Trabajo Práctico Final
Circuitos Lógicos Programables

IP Core para manejo de encoder rotativo incremental con FPGA

Ing. Ignacio Majul

Agosto de 2020

Carrera de Especialización en Sistemas Embebidos

10ma Cohorte

Índice de contenido

1. Introducción

1.1. Objetivo

El objetivo de esta memoria es mostrar el proyecto implementado para el manejo de un encoder rotativo incremental con una plataforma basada en lógica reconfigurable (FPGA) y un Hardcore de la familia ZYNQ 7000 basado en arquitectura Cortex-A. Esta memoria fue presentada como trabajo práctico final de la materia Microarquitecturas y Softcores de la CESE y reutiliza el código HDL desarrollado como trabajo final de la materia Circuitos Lógicos Programables de la CESE.

1.2. Herramientas utilizadas

Para realizar este trabajo práctico se utilizó un encoder rotativo incremental de 360 pulsos por revolución, una placa de desarrollo Arty Z7-10 y una Laptop. La herramienta de software para el desarrollo fue Vivado Design Suite de Xilinx, el bloque IP Integrator para la implementación del sistema de procesamiento (PS) y el IDE SDK.

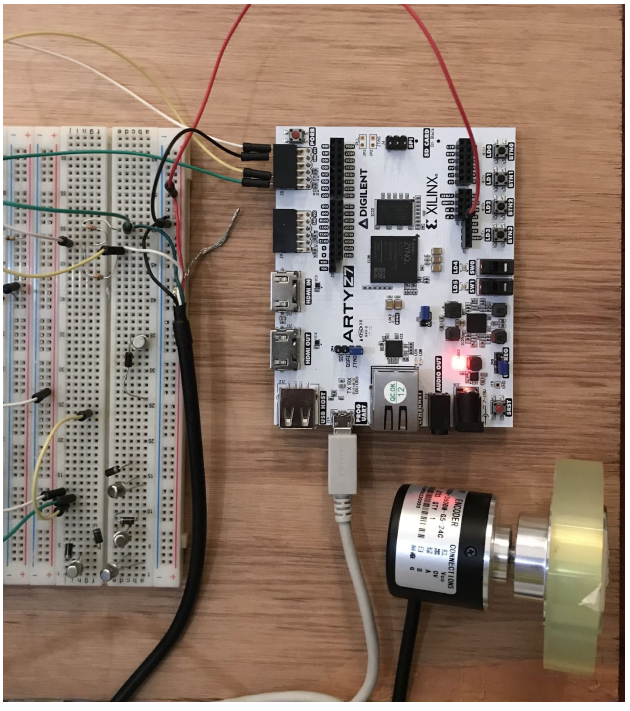
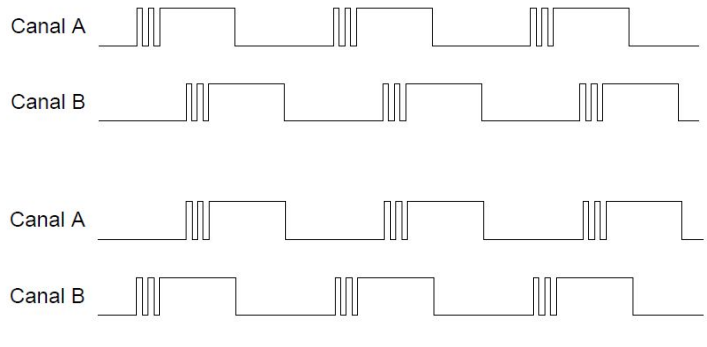


Figura 1: Banco de pruebas

1.3. Descripción del sistema

El encoder LPD3806-360BM genera una señal cuadrada de DC = 50%, desfasadas 90° eléctricos por cada uno de los dos canales de salida. Cada señal genera 360 pulsos por cada revolución completa del encoder. La posición relativa del encoder, respecto a su punto inicial, se determina contando la cantidad de pulsos generados en ese intervalo. Para determinar el sentido de giro (horario o antihorario) se debe evaluar cuál de las dos señales está adelantada respecto a la otra:



Horario

Antihorario

Para obtener una lectura confiable es preciso filtrar las señales de entrada de manera de eliminar cualquier posible rebote que pudiera aparecer y que pudiera traducirse en una interpretación errónea de su comportamiento.

Las señales de entrada del sistema son:

- Clock: 50 MHz (escalar)
- Canales A y B del encoder (escalar)
- Pulsador de reset (escalar)

Las señales de salida del sistema son:

- Sentido de giro (escalar)
- Posición del encoder (vector de 32 bits)

Las tres señales digitales de entrada (Canal A, Canal B y Reset) son canalizadas hacia el bloque IP desarrollado para ser interpretadas y analizadas por la lógica reconfigurable.

Las dos señales de salida (posición y sentido de giro) se montan sobre el bus de comunicaciones y son enviadas al sistema de procesamiento para ser tomadas por el microcontrolador.

2. Desarrollo del IP Core

2.1. Sistema base

El sistema base se compone del microcontrolador contenido en la placa Arty Z7-10, de la familia ZYNQ 7000 y basado en un dual-core Cortex-A9.

Los periféricos habilitados para la inclusión del IP fueron: la UART_0 para comunicación del microcontrolador con la PC y el bus AXI para comunicación entre PL y PS. El bus AXI se utiliza en su variante AXI4-Lite debido a las prestaciones necesarias para este proyecto. La operación que se ejecuta es “Read” la cual envía mensajes desde el AXI Slave hacia el AXI Master, escribiendo los registros de 32 bits que se disponen en el bus en las direcciones de memoria informadas por el Master.

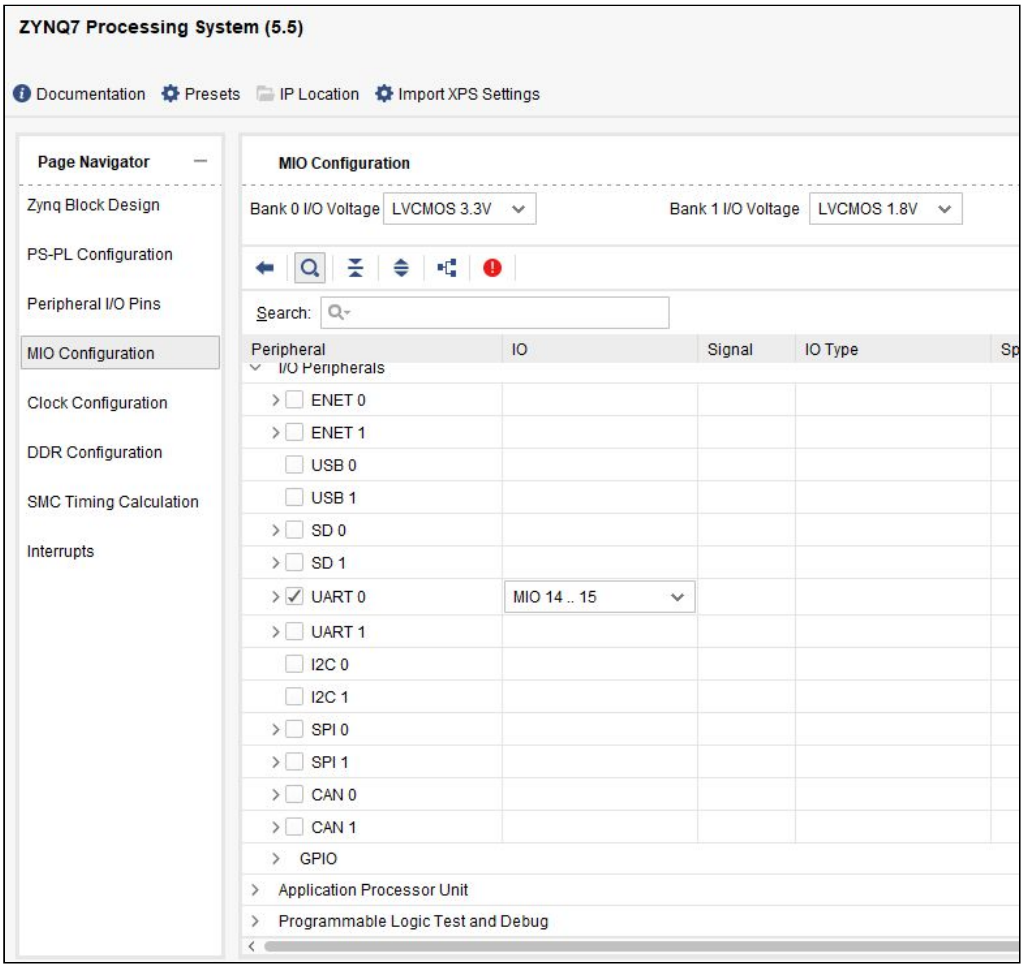


Figura 2. Configuración del sistema base.

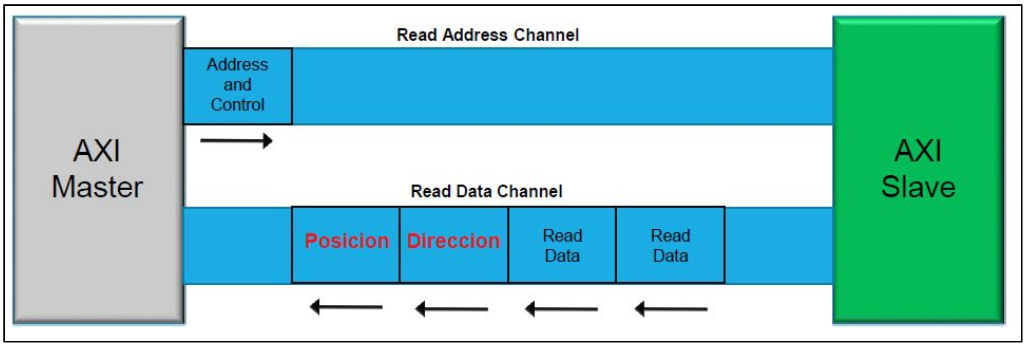


Figura 3. Operación “Read” de AXI4-Lite.

2.2. IP Propio

El IP propio contiene los archivos HDL que describen el funcionamiento de la lógica de manejo de señales del encoder. Las entradas del sistema son mapeadas a pines de propósito general de la placa (JB_1N y JB_1P) y las salidas son sobre-escritas en los registros slv_reg_0 y slv_reg_1 del bus AXI.

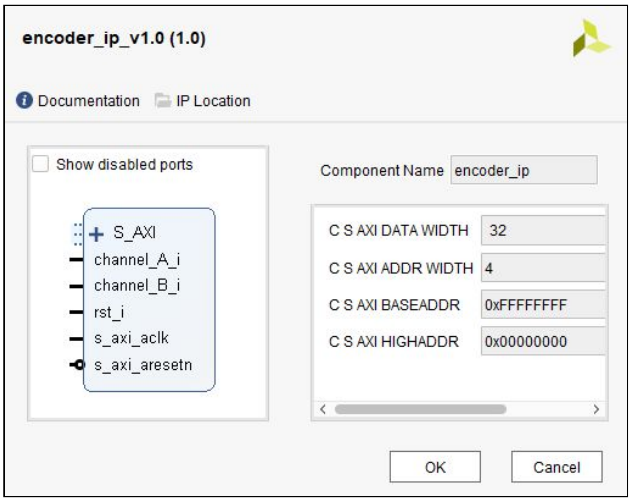


Figura 4. Configuración IP propio

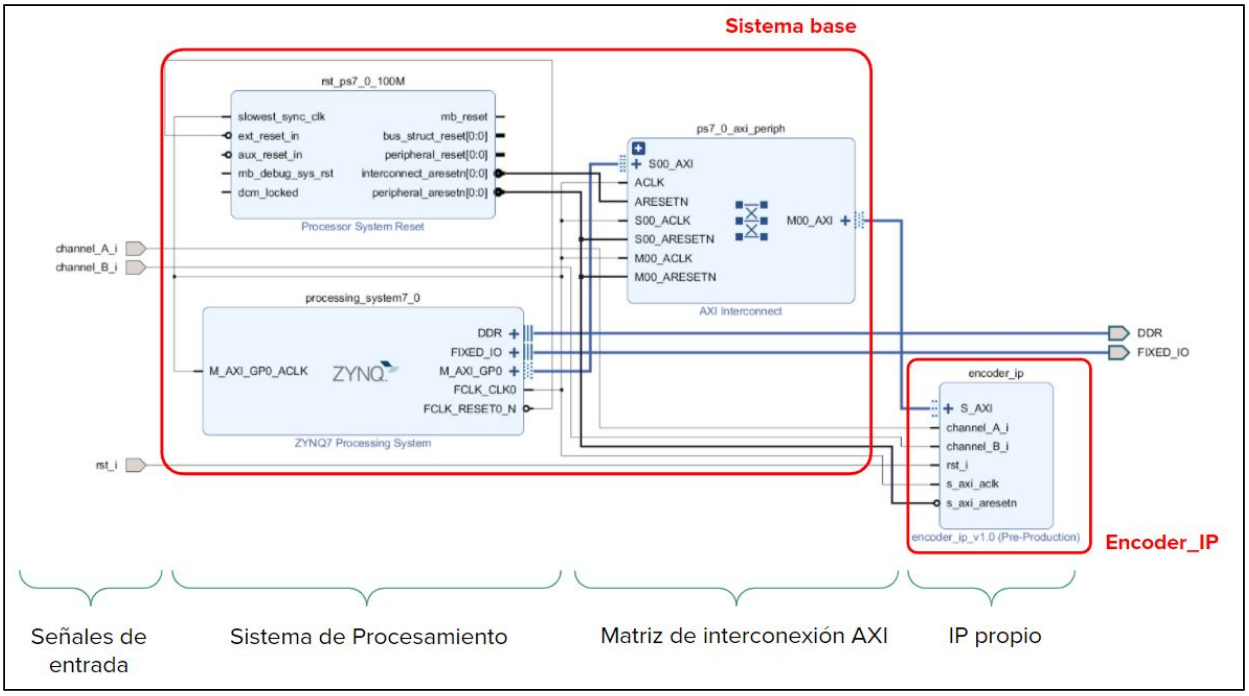


Figura 5. Diagrama de bloques general

2.3. Configuración HDL

Para incorporación de las entradas digitales y salidas por comunicación se debió adecuar los archivos de bajo nivel del bus AXI y se instancia la lógica de usuario:

Se agregan los puertos en la lógica de la interfaz AXI

```
-- component declaration
component encoder_ip_v1_0_S_AXI is
    generic (
    C_S_AXI_DATA_WIDTH  : integer   := 32;
    C_S_AXI_ADDR_WIDTH  : integer   := 4
    );
    port (
    CHANNEL_A_I : in std_logic;
    CHANNEL_B_I : in std_logic;
    RST_I: in std_logic;

    S_AXI_ACLK  : in std_logic;
    S_AXI_ARESETN   : in std_logic;
    S_AXI_AWADDR    : in std_logic_vector(C_S_AXI_ADDR_WIDTH-1 downto 0);
    S_AXI_AWPROT    : in std_logic_vector(2 downto 0);
    S_AXI_AWVALID   : in std_logic;
    S_AXI_AWREADY   : out std_logic;
    S_AXI_WDATA : in std_logic_vector(C_S_AXI_DATA_WIDTH-1 downto 0);
    S_AXI_WSTRB : in std_logic_vector((C_S_AXI_DATA_WIDTH/8)-1 downto 0);
```

Figura 6. Modificación instancia AXI

Se modifican los registros de salida del bus AXI

```
variable loc_addr :std_logic_vector(OPT_MEM_ADDR_BITS downto 0);
begin
    -- Address decoding for reading registers
    loc_addr := axi_araddr(ADDR_LSB + OPT_MEM_ADDR_BITS downto ADDR_LSB);
    case loc_addr is
      when b"00" =>
        reg_data_out <= pos_reg;
      when b"01" =>
        reg_data_out <= dir_reg;
      when b"10" =>
        reg_data_out <= slv_reg2;
      when b"11" =>
        reg_data_out <= slv_reg3;
      when others =>
        reg_data_out  <= (others => '0');
    end case;
end process;
```

Figura 7. Modificación registros AXI

Lógica de usuario

```
-- Add user logic here
U1: entity work.encoder
   generic map(
        N => 32,
        debounce_time => 10,
        freq_clk => 50
   )
    port map(
        clk_i        => S_AXI_ACLK,
        pos_o => pos_reg,
        dir_o => dir_aux,
        channel_A_i => CHANNEL_A_I,
        channel_B_i => CHANNEL_B_I,
        rst_i =>  RST_I
    );
-- User logic ends
```

Figura 8. Incorporación lógica de usuario

2.4. Código C

Para lectura de las señales provenientes del bus AXI se ejecutó el siguiente script en lenguaje C, desde el IDE SDK provisto por Vivado. El código realiza la operación de lectura cíclica sobre los registros propios del Bus y cuyas direcciones se encuentran indicadas en las definiciones de los archivos de cabecera.

Para visualizar las señales obtenidas del microcontrolador se las imprime por la terminal serie incluida como periférico asociado.

```c
#include "xparameters.h"
#include "xil_io.h"
#include "encoder_ip.h"

#define PRESCALER 1440   // 1440 pulsos por revolucion

//====================================================

int main (void) {

    int posicion, direccion;
    char *sentido[2]={"antihorario", "horario"};

    xil_printf("-- Microarquitecturas y Softcores - TP Final - Ignacio Majul --\r\n");

   while(1)
   {
        posicion = ENCODER_IP_mReadReg(XPAR_ENCODER_IP_S_AXI_BASEADDR, ENCODER_IP_S_AXI_SLV_REG0_OFFSET);
        direccion = ENCODER_IP_mReadReg(XPAR_ENCODER_IP_S_AXI_BASEADDR, ENCODER_IP_S_AXI_SLV_REG1_OFFSET);
        xil_printf("Avance relativo: %d revoluciones. Sentido: %s \r\n", posicion/PRESCALER, sentido[direccion]);
        usleep(50000);
   }

}
```

Figura 9. Código C

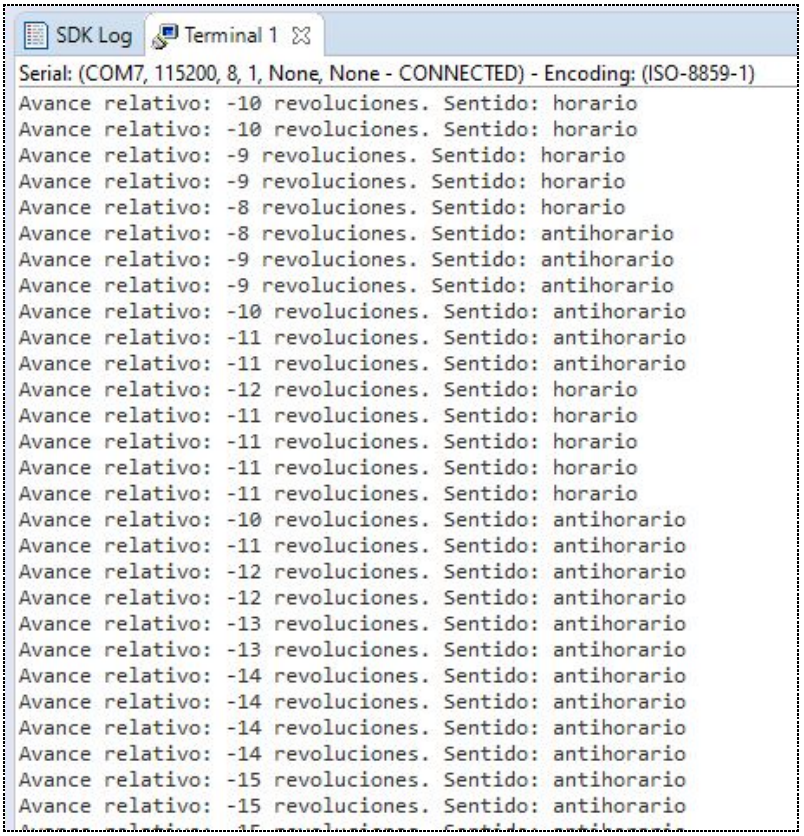


Figura 10. Reporte de la terminal.

3. Conclusiones

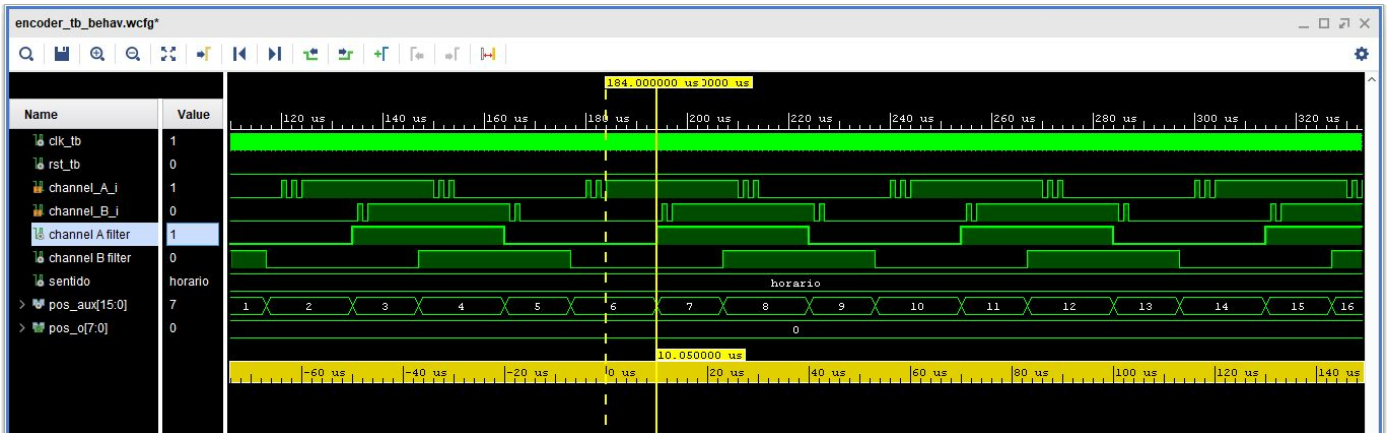
La implementación de la solución para manejo de encoders rápidos y de alta resolución resultó ser muy eficiente con el uso de lógica programable ya que logra una total independencia en el procesamiento de las señales de entrada y logra un eficaz filtro que evita errores de interpretación.

La incorporación de un IP propio y el empaquetado dentro de un sistema que incluye el microcontrolador permitió lograr independencia entre la lógica reconfigurable, que garantiza el funcionamiento correcto debido al paralelismo real de procesamiento, y la interpretación de las señales ya procesadas por las tareas secuenciales que realiza el código embebido en el microcontrolador.

El sistema compuesto por PL + PS presenta además claras ventajas frente a un sistema análogo ejecutado únicamente por un microcontrolador no sólo por lograr tratamiento de las señales de entrada en simultáneo sino porque evita la elevada cantidad de interrupciones molestas que debería manejar el microcontrolador para este mismo propósito.

Anexo I - Simulaciones

Verificación del cambio en el sentido de giro:



Verificación de desfasaje por filtrado de rebotes: