ELO212: Laboratorio de Sistemas Digitales
Semestre I-2020
Guía de Actividades Sesión 7 (Evaluada)

28, 30, 31 de Julio de 2020

1. Objetivos.

- Integración de lo estudiado y desarrollado en sesiones anteriores para el diseño y verificación de una calculadora de 16 bits con notación polaca inversa.

2. Indicaciones generales sobre la evaluación.

Las siguientes actividades serán revisadas durante la sesión de laboratorio. Cuando complete una actividad, debe avisar al profesor o ayudante para mostrar sus resultados y responder preguntas asociadas a su diseño.

Se recomienda encarecidamente que adelante su trabajo y llegue al menos con las partes fundamentales probadas. La estadística histórica indica que no alcanzará a terminar el trabajo si no llega con algo avanzado a la sesión.

Recuerden que todo diseño debe empezar con un plan y una descripción sobre que es lo que deben implementar. En el contexto de la asignatura, esto implica realizar un diagrama de alto nivel de los módulos que necesitarán (cajitas), y como se conectarán entre ellos (flechitas que conectan las cajitas). No toque la herramienta de diseño ni escriba una línea de código sin haber realizado este paso previo. Se recuerda que las actividades no se revisarán y tampoco se responderán consultas si no tiene un diagrama de bloques de su diseño. El pensar que *"tengo todo en mi cabeza, solo necesito unos minutos para escribirlo"*, es un salto directo al fracaso.

Para el proceso de revisión consideren lo siguiente:

- La revisión de cada actividad es en formato **todo o nada**. No hay puntaje parcial por tener partes de la actividad o algo que este *"casi funcionando"* o *"que le falte poquito"*.
- A parte de verificar la funcionalidad, durante la revisión de la actividad deberá mostrar los reportes del proceso de síntesis lógica. Debe entender los mensajes entregados por la herramienta para estas etapas.
- Mostrar el diagrama de bloques y no tener latches en su diseño son requisitos para todas las actividades. La actividad no se revisará si no se cumple lo anterior.

- Las actividades son revisadas en el orden que están planteadas. No se asignará puntaje a una actividad si no se ha completado la anterior.
- Deben seguir las especificaciones de diseño. Si se dice que las entradas son P y Q, no entregue su diseño con entradas A y B. Para los requerimientos que no están explícitamente indicados en el enunciado, puede tomar las decisiones que estime conveniente. Estas decisiones deben quedar documentadas como comentarios en el código, y deben ser consecuentes con ellas.
- Durante la sesión deberá mostrar al staff de su paralelo las actividades funcionando y deberá responder preguntas sobre ello. Cualquier integrante del grupo debe ser capaz de responder las preguntas. Si las respuestas no son satisfactorias, no se revisará la actividad.
- La revisión realizada durante la sesión solo tiene caracter temporal o de primer filtro. Luego de terminar la revisión durante la sesión, debe enviar los archivos fuente de su proyecto al profesor del paralelo. La evaluación final de cada actividad estará sujeta a una verificación funcional exhaustiva que se realizará posterior a la sesión.
- La verificación exhaustiva se revisará en base al comportamiento de entrada salida del módulo principal. **Siga cuidadosamente las especificaciones para el módulo principal, usando exactamente el mismo nombre del módulo y los mismos nombres de entrada y salida para los pines especificados.** Dentro del módulo puede tomar las decisiones de diseño en término de señales internas y estructura de los submódulos internos, siempre y cuando no afecten el comportamiento de entrada/salida.
- Para que una actividad esté correcta, **esta debe estar correcta en términos de funcionalidad lógica, apegarse a las especificaciones, y tener las salidas con las polaridades correctas. Cualquier diferencia en cuanto a especificaciones y requerimientos, hará que su diseño esté incorrecto.**
- Si alguna actividad no pasa la verificación exhaustiva, se considerará como no entregada a tiempo. En este caso deberá mostrarla corregida en la próxima sesión, aplicando los descuentos por atraso como está especificado en el reglamento.
- Más detalles sobre el proceso de evaluación de experiencias se pueden ver en el reglamento de evaluación de la asignatura.

3. Indicaciones específicas para las actividades de la sesión.

Para la implementación de sus módulos, evalúe detenidamente las siguientes indicaciones. El no cumplir con estas especificaciones hará que su descripción esté incorrecta:

- Los módulos top deben seguir exactamente los nombres especificados en los templates.
- Para señales multibit, debe seguir exactamente el orden especificado en los comentarios.
- Para efectos de esta evaluación, considere un reloj global para toda la lógica, sin necesidad de incorporar divisores de reloj[(1)].

[1]Para efectos de implementación final en un sistema físico, si deberá ajustar estos valores para operar correctamente. Sin embargo, estos ajustes harían la simulación extremandamente lenta

- No modifique el valor por defecto de ciclos en el debouncer para considerar una presión válida del botón. Mantenga el valor $N = 10^{(2)}$.

Para la entrega de sus archivos para revisión, considere las siguientes indicaciones. El no cumplir con estas especificaciones hará que su entrega no sea válida:

- Incluya todos los archivos de diseño necesarios en formato SystemVerilog (.sv) en una carpeta por actividad. Incluya solo los archivos de diseño de hardware sintetizable, verificando que incluye todos los módulos y submódulos para cada actividad. No es necesario incluir sus testbenches o archivos adicionales de simulación (el staff utilizará unos propios para la validación).
- El archivo asociado al top module para cada actividad debe tener el mismo nombre del módulo indicado en el template entregado. La cantidad y nombre de archivos de submódulos quedan a su criterio.
- Una vez haya verificado que su carpeta principal contiene todo lo necesario, empaquete todo en un solo archivo .zip.
- En la sesión se le entregara un enlace para que pueda subir su archivo entregable, el cual será revisado en detalle posterior a la sesión.

4. Actividades evaluadas.

4.1. Calculadora notación polaca inversa. (40 puntos)

Integre lo realizado en las sesiones anteriores para implementar una calculadora básica que opere mediante notación polaca inversa. Toda la funcionalidad principal debe integrarse dentro del siguiente módulo:

```
1
2  module Act1_RPCalculator
3  #(parameter N_debouncer = 10)
4  ( input  logic        clk, resetN, //resetN proviene de CPU_reset de la tarjeta
5    input  logic        Enter,
6    input  logic [15:0] DataIn,
7    output logic [3:0]  Flags, // Flags[3]=N, Flags[2]=Z, Flags[1]=C, Flags[0]=V
8    output logic [15:0] ToDisplay,
9    output logic [X:0]  CurrentState //Valor de X depende de su cod. de estados.
10   );
```

La Figura 1 muestra un diagrama de alto nivel como referencia de la funcionalidad a implementar. **Esto es solo una referencia de alto nivel, puede modificar el diagrama (por ejemplo, agregar estados o señales internas) y la lógica dentro del módulo principal a su criterio**, siempre considerando las siguientes especificaciones:

- Los operandos y el resultado son de 16 bits. **El resultado y los flags deben seguir el mismo comportamiento de la ALU entregada como referencia posterior a la sesión evaluada.**

[2]En la misma línea anterior, ese valor es suficiente para comprobar funcionalidad. En la práctica, el valor de N puede estar en el orden de cientos de miles de ciclos de reloj, lo que haría eterna la simulación.

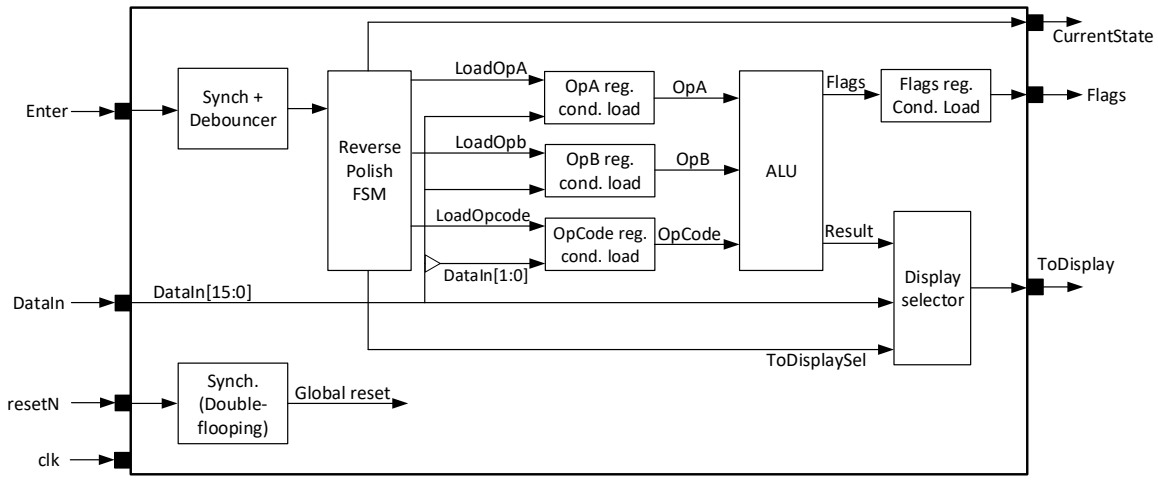

Figura 1: Diagrama de alto nivel de referencia para la calculadora con notación polaca inversa. Puede modificar señales y/o módulos, respetando los requerimientos funcionales.

- Toda la lógica secuencial sincrónica usa como reloj base la señal `clk` (por simplicidad, las conexiones del reloj no se muestran en el diagrama).
- Asuma que las señales `Enter` y `resetN` son generadas mediante pulsadores electromecánicos externos, por lo cual deben estar debidamente sincronizadas en base al reloj `clk` (mediante un *double flopping*). Además, el botón `Enter` debe pasar por un filtro anti-rebotes, de forma que por cada pulsación intencional del botón correspondiente se genere un pulso limpio que se usa como entrada para la lógica sincrónica. El parámetro `N_debouncer` del módulo especifica el número mínimo de ciclos que la señal de entrada desde el pulsador debe mantenerse estable para detectarlo como un pulso válido. Se reitera que para efectos de simulación, mantenga este parámetro en su valor por defecto $N = 10$.
- Los operandos y el Opcode que ingresan a la ALU provienen de bancos de registros con carga condicional, cuyas señales de carga provienen de una máquina de estados.
- La señal `resetN` genera una señal reset global para toda la lógica dentro del módulo. **Asuma que esta señal viene con lógica negada**. Cuando ocurra un pulso de reset, todos los valores almacenados en cualquier registro deben tomar el valor 0.

El diagrama entregado es solo referencial, y puede realizar los cambios que estime necesarios a nivel interno para lograr el comportamiento requerido. **Recuerde que debe mantener la interfaz y pines del módulo principal.**

La Figura 2 muestra un diagrama referencial para la secuencia de acciones definidas en la notación polaca inversa. Modifique este diagrama como estime conveniente (puede agregar o quitar estados y señales) para implementar la funcionalidad del módulo `Reverse Polish FSM`, el cual debe operar de acuerdo a lo siguiente :

- El estado `Wait_OPA` permite ingresar el valor del primer operando de la ALU. Mientras la FSM se encuentra en este estado, la señal `toDisplay` debe mostrar el valor de la señal `DataIn`. Al

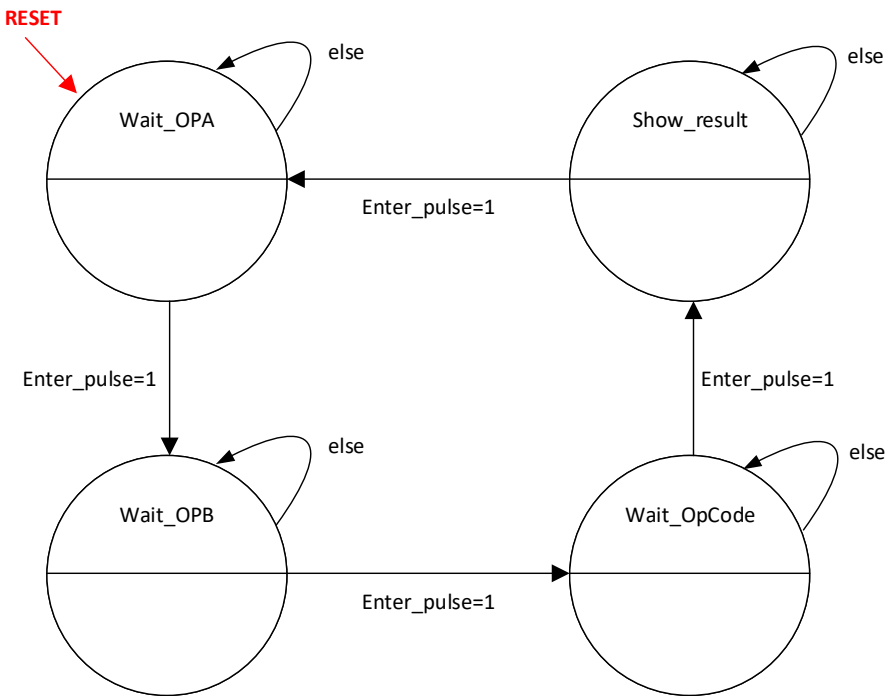

Figura 2: Base de máquina de estados para funcionalidad de calculadora en notación polaca inversa.

detectarse un pulso de presión del botón `Enter`, el valor de `DataIn` debe quedar registrado en un banco de registros cuya salida se conecta en forma directa al primer operando de la ALU.

- El comportamiento anterior también aplica para el estado `Wait_OPB`.
- El estado `Wait_OpCode` permite ingresar el código para la operación a realizar. La codificación para el OpCode es la siguiente: (a) `DataIn`[1:0]= 00: suma; (b) `DataIn`[1:0]= 01: resta; (c) `DataIn`[1:0]= 10: OR bit a bit; (d) `DataIn`[1:0]= 11: AND bit a bit. Mientras la FSM se encuentra en este estado, la señal `toDisplay` debe mostrar el valor de la señal `DataIn`. El código de la operación queda registrado al momento de ocurrir una pulsación del botón `Enter`.
- El estado `Show_result` muestra el resultado de la operación especificada en los pasos anteriores. La señal de salida asociada a los Flags debe mostrar en todo momento los flags obtenidos en la última operación realizada. **El valor de los Flags solo se actualiza cuando se llega al estado `Show_result` o bien se detecta un reset global.**
- La salida `CurrentState` indica en que estado se encuentra actualmente la máquina de estados. El ancho y valores de esta señal dependerán del número y la codificación de estados utilizada en su diseño (abierto a decisión de diseño).

4.2. Agregar función UNDO. (40 puntos)

Modifique el diseño obtenido en la actividad anterior para integrar la función UNDO, la cual permite retroceder en la secuencia establecida en la notación polaca inversa para hacer alguna modificación en los datos ingresados.

Para esto, modifique la interfaz del módulo principal de acuerdo a lo siguiente:

```
1
2  module Act2_RPCalculator
3  #(parameter N_debouncer = 10)
4  ( input  logic         clk, resetN,
5    input  logic         Enter, Undo,
6    input  logic [15:0]  DataIn,
7    output logic [3:0]   Flags,
8    output logic [15:0]  ToDisplay,
9    output logic [X:0]   CurrentState
10   );
```

Asuma que la señal `Undo` también proviene de un pulsador mecánico, por lo que debe ser debidamente sincronizada y filtrada para generar un pulso cada vez que se presione.

Modifique su máquina de estados para que, estando en un estado de la notación polaca inversa, sea posible devolverse al estado anterior. Cada uno de los estados debe mantener la misma funcionalidad especificada en la actividad anterior. Por ejemplo, si la máquina se encuentra en la etapa de ingreso del OpCode, al presionar el botón `Undo` esta debe volver al estado de ingreso del operando B. Esto permitiría actualizar el valor del operando B, sin modificar el valor del operando A que ya había sido ingresado. Siempre debe ser posible devolverse desde un estado al estado anterior, con la excepción del estado `Wait_OPA`. Si la máquina se encuentra en el estado `Wait_OPA`, solo es posible moverse al estado `Wait_OPB`.

4.3. Agregar interfaz para visualización de números en display. (20 puntos)

Una vez terminadas las actividades anteriores, agregue a su calculadora la capacidad de enviar los números de operandos y resultados en formato de visualización para el display de 7 segmentos disponible en la tarjeta Nexys4 DDR. Para esto, debe agregar la funcionalidad para el bloque mostrado en el cuadrado rojo en la Figura 3. Notar que, habiendo seguido un diseño modular, solo necesitaría agregar módulos adicionales a la salida, sin requerir cambios en los diseños anteriores.

Para esta actividad, utilice el siguiente módulo:

```
1
2  module Act3_RPCalculator
3  #(parameter N_debouncer = 10)
4  ( input  logic           clk, resetN,
5    input  logic           Enter, Undo, DisplayFormat,
6    input  logic   [15:0]  DataIn,
7    output logic   [3:0]   Flags,
8    output logic   [X:0]   CurrentState,
9    output logic   [6:0]   Segments, //Segments packed as Segment[0]=CA, Segment
       [1]=CB, Segment[2]=CC, etc. (see datasheet)
10   output logic   [4:0]   Anodes , //Anodes packed as Anodes[0]=AN0, Anodes[1]=AN1
       , etc. (see datasheet)
11   );
```

La señal `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). Asuma que esta señal proviene de un switch de los disponibles en la tarjeta, el cual puede cambiar su valor en cualquier instante. Para convertir el número binario a una represen-

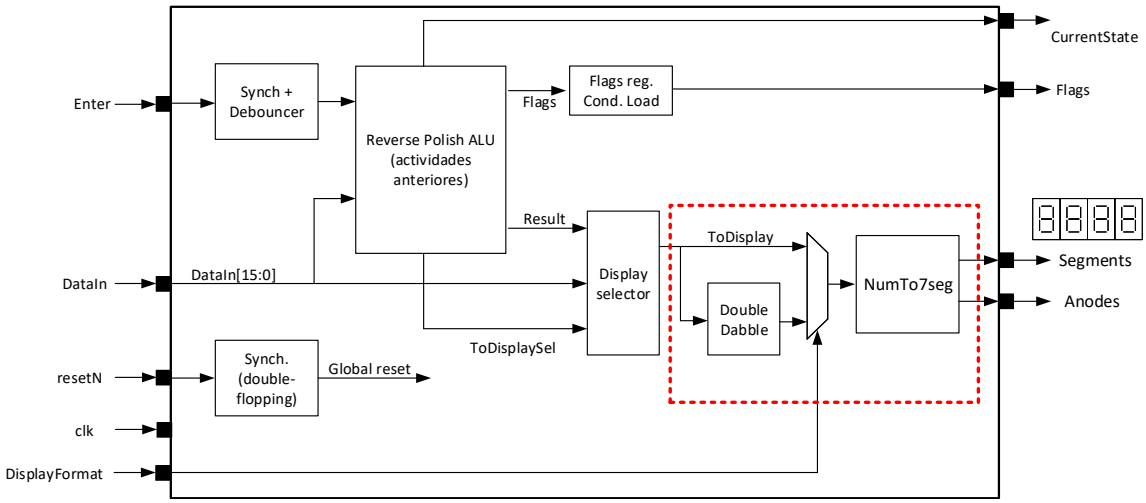

Figura 3: Calculadora con capacidad para mostrar el valor decimal de los operandos y resultados en displays de 7 segmentos.

tación decimal, debe investigar y utilizar el algoritmo *Double Dabble*. En el repositorio del curso se encuentra la carpeta *double-dabble-32bits*, el cual contiene una descripción en Verilog de este algoritmo que puede estudiar y utilizar en su diseño, o bien usarlo como base para hacer su propia versión (recurde: *Trust but verify*). Reutilice sus diseños de sesiones anteriores para implementar el módulo de conversión de la representación binaria a display de 7 segmentos. Notar que al operar con números de 16 bits, necesitará 4 displays para mostrar el valor hexadecimal, y 5 displays para el valor decimal sin signo.

Verifique cuidadosamente que las salidas hacen match con lo indicado en el enunciado y el datasheet (sobretodo al determinar las polaridades de las señales requeridas). Para efectos de validación funcional, 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)[3].

[3] Ver comentarios anteriores.