

**MDR**

**Nombre:**

Jorge Mizael Rodríguez Gutiérrez

Luis Roberto Lomelí Plascencia

**Expediente:**

IE698323

IE700093

**Materia:**

Diseño y verificación

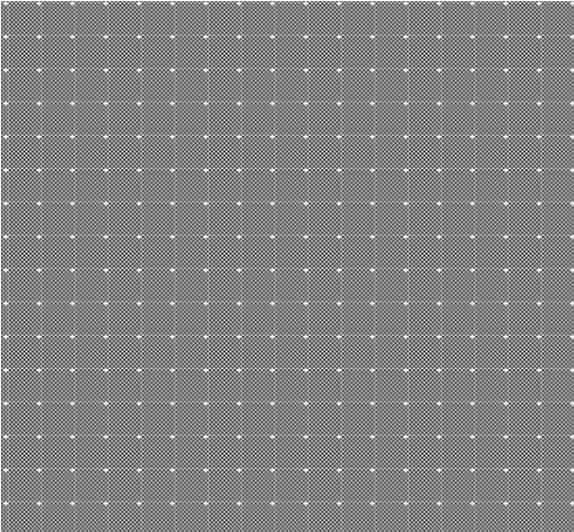
**Fecha:**

16/Abril/2019



Esta práctica tiene como objetivo el desarrollar un módulo aritmético que calcule la multiplicación, división y raíz cuadrada de un número entero con signo.

La interfaz del Multiplicador, Divisor y Raíz cuadrada (MDR) cuadrada se muestra se muestra en la Fig. 1 y sus señales se describen en la Tabla 1.



*Figura 1: Interfaz del MDR*

|  |  |
| --- | --- |
|  | Entradas |
| **Señal** | Descripción |
| ***Data*** | Es el puerto de entrada de datos de 16 bits. |
| ***Start*** | Cuando esta señal es igual a 1 lógico, el modulo comienza a trabajar. |
| ***Load*** | Cuando es igual a 1 lógico, el valor que se encuentra en la señal ***Data*** se carga dentro del MDR. |
| ***Op*** | Selecciona la operación a realizar:  0: Multiplicación  1: División  2: Raíz cuadrada |
| ***clk*** | Señal de reloj. |
| ***reset*** | Señal de reinicio. |
|  | Salidas |
| **result** | Este puerto entrega el resultado de la operación ejecutada, tiene un ancho de 16 bits. |
| **ready** | Cuando es igual a 1 lógico indica que el resultado en el puerto **Result** es válido. |
| **Remainder** | Es el puerto donde se muestra el residuo de la operaciones de división y raíz cuadrada. |
| **Load X** | Cuando es igual a 1 lógico indica que se debe colocar el valor de **X** a cargar en el puerto **Data**. |
| **Load Y** | Cuando es igual a 1 lógico indica que se debe colocar el valor de **Y** a cargar en el puerto **Data**. |
| **error** | Indica un error en el resultado. |

# Descripción funcional

El modulo a diseñar e implementar realiza la multiplicación, división o raíz cuadrada de dos números de 16 bits**X** y**Y** en forma secuencial, **X** y **Y** son números con signo a complemento a dos.

Los datos **X** y **Y** son cargados al MDR de la siguiente forma:

Cuando se presiona **Start**, tal que **Start** == 1, al siguiente pulso de reloj la salida **Load X** debe ponerse en uno,indicando que se debe colocar **X** en el puerto **Data**, una vez colocado se presiona **Load**, tal que cuando **Load** == 1 se carga **X** en el MDR.

Después de cargar **X**, la **Load** **Y** de debe poner en uno para indicar que se debe cargar el dato **Y,** seguido de esto se coloca el valor **Y** en el puerto **Data** y se presiona **Load**. Una vez cargados los datos **X** y **Y**, se comienza el cálculo de la operación que indique el puerto **Op**. Durante el proceso de cálculo, los resultados parciales no deben de mostrarse en el puerto **Result**, y cuando se alcance el valor resultante de **X** y **Y**, la señal **ready** debe ponerse en uno lógico.

Los resultados parciales deben mostrarse en 16 leds de la tarjeta DE2-115, además el resultado final se mostrará en los displays de 7 segmentos con un led que indique cuando el resultado es negativo, todo esto una vez que **ready** ==1.

Cuando la operación a ejecutar es raíz cuadrada, el único dato a cargar es el valor de **X**.

La bandera de error se activa cuando se detecte los siguientes casos:

* La multiplicación sobre pase los bits de salida, por ejemplo X=0xFFFF por Y=0xFFFF.
* División entre cero;
* Calculo de la raíz cuadrada de un número negativo.

En el caso de existir sobre flujo la salida del MRD se debe saturar, es decir, Resultado = 0xFFFF.

# Restricciones

1. Los diferentes algoritmos a implementar se deben a hacer en forma secuencial.
2. El algoritmo de multiplicación a implementar debe ser el algoritmo de Booth
3. El algoritmo de división es a libre elección, como recomendación se puede usar el algoritmo de CORDIC.
4. El algoritmo de raíz cuadra debe ser el expuesto en clase.
5. El diseño y la implementación solo puede contener un sumar/ restador.
6. El diseño y la implementación no se puede hacer uso de multiplicadores
7. Todas las entradas deben ser registradas 8. El reloj del módulo debe ser al menos 5Mhz.
8. La implementación en FPGA debe hacer uso de un PLL para la generación de reloj, el cual se puede combinar con un divisor con contadores síncronos para obtener la frecuencia deseada.
9. El PLL y el divisor síncrono tiene que estar encapsulado en un módulo llamado generador de reloj.
10. El MDR debe ser capaz de procesar operaciones
11. En los displays de 7 segmentos no se deben mostrar los resultados parciales, solo el final cuando ready == 1.
12. Todo aquello que no esté definido es la especificación de la práctica queda a libertad del diseñador.
13. La implementación tiene que hacer uso de un package para la definición de tipos de datos enumerados, tipos definidos. Debe hacer uso de interfaces y todos aquellos nuevos elementos del lenguaje de descripción de hardware SystemVerilog.

# Entregables

Un reporte que contenga:

* Descripción de la arquitectura propuesta, la cual debe incluir la descripción funcional de cada módulo que compone el diseño. Esta descripción debe ser estilo Technical Design Specification (TDS).
* Simulación funcional en ModelSim con su script. El script de compilación debe contener una macro para hacer bypass del eliminador de rebotes, PLL y/o divisor de reloj para fines de simulación.

o Para cada caso de ejecución multiplicación, división y raíz cuadrada.

* Análisis de señales utilizando SignalTap reportado vía video en Youtube o cualquier plataforma de streaming como video público.
* Resultados de síntesis en términos de logic elements (LEs), y la frecuencia máxima de operación del diseño.

Archivos de implementación con SystemVerilog.

* Todas las fuentes deben ser comentadas, de lo contrario el modulo en particular no tendrá valor

# Evaluación

Estos son los elementos a evaluar y sus respectivos pesos:

1. **Funcionalidad**. Se considera que MDR funciona solo si se compila sin errores en Quartus y Modelsim y SignalTap. Este aspecto tiene un peso del 30%. Si la descripción no se puede compilar, o la simulación no termina o se aborta, se obtendrán cero puntos en este apartado. Al momento de revisión se debe mostrar las señales internas del módulo a través del SignalTap.

1. **Reporte**. Se evaluará claridad y sencillez en la redacción, ortografía, que se presenten las decisiones clave que tuvieron que tomar para resolver el problema, que contenga la información solicitada y que los datos de los integrantes estén completos y sean correctos. El peso de este apartado es de 30%.

1. **Preguntas**. Durante la revisión de la práctica, el profesor hará varias peguntas a ambos integrantes del equipo. Cada pregunta va dirigida a un integrante en particular (el otro integrante no puede intervenir). Las preguntas son sobre la práctica y sobre los conceptos necesarios para realizarla. Se cuestionarán también las razones por las que decidió implementar una solución específica. Ambos integrantes deberán estar enterados de todos los aspectos de la práctica. Este apartado tiene un peso del 40%. Los puntos obtenidos dependerán de la proporción de preguntas contestadas correctamente. La nota obtenida en este apartado es individual.

**Recomendación:**

Para la simulación se puede omitir el circuito generador de frecuencia.

**Puntos extra:**

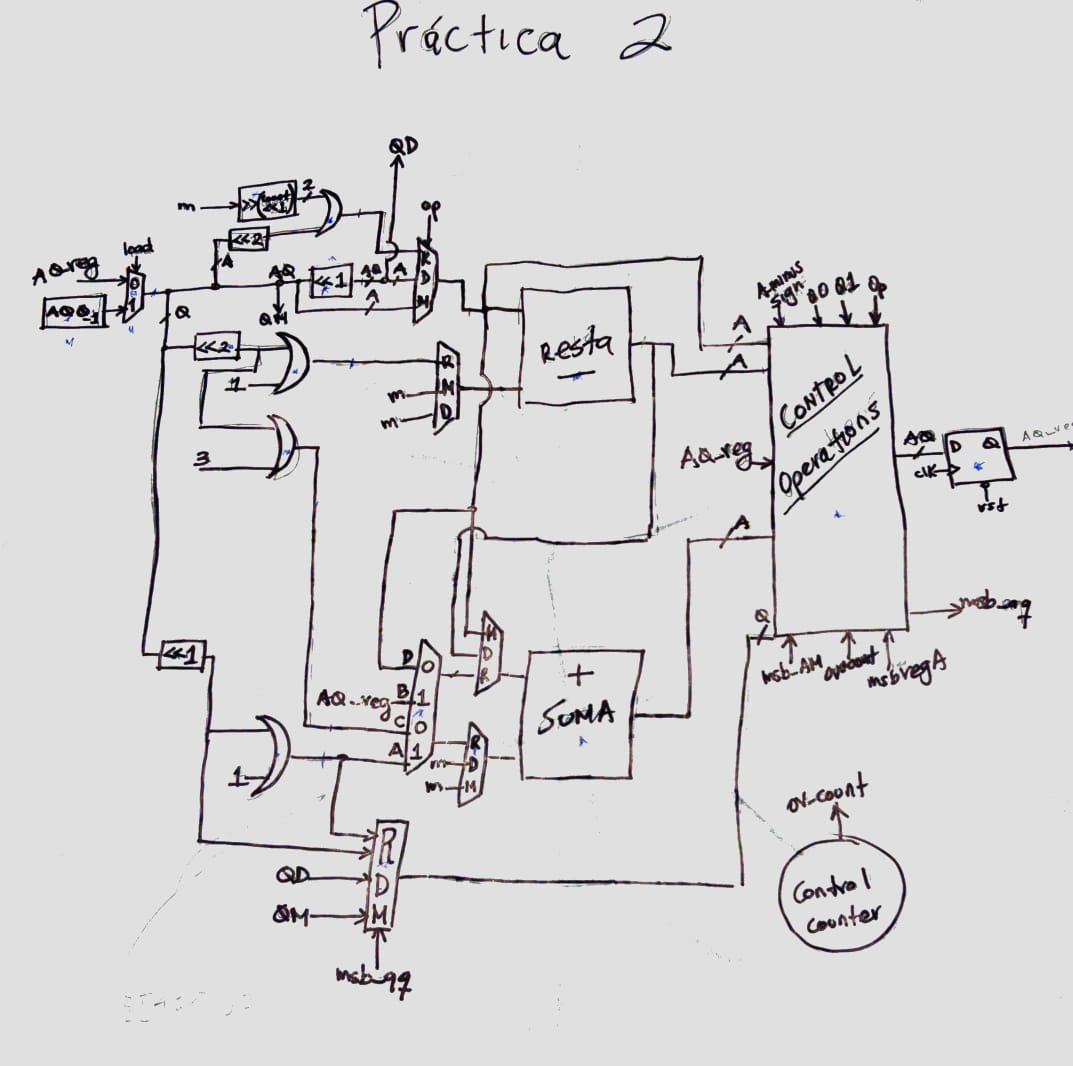
Se considerarán 20 extras en el caso de integrar el teclado PS2 y el display de LCD. Es decir, introducir datos por teclado y visualizarlos por el display de LCD.

Se considerarán 10 puntos extras si la división se implementa con el algoritmo de CORDIC.

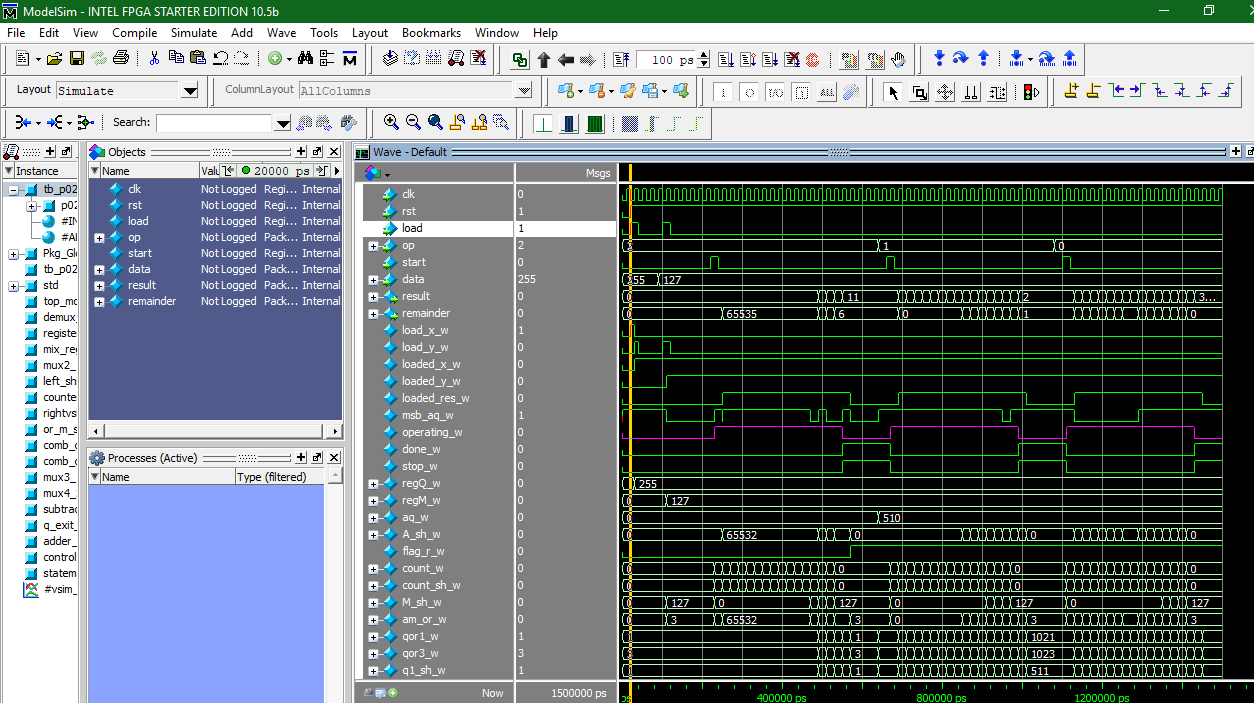
**DESCRIPCIÓN FUNCIONAL**

|  |  |
| --- | --- |
| Modulo | Descripción |
| shifting | Este modulo se encarga de realizar un shift dependiendo de su parámetro de entrada |
| Separator | Este modulo se encarga de separar las entradas y salidas para el display de 7 segmentos (es parte del algoritmo double dabble ) |
| Segments | Este módulo se encarga de sacar un resultado en números a los displays de 7 segmentos de la tarjeta. |
| Comparer | Este módulo sirve para desplazar las unidades, decenas y centenas del algoritmo double dabble |
| btd | Algoritmo de double dabble para mandar el resultado a los displays de 7 segmentos. |
| Mux\_results | Este modulo se encarga de sacar el resultado correcto dependiendo de la operación seleccionada |
| Mux3\_1 | Este modulo se encarga de sacar una salida teniendo tres diferentes a la entrada gracias a un selector. |
| Substractor | Este modulo se encarga de realizar la resta ante dos entradas. |
| Statemachines | Este modulo se encarga de dirigir el proceso de datos haciendo un cambio de estados y una sincronización adecuada. |
| Rightshift | Este modulo realiza un shift a la entrada, dependiendo de la segunda entrada. |
| Register | Este modulo se encarga de registrar las entradas y salidas del sistema. |
| Q\_exit | Elige sacar la Q adecuada, ya que para la practica es necesario sacar diferentes Q dependiendo de la operación. |
| Or m |  |
| Mux 4\_2 | Este modulo se encarga de sacar dos salidas ante 4 entradas, esta función es parecida al multiplexor de 2 a 1 |
| Mux 2\_1 | Este modulo se encarga de sacar una salida ante dos entradas, dependiendo de lo que diga nuestro selector. |
| Mix\_regs | Este módulo realiza una compuerta or de dos diferentes entradas. |
| Left\_shift | Este moduo se encarga de hacer un shift hacia la izquierda del dato de entrada. |
| Demux | Permite la salida de una señal de datos, ante dos entradas, este modulo sirve para dejar entrar valores al mdr |
| Debouncer | Este modulo se encarga de eliminar los rebotes de la señal recibida por los botones. |
| Counter | Este modulo se encarga de llevar el conteo de los ciclos de reloj que necesitan as operaciones para procesar el resultado. |
| Control operation | Este modulo se encarga de dejar salir el dato de la operación adecuada al registro de salida, este módulo recibe datos procesados de diferentes operaciones y decide cual deja salir. |
| Comb\_qshift2 | Este módulo se encarga de procesa una entrada obteniendo dos salidas, la primer salida solo shiftea hacia la izquierda la entrada y aplicar una or de con uno al bit menos significativo, la segunda de realizar el shifteo y aplicar una compuerta or con un 3 en los bits menos significativo. |
| Comb\_qshift1 | Este módulo se encarga de procesa una entrada obteniendo dos salidas, la primer salida solo shiftea hacia la izquierda la entrada, la segunda de realizar el shifteo y aplicar una compuerta or en el bit menos significativo. |
| Adder | Este modulo se encarga de realizar la suma aritmética de dos entradas |

**Diagrama RTL de nuestro diseño**

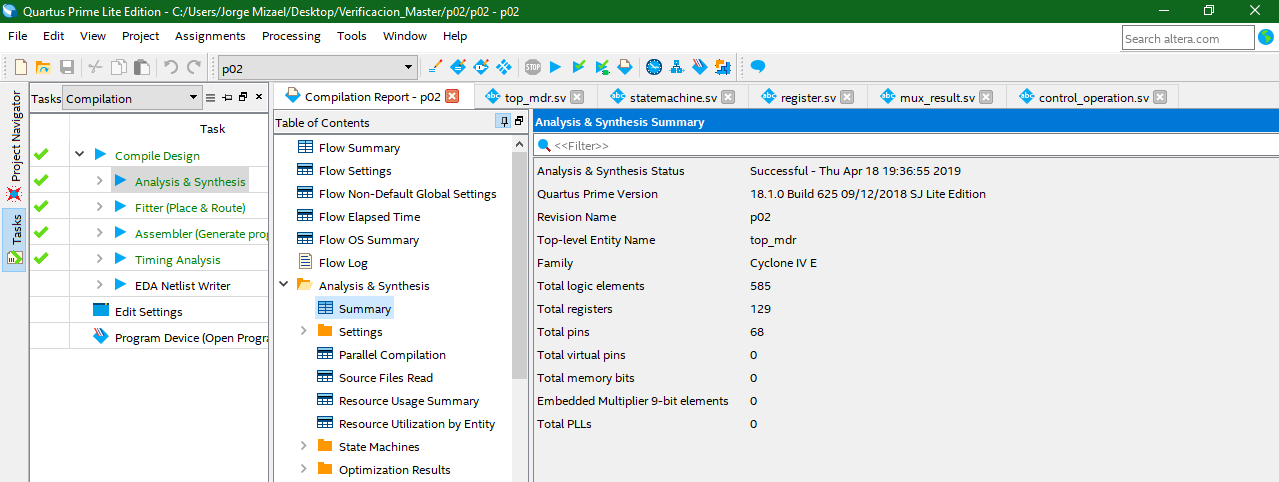
****

**Simulación en Modelsim**

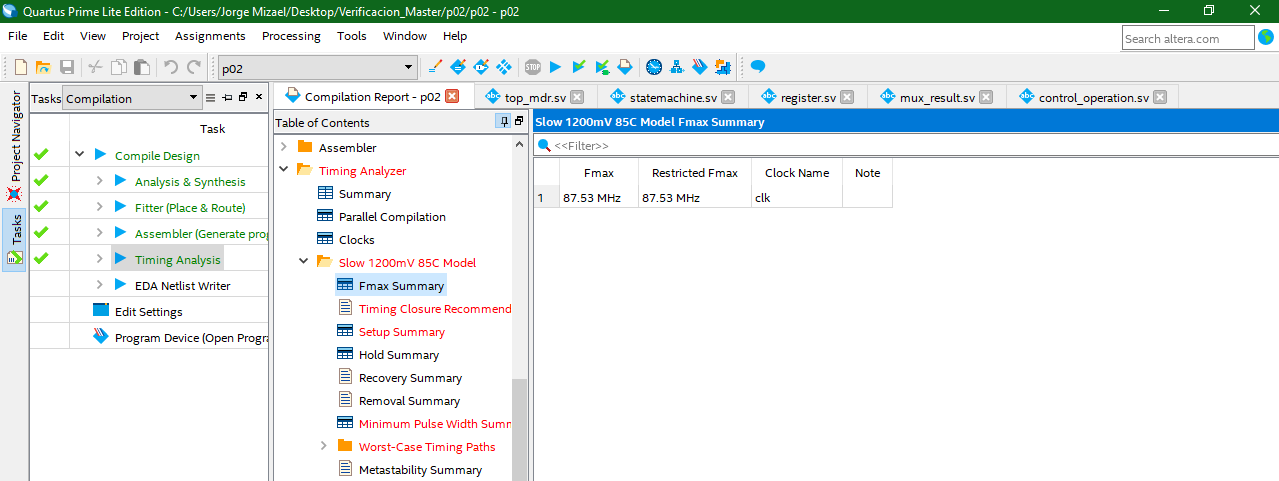


Esta imagen a grandes rasgos representa las tres operaciones realizadas por nuestro mdr

**Cantidad de logic elements**



**Frecuencia máxima de operación**



**Conclusiones**

La practica fue algo complicada ya que tuvimos que hacer funcionar 3 algoritmos, para poder hacer la practica decidimos hacer funcionar cada uno de ellos por separado para comprenderlos, después de esto decidimos juntarlos todos apegándonos a las restricciones solicitadas, en general fue una practica retadora, ya que no la logramos terminar por completo para poder empezar con la practica 3

github

<https://github.com/lrolomeli/verificacion_2019_mizael_luis>