XD

Soc-eame  | soceame.wordpress.com

diseño de un SoC basado en Zynq-7000

Índice

[Introducción 2](#_Toc61470104)

[¿Qué es un SoC? 3](#_Toc61470105)

[Diferencia entre un SoC, un microcontrolador y una FPGA 3](#_Toc61470106)

[Comienzo 4](#_Toc61470107)

[Número de pines 4](#_Toc61470108)

[Pinout 6](#_Toc61470109)

[Nombres de los pines 6](#_Toc61470110)

[Definición de los pines. 9](#_Toc61470111)

[Huellas de los encapsulados 10](#_Toc61470112)

[Ejemplo 11](#_Toc61470113)

[Introducción 11](#_Toc61470114)

[Documentación 11](#_Toc61470115)

[Pinout 11](#_Toc61470116)

[Símbolo. 12](#_Toc61470117)

# Introducción

En este documento se detallará de la forma más extensa y detallada la forma de crear un SoC basado en un Zynq-7000 de Xilinx, detallando dónde encontrar los documentos y la información necesaria para la construcción de uno, además también se incluye cómo se configura en Vivado para que reconozca la nueva placa como una placa preconfigurada.

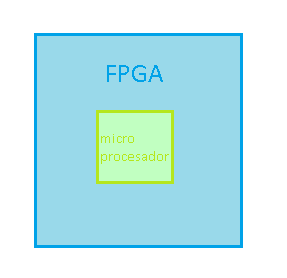
Al final del documento se incluye un ejemplo de cómo construir un SoC basado en un núcleo xc7z010-clg400 [este ejemplo se corresponde con la construcción de una Zynq-eame 0]

Este documento ha sido desarrollado por el blog <https://soceame.wordpress.com/> Para cualquier aporte o mejora [soceame@gmail.com](mailto:soceame@gmail.com).

Y para cualquier aportación económica para futuros proyectos <http://paypal.me/soceame>

# ¿Qué es un SoC?

Un SoC es la combinación de la electrónica de una FPGA con un microprocesador embebido en un chip.



## Diferencia entre un SoC, un microcontrolador y una FPGA

|  |  |  |  |
| --- | --- | --- | --- |
|  | SoC | Microcontrolador | FPGA |
| Microprocesador | ✔ | ✔ | ❌  (pero se le puede configurar un soft-core)\* |
| Hardware programable | ✔ | ❌ | ✔ |
| Lenguajes de programación | HDLs y lenguajes de medio-bajo nivel | Lenguajes de bajo-medio nivel: C, C++ | HDLs\*\*:  VHDL, Verilog |
| Precio del chip | Medio-alto  (de la familia Zynq-7000, de 35€ hasta 20.000€) | Bajo | Bajo-Medio |

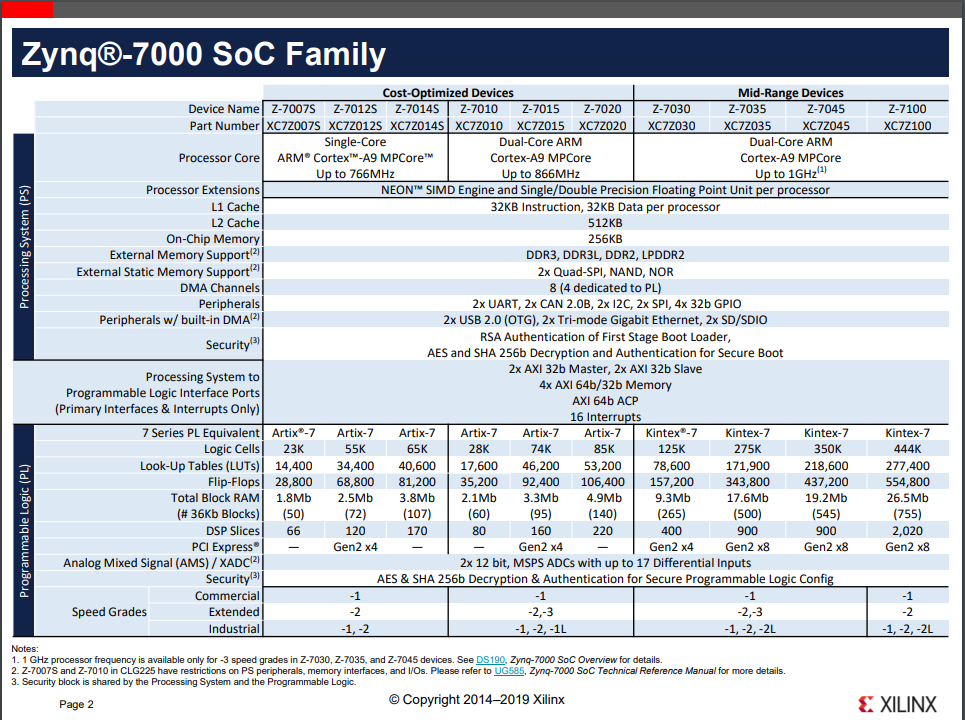
\* Un soft-core en un microprocesador creado usando las puertas lógicas internas del chip

\*\* Los lenguajes HDL son “lenguajes de descripción de hardware” por lo que con ellos se puede programar el hardware

# Comienzo

Lo primero que hay que hacer para construir una Zynq-7000 es saber los tipos de Zynq que hay.

Para ello se recurre al [Product Selection Guide](https://www.xilinx.com/support/documentation/selection-guides/zynq-7000-product-selection-guide.pdf) de Xilinx para la Zynq-7000. En él se encuentra una tabla como la siguiente en la que se desglosa qué contienen internamente los diferentes tipos de Zynq.

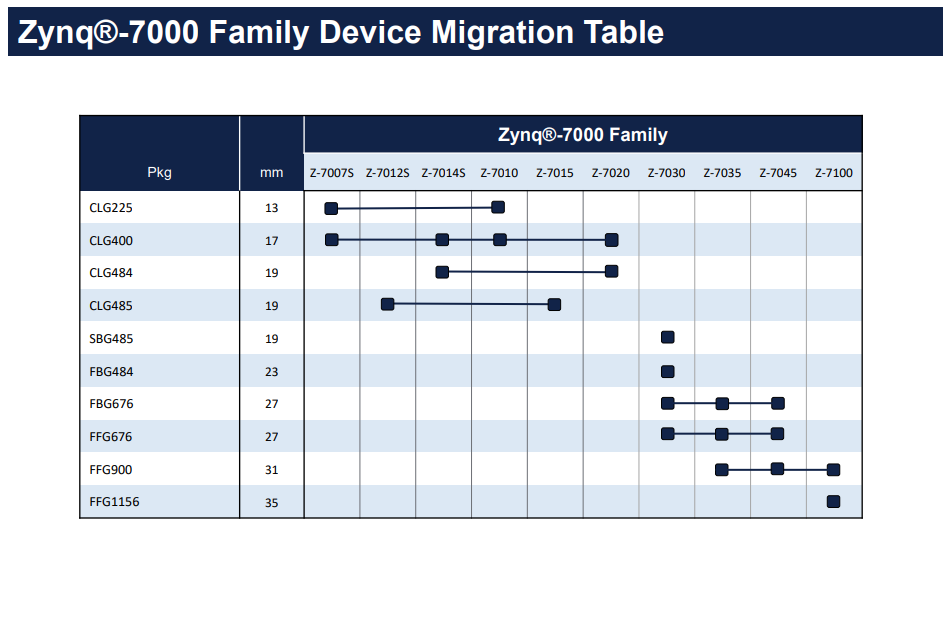


Como se puede ver hay una parte de la familia que solo tiene un microprocesador, mientras el resto tiene dos microprocesadores dentro con diferentes velocidades.

Otro factor condicionante a la hora de elegir un chip son las puertas lógicas internas, para ello es conveniente elegir más puertas lógicas de las que se van a necesitar para evitar problemas de falta de celdas.

## Número de pines

Para construir un SoC también hay que tener en cuenta el número de pines que va a tener la salida para poder crear el símbolo en el programa de diseño. En el mismo documento de antes existe la tabla que determina los pines que puede tener cada chip



Según la tabla si se quiere construir un SoC con el Z-7007S se puede elegir el encapsulado CLG225 y el CLG400, se diferencian en que el encapsulado CLG225 tiene 225 pines en una matriz de 15 filas x 15 columnas, y el CLG400 igual pero con 400 pines.

A partir de ahora se va a documentar la construcción de SoC.

Para una mayor documentación sobre la familia Zynq-7000 👇

<https://www.xilinx.com/support/documentation/data_sheets/ds190-Zynq-7000-Overview.pdf>

# Pinout

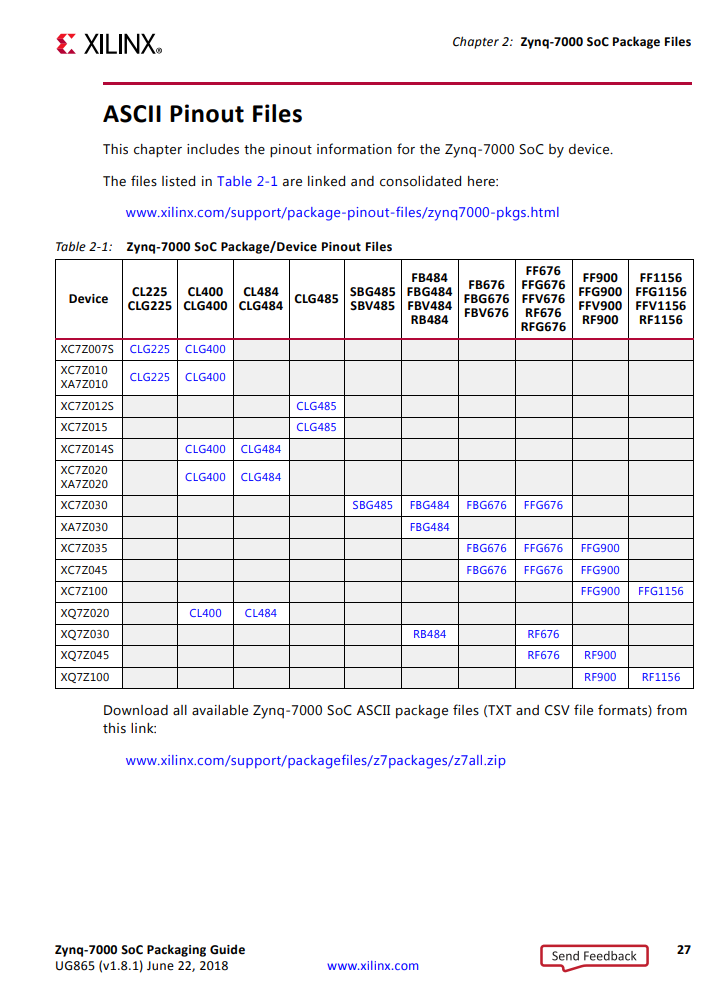
Para localizar la configuración del número y el tipo de pines se tiene que recurrir al documento [Packaging and Pinout](https://www.xilinx.com/support/documentation/user_guides/ug865-Zynq-7000-Pkg-Pinout.pdf).

En este documento figuran los nombres de los pines, la definición de los pines y las huellas de los diferentes encapsulados.

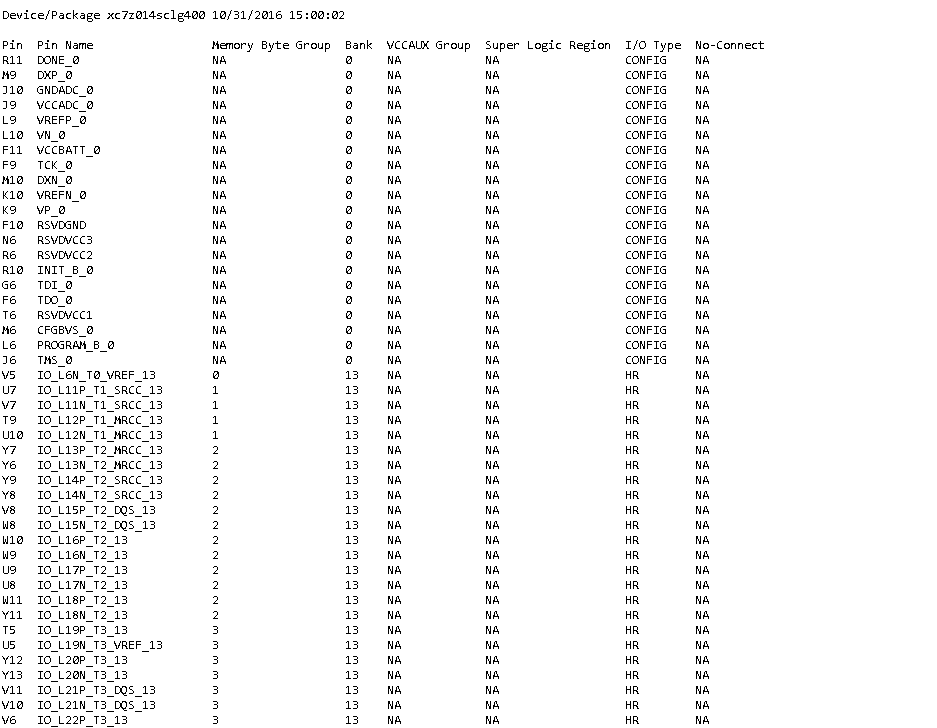
## Nombres de los pines

❗❗ Esto es extremadamente importante porque NO figuran implícitamente en el documento, figuran como un enlace a otros documentos❗❗

En la página 27 del documento figuran los enlaces a los respectivos documentos en función del chip y del encapsulado



Estos enlaces llevan a páginas como la siguiente, en la que los pines están clasificados en bancos.



Los campos importantes son:

En la primera columna figura el número del pin\*(este número es importante porque luego se utilizará para generar el XDC).

La segunda columna se corresponde con el nombre del pin, este nombre es importante para saber cuales son los bits de mayor o menor peso o la dedicación que tiene

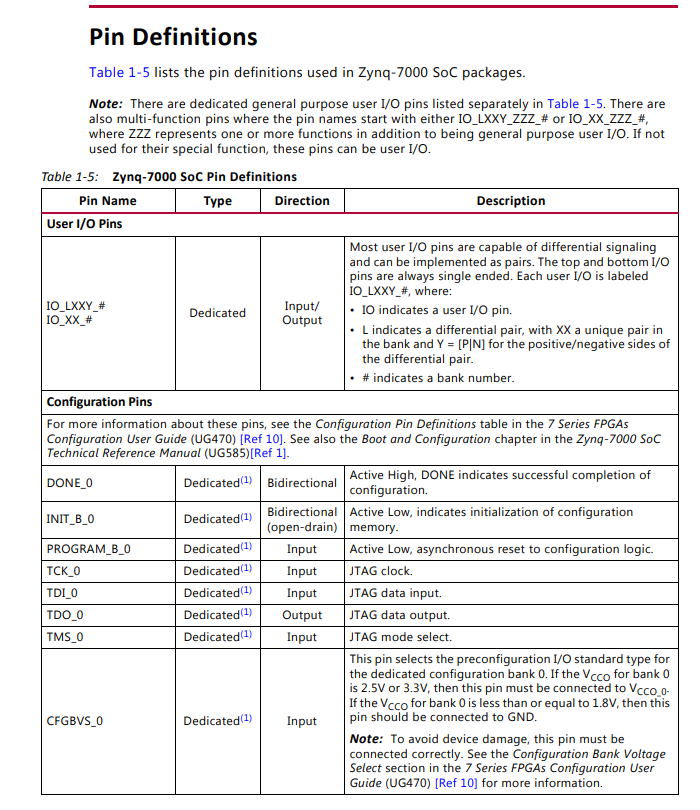
La cuarta columna se corresponde con el banco de pines al que pertenecen los pines( esta división por bancos se puede utilizar para crear el símbolo del componente)

La octava columna se corresponde con el tipo de pin, CONFIG para configuración, HR para configurar por hardware, MIO para pines de tipo MIO, DDR pines para la memoria Ram y NA para alimentaciones y masas.

\*el número del pin se corresponde con la numeración de los encapsulados en forma de matriz cuadrada, las columnas están ordenadas por números, y las filas por letras. Por ejemplo, el formato CLG25 tiene 5 filas y 5 columnas, por lo que la primera fila empieza por el pin A1 y acaba por el pin A5, la siguiente fila por el B1 – B5, y así hasta la última fila, E1-E5.

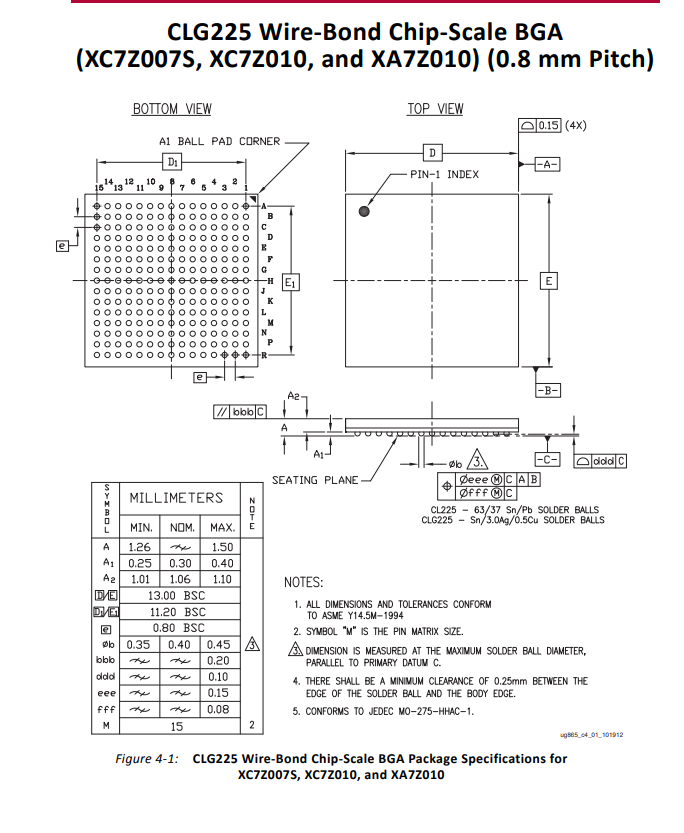
## Definición de los pines.

De la página 12 en adelante del documento figura la definición de los pines.



# Huellas de los encapsulados

Al final del documento figuran las estructuras de los pines en chip y las huellas



\*\*Estas huellas con del chip que se va a soldar, NO las huellas que se tienen que hacer en la PCB, estas últimas son las simétricas con el eje vertical. La huella que se tiene que fabricar en la PCB tiene que tener el pin A1 en la parte superior izquierda.

# Ejemplo

## Introducción

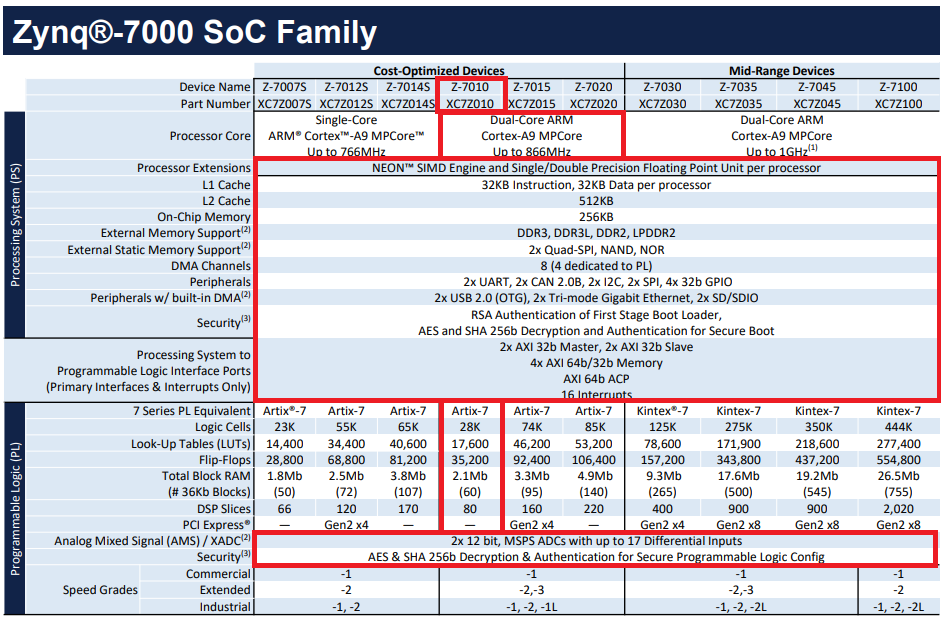
Debido a que explicar cómo construir un SoC es complejo ahora se va a explicar con un ejemplo.

Se va a construir una Zynq-eame 0 que está basada en un XC7Z010-CLG400.

Zynq-eame es una placa creada para Open Hardware por el blog soceame.wordpress.com utilizando KiCad(software Open Source)

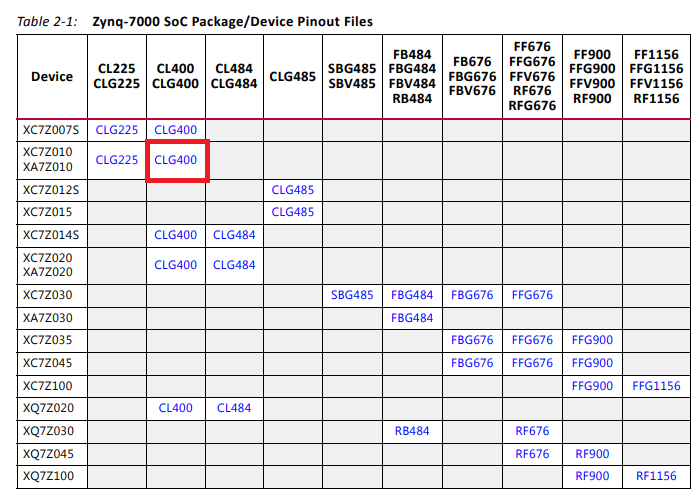
## Documentación

Un XC7Z010 consta de las características que nos da la tabla.



## Pinout

Para el Pinout se recurre a la tabla



Que nos lleva a la [página](https://www.xilinx.com/support/packagefiles/z7packages/xc7z010clg400pkg.txt) con los pines se parados por bancos. A partir de esta separación por pines se pueden hacer los símbolos en el programa de diseño de PCBs.

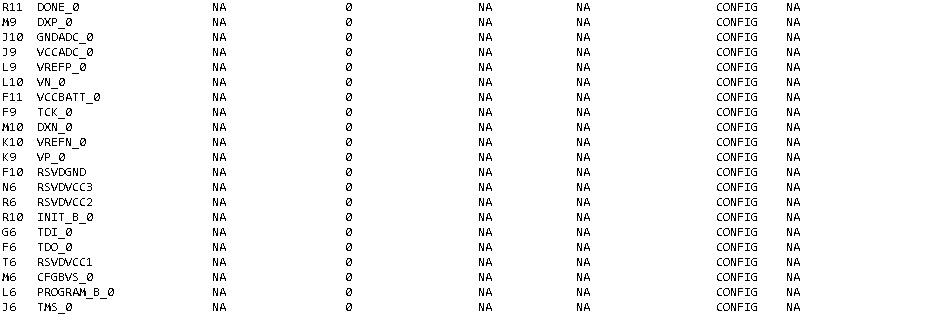
## Símbolo.

Para hacer los símbolos se va a recurrir a la separación por bancos de la que se ha hablado anteriormente.

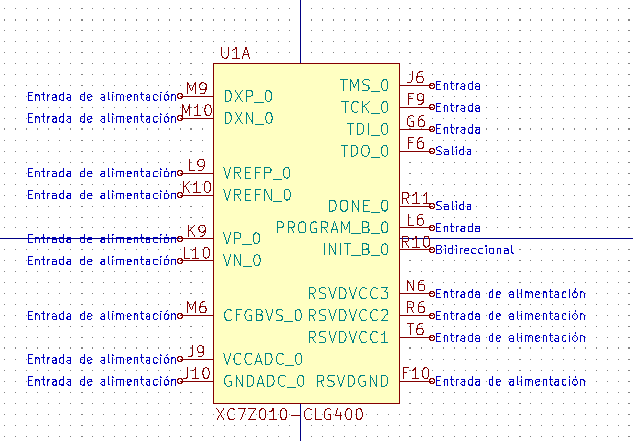
* **Bank 0**

Este banco está reservado para las referencias, por ejemplo, la tensión del ADC o la referencia de temperatura

* + Pinout



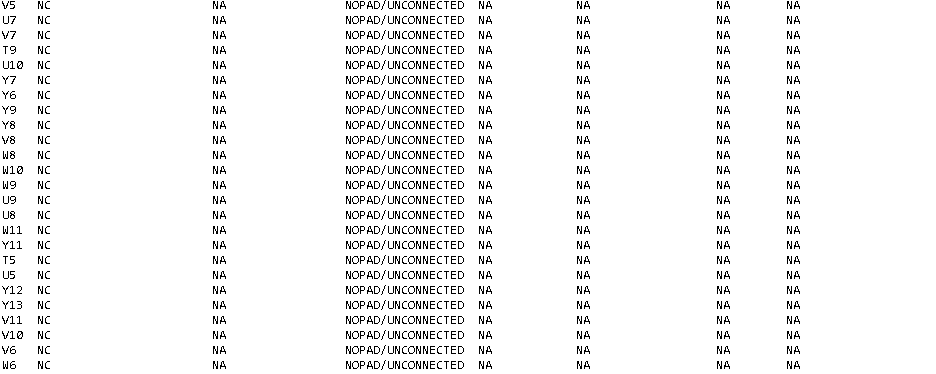
* + Símbolo



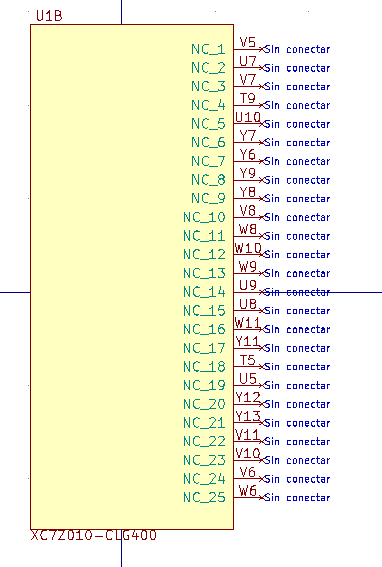
* **Unconnected**

Estos pines en el XC7Z010 no están conectados pero en otros formatos sí que lo pueden estar

* + Pinout



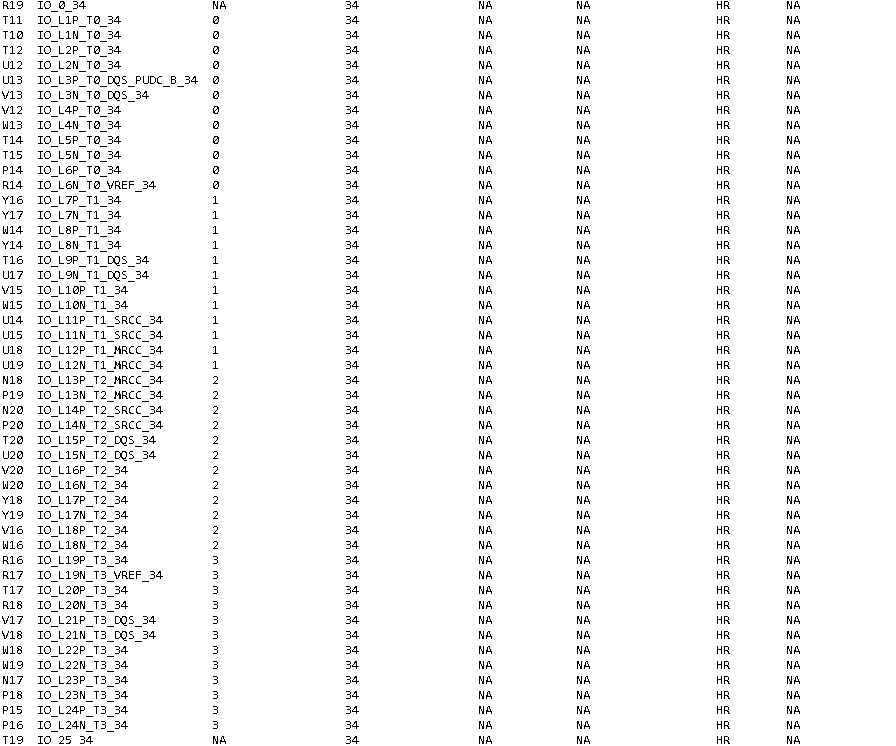
* + Símbolo



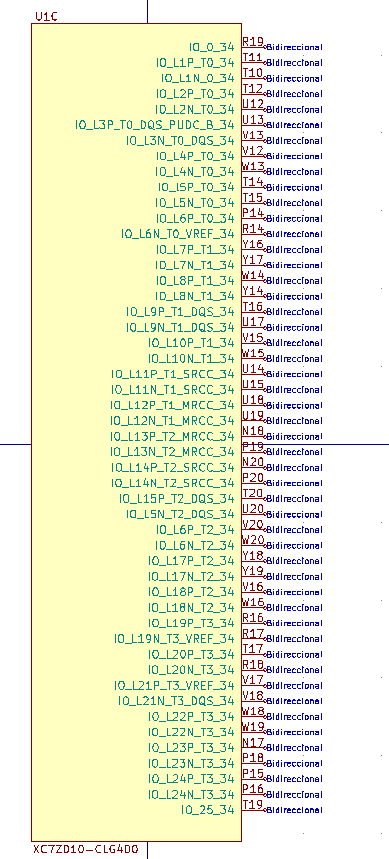
* **Bank 34**

Estos pines son configurables como entrada y salida por hardware. Entre los pines de este banco hay pines que pueden tener otro tipo de configuración.

* + Pinout



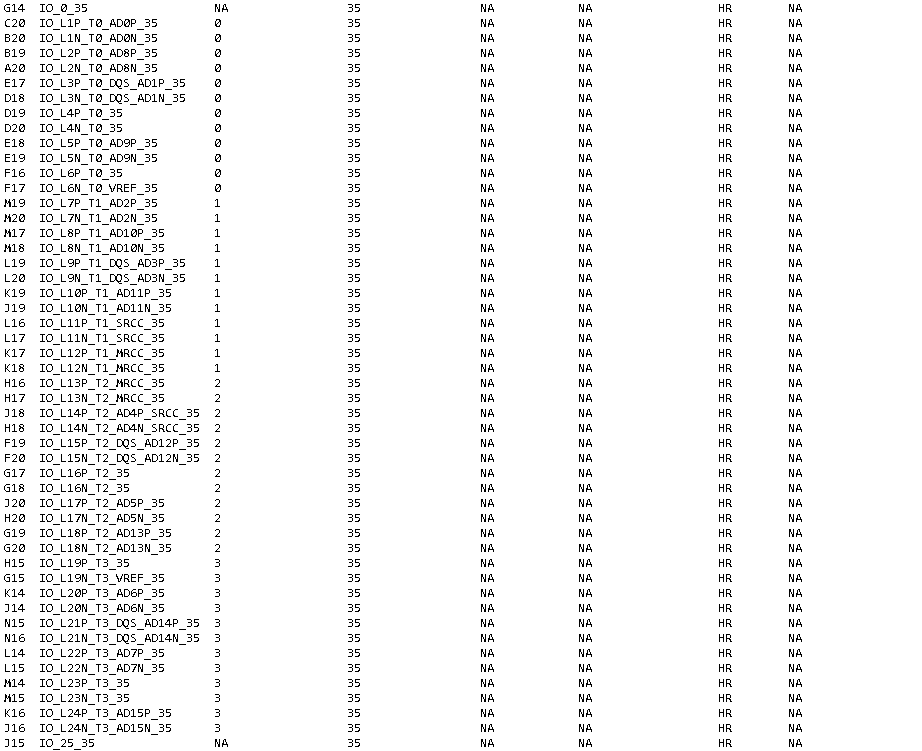
* + Símbolo



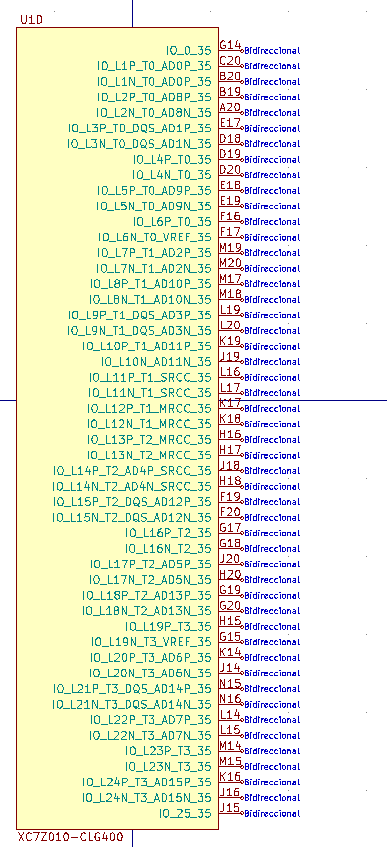
* **Bank 35**

Estos pines son configurables como entrada y salida por hardware. Entre los pines de este banco hay pines que pueden tener otro tipo de configuración.

* + Pinout



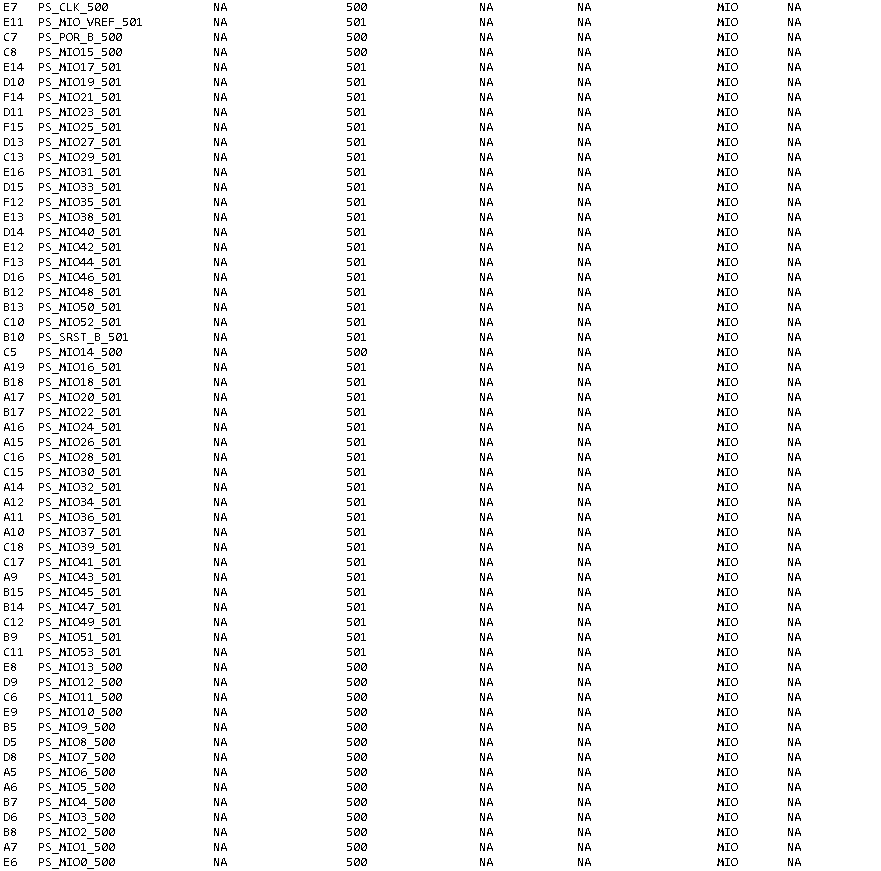
* + Símbolo



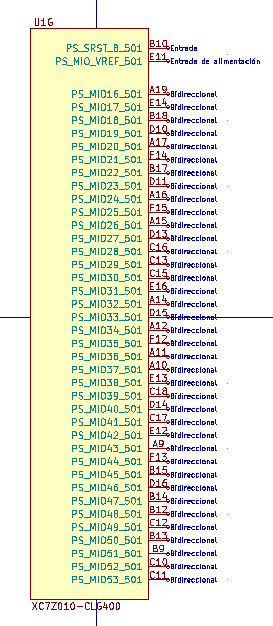
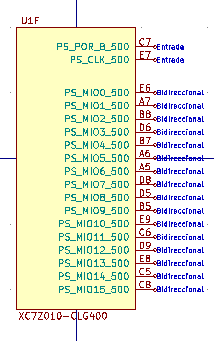
* **Bank 500 & 501**

Estos bancos están destinados para los pines de tipo MIO (Multiple IO)

* + Pinout



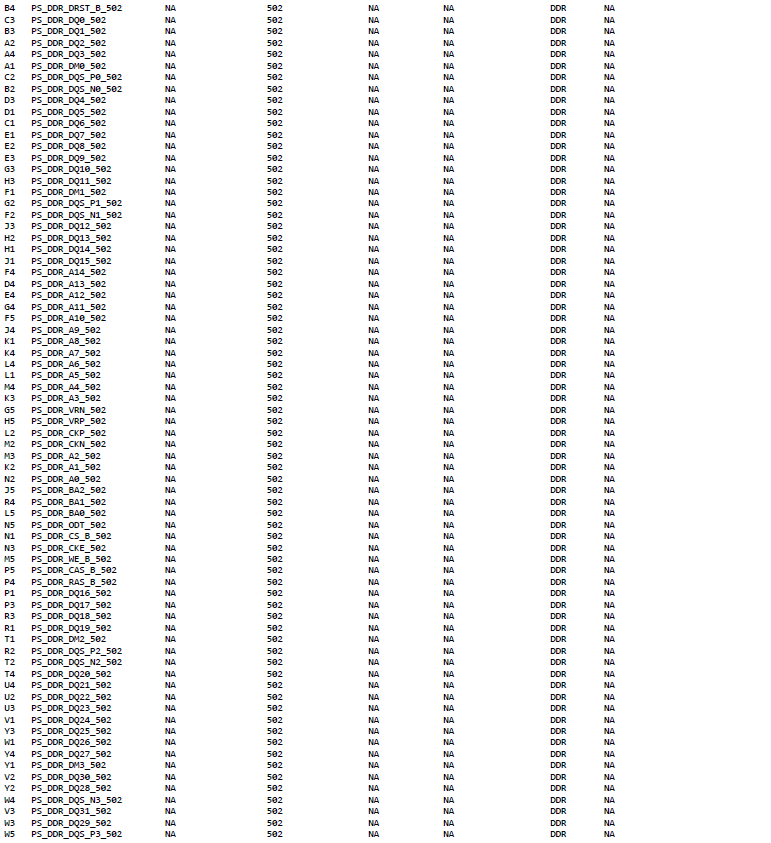
* + Símbolo



* **Bank 502**

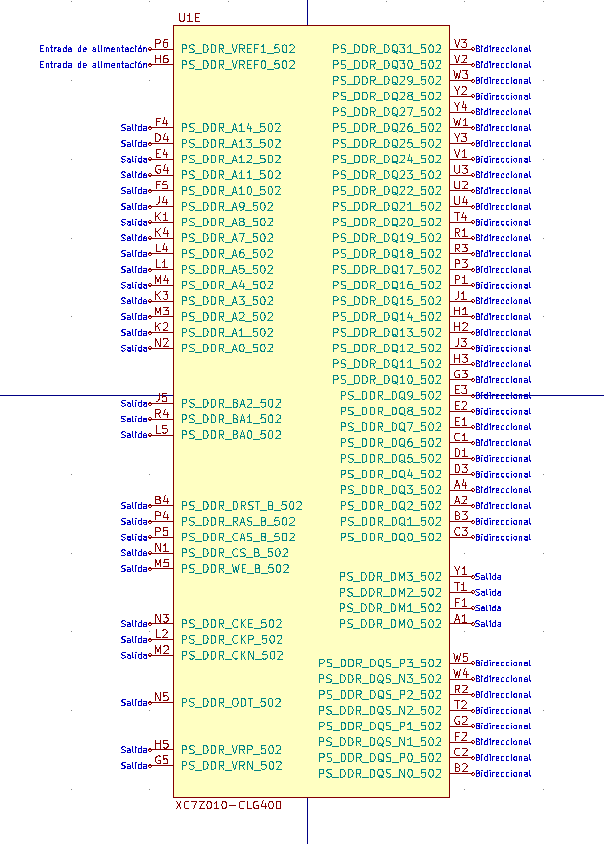
Este banco está destinado únicamente a la memoria Ram, 15 bits de Address y 32 bits de Datos

* + Pinout

👆estos dos últimos pines están al final del documento

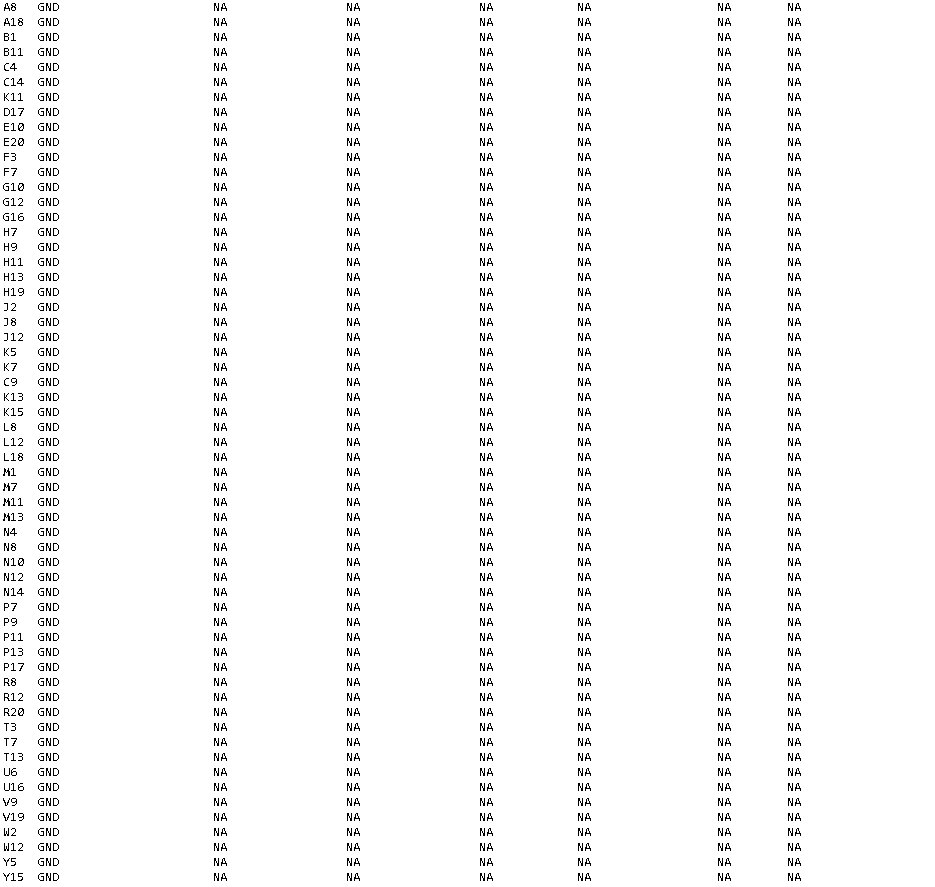
* + Símbolo



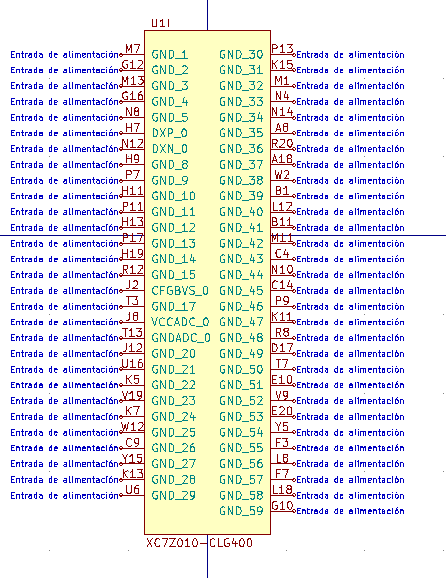
* **Bank GND**

Este banco está destinado únicamente a masas

* + Pinout



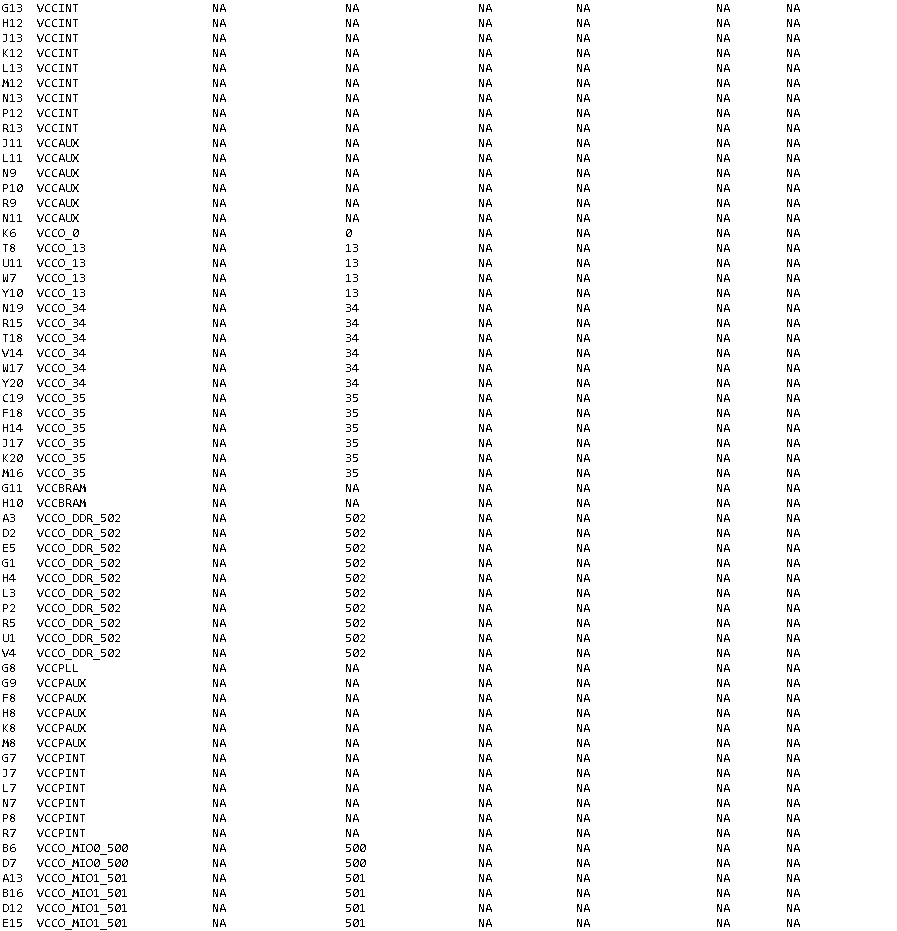
* + Símbolo



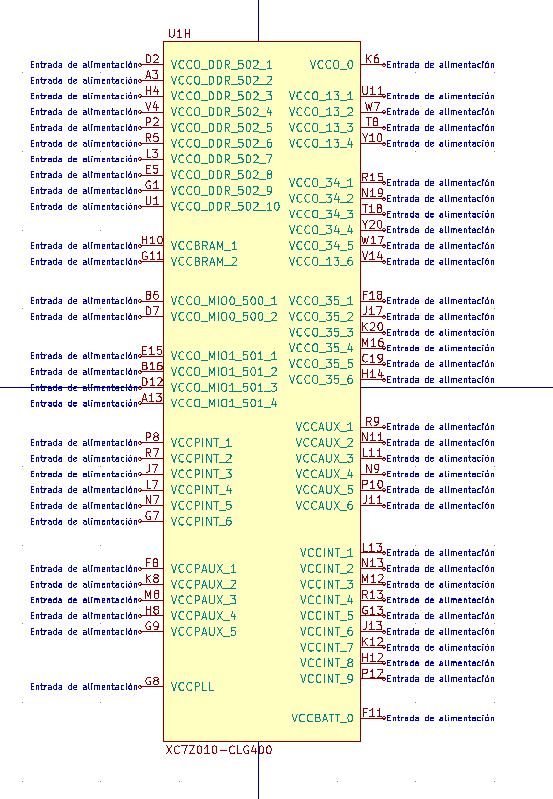
* **Bank VCC**

Este banco está destinado únicamente a las diferentes alimentaciones del sistema.

* + Pinout



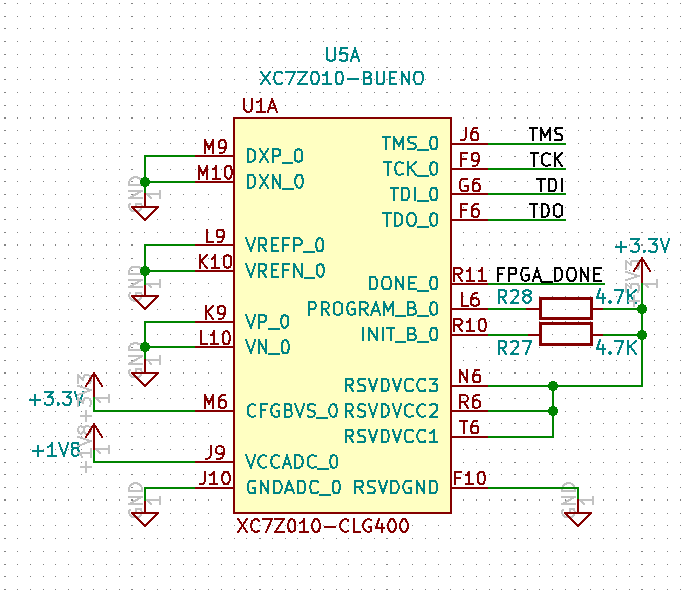
* + Símbolo



# Conexión

Para la conexión de va a desglosar para cada pin cómo debería estar conectado.

* **Bank 0**



DXP\_0 y DXN\_0 a masa porque no se va a conectar un diodo para el sensado de la temperatura.

VREFP\_0, VREFN\_0, VP\_0 y VN\_0 a masa porque no se va a conectar un ADC.

CFBVS\_0 a 3.3V para preconfigurar los pines

VCCADC\_0 es la tensión de alimentación del XADC, tiene que estar a 1.8V

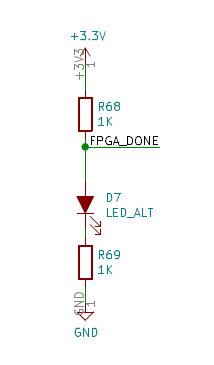
GNDADC\_0 es la masa del XADC, se conecta a la masa general porque no se va a utilizar el ADC.

RSVDVCC[3:1] tienen que ir a la tensión Vcco\_0, en el caso de la Zynq-eame 0 a 3V3.

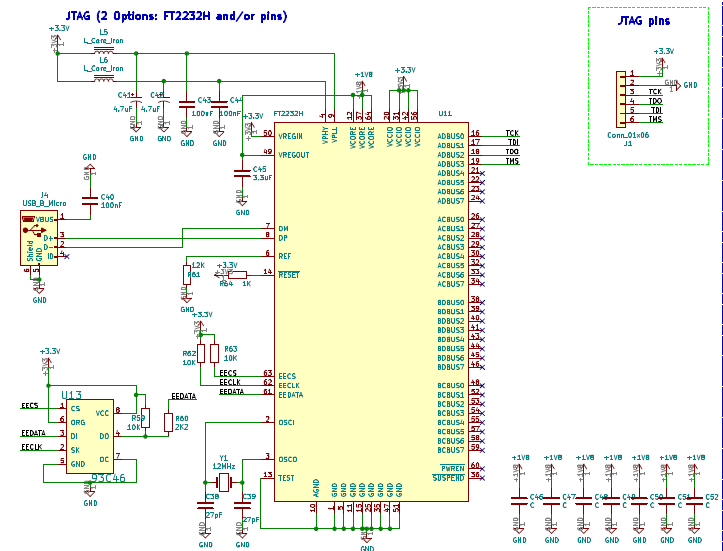
INIT\_B\_0 indica la inicialización de la configuración de la memoria, se activa a nivel bajo.

PROGRAM\_B\_0 reset asíncrono activo a nivel bajo. Se conecta a 3V3 porque no se resetea la placa.

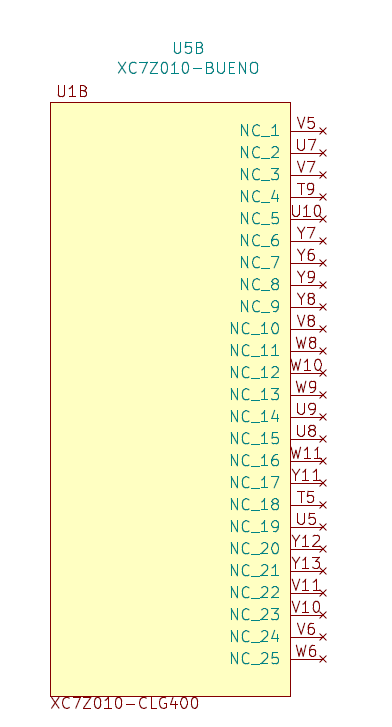
DONE\_0 indica que la configuración se ha completado, activo a nivel alto. En la Zynq-eame se conecta a un diodo LED para indicar si se ha cargado la placa.



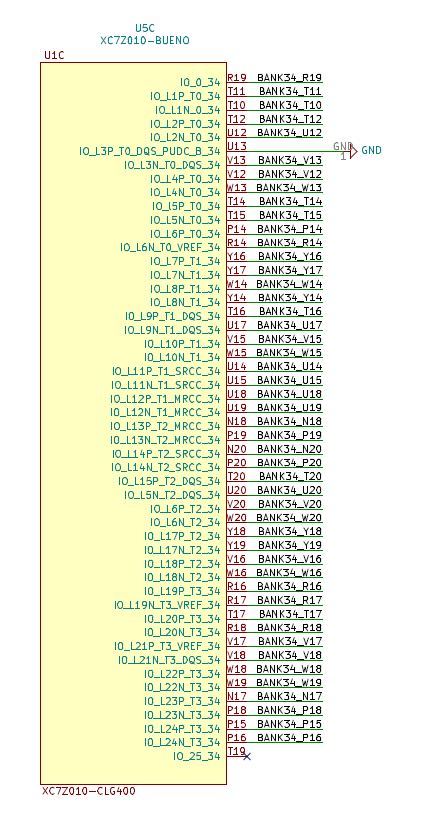
TMS\_0, TCK\_0, TDI\_0 y TDO\_0 son los pines del JTAG, estos pines pueden ir a un cable depurador o a un chip que permita cargar el programa por USB. En la Zynq-eame se utiliza el FT2232H, se necesita de una EEPROM y un oscilador de 12MHz pero elimina la necesidad de un cable depurador.



* **Bank Unconnected**



* **Bank 34**



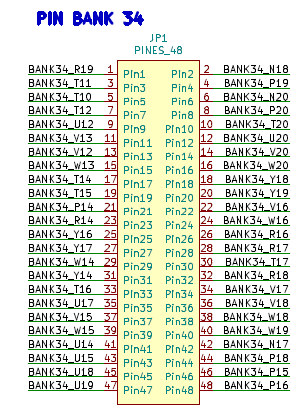
Todos lo pines IO son pines que se pueden configurar por Hardware y algunos de ellos pueden adquirir otras configuraciones.

Los pines IO\_Ln[P/N] son pines diferenciales utilizados por el XADC.

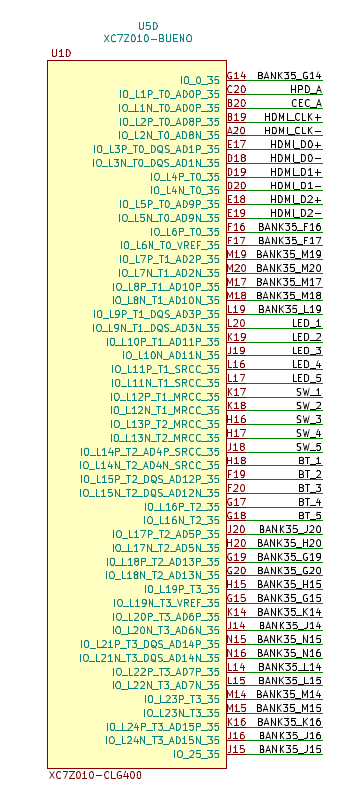
El pin PUBC\_B es un pin que activa las resistencias de pull-up del sistema después del arranque y durante la configuración. Activa a nivel bajo, tiene que ir directamente a GND o a VCCO\_34

SRCC y MRCC estos pines permiten que un reloj funcione en varias regiones(MRCC) o en solo una (SRCC). Estos pines tienen interés en las FPGAs para conectar el oscilador de la frecuencia a ellas. La Zynq-eame no lleva conectado un reloj a un pin MRCC por eso no puede ser utilizada como una FPGA sin activar el reloj interno del Zynq.

En la Zynq-eame 0 todo el banco va a pines de salida, menos el pin PUBC\_B que va a masa y el pin T19 que no se conecta porque no tiene un pin de salida.



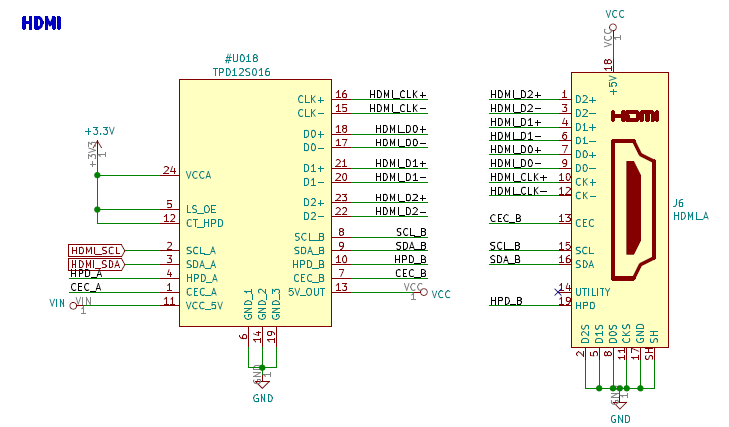
* Bank 35



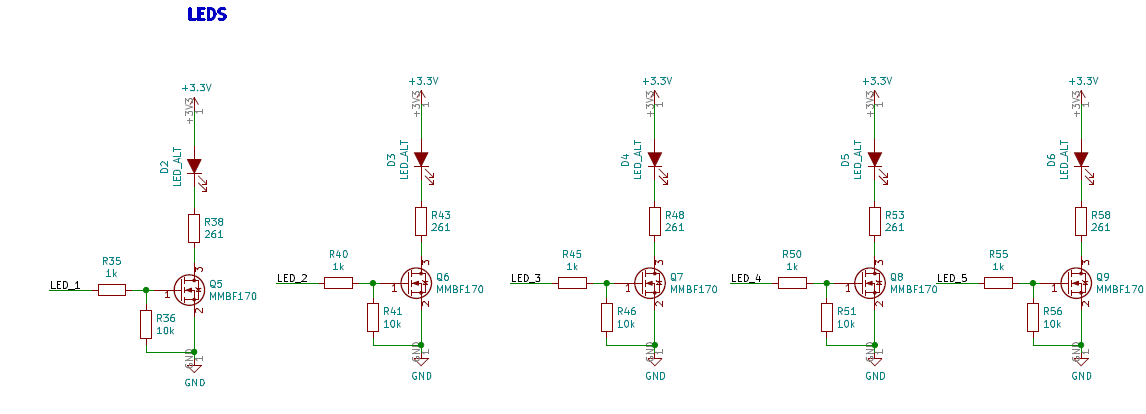
Pasa igual que en el banco 34, hay pines con múltiples funciones.

Este banco se utiliza entero.

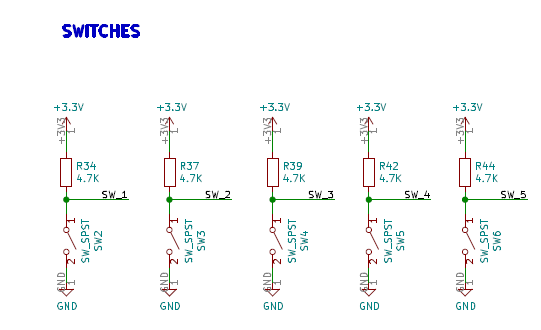
Se conectan los pines HDMI a los pines elegidos, éstos van a un TPD12S016 que va conectado al HDMI hembra



Les pines que se van a utilizar para los leds llevan un transistor para funcionar.

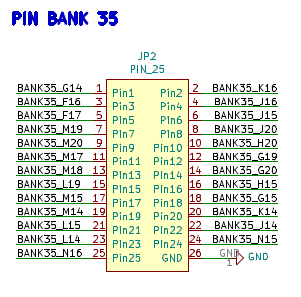


Los pines que van a los switches son activos a nivel bajo.

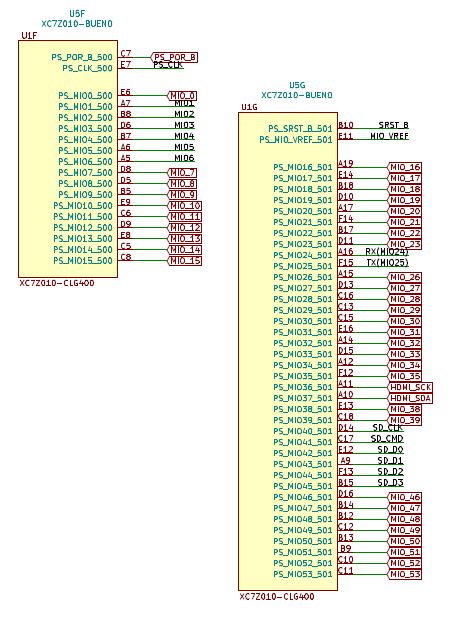


Los pines que van a los botones son activos a nivel alto.

Y el resto de los pines va a pines de salida.

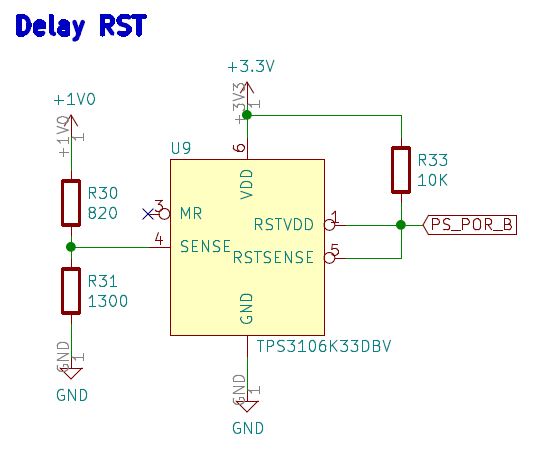


* Bank 500 & 501

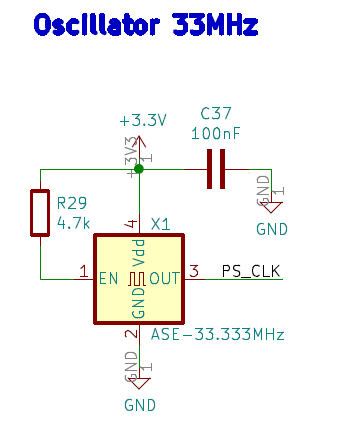


Banco 500:

PS\_POR\_B utiliza para mantener un reset al arranque del sistema, el sistema que se utiliza es un supervisor que mantiene el sistema reseteaso. Para eso se utiliza el [dispositivo que propone Xilinx](https://www.xilinx.com/Attachment/AR65240_-_Example_PS_POR_B_Supervisor_Circuit.pdf), un TPS3106K33DBV, que mantiene el reset durante 130ms[según el [datasheet](https://www.ti.com/lit/ds/symlink/tps3106.pdf?ts=1610541218328)].

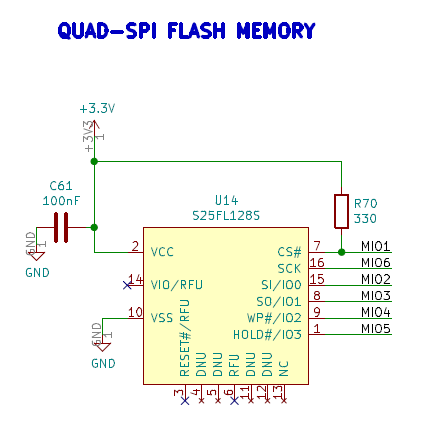


PS\_CLK este es el reloj del sistema y tiene que ir conectado a un oscilador entre 30 y 60MHz, en la Zynq-eame se conecta a 33.333MHz.



Los pines MIO se utilizan para pines de comunicación multiplexada.

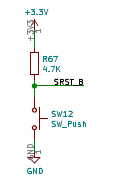
Por eso la multiplexación de los pines 1 al 6 es la de una QUAD-SPI FLASH MEMORY



El resto de pines va a pines de salida con el resto de pines del bank MIO 501

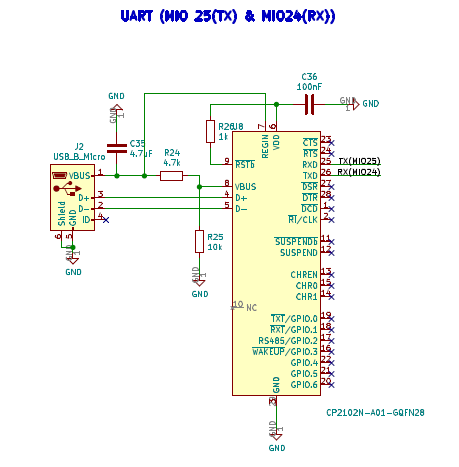
Bank 501:

PS\_SRST este pin fuerza el reseteo del sistema, activo a nivel bajo. En una Zynq-eame se conecta a un pulsador.

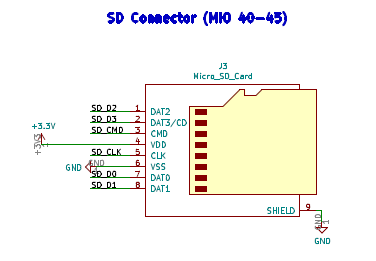


MIO\_VREF proporciona de una tensión de referencia para RGMII (Fast Ethernet) si no se usa se puede dejar al aire. Si se conecta una interfaz RGMII se tiene que usar un divisor de tensión de 1/2, por lo que si la interfaz es de 1.8V a la entrada tiene que haber 0.9V. La Zynq-eame 0 no tiene conexión a Fast Ethernet por lo que se deja al aire.

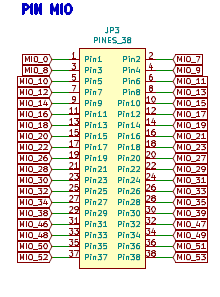
Lo pines mio 24 y 25 son pines MIO que tienen la posibilidad de configurar la UART. Estos pines son lo que se usan para la comunicación USB.



Los pines 40-45 son pines que permiten la conexión de una memoria SD.



El resto de los pines va conectado a pines externos

.

La referencia para la conexión de los pines MIO es la siguiente tabla.

