Escuela de Ingeniería Electrónica
Licenciatura en Ingeniería Electrónica
EL3313 Taller de Diseño Digital
II Semestre 2021

TEC | Tecnológico de Costa Rica

Informe de Laboratorio

Lab. 4: Interfaces con periféricos

Bosco Richmond, Francis Guindon, Christian Arguedas

Resumen

En este documento se implementa en la fpga *basys 3*, el desarrollo de sistemas computacionales con interacciones de dispositivos de entradas y salidas como botones, interruptores, display de 7 segmentos y la implementación del PMOD LCD, tomando en cuenta la frecuencia de 100 MHz de trabajo y rebote presente en los botones y problemas de sincronización en los interruptores del *basys 3*. Resolviendo dichos problemas con circuitos digitales con el fin de permitir el funcionamiento correcto del *basys 3* al interactuar con sus salidas y entradas.

1. Cuestionario previo

1. Revise la documentación para el PMOD - CLP, la cual se puede encontrar en `https://digilent.com/reference/pmod/pmodclp/start`. En esta página encontrará información acerca del módulo y algunos ejemplos.

   El PMOD - CLP es una pantalla de cristal líquido (LCD) de 16 caracteres por fila y de dos filas. Utiliza un interfaz de 8 bits paralelos. Tiene alrededor de 200 opciones de caracteres. La pantalla opera a $3{,}3\,V$ voltios y cuenta con 18 pines. Los pines tienen la especificación mostrada en la Tabla 1.

   En el enlace anterior se puede encontrar una lista de los comandos principales para operar con el display LCD. Los comandos principales involucran configurar el comportamiento del cursor, de la posición de escritura en la pantalla, la escritura y lectura de caracteres a la pantalla y la limpieza de la pantalla.

Tabla 1: Especificación SPI expandida

Pmod CLP Connector Signals					
Header J1 - Top Half			Header J1 - Bottom Half		
Pin	Signal	Description	Pin	Signal	Description
1	DB0	Data Bit 0	7	DB4	Data Bit 4
2	DB1	Data Bit 1	8	DB5	Data Bit 5
3	DB2	Data Bit 2	9	DB6	Data Bit 6
4	DB3	Data Bit 3	10	DB7	Data Bit 7
5	GND	Power Supply Ground	11	GND	Power Supply Ground
6	VCC	Positive Power Supply	12	VCC	Positive Power Supply
Header J2					
Pin	Signal	Description			
1	RS	Register Select: High for Data Transfer, Low for Instruction Transfer			
2	R/W	Read/Write signal: High for Read mode, Low for Write mode			
3	E	Read/Write Enable: High for Read, falling edge writes data			
4	NC	Optional back-light enable (not connected on the Pmod CLP)			
5	GND	Power Supply Ground			
6	VCC	Positive Power Supply			

2. Investigue sobre el mapeo de periféricos en memoria.

   E/S mapeado en memoria complementa tanto la relación de entrada y salida entre la unidad de procesamiento y los periféricos del ordenador abordando tanto la memoria y los dispositivos E/S. Como la memoria y los registros de los dispositivos E/S están asociados a direcciones lo cual puede referirse a posiciones de la memoria RAM o a la memoria del dispositivo E/S lo cual permite que dichos dispositivos monitorean el bus de direcciones de la CPU ante cualquier acceso a un dispositivo asignado conectando el bus de datos al registro del hardware. Para lograr esto se deben reservar las direcciones para los dispositivos y que no puedan ser accesados por la memoria física tanto de manera temporal como permanente [1].

3. Investigue sobre la asignación de pines en *Vivado* por medio del archivo de *constraints*.

   Herramientas de síntesis, como Vivado, permiten definir la asignación de puertos físicos de un FPGA al módulo escrito en un HDL. Esta asignación se logra mediante un archivo de texto de "contraints" con extensión ".xdc". En general los archivo de "contraints" pueden ser de tipo cronométricos, donde definen los relojes que se utilizan en el diseño o de tipo físicos, donde definen el mapeo de puertos del diseño a puertos del FGPA. El archivo de constraints físicos contiene comandos de terminal Tcl del siguiente tipo:

```
set_property PACKAGE_PIN U17 [get_ports btnd]
set_property IOSTANDARD LVCMOS33 [get_ports btnd]
```

   Aquí, `U17` se refiere a un puerto del FPGA específico. Para el Basys 3, este puerto se refiere al botón inferior. `btnd` se refiere al nombre del puerto en el módulo escrito en el HDL. El primer comando se encarga de la asignación del puerto mientras que el segundo comando

se encarga de definir el estándar de entrada/salida como LVCMOS33 (se refiere a que el FPGA opera con tecnología CMOS de bajo voltage a 3.3 V).

4. Investigue sobre las mejores practicas para la asignación de relojes y división de frecuencia en FPGAs.

   Para la asignación de relojes en FPGAs existen diferentes diferentes metodos, uno de los más recomendados es utilizar un modulo en el cual se realice la división de la frecuencia, el cual recibe como entrada el reloj del sistema y como salida se tendrá una fracción del reloj del sistema según la expresión (1), donde N representa la cantidad de veces que se dividirá la frecuencia [2].

$$N = \frac{f_{in}}{f_{out}} \tag{1}$$

   Estas divisiones de frecuencia se realizan por medio de contadores y cada vez que el conteo llegue al valor de N se activará la salida. En la Figura 1 se observa un modulo en Verilog donde se implementa el principio mencionado.

```
1  module clk_even_div(
2                          input                   clk,
3                          input                   reset,
4                          output reg              clk_div=0
5                                                                );
6  reg [2:0]cnt_div=0;
7  // Realizar la función de conteo del contador
8  always @(posedge clk)
9          begin
10                 if(reset)begin
11                         cnt_div<=3'd0;
12                         clk_div<=1'b0;
13                 end
14                 else if(cnt_div==3'd6)begin
15                         cnt_div<=3'd0;
16                         clk_div<=~clk_div;
17                 end
18                 else begin
19                         cnt_div<=cnt_div+1'b1;
20                 end
21         end
22 endmodule
```

Figura 1: Circuito de división de reloj [2]

5. Investigue sobre el fenómeno de rebotes y ruido en pulsadores e interruptores. Además, investigue sobre técnicas digitales (circuitos) para eliminar este fenómeno.

   El efecto rebote es cuando al presionarse un botón la señal proveída no es del todo limpia generando errores especialmente en el ámbito digital, donde un switch con un pull-up resistivo muestra bastantes errores al presionarse y del mismo modo al soltarse [3].

   Para mitigar el efecto rebote de manera digital, la idea básica es muestrear la señal del interruptor a intervalos regulares y filtrar los errores, un ejemplo de circuito anti rebote digital es el que se muestrea en la Figura 2 donde se pueden observar tres partes principales las cuales son:

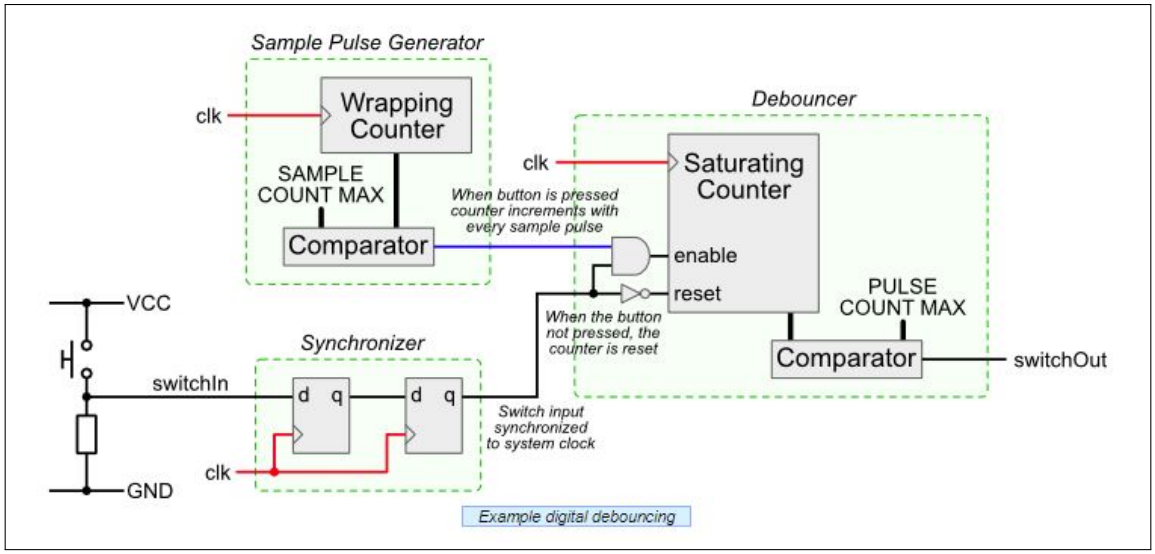

Figura 2: Circuito anti rebote digital [3]

- **Generador de pulsos de muestra** Consiste en un contador que incrementa cada pulso de reloj y cuando alcanza el máximo realiza el cambio en la entrada y se resetea el contador.
- **Sincronizador** Este componente se asegura que la señal de cambio del interruptor esté sincronizada con el reloj del sistema, minimizando la posibilidad de metaestabilidad.
- **Anti rebotador** El anti rebotador es otro contador, que al alcanzar el máximo se resetea y realiza la conmutación de la salida.

6. Investigue sobre los problemas de metastabilidad cuando se tienen entradas asíncronas en circuitos digitales. Investigue sobre circuitos que permitan la sincronización de entradas como pulsadores e interruptores.

   La metaestabilidad es cuando la salida trata de seguir a la entrada pero debido a cambios repentinos o aleatorios, la salida no logra seguir el resultado y su comportamiento no llega del ser todo 1 o 0 como se ve en la Figura 3 Es decir, para entradas asíncrona, comúnmente, ante el reinicio asíncrono provoca desincronizacion entre el reloj de activación y la señal asíncrona; generando problemas de en el tiempo de espera resultando en metaestabilidad con configuraciones insatisfactorias en la retención de datos. La metaestabilidad puede provocar serios errores desde afectar la lógica del sistema como detenerse por completo

ante mutaciones y fallas en los datos. Para sus solución se propone la implementación de FIFO Figura4 y para los relojes asíncronos se usa el método de reinicio asíncrono y liberación síncrona Figura5.

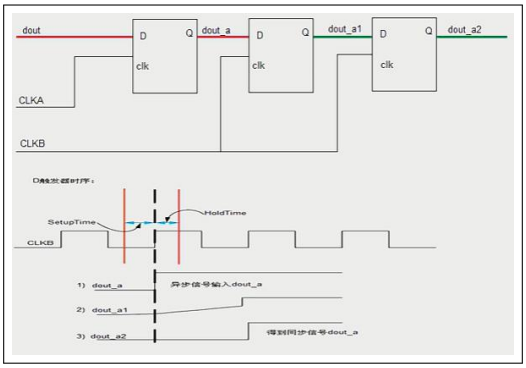

Figura 3: Ejemplo de metaestabilidad

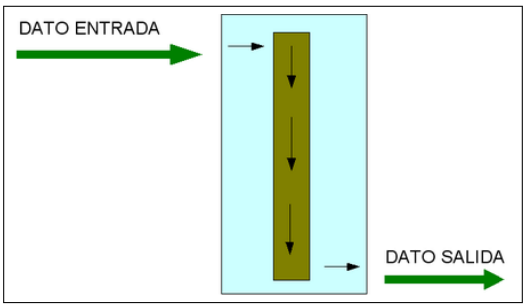

Figura 4: Circuito FIFO(First in - First out)

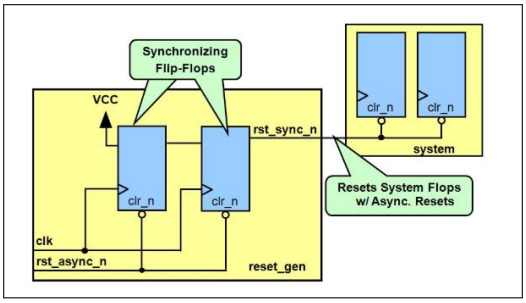

Figura 5: Método de reinicio asíncrono y liberación síncrona

2. Problema 1. Decodificador hex-to-7-segments

El primer problema de diseño en este laboratorio se trata de crear un módulo para el Basys 3 que generar valores pseudoaleatorios y que los muestre, en formato hexadecimal, en el display de siete segmentos que trae el Basys 3. El instructivo del proyecto [4] trae un diagrama de bloques básico para el módulo mostrado en la Figura 6.

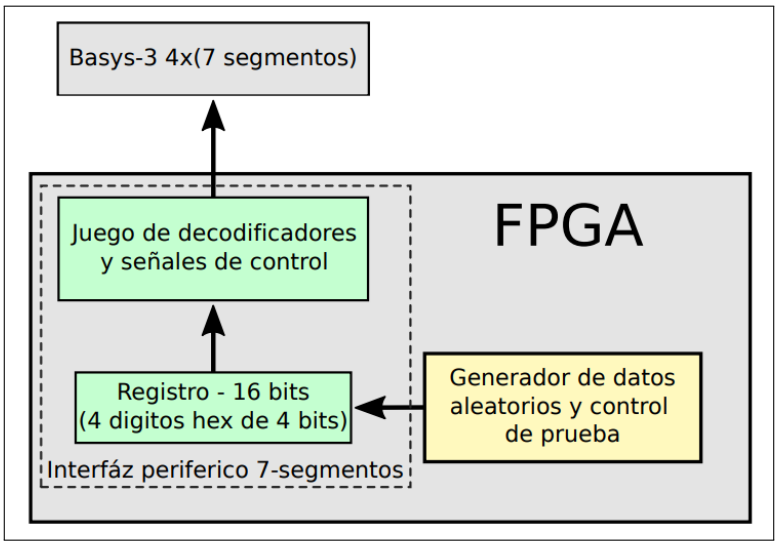

Figura 6: Diagrama de bloques básico para el experimento 1

El diagrama completo del diseño resultante de este decodificador se muestra en la Figura 7

El Contador de Delay Ajustable actuó como un divisor de frecuencia del reloj, donde incrementó el contador de salida cada cierta cantidad parametrizada de ciclos del reloj del Basys 3 (que opera a 10 MHz). Se utilizó el LFSR provisto en un laboratorio pasado por el profesor para generar los datos seudo aleatorios. Estos se almacenaron en el Registro PIPO (Parallel In Parallel Out) de 16 bits. A partir de ahí, un módulo Interfáz para Display de Siete Segmentos tomó el dato y lo decodificó a valores de 7 segmentos. Para lograr esto, se dividió los 16 bits del dato en 4 grupos de 4 bits, de tal manera que cada grupo representó un carácter hexadecimal. Cada grupo de 4 bits se decodificó a su representación de 7 segmentos. Luego un Contador con Delay Ajustable se utilizó para implementar un display rodante, donde se muestra un dígito a la vez en el display por un periodo imperceptible.

Una parte de los resultados de la simulación post implementación con auto chequeo se muestran en la Figura 8. La última columna muestra cómo se rueda el ánodo activo (los ánodos son activos bajos). Rodar en este contexto se refiere a activar los ánodos uno a la vez en rotación por un periodo específico. Además en la penúltima columna se observa cómo se modifica los puertos de siete segmentos cada vez que se activa el ánodo menos significativo. Esto es característico de la secuencia generada por un LFSR de 16 bits con una semilla de 0, donde los valores iniciales solo cambian los 4 bits menos significativos.

Se logró implementar el experimento 1 sobre el Basys 3 de tal manera que se observaba el cambio del display cada 2 segundos según el dato del LFSR. Se lograba observar 4 dígitos distintos en cada display, de tal manera que el display rodante logró cumplir su propósito.

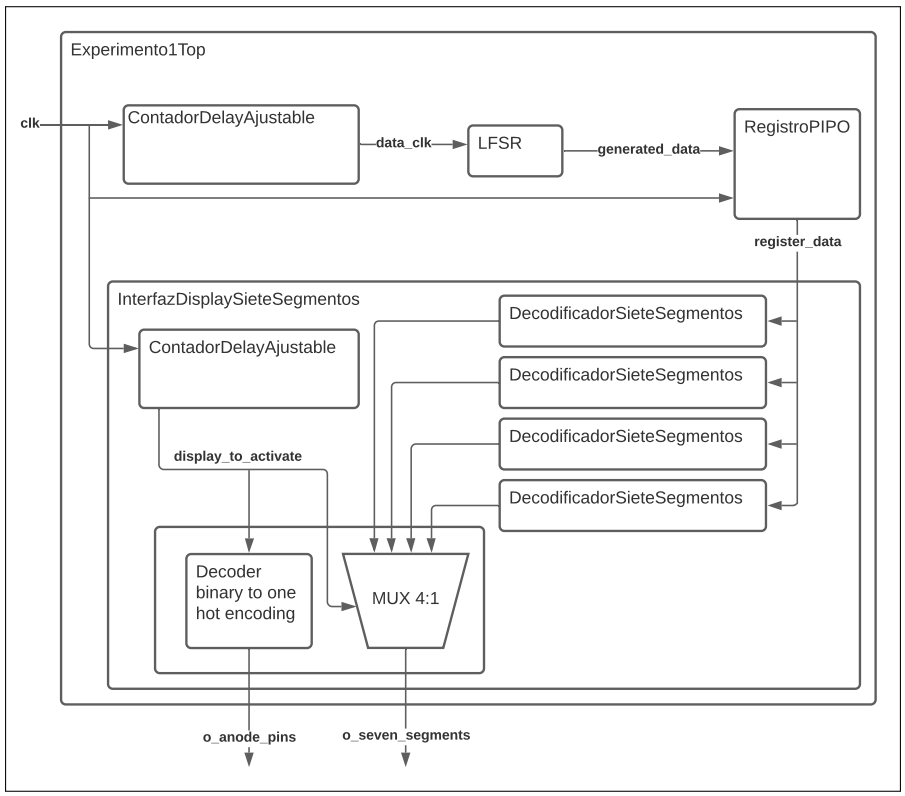

Figura 7: Diagrama de bloques completo para el experimento 1

```
1 | 00000011 1101 | 00000011 1101 |
1 | 00000011 1011 | 00000011 1011 |
1 | 00000011 0111 | 00000011 0111 |
1 | 10011111 1110 | 10011111 1110 |
1 | 00000011 1101 | 00000011 1101 |
1 | 00000011 1011 | 00000011 1011 |
1 | 00000011 0111 | 00000011 0111 |
1 | 00001101 1110 | 00001101 1110 |
1 | 00000011 1101 | 00000011 1101 |
1 | 00000011 1011 | 00000011 1011 |
1 | 00000011 0111 | 00000011 0111 |
1 | 00011111 1110 | 00011111 1110 |
```

Figura 8: Resultados de simulación post implementación con auto chequeo. La primera columna muestra valores de entrada para el reset. Las siguientes dos columnas muestran los valores esperados para los valores para los puertos siete segmentos y los puertos de ánodo de los displays. Las últimas dos columnas muestran los valores simulados para estos puertos

3. Problema 2. Diseño antirebotes y sincronizador

Se empleo el desarrollo del sistema de bloques con flipflops y un slowclock [3] de la figura 9, para el slowclock se utilizo un valor máximo de 18 750 000 de cuentas por cada ciclo de reloj

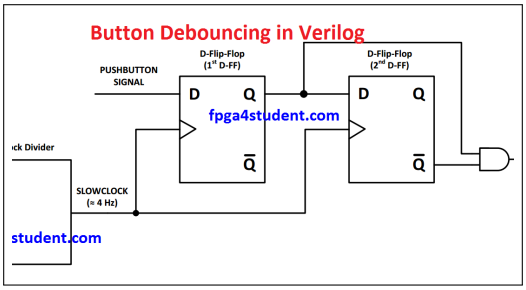

Figura 9: Diagramas de bloques implementado para el sistema antirebote

con el fin de bajar la frecuencia de 100 MHZ a 5.33 Hz aproximadamente eliminando parte del ruido posteriormente con los flipflops al tener que esperar cierto tiempo el alto para que entren en funcionamiento eliminando así el debouncing.

Empleando el contador de prueba proporcionado para este experimento, sumado a la implementación del display de la fpga, se logró implementar el incremento de cuenta en el display cada vez que se presionaba el botón, adicionando un boton de reset con el mismo sistema de debouncing para reiniciar la cuenta observable en el display solo cuando es presionado. El botón en caso de ser presionado por mucho tiempo lo cuenta como una única vez y en caso de apretarse muchas veces seguidas, cuenta únicamente las veces que fue presionado.

4. Problema 3. Interfaz para display de cristal líquido

Para la realización de este problema se analizó el problema y se elaboró el diagrama de bloques mostrado en la Figura 10. A partir del cual se pudieron realizar el generador de datos y pruebas, los registros y la lógica de la selección de las señales.

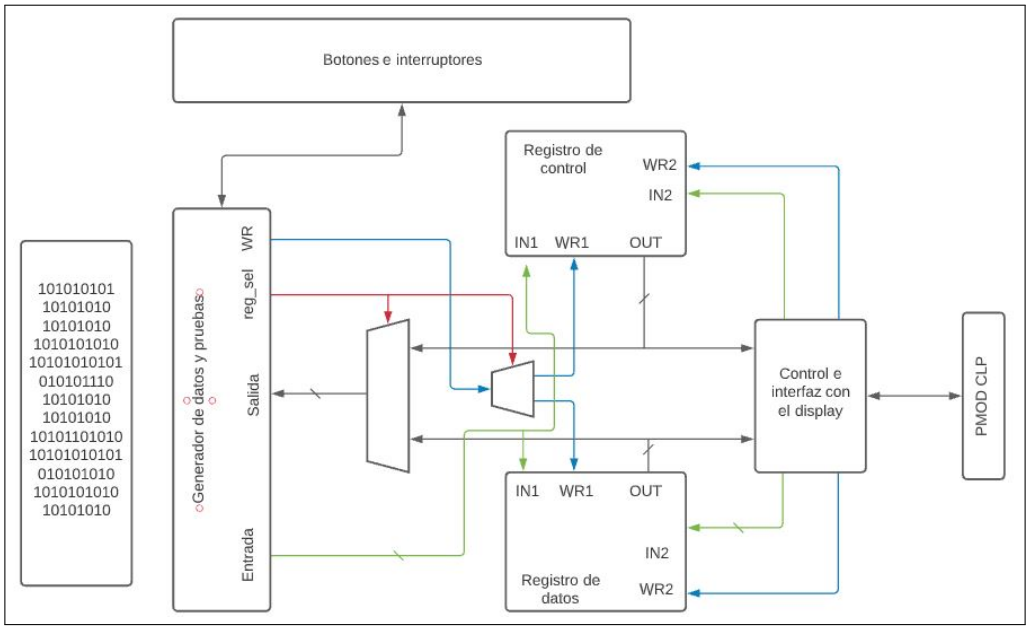

Figura 10: Diagrama de bloques para el experimento 3

Para el control e interfaz con el display se realizó el diagrama de bloques interno que se observa en la Figura 11. De este se obtuvieron los módulos de traductor, temporizador e inout.

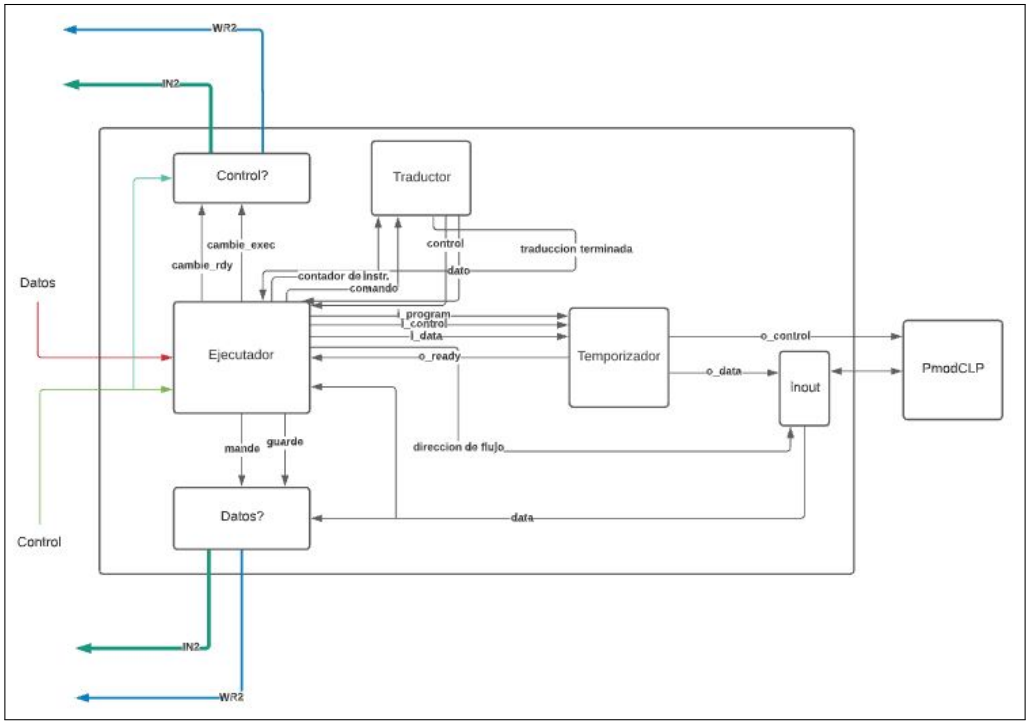

Figura 11: Diagrama de bloques para la interfaz con el display

Finalmente se realizó el diagrama de flujo mostrado en la Figura 12. a partir del cual se construyó el ejecutador.

5. Problema 4. Mini-calculadora

Para empezar el diseño de la mini calculadora se propuso el diagrama de bloques general que se muestra en la Figura 13, a partir del cual se obtuvo la interfaz entre los botones y la ALU, en este modulo se toman los números A y B de los interruptores y de los pulsadores se toma la operación deseada, el módulo se diseño de manera que se envíen los datos al seleccionar la operación, lo cual significa que se puede omitir el debouncing para los interruptores y aplicarlo solo a los pulsadores. Entre las salidas del modulo se encuentran los números A y B y el comando de instrucción para seleccionar la operación que realizará la ALU. Se debe mencionar que se hizo uso de la ALU realizada en el laboratorio 2.

Para realizar la interfaz con el display se planteo el diagrama de bloques que se observa en la Figura 14. Para este experimento se quizó trabajar con la interfaz del experimento 3, por lo que se planteo el modulo Traductor ALU a display, el cual recibe el numero a, b y el resultado, los cuales se deben convertir a BCD con el fin de mostrar el valor decimal en el display LCD, además de la operación, para determinar el simbolo a mostrar en la pantalla.

Para realizar el Traductor ALU a display se planteó el diagrama de flujo el cual también funciona como máquina de estados el cual se muestra en la Figura 15.

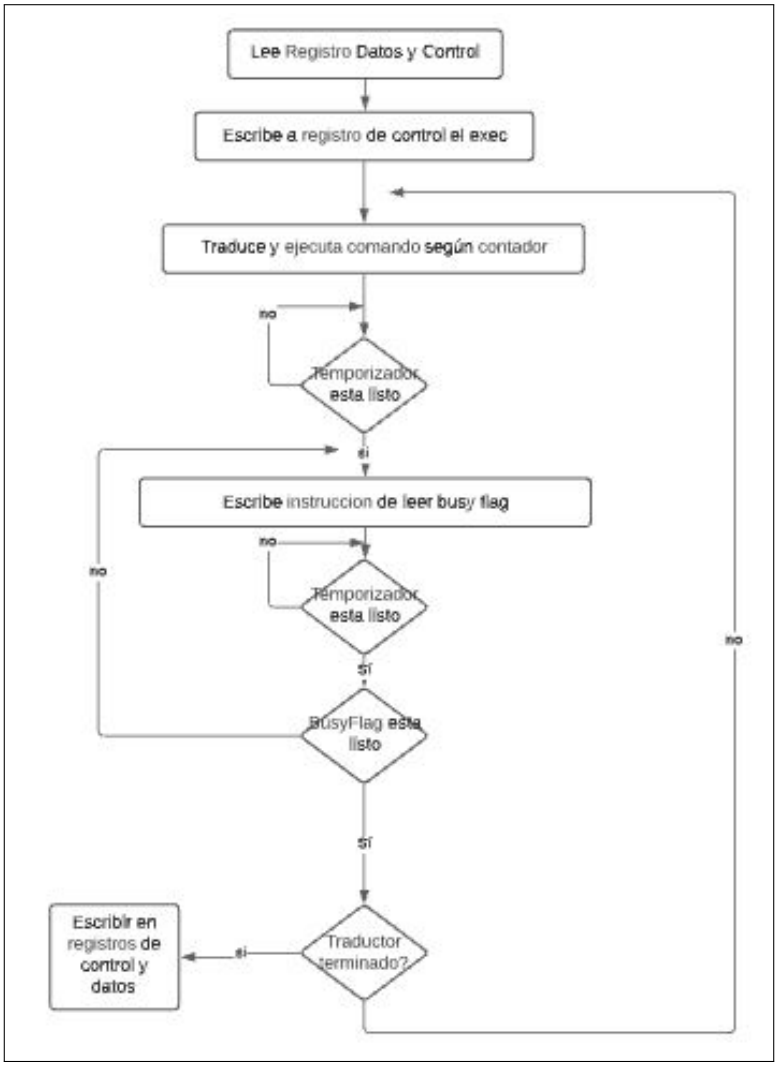

Figura 12: Diagrama de flujo del ejecutador

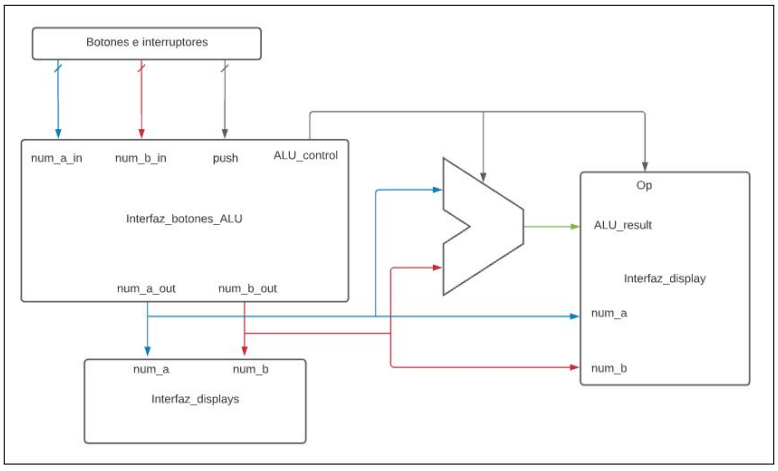

Figura 13: Diagrama de bloques del experimento 4

Este modulo se encarga de seleccionar la acción a realizar en el display según los datos recibidos,

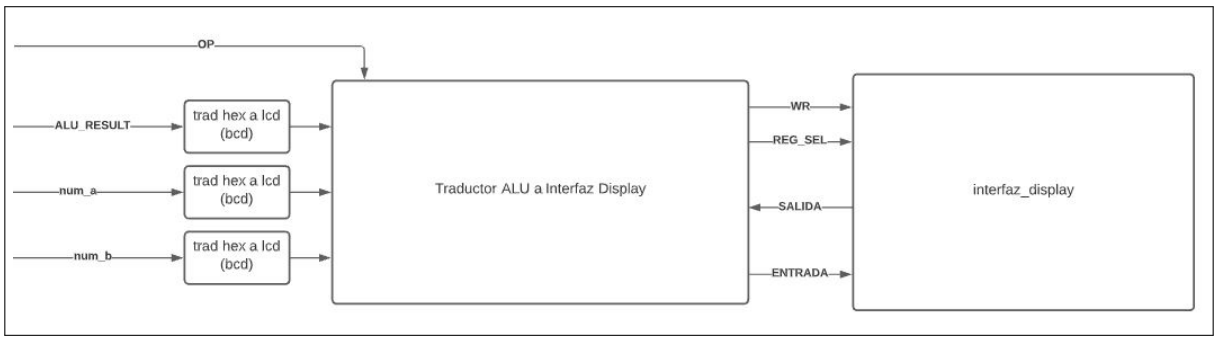

Figura 14: Diagrama de bloques del experimento 4

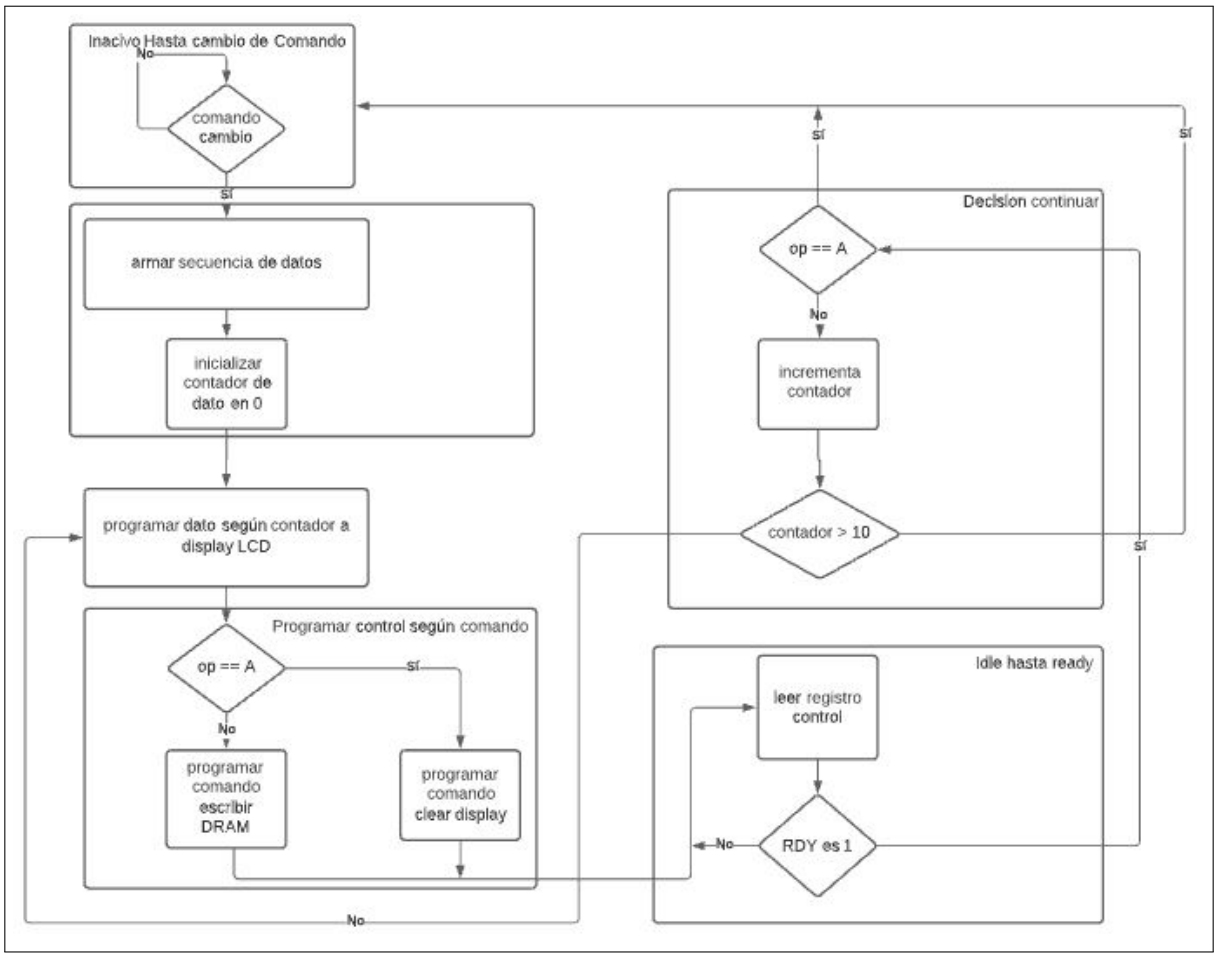

Figura 15: Diagrama de flujo/bloques del Traductor ALU a interfaz display

la operación A_H es una operación extra la cual no está dentro de los comandos de la ALU, por lo que no interfiere en la misma y corresponde al reset de la pantalla, por lo que ejecutará el comando Clear Display.

6. Análisis de resultados

Cada módulo del interfáz de siete segmentos logró correr exitosamente su testbench con autocheqeo de post síntesis y post implementación. Para estas simulaciones se ajustaba los parámetros de los Contadores con Delay Ajustable para lograr un cambio de display activo según el display rodante cada ciclo y un cambio del dato del LFSR cada 4 ciclos. En la práctica estos módulos lograron operar con los delays correspondientes de tal manera que los displays se activaban más rápido de lo percibido a simple vista y el dato del LFSR se cambiaba cada 2 segundos.

La implementación del circuito *debouncing* propuesto en la figura 9 en la fpga pasó las pruebas del testbench con un ruido introducido en el botón de prueba, siendo ignorado por el circuito y procesando el resultado como un único impulso como se esperaba; traducido en la práctica se logró de manera exitosa el conteo por parte del botón sin valores indeseados y la efectividad del reset para inicializar y reiniciar de manera adecuada el contador y el display de 7 segmentos; empleando el debouncing tanto en el botón de conteo como el botón de reset, logrando un conteo eficiente sin perturbaciones por el efecto rebote.

Al finalizar el problema 3, se confirmó el funcionamiento de los módulos de generación de datos y pruebas, registros y la lógica de selección, mediante testbenches de auto chequeo para cada módulo, con respecto al control e interfaz con el display se logró implementarlo en la FPGA, aunque se notaron algunos problemas al colocar los datos, los cuales pueden deberse a la temporización de este ultimo módulo o al ajuste del contador para el debouncing el cual puede generar problemas si es muy alto o muy bajo, para la implementación del anti rebote en este experimento se tomó en cuenta un botón de *send* para enviar los datos y el control de los botones e interruptores mediante un pulsador, por lo que en este caso solo se requiere utilizar debouncing para el pulsador.

Para el experimento 4 se realizaron todos los módulos necesarios, aunque no fue posible implementarlo por motivos de tiempo, la idea principal fue reutilizar la interfaz con el display del experimento anterior y acoplar los datos de este experimento a dicha interfaz, esto se hizo mediante un traductor el cual recibe los números a, b y el resultado traducidos a BCD y la operación, y se encarga de enviar los datos a la interfaz uno por uno, cada vez que la bandera de RDY esté en alto y de esta manera mostraría los datos necesarios al display, para este experimento se trató al pulsador de la operación como un botón de envío, lo cual no causaría conflicto al estar cambiando los interruptores ya que el programa solo realizaría la operación al presionar el respectivo botón y continuaría con la siguiente cuando haya finalizado la anterior.

7. Conclusiones

Tanto para los testbench para comprobar el comportamiento de los circuitos propuestos para la solución de cada problema y la implementación en la FPGA se debió tomar en cuenta los distintos ciclos de reloj de la herramienta de vivado y la frecuencia real del FPGA ajustando sus valores para poder observar adecuadamente sus valores en la simulación y la práctica siendo más lento en la simulación al trabajarse con un reloj simulado más lento que el del *basys 3*.

Se logró realizar la síntesis e implementación de todos los módulos propuestos en este laboratorio, pudiendo programarlos a la FPGA de manera exitosa, a excepción del experimento 4, aunque, de

la misma manera los módulos de este ultimo fueron realizados, sintetizados e implementados por separado.

Referencias

[1] clemson, *E / S - entrada / salida"(PDF)*, aug 2011.

[2] P. clic, "Diseño de divisor de frecuencia basado en fpga." [Online], 2020. https://programmerclick.com/article/30441050695/.

[3] A. Greensted, "Switch debouncing." [Online], jun 2010. http://www.labbookpages.co.uk/electronics/debounce.html.

[4] J. Castro-Godínez and P. M. Ponce, "Lab. 4: Interfaces con periféricos." [Online], 2021.