



ugr | Universidad
de **Granada**

TRABAJO FIN DE GRADO
INGENIERÍA EN INGENIERÍA INFORMÁTICA

FPGAs de Xilinx

Plataforma didáctica para desarrollo de sistemas basados en
FPGAs de Xilinx

Autora

Elena Cantero Molina (alumna)

Directora

María Begoña del Pino Prieto (tutora)



ESCUELA TÉCNICA SUPERIOR DE INGENIERÍAS INFORMÁTICA Y DE
TELECOMUNICACIÓN

GRANADA, 13 DE AGOSTO DE 2020

Índice general

1. Resumen y palabras clave	1
2. Resumen extendido y palabras clave en inglés	3
3. Motivación e introducción	5
3.1. Sistemas basados en dispositivos FPGAs	5
3.1.1. Aplicaciones	6
3.2. Niveles de síntesis automática	7
3.3. Plataformas de desarrollo	7
3.4. Estructura de la memoria	7
4. Objetivos del trabajo	9
5. Resolución del trabajo	11
5.1. Materiales	11
5.2. Metodología	19
5.3. Desarrollo de módulos hardware específicos	19
5.4. Casos prácticos	19
6. Conclusiones y vías futuras	21

Capítulo 1

Resumen y palabras clave

Capítulo 2

Resumen extendido y palabras clave en inglés

Capítulo 3

Motivación e introducción

3.1. Sistemas basados en dispositivos FPGAs

Las FPGAs (*Field Programmable Gate Arrays*), son dispositivos semiconductores basados en matrices de bloques lógicos configurables (**CLB**) que están conectados mediante interconexiones programables [6]. Estas FPGAs pueden ser reprogramadas para algún trabajo específico o para cambiar los requisitos de funcionalidad después de su fabricación. Algunas pueden ser programadas una sola vez mientras que otras pueden ser reprogramadas una y otra vez. A estos dispositivos que son programados una única vez son referidos como **OTP** (*one-time programmable*).

Field Programmable, se refiere al hecho de que su programación se hace "en el campo."^a diferencia de otros dispositivos que su funcionalidad está programada por el fabricante [4].

Hay muchos tipos diferentes de circuitos integrados digitales, entre los que destacamos **PLDs** (*Programmable Logic Devices*), **ASICs** (*Application-Specific Integrated circuits*), **ASSPs** (*Application-Specific Standard Parts*) y **FPGAs**.

Los **PLDs** son dispositivos con una arquitectura interna predeterminada por el fabricante, creados para ser configurados por ingenieros en el campo para realizar diferentes funciones. En comparación a las **FPGAs**, contiene un número limitado de puertas lógicas y las funciones que se suelen implementar son más pequeñas y simples.

Por otro lado los **ASICs** y los **ASSPs** contienen cientos de millones de puertas lógicas y se usan para crear grandes y complejas funciones. Ambos están basados en los mismo procesos de diseño y tecnologías. La única diferencia es que un **ASIC** está diseñado y fabricado para ser usado por una compañía específica, mientras que un **ASSP** es comercializado a muchos

clientes.

Así, las **FPGAs** se encuentran entre los **PLDs** y los **ASICs** porque su funcionalidad puede ser diseñada en el campo como los **PLDs**, pero pueden contener millones de puertas lógicas y ser usadas para implementar funciones complejas que previamente sólo podían ser realizadas usando **ASICs**.

El coste de un diseño de **FPGA** es mucho menor que el de uno de un **ASIC**. Al mismo tiempo, los cambios de diseño implementados son más fáciles en **FPGAs** y el tiempo de comercialización es más rápido [3].

3.1.1. Aplicaciones

A mediados del año 1980 llegaron las FPGAs, que eran usadas para implementar lógicas simples, máquinas de estados con una complejidad media y tareas de procesamiento de datos. A principios de los 90s, el mercado en el que se vendían se extendió al área de las telecomunicaciones debido a que el tamaño y sofisticación de las mismas empezaron a crecer. A finales de los 90s, el uso de las FPGAs en aplicaciones de consumo e industriales tuvo un enorme crecimiento.

Las FPGAs a menudo son utilizadas para crear prototipos de diseños ASIC o para tener un plataforma hardware donde verificar la implementación física de nuevos algoritmos [4].

Actualmente se pueden encontrar FPGAs de alto rendimiento con millones de puertas. Algunos de estos dispositivos tienen núcleos de microprocesador integrados, dispositivos de entrada-salida de alta velocidad y similares. El resultado es que actualmente las FPGAs pueden ser usadas para implementar casi cualquier cosa en distintos ámbitos como por ejemplo:

- **Aeroespacial y defensa**
- **Audio**
- **Automotriz**
- **Broadcast**
- **Electrónica**
- **Centro de datos**
- **Computación de alto rendimiento**
- **Industria**
- **Medicina**

3.2. Niveles de síntesis automática

3.3. Plataformas de desarrollo

3.4. Estructura de la memoria

Capítulo 4

Objetivos del trabajo

Capítulo 5

Resolución del trabajo

5.1. Materiales

La familia **Zynq-7000** integra un sistema completo con un procesador *ARM Cortex-A9 MPCore*. Esta familia de SoCs está diseñada para llevar a cabo aplicaciones de compleja dificultad como la video-vigilancia, sistemas inalámbricos o la automatización de fábrica.

El software de Xilinx **ISE** no estaba preparado para soportar la complejidad y capacidad de un diseño de una FPGA con un procesador ARM. *Vivado Design Suite* 5.2 fue desarrollado para FPGAs con más capacidad y permite compilaciones de descripciones basadas en *C* gracias a la funcionalidad de síntesis de alto nivel.

Dentro de la Familia Zynq 7000 encontramos la tarjeta **ZYBO** (*ZYbo BOard*). Es una plataforma de desarrollo de circuito digital y software integrado de nivel de entrada y lista para usar, y está construida alrededor del miembro más pequeño de la familia Zynq-7000, el **Z-7010**. Se basa en la arquitectura **AP SoC** (*Xilinx All Programmable System-on-Chip*), que integra un procesador de doble núcleo ARM Cortex-A9 con lógica *Xilinx 7-series FPGA*.

La Zynq 7010 Ap SoC ofrece las siguientes características 5.1:

- Procesador dual-core Cortex-A9 de 650Mhz
- Controlador de memoria DDR3 con 8 canales DMA
- Controladores periféricos de alto ancho de banda: 1G Ethernet, USB 2.0, SDIO
- Controladores periféricos de bajo ancho de banda: SPI, UART, CAN, *I²C*

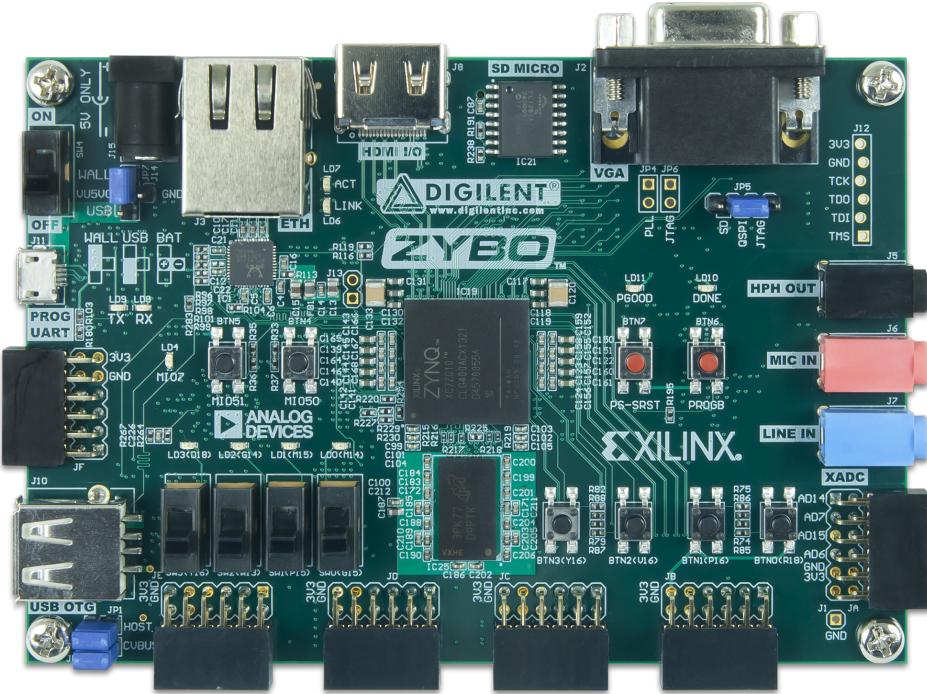


Figura 5.1: ZYBO Zynq-7000 Development Board

- Lógica Reprogramable equivalente a Artix-7 FPGA
- ZYNQ XC7Z010-1CLG400C
- Puerto HDMI
- Puerto VGA de 16 bits por pixel
- EEPROM externo
- Códec de audio con salida de auricular y micrófono
- GPIO: 6 botones, 4 interruptores, 5 LEDs
- 6 conectores Pmod

Zybo es compatible con *Vivado Design Suite* de Xilinx así como con el conjunto de herramientas ISE/EDK. Estas herramientas combinan el diseño lógico FPGA con el desarrollo software de ARM en un flujo de diseño intuitivo. Se pueden utilizar para diseñar sistemas de cualquier complejidad, desde un sistema operativo completo hasta un programa simple que controla algunos LEDs.

Vivado Design Suite es un entorno de diseño integrado (**IDE**) de Xilinx para la síntesis y análisis de diseños HDL. Vivado incluye el simulador lógico **ISim** (*ISE simulator*). Además incluye síntesis a alto nivel con una herramienta que convierte código C a lógica programable.

Está formado por 4 componentes[5]:

- **Vivado High-Level Synthesis** - Permite usar programas en *C*, *C + +* y *SystemC* en dispositivos Xilinx sin necesidad de crear un RTL manualmente. Aumenta la productividad del desarrollador y admite clases, plantillas, funciones y sobrecarga de operadores.
- **Vivado Simulator** - Es un simulador de lenguaje compilado que admite scripts TCL (*Tool Command Language*) en lenguaje mixto.
- **Vivado IP Integrator** - Permite integrar y configurar IP desde la biblioteca propia de Xilinx.
- **Vivado TCL Store** - Es un sistema de comandos para desarrollar complementos para Vivado además de agregar y modificar las capacidades de Vivado. Todas las funciones de Vivado se pueden controlar con los scripts TCL.

Para trabajar con Vivado, se puede hacer tanto trabajando con la TCL o directamente con la GUI de Vivado IDE [7].

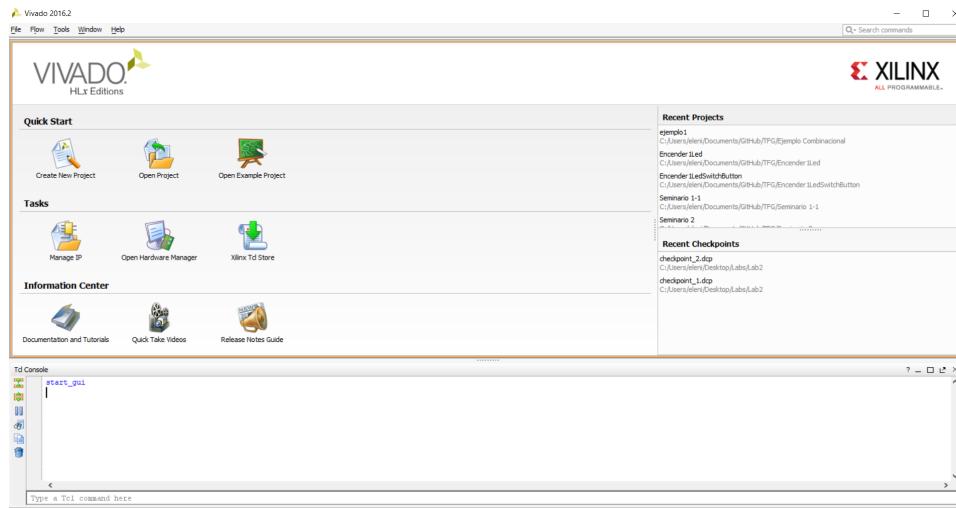


Figura 5.2: Vivado IDE

La sección **Quick Start** nos proporciona fácil acceso a la creación de un nuevo proyecto, abrir proyectos existentes o abrir proyectos de ejemplo

ofrecidos por Xilinx. Además, en la sección **Recent Projects** se pueden abrir proyectos usados recientemente.

En la sección **Tasks** encontramos el acceso a **Manage IP** que nos permite crear una ubicación IP para configurar y administrar IPs de forma remota. Se puede usar el catálogo de IP de Vivado para personalizar la IP. **Open Hardware Manager** nos ayuda a programar el diseño en el dispositivo. **Xilinx TCL Store** es un repositorio de código TCL. Da acceso a múltiples scripts para resolver problemas y mejorar la productividad.

La última sección es **Information Centre** donde se encuentra el acceso directo a la documentación, tutoriales y videos sobre lo que se puede hacer con esta herramienta.

Los componentes principales de la imagen 5.3 son:

1. *Menu Bar* 5.4
2. *Main Toolbar* 5.6
3. *Flow Navigator* 5.7
4. *Layout Selector* 5.8
5. *Data Windows Area* 5.10
6. *Workspace* 5.11
7. *Menu Command Search Field* 5.5
8. *Project Status Bar* 5.9
9. *Results Windows Area* 5.12

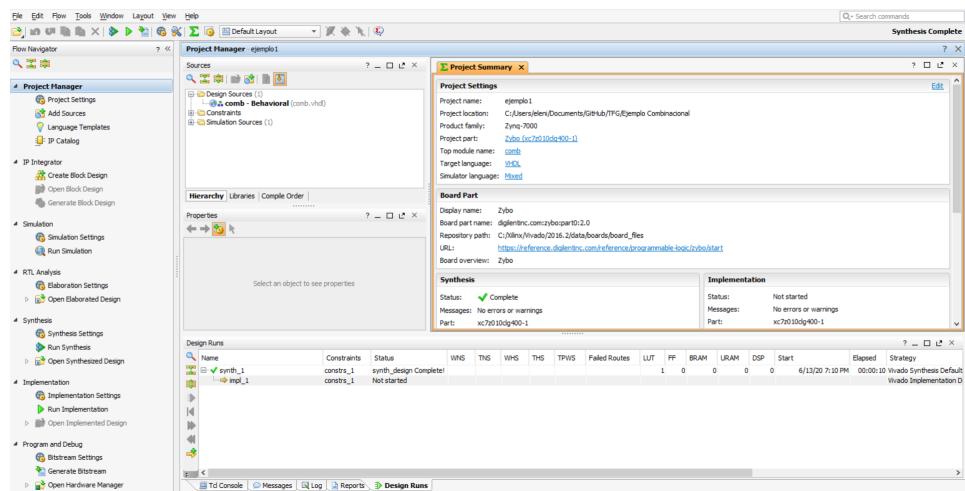


Figura 5.3: Entorno Principal Vivado IDE

Menu Bar nos da acceso a los comandos de Vivado IDE. Normalmente, cuando se inicia un proyecto, no todos los comandos están disponibles, sino que algunos se muestran cuando el diseño está activo.



Figura 5.4: *Menu Bar*

Menu Command Search Field se encuentra a la derecha del anterior y permite localizar y ejecutar un comando de manera más rápida. La lista de comandos que aparecen en la búsqueda están basados en el contexto del proyecto del diseño actual.



Figura 5.5: *Menu Command Search Field*

Main Toolbar nos da acceso a los comandos más usados en Vivado IDE. Si se pone el cursor en alguno de estos comandos, Vivado ofrece más información acerca del mismo.



Figura 5.6: *Main Toolbar*

Flow Navigator permite acceder a comandos y herramientas que van desde abrir diseños a crear un archivo bitstream. Las diferentes secciones permiten hacer lo siguiente:

- **Project Manager:** Cambio de ajustes generales, añadir o crear archivos o abrir el Catálogo de IPs
- **IP Integrator:** Crear, abrir o generar un bloque de diseño.
- **Simulation:** Cambio de ajustes de simulación o simular un diseño activo.
- **RTL Analysis:** Abrir un diseño elaborado o generar un diseño de diagrama de circuitos RTL.
- **Synthesis:** Cambio de ajustes de síntesis, sintetizar un diseño activo o abrir el diseño sintetizado.

- **Implementation:** Cambio de ajuste de implementación, implementar un diseño activo o abrir el diseño implementado.
- **Program and Debug:** Cambio de ajustes del bitstream, generar un archivo bitstream o abrir una sesión hardware.

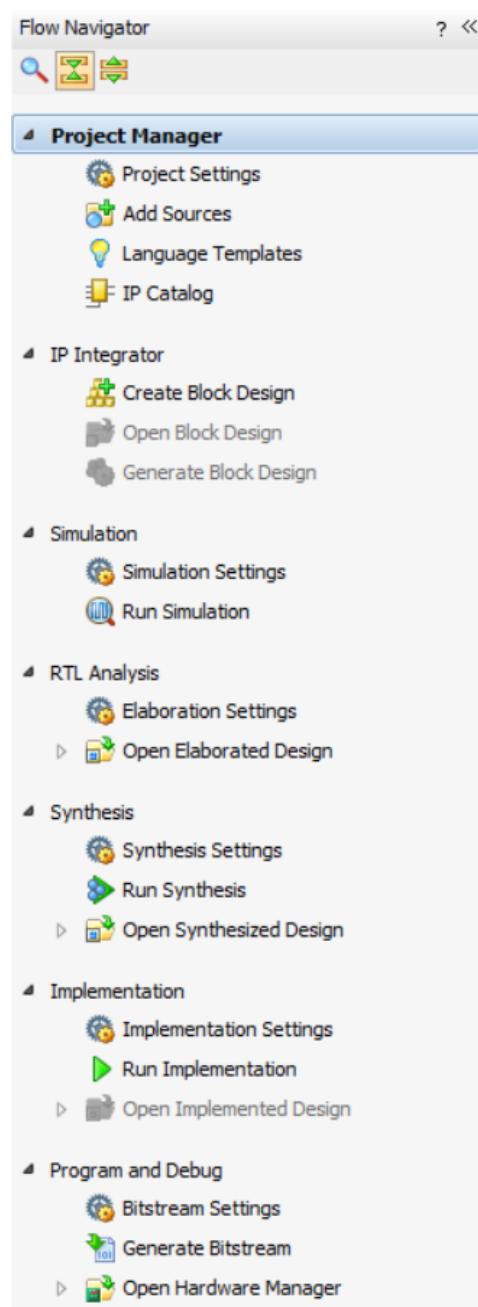


Figura 5.7: *Flow Navigator*

Layout Selector proporciona el diseño de ventanas predefinidas para facilitar el proceso de diseño. Entre las opciones tenemos:

- **Default Layout:** Analización del diseño con el mínimo número de ventanas
- **I/O Planning:** Definición de restricciones de ubicación I/O y colocación de puertos.
- **Clock Planning:** Planificación y colocación de los recursos del reloj del diseño.
- **Floorplanning:** Gestionar particiones y tareas jerárquicas.
- **Timing Analysis:** Ejecutar informes de tiempo y analizarlo.

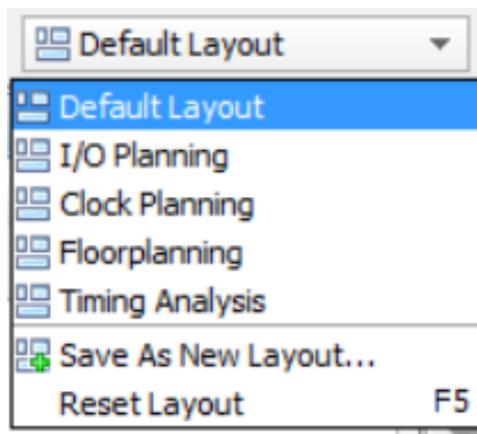


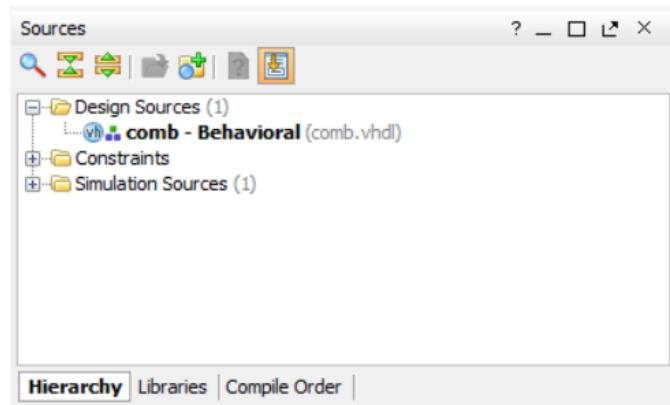
Figura 5.8: *Layout Selector*

Project Status Bar da información sobre el estado actual del diseño activo.

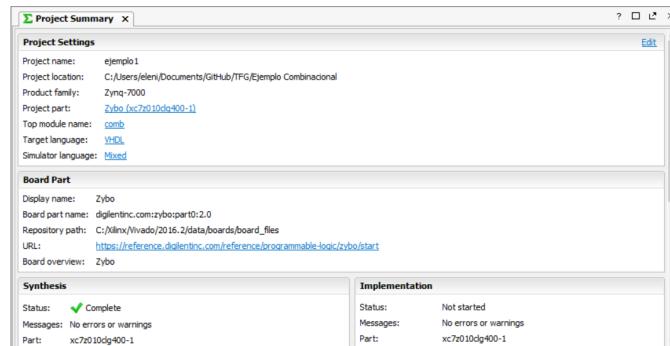


Figura 5.9: *Project Status Bar*

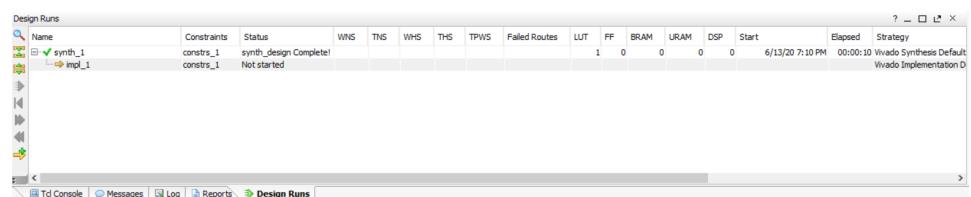
Data Windows Area muestra información sobre los archivos que forman el diseño.

Figura 5.10: *Data Windows Area*

Workspace muestra ventanas como el editor de textos o el diseño del diagrama de circuitos, entre otros.

Figura 5.11: *Workspace*

Results Windows Area presenta los resultados de los comando ejecutados. Además se muestran distintas ventanas, como *Tcl Console*, *Messages*, *Log*, *Reports* y *Design Runs*.

Figura 5.12: *Results Windows Area*

5.2. Metodología

5.3. Desarrollo de módulos hardware específicos

5.4. Casos prácticos

Capítulo 6

Conclusiones y vías futuras

Bibliografía

- [1] Digilent. Zybo fpga board reference manual. https://reference.digilentinc.com/_media/zybo:zybo_rm.pdf.
- [2] Clive Maxfield. Conceptos fundamentales de los fpga: ¿qué son los fpga y por qué son necesarios? <https://www.digikey.es/es/articles/fundamentals-of-fpgas-what-are-fpgas-and-why-are-they-needed>.
- [3] Clive Maxfield. *The design warrior's guide to FPGAs: devices, tools and flows*. Elsevier, 2004.
- [4] Clive Maxfield. *FPGAs: instant access*. Elsevier, 2011.
- [5] Wikipedia. Xilinx vivado. https://en.wikipedia.org/wiki/Xilinx_Vivado.
- [6] Xilinx. Field programmable gate array (fpga). <https://www.xilinx.com/products/silicon-devices/fpga/what-is-an-fpga.html>.
- [7] Xilinx. Vivado design suite user guide : Using the vivado ide. https://www.xilinx.com/support/documentation/sw_manuals/xilinx2016_2/ug893-vivado-ide.pdf.