INSTITUTO TECNOLÓGICO DE COSTA RICA
ESCUELA DE INGENIERÍA ELECTRÓNICA
EL3313 TALLER DE DISEÑO DIGITAL
PROF. M.SC. KALEB ALFARO BADILLA
GRUPO 20
I SEMESTRE 2024

Laboratorio 2: Lógica Secuencial

PLANTEAMIENTO DE LA SOLUCIÓN

Realizado por:

Anthony Artavia Salazar
Yailyn Priscilla Campos Zamora
Samuel Montenegro Gómez

07/03/2024

Índice

1. EJERCICIO 1

A partir de [1], la frecuencia interna del reloj de entrada de la FPGA a utilizar, Nexys 4 DDR, es de 100 MHz.

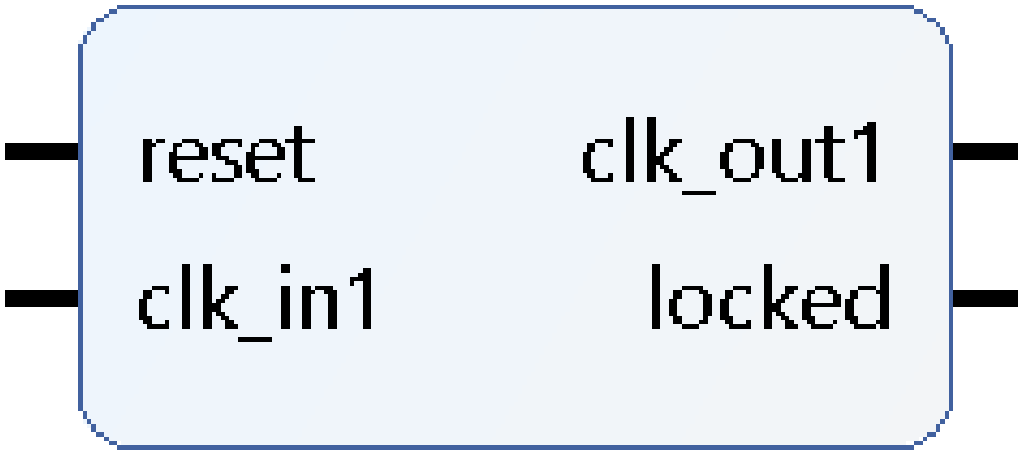

Figura 1.1: diagrama de bloques para ejercicio 1.

Es importante mencionar que la entrada clk_in1 hace referencia a lo que sería el reloj de entrada de la FPGA, el cual corresponde a una frecuencia de 100MHz, el reloj de salida a obtener deberá ser de 10MHz, la señal reset va funcionar como una señal de control y locked es parte de la sintáxis de creación de esete IP-CORE

2. EJERCICIO 2

Para solucionar el "problema"de rebotes de señales, es necesario implementar 2 flip flops, los cuales serán útiles para evitar que esto suceda además de ser parte de la sincronización.

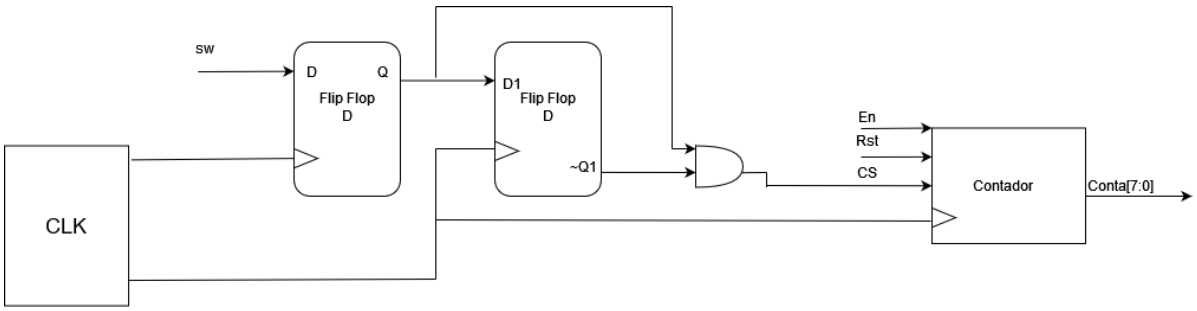

Figura 2.1: Diagrama de bloques para el sistema del ejercicio 2.

Se está tomando como señal de entrada un switch, aunque el concepto de implementación para un pushbutton es muy similar

3. EJERCICIO 3

La solución requiere implementar los bloques contador de 2 bits, divisor de reloj, codificador de tecla y sincronizador y antirrebote de la siguiente figura:

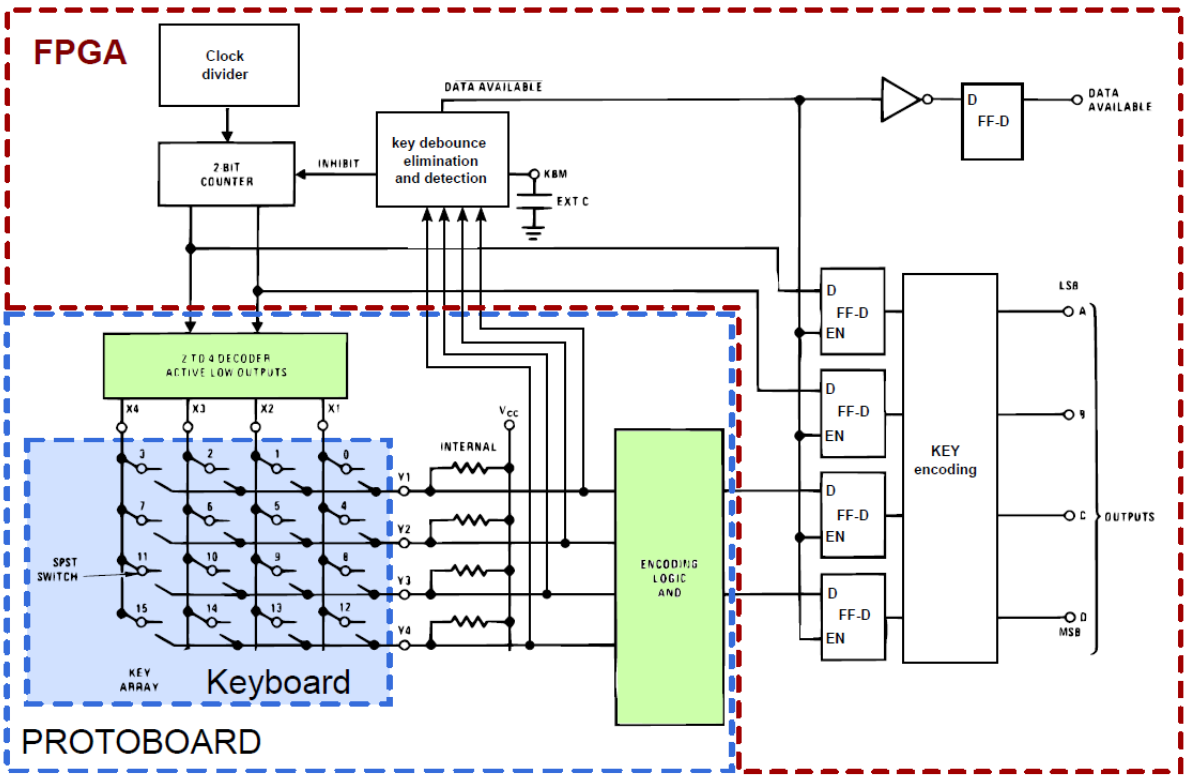

Figura 3.1: Diagrama de bloques para el sistema del ejercicio 3.

Para llevar a cabo lo mencionado, se hará el diseño de cada bloque por medio de tablas de verdad y diagramas.

Contador de 2 bits

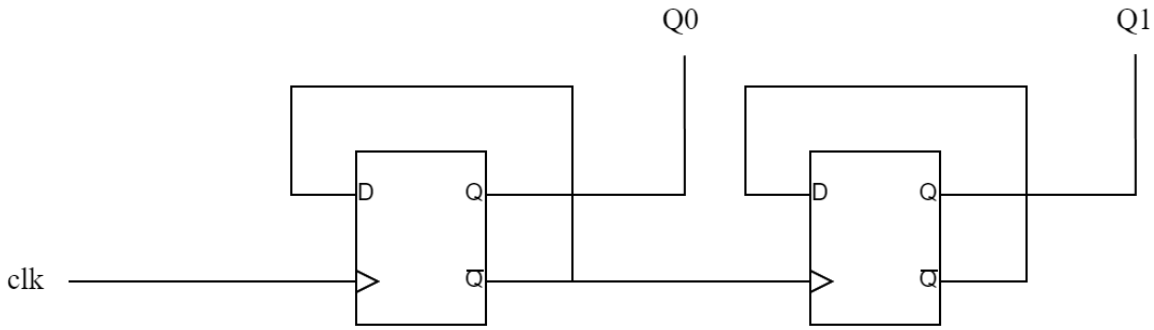

Figura 3.2: Circuito para un contador de 2 bits.

A partir del diagrama de la figura 3.2, se realizó lo siguiente:

Tabla 3.1: Tabla de verdad para el contador de 2 bits.

Ciclo de reloj	Q0	Q1
0	0	0
1	0	1
2	1	0
3	1	1
4	0	0
5	0	1
6	1	0
7	1	1
8	0	0

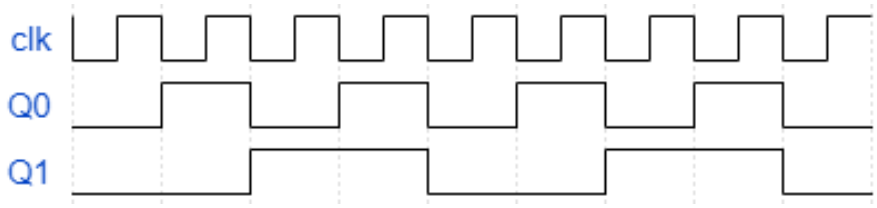

Figura 3.3: Diagrama temporal para un contador de 2 bits.

Divisor de reloj

El circuito resulta bastante similar al empleado para el contador de 2 bits, sin embargo, la cantidad de flip-flops que se coloquen depende de lo mucho que se desee disminuir la frecuencia de reloj, ya que por cada flip-flop que se coloque, se disminuirá la frecuencia a la mitad. Dicho esto, para un solo -flip-flop se tendría lo siguiente:

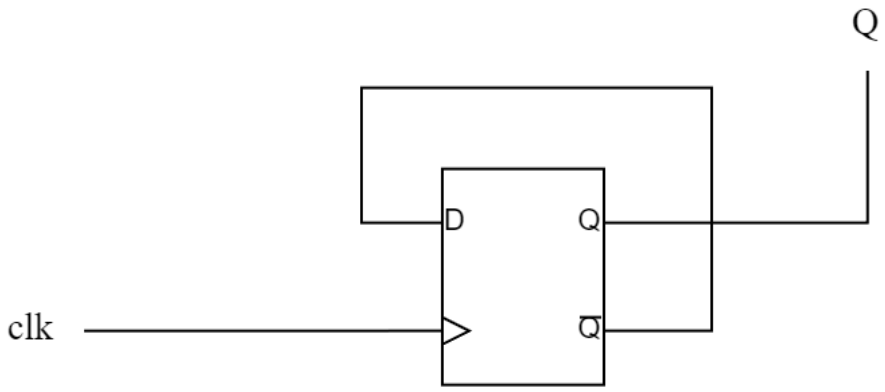

Figura 3.4: Circuito para un divisor de reloj.

Tabla 3.2: Tabla de verdad para el divisor de reloj.

Ciclo de reloj	Q
0	0
1	0
2	1
3	1
4	0

Figura 3.5: Diagrama temporal para un divisor de reloj.

Sincronizador y antirrebote
Para este bloque se van a poseer 4 circuitos similares al de la siguiente figura:

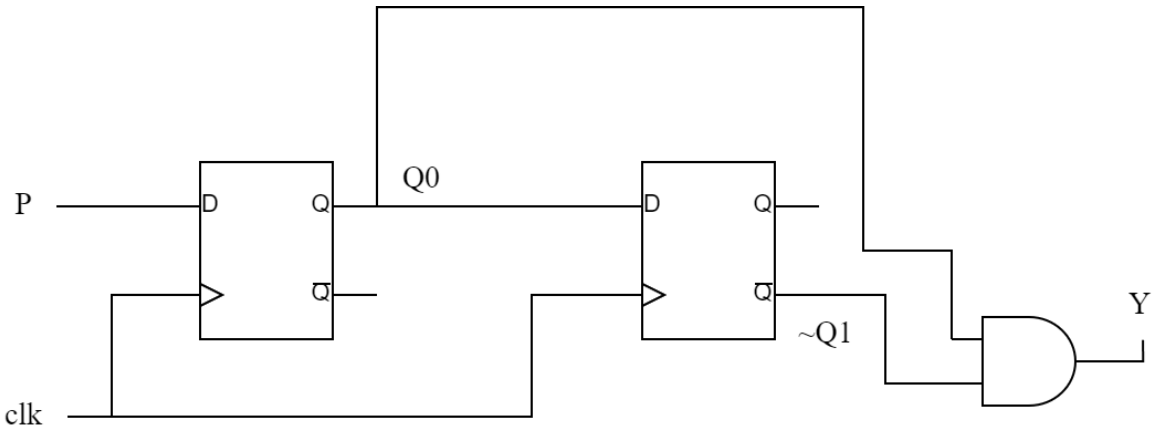

Figura 3.6: Circuito para un antirrebote.

Lo anterior se debe a que se van a detectar 4 posibles pulsaciones provenientes de los botones. Dicho esto, se tendría lo de a continuación

Tabla 3.3: Tabla de verdad para el antirrebote.

Ciclo de reloj	P	Q0	Q1	Y
0	0	0	1	0
1	X	0	1	0
2	1	1	1	1
3	1	1	0	0
4	1	1	0	0
5	X	1	0	0
6	0	0	0	0
7	0	0	1	0
8	0	0	1	0

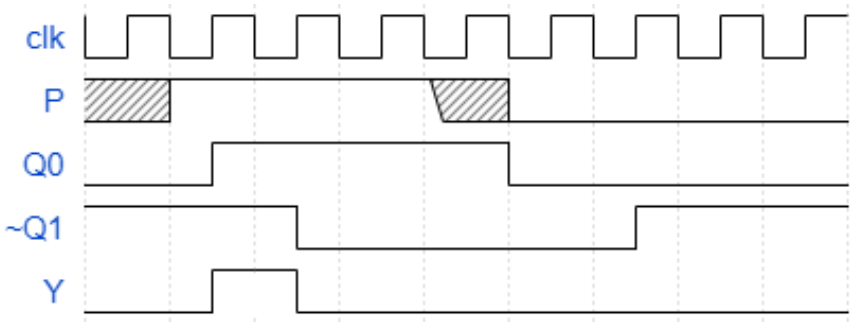

Figura 3.7: Diagrama temporal para un antirrebote.

De lo realizado, cabe mencionar que P corresponde al botón que se presiona y Y a la señal de un pulso que se detecta por la acción de este una vez que ha pasado por el antirrebote.

4. EJERCICIO 4

Para solucionar este ejercicio, se deben desarrollar los siguientes módulos:

- **random_tester** para generar un número aleatorio de 16 así como una señal variable de 1 bit.
- **register_16b** para lectura y escritura de un dato de 16 bits.
- **seven_segment_display_decoder** para decodificar un número de 16 bits a su representación en 4 displays de 7 segmentos, uno para cada grupo de 4 bits (dígito hexadecimal).

Los módulos anteriores se van a interconectar dentro del módulo superior **hex_to_seven_segment** como se indica en la figura 4.1.

El módulo **random_tester** consiste en un LFSR (*Linear Feedback Shift Register*) adaptado de [2], el cual cuenta con la salida extra WE (*write reg*) que alterna entre 0 y 1 por medio de un inversor con el fin de estimular la señal de escritura del módulo **register_16b**.

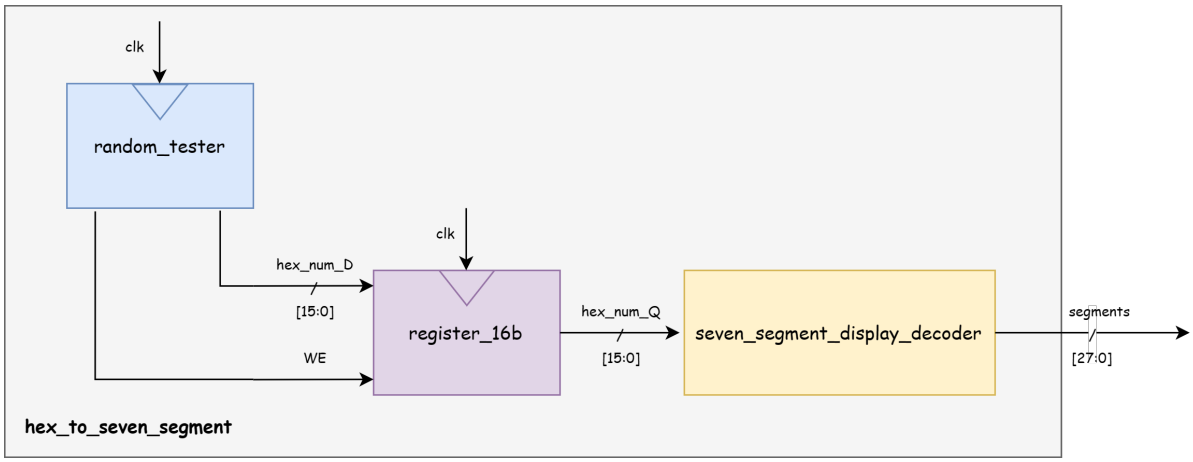

Figura 4.1: Diagrama de bloques para el sistema del ejercicio 4.

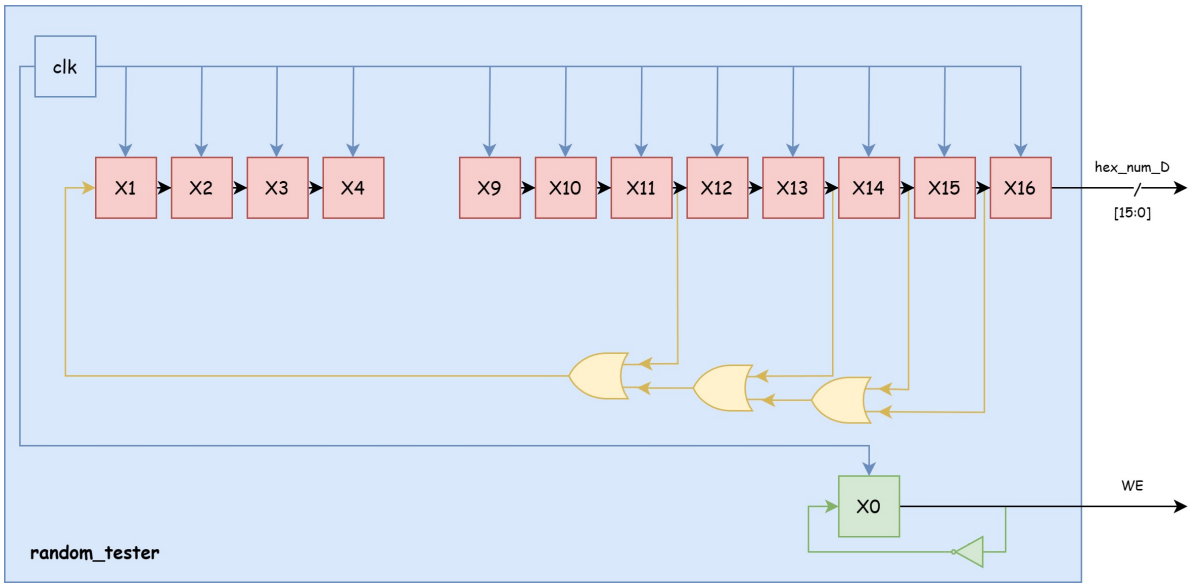

Figura 4.2: Estructura interna del bloque **random_tester** adaptado de [2].

5. EJERCICIO 5

Para la creación de un PC Counter según las especificaciones del enunciado es necesario desarrollar la siguiente estructura

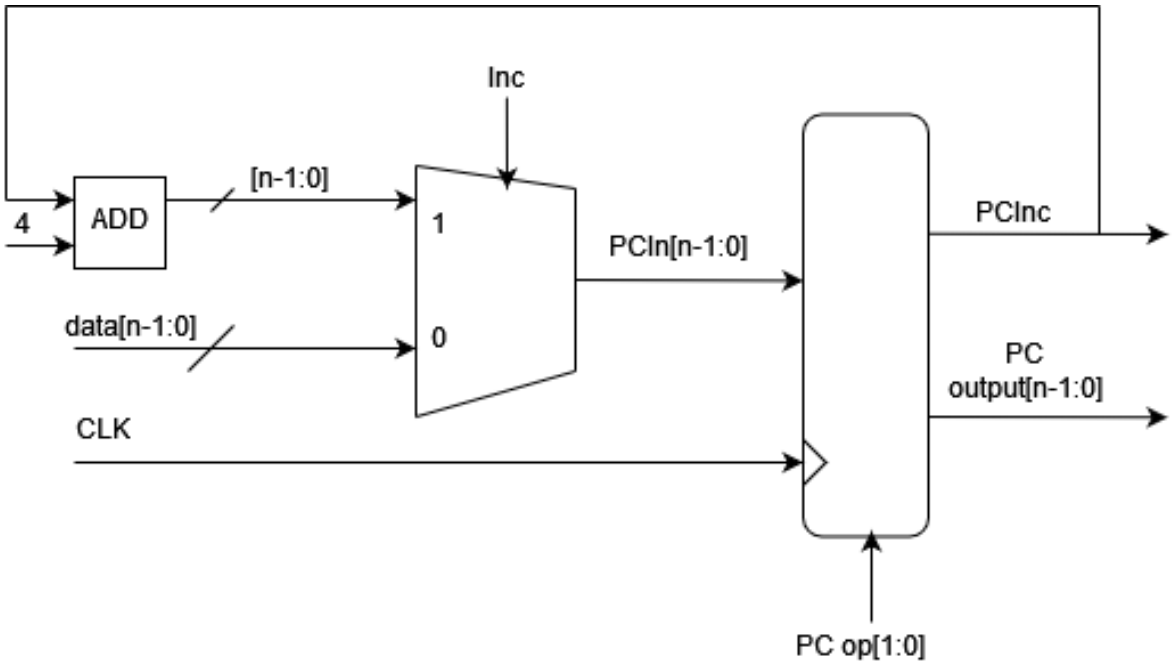

Figura 5.1: Estructura interna del bloque PC Counter

Tanto Inc como Pcop son señales de control. Se desarrollará un mux para controlar la dirección de entrada de lo datos, conociendo si la dirección guardada es la actual, cuyo usuario busca acceder, o si necesita seguir buscando la dirección por medio de sumador+4. Una vez localizada la entrada, el módulo PC Counter obtendrá en su salida la

6. EJERCICIO 6

Para solucionar este ejercicio se van desarrollar los siguientes módulos parametrizados:

- **demux** para demultiplexar un dato de entrada hacia alguno de los registros.
- **mux** para multiplexar el dato contenido en algún registro hacia una salida de lectura.

Los módulos anteriores se van a interconectar dentro del módulo superior, que también es parametrizado, **register_file** como se indica en la figura 6.1. Donde n permite obtener el número de registros del banco como 2^n registros, y w es el ancho de los datos en bits.

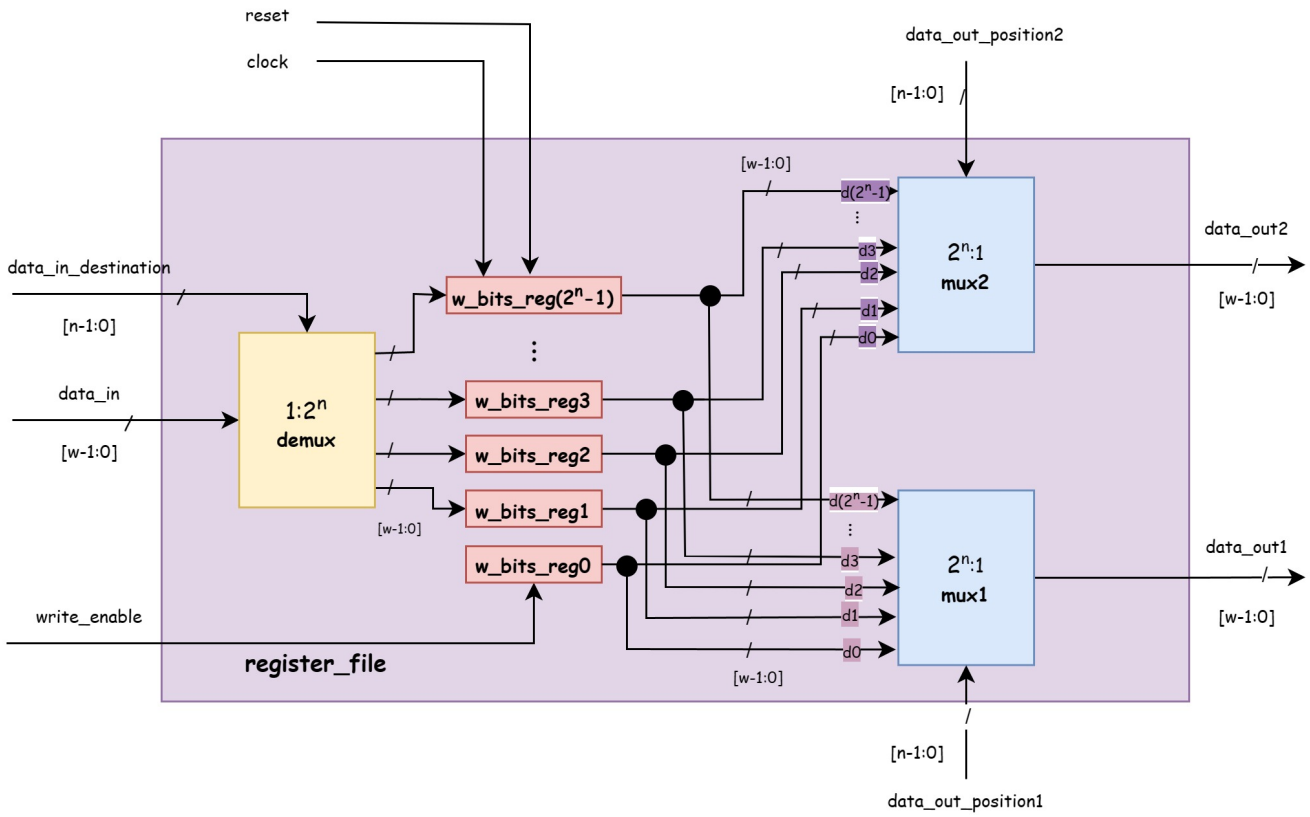

Figura 6.1: Diagrama de bloques para el diseño del banco de registros del ejercicio 6.

7. EJERCICIO 7

Para solucionar este ejercicio, es necesario:

- Diseñar una máquina de estados finitos con dos modos para el control del sistema.
- Crear un módulo superior en el que se interconecten los módulos previamente hechos: banco de registros (ejercicio 6), ALU (ejercicio 6, laboratorio 1), decodificador para display de siete segmentos (ejercicio 4) e interfaz para teclado (ejercicios 2 y 3).

De manera general, el algoritmo que manipula el sistema puede ser descrito por medio del siguiente diagrama de flujo:

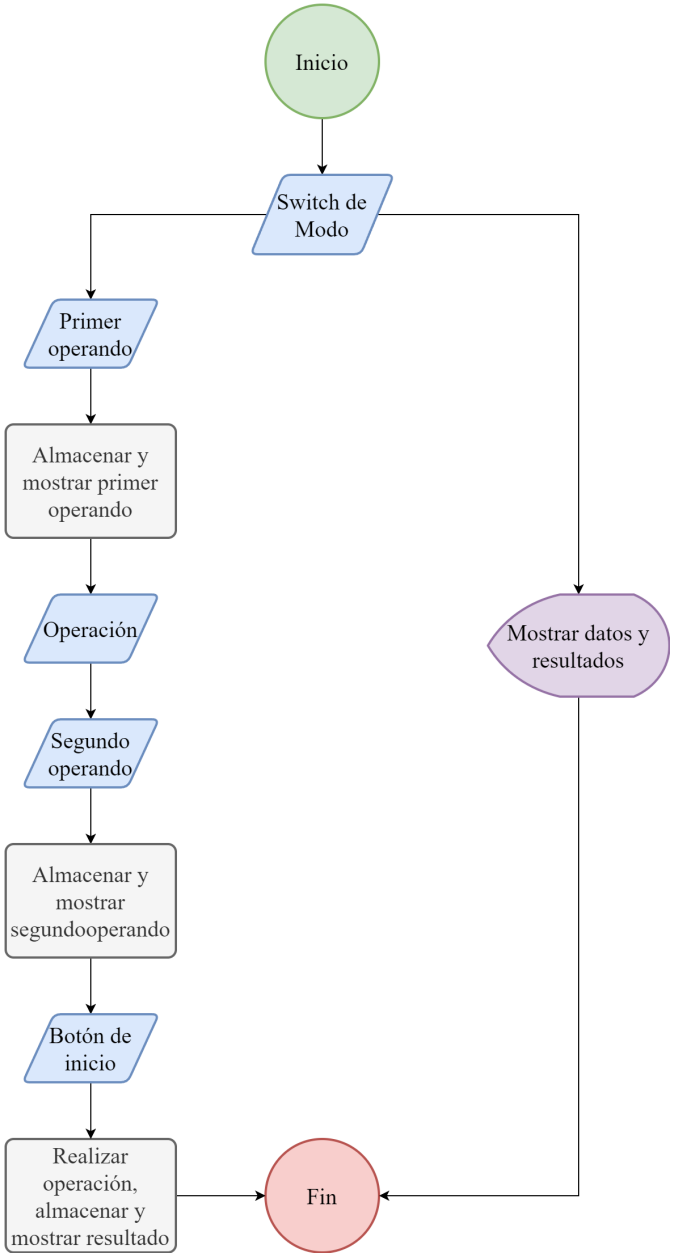

Figura 7.1: Diagrama de flujo del ejercicio 7.

Para la unidad de control del sistema se diseña una máquina de estados finitos de Moore con los dos modos solicitados. En la figura 7.2 se muestra el diagrama de transición de estados para el modo 1 y en la figura 7.3 el diagrama correspondiente para el modo 2. Estos diagramas se realizaron a un nivel general, se irán detallando conforme avance el laboratorio. Cada uno de estos modos se inicia al presionar un *switch* asignado, en cualquier momento.

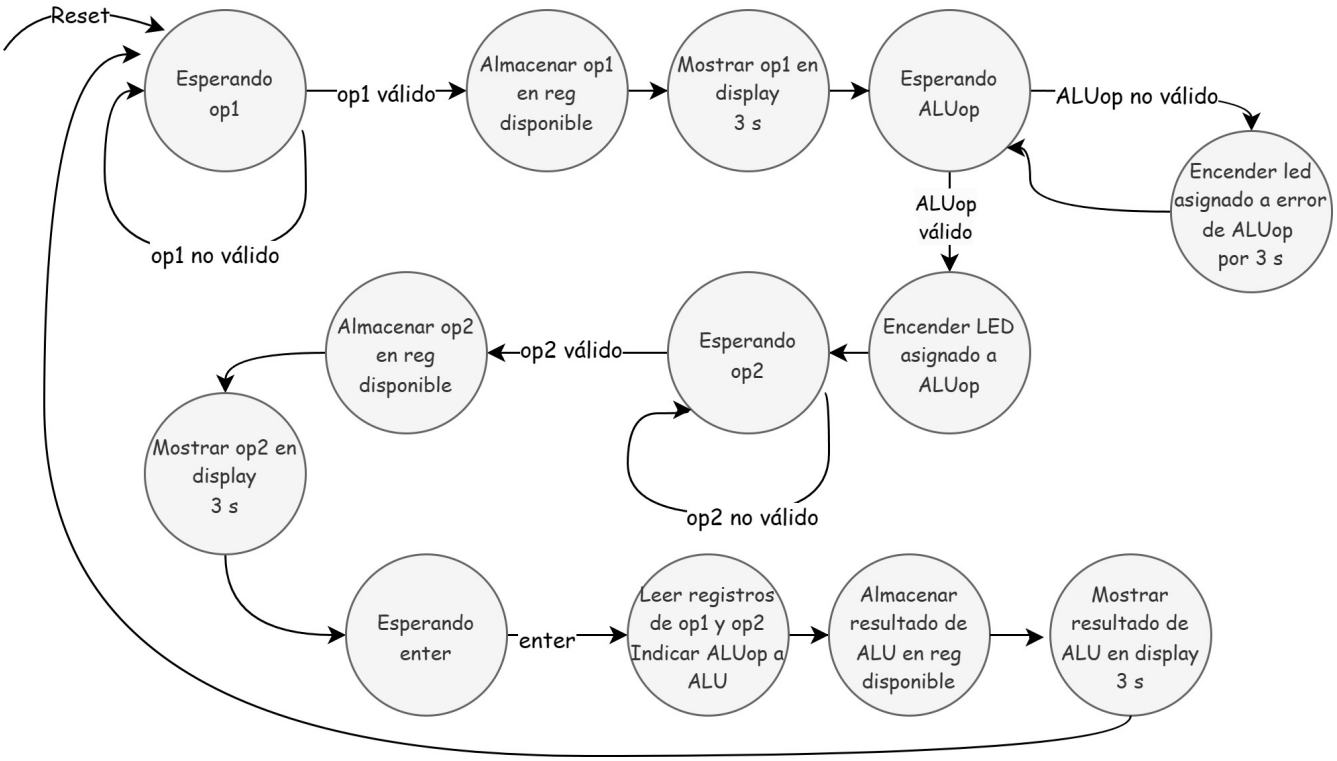

Figura 7.2: Diagrama de transición de estados para el modo 1 de la FSM de Moore, de la mini unidad de cálculo del ejercicio 7.

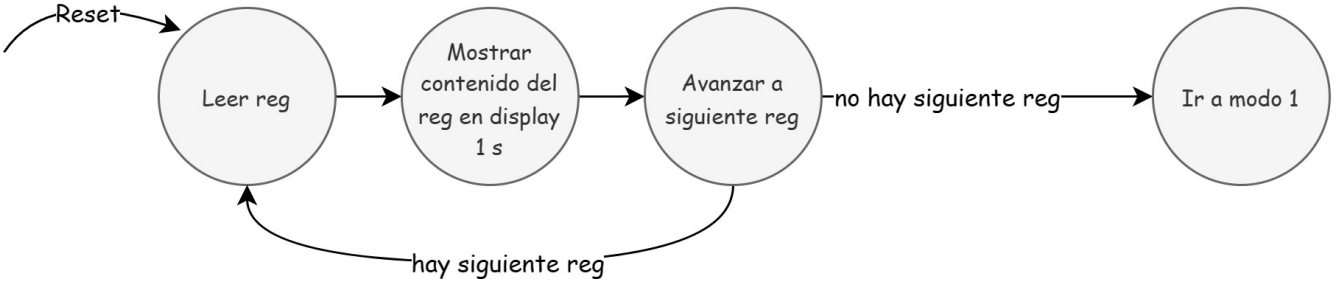

Figura 7.3: Diagrama de transición de estados para el modo 2 de la FSM de Moore, de la mini unidad de cálculo del ejercicio 7.

REFERENCIAS

[1] Digilent, "Nexys 4 DDR Reference Manual". [En línea]. Disponible: `https://digilent.com/reference/programmable-logic/nexys-4-ddr/reference-manual`.

[2] S. Hathwalia and M. Yadav, "Design and Analysis of a 32 Bit Linear Feedback Shift Register Using VHDL," Int. Journal of Engineering Research and Applications, vol. 4, no. 6, pp. 99–102, Jun-2014.