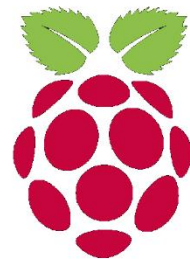
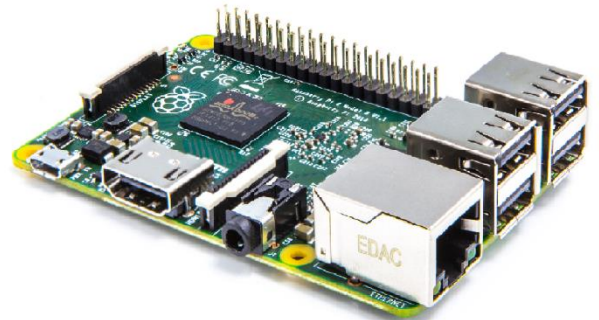
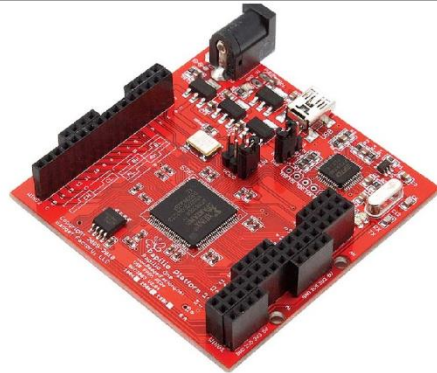
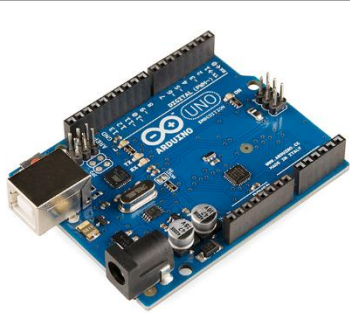


CURSO 2016-2017

MEMORIA DE PRÁCTICAS LABORATORIO DE DESARROLLO DE HARDWARE



Autor: Carlos Crespo Jiménez

NIF: 28785141-C

Grado de Ingeniería de Computadores
Memoria de Prácticas de Laboratorio de
Desarrollo de Hardware.

Índice

Plataforma Papilio/ZPUino	3
1.1 Objetivos	3
1.2 Introducción	3
1.3 Instalación del entorno de desarrollo Papilio DesignLab (para nuestra FPGA)	6
1.4 Diferencias entre IDE Arduino vs Papilio DesignLab	6
1.5 Encendido y apagado de un led mediante inversor	6
1.6 Encendido de un led a través del puerto serie	8
1.7 Placa Papilio como analizador lógico	9
1.8 Añadiendo un periférico a nuestro SoC	12
1.9 Conclusión	15

1.1 Objetivos

Los objetivos mínimos en la plataforma Papilio/ZPUino son:

- Conocer la plataforma Papilio.
- Instalar y configurar el entorno de trabajo de papilio: Design Lab.
- Conocer que se puede hacer desde Design Lab sobre las placas Papilios:
 - o Diseñar circuitos a nivel de captura de esquemáticos e implementarlo en la FPGA.
 - o Cargar el SoC ZPUino.
 - o Desarrollar algunos de los Sketches para ZPUino.
 - o Modificar el SoC ZPUino añadiendo algún nuevo periférico y desarrollar algún Sketch que lo utilice.
 - o Configurar la placa Papilio para que funcione como un analizador lógico.

1.2 Introducción



¿Qué es ZPUino?

Es un SoC implementado en VHDL basado en el microprocesador de 32 bits de ZPU de Zylín. Tanto el diseño del ZPUino como las especificaciones del microprocesador son abiertas (libres). Al ser diseño soft-core, se pueden añadir nuevos periféricos al SoC según las necesidades.

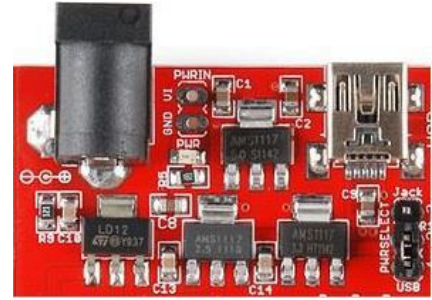
ZPUino es un entorno equivalente a Arduino, ya que se ha adaptado el IDE para ser compatible con él. Como veremos el entorno también nos permitirá modificar el SoC a nivel hardware para añadir nuevos periféricos. El entorno de desarrollo para Zpuino que usaremos es Design Lab. En esta asignatura, nuestra placa será la Papilio one 500k.



¿Cuáles son las características de la placa Papilio one 500k?

La Papilio One es una placa de desarrollo de código abierto basada en la FPGA Xilinx Spartan 3E FPGA, proporcionando una gran cantidad de lógica digital. Cuenta con 48 líneas de entrada/salida, de doble canal USB, programador JTAG integrado, 4 fuentes de alimentación y un conector de alimentación. La placa Papilio permite elaborar circuitos sin necesidad de invertir mucho tiempo y energía en aprender Verilog.

Tiene cuatro carriles independientes de energía a 5V, 3.3V, 2.5V y 1.2V. La energía se suministra a través de la entrada USB. El voltaje de entrada recomendado es entre 6.5 y 10V. Además del USB, la placa puede ser alimentada desde una fuente de alimentación externa o una batería.



Tiene dos canales de conexión USB para JTAG (que funcionan a 2.5V) y comunicaciones serie UART implementadas con FT2232D (los JTAG conectados a él van a 3.3V). También tiene una memoria EEPROM para almacenar la configuración del chip USB FT2232.



¿Cuáles son las características de la FPGA Spartan 3E?

- ✓ Tiene 20 bloques de memoria BRAM de 18Kbit cada uno, tiene un máximo de 360Kbits de memoria SRAM pero sólo 320Kbits son útiles.
- ✓ Tiene un oscilador de 32MHz que puede ser usado por el DCM de Xilinx para generar cualquier velocidad de reloj que se necesite.
- ✓ Tiene una huella VTQFP-100 que soporta Xilinx XC3S100E, XC3S250E y XC3S500E.
- ✓ La entrada y la salida se puede configurar para soportar 1.2V, 2.5V o 3.3V.



Además, usa un flash SPI del tipo 4Mbit SST SST25VF040B que proporciona mucho espacio para un archivo de bits de arranque y los datos del usuario.

El bloque de entrada/salida proporciona una interfaz programable entre los pines y la lógica interna Spartan-3E. Todos sus pines estarán en alta impedancia durante la configuración, durante la carga del programa.

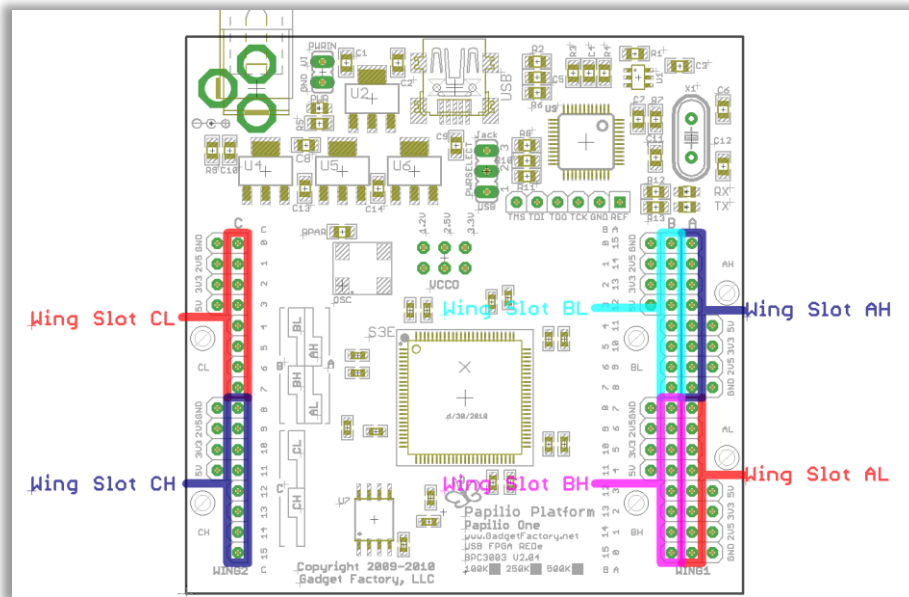


¿Qué son las WINGS?

Son placas de expansión adaptadas al conector de expansión de Papilio. Estos conectores (wing slots), en nuestra Papilio ONE se diferencian por distintas letras que en nuestra práctica las usaremos para conectarnos con los diversos componentes. Además, en esta placa podremos nombrar a estos pines como lo hacía Arduino, mediante números, que vienen representados en la siguiente imagen.

Las 48 líneas bidireccionales de entrada/salida se pueden dividir en:

- ✓ 1 Ala de 32bit o
- ✓ 3 Alas de 16bit o
- ✓ 6 Alas de 8bit



🤔 ¿Qué es una FPGA?

Una FPGA (del inglés Field Programmable Gate Array) es un dispositivo programable que contiene bloques de lógica cuya interconexión y funcionalidad puede ser configurada 'in situ' mediante un lenguaje de descripción especializado. La lógica programable puede reproducir desde funciones tan sencillas como las llevadas a cabo por una puerta lógica o un sistema combinacional hasta complejos sistemas en un chip.



Las FPGAs se utilizan en aplicaciones similares a los ASICs, sin embargo son más lentas, tienen un mayor consumo de potencia y no pueden abarcar sistemas tan complejos como ellos. A pesar de esto, las FPGAs tienen las ventajas de ser reprogramables (lo que añade una enorme flexibilidad al flujo de diseño), sus costes de desarrollo y adquisición son mucho menores para pequeñas cantidades de dispositivos y el tiempo de desarrollo es también menor.

Ciertos fabricantes cuentan con FPGAs que sólo se pueden programar una vez, por lo que sus ventajas e inconvenientes se encuentran a medio camino entre los ASICs y las FPGAs reprogramables.

1.3 Instalación del entorno de desarrollo Papilio DesignLab (para nuestra FPGA)

Para comenzar en la plataforma nos introduciremos en el desarrollo y programación de FPGA's con nuestra placa Papilio One 500k (basada en la Spartan3E500 de Xilinx). El sistema de desarrollo es el Papilio DesignLab 1.0.7 (<http://10.1.15.78/~bellido>). En el laboratorio utilizaremos equipos informáticos con Linux 64 bits.

Pasos a seguir para realizar la instalación del entorno de desarrollo Papilio DesignLab 1.0.7:

- 1) Descargamos de la web del profesor el Papilio DesignLab 1.0.7.
- 2) Descomprimos el archivo que nos hemos descargamos.
- 3) Posteriormente nos vamos a la consola/terminal de Linux (Ubuntu) y ejecutamos el archivo con permiso de superusuario: `sudo ./Ubuntu-setup.sh`
- 4) Si no tuviéramos instalado el "JRE" de java tendremos que instalarlo también, escribiendo por consola: `sudo apt-get install default-jre`.
- 5) Para comenzar a utilizar Papilio DesignLab 1.0.7 tendremos que ejecutarlo en la consola de la siguiente forma: `sudo ./DesignLab`

1.4 Diferencias entre IDE Arduino vs Papilio DesignLab

Los dos entornos son muy parecidos, excepto que en Papilio DesignLab podemos apreciar nuevas opciones debajo de la barra de herramientas ya que para estas placas tenemos la posibilidad de programar su funcionamiento físico mediante el ISE de Xilinx.

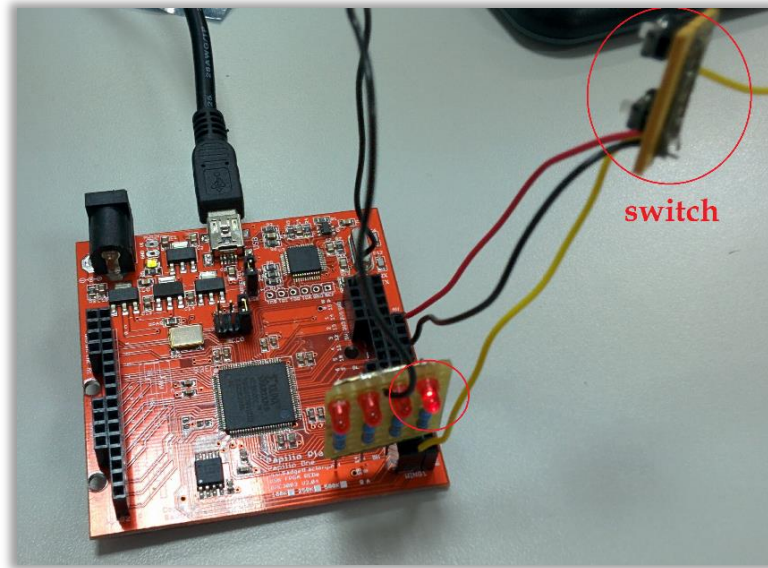
1.5 Encendido y apagado de un led mediante inversor

Para realizar esta práctica antes de todo debemos de programar nuestra propia FPGA en la placa. Asignaremos los pines correspondientes para el correcto funcionamiento. Para ello hemos seguido los pasos descritos en este tutorial: <http://gadgetfactory.net/learn/2015/05/03/designlab-make-a-simple-fpga-circuit-2/>

Para cargar bien el fichero bitfile hay que hacer un cambio en el nombre del fichero que genera ISE.

- ✓ Entraremos en la carpeta <proyecto>/circuit/500K/.
- ✓ Borraremos el archivo `papilio_one_500k.bit`.
- ✓ Renombraremos `Papilio_One_500K.bit` a `papilio_one_500k.bit`.

Pulsaremos en “Load Circuit” (una vez tengamos el código terminado) para subirlo a la placa. Después de haber seguido al pie de la letra el anterior tutorial, procederemos a cargar nuestro sketch. Al finalizar la práctica podremos controlar el encendido y el apagado del led a través de switch.



✚ Código en ZPUino

```
int led = 0;

void setup() {
  // put your setup code here, to run once:

  pinMode(led, OUTPUT);
}

void loop() {
  digitalWrite(led, HIGH); // turn the LED on (HIGH is the
  voltage level)
  delay(1000);             // wait for a second
  digitalWrite(led, LOW);  // turn the LED off by making the
  voltage LOW
  delay(1000);             // wait for a second
}
```


1.6 Encendido de un led a través del puerto serie

En este ejercicio vamos a controlar el encendido y apagado de un led usando el puerto serie. Para comprobar que funciona correctamente hemos colocado un LED en el pin 0 y un switch al pin que controla dicho LED (pin 1). Para que el LED quede apagado o encendido en función de la posición del switch suprimiremos el for para evitar mostrar todos los LEDs:

```
//This section blinks the LED's and keeps them solid if a button is pressed.
delay(200); // wait for a second
ledState = !ledState;
//for (int thisPin = 0; thisPin < buttonCount; thisPin++) {
// read the state of the pushbutton value:
buttonState = digitalRead(buttonPins[0]);

// check if the pushbutton is pressed.
// if it is, the buttonState is HIGH:
if (buttonState == HIGH) {
// turn LED on:
digitalWrite(ledPins[thisPin], HIGH);
Serial.println("LED: Encendido");
}
else {
// toggle LED:
digitalWrite(ledPins[thisPin], LOW);
Serial.println("LED: Apagado");
}
delay(3000);
//}
```

A continuación, tendremos que hacer la última modificación para controlar el LED por el puerto serie, es decir, desde la línea de comandos y evitando el switch.

🔧 Código en ZPUino

```
int led = 0;

void setup(){
  pinMode(led,OUTPUT);
  Serial.begin(9600);
}

void loop(){
  if(Serial.available()){
    char c = Serial.read();
    if(c == 'H'){
      digitalWrite(led,HIGH);
    }else if(c == 'L'){
      digitalWrite(led,LOW );
    }
  }
}
```


Código Python

```
import serial

zpuino = serial.Serial('/dev/ttyUSB2', 9600)

print("Starting!")

while True:
    comando = raw_input('Introduce un comando: ') #Input
    zpuino.write(comando) #Mandar un comando hacia ZPUino
    if comando == 'H':
        print('LED ENCENDIDO')
    elif comando == 'L':
        print('LED APAGADO')

zpuino.close() #Finalizamos la comunicacion
```

1.7 Placa Papilio como analizador lógico

¿Qué es un analizador lógico?

Un analizador lógico es un instrumento de medida que captura los datos de un circuito digital y los muestra para su posterior análisis, de modo similar a como lo hace un osciloscopio, pero a diferencia de este, es capaz de visualizar las señales de múltiples canales. Además de permitir visualizar los datos para así verificar el correcto funcionamiento del sistema digital, puede medir tiempos entre cambios de nivel, número de estados lógicos, etc. La forma de capturar datos desde un analizador lógico es conectando una punta lógica apropiada en el bus de datos a medir.

Desarrollo

Una de las aplicaciones que tienen las placas papilios es poder funcionar como analizadores lógicos. Siguiendo el siguiente tutorial: (<http://gadgetfactory.net/learn/2015/07/30/designlab-using-papilio-as-stand-alone-logic-analyzer/>) podremos comprobar qué tal funciona nuestra placa Papilio como analizador lógico. Anotación: Hay que elegir bien la placa Papilio One 500k, no la que sale en el tutorial.

Existe un problema con el cliente software del analizador lógico con el java y no se ejecuta correctamente en estos PCs. Para solucionarlo tenemos que seguir los siguientes pasos:

- Debemos descargarnos el siguiente programa: <https://10.1.15.75/~bellido/ols-0.9.7.2-full.tar.gz>
- Extraer y ejecutar el script:
 - \$ cd <carpeta de OLS>
 - \$ sudo ./run.sh

Para poder utilizar nuestra placa como analizador lógico tendremos que cargar un Sketch básico llamado “Fade” que contiene el siguiente código:

```
/*
  Fade

  This example shows how to fade an LED on pin 9
  using the analogWrite() function.

  The analogWrite() function uses PWM, so if
  you want to change the pin you're using, be
  sure to use another PWM capable pin. On most
  Arduino, the PWM pins are identified with
  a "~" sign, like ~3, ~5, ~6, ~9, ~10 and ~11.

  This example code is in the public domain.
  */

int led = 9;          // the PWM pin the LED is attached to
int brightness = 0;    // how bright the LED is
int fadeAmount = 5;    // how many points to fade the LED by

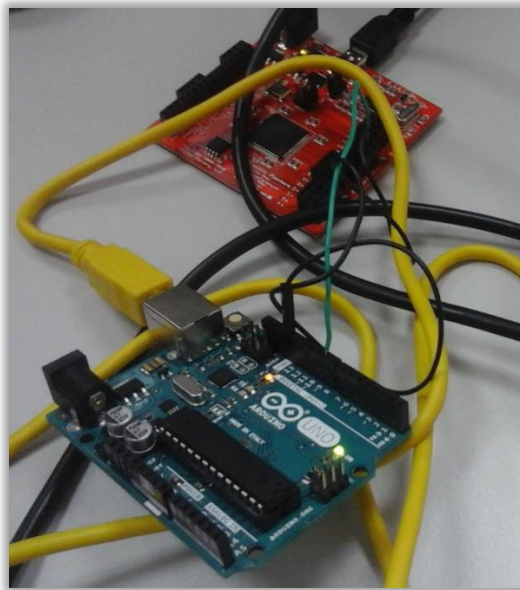
// the setup routine runs once when you press reset:
void setup() {
  // declare pin 9 to be an output:
  pinMode(led, OUTPUT);
}

// the loop routine runs over and over again forever:
void loop() {
  // set the brightness of pin 9:
  analogWrite(led, brightness);

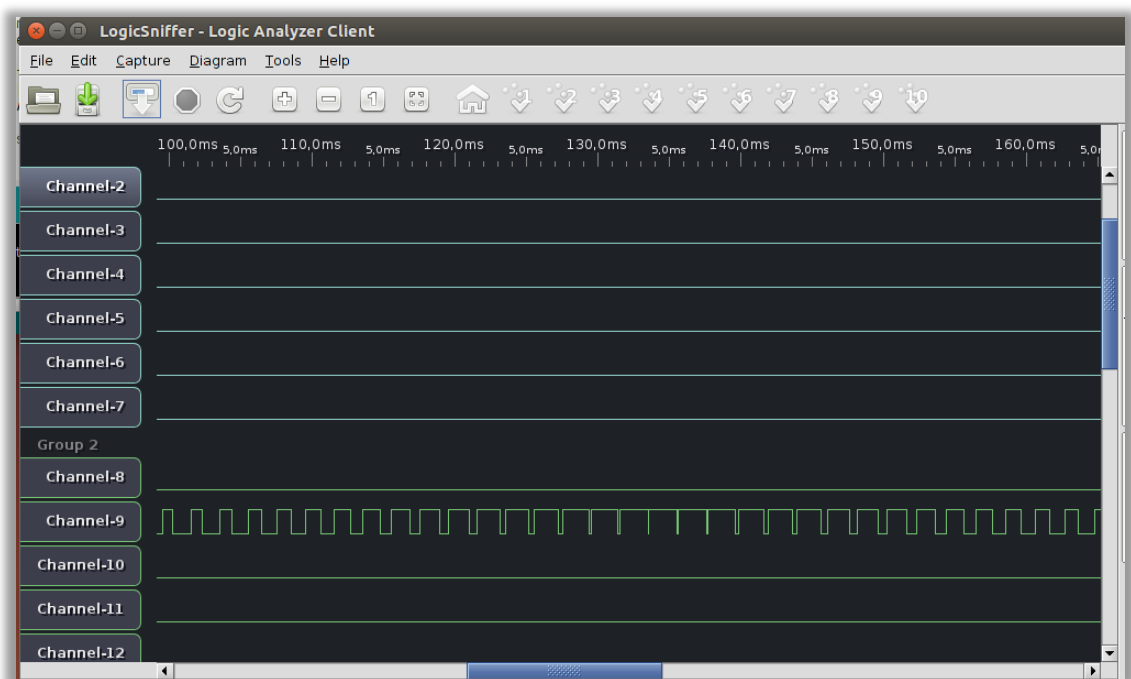
  // change the brightness for next time through the loop:
  brightness = brightness + fadeAmount;

  // reverse the direction of the fading at the ends of the fade:
  if (brightness <= 0 || brightness >= 255) {
    fadeAmount = -fadeAmount;
  }
  // wait for 30 milliseconds to see the dimming effect
  delay(1); //Modificar a 1 ms
}
```

Conectaremos nuestra placa Papilio a nuestra placa Arduino. En mi caso, la conexión se hará sobre el pin 9 de la placa de Arduino. Por el canal 9, saldrá toda la información.



🔌 Resultado final analizador lógico



1.8 Añadiendo un periférico a nuestro SoC



¿Qué son las UART?

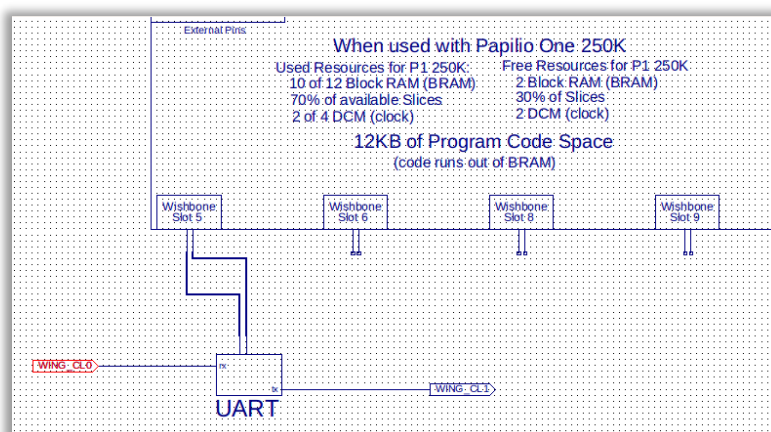
Es el dispositivo que controla los puertos y dispositivos serie. Se encuentra integrado en la placa base o en la tarjeta adaptadora del dispositivo. En este caso la podemos crear en nuestra FPGA. Las funciones principales de chip UART son: manejar las interrupciones de los dispositivos conectados al puerto serie y convertir los datos en formato paralelo, transmitidos al bus de sistema, a datos en formato serie, para que puedan ser transmitidos a través de los puertos.



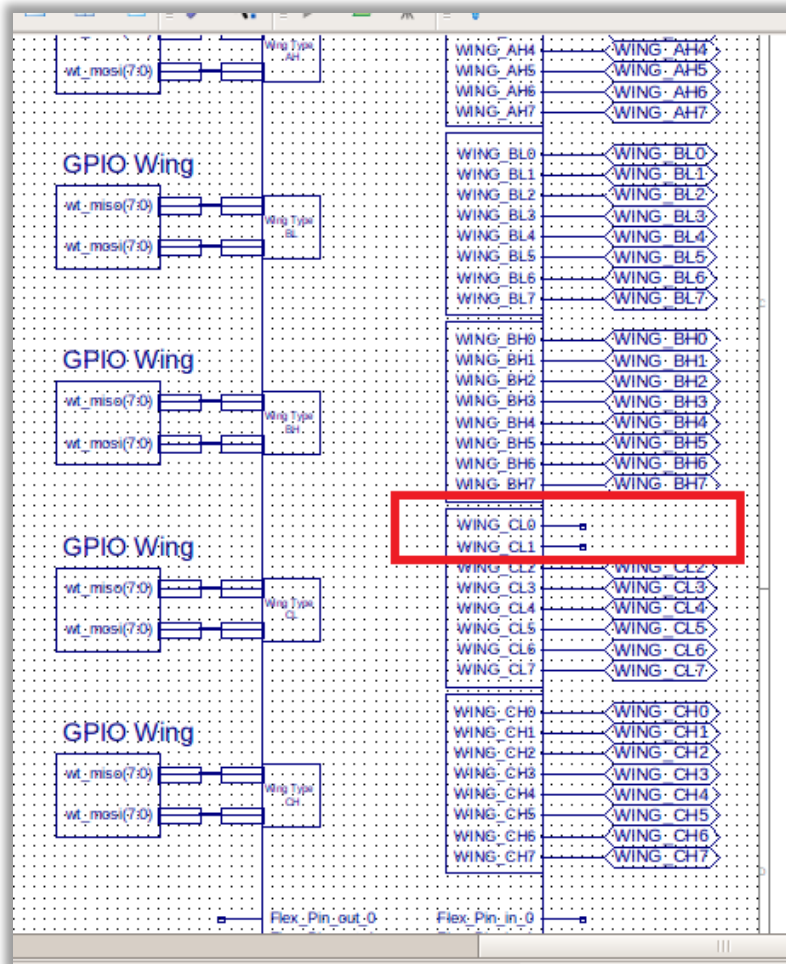
Desarrollo

A diferencia del ejercicio anterior, esta vez tendremos que usar el puerto serie del PC (pondremos una UART en lugar de un inversor). Para ello tendremos que configurar de nuevo el diseño de nuestra FPGA tal y como viene explicado en el siguiente tutorial: <http://gadgetfactory.net/learn/2015/05/15/designlab-make-a-custom-zpuino-system-on-chip-2/>

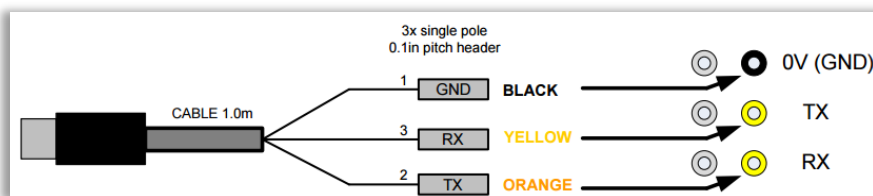
Cuando se abra el esquemático añadiremos una UART a Wishbone Slot 5 de la siguiente manera:



Borraremos los cables WING_CLO y WING_CL1. Una vez hayamos realizado estos pasos sintetizaremos en Papilio_Pro.sch de la ventana Desing y posteriormente para finalizar haremos clic en Generate Programming File.



Una vez tengamos la UART disponible en los pines WA0 y WA1 debemos conectarnos al PC con el cable TTL-RS232-USB (de 1 metro de longitud).



Anotación: conectaremos el cable serie atendiendo a las explicaciones mencionadas por el profesor, la salida Tx del cable debe de ir a la Rx de la placa y la Rx del cable a la Tx de la placa.

En un extremo tiene un conector USB que va al ordenador. En el otro extremo tiene tres pines:

- **Pin 1** (cable negro): irá conectado a tierra (GND).
- **Pin 2** (cable naranja): es la salida asíncrona de los datos, el transmisor (TXD), irá conectado al receptor de la placa (WING_ CL0).

- **Pin 3** (cable amarillo): es la entrada asíncrona de datos, el receptor (RXD), irá conectado al transmisor de la placa (WING_ CL1).
- El **LED** lo conectaremos al pin WING_Slot_AH 6 (Se puede poner en cualquier pin).

Para finalizar, modificaremos nuestro código ZPUino y añadiremos un nuevo fichero Python igual que hicimos en prácticas anteriores.

Código en ZPUino

```
HardwareSerial mySerial( WishboneSlot(5));

int led = 0;
int led1 = 6;

void setup(){
  pinMode (led, OUTPUT);
  pinMode (led1,OUTPUT);
  mySerial.begin(9600);
}

void loop(){
  //digitalWrite(led1, HIGH);
  //delay(1000);
  //digitalWrite(led1, LOW);
  //delay(1000);

  if(mySerial.available()){
    char c = mySerial.read();
    if (c == 'H'){
      digitalWrite(led, HIGH);
    }else if( c == 'L'){
      digitalWrite(led, LOW);
    }
  }
}
```

Código en Python

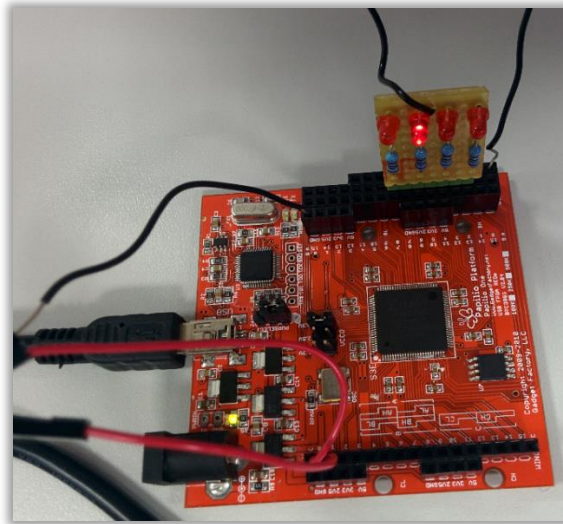
```
import serial

Zpuino = serial.Serial('/dev/ttyUSB2', 9600);

print ("Inicializando")

while 1:
  comando = raw_input('Introduce comando H o L: ')
  Zpuino.write (comando)
  if comando == 'H':
    print('LED ENCENDIDO')
  elif comando == 'L':
    print('LED APAGADO')
  Zpuino.close()
```

✚ Resultado final



1.9 Conclusión

En las dos sesiones de ZPUino que hemos tenido me ha ayudado a comprender la cantidad de cosas que podemos hacer con dicha plataforma y placa Papilio. Durante las dos sesiones he podido finalizar de forma óptima todas las tareas / ejercicios propuestos entendiendo perfectamente los códigos y el funcionamiento de la placa Papilio. Esto también ha sido debido a que los códigos utilizados son muy similares a los códigos utilizados en Arduino.

En definitiva, puedo decir que las dos sesiones han sido muy entretenidas y divertidas.