# ELO212: Laboratorio de Sistemas Digitales
# Guía de Actividades Sesión 8 (Evaluada)

12 al 16 de junio de 2023

## 1. Requisitos de entrada.

- Haber **completado y entendido** las actividades de todas las sesiones anteriores.
- Contar con un repositorio de módulos diseñados, simulados, y verificados de las sesiones previas.

## 2. Objetivos

- Agregar funcionalidad a la calculadora con notación polaca inversa desarrollada en la sesión anterior, considerando módulos y restricciones impuestas por los periféricos de la tarjeta de desarrollo para su implementación física.
- Familiarizarse con el flujo completo de diseño, implementación, y verificación en hardware para FPGAs, entendiendo la información entregada por cada una de las etapas del proceso utilizando Vivado.

## 3. Indicaciones generales.

Las actividades a realizar en esta sesión incluyen actividades evaluadas en base a simulación mediante tests exhaustivos (similar a lo realizado en sesiones previas), y también evaluación de hardware implementado en la tarjeta de desarrollo que tiene disponible para su grupo.

Para la revisión de las actividades de simulación, considere el mismo criterio e indicaciones utilizadas para las sesiones evaluadas previas. Las actividades de implementación se revisarán durante la sesión mediante inspección de la funcionalidad del diseño implementado en la tarjeta de desarrollo. Todas las actividades solo serán revisadas luego de que el grupo responda una interrogación en que muestren dominio de los conceptos y técnicas asociadas a lo que están entregando. En esta ocasión, la interrogación incluirá consultas asociadas a la interpretación de reportes entregados por Vivado para las etapas de síntesis e implementación (uso de recursos, codificación de máquinas de estado, identificar los mensajes de warnings y errores, etc.).

Las actividades serán revisadas en el orden en que están planteadas. No se responderán consultas o revisará una actividad si no ha completado la actividad anterior.

Durante la sesión, el staff priorizará la revisión de actividades e interrogaciones por sobre la atención de consultas generales. Para hacer un uso eficiente del tiempo durante la sesión, prefiera agrupar las dudas y tenga el material asociado listo al momento de llamar al staff.

### 3.1. Indicaciones de formato para entrega.

Los archivos entregables para la revisión de las actividades mediantes simulación deben seguir estrictamente las especificaciones que se le indican a continuación:

1. Debe crear una carpeta con el nombre *GPPXX_Sesion8_2023*, donde $PP$ indica el paralelo (LU, MA, MI, o VI) y $XX$ el número de grupo con dos dígitos (00, 01, 02, etc.). Por ejemplo, la carpeta para el grupo 5 del paralelo del Lunes se llamaría `GLU05_Sesion8_2023`.
2. Dentro de dicha carpeta debe crear una subcarpeta para cada actividad con el nombre *ActividadY*, donde $Y$ indica el número de la actividad.
3. Cada subcarpeta debe subir únicamente los archivos denominados "Design Sources" (archivos de diseño) asociados a cada una de las actividades. Para esta sesión, esto solo considera archivos con extensión *.sv*. Notar que **NO** debe incluir archivos de simulación, constraints, o archivos temporales generados por Vivado. Se descontará puntaje por formato de entrega si agrega archivos innecesarios.
4. Cada una de las actividades debe contar con un módulo principal (top module) con el nombre *S8_actividadY*, donde *Y* es el número de la actividad correspondiente. Recordar que los nombres de las entradas y salidas deben ser los mismos que se indican en la presente guía (tener en cuenta que SystemVerilog es case-sensitive, por ende $P \neq p$).

Recordar que **será responsabilidad del grupo entregar de manera correcta los archivos**. Ante cualquier duda sobre las indicaciones de entrega, contacte al staff o consulte por Aula **antes de su sesión**.

## 4. Actividades evaluadas

Para cada una de las siguientes actividades, se solicita que registren el ***tiempo de trabajo efectivo*** aproximado en horas invertido por su grupo para cada actividad. Esto considera actividades como búsqueda de información, discusión grupal sobre aspectos de diseño, descripción de código, simulaciones, etc. Tiempos de ocio o distracciones en el medio de sesiones de trabajo, si bien son necesarias, no cuentan como tiempo efectivo de trabajo.

**Indicación general para simulaciones:** Para garantizar un comportamiento determinístico en sus simulaciones, evite realizar cambios en las señales de entrada simultáneos a los cantos de reloj a los cuales son sensibles los FFs correspondientes (típicamente cantos positivos de reloj). Es decir, cada vez que haga un cambio de entrada en la simulación asegúrese que este desfasado con respecto al canto positivo más cercano. Realizar cambios de entrada simultáneos a los cantos activos de reloj no es un error en sí, pero puede generar comportamiento no determinístico en las salidas debido al scheduling de tareas especificado en el estándar SystemVerilog. Para mayor informa-

ción, puede revisar el siguiente documento: `http://www.sunburst-design.com/papers/CummingsSNUG2006Boston_SystemVerilog_Events.pdf`

Las dos primeras actividades descritas a continuación se evaluarán mediante evaluación exhaustiva **posterior al término de la sesión**, mientras que la tercera actividad se evaluará mediante inspección directa de la implementación en hardware **durante la sesión**. Ambos tipos de actividades consideran objetivos de aprendizaje y criterios distintos, por lo que se analizarán en forma aislada. Esto implica que el hecho que la actividad de implementación haya sido calificada como correcta mediante inspección visual del hardware, no implica que las actividades de simulación estén automáticamente correctas. Esto puede ocurrir, por ejemplo, en el caso de que la implementación física contenga un fallo de diseño que no sea visualmente perceptible debido a las bajas frecuencias de operación, o que su diseño funcione pero no haya seguido alguna especificación explícita descrita en el enunciado, como nombre de las señales de entrada/salida o formato de entrega de archivos fuente.

## 4.1. Módulo de sincronización y filtrado para señales generadas por pulsadores electromecánicos [Simulación]. (20 puntos)

En esta actividad deberá modificar la calculadora diseñada en la sesión anterior, agregando módulos y adaptando la lógica para que pueda funcionar bajo las restricciones impuestas por el hardware y periféricos de la tarjeta de desarrollo Nexys4DDR/NexysA7 que tiene disponible.

Específicamente, tomando como base el diseño de la última actividad de la guía anterior (incluyendo función Undo), se pide reemplazar las instancias del módulo `Level-to-Pulse` por un módulo anti-rebotes. Esto es para filtrar y sincronizar las señales de entrada `Enter` y `Undo` provenientes de pulsadores no ideales, generando así las señales que gatillarán los cambios de estados para la calculadora polaca inversa. Además, debe modificar su descripción para que la señal de reset global opere con lógica negada.

Utilice la siguiente definición para el módulo top de esta actividad:

```
module S8_actividad1 #(
  parameter N_DEBOUNCER = 10
)(
  input  logic         clk,
                       resetN,
                       Enter,
                       Undo,
                       DisplayFormat,
  input  logic [15:0] DataIn,

  output logic [ 6:0] Segments, //solo segmentos, no considere el punto.
  output logic [ 7:0] Anodes,
  output logic [ 4:0] Flags,
  output logic [ 2:0] Status
);
```

La Figura 1 muestra un diagrama de alto nivel para la funcionalidad a implementar. Notar que la parte principal del diseño ya fue realizado en la sesión anterior, y solo se debe agregar los módulos debouncer marcados en rojo.

Considere las siguientes especificaciones:

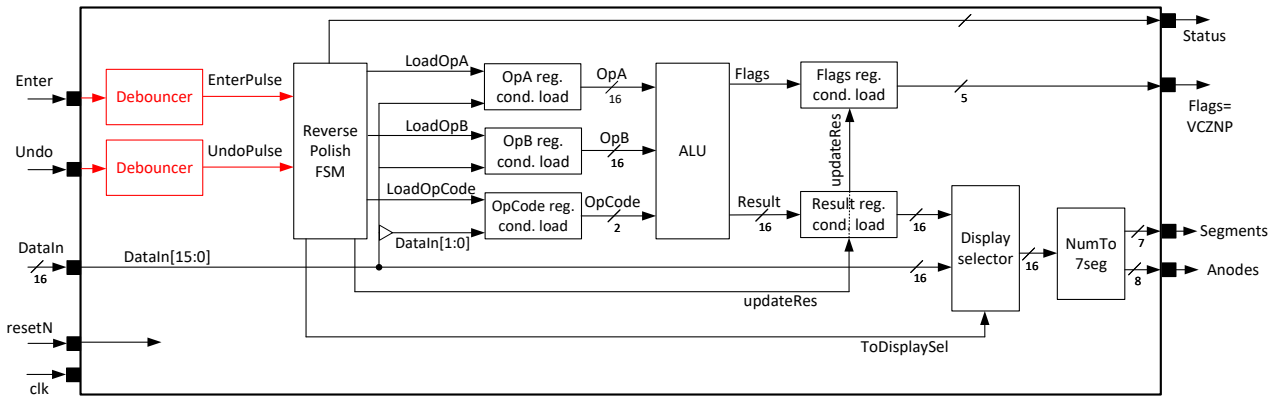


Figura 1: Diagrama de alto nivel para funcionalidad básica de calculadora. Los bloques son propuestos, y puede organizarlos internamente como estime conveniente, cuidando que las señales de entrada y salida cumplan con las especificaciones.

- La señal `resetN` opera con lógica negada (activa en bajo) y se usa en forma sincrónica. Cuando la señal `resetN` se encuentre en bajo al momento de ocurrir un canto de reloj, todos los valores almacenados en cualquier registro deben tomar el valor 0.
- Las entradas `Enter` y `Undo` son generadas mediante pulsadores electromecánicos externos no ideales, por lo cual deben sincronizarse con el reloj base `clk` y filtrarse para eliminar rebotes. Utilice para este fin un módulo antirebotes que contenga un parámetro `N_DEBOUNCER` que permita especificar el número mínimo de ciclos que la señal de entrada desde el pulsador debe mantenerse estable para detectarlo como un pulso válido. Cada presión física del botón debe detectarse solo como una presión, independiente del tiempo que el botón se mantenga presionado. Use el módulo antirebotes basado en FSM disponible en el repositorio GIT del curso. En este módulo, el parámetro `N_DEBOUNCER` del módulo top corresponde al parámetro `DELAY` del módulo anti-rebote. Para generar las señales `EnterPulse` y `UndoPulse` utilice la salida `pressed_pulse` del modulo anti-rebote (cada pulsación del botón correspondiente, debe generarse un pulso limpio de un ciclo de duración que se usa como entrada para la lógica sincrónica.).
- Asuma que la entrada `resetN` no requiere ser filtrada y por lo tanto no requiere un debouncer.
- Las salidas `Segments` y `Anodes` deben considerar el manejo de 8 displays. Tal como se estableció en las guías anteriores, debe agregar padding para implementar extensión por 0.

Verifique la funcionalidad de su diseño para distintos valores del parámetro `N_DEBOUNCER`.

## 4.2. Visualización de valores decimales en displays de 7 segmentos [Simulación]. (30 puntos)

Al diseño de la actividad anterior, agregue la funcionalidad para el bloque mostrado en rojo en la Figura 2. Notar que, habiendo seguido un diseño modular, solo necesitaría agregar módulos adicionales, sin requerir cambios en los diseños de las actividades anteriores.

Para esta actividad, utilice el siguiente módulo top:

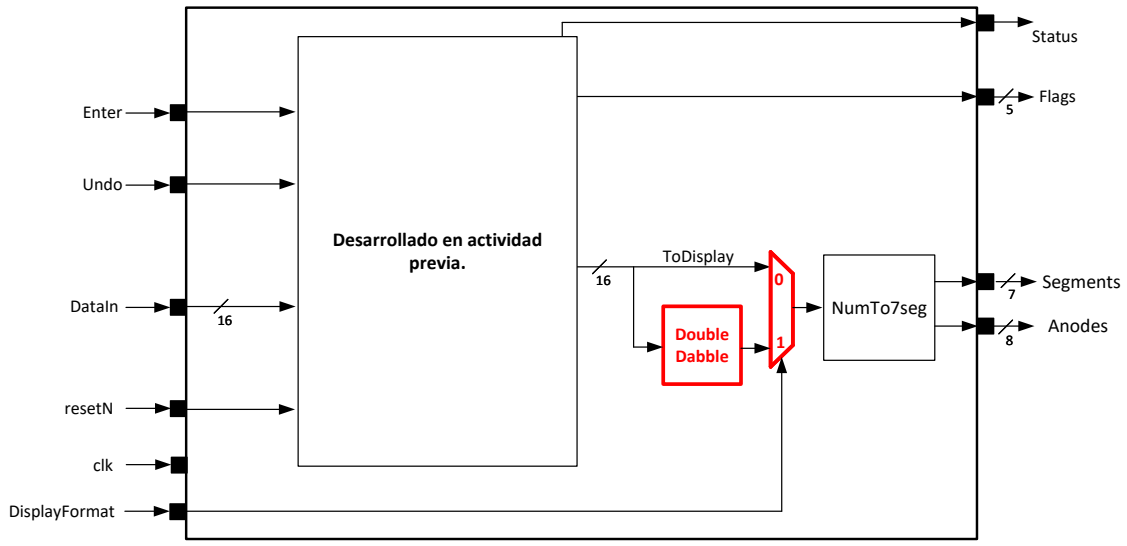


Figura 2: Base de máquina de estados para funcionalidad de calculadora en notación polaca inversa incluyendo valores de salida para cada estado.

```
module S8_actividad2 #(
  parameter N_DEBOUNCER = 10
)(
  input  logic         clk,
                       resetN,
                       Enter,
                       Undo,
                       DisplayFormat,
  input  logic [15:0]  DataIn,

  output logic [ 6:0]  Segments, //solo segmentos, no considere el punto.
  output logic [ 7:0]  Anodes,
  output logic [ 4:0]  Flags,
  output logic [ 2:0]  Status
);
```

La entrada `DisplayFormat` permite cambiar la representación de los números de salida (señal `ToDisplay` en diseños anteriores) entre hexadecimal (`DisplayFormat`=0) y decimal sin signo (`DisplayFormat`=1). Para convertir el número binario a una representación decimal que pueda ser visualizada en un display de 7 segmentos, debe utilizar el algoritmo *Double Dabble* [1]. En el repositorio del curso se encuentra la carpeta *double-dabble-32bits*, el cual contiene una descripción HDL de este algoritmo. El módulo entregado se encuentra probado y es completamente funcional, y si bien su uso es bastante intuitivo, intencionalmente se entrega con poca documentación para su uso. Analice, simule, y verifique la funcionalidad de este módulo en forma aislada en un ejemplo funcional mínimo para entender como utilizarlo e interpretar sus salidas. Prepare una lista de vectores de entradas con sus salidas esperadas, y verifique que esto se cumple en la simulación, analizando el comportamiento transiente y final de la salida. Solo una vez que entienda y haya verificado el funcionamiento general de entrada/salida y los requerimientos para realizar conversiones correctas [2],

[1] `https://en.wikipedia.org/wiki/Double_dabble`

[2] Hint: determine el tiempo que demora una conversión y los requerimientos de la entrada cuando hay una conversión

agréguelo a su diseño completo ya probado. Notar que el diagrama de alto nivel entregado es solo de referencia, y puede requerir agregar señales y lógica adicional o modificar el módulo de referencia para integrarlo a su sistema. Si sigue las indicaciones y estudia el módulo de conversión en forma aislada, la integración del sistema completo debería ser trivial.

En este caso, cualquiera sea el formato de presentación del número (hexadecimal o decimal), debe considerar los 8 displays disponibles en la tarjeta para mostrar el número, por lo que debe extender la señal `ToDisplay` original a 32 bits agregando padding de ceros a la izquierda para implementar extensión por 0.

Verifique cuidadosamente que las salidas coinciden con las especificaciones entregadas en el enunciado y el datasheet, siguiendo los requerimientos de polaridad y orden de pines especificado en la última sesión evaluada. Para efectos de validación funcional, y tal como se hizo en la última sesión evaluada, en el driver del display utilice el mismo reloj base del sistema para multiplexar los ánodos (no es necesario agregar un divisor de reloj para generar frecuencias más lentas). Recuerde también que en todo momento debe haber a lo más un display encendido, y cada display se debe encender por 1 ciclo de cada 8 ciclos de reloj. Durante sus simulaciones, asegúrese que en la lógica de controlador de display no se generan señales internas en alta impedancia (Z) durante la operación normal.

### 4.3. Implementación de calculadora en tarjeta de desarrollo (50 puntos).

Habiendo simulado y verificado la lógica principal de la calculadora, siga el flujo de implementación para validar la funcionalidad de su descripción en el hardware.

Considere el siguiente mapeo de entradas a periféricos en la tarjeta de desarrollo:

- `clk` se mapea al reloj de 100 MHz disponible por defecto en la tarjeta.
- `DataIn[15:0]` se mapea a los switches (indicados como SW[15:0]).
- `resetN` se mapea al botón `CPU_ResetN` .
- `Enter` se mapea al botón `BTNC` .
- `Undo` se mapea al boton `BTNR` .
- `DisplayFormat` se mapea al boton `BTND`.
- `Segments` y `Anodes` se mapean a los pines para que el valor de los operandos y resultado se muestre en los displays de las posiciones menos significativas (debe usar los 8 displays utilizando extensión a 0).
- `Flags` se mapea a los cinco LEDs menos significativos (indicados como `LD[4:0]` en la placa de la tarjeta).
- `Status` se mapea a los LEDs más significativos (indicados como `LD[15:13]` en la placa de la tarjeta.).

en curso.

Recuerde que debe agregar lógica de divisor de reloj y lo que considere necesario para que los displays se vean en forma clara (estáticos, no muy tenues ni muy brillantes). Considere una frecuencia de encendido entre 24 y 30 Hz aproximados por display.

Una vez se asegure de cumplir con los requerimientos funcionales mínimos indicados, siéntase libre en agregar funcionalidad adicional para facilitar el uso y depuración de su dispositivo. Cualquier cosa que añada no debe modificar los requerimientos mínimos explícitamente indicados en el enunciado.