

# Manejo de puertos PIO y LCD en el NIOS II

Por: Kalun José Lau Gan

2021

1

## Revisión (changelog) del documento:

- 13-noviembre-2020: Redacción inicial del documento
- 26-mayo-2021: Se completó el detalle faltante de la asignación de las señales del módulo “Optrex 16207 LCD Controller Core” en el Quartus II. Se añadió mas ejemplos de uso de los PIO.

2

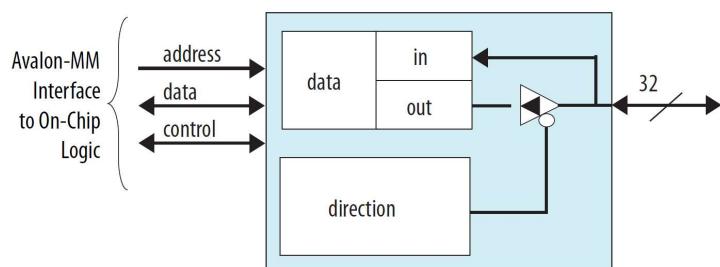
## Agenda:

- Manipulación de puertos de entrada y salida (módulo PIO)
- Ejemplos

3

## Manipulación de puertos en el NIOS II

- Referencia: Manual de IPs de NIOS II, sección “PIO”.



- Revisar librería *altera\_avalon\_pio\_regs.h*

4

## Aspectos iniciales:

- Los PIO pueden configurarse de 1 bit a 32 bits de ancho.
- Pueden ser entradas, salidas o bidireccionales.
- Los registros implicados para el uso del PIO son de 32 bits mostrados a continuación:

Table 273. Register Map for the PIO Core

Offset	Register Name	R/W	(n-1)	...	2	1	0
0	data	read access	R	Data value currently on PIO inputs			
		write access	W	New value to drive on PIO outputs			
1	direction (1)	R/W		Individual direction control for each I/O port. A value of 0 sets the direction to input; 1 sets the direction to output.			
2	interruptmask (1)	R/W		IRQ enable/disable for each input port. Setting a bit to 1 enables interrupts for the corresponding port.			
3	edgecapture (1), (2)	R/W		Edge detection for each input port.			
4	outset	W		Specifies which bit of the output port to set. Outset value is not stored into a physical register in the IP core. Hence it's value is not reserved for future use.			
5	outclear	W		Specifies which output bit to clear. Outclear value is not stored into a physical register in the IP core. Hence its value is not reserved for future use.			
<b>Note :</b>							
1. This register may not exist, depending on the hardware configuration. If a register is not present, reading the register returns an undefined value, and writing the register has no effect.							
2. If the option <b>Enable bit-clearing for edge capture register</b> is turned off, writing any value to the edgecapture register clears all bits in the register. Otherwise, writing a 1 to a particular bit in the register clears only that bit.							

5

## Librería altera\_avalon\_pio\_regs.h

```

31 #ifndef __ALTERA_AVALON_PIO_REGS_H__
32 #define __ALTERA_AVALON_PIO_REGS_H__
33
34 #include <iom.h>
35
36 #define IOADDR_ALTERA_AVALON_PIO_DATA(base)      __IO_CALC_ADDRESS_NATIVE(base, 0)
37 #define IORD_ALTERA_AVALON_PIO_DATA(base)        IORD(base, 0)
38 #define IOWR_ALTERA_AVALON_PIO_DATA(base, data)   IOWR(base, 0, data)
39
40 #define IOADDR_ALTERA_AVALON_PIO_DIRECTION(base)  __IO_CALC_ADDRESS_NATIVE(base, 1)
41 #define IORD_ALTERA_AVALON_PIO_DIRECTION(base)    IORD(base, 1)
42 #define IOWR_ALTERA_AVALON_PIO_DIRECTION(base, data) IOWR(base, 1, data)
43
44 #define IOADDR_ALTERA_AVALON_PIO_IRQ_MASK(base)   __IO_CALC_ADDRESS_NATIVE(base, 2)
45 #define IORD_ALTERA_AVALON_PIO_IRQ_MASK(base)     IORD(base, 2)
46 #define IOWR_ALTERA_AVALON_PIO_IRQ_MASK(base, data) IOWR(base, 2, data)
47
48 #define IOADDR_ALTERA_AVALON_PIO_EDGE_CAP(base)   __IO_CALC_ADDRESS_NATIVE(base, 3)
49 #define IORD_ALTERA_AVALON_PIO_EDGE_CAP(base)     IORD(base, 3)
50 #define IOWR_ALTERA_AVALON_PIO_EDGE_CAP(base, data) IOWR(base, 3, data)
51
52
53 #define IOADDR_ALTERA_AVALON_PIO_SET_BITS(base)   __IO_CALC_ADDRESS_NATIVE(base, 4)
54 #define IORD_ALTERA_AVALON_PIO_SET_BITS(base)     IORD(base, 4)
55 #define IOWR_ALTERA_AVALON_PIO_SET_BITS(base, data) IOWR(base, 4, data)
56
57 #define IOADDR_ALTERA_AVALON_PIO_CLEAR_BITS(base)  __IO_CALC_ADDRESS_NATIVE(base, 5)
58 #define IORD_ALTERA_AVALON_PIO_CLEAR_BITS(base)    IORD(base, 5)
59 #define IOWR_ALTERA_AVALON_PIO_CLEAR_BITS(base, data) IOWR(base, 5, data)
60
61
62
63 /* Definitions for direction-register operation with bi-directional PIOs */
64 #define ALTERA_AVALON_PIO_DIRECTION_INPUT 0
65 #define ALTERA_AVALON_PIO_DIRECTION_OUTPUT 1
66
67 #endif /* __ALTERA_AVALON_PIO_REGS_H__ */

```

6

## Aspectos iniciales:

- Los PIO se declaran en el QSys
  - De acuerdo a los requerimientos se considerará puertos de entrada o de salida (no es recomendable trabajar con bidireccionales en esta etapa inicial).
  - Tener en cuenta la dirección asignada a cada PIO y revisar que no haya conflicto.
  - Asignar un nombre a la conexión externa del puerto para que aparezcan en el PinPlanner



7

## Aspectos iniciales:

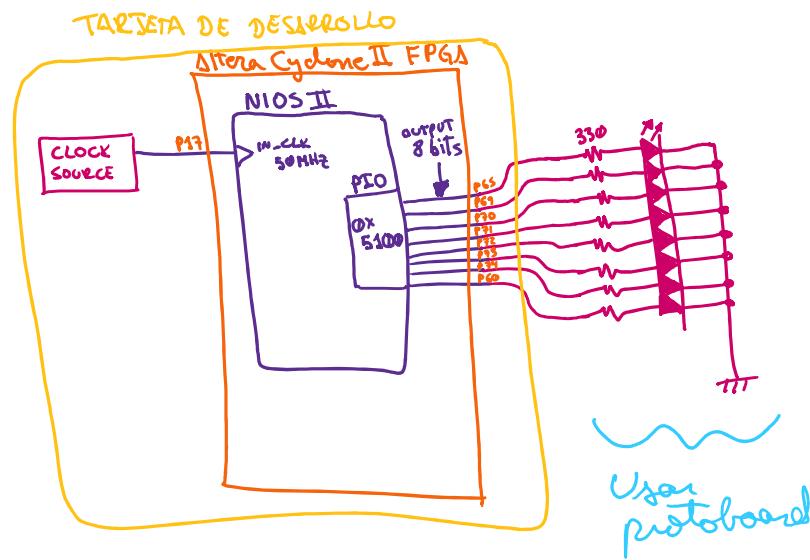
- Librería a incluir en el programa:
  - #include "altera\_avalon\_pio\_regs.h"
- Escritura de dato en el PIO:
  - IOWR\_ALTERA\_AVALON\_PIO\_DATA( [dirección del PIO], [dato]);
  - Ejemplo:
 

```
IOWR_ALTERA_AVALON_PIO_DATA(0x5100, 0x5A);
```
- Lectura de dato en el PIO:
  - IORD\_ALTERA\_AVALON\_PIO\_DATA( [dirección del PIO] );
  - Ejemplo (se tendrá que declarar una variable donde se alojará lo leído):
 

```
data_in = IORD_ALTERA_AVALON_PIO_DATA(0x5200);
```

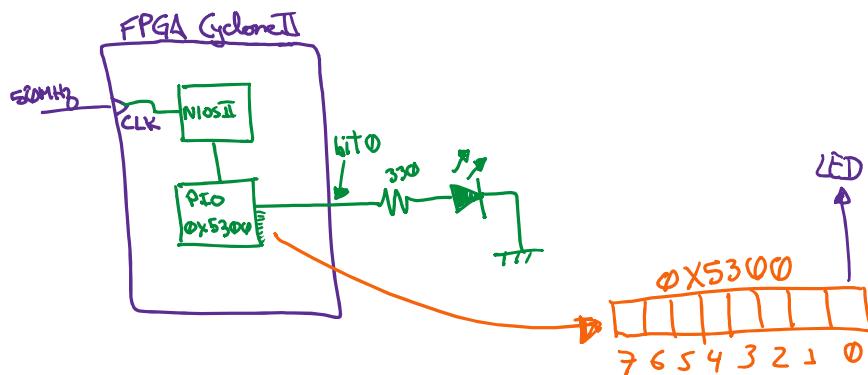
8

Ejemplo: PIO de salida de 8 bits en dirección 0x5100 conectado a ocho LEDs



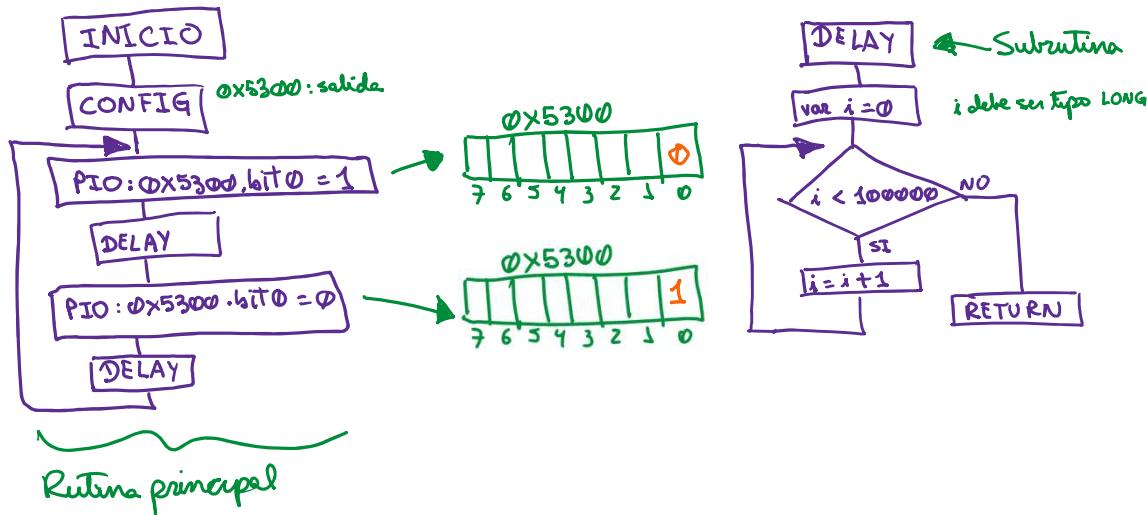
9

Ejemplo 1: Titilar un LED a través del bit0 del registro PIO de 8bits con dirección 0x5300



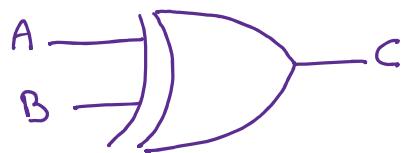
10

Ejemplo 1: Titilar un LED a través del bit0 del registro PIO de 8bits con dirección 0x5300

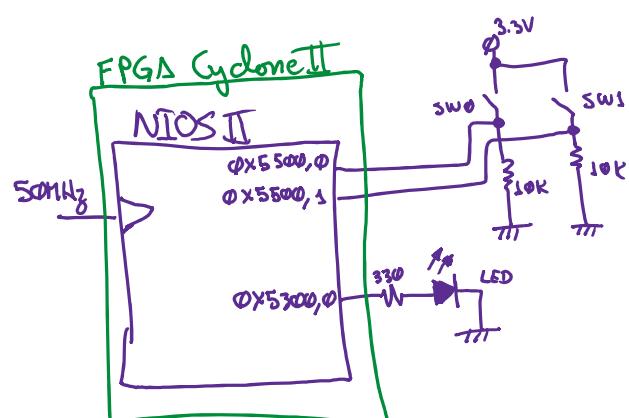


11

Ejemplo 2: Modelar una compuerta XOR en diagrama de flujo



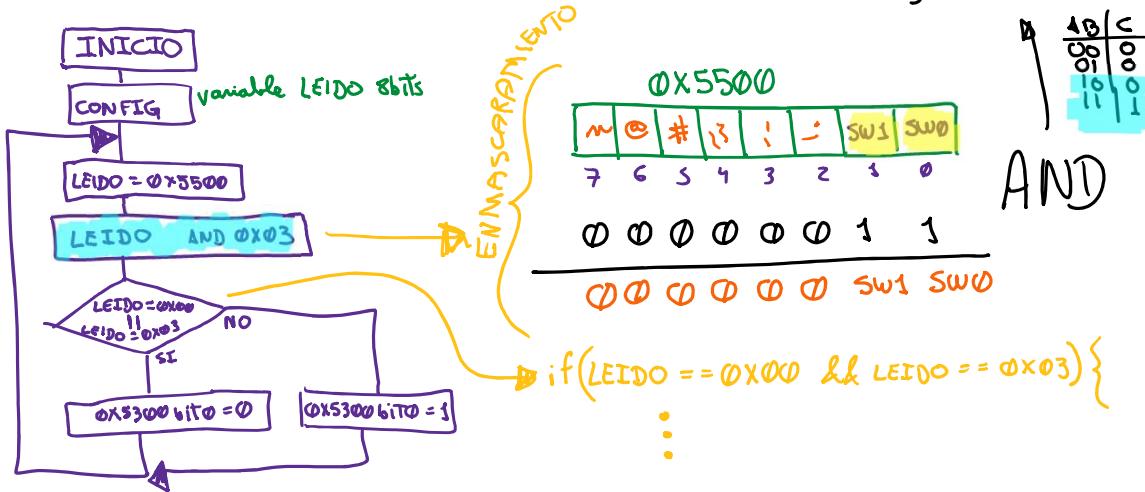
A	B	C
0	0	0
0	1	1
1	0	1
1	1	0



$0x5500: \text{PIO en input}$   
 $0x5300: \text{PIO en output}$

12

## Ejemplo 2: Modelar una compuerta XOR en diagrama de flujo



13

## Ejemplo 3: PIO de salida de 8 bits en dirección 0x5100

- El siguiente programa emite a través del PIO 0x5100 tres datos de manera secuencial y cíclica con un retardo definido por la subfunción `delay_s`
- Los datos son códigos en ASCII de la sigla UPC
- El comando `alt_putstr` se usa para enviar mensajes a la consola del NIOS II en el Eclipse

```

#include "sys/alt_stdio.h"
#include "system.h"
#include "altera_avalon_pio_regs.h"

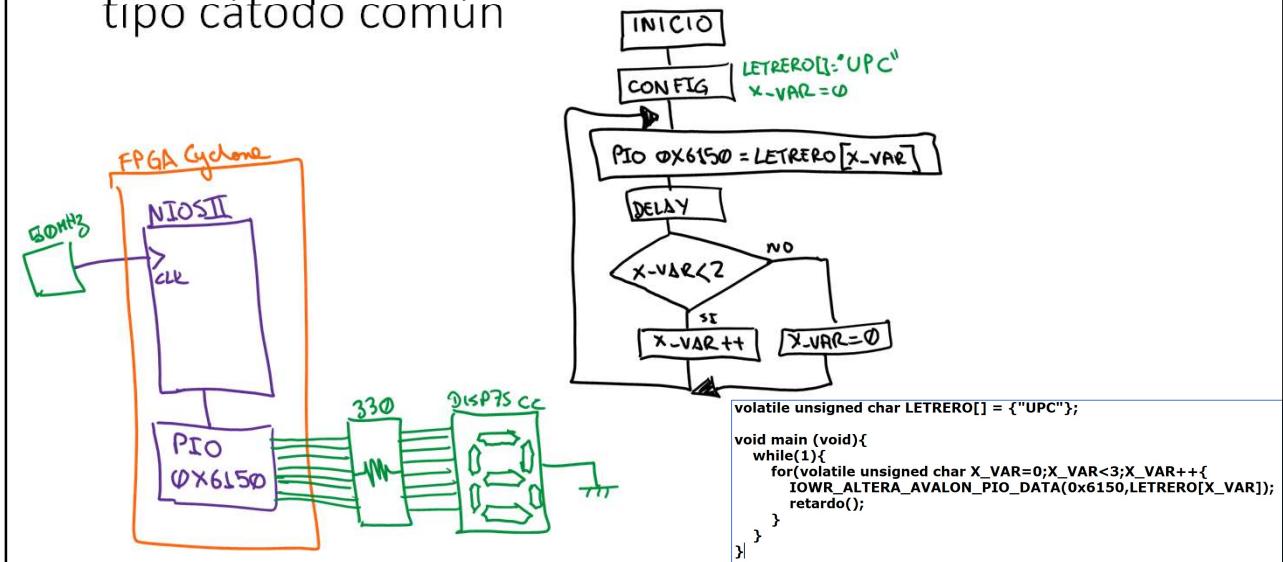
void delay_s(int tiempo){
    volatile int interno = 0;
    while (interno < tiempo*10000){
        interno++;
    }
}

int main()
{
    alt_putstr("Hello from Nios II!\n");
    /* Event loop never exits. */
    while (1){
        IOWR_ALTERA_AVALON_PIO_DATA(0x5100, 0x55); //Emite letra U
        delay_s(30);
        IOWR_ALTERA_AVALON_PIO_DATA(0x5100, 0x50); //Emite letra P
        delay_s(30);
        IOWR_ALTERA_AVALON_PIO_DATA(0x5100, 0x43); //Emite letra C
        delay_s(30);
    }
    return 0;
}

```

14

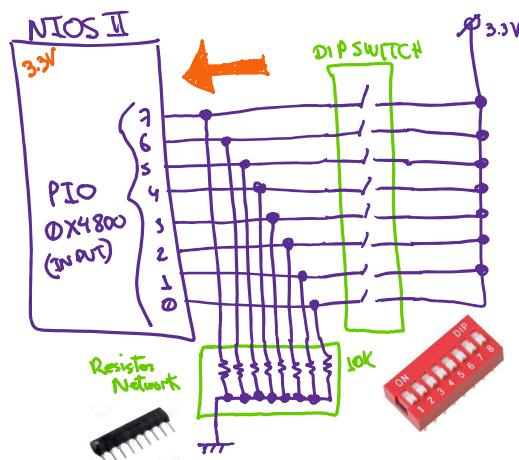
Ejemplo 4: Desarrollar un algoritmo para visualizar las siglas UPC en un display de siete segmentos del tipo cátodo común



15

## Lectura de un PIO en el NIOS II

- Ejemplo5: Conectar un DIP switch en un PIO 0x4800



16

## Lectura de un PIO en el NIOS II

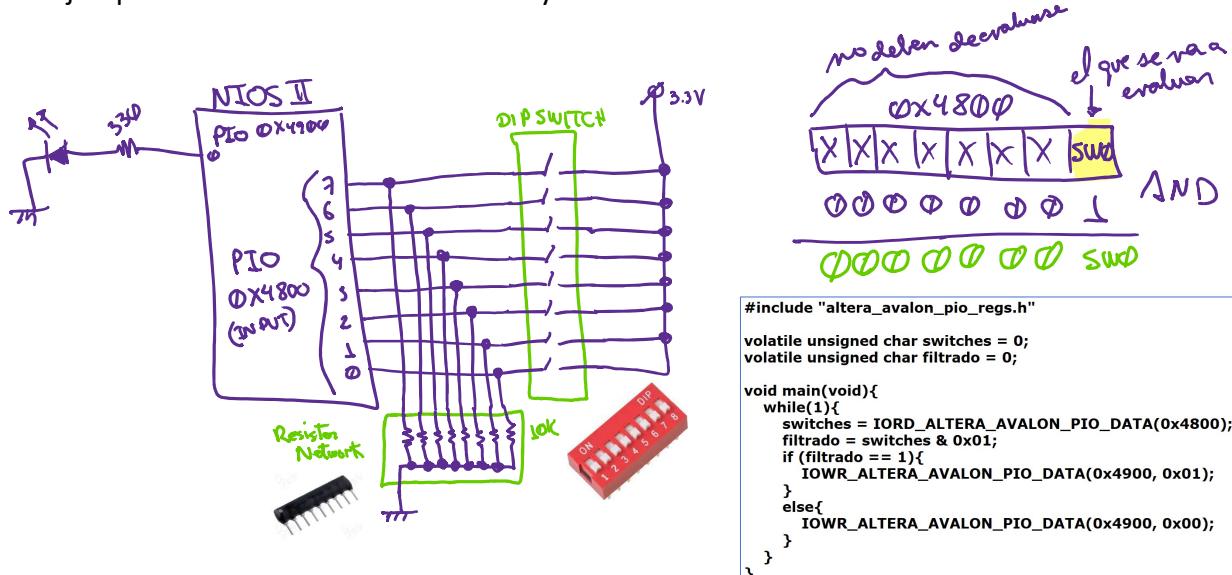
- Ejemplo5: Lectura del PIO en 0x4800 en código C del NIOSII

```
#include "altera_avalon_pio_regs.h"
volatile unsigned char switches = 0;
void main(void){
    while(1){
        switches = IORD_ALTERA_AVALON_PIO_DATA(0x4800);
    }
}
```

17

## Lectura de un PIO en el NIOS II

- Ejemplo 5: Lectura de bit0 en 0x4800 y enviarlo a bit0 de 0x4900



18

## Ejemplo 6: Desarrollar un negador lógico de un bit

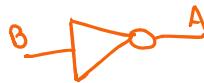
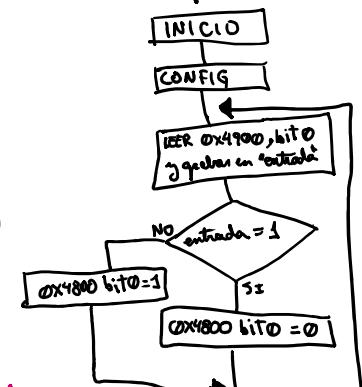
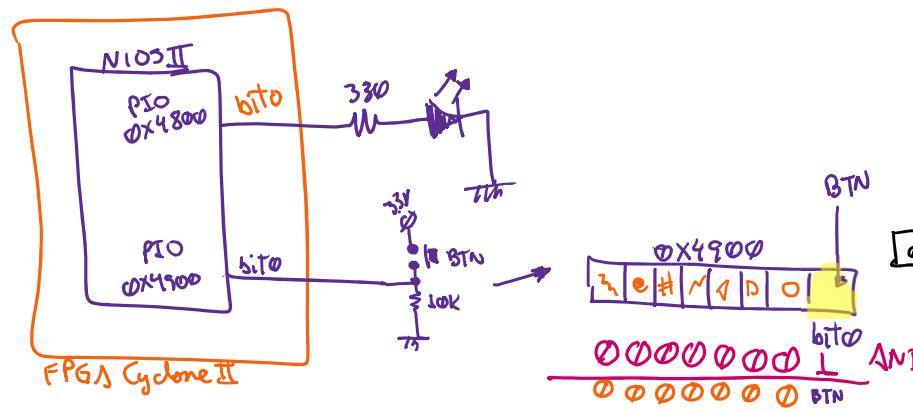


Diagrama de flujo:



✓ Flujo de  
enmascaramiento



19

## Ejemplo 6: Desarrollar un negador lógico de un bit - Código en C para NIOS II:

```

#include "altera_avalon_pio_regs.h"

volatile unsigned char switches = 0;
volatile unsigned char filtrado = 0;

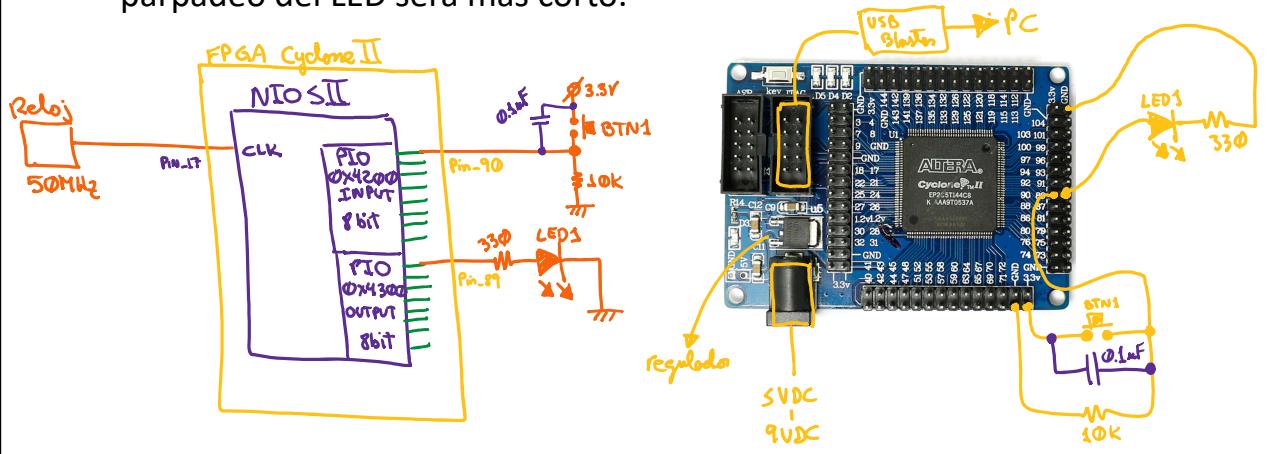
void main(void){
    while(1){
        switches = IORD_ALTERA_AVALON_PIO_DATA(0x4900);
        filtrado = switches & 0x01;
        if (filtrado == 1){
            IOWR_ALTERA_AVALON_PIO_DATA(0x4800, 0x00);
        }
        else{
            IOWR_ALTERA_AVALON_PIO_DATA(0x4800, 0x01);
        }
    }
}

```

20

## Ejemplo 7: Lectura de PIO entradas

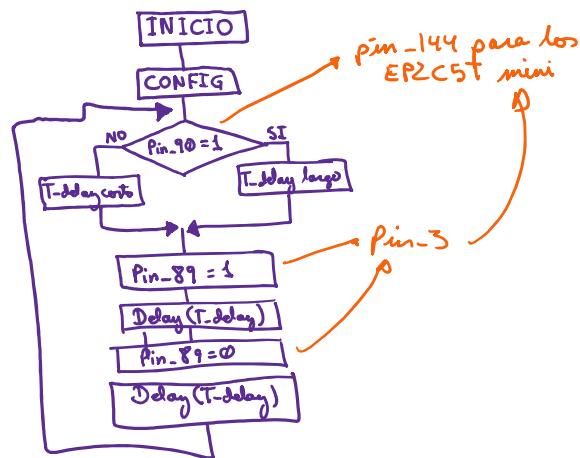
- En el siguiente circuito, cuando se presione el botón el tiempo de parpadeo del LED será mas corto.



21

## Ejemplo 7: Lectura de PIO entradas

- Algoritmo en diagrama de flujo



22

## Ejemplo 7: Lectura de PIO entradas

- Código en C para NIOS II

```
#include "sys/alt_stdio.h"
#include "altera_avalon_pio_regs.h"
#include "system.h"

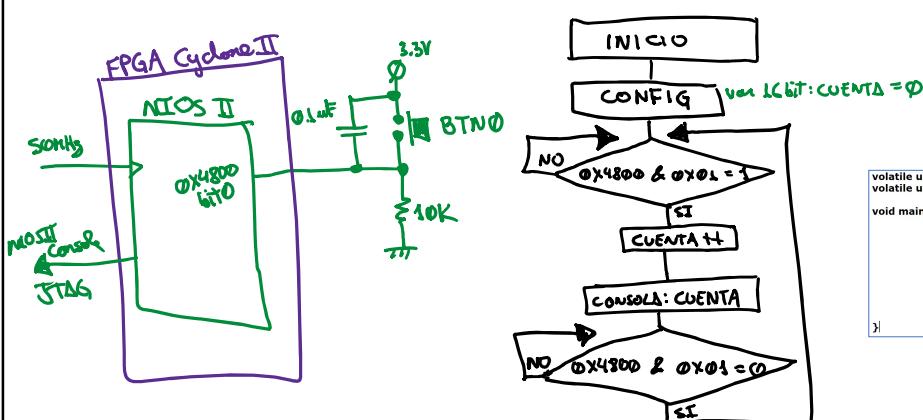
volatile unsigned char dato_in = 0;
volatile long tiempo = 0;

void main(void)
{
    alt_putstr("Hello from Nios II!\n");
    while(1){
        dato_in = (IORD_ALTERA_AVALON_PIO_DATA(0x4200)) & 0x01;
        if(dato_in == 1){
            tiempo = 200000;
        }
        else{
            tiempo = 100000;
        }
        IOWR_ALTERA_AVALON_PIO_DATA(0x4300, 0x01);
        usleep(tiempo);
        IOWR_ALTERA_AVALON_PIO_DATA(0x4300, 0x00);
        usleep(tiempo);
    }
    return 0;
}
```

Nota: Se está haciendo un enmascaramiento al momento de leer el PIO entrada para que solo se tome en cuenta el bit0 del puerto.

23

## Ejemplo 8: Detectar la cantidad de veces que se presiona un botón y visualizarlo como mensaje en la consola del NIOS II



```
volatile unsigned char LEIDO = 0;
volatile unsigned int CUENTA = 0;

void main (void){
    while(1){
        LEIDO = IORD_ALTERA_AVALON_PIO_DATA(0x4800);
        LEIDO = LEIDO & 0x01
        if (LEIDO == 1){
            CUENTA = CUENTA + 1;
            alt_putstr(CUENTA);
        }
    }
    while(LEIDO == 1);
}
```

24