

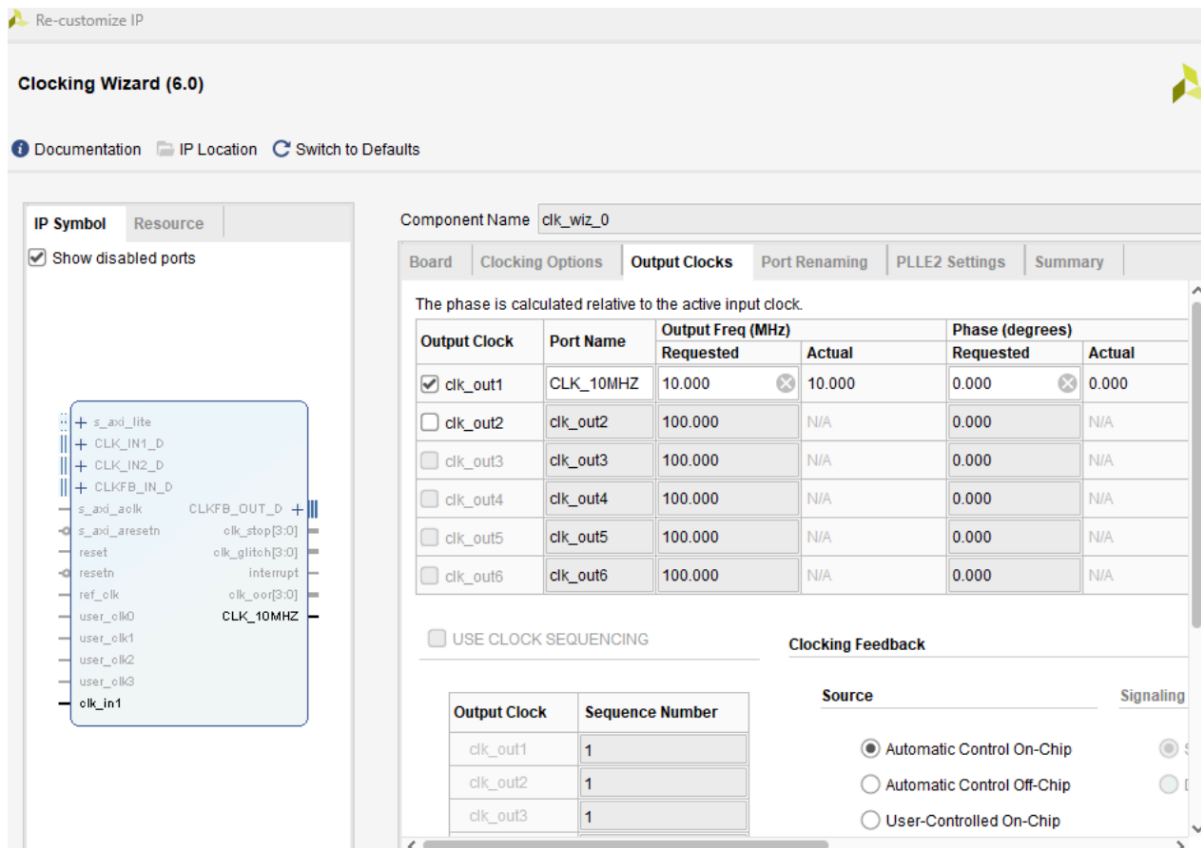
Laboratorio 2:
Lógica secuencial
Planteamiento

Profesor
Kaled Alfaro Bonilla

Estudiantes
Fiorela Chavarría Castrillo
William Sánchez Ramírez
Rolando Vega Marino

I Semestre 2024

Ejercicio 1: Uso del PLL IP-core. El reloj externo de 100 MHz puede ser configurado a través del constraints de la FPGA Para generar un reloj interno de 10 MHz se configura un bloque Clocking Wizard. A continuación, se describe cómo utilizar el IP-Core, Clocking Wizard, en Vivado, en la sección de “IP Catalog” en la lista de IP-Cores, Clocking Wizard se configuran los parámetros necesarios tanto de entradas como de salidas, 100MHz y 10Mhz respectivamente



Re-customize IP

Clocking Wizard (6.0)

Documentation IP Location Switch to Defaults

Component Name: clk_wiz_0

Board Cloning Options **Output Clocks** Port Renaming PLL2 Settings Summary

The phase is calculated relative to the active input clock.

Output Clock	Port Name	Output Freq (MHz)	Phase (degrees)
		Requested	Actual
<input checked="" type="checkbox"/> clk_out1	CLK_10MHZ	10.000	0.000
<input type="checkbox"/> clk_out2	clk_out2	100.000	N/A
<input type="checkbox"/> clk_out3	clk_out3	100.000	N/A
<input type="checkbox"/> clk_out4	clk_out4	100.000	N/A
<input type="checkbox"/> clk_out5	clk_out5	100.000	N/A
<input type="checkbox"/> clk_out6	clk_out6	100.000	N/A

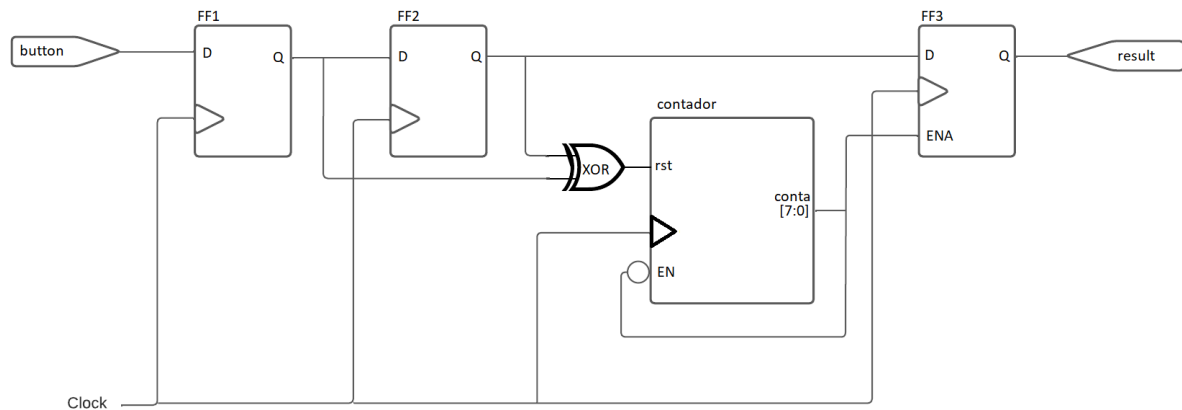
☐ USE CLOCK SEQUENCING

Clocking Feedback

Output Clock	Sequence Number	Source	Signaling
clk_out1	1	Automatic Control On-Chip	
clk_out2	1	Automatic Control Off-Chip	
clk_out3	1	User-Controlled On-Chip	

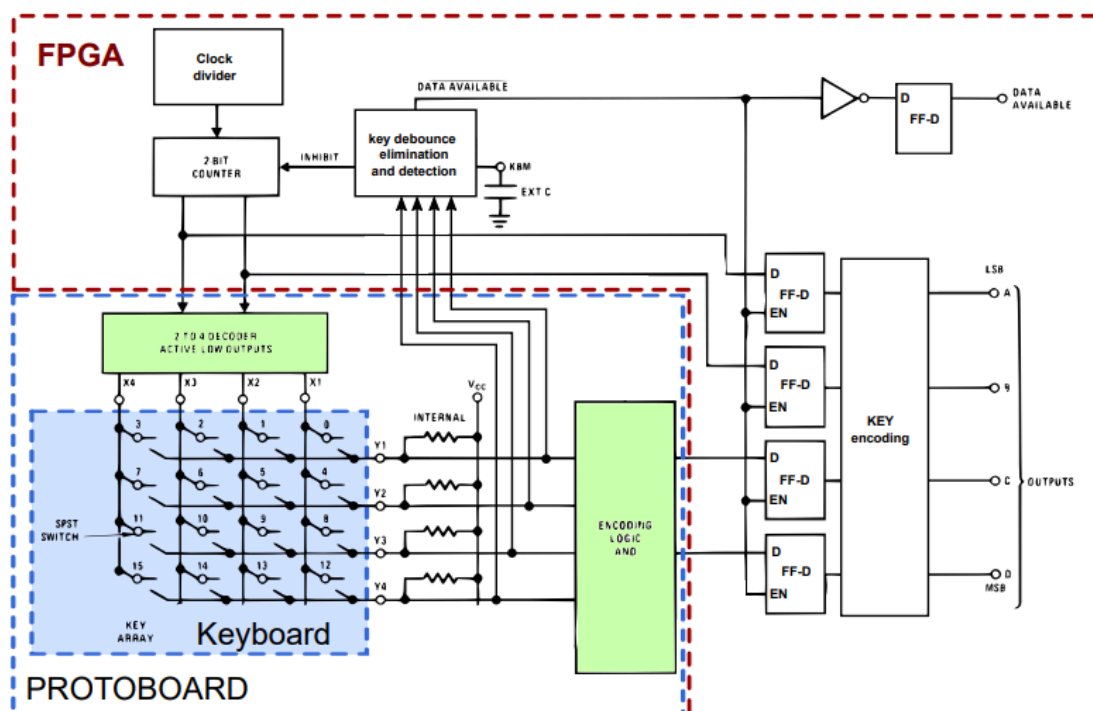
Ejercicio 2. Diseño antirebotes y sincronizador

Para este ejercicio se implementarán dos módulos de diseño: uno para el anti rebotes llamado “Debounce” y otro para el sincronizador llamado “Sync”, además se realizará un testbench donde se instancian las señales de ambos módulos de diseño llamado “tb_SyncDebounce”. El módulo Debounce filtra el ruido o fluctuaciones de la señal del botón para así proporcionar una salida estable. El módulo Sync sincroniza la señal del botón asincrónica con un clock.



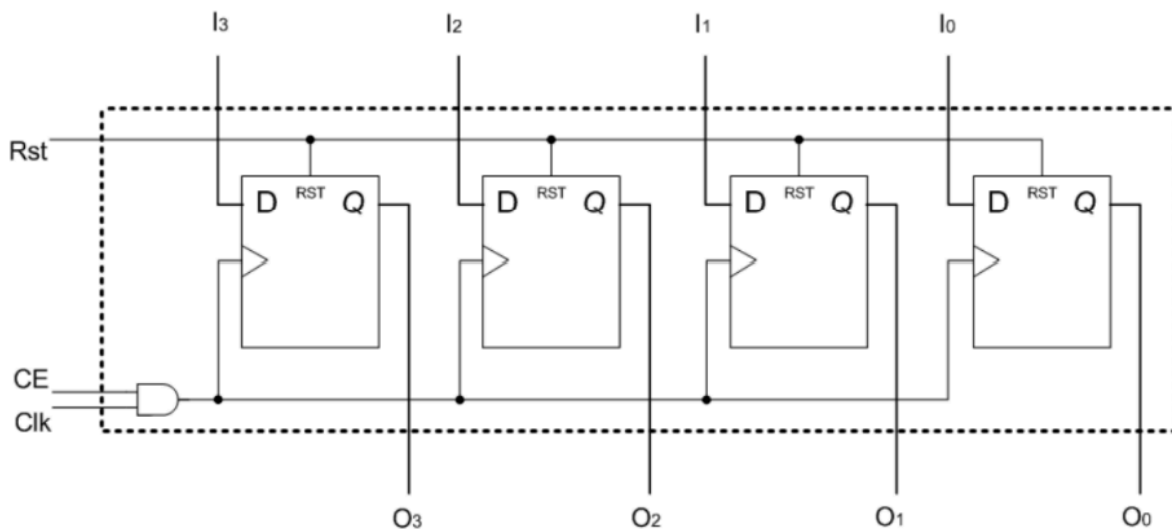
Ejercicio 3: Interfaz para teclado hexadecimal

Para este ejercicio se tiene que buscar individualmente el funcionamiento de cada uno de los bloques a implementar, para el contador de 2 bits se necesita una señal de reloj y una variable que aumenta el valor con cada ciclo del reloj, cuando llega al límite se reinicia y empieza otra vez la cuenta. El divisor de relojes toma la señal de reloj original y se diseña un circuito que transforma la señal de reloj en una señal deseada con una frecuencia diferente. Para el codificador se define primero los valores de entrada y los valores de salida deseados, con esto se realiza la tabla de la verdad. Para el módulo de sincronizador y antirebote se necesita primero analizar el funcionamiento del mismo módulo, definir las entradas y salidas. Cuando se tengan todos los módulos listos, se tienen que identificar las conexiones dentro del diagrama y con esto se realiza la implementación de todos los códigos dentro de un módulo principal donde están todos los módulos anteriores.



Ejercicio 4: Decodificador hex-to-7-segments

En este ejercicio empieza con la estructura de un registro con carga paralela, con activación de reloj: El registro con carga paralela puede incluir una señal de habilitación de reloj CE (Clock Enable).



Si la señal CE está en un estado lógico bajo ($CE=0$), la entrada de reloj que se aplica a los biestables del registro siempre será de valor lógico bajo ($CLK=0$). En cambio, si la señal CE está en un estado lógico alto ($CE=1$), la entrada de reloj CLK se conectará directamente a los biestables del registro. Donde su tabla de funcionalidad es la siguiente:

Rst	Clk	CE	O ₃	O ₂	O ₁	O ₀
1	X	X	0	0	0	0
0	↑	0	no cambia			
0	↑	1	I ₃	I ₂	I ₁	I ₀

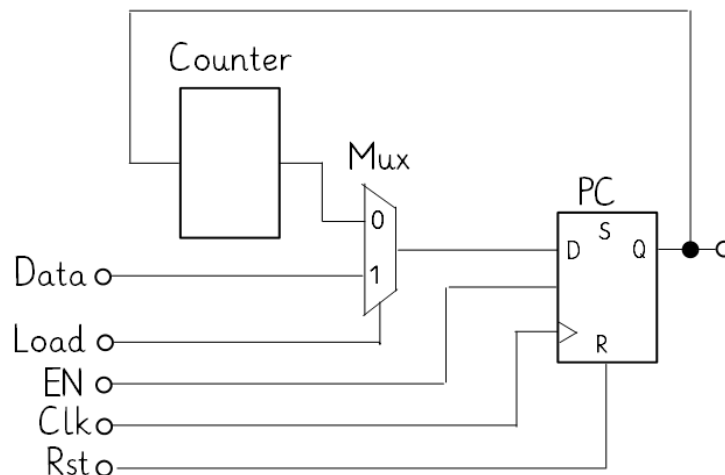
Además de esto, tomamos de ejemplo el decodificador hecho en el primer laboratorio para generar los dos juegos, uno un Registro-16 bits (4 dígitos hex de 4 bits) y el segundo un juego de decodificadores y señales de control. Del decodificador del laboratorio 1 tenemos la siguiente tabla de verdad:

Inputs				Segments							Display
A	B	C	D	a	b	c	d	e	f	g	
0	0	0	0	1	1	1	1	1	1	0	0
0	0	0	1	0	1	1	0	0	0	0	1
0	0	1	0	1	1	0	1	1	0	1	2
0	0	1	1	1	1	1	1	0	0	1	3
0	1	0	0	0	1	1	0	0	1	1	4
0	1	0	1	1	0	1	1	0	1	1	5
0	1	1	0	1	0	1	1	1	1	1	6
0	1	1	1	1	1	1	0	0	0	0	7
1	0	0	0	1	1	1	1	1	1	1	8
1	0	0	1	1	1	1	1	0	1	1	9
1	0	1	0	1	1	1	0	1	1	1	A
1	0	1	1	0	0	1	1	1	1	1	b
1	1	0	0	1	0	0	1	1	1	0	C
1	1	0	1	0	1	1	1	1	0	1	d
1	1	1	0	1	0	0	1	1	1	1	E
1	1	1	1	1	0	0	0	1	1	1	F

Posteriormente se agrega un módulo de pruebas utilizando un Linear-Feedback Shift Register, donde se verificar el recorrido de todo el registro y generando datos aleatorios que serán mostrados en el display de 7 segmentos, además de la implementación de su respectivo testbench, tanto con datos manuales como aleatorios con el fin de verificar su correcto funcionamiento previo a la implementación en la FPGA.

Ejercicio 5. Program Counter (PC)

En el caso de este ejercicio se pretende realizar un PC que posea un ancho de contador de 8 bits, ya que es un rango adecuado de direcciones de memoria y no requiere de tantos recursos de la FPGA. El módulo de PC tendrá entradas de clock, reset y un selector de operaciones. Se hace uso de FF tipo D para almacenar el valor del contador.

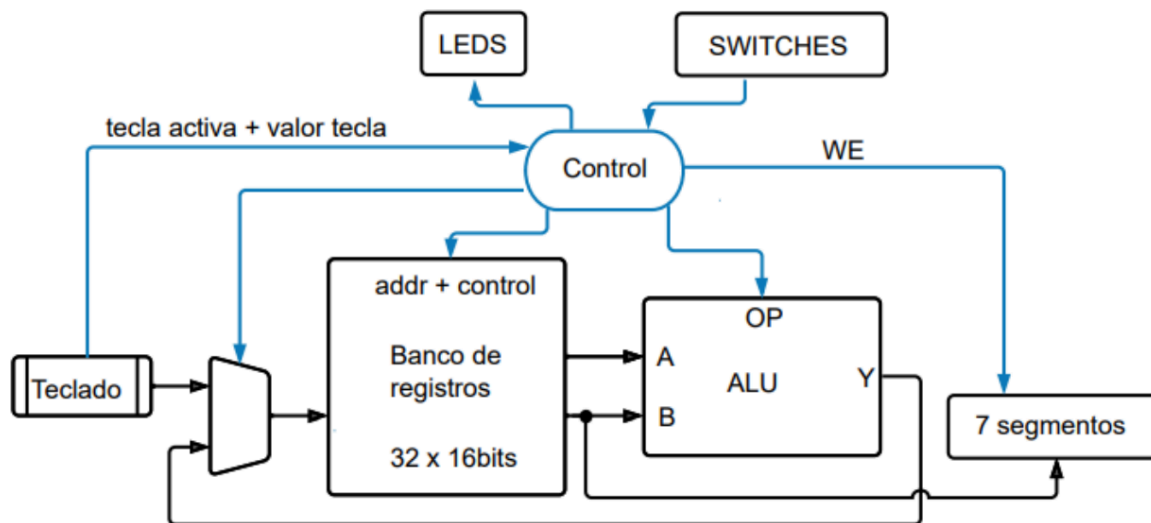


Ejercicio 6: Banco de Registros

Para este ejercicio primero se debe de tener en cuenta que un banco de registros es un conjunto de módulos. Primero se tiene el módulo de registro que cuenta con una entrada de reloj, un reset, un dato, un enable y una salida, este módulo se encarga de almacenar el dato de entrada si el enable está en alto, si se encuentra el enable en bajo la salida mantiene el valor anterior y solamente si el reset se activa el valor se actualiza como un 0. Cuenta con un módulo de selección donde se implementa un mux para seleccionar la entrada que se desea guardar en el registro, este módulo cuenta con una cantidad de entradas específicas y un selector de N bits y cuenta con una salida para el dato seleccionado. Al tener identificados los módulos y la función de cada uno se pueden instanciar en un módulo principal que va a ser el banco de registros

donde se va a conectar todo. Ya con el banco se busca la forma de realizar el testbench con los valores aleatorios para realizar las pruebas correspondientes para conocer si los valores son los esperados.

Ejercicio 7: Mini unidad de cálculo



Control: El módulo controlador se encarga de controlar las señales de activación/desactivación de los distintos módulos que son parte de la unidad de cálculo a la vez que determina el modo de operación de la unidad y maneja ciertas salidas como códigos de error.

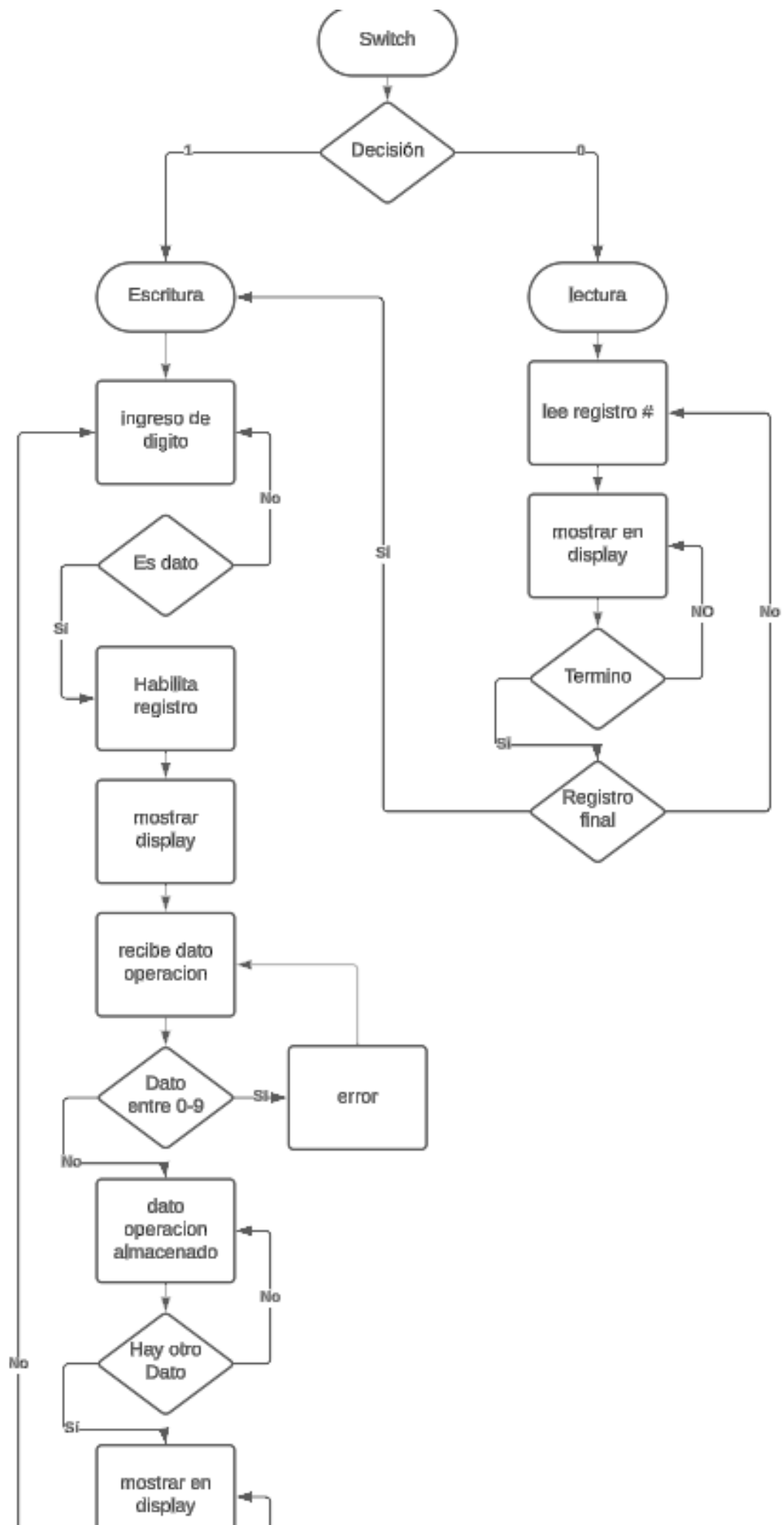
Banco de registros: El banco de registros de la calculadora funcionará en esencia de la misma manera que en el ejercicio 5, con la diferencia que el *we* será una señal que proviene del control de la calculadora, no será parametrizable, por lo que las palabras serán de 16 bits y serán 32 registros, las entradas del banco de registros vendrán de un demultiplexor cuyas entradas son el teclado y la salida de la calculadora, mediante este demux se escribirán las entradas y la salida de la calculadora para guardar las operaciones que se han hecho. El select del demux vendrá de la unidad de control de la calculadora. Los modos de escritura y lectura serán controlados por la unidad de control de manera que se puede decidir si se quieren hacer cálculos (solamente escritura) o si se quieren revisar los cálculos anteriores (solamente lectura).

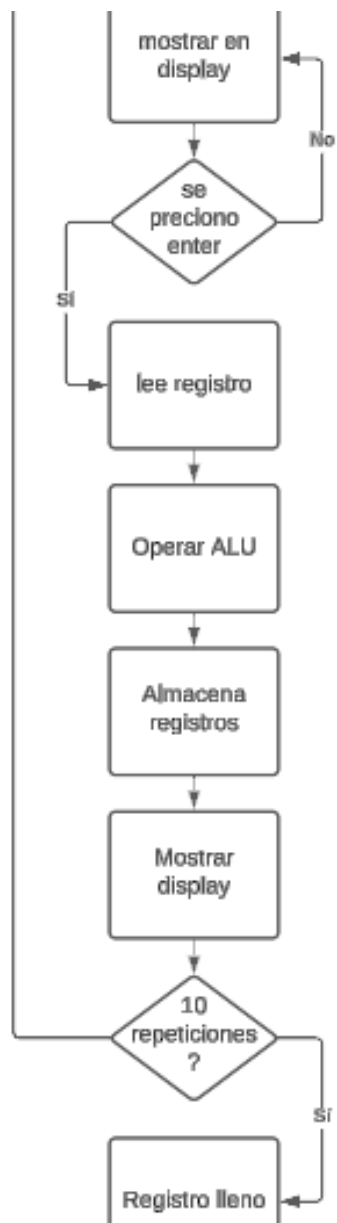
Teclado Hexadecimal: El teclado en la mini unidad de cálculo va a trabajar como entrada para la unidad de control. El teclado va a tener todo el mismo

funcionamiento que se explica en el ejercicio 3 con la diferencia que el valor de salida que tenía el teclado que iba a ser representado en el display de 7 segmentos ahora pasa a ser un dato que se va a utilizar para los calculos. Es decir, el valor de la tecla presionada va a ser guardado en los registros y luego ese valor se utilizará en la ALU para realizar las operaciones. De igual manera la operación a realizar será seleccionada desde el teclado.

Display 7 segmentos: El display de 7 segmentos en su primer modo, se utiliza para mostrar los números y resultados de las operaciones realizadas en la ALU (Unidad Lógica Aritmética). Dado que el sistema está diseñado para trabajar con datos de 16 bits, es posible que algunos números sean demasiado grandes para mostrarlos en un solo display de 7 segmentos. En ese caso, se pueden mostrar solo los dígitos más significativos o usar un enfoque de desplazamiento para mostrar todos los dígitos a lo largo del tiempo. Por ejemplo, cuando se ingresan datos desde el teclado y se almacenan en el banco de registros, el sistema utiliza el display de 7 segmentos para mostrar los datos ingresados. Además, cuando se realizan operaciones en la ALU y se almacenan los resultados en el banco de registros, el resultado también se muestra en el display. En su segundo modo, el modo de visualización, se recorren los datos ingresados y los resultados almacenados, el display de 7 segmentos se utiliza para mostrar cada valor durante dos segundos antes de pasar al siguiente. Esto permite al usuario ver los números y resultados almacenados en el banco de registros seleccionado.

Diagrama de flujos





máquina de estados

