# Documentación Ejercicio 5: Mini unidad de cálculo

Para la elaboración de la mini unidad de cálculo se utiliza una implementación de una máquina de estados (FSM). Específicamente se utiliza una máquina de estados de Mealy. Esta maquina se encarga de moverse por cada uno de los estados necesarios para cumplir con la unidad de cálculo. Esta unidad se basa en la siguiente imagen:

A diagram of a light switch

AI-generated content may be incorrect.

En el desarrollo de esta unidad se utilizan módulos de ejercicios anteriores. Estos son: ALU, Banco de registros, LFSR, Mux y display 7 segmentos. Por lo que para mas detalle de cada uno de estos modulo por favor referirse a la documentación individual de cada módulo.

## FSM Mealy:

Como se mencionó anteriormente la máquina desarrollada es una maquina de Mealy. Esto porque la maquina de Mealy no solo depende del estado actual, sino que depende de la entrada al mismo tiempo. Esto hace que la maquina sea más rápida pero más complicada de implementar. Esta decisión se hizo ya que se quiere tener mayor control del movimiento de la máquina.

La maquina final tiene un total de 10 estados diferentes de los cuales se utilizan 9 para cumplir con las especificaciones de la unidad de cálculo. E implementando el uso de función enum para facilitar la identificación de cada uno.

Ya que unidad trabaja un total de tres datos por ciclo (dato 1, dato 2 y resultado), se implementa una variable llamada op\_counter que empieza en 0. Esta variable cumple dos funciones. La primera es llevar una cuenta de la cantidad de operaciones realizadas. Ya que como máximo puede realizar 30 operaciones. Donde luego necesita volver a la posición 0 en memoria para almacenar datos. Pero el segundo propósito de esta variable es manejar la dirección de almacenamiento de los datos. Esto lo hace la siguiente manera. En la lógica de señales de salida se tiene que el dato 1 siempre se almacena en op\_counter\*3, el dato 2 en op\_counter\*3 + 1 y el resultado en op\_counter\*3+ 2. Esto se hace ya que al op\_counter comenzar siempre en 0 se tiene que en el primer ciclo las direcciones de memoria son 0, 1 y 2. Para el segundo ciclo op\_counter = 1, por lo que las direcciones de memoria en la segunda iteración son 3, 4 y 5. Y así sucesivamente, logrando que no haya un error en las direcciones de memoria.

Además de esto se implementan tres contadores, dos de estos se utilizan para esperar los dos segundos que se presentan los datos en el display. Esto se hace mediante una condición if, que se encarga de verificar que la variable correspondiente sea igual a 20000000. Si no lo es le suma y si lo hace activa la bandera. El tercer contador se utiliza para el segundo modo y cuenta hasta 30, para recorrer cada una de las posiciones de memoria.

Seguidamente se tiene una sección que utiliza la variable step para mantenerse en un mismo estado por exactamente dos ciclos. Esto lo hace al comenzar con step en 0 y chequea si step =1. Como no lo es lo niega y el próximo ciclo hace lo mismo. Pero esta vez sí es 1, así que se puede continuar con el siguiente estado. Es una sección bastante sencilla, pero permite a los estados 1 y 2 mantenerse en ese estado por los dos ciclos que ocupan.

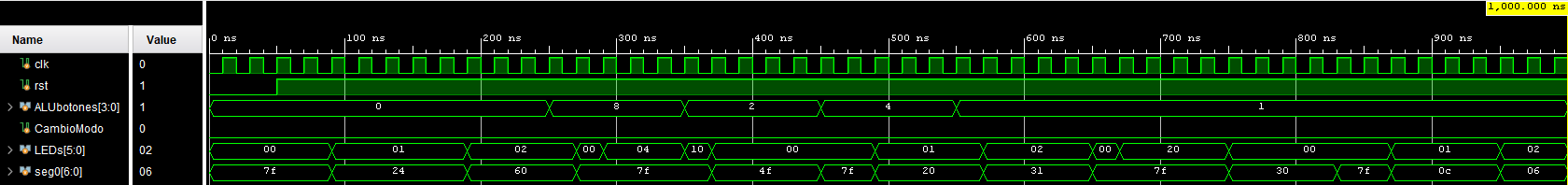
Finalmente se tienen las secciones de lógica de transición de estados y lógica de señales de salida. Estas es mejor leerlas directamente del código ya que están de forma bastante explicita y fáciles de entender.

## Top:

El top de la unidad aparte de ser una interconexión de módulos implementa el reloj para la FPGA. Utilizando el IP-Core de vivado, específicamente el Clocking Wizard para establecer el reloj de la FPGA en 10 MHz. Con esto se logra configurar el reloj interno de la FPGA para que cumpla con los requisitos de la unidad.

## Resultados:

Para comprobar los resultados en la simulación de vivado se reducen los contadores de los display para poner ver múltiples iteraciones en una sola simulación. Es de esto que se obtiene la siguiente imagen:



En la imagen anterior seg0 es el dato en el display. Es de esto que podemos ver que se generan los primeros dos datos para ser utilizados en el calculo de la ALU. Además, se puede apreciar como los leds se encienden correspondientemente al dato presentado. 01 para dato 1 y 02 para dato 2. Similarmente el valor de los leds también cambia para representar la operación que se está ejecutando. Debido al comportamiento de los leds en la fpga 7f es 0. Es decir que en ese memento el display no tiene un dato a presentar.