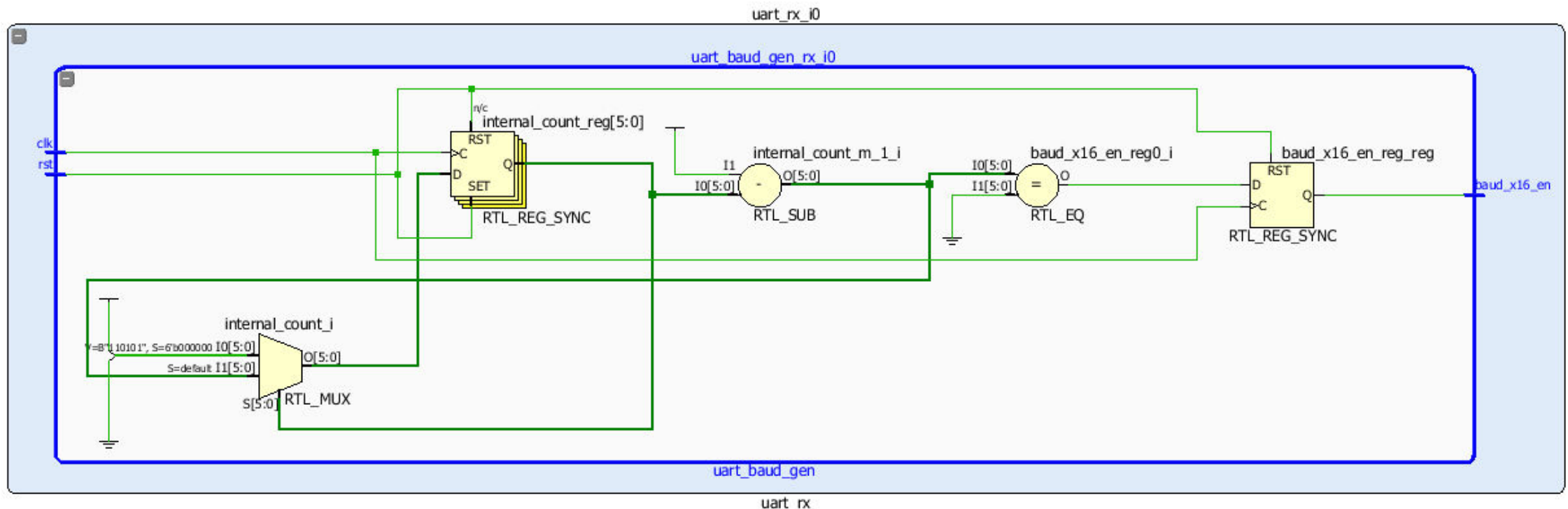




# Diseño en estado Elaborado (Elaborated)

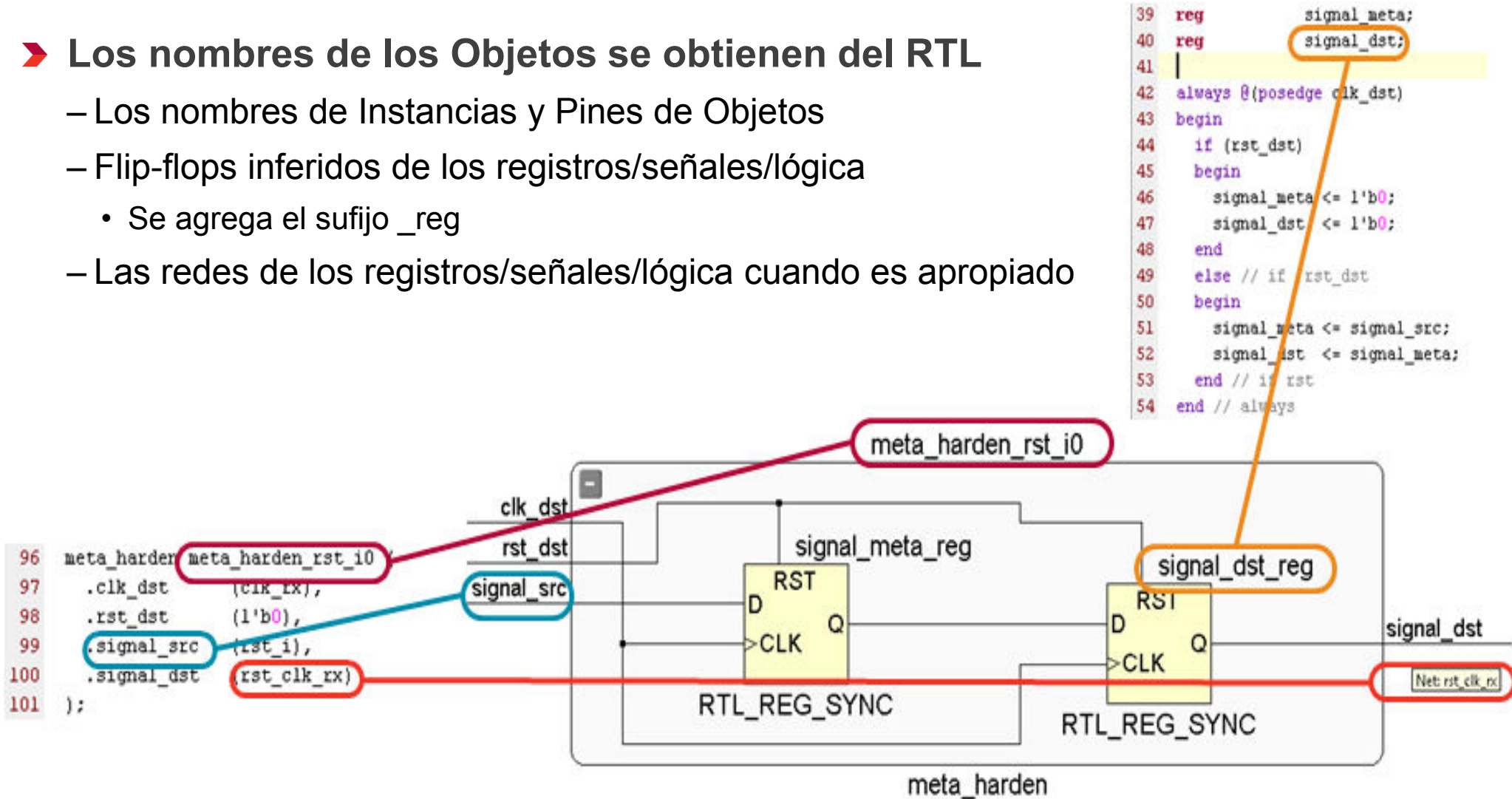
## ➤ Representación del diseño antes de la síntesis

- Netlist interconectada con una estructura jerárquica y genérica de celdas
  - Instancias de módulos/entidades
  - Representaciones genéricas de componentes de hardware
    - Compuertas AND, OR, buffers, multiplexores, sumadores, comparadores, etc...



# Nombres de los Objetos:

- Los nombres de los Objetos se obtienen del RTL
  - Los nombres de Instancias y Pines de Objetos
  - Flip-flops inferidos de los registros/señales/lógica
    - Se agrega el sufijo \_reg
  - Las redes de los registros/señales/lógica cuando es apropiado



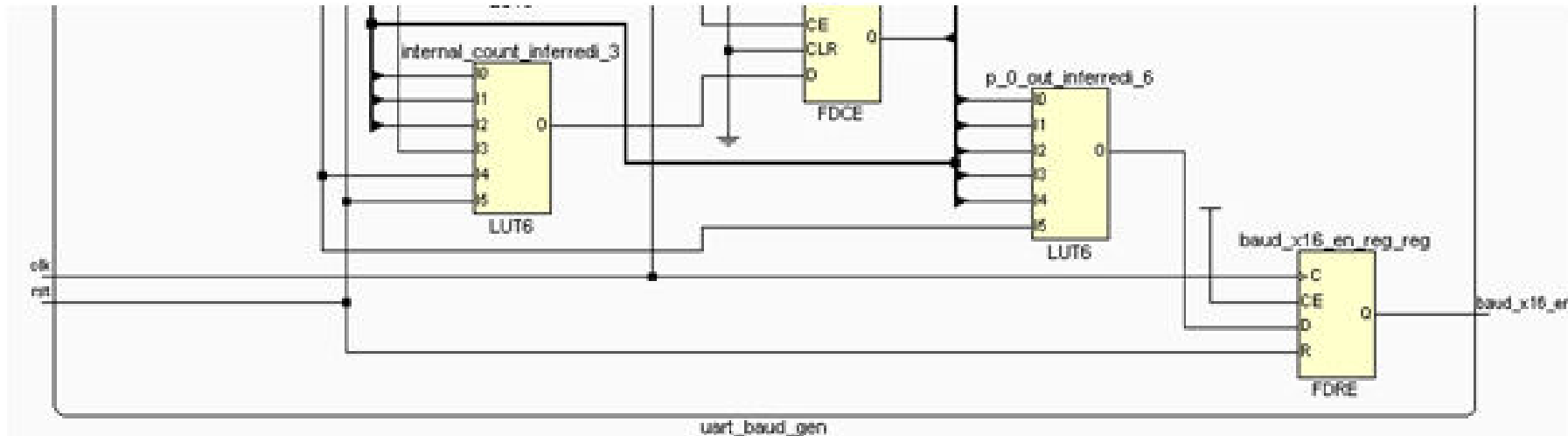
# Diseño en estado Sintetizado (Synthesized)

## ➤ Representación del diseño despues de la síntesis

– Netlist interconectada de elementos básicos (Basic Elements – BELs)

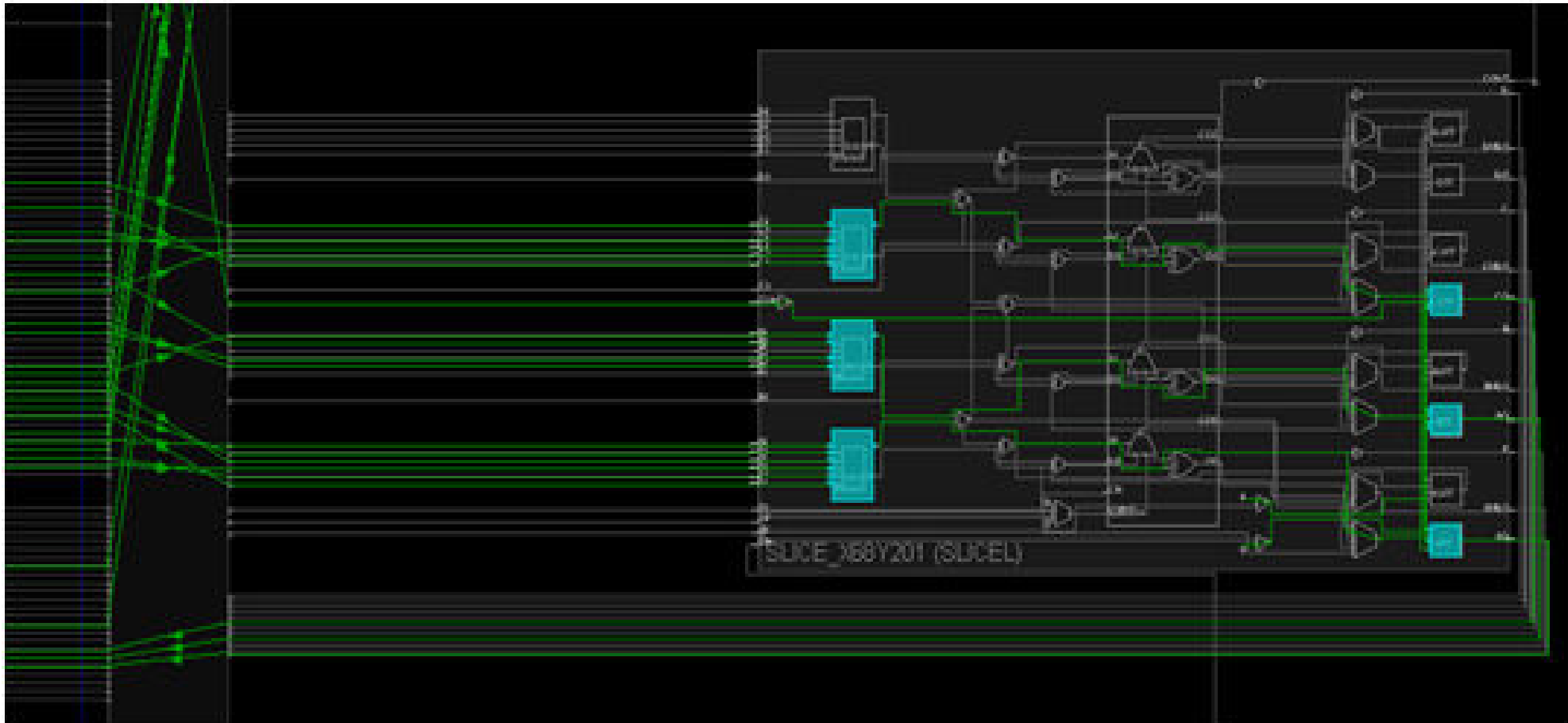
- Instancias de módulos/entidades
- BELs
  - LUTs, flip-flops, carry chain, MUXes
  - Block RAMs, bloques DSP
  - Elementos de temporización (BUFG, BUFR, MMCM, ...)
  - Elementos de I/O (IBUF, OBUF, I/O flip-flops)

## ➤ Los nombres de Objetos son los mismos que la netlist "Elaborated" cuando es posible



# Diseño en estado Implementado (Implemented)

- **Representación del diseño durante y después del proceso de implementación**
  - Estructuralmente similar al diseño sintetizado
  - Las celdas tienen ubicación definida y las redes se mapean al ruteo específico del dispositivo



# Estructura de un Proyecto

- **Toda la información del proyecto está almacenada en un directorio con el nombre del proyecto (*project\_name* directory) conteniendo los siguientes subdirectorios:**
  - *project\_name.xpr* : Archivo de configuración del proyecto
  - *project\_name.runs*: Directorio con los resultados de las iteraciones de las herramientas
  - *project\_name.srcs* : Directorio con los archivos fuente HDL, netlists, y restricciones XDC
  - *project\_name.data*: Directorio con la información de ubicación en el dispositivo



# Archivos de Journal y Log

## ➤ Archivo de Journal (vivado.jou)

- Contiene los comandos Tcl ejecutados por el IDE

## ➤ Archivo Log (vivado.log)

- Contiene todos los mensajes producidos por el IDE, incluyendo los comandos Tcl y sus resultados, mensajes de info/advertencia/error, etc.

## ➤ Ubicación de estos archivos

- Linux: el directorio desde donde se invoca el IDE
- Windows invocado a través del icono: %APPDATA%\Xilinx\Vivado o C:\Users\<user\_name>\AppData\Roaming\Xilinx\Vivado
- Windows invocado a través de la línea de comandos: el directorio desde donde se invoca el IDE
- Desde la interfaz gráfica (GUI)
  - Seleccionar File > Open Log File
  - Seleccionar File > Open Journal File

# Puntos de chequeo (Checkpoints)

## ➤ Guarda la base de datos a disco

- Netlist (edif, eventualmente una netlist verilog)
- Restricciones (xdc)
- Información de ubicación/ruteo/configuración (xdef)

## ➤ Comandos Tcl

- write\_checkpoint <filename>
  - Genera un archivo filename.dcp, que es un archivo comprimido
  - Se puede descomprimir a los archivos filename.edf, filename.xdc, filename.xdef, filename.wdf, y dcp.xml
- read\_checkpoint <filename>

## ➤ Se pueden leer y escribir desde el IDE

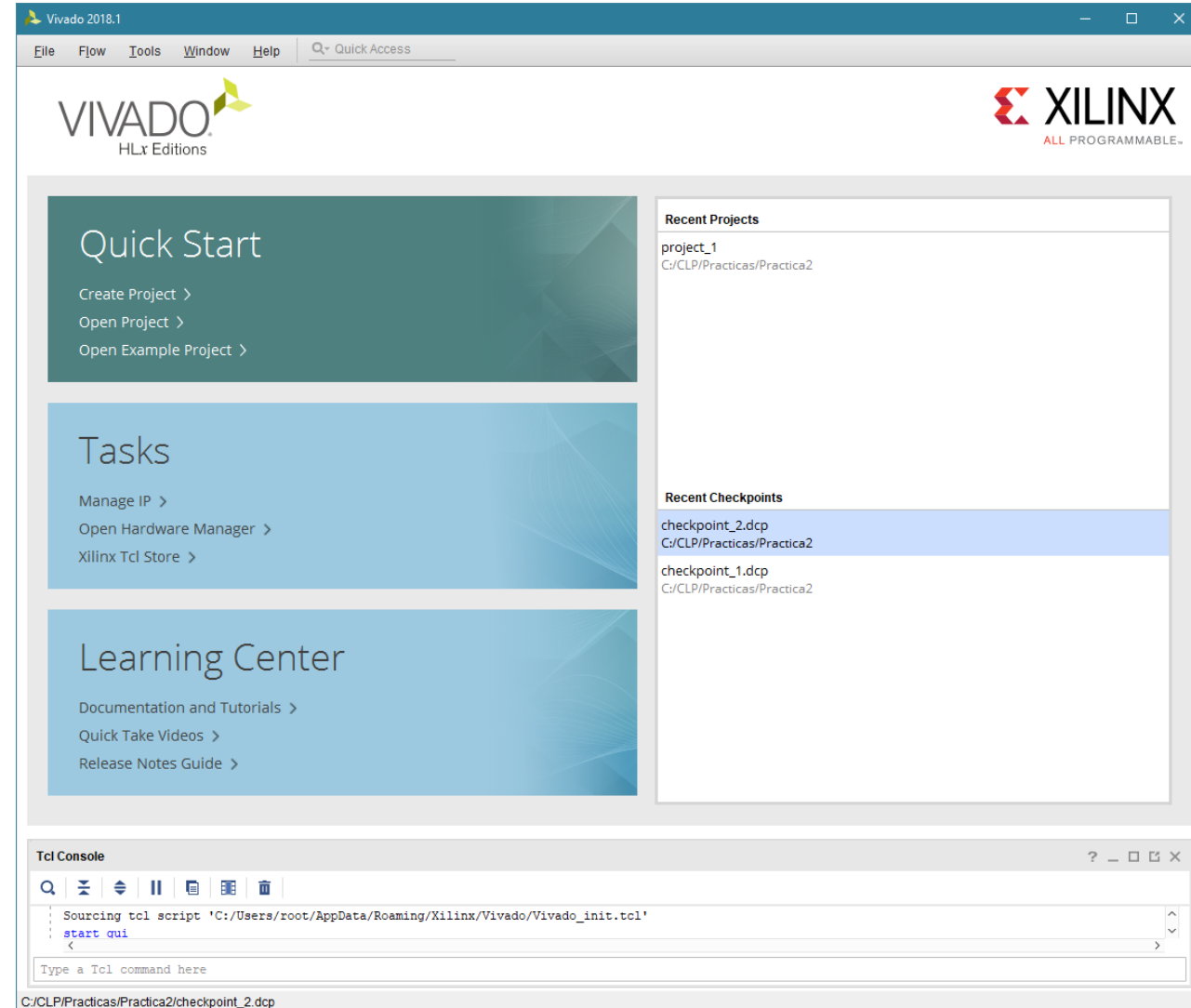
## ➤ Se pueden abrir y modificar los checkpoints desde el IDE

# Contenido

- Características del Entorno de Desarrollo Vivado
- Introducción a las distintas herramientas del Entorno
- **Ciclo de Desarrollo**
- Resumen

# Pantalla de Inicio

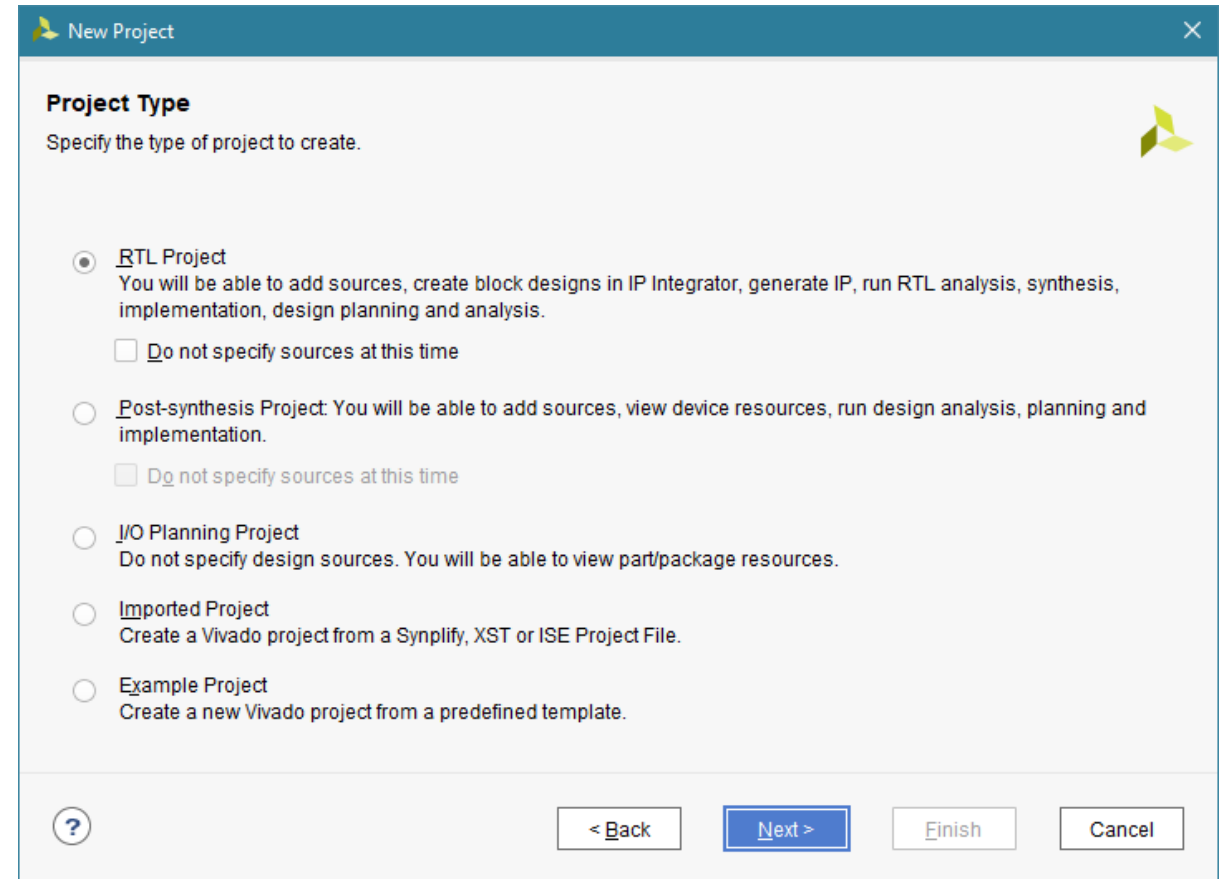
- **Todos los botones están agrupados por funcionalidad**
- **Inicio Rápido**
  - Abrir proyectos anteriores
  - Crear proyectos nuevos o de ejemplo
- **Tareas**
  - Gestión de IP, gestión de hardware, repositorio de scripts Tcl (Tcl store)
- **Enlaces**
  - Documentación y Tutoriales
  - Videos
  - Release Notes
- **Consola Tcl**



# Asistente para nuevos proyectos (New Project Creation Wizard)

## ➤ Al crear un proyecto nuevo hay 5 opciones:

- RTL
  - Es el más genérico.
- Post-synthesis
  - Los archivos fuente están en formato EDIF o NGC
- I/O planning
  - Para ubicación de pines de E/S
  - No tiene archivos fuente
- Import Project
  - Importar un proyecto desde Synplify, XST, o ISE
- Example Project
  - Crear un proyecto de ejemplo



The screenshot shows the 'New Project' dialog box with the 'Project Type' section. The 'RTL Project' option is selected. The dialog includes a title bar, a close button, a help icon, and navigation buttons at the bottom.

**New Project**

**Project Type**  
Specify the type of project to create.

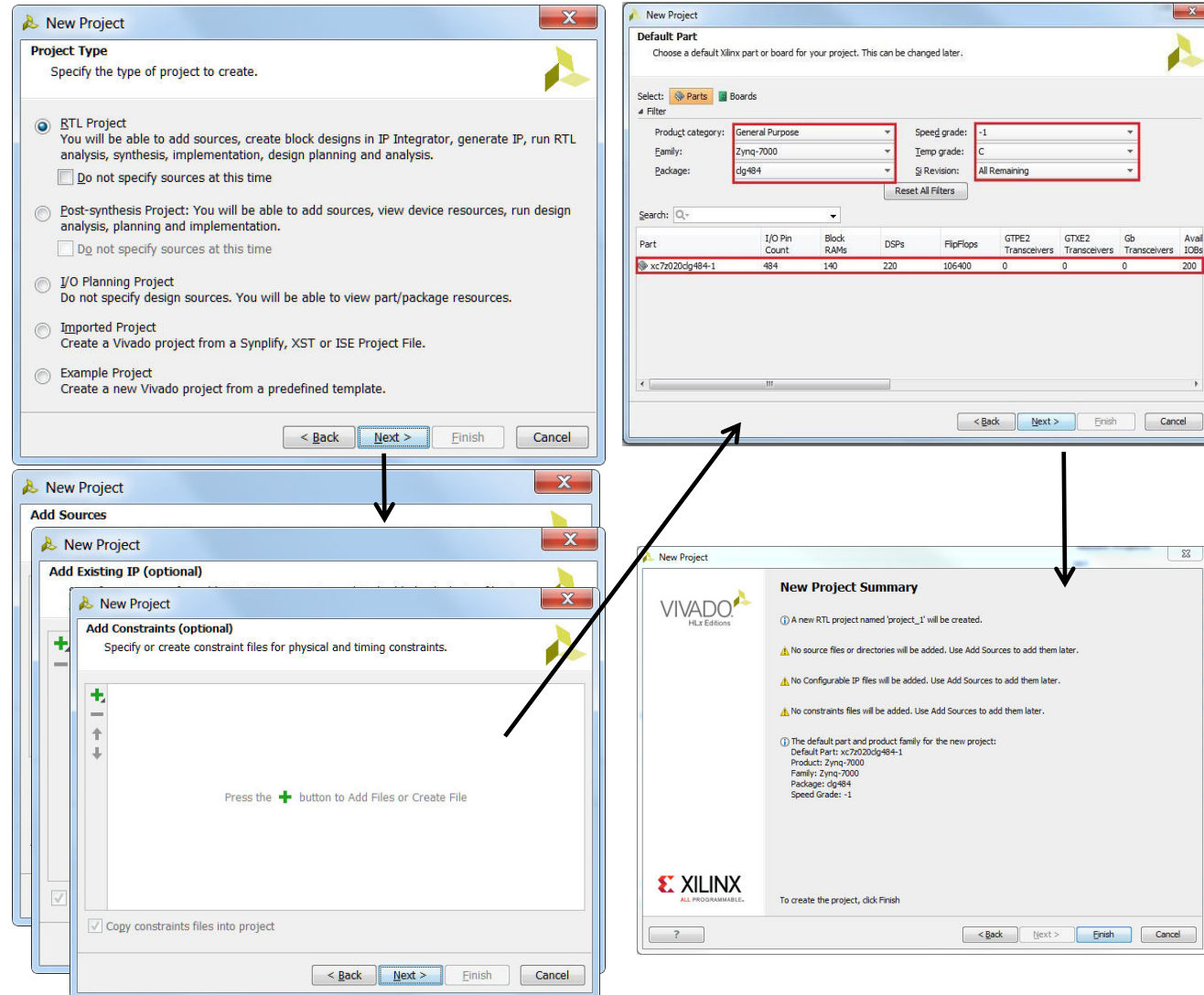
- ☒ **RTL Project**  
You will be able to add sources, create block designs in IP Integrator, generate IP, run RTL analysis, synthesis, implementation, design planning and analysis.  
☐ Do not specify sources at this time
- ☐ **Post-synthesis Project**: You will be able to add sources, view device resources, run design analysis, planning and implementation.  
☐ Do not specify sources at this time
- ☐ **I/O Planning Project**  
Do not specify design sources. You will be able to view part/package resources.
- ☐ **Imported Project**  
Create a Vivado project from a Synplify, XST or ISE Project File.
- ☐ **Example Project**  
Create a new Vivado project from a predefined template.

? < Back Next > Finish Cancel



# Creación de un proyecto (RTL/Synthesized Design)

- Definir el nombre y ubicación del proyecto
- Seleccionar los archivos fuente
  - Se puede agregar código Verilog o VHDL
- Seleccionar netlists sintetizadas
- Seleccionar archivo de restricciones
  - Puede haber más de un archivo y/o un archivo específico para las IP presintetizadas
- Referenciar los archivos originales o importarlos y copiarlos al proyecto



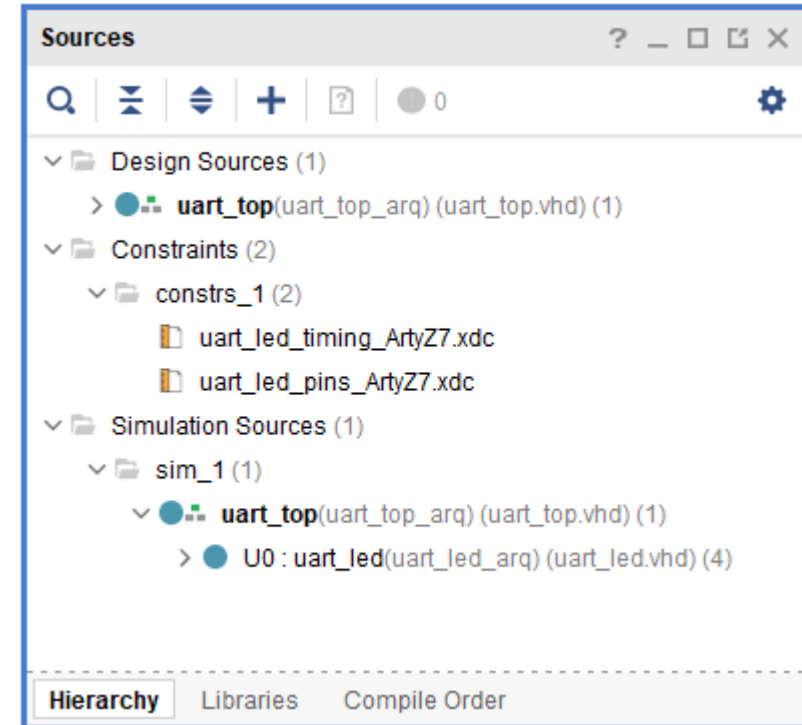
# Gestión del archivo de restricciones

## ➤ Las restricciones son un conjunto de archivos XDC

- Un proyecto puede tener más de un conjunto de restricciones
- Para ello, el conjunto a utilizar debe configurarse a “activo” (“active”)
  - Eso se hace presionando el boton derecho y seleccionando la opcion “Make Active”

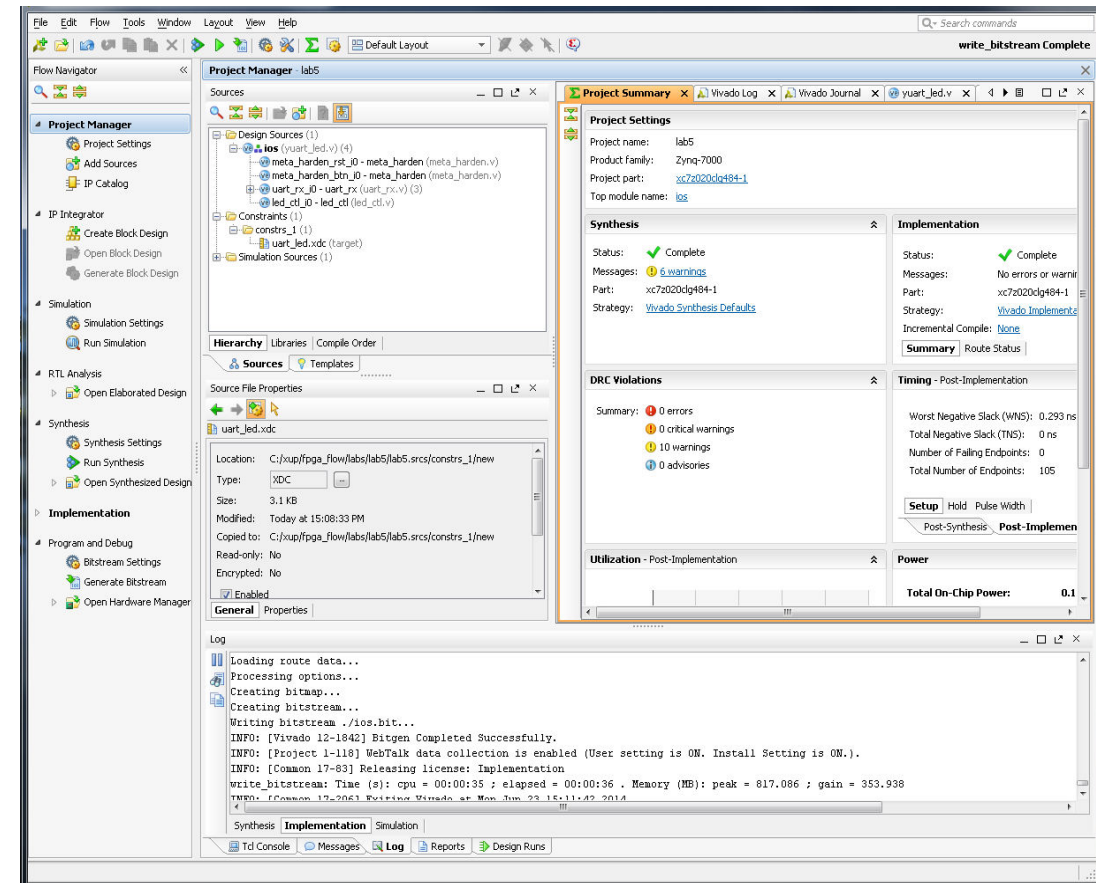
## ➤ Archivo XDC activo

- Es el archivo XDC de un conjunto de restricciones al cual se le van a agregar nuevas restricciones
  - Por ejemplo desde el asistente para restricciones (Constraints Wizard)
- El archivo XDC activo se especifica con el botón derecho y seleccionando “Set As Target Constraint File”



# Navegador del proyecto (Project Navigator)

- Se utiliza para gestionar los archivos fuente, configurar la IP y ver los detalles del proyecto
- Permite pasar por las distintas etapas del ciclo de desarrollo y aplicar las distintas herramientas
- **Visor de código fuente (Sources)**
  - Presentación jerárquica de código fuente, incluyendo los archivos de restricciones
  - Vista de librerías e IP
    - HDL y netlists, incluyendo referencias a librerías y ubicación en disco
- **Resumen del Proyecto (Project Summary)**
  - Muestra la utilización del dispositivo, temporización y demás información del diseño
- **Consola Tcl:**
  - Mensajes de Compilación, Reportes, y Ejecución

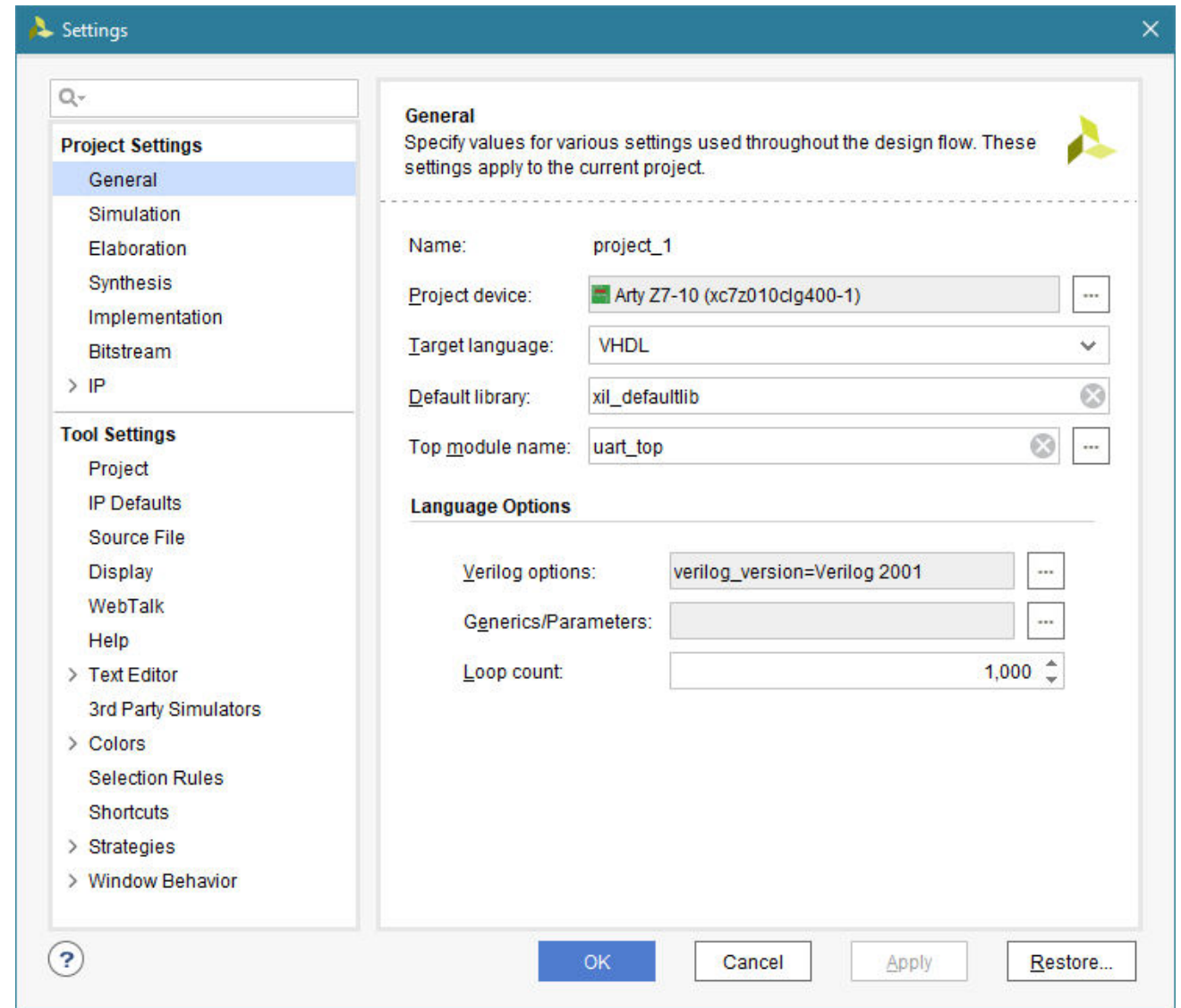


# Configuración de un Proyecto

## ➤ Configuración General:

- Seleccionar dispositivo
- Seleccionar lenguaje HDL
- Elegir herramienta de simulación (normalmente Vivado Simulator)
- Nombre del Top Module
- Opciones específicas del lenguaje hdl

## ➤ Cada módulo tiene sus propias opciones de configuración



# Navegador de Ciclo de Desarrollo (Flow Navigator) para proyectos RTL (RTL Projects)

## ➤ Configurar los archivos fuente del proyecto

- Agregar archivos fuente HDL, archivos de restricciones, archivos de simulación y bloques lógicos predefinidos (Intellectual Property – IP)

## ➤ Integrar IP (IP Integrator)

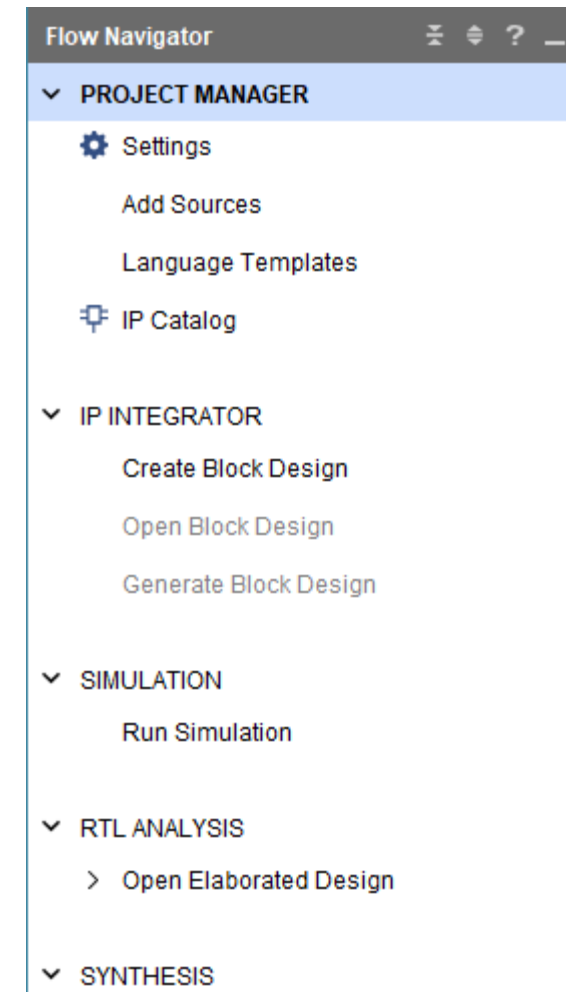
- Crear, abrir o generar un bloque lógico predefinido (IP)

## ➤ Ejecutar una simulación (Run Simulation)

- Se utiliza el simulador XSIM
- Se pueden realizar simulaciones de comportamiento (Behavioral), post-synthesis (estimación de tiempos), y post-implementation (temporización real del sistema)

## ➤ Análisis RTL (RTL Analysis)

- Abrir un diseño elaborado (Open Elaborated Design) y configurar las opciones de análisis





# Navegador de Ciclo de Desarrollo (Flow Navigator) para proyectos RTL (RTL Projects)

## ➤ Ejecutar la herramienta de síntesis (Run Synthesis)

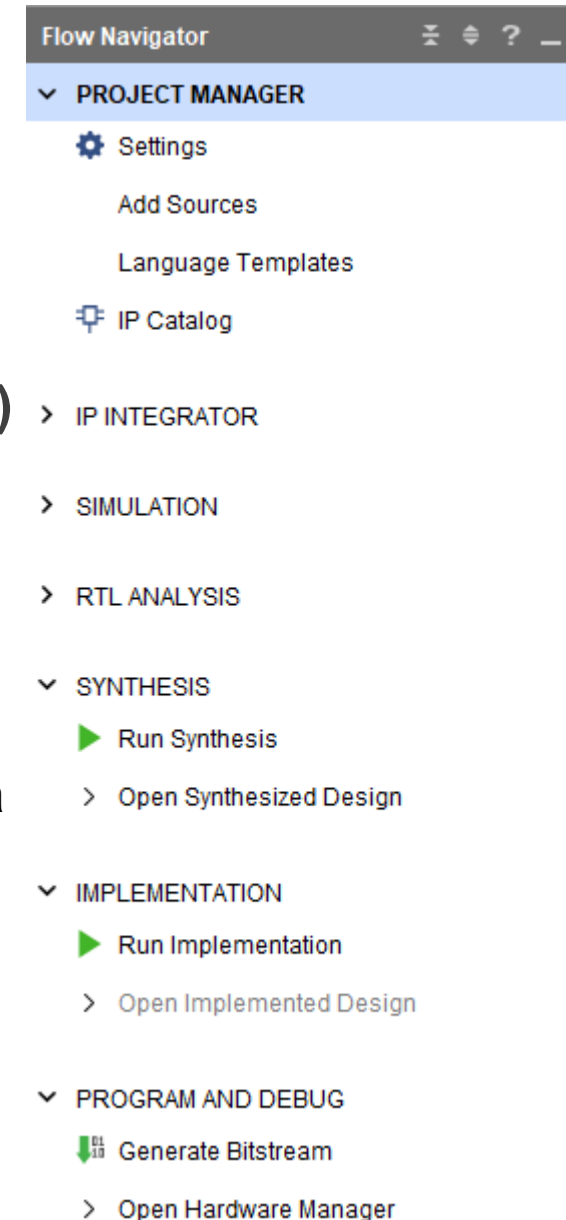
- Se busca cumplir los requerimientos de temporización
- Para cargar una netlist ya sintetizada: Open Synthesized Design button

## ➤ Ejecutar la herramienta de implementación (Run Implementation)

- Realiza la ubicación física en el dispositivo y el ruteo.
- Para cargar un sistema ya implementado: Open Implemented Design

## ➤ Ejecutar las herramientas de configuración de la FPGA y depuración

- Generate Bitstream se utiliza para generar el archivo de configuración de la FPGA
- Hardware Manager se utiliza para configurar la FPGA; también incluye herramientas para la depuración del sistema



# Simulación

- El simulador incluido (XSIM) permite simulaciones RTL, de netlist, y de temporización

- IP INTEGRATOR

- ▼ SIMULATION

- Run Simulation

- ▼ RTL ANALYSIS

- Open Elaborated Design

- ▼ SYNTHESIS

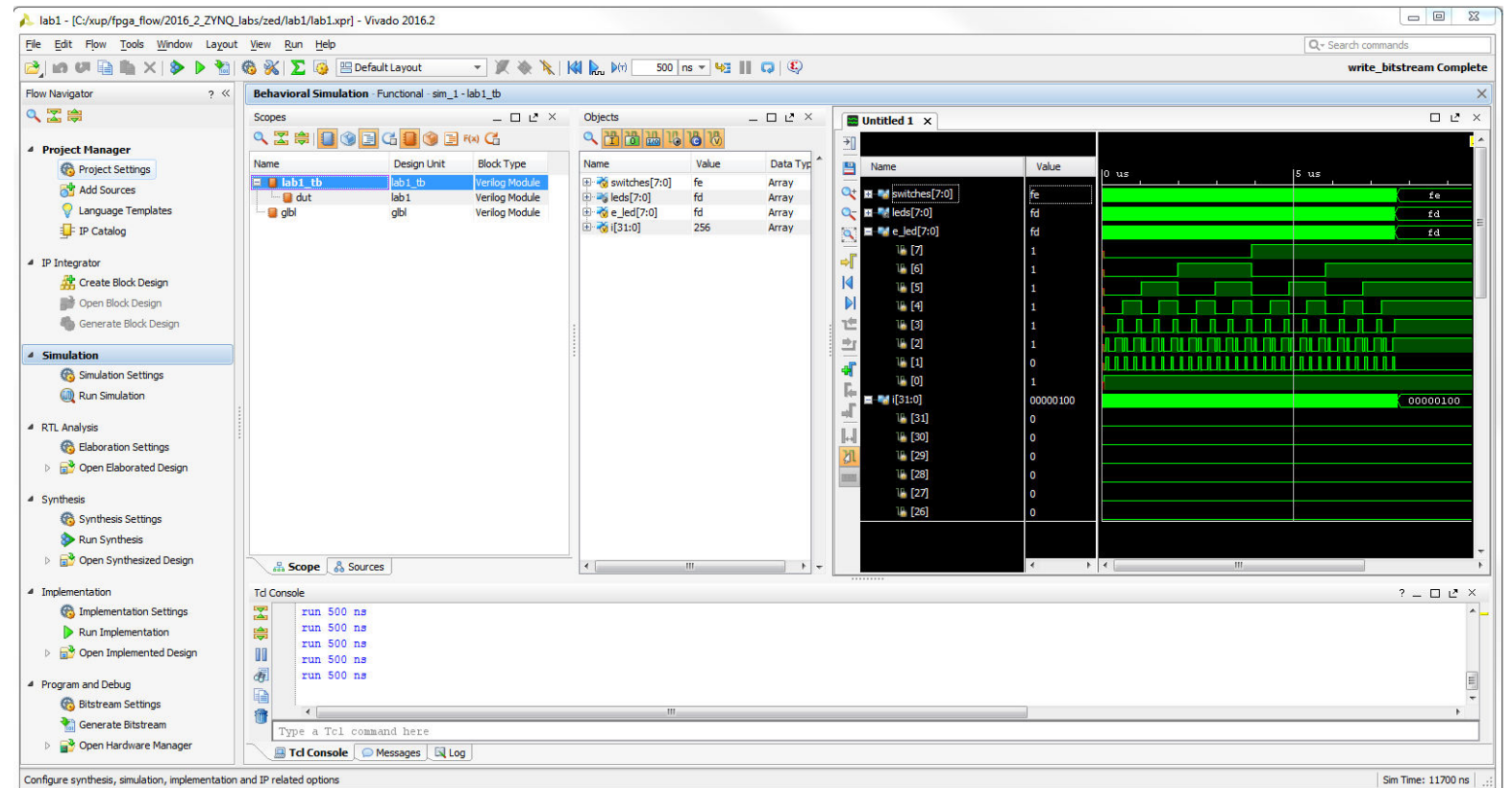
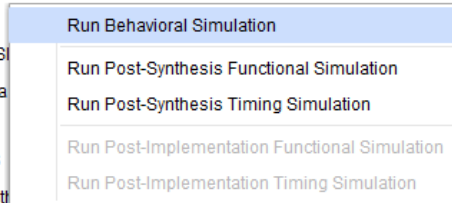
- Run Synthesis

- Open Synthesized Design

- ▼ IMPLEMENTATION

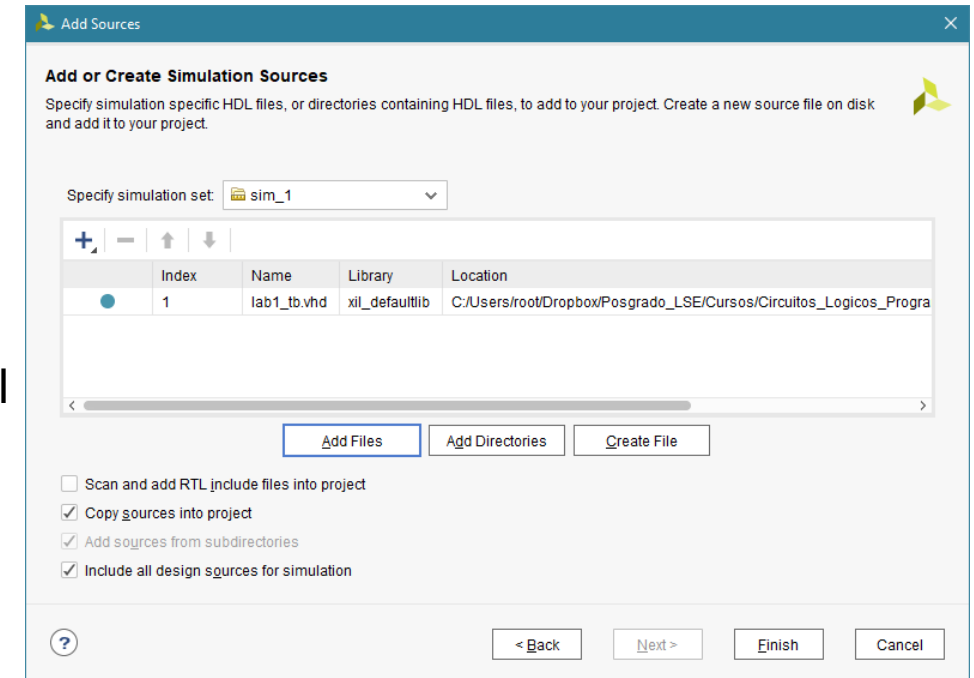
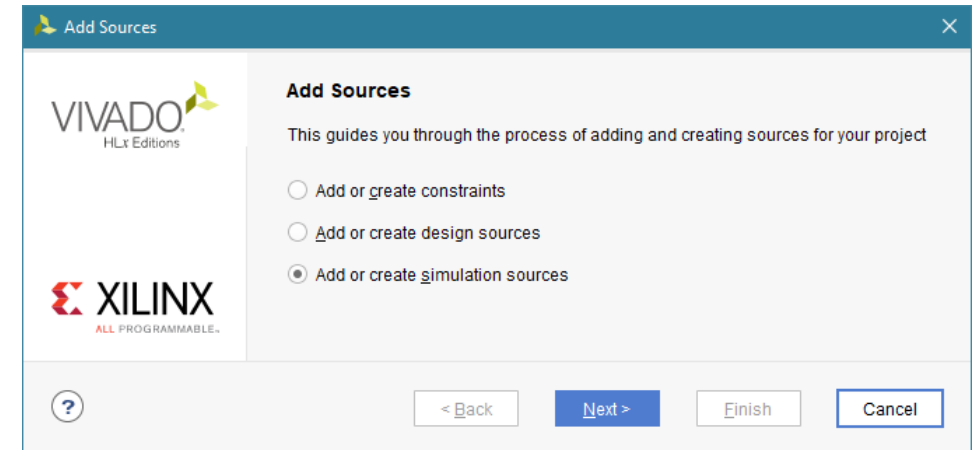
- Run Implementation

- Open Implemented Design



# Agregado de archivos de pruebas (Testbench)

- Los archivos de pruebas (\*.V or \*.VHD) se agregan en forma separada de los archivos de descripción HDL del sistema
  - En “Flow Navigator”, seleccionar *Add Sources*
  - Seleccionar *Add or Create Simulation Sources* y después “Next”
  - Presionar el símbolo “+” azul, y agregar archivos o directorios si el archivo de pruebas ya existe; o *Create File* para crear un nuevo archivo de prueba
  - Seleccionar el tipo de archivo :Verilog, Verilog Header, SystemVerilog, o VHDL
  - Navegar y seleccionar el archivo existente o ingresar el nombre de archivo para uno nuevo

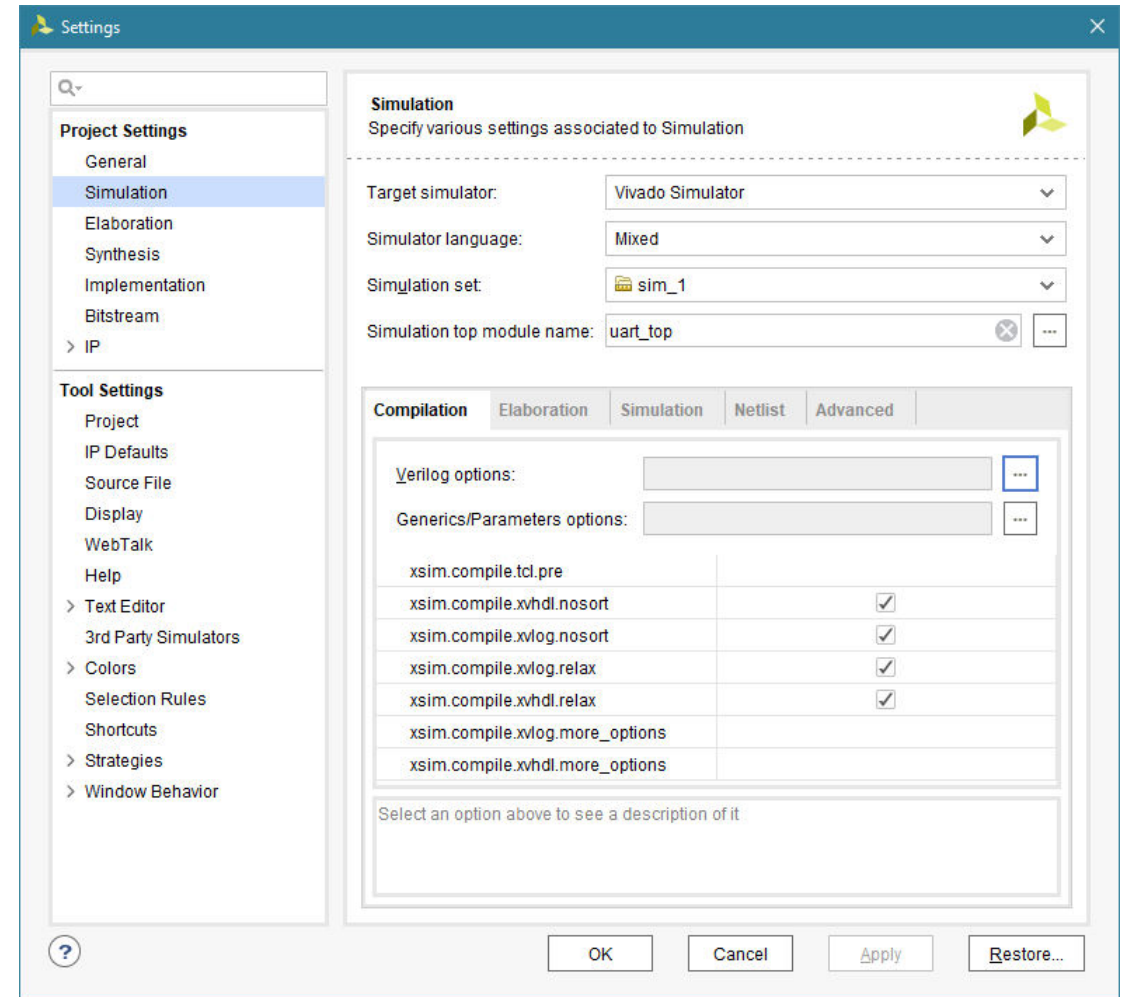


# Configuración de la Simulación

## ► Permite seleccionar opciones específicas de la compilación y simulación

- Desde *Flow Navigator*, seleccionar *Simulation Settings*
- El top module de la simulación es el archivo de pruebas (testbench) especificado
- Se pueden agregar opciones adicionales
  - Opciones de compilación en *Compilation*
  - Opciones de simulación en *Simulation*

## ► Hay una guía específica para simulación: Vivado Design Suite Simulation Guide (UG900)

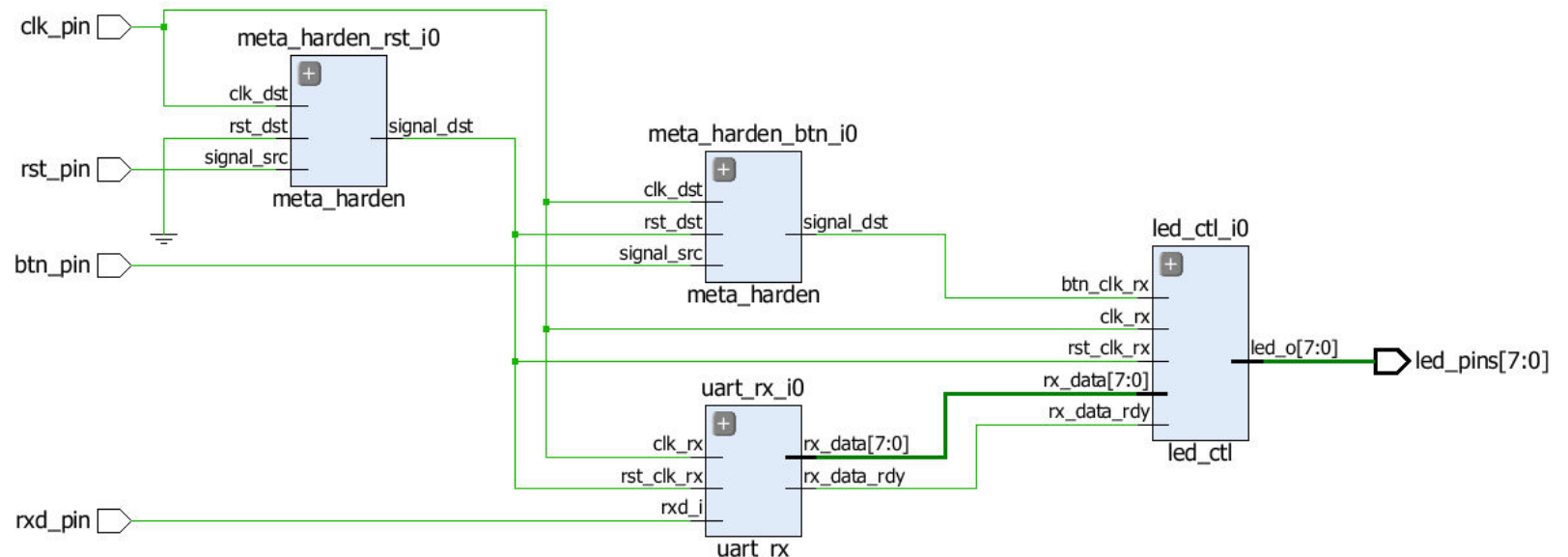
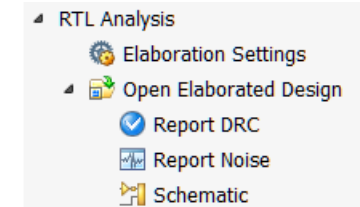


# Resultados de la Simulación

- **Gráficos con las formas de onda**
- **Se pueden agregar marcadores, medir tiempos y realizar ampliaciones para ver intervalos de tiempo menores**
- **Los Buses se pueden expandir a sus señales individuales**
- **Se pueden crear grupos de señales relacionadas mediante divisores (Dividers)**
- **Hay una Consola que muestra mensajes de salida de la simulación**
- **Si no se configura otra cosa, se ven las señales declaradas en el módulo principal (top module)**

# Después del proceso de Elaboración (Elaboration Process)

- Los fuentes, RTL Netlists, y restricciones están disponibles
- Se puede acceder a las propiedades de los items
- Se generan reportes DRC
- Está disponible una vista esquemática RTL
- Las vistas se pueden seleccionar por objetivos
  - Clock Planning
  - I/O Planning
  - Floorplanning



# Después del proceso de Síntesis (Synthesis Process)

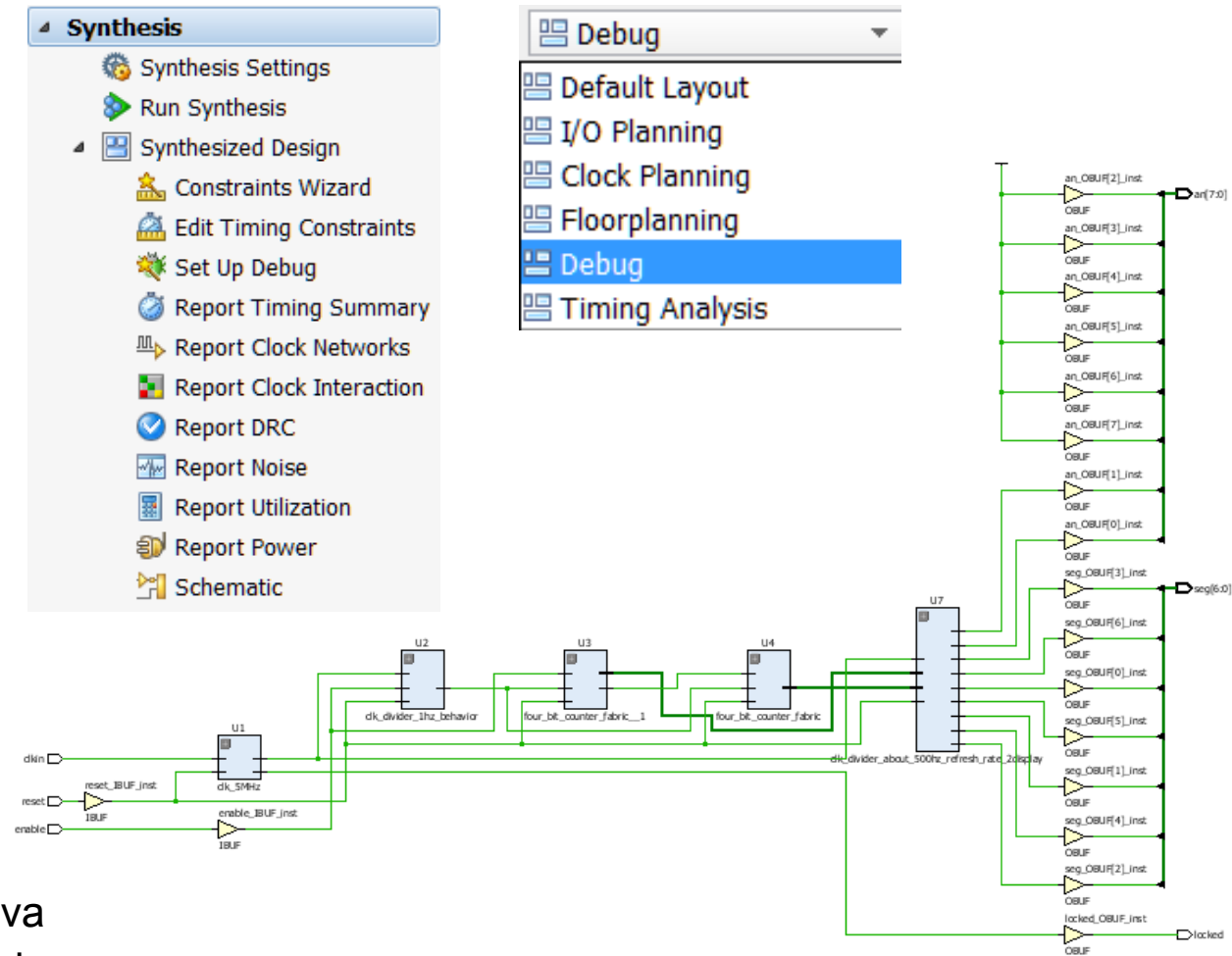
## ➤ El Flow Navigator permite acceder a:

- Constraints Wizard,
- Edit Timing Constraints,
- Set Up Debug,
- Report Timing Summary,
- Report Clock Networks,
- Report Clock Interaction,
- Report DRC,
- Report Noise,
- Report Utilization,
- Report Power,
- and Schematic utilities

## ➤ Está disponible la vista esquemática, incluyendo buffers de entrada/salida

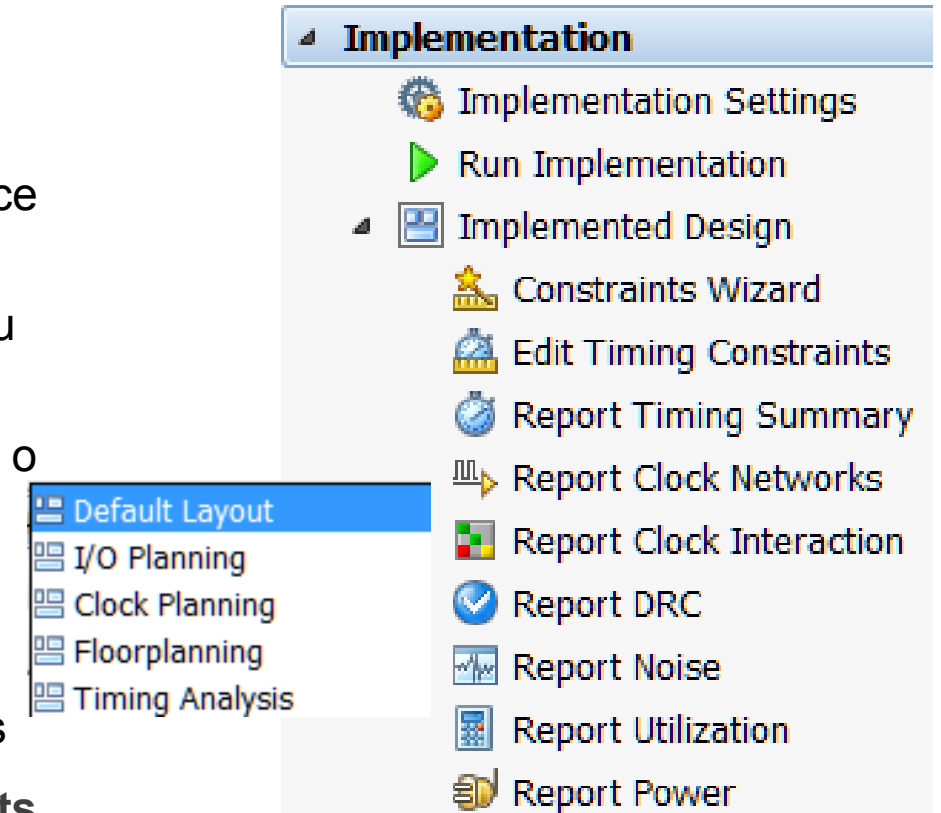
## ➤ Las vistas se pueden seleccionar por objetivo:

- I/O Planning, Clock Planning, Floorplanning, Debug, Timing Analysis
  - Toda la información de temporización es estimativa hasta que el sistema este totalmente implementado



# Después del proceso de Implementación (Implementation Process)

- **Las pestañas de Fuentes y Netlist no cambian**
  - Cuando se selecciona un recurso en la pestaña Netlist, se muestra su ubicación física en el dispositivo en la vista Device view
  - Cuando se selecciona un ruteo, la ubicación de la lógica y su conexionado se muestra en la vista Device view
  - Esto permite realizar un análisis de temporización y reubicar o rerutear recursos que no cumplen sus requisitos de temporización
- **Las vistas se seleccionan por objetivos**
  - I/O Planning, Clock Planning, Floorplanning, Timing Analysis
- **Se puede acceder al asistente de restricciones (Constraints Wizard)**
- **Los resultados de temporización están en el reporte *Timing Summary***





# Contenido

- Características del Entorno de Desarrollo Vivado
- Introducción a las distintas herramientas del Entorno
- Ciclo de Desarrollo
- **Resumen**

# Resumen

- **Características del entorno de desarrollo**
  - Se puede estimar el desempeño del diseño desde las primeras etapas
  - Los distintos reportes permiten verificar el cumplimiento de las restricciones de temporización
  - La consola Tcl permite automatizar tareas mediante scripting
- **Todas las herramientas utilizan el mismo modelo de datos a lo largo del proceso de desarrollo**
  - Esto unifica los procesos de síntesis e implementación
- **El desarrollo puede ser mediante scripting (cuando no está basado en un proyecto) y puede ser basado en un proyecto, utilizando mayoritariamente la interfaz gráfica**
  - Todos los comandos están disponibles a través de la consola Tcl
- **Las restricciones se describen en un único lenguaje (common constraint language – XDC) durante todo el desarrollo**
  - Esto permite unificar las restricciones de síntesis y de implementación
- **Para la mayoría de los diseños, las herramientas se invocan desde la interfaz gráfica (mediante botones)**
- **Hay disponibles herramientas, configuraciones avanzadas y scripting para desarrollos complejos.**