



ugr | Universidad
de **Granada**

TRABAJO FIN DE GRADO
INGENIERÍA EN INGENIERÍA INFORMÁTICA

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, 16 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
3.2. Niveles de síntesis automática	8
3.3. Plataformas de desarrollo	8
3.4. Estructura de la memoria	9
4. Objetivos del trabajo	11
5. Resolución del trabajo	13
5.1. Materiales	13
5.1.1. Familia Zynq-7000	13
5.1.2. Tarjeta Zybo	13
5.1.3. Vivado	16
5.2. Metodología	23
5.3. Desarrollo de módulos hardware específicos	23
5.4. Casos prácticos	23
6. Conclusiones y vías futuras	25

Índice de figuras

3.1. <i>Arquitectura de una FPGA</i>	6
5.1. ZYBO Zynq-7000 Development Board	14
5.2. Arquitectura Zynq AP SoC	15
5.3. Vivado IDE	17
5.4. Entorno Principal Vivado IDE	18
5.5. <i>Flow Navigator</i>	20
5.6. <i>Default Layout</i>	21
5.7. <i>I/O Planning</i>	21
5.8. <i>Clock Planning</i>	22
5.9. <i>Floorplanning</i>	22
5.10. <i>Timing Analysis</i>	22

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 [7].

Los **CLB** constan de celdas lógicas llamadas “*Slices*”, formadas por LUTs (*tablas de consulta*), flip-flops y multiplexores de entrada y salida (Figura 3.1). Una LUT almacena una lista de salidas lógicas para cualquier combinación de entradas.

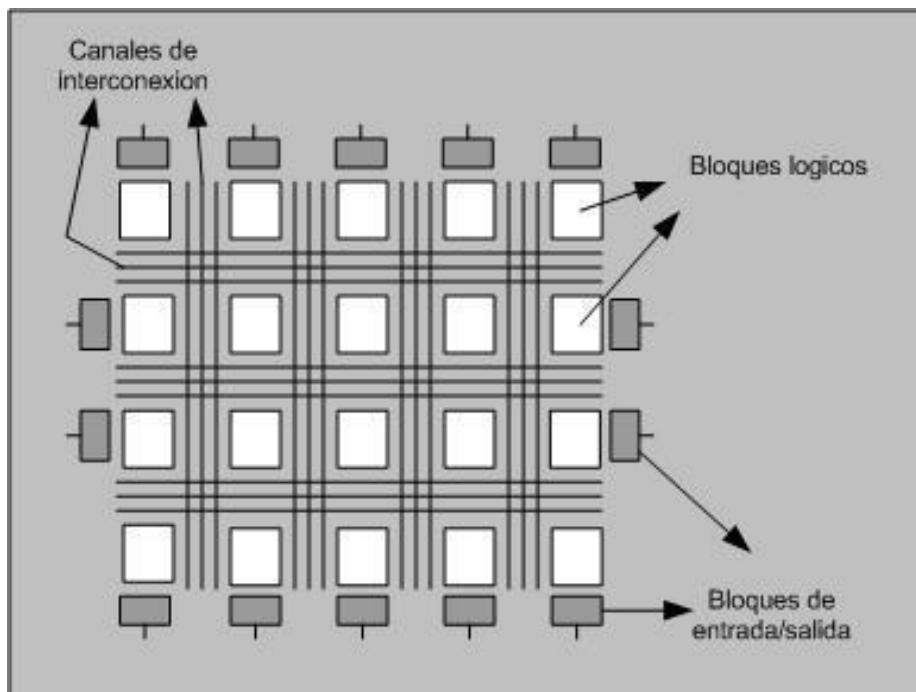


Figura 3.1: Arquitectura de una FPGA

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 mismos procesos de diseño y tecnologías y pueden ser usados por millones de usuarios y compañías. La única diferencia es que un **ASIC** está diseñado y fabricado para una aplicación en concreto, mientras que un **ASSP** lo está para un dominio de aplicaciones.

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].

Las FPGAs **SoC**(*System-on-chip*) tienen una gran capacidad de procesamiento para adaptarse a diferentes aplicaciones. Un SoC de bajo costo y consumo se puede enfocar en aplicaciones de gran volumen como tarjetas de procesamiento de vídeo o protocolo de puentes. Sin embargo, hay otros SoCs que se enfocan en aplicaciones de alto rendimiento en comunicaciones o computación de alto rendimiento.

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**
- **Emulación y Prototipado**
- **Audio**
- **Automoción**
- **Broadcast**
- **Electrónica de consumo**
- **Centro de procesamiento de datos**
- **Computación de alto rendimiento**
- **Industria**
- **Medicina**
- **Comunicaciones**
- **Inteligencia Artificial**

- Procesamiento de imágenes
- Seguridad

3.2. Niveles de síntesis automática

Una de las características principales de un lenguaje de descripción hardware es que a partir de una descripción RTL se puede generar un circuito físico. La síntesis es el paso de un nivel de descripción a uno de nivel inferior.

La síntesis física consiste en la ubicación, es decir, decidir dónde colocar todos elementos lógicos, y en el enrutamiento, en el que se decide cómo se interconectan los elementos en la FPGA.

La síntesis RT-lógica es un proceso en el se crea un diseño RTL (*Register-Transfer Level*), que es una abstracción del diseño en el que se modela el circuito digital, y luego esa representación RTL es convertida a una mezcla de registros y ecuaciones booleanas equivalentes.

La principal diferencia entre síntesis RT y síntesis de alto nivel es que la primera parte de una descripción en la que de forma explícita se especifican las operaciones que deben realizarse en cada ciclo de reloj, mientras que la planificación de operaciones en ciclos de reloj se realiza de forma automática en la segunda.

La síntesis de alto nivel une hardware y software de manera que los diseñadores hardware pueden trabajar con un alto nivel de abstracción y los desarrolladores software pueden acelerar las pas partes computacionalmente complejas de sus algoritmos en una FPGA.

El uso de una metodología de diseño de síntesis de alto nivel permite desarrollar algoritmos con respecto a la implementación ya que consume tiempo de desarrollo, validar el correcto funcionamiento de un diseño de forma más rápida que con lenguajes de descripción hardware tradicionales o crear implementaciones hardware de alto rendimiento

3.3. Plataformas de desarrollo

Actualmente hay muchas empresas que fabrican FPGAs, pero en el top 5 se pueden encontrar *Xilinx*, *Altera*, *Lattice Semiconductor*, *Microsemi* (antiguo *Actel*) y *QuickLogic*. Tanto Xilinx como Altera ocupan un 89 % del mercado, siendo Xilinx el líder desde hace muchos años. Xilinx tiene bastante variedad de FPGAs en cuanto a coste y rendimiento. Actualmente, la serie *Virtex* y la serie *Zynq-7000* de SoC ocupan el mercado de gama alta, la serie *Kintex* de gama media y la serie *Artix* de gama baja junto con la *Spartan* que ha sido retirada del mercado.

La serie *Virtex* integra lógica **FIFO** y **ECC**, bloques **Ethernet MAC**, bloques **DSP** (*Procesador de señales digitales*), controladores **PCI-Express**. Además incluye hardware embebido con una función fija para funciones que se usan comúnmente como multiplicadores o memoria.

La serie *Kintex* se caracteriza por consumir menos energía que la serie anterior, incluyendo alto rendimiento y elementos necesarios para aplicaciones que tengan mucho volumen.

La serie *Artix* se basa en la arquitectura unificada de la serie *Virtex*. Esta serie está diseñada para aplicaciones con rendimiento de bajo consumo.

Dependiendo de la síntesis que queramos realizar podemos encontrar distintos software:

■ **Herramientas de síntesis RT-lógica:**

- *Synplify Pro, Synplify Premier (Synopsis)*
- *Precision RTL Plus, LeonardoSpectrum (Mentor Graphics)*
- *Quartus (Altera)*
- *Vivado (Xilinx)*

■ **Herramientas de síntesis de alto nivel:**

- *Synphony C Compiler (Synphony)*
- *Impulse coDeveloper (Impulse C)*
- *Vivado High Level Synthesis (Xilinx)*
- *SDSoc (Xilinx)*
- *SDAccel (Xilinx)*
- *Intel SDK for OpenCL (Intel Altera)*
- *Intel HLS Compiler (Intel Altera)*

La última herramienta comercializada por Xilinx, *Vitis* es un entorno de desarrollo de aplicaciones que sustituye a las herramientas *SDSoc* y *SDAccel* que permite utilizar tanto FPGAs en tarjetas aceleradoras on premise y en la nube, como FPGAs con procesadores empotrados. Incorpora una herramienta de síntesis de alto nivel (**Vitis HLS**) que pretende reducir las diferencias entre escribir funciones para su ejecución software o para su implementación hardware. Y se dispone incluso de bibliotecas con funciones prediseñadas para diferentes dominios de aplicación (inteligencia artificial, visión, etc.).

Las plataformas de desarrollo con propósito académico que comercializa Xilinx son [9]:

- **7-series** - *Spartan-7, Artix-7, Kintex-7, Virtex-7*
- **Zynq** - *ZYBO, ZYBO Z7, ZedBoard*
- **Spartan** - *Spartan-6, Spartan-3E*
- **Virtex** - *Virtex-6, Virtex-5, Virtex-4, Virtex-2P*

3.4. Estructura de la memoria

Capítulo 4

Objetivos del trabajo

Capítulo 5

Resolución del trabajo

5.1. Materiales

5.1.1. Familia Zynq-7000

La familia **Zynq-7000** integra un sistema completo con un procesador *ARM Cortex-A9 MPCore* con una lógica genérica que permite la configuración de módulos hardware específicos. Esta familia de SoCs está diseñada para llevar a cabo aplicaciones de compleja dificultad como la video-vigilancia o sistemas inalámbricos.

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* (Figura 5.3) fue desarrollado para FPGAs con más capacidad y permite compilaciones de descripciones basadas en *C* gracias a la funcionalidad de síntesis de alto nivel.

5.1.2. Tarjeta Zybo

Una FPGA de la familia Zynq 7000 es incluida en tarjeta **ZYBO** (*ZYBO Board*). Es una plataforma de desarrollo de circuito digital, 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 (Figura 5.1) [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, I^2C

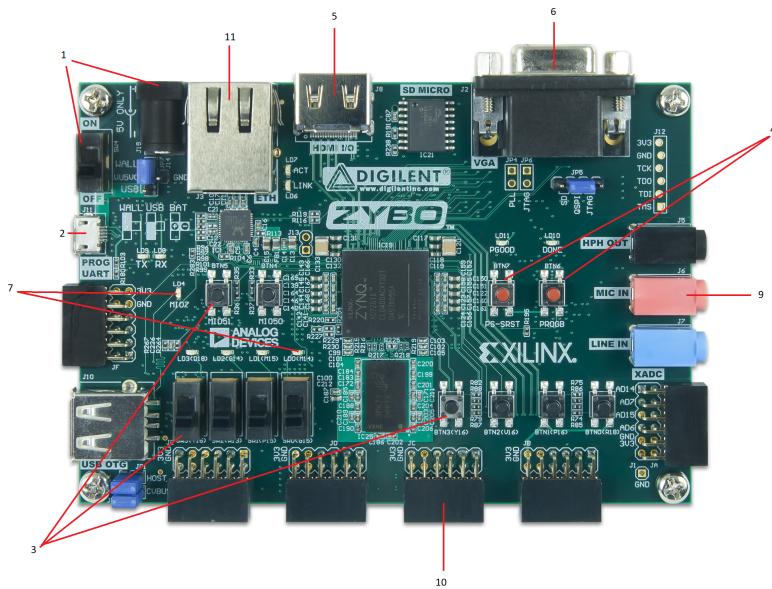


Figura 5.1: ZYBO Zynq-7000 Development Board

- 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

La arquitectura Zynq AP SoC está dividida en dos partes (Figura 5.2), el sistema de procesamiento (*PS*) y la lógica programable (*PL*). La *PL* usada es parecida a la de la FPGA *Xilinx 7-series Artix*, excepto porque contiene buses y puertos dedicados que hacen que esté acoplado fuertemente al *PS*. Además, la *PL* no tiene la misma configuración hardware como las FPGAs 7-series y tiene que ser configurada por el procesador o por el puerto JTAG.

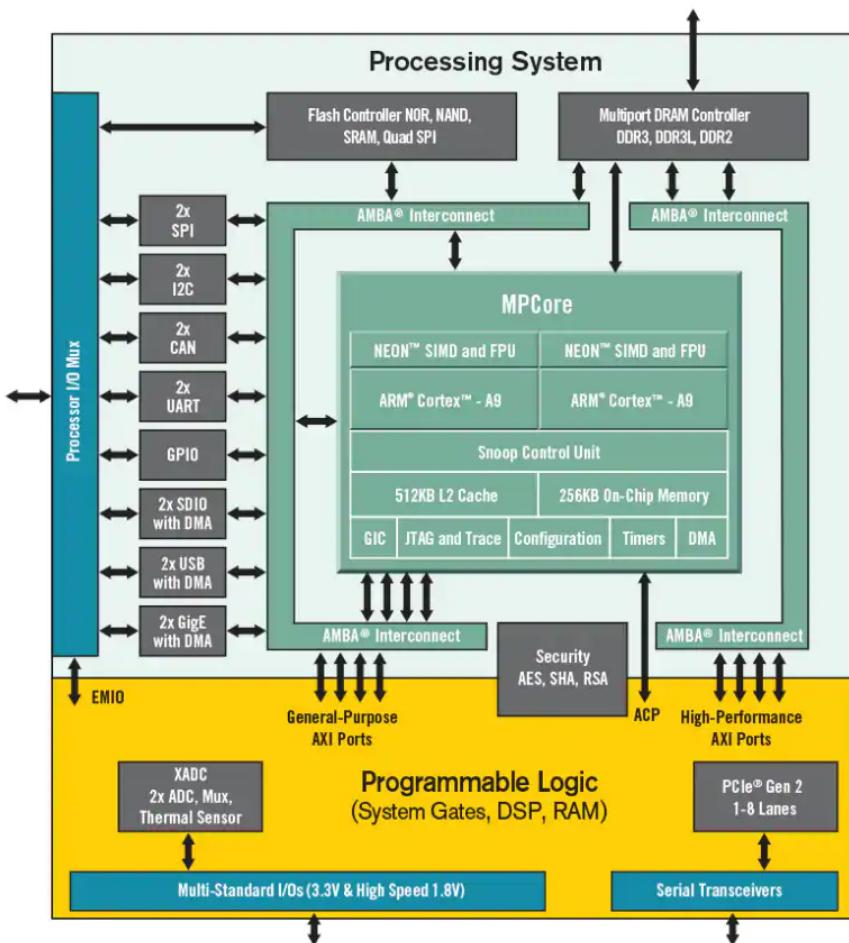


Figura 5.2: Arquitectura Zynq AP SoC

El PS consiste en un conjunto de componentes como la *APU* (Unidad de Procesamiento de Aplicaciones) que incluye dos procesadores Cortex-A9, *AMBA* (Arquitectura de Bus de Microcontrolador Avanzada), Controlador de Memoria *DDR3*, y varios controladores periféricos con las entradas y salidas multiplexadas a 54 pines dedicados (*MIO*). Los controladores periféricos están conectados al procesador mediante la interconexión *AMBA* y la PL está conectada de la misma manera.

Los elementos que forman la tarjeta Zybo son (Figura 5.1):

1. Interruptor y Conector de alimentación
2. Botones e interruptores
3. Botones de reset
4. Puerto HDMI

- 5. Puerto VGA
- 6. LEDs
- 7. Conectores de audio
- 8. Conectores Pmod
- 9. Conector Ethernet

La tarjeta Zybo incluye cuatro interruptores, cuatro botones y cuatro LEDs individuales a la PL. Además hay 2 botones y un LED conectados directamente al PS vía través de pines MIO. Adicionalmente hay un LED de encendido de la tarjeta, otros dos para el estado del puerto USB y un último LED para el estado de la programación de la FPGA.

Uno de los **botones de reset** reestablece la PL, que permanecerá sin configurar hasta ser reprogramado de nuevo. El otro reinicia el dispositivo sin afectar al entorno de depuración.

Nos encontramos con tres **Conectores de audio**, dos entradas para micrófono y línea estéreo y una salida para auriculares.

El **Puerto VGA** (*Video Graphics Array*) es una interfaz que recibe señal a través de un conector VGA. Éste se suele usar para conectar dispositivos. Está compuesto de 15 pines y cada uno tiene su propia función, entre las cuales está la de transferir los colores rojo, azul y verde, la sincronización horizontal y la sincronización vertical.

El **Puerto HDMI** es un conector de entrada y salida con el que se puede transmitir vídeo compatible con HDMI o DVI.

Los **Conectores Pmod** (*Módulos Periféricos*) son unos conectores con estándares de módulos periféricos, para ampliar la capacidad de la lógica programable. Se comunican mediante 6,8 ó 12 pines para transportar señales de control digital. En nuestro caso, son 12 pines (2x6). Hay 6 conectores Pmod con distinto comportamiento en esta tarjeta y cada uno pertenece a una de las cuatro categorías, “standard”, “MIO connected”, “XADC” y “high-speed”.

5.1.3. Vivado

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. 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 su propio simulador lógico y además está la posibilidad de usar otros simuladores como *ModelSim*, *Mentor Questa*, *Cadence IES* o *Synopsys VCS*. Además incluye síntesis a alto nivel con una herramienta que convierte código C a lógica programable.

Está formado por 4 componentes:

- **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 controlado por eventos de lenguaje de descripción hardware (**HLD**) que admite simulación de comportamiento y tiempos. Además admite scripts TCL (*Tool Command Language*) en lenguaje mixto, es decir, admite lenguajes como *Verilog*, *SystemVerilog* y *VHDL*.
- **Vivado IP Integrator** - Permite integrar y configurar IP (“*Intellectual Property*”) 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.

En concreto, la versión que se ha utilizado es con Vivado 2016.2. Para trabajar con él, se puede hacer tanto trabajando con la TCL o directamente con la GUI de Vivado IDE [8].

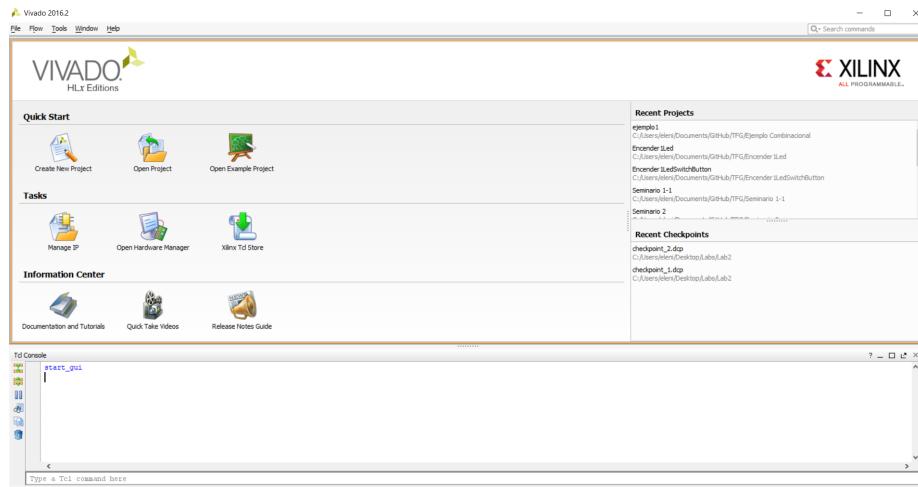


Figura 5.3: Vivado IDE

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 ver el catálogo de IP, personalizar IP y generar productos de salida. **Open Hardware Manager** nos permite conectar la tarjeta y descargar un programa en el dispositivo FPGA. **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 Figura 5.4 son:

1. *Menu Bar*
2. *Main Toolbar*
3. *Flow Navigator*
4. *Layout Selector*
5. *Data Windows Area*
6. *Workspace*
7. *Menu Command Search Field*
8. *Project Status Bar*
9. *Results Windows Area*

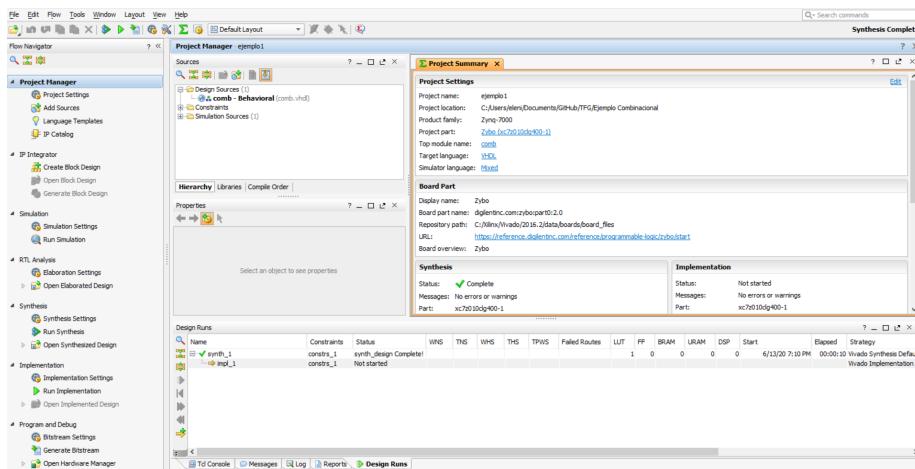


Figura 5.4: Entorno Principal Vivado IDE

Flow Navigator (Figura 5.5) 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 ventana para conectar la tarjeta FPGA y programarla.

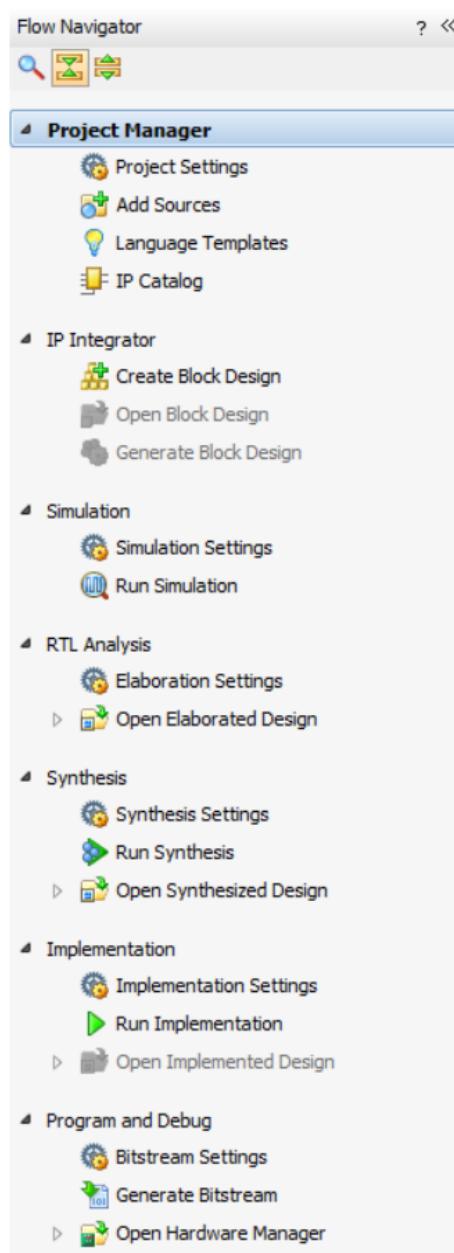


Figura 5.5: *Flow Navigator*

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

- **Default Layout:** Muestra el diseño con el mínimo número de ventanas, con un resumen global del diseño (Figura 5.6).
- **I/O Planning:** Definición de restricciones de ubicación I/O y colocación

de puertos (Figura 5.7).

- **Clock Planning:** Planificación y colocación de los recursos del reloj del diseño (Figura 5.8).
- **Floorplanning:** Gestionar particiones y tareas jerárquicas (Figura 5.9).
- **Timing Analysis:** Ejecutar informes de tiempo y analizarlo (Figura 5.10).

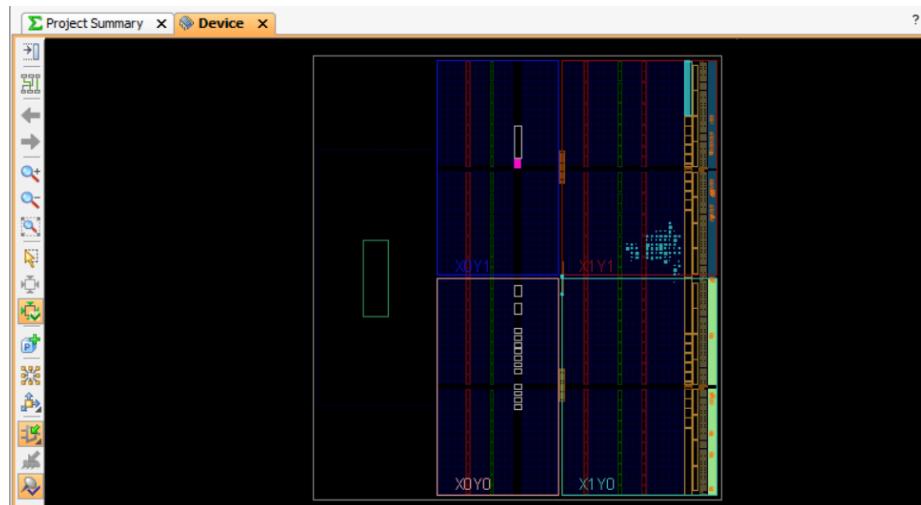


Figura 5.6: *Default Layout*

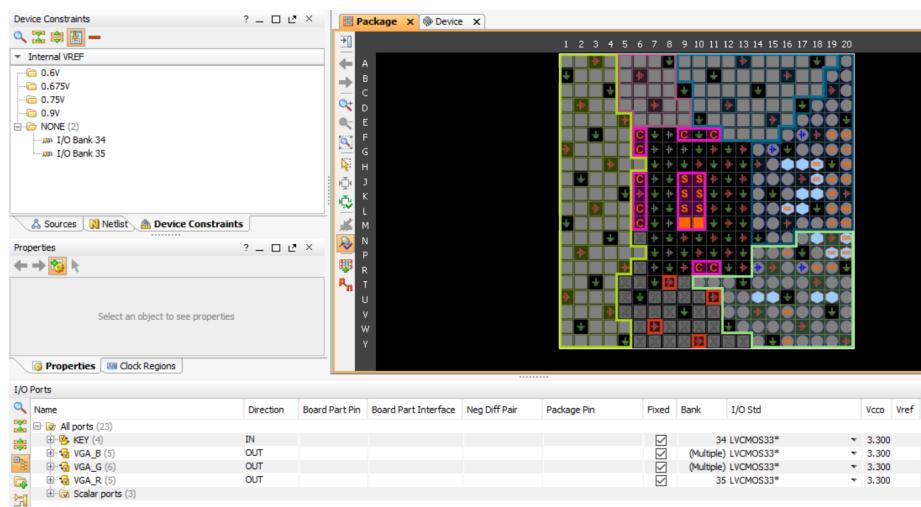
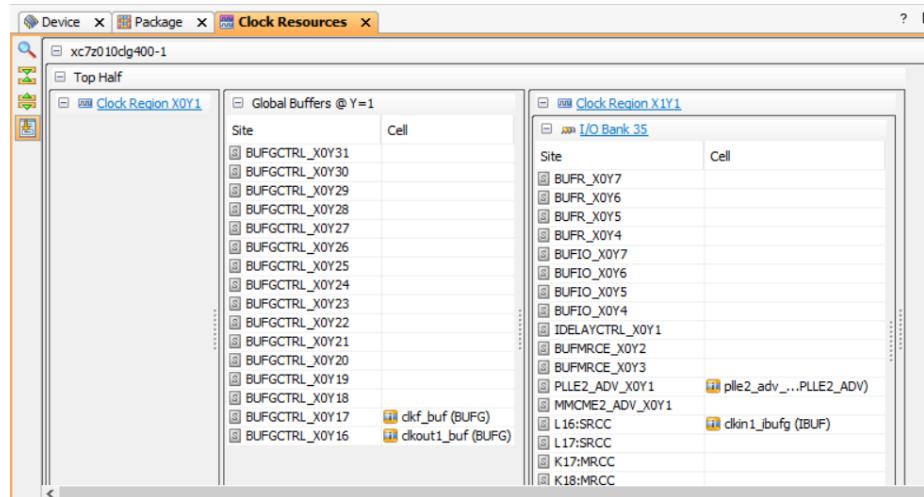
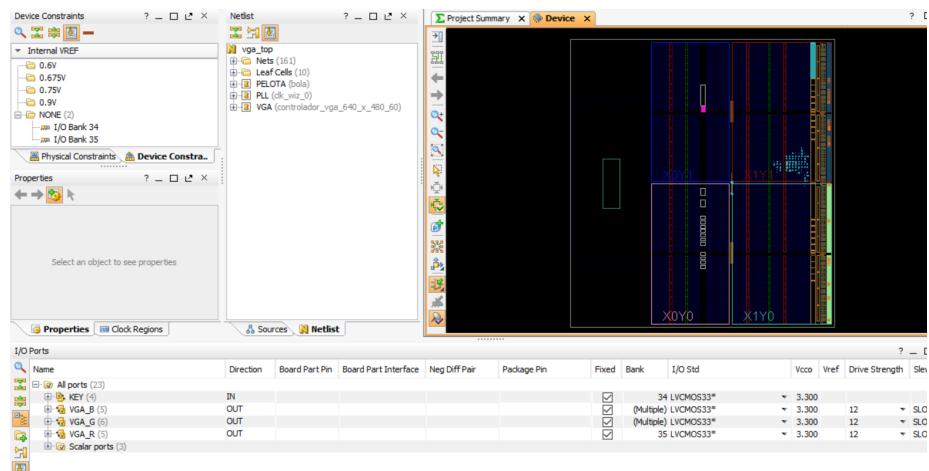
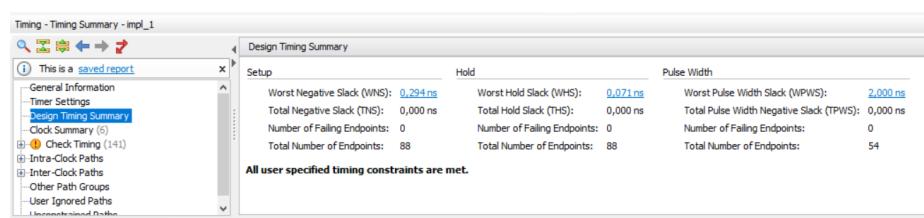


Figura 5.7: *I/O Planning*

Figura 5.8: *Clock Planning*Figura 5.9: *Floorplanning*Figura 5.10: *Timing Analysis*

Project Status Bar da información sobre el estado actual del diseño activo. *Data Windows Area* muestra información sobre los archivos que forman el

diseño. *Workspace* muestra ventanas como el editor de textos o el diseño del diagrama de circuitos, entre otros. Y *Results Windows Area* presenta los resultados de los comandos ejecutados. Además se muestran distintas ventanas, como *Tcl Console*, *Messages*, *Log*, *Reports* y *Design Runs*.

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] Yang Sun, Kiarash Amiri, Guohui Wang, Bei Yin, Joseph R Cavallaro, and Tai Ly. High-level design tools for complex dsp applications. Elsevier, Waltham, MA, 2012.
- [6] Wikipedia. Xilinx. <https://en.wikipedia.org/wiki/Xilinx>.
- [7] Xilinx. Field programmable gate array (fpga). <https://www.xilinx.com/products/silicon-devices/fpga/what-is-an-fpga.html>.
- [8] Xilinx. Vivado design suite user guide : Using the vivado ide. https://www.xilinx.com/support/documentation/sw_manuals/xilinx2016_2/ug893-vivado-ide.pdf.
- [9] Xilinx. Xup students. <https://www.xilinx.com/support/university/students.html#overview>.