Cómo trabajar con el bloque IP AXI_GPIO

Creador: David Rubio G.

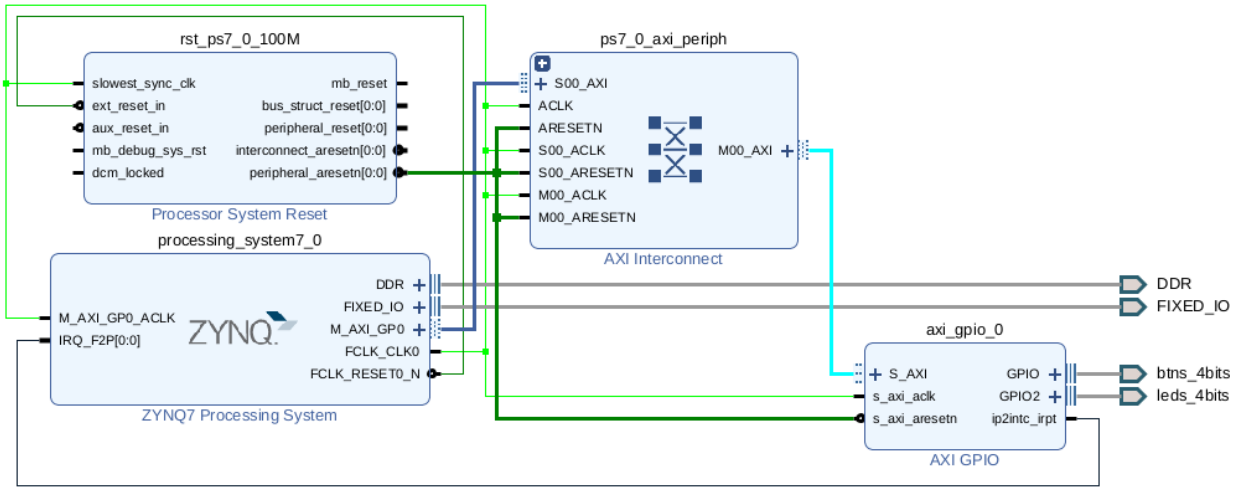
Entrada: https://soceame.wordpress.com/2024/06/24/como-trabajar-con-el-bloque-ip-axi_gpio/

Blog: https://soceame.wordpress.com/

GitHub: https://github.com/DRubioG

Fecha última modificación: 22/02/2025

Uno de los fundamentos más básicos de un SoC es el trabajo con señales discretas. Para ello Xilinx dispone de un bloque IP llamado **AXI_GPIO**, que se comunica por AXI con el SoC.

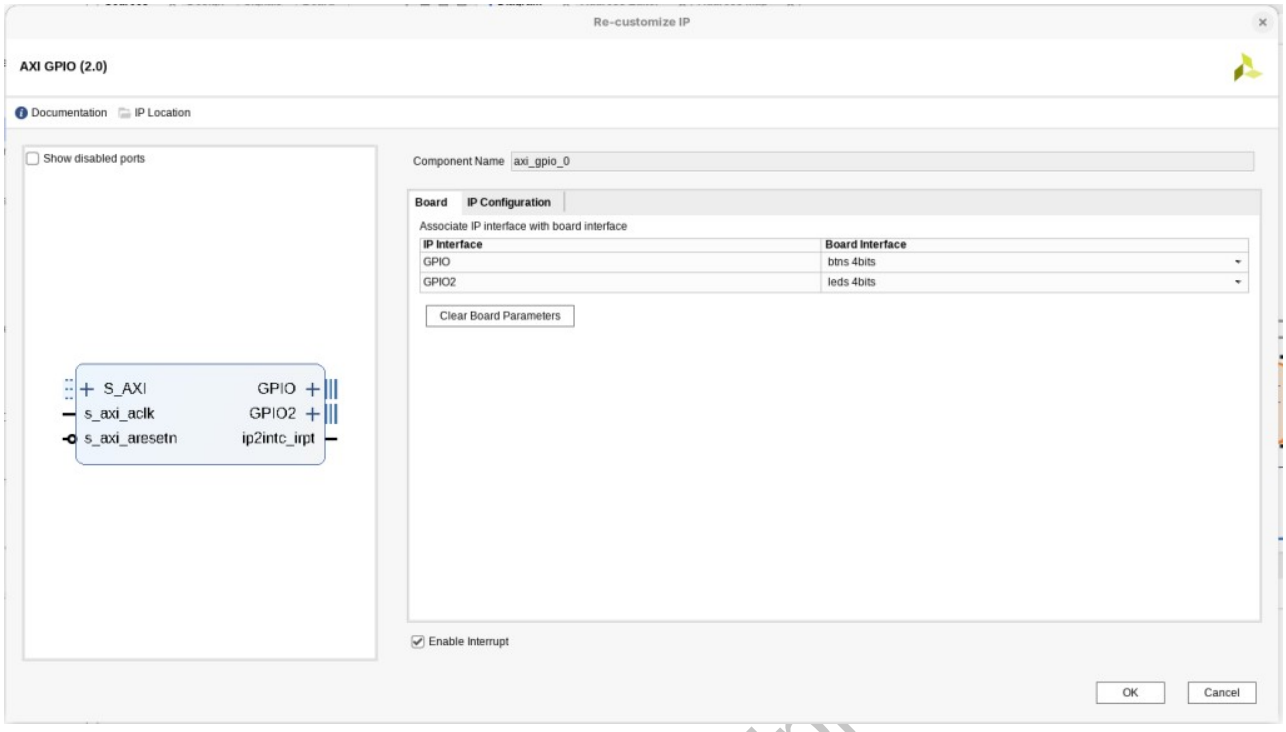


Vivado

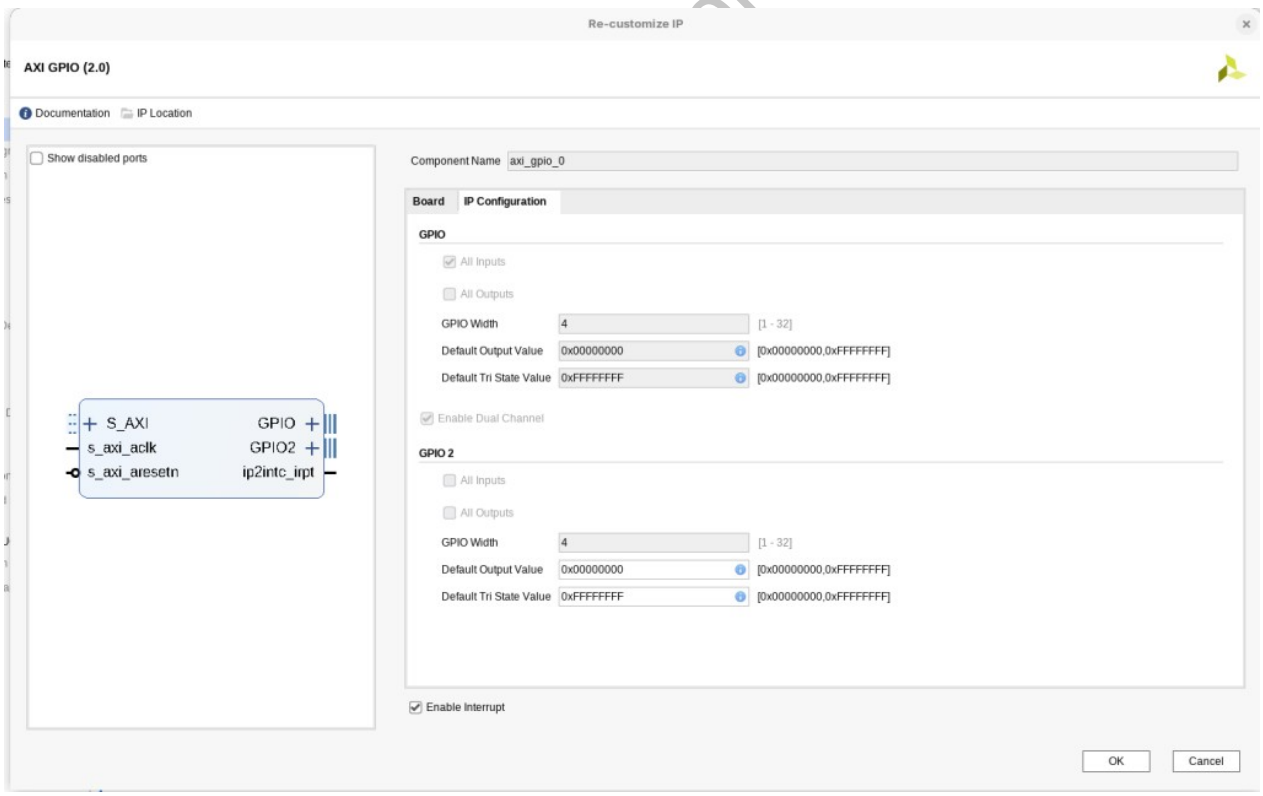
Para usar un bloque AXI_GPIO, primero hay que instanciarlo, para ello pulsamos Ctrl+i o le damos al símbolo del ‘+’ del *block design*.

Al abrirlo aparece una pestaña como la siguiente, en esta pestaña aparece en Board interfaces la interfaz preconfigurada de la placa que estás usando *(en caso de tenerla, si no tienes metida la placa en Vivado y estás trabajando con un chip genérico, te aparecerá directamente la pestaña IP configuration)*

https://soceame.wordpress.com/2024/06/24/como-trabajar-con-el-bloque-ip-axi_gpio/



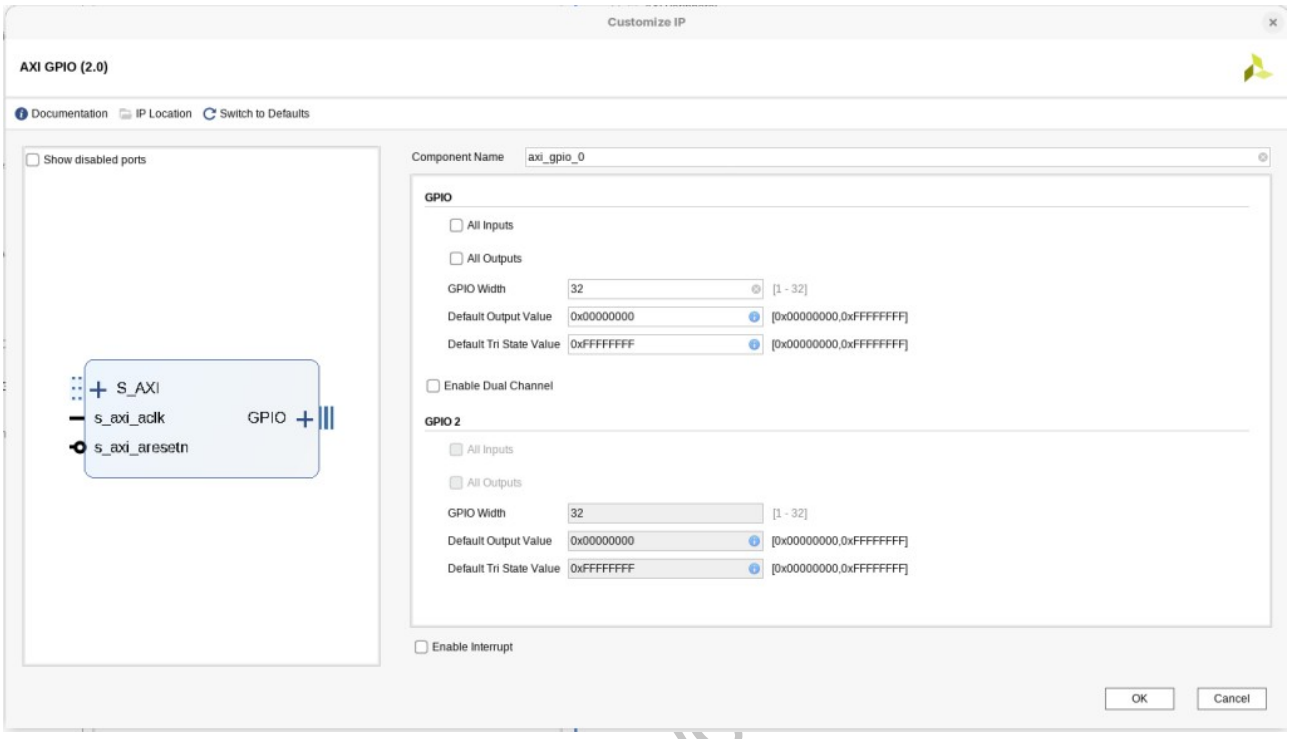
Y en la pestaña IP Configuration aparece toda la información del bloque IP.



Estos bloque IP pueden tener hasta dos interfaces (GPIO y GPIO2). Y pueden ser definidas como «Todo entradas», «Todo salidas» y mixto.

También se puede seleccionar cuantas salidas quieres tener en GPIO Width.

NOTA: *si trabajas con la configuración de una placa, muchas de estas opciones ya vienen marcadas como en la imagen anterior. La pestaña normal es la siguiente.*



La casilla de **Enable Dual Channel** es para activar o desactivar la segunda interfaz.

También se puede seleccionar el valor por defecto de los bloques IP tanto de salida (si no esta marcada la opción de todo entrada) como en triestado (si no están marcadas las opciones de «Todo entradas» o «Todo salida»).

Y por último la casilla de **Enable Interrupt** lo que hace es activar el puerto de salida de interrupción. Este puerto genera una señal de interrupción si alguna entrada se ha activado. Esta interrupción se canaliza al SoC (o a un *Concat* si hay más de una línea de interrupción).

Vitis

Una vez tenemos configurado el bloque AXI_GPIO y hemos generado el bitstream nos vamos a Vitis, generamos un proyecto.

Para poder configurar el bloque AXI_GPIO se tiene que declarar las librerías.

```
#include "xparameters.h"
#include "xgpio.h"
```

La primera librería es la que contiene todas las direcciones base, y la segunda la que contiene todas las funciones para trabajar con el AXI_GPIO.

Lo siguiente que se necesita es definir los canales. *En la primera imagen del blog aparece que los switches están en la interfaz GPIO, y los leds en la GPIO2*

```
#define LED_CHANNEL 2
#define SW_CHANNEL 1
```

También, se define cuál es la dirección base del bloque IP.

```
#define GPIO_EXAMPLE_DEVICE_ID  XPAR_GPIO_0_DEVICE_ID
```

Lo siguiente es definir los bits que se van a utilizar, en mi caso solo voy a utilizar 4 bits.

```
#define LED 0x0F
#define SW  0x0F
```

Y también, la instancia del bloque IP, se requiere de dos porque tenemos dos interfaces.

```
XGpio Gpio;
XGpio Gpio2;
```

Bien, estos son los paso previos a la ejecución. Ahora, definimos las dos interfaces y le asignamos la dirección base de los GPIO.

```
Status = XGpio_Initialize(&Gpio, GPIO_EXAMPLE_DEVICE_ID);

Status = XGpio_Initialize(&Gpio2, GPIO_EXAMPLE_DEVICE_ID);
```

Lo siguiente es definir la dirección de los GPIO. Para ello hay que tener en cuenta que ‘0’ son pines de salida y ‘1’ son pines de entrada.

```
XGpio_SetDataDirection(&Gpio, LED_CHANNEL, ~LED);
XGpio_SetDataDirection(&Gpio2, SW_CHANNEL, SW);
```

Con todo lo anterior ya se puede empezar a trabajar con lo GPIO. Para ello se hace uso de las siguientes funciones:

- u32 dato = **XGpio_DiscreteRead**(&<instancia>, <canal>) : esta función lee el valor de las señales de entrada, y lo devuelve en una variable de 32 bits.
- **XGpio_DiscreteWrite**(&<instancia>, <canal>, <posición>) : esta función escribe un ‘1’ en la posición indicada
- **XGpio_DiscreteClear**(&<instancia>, <canal>, <posición>) : esta función borra la posición de salida.

Existen más funciones pero las básicas son estas. Ahora te pongo un ejemplo para que veas como se usan.

Ejemplo

Este ejemplo está basado en el ejemplo base *xgpio_example* de Vivado. La función de este código es parpadear el led cuyo botón es pulsado.

```
#include "xparameters.h"
#include "xgpio.h"
#include "xil_printf.h"
```

```
#define SW  0x0F

#define GPIO_EXAMPLE_DEVICE_ID  XPAR_GPIO_0_DEVICE_ID

#define LED_DELAY     10000000

#define LED_CHANNEL 2
#define SW_CHANNEL 1

XGpio Gpio;
XGpio Gpio2;

int main(void)
{
        int Status;
        volatile int Delay;

        Status = XGpio_Initialize(&Gpio, GPIO_EXAMPLE_DEVICE_ID);
        if (Status != XST_SUCCESS) {
                xil_printf("Gpio Initialization Failed\r\n");
                return XST_FAILURE;
        }
        Status = XGpio_Initialize(&Gpio2, GPIO_EXAMPLE_DEVICE_ID);
        if (Status != XST_SUCCESS) {
                xil_printf("Gpio Initialization Failed\r\n");
                return XST_FAILURE;
        }

        XGpio_SetDataDirection(&Gpio, LED_CHANNEL, ~LED);
        XGpio_SetDataDirection(&Gpio2, SW_CHANNEL, SW);

        while (1) {
//leo el botón pulsado
                u32 sw_read = XGpio_DiscreteRead(&Gpio2, SW_CHANNEL);

// enciendo el led seleccionado
                XGpio_DiscreteWrite(&Gpio, LED_CHANNEL, sw_read);

                for (Delay = 0; Delay < LED_DELAY; Delay++);

// apago el led seleccionado
                XGpio_DiscreteClear(&Gpio, LED_CHANNEL, sw_read);

                for (Delay = 0; Delay < LED_DELAY; Delay++);
        }
}
```


Interrupción

Y por último para mencionar tenemos la posibilidad de generar una interrupción al pulsar un botón.

Para ello se tiene que configurar una interrupción como en este código. (También se puede usar como ejemplo base el que da Vitis *xgpio_intr_tapp_example*).

Lo único es tener en cuenta que tienes que borrar la interrupción cada vez que se ejecuta con la función *XGpio_InterruptClear.*

**En una próxima entrada explicaré como construir rutinas de interrupción en una Zynq.*