

Introducción a FPGAs

y repaso de Lógica Digital

David Alejandro González Márquez

Programación de softcores en FPGAs
Programa de Profesoras/es Visitantes
Departamento de computación
Universidad de Buenos Aires

Clase disponible en: <https://github.com/fokerman/fpgaSoftcoreProgrammingCourse>

La materia: Programación de Softcores en FPGAs

Objetivo de este curso:

El aprendizaje de la programación de FPGAs guiada por el desarrollo de *softcores processors*.

Temas y calendario:

- T0 Repaso de **lógica digital** e Introducción a las FPGAs
- T1 Lenguajes de descripción de Hardware, **Verilog**.
- T2 Modelo de **tiempos** y delay. Definición de **Testbench**.
- T3 **Arquitectura** de procesadores, modelos de cómputo.
- T4 **Microarquitectura**, diseños de múltiples ciclos y un solo ciclo.
- T5 Técnicas de segmentación de instrucciones (**pipeline**). Diseño de etapas y tiempos.
- T6 Introducción a **softcores**: Picoblaze y Microblaze, características y casos de uso. Modelado e interacción entre múltiples cores.

Clases:

- Todas las clases van a estar disponibles en [github](#) con sus fuentes.
Son libres de enviar PRs para mejorar el material o incluso agregar comentarios.

La materia: Programación de Softcores en FPGAs

Requerimientos:

- Notebook con Linux + Software de diseño Vivado de Xilinx.

Material de estudio y práctica:

- Clases teóricas y prácticas.
- Guía práctica de ejercicios de Verilog.
- Bibliografía y Papers.

Forma de evaluación:

- Trabajo Práctico de Microarquitectura → [Nota numérica grupal](#).
- Presentación de temas → [Presentación en clase en grupo](#).
- Evaluación oral/escrita de final de curso → [Nota numérica](#).

Calificación final:

- Promedio entre evaluación final y trabajo práctico.

Bibliografía

- **“Digital Design and Computer Architecture”**, Second Edition
David Money Harris, Sarah L. Harris - Morgan Kaufmann - 2013
- **“Diseño Digital”**, Tercera Edición
M. Morris Mano - Pearson - 2003
- **“Essentials of Computer Organization and Architecture”**, 5th Edition
Linda Null, Julia Lobur - Jones and Bartlett Publishers - 2018.
- **“Introduction to Computing Systems”**, Third Edition
Yale N. Patt, Sanjay J. Patel - McGraw-Hill - 2019
- **“Computer Organization and Design: The Hardware/Software Interface”**, Fifth Edition
David A. Patterson, John L. Hennessy - Morgan Kaufmann - 2014
- **“Synthesis of Arithmetic Circuits FPGA, ASIC, and Embedded Systems”**
Jean-Pierre Deschamps, Gery Jean Antoine Biol, Gustavo D. Sutter - John Wiley & Sons - 2006
- **“CMOS VLSI Design: A Circuits and Systems Perspective”**, Fourth Edition.
Neil H. E. Weste, David Money Harris - Pearson - 2011
- **“Computer Architecture: A Quantitative Approach”**, Sixth Edition.
John L. Hennessy, David A. Patterson - Morgan Kaufmann - 2019
- **“Digital Design and Verilog HDL Fundamentals”**
Joseph Cavanagh - CRC Press, Taylor & Francis Group - 2008

Agenda

1 - Repaso de Lógica Digital

2 - Introducción a los FPGAs

Componentes Básicos: Compuertas

Las compuertas nos permiten hacer operaciones con las señales eléctricas (0 y 1).

Vamos a utilizar compuertas para construir circuitos que respeten el comportamiento de una determinada función. Estos circuitos se denominan **combinatorios**.



\bar{A}	
A	NOT
0	1
1	0

$A \cdot B$		
A	B	AND
0	0	0
0	1	0
1	0	0
1	1	1

$A + B$		
A	B	OR
0	0	0
0	1	1
1	0	1
1	1	1

$\overline{A \cdot B}$		
A	B	NAND
0	0	1
0	1	1
1	0	1
1	1	0

$\overline{A + B}$		
A	B	NOR
0	0	1
0	1	0
1	0	0
1	1	0

$A \oplus B$		
A	B	XOR
0	0	0
0	1	1
1	0	1
1	1	0

Construir funciones booleanas con su tabla de verdad

Dos formas canónicas de expresiones booleanas:

- **Suma de Productos**

Expresión que hace la suma de todas las combinaciones que resulten en 1.

- **Producto de Sumas**

Expresión que hace el producto de todas las combinaciones que resulten en 0.

Ejemplo:

A	B	$F(A, B)$
0	0	1
0	1	0
1	0	1
1	1	0

Suma de Productos

$$F(A, B) = (\bar{A} \cdot \bar{B}) + (A \cdot \bar{B})$$

Producto de Sumas

$$F(A, B) = (A + \bar{B}) \cdot (\bar{A} + \bar{B})$$

Construir circuitos a partir de una función booleana

Ejemplo:

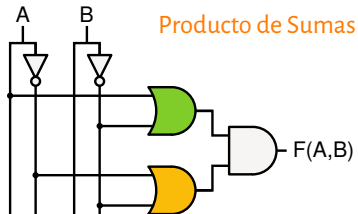
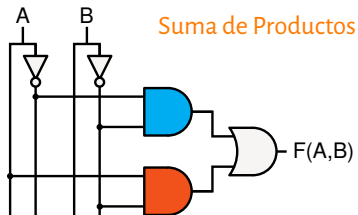
A	B	$F(A, B)$
0	0	1
0	1	0
1	0	1
1	1	0

Suma de Productos

$$F(A, B) = (\bar{A} \cdot \bar{B}) + (A \cdot \bar{B})$$

Producto de Sumas

$$F(A, B) = (A + \bar{B}) \cdot (\bar{A} + B)$$



Construir circuitos a partir de una función booleana

Ejemplo:

A	B	$F(A, B)$
0	0	1
0	1	0
1	0	1
1	1	0

Suma de Productos

$$F(A, B) = (\bar{A} \cdot \bar{B}) + (A \cdot \bar{B})$$

$$F(A, B) = (\bar{A} + A) \cdot \bar{B}$$

$$F(A, B) = 1 \cdot \bar{B}$$

$$F(A, B) = \bar{B}$$

Producto de Sumas

$$F(A, B) = (A + \bar{B}) \cdot (\bar{A} + \bar{B})$$

$$F(A, B) = (A \cdot \bar{A}) + \bar{B}$$

$$F(A, B) = 0 + \bar{B}$$

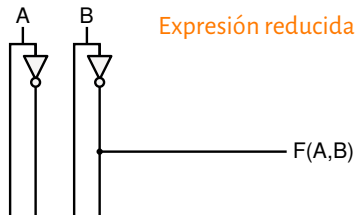
$$F(A, B) = \bar{B}$$

Suma de Productos

$$F(A, B) = (\bar{A} \cdot \bar{B}) + (A \cdot \bar{B})$$

Producto de Sumas

$$F(A, B) = (A + \bar{B}) \cdot (\bar{A} + \bar{B})$$



Uso de compuertas para seleccionar y habilitar

Habilitar



AND de X y 1 deja pasar X



AND de X y 0 no deja pasar X



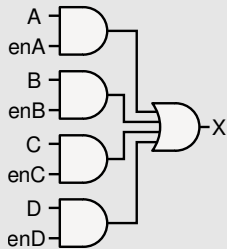
OR de X y 0 deja pasar X

Seleccionar



OR de Y y 0 deja pasar Y

Ejemplo



Las señales enA, enB, enC y enD son de habilitación, para los datos de las señales A, B, C y D.

Circuitos Combinatorios: Decodificadores y Codificadores

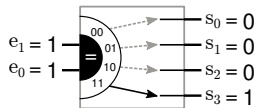
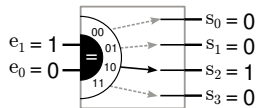
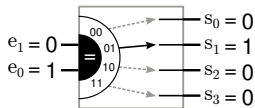
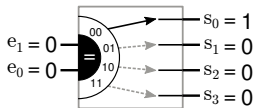
Decodificador



Circuito que toma n entradas (e_0 a e_{n-1}) y genera 2^n salidas (s_0 a s_{2^n-1}).

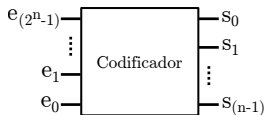
Coloca un 1 en la salida codificada por la entrada.
El resto de las salidas quedan en 0.

Ejemplo - Decodificador de 2 entradas



Circuitos Combinatorios: Decodificadores y Codificadores

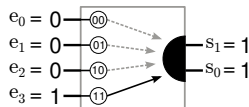
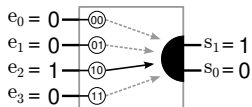
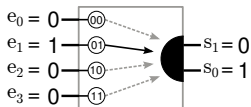
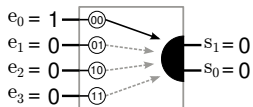
Codificador



Circuito que toma 2^n entradas (e_0 a e_{2^n-1}) y genera n salidas (s_0 a s_{n-1}).

Expone en las salidas la codificación de la única entrada en 1.
Este circuito no permite que más de una entrada esté en 1.

Ejemplo - Codificador de 4 entradas

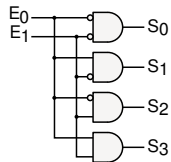


Circuitos Combinatorios: Decodificadores y Codificadores

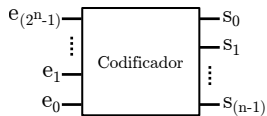
Decodificador



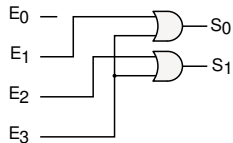
entradas		salida			
E_1	E_0	S_0	S_1	S_2	S_3
0	0	1	0	0	0
0	1	0	1	0	0
1	0	0	0	1	0
1	1	0	0	0	1



Codificador

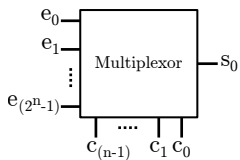


entradas				salidas	
E_0	E_1	E_2	E_3	S_1	S_0
1	0	0	0	0	0
0	1	0	0	0	1
0	0	1	0	1	0
0	0	0	1	1	1



Circuitos Combinatorios: Multiplexores y Demultiplexores

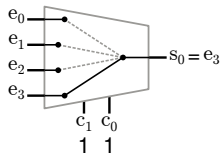
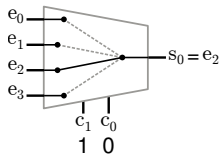
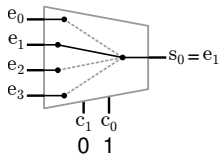
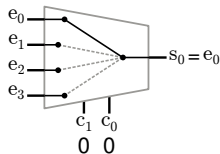
Multiplexor



Circuito que toma n entradas de control (c_0 a c_{n-1}), 2^n entradas de datos (e_0 a e_{2^n-1}) y genera una salida s_0 .

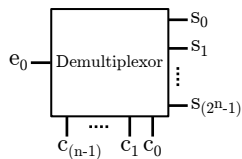
Dependiendo del valor de las entradas de control, selecciona una de las entradas de datos y la expone en la salida.

Ejemplo - Multiplexor de 4 entradas



Circuitos Combinatorios: Multiplexores y Demultiplexores

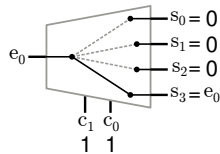
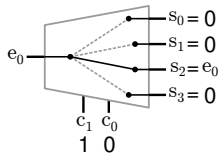
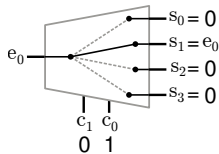
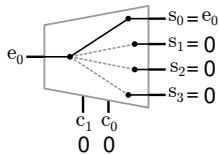
Demultiplexor



Circuito que toma n entradas de control (c_0 a c_{n-1}), una entrada de datos e_0 y genera 2^n salidas (s_0 a s_{2^n-1}).

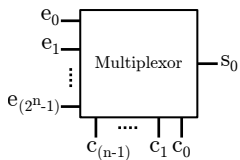
Expone la entrada en una de las salidas, dependiendo del valor de las entradas de control.

Ejemplo - Demultiplexor de 4 entradas

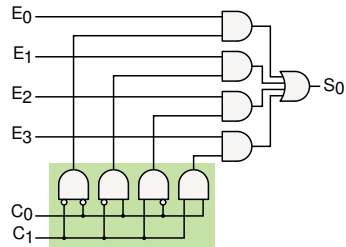


Circuitos Combinatorios: Multiplexores y Demultiplexores

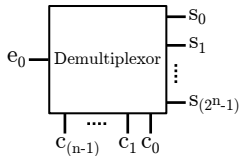
Multiplexor



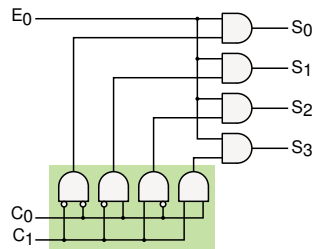
entradas						salida
E ₀	E ₁	E ₂	E ₃	C ₁	C ₀	S ₀
e ₀	e ₁	e ₂	e ₃	0	0	e ₀
e ₀	e ₁	e ₂	e ₃	0	1	e ₁
e ₀	e ₁	e ₂	e ₃	1	0	e ₂
e ₀	e ₁	e ₂	e ₃	1	1	e ₃



Demultiplexor



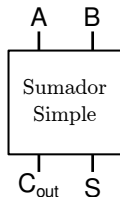
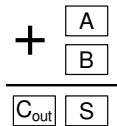
entradas			salidas			
E ₀	C ₁	C ₀	S ₀	S ₁	S ₂	S ₃
e ₀	0	0	e ₀	0	0	0
e ₀	0	1	0	e ₀	0	0
e ₀	1	0	0	0	e ₀	0
e ₀	1	1	0	0	0	e ₀



Circuitos Combinatorios: Circuitos Aritméticos

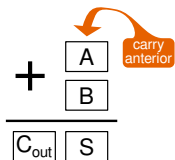
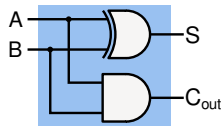
Conjunto de circuitos que nos permiten realizar **operaciones matemáticas**.

Sumador Simple



A	B	C _{out}	S
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

$$C_{out} = A \cdot B$$
$$S = (\bar{A} \cdot B) + (A \cdot \bar{B}) = A \oplus B$$

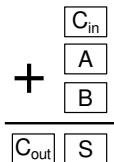


Sumador simple de 1 bit, nos permite sumar dos bits A y B.
El resultado se expresa como S y el carry como C_{out}.

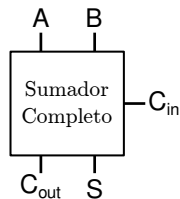
Si queremos sumar más de un bit, necesitamos también sumar el carry de la operación anterior.

Circuitos Aritméticos

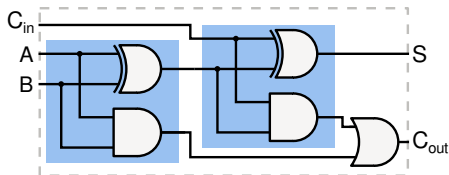
Sumador Completo



Un sumador completo de 1 bit, nos permite sumar dos bits A y B, y además el carry de la operación anterior C_{in} . El resultado se expresa como S y un carry C_{out} .



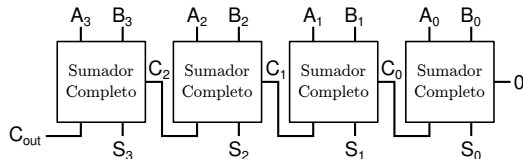
C_{in}	A	B	C_{out}	S
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1



Circuitos Aritméticos

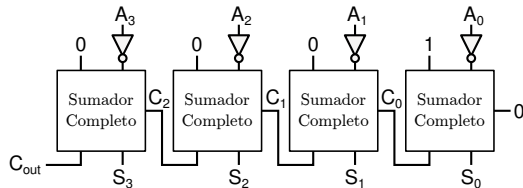
Sumador de 4 bits

Utilizando 4 sumadores completos, podemos construir un sumador de 4 bits.



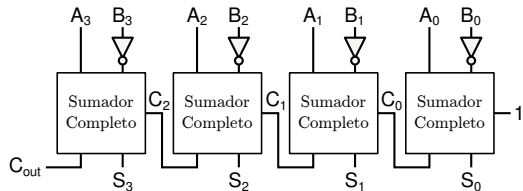
Inversor aditivo de 4 bits

Si negamos la entrada y le sumamos uno. Obtenemos el inverso aditivo de un número en complemento a 2.



Restador de 4 bits

Si sumamos un número con el inverso aditivo de otro obtenemos un circuito restador.

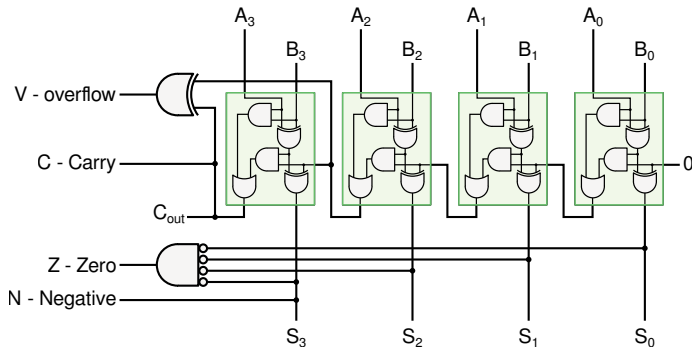


Circuitos Aritméticos - Flags

Los circuitos aritméticos además de generar resultados de sus operaciones, también generan lo que se conoce como **palabra de estado**.

La palabra de estado contiene una serie de **Flags**.

Vamos a ver algunos de ellos y cómo se construyen.



N	Negative	Indica si el número es negativo en complemento a 2.
Z	Zero	Indica si el número es cero en complemento a 2.
C	Carry	Indica si la operación en complemento a 2 genera acarreo.
V	Overflow	Indica si el resultado no es representable en complemento a 2.

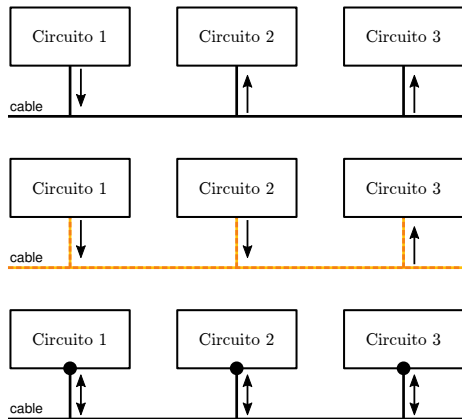
Lógica de tres estados

Supongamos tener un **cable** que conecta tres circuitos diferentes. Si un circuito escribe una señal, esta es leída por los otros.

Ahora, no es posible que dos circuitos escriban **simultáneamente** una señal.

A pesar de que los circuitos puedan acordar no escribir simultáneamente.

Necesitaríamos algún dispositivo que **permita decidir** si leemos o escribimos un cable.



No podemos cambiar el cable para leer o escribir, pero sí podemos **desconectarlo**.

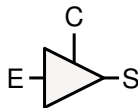
Lógica de tres estados

Un componente de 3 estados es un circuito electrónico que presenta a su salida tres estados posibles:

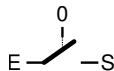
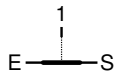
0 Estado lógico Cero.

1 Estado lógico Uno.

hi-Z No estado. Desconectado (*Alta impedancia*).



C	E	S
0	0	hi-Z
0	1	hi-Z
1	0	0
1	1	1



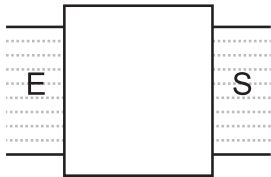
Este componente nos permite **desconectar** circuitos, para así decidir cuándo escribir o leer de un cable.

Más adelante:

Vamos a construir registros bidireccionales y utilizar buses, donde usaremos este componente.

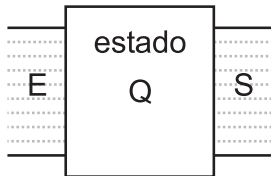
Circuitos Secuenciales

Circuitos Combinacionales



La salida está determinada únicamente por la entrada del circuito

Circuitos Secuenciales



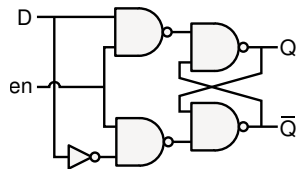
La salida está determinada por la entrada y el **estado interno** del circuito

El estado estará determinado por una memoria, pero **¿cómo almacenamos bits?**

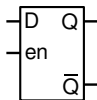
Flip-Flops (*biestable*)

Los flip-flop son circuitos *biestables* que permiten **almacenar 1 bit** de memoria.

Tipo D: Activado por nivel.

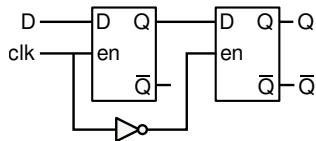


en	D	Q_n	Q_{n+1}
1	0	Q_n	0
1	1	Q_n	1
0	x	Q_n	Q_n

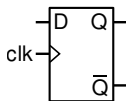


Si la señal **en** está en 1, el estado Q toma el valor de la señal **D**.

Tipo D Master-Slave: Activado por flanco.

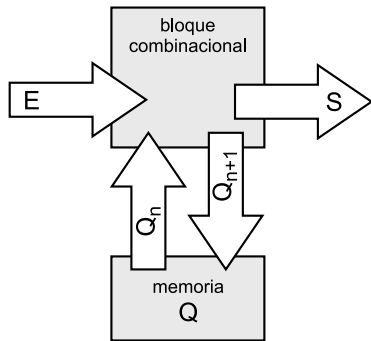


en	D	Q_n	Q_{n+1}
1	0	Q_n	0
1	1	Q_n	1
0	x	Q_n	Q_n



Si la señal **clk** cambia de 1 a 0 (flanco descendente), Q toma el valor del último valor guardado durante el puso 1 de la señal **clk**.

Circuitos Secuenciales



Un circuito secuencial, se puede separar en dos partes:

- 1 un *bloque combinacional*
- 2 un *bloque con memoria*

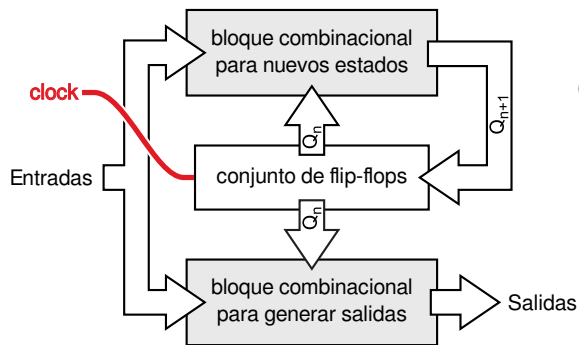
La memoria almacena bits que determinan el estado del circuito.

Las entradas del bloque combinacional son las entradas (E) y las salidas de la memoria (Q_n).

El bloque combinacional genera la salida del circuito (S) y el nuevo estado de la memoria (Q_{n+1}).

Circuitos Secuenciales

El esquema anterior lo podemos implementar usando flip-flops de la siguiente forma:



Contamos con dos bloques combinacionales para generar:

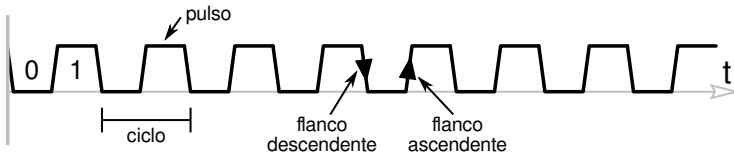
- 1 los nuevos estados de los flip-flops
- 2 y otro para generar las salidas del circuito

Q_n representa el estado actual de los flip-flops, mientras que Q_{n+1} representa el estado siguiente.

Pero, **¿cuándo se modifica el estado interno de un circuito?**

Reloj (Clock)

El reloj o señal de reloj, se utiliza para **temporizar** los cambios que se suceden en un circuito. Permite coordinar el momento exacto en que se **modifica** el estado de los biestables.



La señal varía entre 0 y 1 en intervalos regulares de tiempo. Su ciclo es simétrico, ya sea comenzando desde 0 o desde 1.

Se denomina **flanco**, al momento en que la señal cambia entre estados.

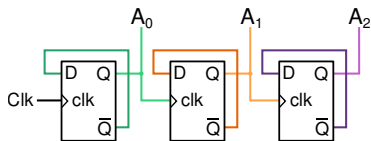
Flanco descendente: Cambia entre 1 a 0.

Flanco ascendente: Cambia entre 0 a 1.

Contadores

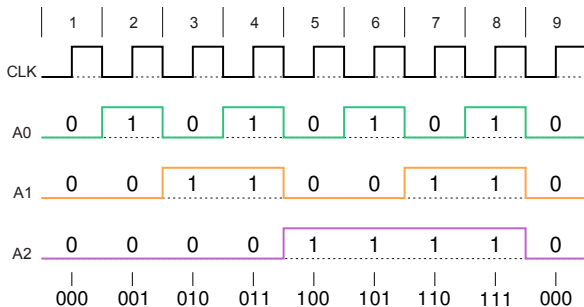
Los circuitos contadores generan una **secuencia** de valores en su salida por cada ciclo de clock.

Ejemplo: Contador módulo potencia de 2 de 3 bits. Cuenta de 000b a 111b.



Cada uno de los flip-flops cambia de estado por el contrario al que tiene almacenado.

El cambio se produce cuando el flip-flop anterior, que alimenta su clock, cambia de estado.



Contadores

Contador módulo potencia de 2

Secuencia ascendente tal que su módulo siempre es una potencia de 2 por cada bit que se agrega al contador.

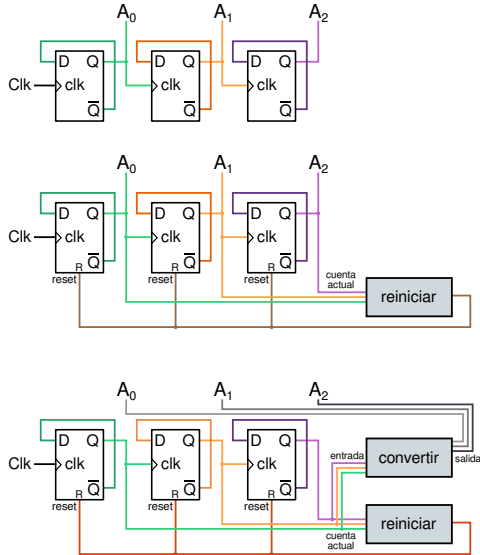
Contador módulo arbitrario

Cuando la secuencia supera valor máximo es reiniciada a cero gracias a un circuito combinatorio provisto para tal fin.

Contador módulo y cuenta arbitraria

Se provee un circuito combinatorio que transforma cada valor de la secuencia en uno arbitrario.

Ejemplos:



Registros

Un registro es un circuito secuencial que contiene un conjunto de n flip-flops asociados, que permiten **almacenar temporalmente** una palabra o grupo de n bits.

Los tipos de registro dependen de la forma en que los datos son **leídos** o **almacenados**:

- 1 Registro **paralelo-paralelo** (entrada paralelo, salida paralelo)
- 2 Registro **serie-paralelo** (entrada serie, salida paralelo)
- 3 Registro **paralelo-serie** (entrada paralelo, salida serie)
- 4 Registro **serie-serie** (entrada serie, salida serie)

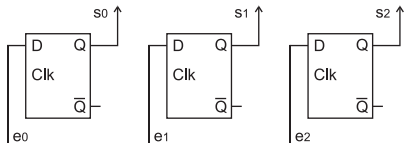
Serie: Los bits entran o salen del registro secuencialmente, uno a continuación del otro.

Paralelo: Los bits entran o salen del registro simultáneamente, todos al mismo tiempo.

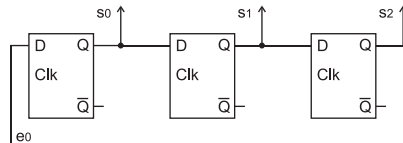
Registros

Ejemplos para registros de 3 bits.

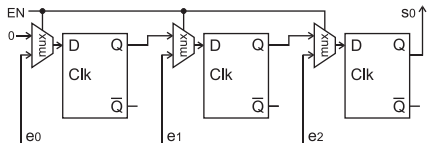
Registro paralelo-paralelo



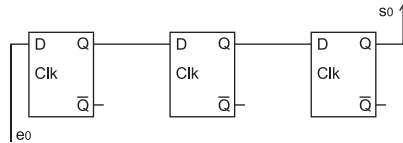
Registro serie-paralelo



Registro paralelo-serie



Registro serie-serie



Registros bidireccionales

En términos prácticos, vamos a querer **conectar registros entre sí**, tanto para leer como para escribir.

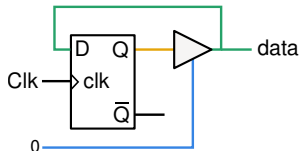
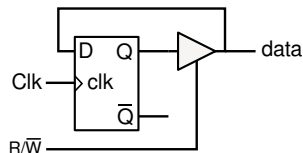
Para esto vamos a necesitar registros que operen **bidireccionalmente**.

Es decir que, utilizando las mismas entradas, podamos escribir y leer el registro.

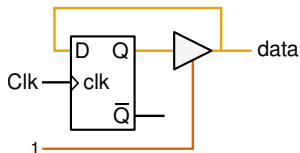
Usando **componentes de 3 estados**, vamos a construir registros bidireccionales.

Esta idea se puede extender a cualquier circuito, no necesariamente registros.

Ejemplo: registro de 1 bit.



Si la señal R/\overline{W} es 0, el valor en la **entrada** data puede llegar a la entrada D del biestable.



Si la señal R/\overline{W} es 1, el estado Q del biestable se expone en la **salida** data.

Agenda

1 - ~~Repaso de Lógica Digital~~

2 - Introducción a los FPGAs

Microprocesador, FPGAs, ASICs

Microprocesador

Unidad de procesamiento principal de una computadora. Ejecuta todo tipo de programas de propósito general. Su arquitectura no puede ser modificada.

FPGAs

Arreglo programable de celdas lógicas. Puede ser configurado para comportarse como cualquier circuito digital. Incluso como un procesador.

ASICs

Es un circuito de aplicación específica (*Application-Specific Integrated Circuit*). Integrado para realizar una sola tarea. Los procesadores son casos particulares de ASICs.

	Microprocesador	FPGAs	ASICs
Utilización	Propósito general	Prototipado/pequeño volumen	Producción en masa
Flexibilidad	Hardware no modificable	Reconfigurable	Completa flexibilidad
Lenguajes de programación	Múltiples	VHDL/Verilog/otros	VHDL/Verilog/otros
Programación	Archivo Ejecutable	Bitstream	Circuito diseñado

Field programmable gate arrays (FPGAs)

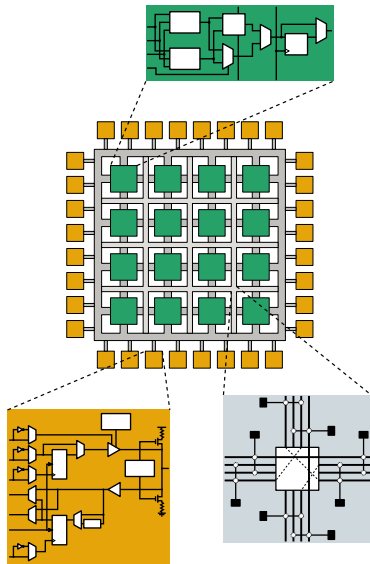
Pueden ser usados para implementar cualquier diseño de Hardware. Usualmente utilizados para prototipar la implementación productiva de un ASIC.

Pueden ser parte de un producto final de alto grado de especialización, dependiendo del volumen y costos.

Desarrollados a principios de 1980 como solución a la “glue logic” necesaria para conectar circuitos integrados.

En 1984 Xilinx™ introduce la tecnología de los FPGA como alternativa para remplazar la “glue logic”.

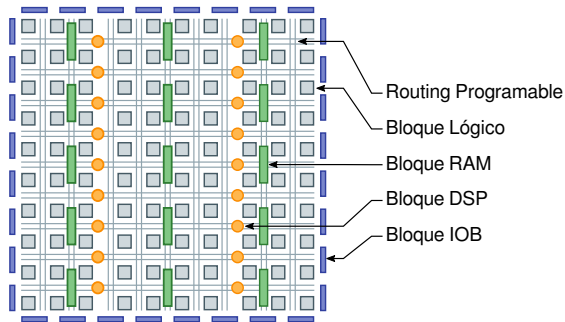
El diseño de circuitos en FPGA reduce el *time to market*, sin requerir un diseño físico, ni de manufactura como un IC.



FPGAs: Conceptos Básicos

La arquitectura de un FPGA consiste en un **arreglo de bloques lógicos y elementos de memoria**. Donde es posible configurar:

- 1 El **comportamiento** de cada bloque como una función lógica particular.
- 2 Las **entradas y salidas** que van a llegar a cada bloque.
- 3 La **interconexión** entre cualquier par de bloques.



Los diferentes FPGA del mercado difieren entre sí por:

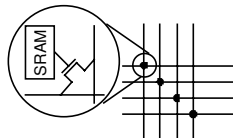
- Cantidad de bloques, funcionalidad y especialización de cada uno.
- Distribución de los bloques.
- Mecanismos de interconexión y ruteo entre bloques.
- Soporte integrado adicional, como procesadores.

FPGAs: Tecnologías de programación

Tres tecnologías básicas para programar el comportamiento de circuitos

- **Basado en SRAM**

Se utilizan *pass-transistors*, *transmission gates*, o *multiplexers* controlados por celdas de SRAM. Rápidamente reconfigurables de forma ilimitada. Requieren un integrado de mayor tamaño. Programación volátil.

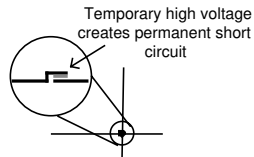


- **Antifuse**

La conexión se funde una vez programada. Más económico que SRAM. Solo puede ser programado una vez. Programación no volátil.

- **EEPROM/Flash**

Similar a SRAM pero con memorias no volátiles. Demoran más tiempo en ser reprogramadas.



Si bien cada fabricante da un nombre particular a su tecnología, o difieren en su implementación, conceptualmente estos tres comportamientos son los más habituales.

FPGAs: LookUp Table (LUT)

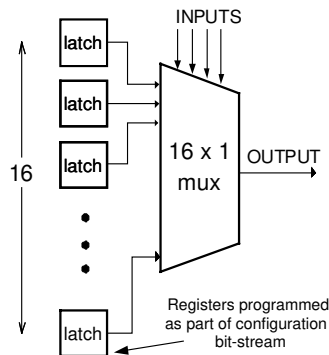
Las LUTs permiten implementar **cualquier función lógica**.

Una n -LUT consiste en una memoria de 2^n bits que por medio de un decodificador permite implementar una función lógica de n bits de entrada y un bit de salida.

Las LUT son configuradas por el bit-stream de programación y no pueden ser alteradas una vez programadas.

Dentro de una celda lógica, no solo tenemos una LUT, sino también multiplexores y Flip-Flops.

Las **celdas lógicas** son las más comunes dentro de una FPGA.



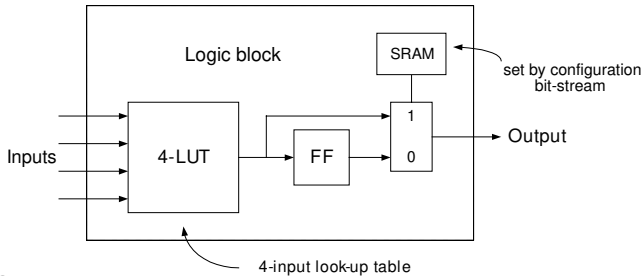
INPUTS		
0000	$F(0,0,0,0)$	← stored in 1 st latch
0001	$F(0,0,0,1)$	← stored in 2 nd latch
0010	$F(0,0,1,0)$	← stored in 3 rd latch
0011	$F(0,0,1,1)$	
0100		
...	...	
1101		
1110	$F(1,1,1,0)$	← stored in 15 th latch
1111	$F(1,1,1,1)$	← stored in 16 th latch

FPGAs: Celda lógica

Una celda lógica básica, contiene una LUT, un Flip-Flop y un multiplexor.

La LUT se configura por medio de celdas de SRAM que indican las respuestas de la función lógica a implementar.

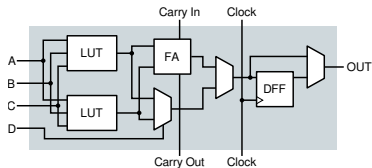
Mientras que el multiplexor de salida se configura para que la respuesta de la celda sea generada por el Flip-Flop o directamente de la función lógica.



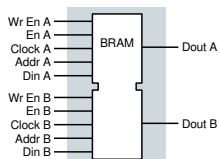
La interconexión y configuración de estas celdas permite implementar cualquier circuito, sin embargo estas pueden ser ineficientes.

Por lo tanto se cuenta con celdas especializadas para distintas **funciones específicas**.

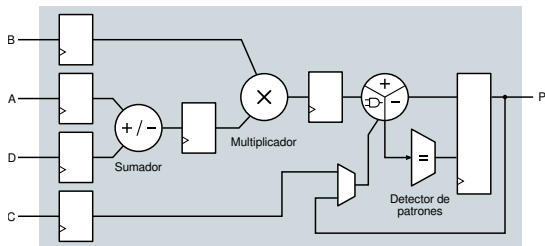
FPGAs: Tipos de celdas o bloques



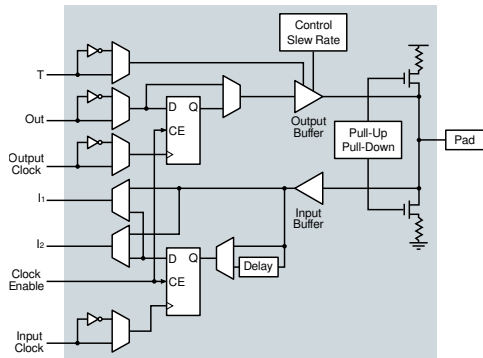
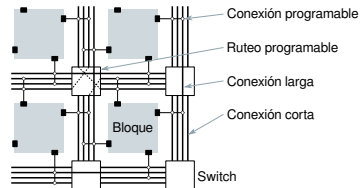
Bloque Lógico



Bloque de Memoria



Bloque DSP



Bloque de Entrada/Salida

FPGAs: Fabricantes

Existen diferentes fabricantes de **chips FPGAs**. Estos diseñan, construyen y venden los integrados que contienen las FPGAs.

Cada fabricante tiene su propio *stack* de desarrollo, con su propio *software* y bibliotecas de primitivas.

Además existen otros fabricantes que solo ensamblan **placas de desarrollo** o **placas específicas con FPGAs**

Para tareas de prototipado o en investigación se utilizan las **placas de desarrollo**, ya que cuentan con múltiples interfaces de entrada/salida para experimentos.

Fabricantes de FPGAs

SRAM-based

- Xilinx
- Altera
- Atmel
- Lattice

Antifuse o Flash

- Actel
- Quick logic

Fabricantes de placas

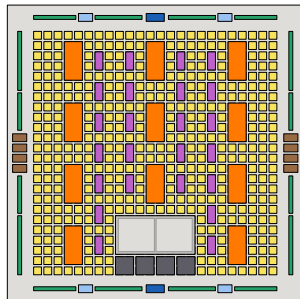
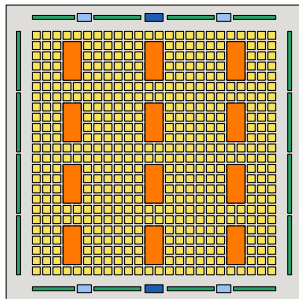
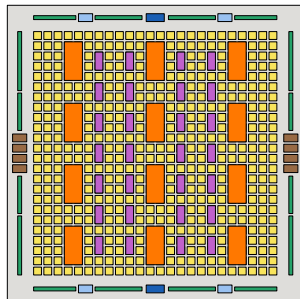
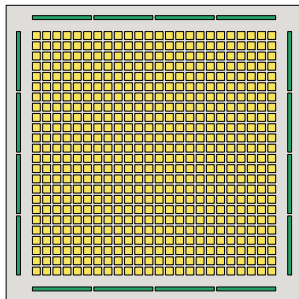
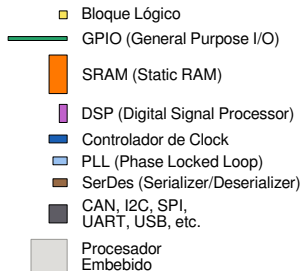
- | | |
|-----------------|--------------------|
| • Digilent | • MYIR Tech |
| • Alinx | • Krtrl |
| • Avnet | • Terasic |
| • Invent Logics | • BittWare |
| • Numato Lab | • Hitech Global |
| • Opal Kelly | • Trenz Electronic |

FPGAs: Tipos de placas

Dependiendo de la cantidad de celdas y tipos de celdas podemos encontrar diferentes configuraciones de placas.

Principalmente tenemos tres tipos:

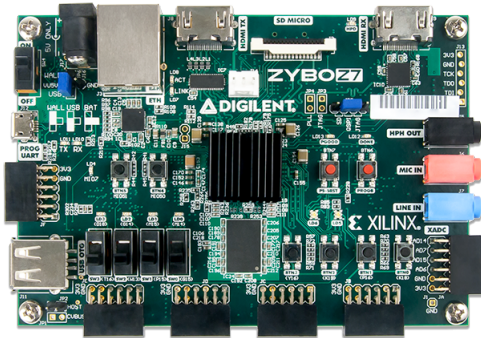
- FPGA pura.
- FPGA System-on-chip (Soc).
- FPGA específico (video, señales, i/o).



FPGAs: Zybo Z7

Features

- 667MHz dual-core Cortex-A9 processor with integrated FPGA.
- 1 GB DDR3L with 32-bit bus @ 1066 MHz.
- Wide range of USB, Ethernet, Video, and Audio connectivity.
- Pmod connectors for adding-on hardware devices.
- Pcam connector for camera sensors with MIPI CSI-2 interface.
- Programmable from JTAG, Quad-SPI flash, and microSD card.



Connectivity and On-board I/O

Video	HDMI Output
	HDMI Input
	Pcam connector
Networking	Gigabit Ethernet
USB	USB-UART
	USB-JTAG Programmer
	USB Host\OTG
Pmod Connectors	6
Switches	4 Slide switches
Buttons	4 Push buttons
	2 MIO Push buttons
LEDs	4 LEDs
	1 MIO LED
	2 RGB LEDs

Electrical

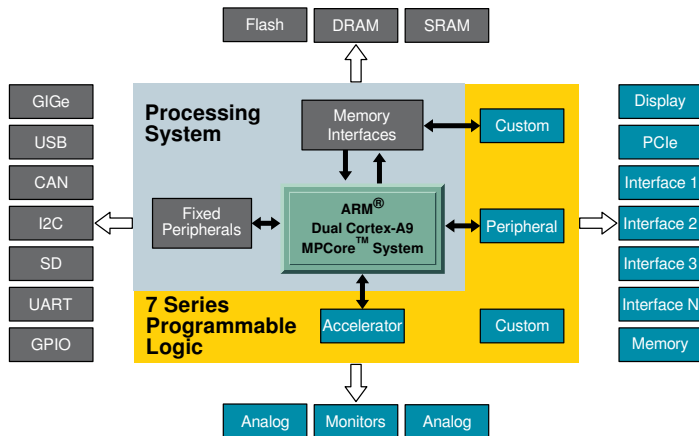
Power USB
5V (2.5mm coaxial) supply
Logic Level 3.3V

Physical

Width 3.3 in (88 mm)
Length 4.8 in (122 mm)

FPGAs: Zybo Z7

Zynq-7000 ARM/FPGA SoC Development Board



Key FPGA Specifications

Part Number	XC7Z020-1CLG400C
Logic slices	13,300
6-input LUTs	53,200
Flip-Flops	106,400
Block RAM	630 KB
DSP Slices	220
Clock Resources	Zynq PLL with 4 outputs 4 PLLs 4 MMCMs
Internal	125 MHz external clock ADC Dual-channel, 1 MSPS

FPGAs: Diseño, Síntesis e Implementación

La construcción de soluciones con FPGAs consta de etapas de diseño y etapas de verificación.

Design Entry: Archivos de diseño en HDL.

Design Synthesis: Convierte el HDL en una representación abstracta usando una biblioteca de primitivas.

Partition (or Mapping): Asigna a cada elemento lógico componentes físicos del dispositivo.

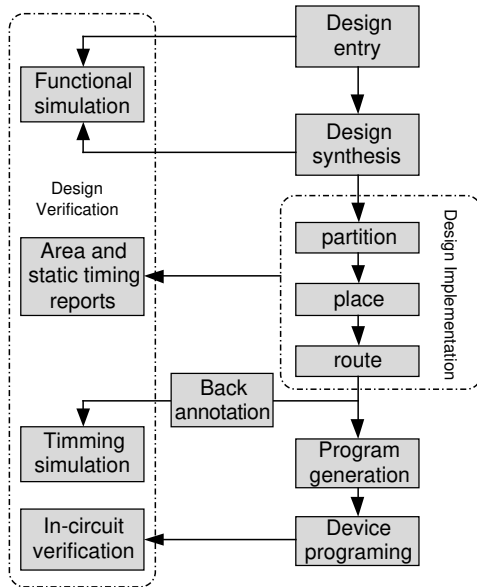
Place: Mapea los componentes físicos a componentes específicos del dispositivo.

Route: Construye las conexiones entre los componentes.

Program Generation: Genera un *bit-stream* para programar el dispositivo.

Device Programming: Carga el *bit-stream* en el dispositivo.

Design Verification: La simulación es utilizada para detectar errores y comprobar su funcionamiento.



Pantalla de inicio.
Desde aquí podemos crear
un **proyecto nuevo** o iniciar
el **administrador de IPs**.



Quick Start

- [Create Project >](#)
- [Open Project >](#)
- [Open Example Project >](#)

Tasks

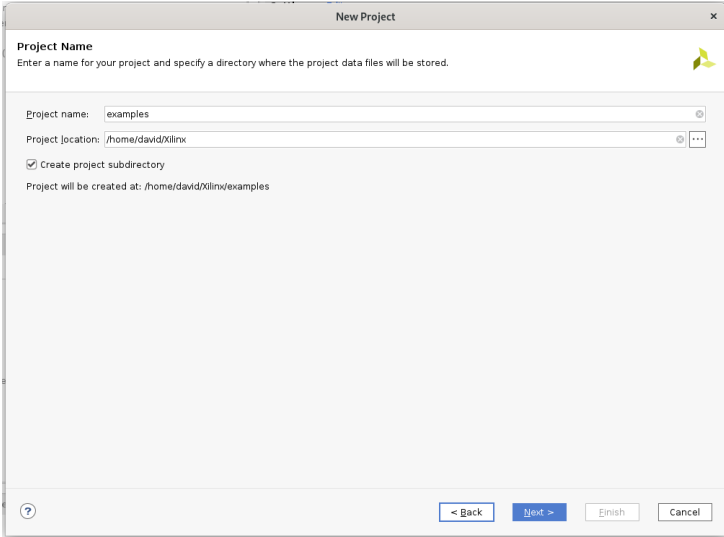
- [Manage IP >](#)
- [Open Hardware Manager >](#)
- [Vivado Store >](#)

Learning Center

- [Documentation and Tutorials >](#)
- [Quick Take Videos >](#)
- [What's New in 2022.2 >](#)

Nombre del proyecto y ubicación.

Consejo:
Crear un proyecto de experimentos, que puedan editar todo el tiempo.
Por otro lado, tengan proyectos independientes para cada tarea.



The screenshot shows the 'New Project' dialog box in Vivado Xilinx. The title bar is 'New Project' with a close button. The main section is titled 'Project Name' and contains the instruction: 'Enter a name for your project and specify a directory where the project data files will be stored.' There is a text input field for 'Project name:' with the value 'examples'. Below it is a text input field for 'Project location:' with the value '/home/david/Xilinx'. To the right of the location field is a button with a gear icon and three dots. A checkbox labeled 'Create project subdirectory' is checked. Below the checkbox, it says 'Project will be created at: /home/david/Xilinx/examples'. At the bottom left is a help button with a question mark icon. At the bottom right are four buttons: '< Back', 'Next >', 'Finish', and 'Cancel'.

Project Name
Enter a name for your project and specify a directory where the project data files will be stored.

Project name: examples

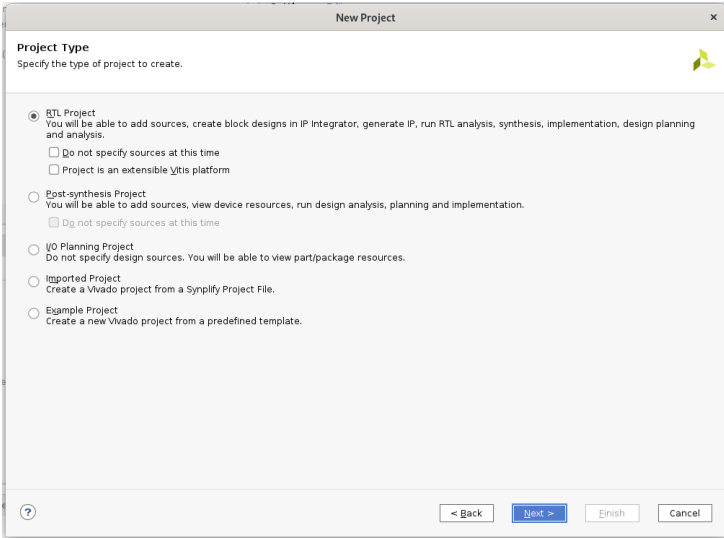
Project location: /home/david/Xilinx

☒ Create project subdirectory

Project will be created at: /home/david/Xilinx/examples

< Back Next > Finish Cancel

Seleccionar tipo de proyecto. Vamos a trabajar con un *RTL project*.



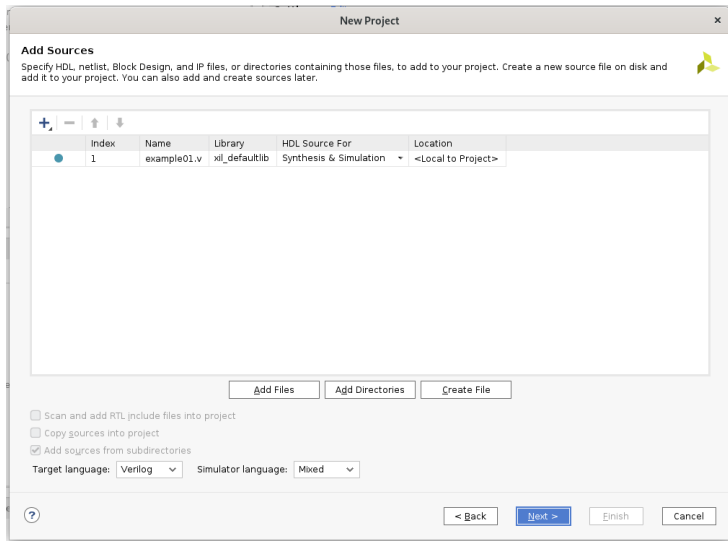
New Project

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
 - ☐ Project is an extensible Vitis platform
- ☐ **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 Project File.
- ☐ **Example Project**
Create a new Vivado project from a predefined template.

[?](#) [< Back](#) [Next >](#) [Finish](#) [Cancel](#)

Seleccionar los archivos *sources*. Podemos crear nuevos o importar ya creados.

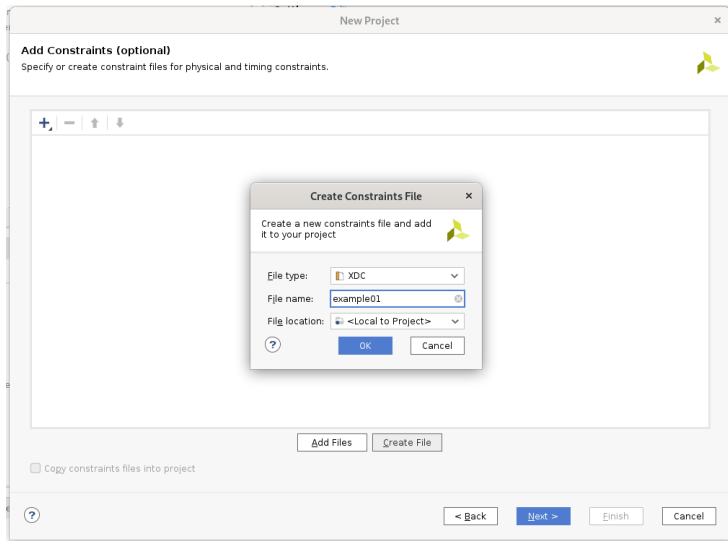


Seleccionar o crear el archivo de *constraints*.
Es el encargado de conectar los *pins* físicos con las entradas o salidas del circuito.

Cada placa tiene su propio archivo master con los nombres específicos de cada uno de los *pins*

Para Zybo Z7:

<https://github.com/Digilent/digilent-xdc/blob/master/Zybo-Z7-Master.xdc>

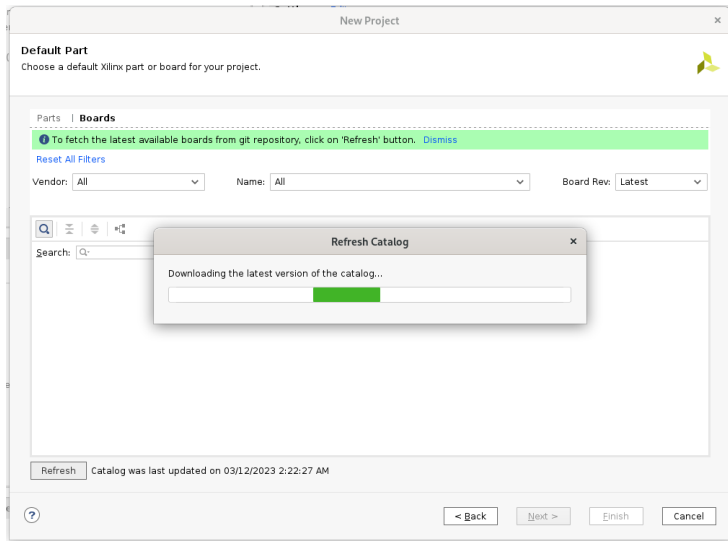


Vivado Xilinx

El IDE requiere tener la **biblioteca de componentes** específica de cada placa.

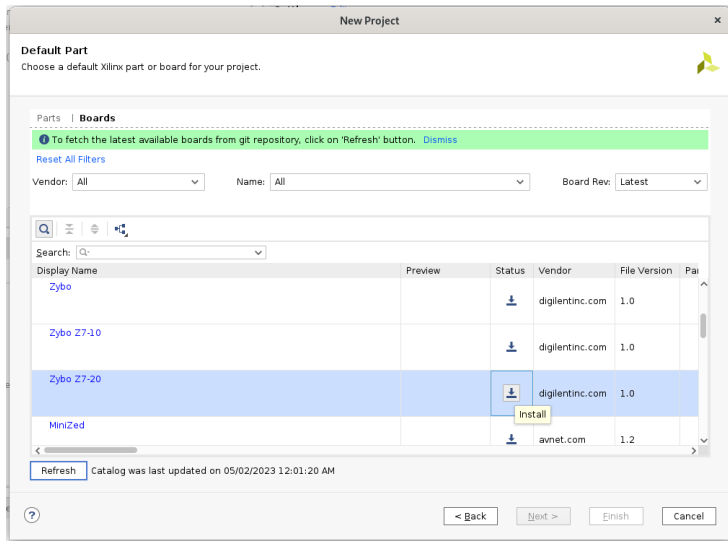
Por lo tanto es necesario tener el **catálogo de placas** completo.

Para esto Vivado provee una opción que permite descargar todas las placas soportadas.



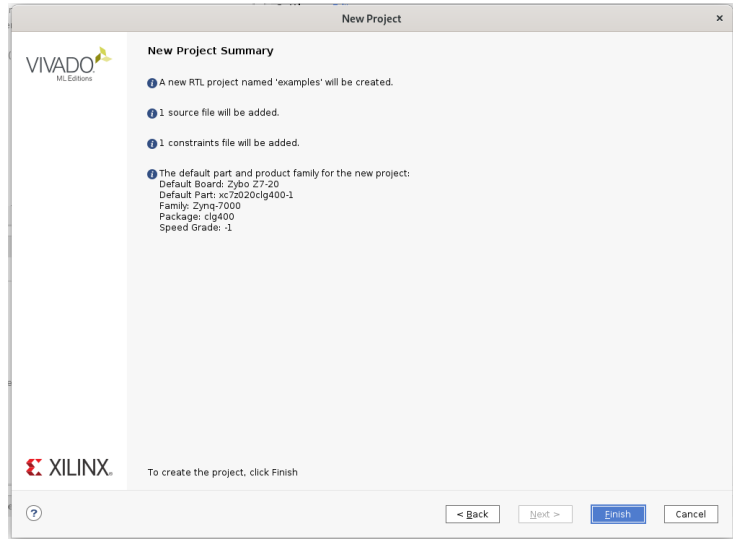
Una vez seleccionada la placa. Podemos descargar la **biblioteca de componentes** asociada a la misma.

En la instalación del software no se recomienda descargar el soporte para todas las placas durante la instalación. Ocupa mucho espacio y resulta más práctico descargarlo cuando se requiera.



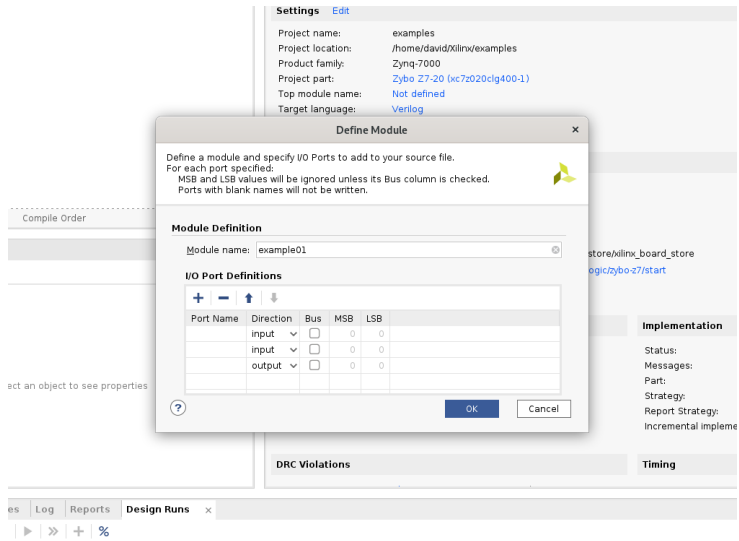
Vivado Xilinx

Una vez completada la creación del proyecto, se presenta un detalle de la configuración a generar.



La ventana de definición de módulos, nos permite crear un modulo nuevo seleccionando sus entradas y salidas.

Esta información será usada directamente para generar código HDL.



FileEditFlowToolsReportsWindowViewHelpQ: Quick Access

Default Layout

Ready

Flow Navigator

PROJECT MANAGER

SettingsAdd SourcesLanguage TemplatesIP Catalog

IP INTEGRATOR

Create Block DesignOpen Block DesignGenerate Block Design

SIMULATION

Run Simulation

RTL ANALYSIS

Open Elaborated Design

SYNTHESIS

Run SynthesisOpen Synthesized Design

IMPLEMENTATION

Run ImplementationOpen Implemented Design

PROGRAM AND DEBUG

Generate BitstreamOpen Hardware Manager

PROJECT MANAGER - examples

Sources

Design Sources (1)
example01 (example01.v)
Constraints (1)
constrs_1 (1)
example01.xdc
Simulation Sources (1)
Utility Sources

HierarchyLibrariesCompile Order

Properties

Select an object to see properties

Project Summary

OverviewDashboard

SettingsEdit

Project name: examples
Project location: /home/david/Xilinx/Vivado/2022.2/hub/board_store/xilinx_board_store
Product family: Zynq-7000
Project part: Zybo Z7-20 (xc7z020clg400-1)
Top module name: Not defined
Target language: Verilog
Simulator language: Mixed

Board Part

Display name: Zybo Z7-20
Board part name: digilentinc.com:zybo-z7-20:part0:1.1
Board revision: B.2
Connectors: No connections
Repository path: /home/david/Xilinx/Vivado/2022.2/hub/board_store/xilinx_board_store
URL: https://digilent.com/reference/programmable-logic/zybo-z7/start
Board overview: Zybo Z7-20

Synthesis

Status: Not started
Messages: No errors or warnings
Part: xc7z020clg400-1
Strategy: Vivado Synthesis Defaults
Report Strategy: Vivado Synthesis Default Reports
Incremental synthesis: Automatically selected checkpoint

Implementation

Status: Not started
Messages: No errors or warnings
Part: xc7z020clg400-1
Strategy: Vivado Implementation Defaults
Report Strategy: Vivado Implementation Default Reports
Incremental implementation: None

DRC Violations

Timing

Tcl ConsoleMessagesLogReportsDesign Runs

NameConstraintsStatusWNSTNSWHSTHSWBSSTPWSTotal PowerFailed RoutesMethodologyRQA ScoreQoR SuggestionsLUTFFBRAMURAMDSPStartElapsedRun Strategy

synth_1constrs_1Not started

impl_1constrs_1Not started

FileEditFlowToolsReportsWindowLayoutViewHelpQ: Quick Access

Default Layout

Flow Navigator

PROJECT MANAGER

SettingsAdd SourcesLanguage TemplatesIP Catalog

IP INTEGRATOR

Create Block DesignOpen Block DesignGenerate Block Design

SIMULATION

Run Simulation

RTL ANALYSIS

Open Elaborated Design

SYNTHESIS

Run SynthesisOpen Synthesized Design

IMPLEMENTATION

Run ImplementationOpen Implemented Design

PROGRAM AND DEBUG

Generate BitstreamOpen Hardware Manager

PROJECT MANAGER - examples

Sources

Design Sources (1)
example01 (example01.v)
Constraints (1)
constrs_1 (1)
example01.xdc
Simulation Sources (1)
Utility Sources

Archivos fuente

Source File Properties

example01.v

Enabled

Location: /home/david/Xilinx/examples/examples.srcs/sources_1/new

Type: Verilog

Library: xil_defaultlib

Size: 0.6 KB

Modified: Today at 00:07:23 AM

Copied to: <Project Directory>/examples.srcs/sources_1/new

Read-only: No

GeneralProperties

Project Summaryexample01.v

/home/david/Xilinx/examples/examples.srcs/sources_1/new/example01.v

timescale 1ns / 1ps
Company:
Engineer:
Create Date: 05/02/2023 12:04:40 AM
Design Name:
Module Name: example01
Project Name:
Target Devices:
Tool Versions:
Description:
Dependencies:
Revision:
Revision 0.01 - File Created
Additional Comments:

module example01(a, b, c);
input a;
input b;
output c;
assign c = a & b;
endmodule

Definición del módulo

Tcl ConsoleMessagesLogReportsDesign Runs

Name	Constraints	Status	WNS	TNS	WHS	THS	WBSS	TPWS	Total Power	Failed Routes	Methodology	RQA Score	QoR Suggestions	LUT	FF	BRAM	URAM	DSP	Start	Elapsed	Run Strategy
synth_1	constrs_1	Not started																			Vivado Synthesis Defaults (Vivado)
impl_1	constrs_1	Not started																			Vivado Implementation Defaults

Etapas del proyecto

FileEditFlowToolsReportsWindowLayoutViewHelpQ: Quick Access

Flow Navigator

PROJECT MANAGER

SettingsAdd SourcesLanguage TemplatesIP Catalog

IP INTEGRATOR

Create Block DesignOpen Block DesignGenerate Block Design

SIMULATION

Run Simulation

RTL ANALYSIS

Open Elaborated Design

SYNTHESIS

Run SynthesisOpen Synthesized Design

IMPLEMENTATION

Run ImplementationOpen Implemented Design

PROGRAM AND DEBUG

Generate BitstreamOpen Hardware Manager

PROJECT MANAGER - examples

Sources

Design Sources (1)
example01 (example01.v)
Constraints (1)
constrs_1 (1)
example01.xdc
Simulation Sources (1)
Utility Sources

Archivos fuente

HierarchyLibrariesCompile Order

Source File Properties

example01.xdc

Enabled

Location: /home/david/Xilinx/examples/examples.srcs/constrs_1/new

Type: XDC

Size: 17.3 KB

Modified: Today at 00:10:17 AM

Copied to: <Project Directory>/examples.srcs/constrs_1/new

Read-only: No

Encrypted: No

GeneralProperties

Project Summaryexample01.vexample01.xdc

/home/david/Xilinx/examples/examples.srcs/constrs_1/new/example01.xdc

```
1 ## This file is a general .xdc for the Zybo Z7 Rev. B
2 ## It is compatible with the Zybo Z7-20 and Zybo Z7-10
3 ## To use it in a project:
4 ## - uncomment the lines corresponding to used pins
5 ## - rename the used ports (in each line, after get_ports) according to the top level signal names in the project
6
7 #set_property -dict { PACKAGE_PIN K18      IOSTANDARD LVCMOS33 } [get_ports { a }]; #IO_L12N_T1_MRCC_35 Sch=btn[0]
8 #set_property -dict { PACKAGE_PIN P16      IOSTANDARD LVCMOS33 } [get_ports { b }]; #IO_L24N_T3_34 Sch=btn[1]
9 #set_property -dict { PACKAGE_PIN M14      IOSTANDARD LVCMOS33 } [get_ports { c }]; #IO_L23P_T3_35 Sch=led[0]
10
11 ##Clock signal
12 #set_property -dict { PACKAGE_PIN K17      IOSTANDARD LVCMOS33 } [get_ports { sysclk }]; #IO_L12P_T1_MRCC_35 Sch=sysclk
13 #create_clock -add -name sys_clk_pin -period 8.00 -waveform {0 4} [get_ports { sysclk }];
14
15 ##Switches
16 #set_property -dict { PACKAGE_PIN G15      IOSTANDARD LVCMOS33 } [get_ports { sw[0] }]; #IO_L19N_T3_VREF_35 Sch=sw[0]
17 #set_property -dict { PACKAGE_PIN P15      IOSTANDARD LVCMOS33 } [get_ports { sw[1] }]; #IO_L24P_T3_34 Sch=sw[1]
18 #set_property -dict { PACKAGE_PIN W13      IOSTANDARD LVCMOS33 } [get_ports { sw[2] }]; #IO_L4N_T0_34 Sch=sw[2]
19 #set_property -dict { PACKAGE_PIN T16      IOSTANDARD LVCMOS33 } [get_ports { sw[3] }]; #IO_L9P_T1_DQ5_34 Sch=sw[3]
20
21 ##Buttons
22 #set_property -dict { PACKAGE_PIN K18      IOSTANDARD LVCMOS33 } [get_ports { btn[0] }]; #IO_L12N_T1_MRCC_35 Sch=btn[0]
23 #set_property -dict { PACKAGE_PIN P16      IOSTANDARD LVCMOS33 } [get_ports { btn[1] }]; #IO_L24N_T3_34 Sch=btn[1]
24 #set_property -dict { PACKAGE_PIN K19      IOSTANDARD LVCMOS33 } [get_ports { btn[2] }]; #IO_L10P_T1_A01P_35 Sch=btn[2]
25 #set_property -dict { PACKAGE_PIN Y16      IOSTANDARD LVCMOS33 } [get_ports { btn[3] }]; #IO_L7P_T1_34 Sch=btn[3]
26
27 ##LEDs
28 #set_property -dict { PACKAGE_PIN M14      IOSTANDARD LVCMOS33 } [get_ports { led[0] }]; #IO_L23P_T3_35 Sch=led[0]
29 #set_property -dict { PACKAGE_PIN P15      IOSTANDARD LVCMOS33 } [get_ports { led[1] }]; #IO_L24N_T3_34 Sch=led[1]
30 #set_property -dict { PACKAGE_PIN G14      IOSTANDARD LVCMOS33 } [get_ports { led[2] }]; #IO_0_35 Sch=led[2]
31 #set_property -dict { PACKAGE_PIN D18      IOSTANDARD LVCMOS33 } [get_ports { led[3] }]; #IO_L3N_T0_DQ0_A01N_35 Sch=led[3]
32
33 ##RGB LED 5 (Zybo Z7-20 only)
34 #set_property -dict { PACKAGE_PIN Y11      IOSTANDARD LVCMOS33 } [get_ports { led5_r }]; #IO_L18W_T2_13 Sch=led5_r
35 #set_property -dict { PACKAGE_PIN T5       IOSTANDARD LVCMOS33 } [get_ports { led5_g }]; #IO_L19P_T3_13 Sch=led5_g
36 #set_property -dict { PACKAGE_PIN Y12      IOSTANDARD LVCMOS33 } [get_ports { led5_b }]; #IO_L20P_T3_13 Sch=led5_b
37
38 ##RGB LED 6
39 #set_property -dict { PACKAGE_PIN U18      IOSTANDARD LVCMOS33 } [get_ports { led6_r }]; #IO_L18P_T3_34 Sch=led6_r
40 #set_property -dict { PACKAGE_PIN U19      IOSTANDARD LVCMOS33 } [get_ports { led6_g }]; #IO_L19P_T3_34 Sch=led6_g
41 #set_property -dict { PACKAGE_PIN U20      IOSTANDARD LVCMOS33 } [get_ports { led6_b }]; #IO_L20P_T3_34 Sch=led6_b
42
43 ##RGB LED 7
44 #set_property -dict { PACKAGE_PIN U21      IOSTANDARD LVCMOS33 } [get_ports { led7_r }]; #IO_L20P_T3_34 Sch=led7_r
45 #set_property -dict { PACKAGE_PIN U22      IOSTANDARD LVCMOS33 } [get_ports { led7_g }]; #IO_L21P_T3_34 Sch=led7_g
46 #set_property -dict { PACKAGE_PIN U23      IOSTANDARD LVCMOS33 } [get_ports { led7_b }]; #IO_L22P_T3_34 Sch=led7_b
47
48 ##RGB LED 8
49 #set_property -dict { PACKAGE_PIN U24      IOSTANDARD LVCMOS33 } [get_ports { led8_r }]; #IO_L22P_T3_34 Sch=led8_r
50 #set_property -dict { PACKAGE_PIN U25      IOSTANDARD LVCMOS33 } [get_ports { led8_g }]; #IO_L23P_T3_35 Sch=led8_g
51 #set_property -dict { PACKAGE_PIN U26      IOSTANDARD LVCMOS33 } [get_ports { led8_b }]; #IO_L24P_T3_34 Sch=led8_b
52
53 ##RGB LED 9
54 #set_property -dict { PACKAGE_PIN U27      IOSTANDARD LVCMOS33 } [get_ports { led9_r }]; #IO_L24P_T3_34 Sch=led9_r
55 #set_property -dict { PACKAGE_PIN U28      IOSTANDARD LVCMOS33 } [get_ports { led9_g }]; #IO_L25P_T3_35 Sch=led9_g
56 #set_property -dict { PACKAGE_PIN U29      IOSTANDARD LVCMOS33 } [get_ports { led9_b }]; #IO_L26P_T3_35 Sch=led9_b
57
58 ##RGB LED 10
59 #set_property -dict { PACKAGE_PIN U30      IOSTANDARD LVCMOS33 } [get_ports { led10_r }]; #IO_L26P_T3_35 Sch=led10_r
60 #set_property -dict { PACKAGE_PIN U31      IOSTANDARD LVCMOS33 } [get_ports { led10_g }]; #IO_L27P_T3_35 Sch=led10_g
61 #set_property -dict { PACKAGE_PIN U32      IOSTANDARD LVCMOS33 } [get_ports { led10_b }]; #IO_L28P_T3_35 Sch=led10_b
62
63 ##RGB LED 11
64 #set_property -dict { PACKAGE_PIN U33      IOSTANDARD LVCMOS33 } [get_ports { led11_r }]; #IO_L28P_T3_35 Sch=led11_r
65 #set_property -dict { PACKAGE_PIN U34      IOSTANDARD LVCMOS33 } [get_ports { led11_g }]; #IO_L29P_T3_35 Sch=led11_g
66 #set_property -dict { PACKAGE_PIN U35      IOSTANDARD LVCMOS33 } [get_ports { led11_b }]; #IO_L30P_T3_35 Sch=led11_b
67
68 ##RGB LED 12
69 #set_property -dict { PACKAGE_PIN U36      IOSTANDARD LVCMOS33 } [get_ports { led12_r }]; #IO_L30P_T3_35 Sch=led12_r
70 #set_property -dict { PACKAGE_PIN U37      IOSTANDARD LVCMOS33 } [get_ports { led12_g }]; #IO_L31P_T3_35 Sch=led12_g
71 #set_property -dict { PACKAGE_PIN U38      IOSTANDARD LVCMOS33 } [get_ports { led12_b }]; #IO_L32P_T3_35 Sch=led12_b
72
73 ##RGB LED 13
74 #set_property -dict { PACKAGE_PIN U39      IOSTANDARD LVCMOS33 } [get_ports { led13_r }]; #IO_L32P_T3_35 Sch=led13_r
75 #set_property -dict { PACKAGE_PIN U40      IOSTANDARD LVCMOS33 } [get_ports { led13_g }]; #IO_L33P_T3_35 Sch=led13_g
76 #set_property -dict { PACKAGE_PIN U41      IOSTANDARD LVCMOS33 } [get_ports { led13_b }]; #IO_L34P_T3_35 Sch=led13_b
77
78 ##RGB LED 14
79 #set_property -dict { PACKAGE_PIN U42      IOSTANDARD LVCMOS33 } [get_ports { led14_r }]; #IO_L34P_T3_35 Sch=led14_r
80 #set_property -dict { PACKAGE_PIN U43      IOSTANDARD LVCMOS33 } [get_ports { led14_g }]; #IO_L35P_T3_35 Sch=led14_g
81 #set_property -dict { PACKAGE_PIN U44      IOSTANDARD LVCMOS33 } [get_ports { led14_b }]; #IO_L36P_T3_35 Sch=led14_b
82
83 ##RGB LED 15
84 #set_property -dict { PACKAGE_PIN U45      IOSTANDARD LVCMOS33 } [get_ports { led15_r }]; #IO_L36P_T3_35 Sch=led15_r
85 #set_property -dict { PACKAGE_PIN U46      IOSTANDARD LVCMOS33 } [get_ports { led15_g }]; #IO_L37P_T3_35 Sch=led15_g
86 #set_property -dict { PACKAGE_PIN U47      IOSTANDARD LVCMOS33 } [get_ports { led15_b }]; #IO_L38P_T3_35 Sch=led15_b
87
88 ##RGB LED 16
89 #set_property -dict { PACKAGE_PIN U48      IOSTANDARD LVCMOS33 } [get_ports { led16_r }]; #IO_L38P_T3_35 Sch=led16_r
90 #set_property -dict { PACKAGE_PIN U49      IOSTANDARD LVCMOS33 } [get_ports { led16_g }]; #IO_L39P_T3_35 Sch=led16_g
91 #set_property -dict { PACKAGE_PIN U50      IOSTANDARD LVCMOS33 } [get_ports { led16_b }]; #IO_L40P_T3_35 Sch=led16_b
92
93 ##RGB LED 17
94 #set_property -dict { PACKAGE_PIN U51      IOSTANDARD LVCMOS33 } [get_ports { led17_r }]; #IO_L40P_T3_35 Sch=led17_r
95 #set_property -dict { PACKAGE_PIN U52      IOSTANDARD LVCMOS33 } [get_ports { led17_g }]; #IO_L41P_T3_35 Sch=led17_g
96 #set_property -dict { PACKAGE_PIN U53      IOSTANDARD LVCMOS33 } [get_ports { led17_b }]; #IO_L42P_T3_35 Sch=led17_b
97
98 ##RGB LED 18
99 #set_property -dict { PACKAGE_PIN U54      IOSTANDARD LVCMOS33 } [get_ports { led18_r }]; #IO_L42P_T3_35 Sch=led18_r
100 #set_property -dict { PACKAGE_PIN U55      IOSTANDARD LVCMOS33 } [get_ports { led18_g }]; #IO_L43P_T3_35 Sch=led18_g
101 #set_property -dict { PACKAGE_PIN U56      IOSTANDARD LVCMOS33 } [get_ports { led18_b }]; #IO_L44P_T3_35 Sch=led18_b
102
103 ##RGB LED 19
104 #set_property -dict { PACKAGE_PIN U57      IOSTANDARD LVCMOS33 } [get_ports { led19_r }]; #IO_L44P_T3_35 Sch=led19_r
105 #set_property -dict { PACKAGE_PIN U58      IOSTANDARD LVCMOS33 } [get_ports { led19_g }]; #IO_L45P_T3_35 Sch=led19_g
106 #set_property -dict { PACKAGE_PIN U59      IOSTANDARD LVCMOS33 } [get_ports { led19_b }]; #IO_L46P_T3_35 Sch=led19_b
107
108 ##RGB LED 20
109 #set_property -dict { PACKAGE_PIN U60      IOSTANDARD LVCMOS33 } [get_ports { led20_r }]; #IO_L46P_T3_35 Sch=led20_r
110 #set_property -dict { PACKAGE_PIN U61      IOSTANDARD LVCMOS33 } [get_ports { led20_g }]; #IO_L47P_T3_35 Sch=led20_g
111 #set_property -dict { PACKAGE_PIN U62      IOSTANDARD LVCMOS33 } [get_ports { led20_b }]; #IO_L48P_T3_35 Sch=led20_b
112
113 ##RGB LED 21
114 #set_property -dict { PACKAGE_PIN U63      IOSTANDARD LVCMOS33 } [get_ports { led21_r }]; #IO_L48P_T3_35 Sch=led21_r
115 #set_property -dict { PACKAGE_PIN U64      IOSTANDARD LVCMOS33 } [get_ports { led21_g }]; #IO_L49P_T3_35 Sch=led21_g
116 #set_property -dict { PACKAGE_PIN U65      IOSTANDARD LVCMOS33 } [get_ports { led21_b }]; #IO_L50P_T3_35 Sch=led21_b
117
118 ##RGB LED 22
119 #set_property -dict { PACKAGE_PIN U66      IOSTANDARD LVCMOS33 } [get_ports { led22_r }]; #IO_L50P_T3_35 Sch=led22_r
120 #set_property -dict { PACKAGE_PIN U67      IOSTANDARD LVCMOS33 } [get_ports { led22_g }]; #IO_L51P_T3_35 Sch=led22_g
121 #set_property -dict { PACKAGE_PIN U68      IOSTANDARD LVCMOS33 } [get_ports { led22_b }]; #IO_L52P_T3_35 Sch=led22_b
122
123 ##RGB LED 23
124 #set_property -dict { PACKAGE_PIN U69      IOSTANDARD LVCMOS33 } [get_ports { led23_r }]; #IO_L52P_T3_35 Sch=led23_r
125 #set_property -dict { PACKAGE_PIN U70      IOSTANDARD LVCMOS33 } [get_ports { led23_g }]; #IO_L53P_T3_35 Sch=led23_g
126 #set_property -dict { PACKAGE_PIN U71      IOSTANDARD LVCMOS33 } [get_ports { led23_b }]; #IO_L54P_T3_35 Sch=led23_b
127
128 ##RGB LED 24
129 #set_property -dict { PACKAGE_PIN U72      IOSTANDARD LVCMOS33 } [get_ports { led24_r }]; #IO_L54P_T3_35 Sch=led24_r
130 #set_property -dict { PACKAGE_PIN U73      IOSTANDARD LVCMOS33 } [get_ports { led24_g }]; #IO_L55P_T3_35 Sch=led24_g
131 #set_property -dict { PACKAGE_PIN U74      IOSTANDARD LVCMOS33 } [get_ports { led24_b }]; #IO_L56P_T3_35 Sch=led24_b
132
133 ##RGB LED 25
134 #set_property -dict { PACKAGE_PIN U75      IOSTANDARD LVCMOS33 } [get_ports { led25_r }]; #IO_L56P_T3_35 Sch=led25_r
135 #set_property -dict { PACKAGE_PIN U76      IOSTANDARD LVCMOS33 } [get_ports { led25_g }]; #IO_L57P_T3_35 Sch=led25_g
136 #set_property -dict { PACKAGE_PIN U77      IOSTANDARD LVCMOS33 } [get_ports { led25_b }]; #IO_L58P_T3_35 Sch=led25_b
137
138 ##RGB LED 26
139 #set_property -dict { PACKAGE_PIN U78      IOSTANDARD LVCMOS33 } [get_ports { led26_r }]; #IO_L58P_T3_35 Sch=led26_r
140 #set_property -dict { PACKAGE_PIN U79      IOSTANDARD LVCMOS33 } [get_ports { led26_g }]; #IO_L59P_T3_35 Sch=led26_g
141 #set_property -dict { PACKAGE_PIN U80      IOSTANDARD LVCMOS33 } [get_ports { led26_b }]; #IO_L60P_T3_35 Sch=led26_b
142
143 ##RGB LED 27
144 #set_property -dict { PACKAGE_PIN U81      IOSTANDARD LVCMOS33 } [get_ports { led27_r }]; #IO_L60P_T3_35 Sch=led27_r
145 #set_property -dict { PACKAGE_PIN U82      IOSTANDARD LVCMOS33 } [get_ports { led27_g }]; #IO_L61P_T3_35 Sch=led27_g
146 #set_property -dict { PACKAGE_PIN U83      IOSTANDARD LVCMOS33 } [get_ports { led27_b }]; #IO_L62P_T3_35 Sch=led27_b
147
148 ##RGB LED 28
149 #set_property -dict { PACKAGE_PIN U84      IOSTANDARD LVCMOS33 } [get_ports { led28_r }]; #IO_L62P_T3_35 Sch=led28_r
150 #set_property -dict { PACKAGE_PIN U85      IOSTANDARD LVCMOS33 } [get_ports { led28_g }]; #IO_L63P_T3_35 Sch=led28_g
151 #set_property -dict { PACKAGE_PIN U86      IOSTANDARD LVCMOS33 } [get_ports { led28_b }]; #IO_L64P_T3_35 Sch=led28_b
152
153 ##RGB LED 29
154 #set_property -dict { PACKAGE_PIN U87      IOSTANDARD LVCMOS33 } [get_ports { led29_r }]; #IO_L64P_T3_35 Sch=led29_r
155 #set_property -dict { PACKAGE_PIN U88      IOSTANDARD LVCMOS33 } [get_ports { led29_g }]; #IO_L65P_T3_35 Sch=led29_g
156 #set_property -dict { PACKAGE_PIN U89      IOSTANDARD LVCMOS33 } [get_ports { led29_b }]; #IO_L66P_T3_35 Sch=led29_b
157
158 ##RGB LED 30
159 #set_property -dict { PACKAGE_PIN U90      IOSTANDARD LVCMOS33 } [get_ports { led30_r }]; #IO_L66P_T3_35 Sch=led30_r
160 #set_property -dict { PACKAGE_PIN U91      IOSTANDARD LVCMOS33 } [get_ports { led30_g }]; #IO_L67P_T3_35 Sch=led30_g
161 #set_property -dict { PACKAGE_PIN U92      IOSTANDARD LVCMOS33 } [get_ports { led30_b }]; #IO_L68P_T3_35 Sch=led30_b
162
163 ##RGB LED 31
164 #set_property -dict { PACKAGE_PIN U93      IOSTANDARD LVCMOS33 } [get_ports { led31_r }]; #IO_L68P_T3_35 Sch=led31_r
165 #set_property -dict { PACKAGE_PIN U94      IOSTANDARD LVCMOS33 } [get_ports { led31_g }]; #IO_L69P_T3_35 Sch=led31_g
166 #set_property -dict { PACKAGE_PIN U95      IOSTANDARD LVCMOS33 } [get_ports { led31_b }]; #IO_L70P_T3_35 Sch=led31_b
167
168 ##RGB LED 32
169 #set_property -dict { PACKAGE_PIN U96      IOSTANDARD LVCMOS33 } [get_ports { led32_r }]; #IO_L70P_T3_35 Sch=led32_r
170 #set_property -dict { PACKAGE_PIN U97      IOSTANDARD LVCMOS33 } [get_ports { led32_g }]; #IO_L71P_T3_35 Sch=led32_g
171 #set_property -dict { PACKAGE_PIN U98      IOSTANDARD LVCMOS33 } [get_ports { led32_b }]; #IO_L72P_T3_35 Sch=led32_b
172
173 ##RGB LED 33
174 #set_property -dict { PACKAGE_PIN U99      IOSTANDARD LVCMOS33 } [get_ports { led33_r }]; #IO_L72P_T3_35 Sch=led33_r
175 #set_property -dict { PACKAGE_PIN U100     IOSTANDARD LVCMOS33 } [get_ports { led33_g }]; #IO_L73P_T3_35 Sch=led33_g
176 #set_property -dict { PACKAGE_PIN U101     IOSTANDARD LVCMOS33 } [get_ports { led33_b }]; #IO_L74P_T3_35 Sch=led33_b
177
178 ##RGB LED 34
179 #set_property -dict { PACKAGE_PIN U102     IOSTANDARD LVCMOS33 } [get_ports { led34_r }]; #IO_L74P_T3_35 Sch=led34_r
180 #set_property -dict { PACKAGE_PIN U103     IOSTANDARD LVCMOS33 } [get_ports { led34_g }]; #IO_L75P_T3_35 Sch=led34_g
181 #set_property -dict { PACKAGE_PIN U104     IOSTANDARD LVCMOS33 } [get_ports { led34_b }]; #IO_L76P_T3_35 Sch=led34_b
182
183 ##RGB LED 35
184 #set_property -dict { PACKAGE_PIN U105     IOSTANDARD LVCMOS33 } [get_ports { led35_r }]; #IO_L76P_T3_35 Sch=led35_r
185 #set_property -dict { PACKAGE_PIN U106     IOSTANDARD LVCMOS33 } [get_ports { led35_g }]; #IO_L77P_T3_35 Sch=led35_g
186 #set_property -dict { PACKAGE_PIN U107     IOSTANDARD LVCMOS33 } [get_ports { led35_b }]; #IO_L78P_T3_35 Sch=led35_b
187
188 ##RGB LED 36
189 #set_property -dict { PACKAGE_PIN U108     IOSTANDARD LVCMOS33 } [get_ports { led36_r }]; #IO_L78P_T3_35 Sch=led36_r
190 #set_property -dict { PACKAGE_PIN U109     IOSTANDARD LVCMOS33 } [get_ports { led36_g }]; #IO_L79P_T3_35 Sch=led36_g
191 #set_property -dict { PACKAGE_PIN U110     IOSTANDARD LVCMOS33 } [get_ports { led36_b }]; #IO_L80P_T3_35 Sch=led36_b
192
193 ##RGB LED 37
194 #set_property -dict { PACKAGE_PIN U111     IOSTANDARD LVCMOS33 } [get_ports { led37_r }]; #IO_L80P_T3_35 Sch=led37_r
195 #set_property -dict { PACKAGE_PIN U112     IOSTANDARD LVCMOS33 } [get_ports { led37_g }]; #IO_L81P_T3_35 Sch=led37_g
196 #set_property -dict { PACKAGE_PIN U113     IOSTANDARD LVCMOS33 } [get_ports { led37_b }]; #IO_L82P_T3_35 Sch=led37_b
197
198 ##RGB LED 38
199 #set_property -dict { PACKAGE_PIN U114     IOSTANDARD LVCMOS33 } [get_ports { led38_r }]; #IO_L82P_T3_35 Sch=led38_r
200 #set_property -dict { PACKAGE_PIN U115     IOSTANDARD LVCMOS33 } [get_ports { led38_g }]; #IO_L83P_T3_35 Sch=led38_g
201 #set_property -dict { PACKAGE_PIN U116     IOSTANDARD LVCMOS33 } [get_ports { led38_b }]; #IO_L84P_T3_35 Sch=led38_b
202
203 ##RGB LED 39
204 #set_property -dict { PACKAGE_PIN U117     IOSTANDARD LVCMOS33 } [get_ports { led39_r }]; #IO_L84P_T3_35 Sch=led39_r
205 #set_property -dict { PACKAGE_PIN U118     IOSTANDARD LVCMOS33 } [get_ports { led39_g }]; #IO_L85P_T3_35 Sch=led39_g
206 #set_property -dict { PACKAGE_PIN U119     IOSTANDARD LVCMOS33 } [get_ports { led39_b }]; #IO_L86P_T3_35 Sch=led39_b
207
208 ##RGB LED 40
209 #set_property -dict { PACKAGE_PIN U120     IOSTANDARD LVCMOS33 } [get_ports { led40_r }]; #IO_L86P_T3_35 Sch=led40_r
210 #set_property -dict { PACKAGE_PIN U121     IOSTANDARD LVCMOS33 } [get_ports { led40_g }]; #IO_L87P_T3_35 Sch=led40_g
211 #set_property -dict { PACKAGE_PIN U122     IOSTANDARD LVCMOS33 } [get_ports { led40_b }]; #IO_L88P_T3_35 Sch=led40_b
212
213 ##RGB LED 41
214 #set_property -dict { PACKAGE_PIN U123     IOSTANDARD LVCMOS33 } [get_ports { led41_r }]; #IO_L88P_T3_35 Sch=led41_r
215 #set_property -dict { PACKAGE_PIN U124     IOSTANDARD LVCMOS33 } [get_ports { led41_g }]; #IO_L89P_T3_35 Sch=led41_g
216 #set_property -dict { PACKAGE_PIN U125     IOSTANDARD LVCMOS33 } [get_ports { led41_b }]; #IO_L90P_T3_35 Sch=led41_b
217
218 ##RGB LED 42
219 #set_property -dict { PACKAGE_PIN U126     IOSTANDARD LVCMOS33 } [get_ports { led42_r }]; #IO_L90P_T3_35 Sch=led42_r
220 #set_property -dict { PACKAGE_PIN U127     IOSTANDARD LVCMOS33 } [get_ports { led42_g }]; #IO_L91P_T3_35 Sch=led42_g
221 #set_property -dict { PACKAGE_PIN U128     IOSTANDARD LVCMOS33 } [get_ports { led42_b }]; #IO_L92P_T3_35 Sch=led42_b
222
223 ##RGB LED 43
224 #set_property -dict { PACKAGE_PIN U129     IOSTANDARD LVCMOS33 } [get_ports { led43_r }]; #IO_L92P_T3_35 Sch=led43_r
225 #set_property -dict { PACKAGE_PIN U130     IOSTANDARD LVCMOS33 } [get_ports { led43_g }]; #IO_L93P_T3_35 Sch=led43_g
226 #set_property -dict { PACKAGE_PIN U131     IOSTANDARD LVCMOS33 } [get_ports { led43_b }]; #IO_L94P_T3_35 Sch=led43_b
227
228 ##RGB LED 44
229 #set_property -dict { PACKAGE_PIN U132     IOSTANDARD LVCMOS33 } [get_ports { led44_r }]; #IO_L94P_T3_35 Sch=led44_r
230 #set_property -dict { PACKAGE_PIN U133     IOSTANDARD LVCMOS33 } [get_ports { led44_g }]; #IO_L95P_T3_35 Sch=led44_g
231 #set_property -dict { PACKAGE_PIN U134     IOSTANDARD LVCMOS33 } [get_ports { led44_b }]; #IO_L96P_T3_35 Sch=led44_b
232
233 ##RGB LED 45
234 #set_property -dict { PACKAGE_PIN U135     IOSTANDARD LVCMOS33 } [get_ports { led45_r }]; #IO_L96P_T3_35 Sch=led45_r
235 #set_property -dict { PACKAGE_PIN U136     IOSTANDARD LVCMOS33 } [get_ports { led45_g }]; #IO_L97P_T3_35 Sch=led45_g
236 #set_property -dict { PACKAGE_PIN U137     IOSTANDARD LVCMOS33 } [get_ports { led45_b }]; #IO_L98P_T3_35 Sch=led45_b
237
238 ##RGB LED 46
239 #set_property -dict { PACKAGE_PIN U138     IOSTANDARD LVCMOS33 } [get_ports { led46_r }]; #IO_L98P_T3_35 Sch=led46_r
240 #set_property -dict { PACKAGE_PIN U139     IOSTANDARD LVCMOS33 } [get_ports { led46_g }]; #IO_L99P_T3_35 Sch=led46_g
241 #set_property -dict { PACKAGE_PIN U140     IOSTANDARD LVCMOS33 } [get_ports { led46_b }]; #IO_L100P_T3_35 Sch=led46_b
242
243 ##RGB LED 47
244 #set_property -dict { PACKAGE_PIN U141     IOSTANDARD LVCMOS33 } [get_ports { led47_r }]; #IO_L99P_T3_35 Sch=led47_r
245 #set_property -dict { PACKAGE_PIN U142     IOSTANDARD LVCMOS33 } [get_ports { led47_g }]; #IO_L101P_T3_35 Sch=led47_g
246 #set_property -dict { PACKAGE_PIN U143     IOSTANDARD LVCMOS33 } [get_ports { led47_b }]; #IO_L102P_T3_35 Sch=led47_b
247
248 ##RGB LED 48
249 #set_property -dict { PACKAGE_PIN U144     IOSTANDARD LVCMOS33 } [get_ports { led48_r }]; #IO_L102P_T3_35 Sch=led48_r
250 #set_property -dict { PACKAGE_PIN U145     IOSTANDARD LVCMOS33 } [get_ports { led48_g }]; #IO_L103P_T3_35 Sch=led48_g
251 #set_property -dict { PACKAGE_PIN U146     IOSTANDARD LVCMOS33 } [get_ports { led48_b }]; #IO_L104P_T3_35 Sch=led48_b
252
253 ##RGB LED 49
254 #set_property -dict { PACKAGE_PIN U147     IOSTANDARD LVCMOS33 } [get_ports { led49_r }]; #IO_L104P_T3_35 Sch=led49_r
255 #set_property -dict { PACKAGE_PIN U148     IOSTANDARD LVCMOS33 } [get_ports { led49_g }]; #IO_L105P_T3_35 Sch=led49_g
256 #set_property -dict { PACKAGE_PIN U149     IOSTANDARD LVCMOS33 } [get_ports { led49_b }]; #IO_L106P_T3_35 Sch=led49_b
257
258 ##RGB LED 50
259 #set_property -dict { PACKAGE_PIN U150     IOSTANDARD LVCMOS33 } [get_ports { led50_r }]; #IO_L106P_T3_35 Sch=led50_r
260 #set_property -dict { PACKAGE_PIN U151     IOSTANDARD LVCMOS33 } [get_ports { led50_g }]; #IO_L107P_T3_35 Sch=led50_g
261 #set_property -dict { PACKAGE_PIN U152     IOSTANDARD LVCMOS33 } [get_ports { led50_b }]; #IO_L108P_T3_35 Sch=led50_b
262
263 ##RGB LED 51
264 #set_property -dict { PACKAGE_PIN U153     IOSTANDARD LVCMOS33 } [get_ports { led51_r }]; #IO_L108P_T3_35 Sch=led51_r
265 #set_property -dict { PACKAGE_PIN U154     IOSTANDARD LVCMOS33 } [get_ports { led51_g }]; #IO_L109P_T3_35 Sch=led51_g
266 #set_property -dict { PACKAGE_PIN U155     IOSTANDARD LVCMOS33 } [get_ports { led51_b }]; #IO_L110P_T3_35 Sch=led51_b
267
268 ##RGB LED 52
269 #set_property -dict { PACKAGE_PIN U156     IOSTANDARD LVCMOS33 } [get_ports { led52_r }]; #IO_L110P_T3_35 Sch=led52_r
270 #set_property -dict { PACKAGE_PIN U157     IOSTANDARD LVCMOS33 } [get_ports { led52_g }]; #IO_L111P_T3_35 Sch=led52_g
271 #set_property -dict { PACKAGE_PIN U158     IOSTANDARD LVCMOS33 } [get_ports { led52_b }]; #IO_L112P_T3_35 Sch=led52_b
272
273 ##RGB LED 53
274 #set_property -dict { PACKAGE_PIN U159     IOSTANDARD LVCMOS33 } [get_ports { led53_r }]; #IO_L112P_T3_35 Sch=led53_r
275 #set_property -dict { PACKAGE_PIN U160     IOSTANDARD LVCMOS33 } [get_ports { led53_g }]; #IO_L113P_T3_35 Sch=led53_g
276 #set_property -dict { PACKAGE_PIN U161     IOSTANDARD LVCMOS33 } [get_ports { led53_b }]; #IO_L114P_T3_35 Sch=led53_b
277
278 ##RGB LED 54
279 #set_property -dict { PACKAGE_PIN U162     IOSTANDARD LVCMOS33 } [get_ports { led54_r }]; #IO_L114P_T3_35 Sch=led54_r
280 #set_property -dict { PACKAGE_PIN U163     IOSTANDARD LVCMOS33 } [get_ports { led54_g }]; #IO_L115P_T3_35 Sch=led54_g
281 #set_property -dict { PACKAGE_PIN U164     IOSTANDARD LVCMOS33 } [get_ports { led54_b }]; #IO_L116P_T3_35 Sch=led54_b
282
283 ##RGB LED 55
284 #set_property -dict { PACKAGE_PIN U165     IOSTANDARD LVCMOS33 } [get_ports { led55_r }]; #IO_L116P_T3_35 Sch=led55_r
285 #set_property -dict { PACKAGE_PIN U166     IOSTANDARD LVCMOS33 } [get_ports { led55_g }]; #IO_L117P_T3_35 Sch=led55_g
286 #set_property -dict { PACKAGE_PIN U167     IOSTANDARD LVCMOS33 } [get_ports { led55_b }]; #IO_L118P_T3_35 Sch=led55_b
287
288 ##RGB LED 56
289 #set_property -dict { PACKAGE_PIN U168     IOSTANDARD LVCMOS33 } [get_ports { led56_r }]; #IO_L118P_T3_35 Sch=led56_r
290 #set_property -dict { PACKAGE_PIN U169     IOSTANDARD LVCMOS33 } [get_ports { led56_g }]; #IO_L119P_T3_35 Sch=led56_g
291 #set_property -dict { PACKAGE_PIN U170     IOSTANDARD LVCMOS33 } [get_ports { led56_b }]; #IO_L120P_T3_35 Sch=led56_b
292
293 ##RGB LED 57
294 #set_property -dict { PACKAGE_PIN U171     IOSTANDARD LVCMOS33 } [get_ports { led57_r }]; #IO_L120P_T3_35 Sch=led57_r
295 #set_property -dict { PACKAGE_PIN U172     IOSTANDARD LVCMOS33 } [get_ports { led57_g }]; #IO_L121P_T3_35 Sch=led57_g
296 #set_property -dict { PACKAGE_PIN U173     IOSTANDARD LVCMOS33 } [get_ports { led57_b }]; #IO_L122P_T3_35 Sch=led57_b
297
298 ##RGB LED 58
299 #set_property -dict { PACKAGE_PIN U174     IOSTANDARD LVCMOS33 } [get_ports { led58_r }]; #IO_L122P_T3_35 Sch=led58_r
300 #set_property -dict { PACKAGE_PIN U175     IOSTANDARD LVCMOS33 } [get_ports { led58_g }]; #IO_L123P_T3_35 Sch=led58_g
301 #set_property -dict { PACKAGE_PIN U176     IOSTANDARD LVCMOS33 } [get_ports { led58_b }]; #IO_L124P_T3_35 Sch=led58_b
302
303 ##RGB LED 59
304 #set_property -dict { PACKAGE_PIN U177     IOSTANDARD LVCMOS33 } [get_ports { led59_r }]; #IO_L124P_T3_35 Sch=led59_r
305 #set_property -dict { PACKAGE_PIN U178     IOSTANDARD LVCMOS33 } [get_ports { led59_g }]; #IO_L125P_T3_35 Sch=led59_g
306 #set_property -dict { PACKAGE_PIN U179     IOSTANDARD LVCMOS33 } [get_ports { led59_b }]; #IO_L126P_T3_35 Sch=led59_b
307
308 ##RGB LED 60
309 #set_property -dict { PACKAGE_PIN U180     IOSTANDARD LVCMOS33 } [get_ports { led60_r }]; #IO_L126P_T3_35 Sch=led60_r
310 #set_property -dict { PACKAGE_PIN U181     IOSTANDARD LVCMOS33 } [get_ports { led60_g }]; #IO_L127P_T3_35 Sch=led60_g
311 #set_property -dict { PACKAGE_PIN U182     IOSTANDARD LVCMOS33 } [get_ports { led60_b }]; #IO_L128P_T3_35 Sch=led60_b
312
313 ##RGB LED 61
314 #set_property -dict { PACKAGE_PIN U183     IOSTANDARD LVCMOS33 } [get_ports { led61_r }]; #IO_L128P_T3_35 Sch=led61_r
315 #set_property -dict { PACKAGE_PIN U184     IOSTANDARD LVCMOS33 } [get_ports { led61_g }]; #IO_L129P_T3_35 Sch=led61_g
316 #set_property -dict { PACKAGE_PIN U185     IOSTANDARD LVCMOS33 } [get_ports { led61_b }]; #IO_L130P_T3_35 Sch=led61_b
317
318 ##RGB LED 62
319 #set_property -dict { PACKAGE_PIN U186     IOSTANDARD LVCMOS33 } [get_ports { led62_r }]; #IO_L130P_T3_35 Sch=led62_r
320 #set_property -dict { PACKAGE_PIN U187     IOSTANDARD LVCMOS33 } [get_ports { led62_g }]; #IO_L131P_T3_35 Sch=led62_g
321 #set_property -dict { PACKAGE_PIN U188     IOSTANDARD LVCMOS33 } [get_ports { led62_b }]; #IO_L132P_T3_35 Sch=led62_b
322
323 ##RGB LED 63
324 #set_property -dict { PACKAGE_PIN U189     IOSTANDARD LVCMOS33 } [get_ports { led63_r }]; #IO_L132P_T3_35 Sch=led63_r
325 #set_property -dict { PACKAGE_PIN U190     IOSTANDARD LVCMOS33 } [get_ports { led63_g }]; #IO_L133P_T3_35 Sch=led63_g
326 #set_property -dict { PACKAGE_PIN U191     IOSTANDARD LVCMOS33 } [get_ports { led63_b }]; #IO_L134P_T3_35 Sch=led63_b
327
328 ##RGB LED 64
329 #set_property -dict { PACKAGE_PIN U192     IOSTANDARD LVCMOS33 } [get_ports { led64_r }]; #IO_L134P_T3_35 Sch=led64_r
330 #set_property -dict { PACKAGE_PIN U193     IOSTANDARD LVCMOS33 } [get_ports { led64_g }]; #IO_L135P_T3_35 Sch=led64_g
331 #set_property -dict { PACKAGE_PIN U194     IOSTANDARD LVCMOS33 } [get_ports { led64_b }]; #IO_L136P_T3_35 Sch=led64_b
332
333 ##RGB LED 65
334 #set_property -dict { PACKAGE_PIN U195     IOSTANDARD LVCMOS33 } [get_ports { led65_r }]; #IO_L136P_T3_35 Sch=led65_r
335 #set_property -dict { PACKAGE_PIN U196     IOSTANDARD LVCMOS33 } [get_ports { led65_g }]; #IO_L137P_T3_35 Sch=led65_g
336 #set_property -dict { PACKAGE_PIN U197     IOSTANDARD LVCMOS33 } [get_ports { led65_b }]; #IO_L138P_T3_35 Sch=led65_b
337
338 ##RGB LED 66
339 #set_property -dict { PACKAGE_PIN U198     IOSTANDARD LVCMOS33 } [get_ports { led66_r }]; #IO_L138P_T3_35 Sch=led66_r
340 #set_property -dict { PACKAGE_PIN U199     IOSTANDARD LVCMOS33 } [get_ports { led66_g }]; #IO_L139P_T3_35 Sch=led66_g
341 #set_property -dict { PACKAGE_PIN U200     IOSTANDARD LVCMOS33 } [get_ports { led66_b }]; #IO_L140P_T3_35 Sch=led66_b
342
343 ##RGB LED 67
344 #set_property -dict { PACKAGE_PIN U201     IOSTANDARD LVCMOS33 } [get_ports { led67_r }]; #IO_L140P_T3_35 Sch=led67_r
345 #set_property -dict { PACKAGE_PIN U202     IOSTANDARD LVCMOS33 } [get_ports { led67_g }]; #IO_L141P_T3_35 Sch=led67_g
346 #set_property -dict { PACKAGE_PIN U203     IOSTANDARD LVCMOS33 } [get_ports { led67_b }]; #IO_L142P_T3_35 Sch=led67_b
347
348 ##RGB LED 68
349 #set_property -dict { PACKAGE_PIN U204     IOSTANDARD LVCMOS33 } [get_ports { led68_r }]; #IO_L142P_T3_35 Sch=led68_r
350 #set_property -dict { PACKAGE_PIN U205     IOSTANDARD LVCMOS33 } [get_ports { led68_g }]; #IO_L143P_T3_35 Sch=led68_g
351 #set_property -dict { PACKAGE_PIN U206     IOSTANDARD LVCMOS33 } [get
```

Bibliografía

- **“Essentials of Computer Organization and Architecture”**, 5th Edition
Linda Null, Julia Lobur - Jones and Bartlett Publishers - 2018.
 - **Chapter 3 - Boolean Algebra and Digital Logic:**
 - 3.2 - Boolean Algebra, 3.3 - Logic Gates
 - 3.7 - Sequential Circuits, 3.6 - Combinational Circuits
 - 3.7.1 - Basic Concepts, 3.7.2 - Clocks, 3.7.3 - Flip-Flops
- **“Syntesis of Arithmetic Circuits FPGA, ASIC, and Embedded Systems”**
Jean-Pierre Deschamps, Gery Jean Antoine Biol, Gustavo D. Sutter - John Wiley & Sons - 2006
 - Chapter 9 - Hardware Plataforms → 9.4 - Field Programmable Gate Array (FPGA) - Pag. 258-266
- **Zynq-7000 SoC Technical Reference Manual (UG585)**
URL: <https://docs.xilinx.com/v/u/en-US/ug585-Zynq-7000-TRM>

¡Gracias!