Universidad de Costa Rica | Facultad de Ingeniería | Escuela de Ingeniería Eléctrica

Anteproyecto IV

Roberto Sánchez Cárdenas — B77059
Gabriel Jiménez Amador — B73895

San José, 31 de agosto de 2021
IE0424 — Laboratorio de Circuitos Digitales

Índice

1. Introducción

Este laboratorio tiene como fin utilizar las tarjeta FPGA en conjunto con los periféricos que la misma soporta.

2. Objetivos

- Investigar cómo utilizar periféricos en la tarjeta de desarrollo.
- Entender el mecanismo de interrupciones de un procesador.
- Realizar interfaz entre un periférico y un procesador sencillo.

3. Investigación previa

3.1. Documente cómo se habilitan las interrupciones en picorv32.

El PicoRV32 posee 32 entradas para controlar las interrupciones por medio de un bit. Es decir, se habilitan o deshabilitan las entradas poniendo un 1 o 0 en las entradas. Además, para habilitar las interrupciones, se debe colocar un 1 en la entrada `ENABLE_IRQ`. [5]

Las IRQs 0-2 se pueden activar internamente mediante las siguientes fuentes de interrupción integradas:

- IRQ 0: Interrupción del timer
- IRQ 1: `EBREAK` / `ECALL` o instrucción ilegal
- IRQ 2: Error de bus (acceso a memoria desalineado)

Estas interrupciones también pueden ser activadas por fuentes externas, como coprocesadores conectados a través de PCPI.

3.2. Documente cómo se procesan las interrupciones en picorv32.

En [5] la creadora del core PicoRV32 indica que las interrupciones poseen una señal `EOI` (End Of Interrupt) que indica el inicio (`HIGH`) y final (`LOW`) del procesamiento de una interrupción por el *interrupt handler*.

PicoRV32 tiene 4 registros (`q0` .. `q3`) adicionales de 32 bits que se utilizan para el manejo de interrupciones. Cuando se llama al *interrupt handler*, el registro `q0` va a contener la dirección de retorno y q1 contiene una máscara de bits de todas las interrupciones que se deben manejar. Esto significa que una llamada al *interrupt handler* necesita dar servicio a más de una interrupción cuando se establece más de un bit en `q1`. Los registros `q2` y `q3` no están inicializados y se pueden usar como almacenamiento temporal al guardar o restaurar valores de registro en el *interrupt handler*.

3.3. Investigue y documente el protocolo serial SPI.

¿Qué señales forman parte de la interfaz?

- Clock (`SCLK`): Señal de reloj para lógica secuencial. [4]
- Chip select (`CS`): Indica el esclavo destino. [4]
- Master out, slave in (`MOSI`) : Se encarga de mover las señales del maestro al esclavo. [4]
- Master in, slave out (`MISO`): Se encarga de mover las señales del esclavo al maestro. [4]

¿Cuál es el significado y la funcionalidad de cada una de las señales que conforman la interfaz?

La señal `SCLK` es una señal que pasa de 0 a 1 y viceversa de manera periódica. Esta es utilizada para sincronizar los datos enviados a la plataforma. También se tiene una señal llamada `CS`, que nos indica cual de las dos partes es el esclavo en un determinado momento, cuando está en alto, el esclavo se conecta al bus de datos. Finalmente se tiene las `MOSI` y `MISO`, estas son las encargadas de mover los datos entre el maestro y el esclavo. [4]

¿Cómo se da la comunicación entre un dos dispositivos (maestro/esclavo).

Para que haya comunicación, inicialmente se debe selecciónar cual será el maestro y cuál el esclavo. Una vez se tiene definifo y estable, se empiezan a utilizar datos desde el maestro hacia el esclaov por medio de las líneas MOSI. El esclavo es capaz de responder por medio de las líneas MISO.

3.4. Documente e investigue cómo leer los datos del acelerómetro presente en la tarjeta Nexys 4 DDR usando la interfaz SPI. ¿Qué registros y direcciones hay que leer para obtener la aceleración en los ejes x, y y z?

La tarjeta Nexys 4 posee un chip ADXL362 cuya función es medir la aceleración que se aplica sobre el dispositivo. Para ello, la interfaz utiliza una comunicación por medio del protocolo SPI. Este dispositivo es capaz de entregar hasta un precisión de 12 bits. [3]

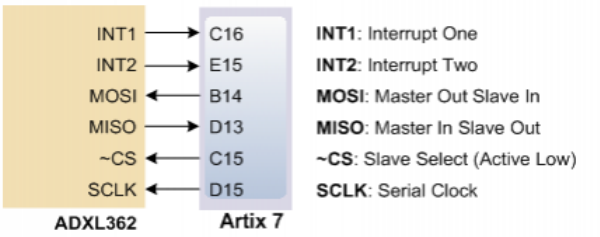

Figura 1: Interfaz del acelerómetro [3]

En la hoja de datos del sensor [1] se pueden encontrar los registros con sus respectivas direcciones, estos se muestran en la siguiente imágenes:

Address: 0x0E, Reset: 0x00, Name: XDATA_L

B7	B6	B5	B4	B3	B2	B1	B0
0	0	0	0	0	0	0	LSB

Address: 0x0F, Reset: 0x00, Name: XDATA_H

B15	B14	B13	B12	B11	B10	B9	B8
SX	SX	SX	SX	MSB	0	0	0

Figura 2: Registro para el eje X [1]

Address: 0x10, Reset: 0x00, Name: YDATA_L

B7	B6	B5	B4	B3	B2	B1	B0
0	0	0	0	0	0	0	LSB

Address: 0x11, Reset: 0x00, Name: YDATA_H

B15	B14	B13	B12	B11	B10	B9	B8
SX	SX	SX	SX	MSB	0	0	0

Figura 3: Registro para el eje Y [1]

Address: 0x12, Reset: 0x00, Name: ZDATA_L

B7	B6	B5	B4	B3	B2	B1	B0
0	0	0	0	0	0	0	LSB

Address: 0x13, Reset: 0x00, Name: ZDATA_H

B15	B14	B13	B12	B11	B10	B9	B8
SX	SX	SX	SX	MSB	0	0	0

Figura 4: Registro para el eje Z [1]

3.5. Documente e investigue cómo configurar una interrupción del acelerómetro presente en la tarjeta Nexys 4 DDR para configurar una interrupción que se dispare solamente cuando haya 'actividad'.

El sensor ADXL362 posee dos registros para interrupciones `INTMAP1` e `INTMAP2`, los cuales permiten detectar la activación de una interrupción. Para accesarlos se utilizan los pines mostrados en la Figura 1: `INT1` e `INT2`.

Para detectar actividad, el ADXL362 compara el valor absoluto del valor de aceleración de 12 bits con signo con el de 11 bits sin signo. Estos datos se guardan en los registros mostrados en la siguiente figura:

Figura 5: Activity threshold registers

Además se tiene un registro de lecturas de tiempo en la dirección `0x22`, este se llama `TIME_ACT`. Éste es el encargado de evitar mediciones erróneas, pues si está activo, solo se van a detectar mediciones que se mantiene por cierto tiempo prolongado. Se puede calcular y configurar el tiempo, se debe usar la siguiente ecuación:

$$\text{time} = \frac{\texttt{TIME\_ACT}}{\texttt{ODR}}$$

Donde `TIME_ACT` es un calor seteado por medio de un registro y `ORD` es la tasa de salida de datos seteada en `FILTER_CTL` (`0x2C`).

4. Diseño

Siguiendo los nombres y conexiones de los pines en el esquemático del Nexys 4 DDR en [2] se construye el archivo de restricción de pines en la Figura 6 para los Ejercicios 2 y 3 y así incluir las conexiones del display de 7 segmentos y las del acelerómetro.

```
set_property CFGBVS VCCO [current_design]
set_property CONFIG_VOLTAGE 3.3 [current_design]

# clk
set_property -dict { PACKAGE_PIN E3 IOSTANDARD LVCMOS33 } [get_ports { clk }];
create_clock -add -name sys_clk_pin -period 10.00 [get_ports {clk}];

# switches
set_property -dict { PACKAGE_PIN V10 IOSTANDARD LVCMOS33 } [get_ports { resetn }];

# leds
set_property -dict { PACKAGE_PIN H17 IOSTANDARD LVCMOS33 } [get_ports { out_byte[0] }];
set_property -dict { PACKAGE_PIN K15 IOSTANDARD LVCMOS33 } [get_ports { out_byte[1] }];
set_property -dict { PACKAGE_PIN J13 IOSTANDARD LVCMOS33 } [get_ports { out_byte[2] }];
set_property -dict { PACKAGE_PIN N14 IOSTANDARD LVCMOS33 } [get_ports { out_byte[3] }];
set_property -dict { PACKAGE_PIN R18 IOSTANDARD LVCMOS33 } [get_ports { out_byte[4] }];
set_property -dict { PACKAGE_PIN V17 IOSTANDARD LVCMOS33 } [get_ports { out_byte[5] }];
set_property -dict { PACKAGE_PIN U17 IOSTANDARD LVCMOS33 } [get_ports { out_byte[6] }];
set_property -dict { PACKAGE_PIN U16 IOSTANDARD LVCMOS33 } [get_ports { out_byte[7] }];

set_property -dict { PACKAGE_PIN V12 IOSTANDARD LVCMOS33 } [get_ports { out_byte_en }];
set_property -dict { PACKAGE_PIN V14 IOSTANDARD LVCMOS33 } [get_ports { trap }];

#7 segment display
# cathodes
set_property -dict { PACKAGE_PIN H15   IOSTANDARD LVCMOS33 } [get_ports { c_out[7] }];
set_property -dict { PACKAGE_PIN L18   IOSTANDARD LVCMOS33 } [get_ports { c_out[6] }];
set_property -dict { PACKAGE_PIN T11   IOSTANDARD LVCMOS33 } [get_ports { c_out[5] }];
set_property -dict { PACKAGE_PIN P15   IOSTANDARD LVCMOS33 } [get_ports { c_out[4] }];
set_property -dict { PACKAGE_PIN K13   IOSTANDARD LVCMOS33 } [get_ports { c_out[3] }];
set_property -dict { PACKAGE_PIN K16   IOSTANDARD LVCMOS33 } [get_ports { c_out[2] }];
set_property -dict { PACKAGE_PIN R10   IOSTANDARD LVCMOS33 } [get_ports { c_out[1] }];
set_property -dict { PACKAGE_PIN T10   IOSTANDARD LVCMOS33 } [get_ports { c_out[0] }];

# anodes
set_property -dict { PACKAGE_PIN U13   IOSTANDARD LVCMOS33 } [get_ports { an_out[7] }];
set_property -dict { PACKAGE_PIN K2    IOSTANDARD LVCMOS33 } [get_ports { an_out[6] }];
set_property -dict { PACKAGE_PIN T14   IOSTANDARD LVCMOS33 } [get_ports { an_out[5] }];
set_property -dict { PACKAGE_PIN P14   IOSTANDARD LVCMOS33 } [get_ports { an_out[4] }];
set_property -dict { PACKAGE_PIN J14   IOSTANDARD LVCMOS33 } [get_ports { an_out[3] }];
set_property -dict { PACKAGE_PIN T9    IOSTANDARD LVCMOS33 } [get_ports { an_out[2] }];
set_property -dict { PACKAGE_PIN J18   IOSTANDARD LVCMOS33 } [get_ports { an_out[1] }];
set_property -dict { PACKAGE_PIN J17   IOSTANDARD LVCMOS33 } [get_ports { an_out[0] }];

# accelerometer
set_property -dict { PACKAGE_PIN E15   IOSTANDARD LVCMOS33 } [get_ports { MISO }];
set_property -dict { PACKAGE_PIN F14   IOSTANDARD LVCMOS33 } [get_ports { MOSI }];
set_property -dict { PACKAGE_PIN F15   IOSTANDARD LVCMOS33 } [get_ports { SCLK }];
set_property -dict { PACKAGE_PIN D15   IOSTANDARD LVCMOS33 } [get_ports { CS }];
set_property -dict { PACKAGE_PIN B13   IOSTANDARD LVCMOS33 } [get_ports { INT[0] }];
set_property -dict { PACKAGE_PIN C16   IOSTANDARD LVCMOS33 } [get_ports { INT[1] }];
```

Figura 6: Archivo de restricciones de localización de pines a utilizar.

4.1. Ejercicio 1

Sabemos que las interrupciones pueden ser leídas en la dirección de memoria 0x00000010, por lo tanto cuando se lee esta y se obtiene un 1, se sabe que se activó una interrupción. Se desea contar el número de interrupciones activadas, por lo que se debe habilitar un contador en

el firmware activado por esta dirección, de modo que el firmware a utilizar se ve de la siguiente manera:

```
#include <stdint.h>

#define LED_REGISTERS_MEMORY_ADD 0x10000000
#define LOOP_WAIT_LIMIT 100
#define INTERRUPT_DIR 0x00000010

static void putuint(uint32_t i) {
*((volatile uint32_t *)LED_REGISTERS_MEMORY_ADD) = i;
}

void main() {
uint32_t number_to_display = 0;
uint32_t counter = 0;

while (1) {
counter = 0;
putuint(number_to_display);
number_to_display++;
while (1) {
if (*((volatile uint32_t *)INTERRUPT_DIR) ){
                counter++;
            }
            putuint(counter);
}
}
}
```

Figura 7: Firmware inicial a utilizar en el Ejercicio 1.

4.2. Ejercicio 2

Para leer la velocidad medida por el acelerometro, se deben leer las direcciones 0x20000000 y 0x30000000 para obtener la velocidad en tanto en y como en z. La velocidad en z se muestra en las primeras 4 pantallas de 7 segmentos y la de y en las siguientes 4. El firmware diseñado para este propósito es el siguiente.

```
#include <stdint.h>
#define LED_REGISTERS_MEMORY_ADD 0x10000000
#define LOOP_WAIT_LIMIT 100
#define ACCELEROMETER_Y 0x20000000
#define ACCELEROMETER_Z 0x30000000

static void putuint(uint32_t i) {
*((volatile uint32_t *)LED_REGISTERS_MEMORY_ADD) = i;
}
static void write_to_addr(uint32_t addr, uint32_t data) {
*((volatile uint32_t *)addr) = data;
}



void main() {
    uint32_t number_to_display = 0;
    uint32_t counter = 0;
    uint32_t acc_z = 0;
    uint32_t acc_y = 0;

    while (1) {
        counter = 0;
        acc_z = *((volatile uint32_t *)ACCELEROMETER_Z); //read
        acc_y = *((volatile uint32_t *)ACCELEROMETER_Y); //read

        write_data(0x0000, acc_z);

        while (counter < LOOP_WAIT_LIMIT) counter++; //delay

        write_data(0x1000, acc_y);

    }
}
```

Figura 8: Firmware inicial a utilizar en el Ejercicio 2.

Además del firware, para utilizar el protocolo de comunicaciones SPI se debe diseñar un medio de control. Esto se logra por medio de una máquina de estados. El diseño de la misma se muestra en alto nivel en la siguiente figura:

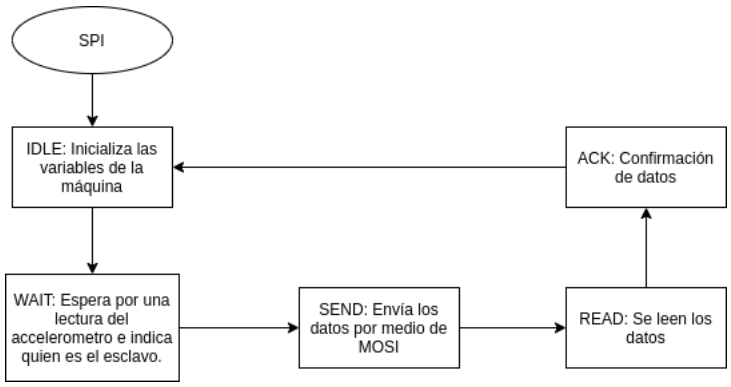

Figura 9: Máquina de control SPI

4.3. Ejercicio 3

Se debe modificar el firmware presentado en la sección anterior de modo que se tomen en cuenta únicamente las interrupciones por el acelerómetro. Se debe leer el registro de STATUS en la dirección 0x0B. Cuando esto ocurre se procede a enviar los datos a las pantallas de 7 segmentos de la misma manera como se trabaja en las partes previas. Una vez terminada la interrupción y enviados los datos por medio de SPI, de devuelve al estado inicial.

Es importante recordar que el registro INTMAP1 es el que presenta la interrupción por el acelerómetro, por lo que se debe verificar su respectivo pin.

Referencias

[1] Analog devices. *Data sheet: ADXL362*. URL: https://www.analog.com/media/en/technical-documentation/data-sheets/ADXL362.pdf.

[2] Inc. Digilent. *Nexys 4 DDR Schematic*. 2014. URL: https://reference.digilentinc.com/_media/reference/programmable-logic/nexys-4-ddr/nexys-4-ddr_sch.pdf.

[3] Diligent. *Nexys 4 Reference Manual*. URL: https://reference.digilentinc.com/programmable-logic/nexys-4/reference-manual.

[4] Kashif Javed Muhammad Tahir. *ARM Microprocessor Systems: Cortex-M Architecture, Programming, and Interfacing*. CRC Press, 2017.

[5] Clifford wolf. *Github: Picorv32*. URL: https://github.com/cliffordwolf/picorv32.