ELO212: Laboratorio de Sistemas Digitales
Guía de Actividades Sesión 7 (Evaluada)

5 al 9 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.
- Investigar en que consiste la notación polaca inversa (`https://mathworld.wolfram.com/ReversePolishNotation.html`) utilizada en varios sistemas computacionales, incluyendo calculadoras comerciales.

2. Objetivos

- Utilizar los conceptos de máquinas de estado para añadir interfaces que permitan interactuar con la ALU para implementar una calculadora básica con notación polaca inversa, agregando funcionalidad a la ALU desarrollada en sesiones anteriores de forma incremental.

3. Indicaciones generales.

Las actividades planteadas en esta guía deben ser entregadas durante la sesión de laboratorio correspondiente a su paralelo. 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. Idealmente traiga todo listo y llegue a la sesión a responder la interrogación y entregar.

Recuerden que todo diseño debe empezar con un plan y una descripción sobre qué 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). En esta sesión, también cobran relevancia los diagramas de estado. Una vez tengan los diagramas, su problema está en gran parte resuelto.

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** sobre el que se pueda discutir. El pensar que *"tengo*

todo en mi cabeza, solo necesito unos minutos para escribirlo y todo debería funcionar", es un salto directo al fracaso.

3.1. Sobre el proceso de revisión de actividades.

Para el proceso de revisión consideren lo siguiente:

- La revisión y evaluación de cada una de las actividades se realizará en dos etapas: (i) una validación básica de requerimientos funcionales mínimos durante la sesión, y (ii) una verificación exhaustiva de requerimientos funcionales y no funcionales posterior a la sesión.
- Durante la sesión, avise al staff cuando complete una o varias actividades para que muestre y explique sus resultados, y responda las preguntas que se le realizarán sobre sus diseños y conceptos asociados. Recuerde que no se atenderán consultas ni se revisará si no cuenta con un diagrama de alto nivel del sistema o el diagrama de estados del sistema que está describiendo.
- Como parte de las preguntas se le puede solicitar que muestre los reportes del proceso de síntesis lógica. Debe entender y poder explicar los mensajes entregados por la herramienta para estas etapas. Para optimizar los tiempos de atención, procure tener las simulaciones y el reporte de síntesis listos antes de llamar a alguien del staff.
- Cualquier integrante del grupo debe ser capaz de responder las preguntas. Si las respuestas no son satisfactorias, o algún integrante no es capaz de responder, no se revisará la actividad hasta que demuestren que TODOS los integrantes del grupo demuestren un entendimiento mínimo sobre lo que están entregando.
- 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.
- El staff solo responderá consultas técnicas específicas en base a lo que Ud. está mostrando para asegurarse que cumpla con requisitos mínimos para ser evaluado. No se les corregirán sus diseños ni responderán preguntas del tipo *¿me pueden decir si está bien lo que hice para poder entregarlo?* Ustedes deciden cuando su código esté listo para ser entregado, y es su responsabilidad verificar que su entrega cumple con todas las especificaciones indicadas en la guía.
- 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.
- Durante la sesión, el staff marcará en una planilla cuando Ud. considere una actividad terminada y haya pasado la interrogación. El tener la actividad marcada como revisada en la sesión no entrega puntaje, y solo asegura que cumple con requerimientos mínimos para que entregue sus archivos de diseño y pasar a la siguiente etapa de revisión mediante tests exhaustivos posterior a la sesión. El formato de entrega de sus archivos se indica en la sección 3.2.
- La verificación exhaustiva se revisará en base al comportamiento de entrada-salida del módulo principal (*top module*) de cada actividad. **Siga cuidadosamente las especificaciones para el módulo principal, usando exactamente los mismos nombres de entrada y salida para los**

pines especificados. Si la guía especifica que la entrada es P, no entregue su diseño con entrada A o p. Si considera que hay requerimientos que no están explícitamente definidos en la guía, puede tomar las decisiones que estime conveniente. Estas decisiones deben quedar documentadas como comentarios en el código, y deben ser consecuentes con ellas. Notar que una decisión de diseño, como su nombre lo indica, es una *decisión*, y como tal debe considerar sus efectos y consecuencias. Notar que esto es distinto a dejar cosas inespecificadas en el diseño para dejar que la herramienta lo interprete como estime conveniente.

- El test exhaustivo se realizará mediante testbenches y scripts avanzados que verifican el correcto comportamiento de su diseño en forma automatizada. Para que esto funcione, su diseño debe estar correcto en términos de funcionalidad lógica, apegarse a las especificaciones, y tener las salidas con las polaridades correctas en base a lo indicado en la guía. El procedimiento de revisión incluyen un paso por herramientas de detección de similaridad y plagio en códigos. Se recuerda que el plagio y comportamiento deshonesto serán tratados con la severidad que amerite el caso.

- La salida del test exhaustivo es una señal de `PASS` o `FAIL`. Una salida `PASS` indica que cumple con las especificaciones, y la actividad estará correcta y recibirá el puntaje correspondiente. Una salida `FAIL` indica que no se cumple alguna de las especificaciones funcionales o no funcionales (nombre de puertos y señales distintos a los indicados, lógica opuesta a los requerimientos eléctricos, inferencia de latches, error en la entrega o formato de archivos, etc.), y su actividad estará incorrecta. Se reitera que la revisión preliminar realizada por el staff durante la sesión es para comprobar funcionalidad mínima y verificar que entiende lo que está entregando, y no es una indicación de que su actividad está completamente correcta. **Es su responsabilidad verificar que se cumplen las especificaciones funcionales y no funcionales**.

- Una actividad marcada como `FAIL` en el test exhaustivo contará como no entregada a tiempo, cualquiera sea la razón del fallo. Esto enfatiza la revisión **todo o nada**. **No hay puntaje parcial por tener partes de la actividad** o algo que está *"casi funcionando"*, *"que le falte poquito"*, o *"sólo me equivoqué en el nombre de un pin"*.

- La revisión exhaustiva se realizará sobre los archivos que entregó dentro de la sesión. Es su responsabilidad asegurarse de que se enviaron todos los archivos necesarios en sus versiones correctas. No se aceptan reclamos del tipo o *"me equivoqué en los archivos que subí, pero ahora puedo enviar los correctos"*, o *"pero el ayudante o el profe me habían dicho durante la sesión que estaba bueno"*.

- Si una de sus actividades es evaluada como `FAIL` en el test exhaustivo, esta actividad contará como no entregada a tiempo. Este resultado le será informado y deberá entregar la actividad corregida en la siguiente sesión, sea esta evaluada o guiada, aplicando los descuentos por atraso como están definidos en el reglamento.

3.2. Indicaciones de formato para entrega.

Como se mencionó previamente, la evaluación constará de dos partes. La segunda parte constará de una verificación exhaustiva de sus diseños, la cual se realizará sobre los archivos fuente que deberá entregar vía Aula USM, los cuales deben seguir estrictamente las especificaciones que se le indican a continuación:

1. Debe crear una carpeta con el nombre *GPPXX_Sesion7*, 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_Sesion7_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 *S7_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 $P \neq p$).

En el repositorio de clases encontrarán una cápsula que expone de manera práctica cada uno de los pasos mencionados anteriormente. Para facilitar su trabajo, en Aula quedará disponible una carpeta base con la estructura y snippets de módulo top para cada actividad. Puede usar esta carpeta para agregar sus archivos fuente y hacer la entrega. 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 información, puede revisar el siguiente documento: `http://www.sunburst-design.com/papers/CummingsSNUG2006Boston_SystemVerilog_Events.pdf`

4.1. Calculadora notación polaca inversa. (34 puntos)

En esta actividad deberá extender lo realizado en las sesiones previas para la ALU con entradas registradas, agregando una máquina de estados que genere las señales de carga para cada banco de

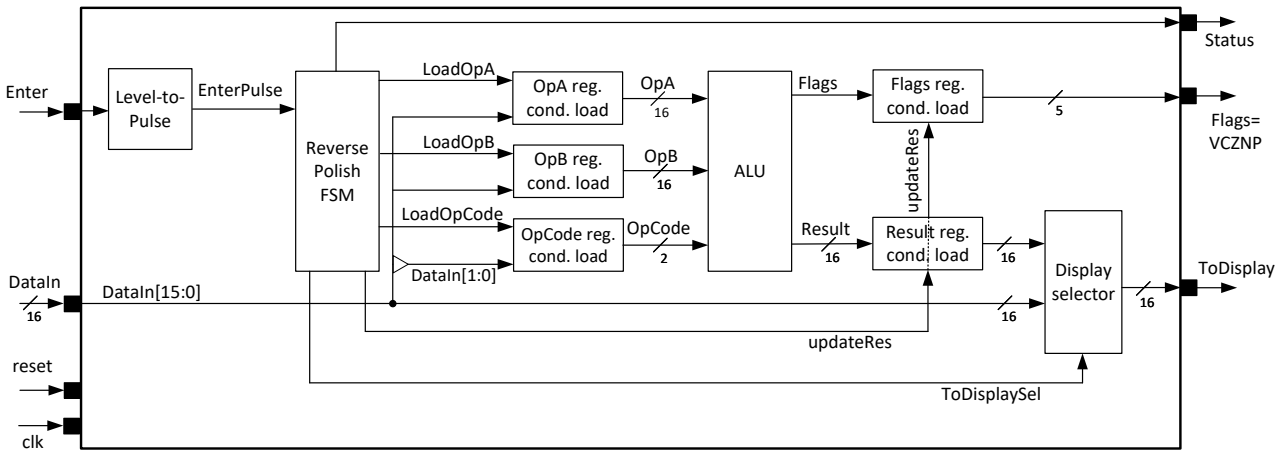

Figura 1: Diagrama de alto nivel para funcionalidad básica de calculadora. Los bloques son propuestos, y puede organizarlos internamente como estime conveniente, cuidando que las señales de entrada y salida cumplan con las especificaciones.

registros en base a una secuencia de pasos ejecutados en cierto orden. Considere que el sistema se implementará en la tarjeta Nexys4DDR/NexysA7, utilizando los switches para setear el bus de datos de entrada `DataIn` de 16 bits, y además un botón `Enter` para indicar el registro de los datos.

Toda la funcionalidad principal debe integrarse dentro del siguiente módulo:

```
module S7_actividad1 #(
  input  logic         clk,
  input  logic         reset,
  input  logic         Enter,
  input  logic [15:0] DataIn,

  output logic [15:0] ToDisplay,    //valor de salida para el Display
  output logic [ 4:0] Flags,  // {V,C,Z,N,P}
  output logic [ 2:0] Status      // Indica de manera secuencial el estado en el
    que se encuentra
);
```

La Figura 1 muestra un diagrama de alto nivel para la funcionalidad a implementar. Considere las siguientes especificaciones:

- Los operandos y el resultado son de 16 bits. **El resultado y los flags deben seguir el mismo comportamiento y formato de la ALU que haya sido evaluada como PASS en la última sesión evaluada.** Puede optar, si lo estima conveniente, por usar la ALU de referencia entregada posterior a la sesión.
- La señal ToDisplay corresponde al número de 16 bits que irá al controlador de display para eventualmente ser desplegado en los displays de siete segmentos de la tarjeta de desarrollo, la cual pude alternar entre `DataIn` y `Result` de la ALU dependiendo del valor `ToDisplaySel`. **En esta actividad no es necesario agregar la lógica para controlar el display, la cual será integrada al sistema posteriormente**.
- 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 la señal `Enter` es generada mediante un pulsador electromecánico externo. Considere que las pulsaciones del pulsador electromecánico se manifiestan como un valor lógico alto en la señal `Enter` con una duración arbitraria. Por simplicidad, en esta actividad se asumirán pulsadores ideales, omitiendo los efectos de rebotes y ruidos electromecánicos. El módulo `Level-to-Pulse` es similar al desarrollado en la sesión previa, y genera un pulso sincronizado `EnterPulse` de un ciclo de reloj cada vez que la señal `Enter` cambia de bajo a alto.
- Los operandos y el Opcode que ingresan a la ALU provienen de bancos de registros con carga condicional, cuyas señales de carga (`load`) provienen de la máquina de estados.
- La señal `reset` genera una señal reset sincrónica global para toda la lógica dentro del módulo. Cuando la señal de reset se encuentre en alto en un canto de reloj, todos los valores almacenados en cualquier registro deben tomar el valor 0 (reset sincrónico).

La Figura 2 muestra un diagrama de estados referencial para la secuencia de acciones definidas en la notación polaca inversa. Complete las transiciones y salidas de este diagrama y úselo para implementar el módulo `ReversePolishFSM`, el cual debe operar de acuerdo a las siguientes especificaciones:

- El estado `Entering_OpA` permite setear el valor del primer operando de la ALU. Mientras la FSM se encuentra en este estado, la señal de salida `ToDisplay` debe mostrar el valor instantáneo de la señal `DataIn`, es decir, el valor de la señal debe actualizarse a medida que el usuario mueva los switches de la tarjeta. Al detectarse la presión del botón `Enter`, la FSM cambia al estado `Load_OpA`.
- El estado `Load_OpA` permite generar un pulso **de exactamente un ciclo de duración** para almacenar en el banco de registros correspondiente el valor de `DataIn` al momento en que se presionó `Enter`.
- Los estados `Entering_OpB` y `Load_OpB` siguen la misma lógica anterior para setear y almacenar el operando B.
- El estado `Entering_OpCode` permite ingresar el código para la operación a realizar utilizando los dos bits menos significativos del bus `DataIn`. La codificación para el OpCode es la siguiente: (a) `DataIn`[1:0]= 00: OR negado bit a bit; (b) `DataIn`[1:0]= 01: AND negado bit a bit; (c) `DataIn`[1:0]= 10: suma; (d) `DataIn`[1:0]= 11: resta. Mientras la FSM se encuentra en este estado, la señal `ToDisplay` debe mostrar el valor de la señal `DataIn`. El estado `Load_OpCode` registra la operación en el banco de registros correspondiente siguiendo el mismo procedimiento que los estados `Load_OpA` y `Load_OpB`.
- El estado `Show_Result` actualiza los flags y el resultado en base a los últimos datos ingresados en los en los pasos anteriores. Mientras se está en este estado, la señal `ToDisplay` toma el valor del resultado de la ALU obtenido a partir de los últimos datos ingresados. La señal de salida asociada a los Flags de mostrar en todo momento los flags obtenidos en la última operación realizada; es decir, este valor solo se actualiza cuando se llega al estado `Show_Result`

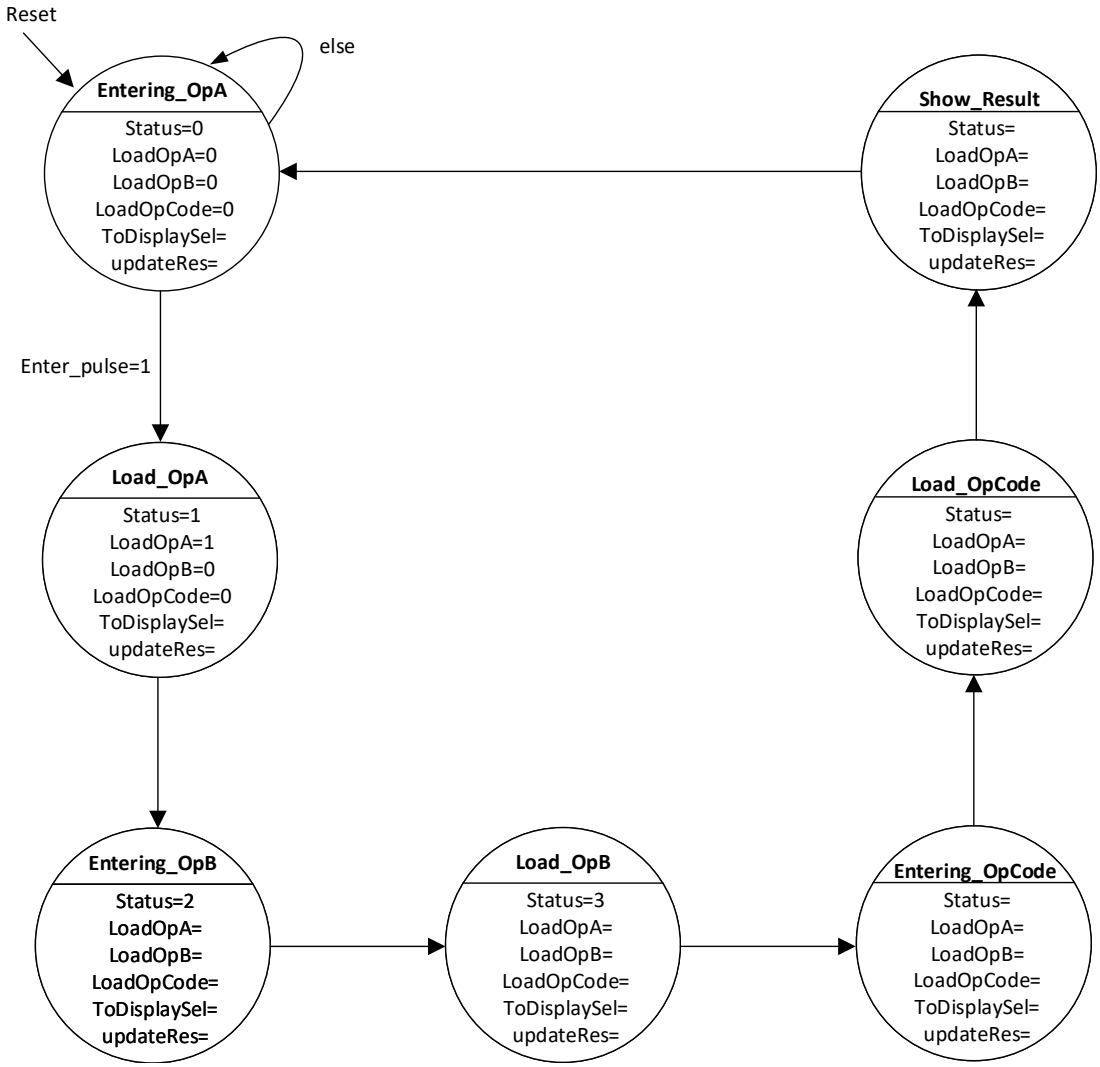

Figura 2: Esqueleto de diagrama de estados de Moore para funcionalidad de calculadora en notación polaca inversa. Antes de hacer la descripción en HDL, debe completar los valores de salida y condiciones de transición para cumplir con el comportamiento especificado.

o bien se detecta un reset global. Cuando se está en este estado, la presión del botón `Enter` lleva al estado `Entering_OpA`, tras lo cual se puede ingresar una nueva secuencia de datos para una nueva operación.

- La salida `Status` corresponde a un número correlativo que indica en que paso o estado de la secuencia se está en un instante dado.

4.2. Agregar funcion UNDO. (34 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:

```
module S7_actividad2 #(
  parameter N_DEBOUNCER = 10,
)(
  input  logic         clk,
  input  logic         reset,
  input  logic         Enter,
                       Undo,
  input  logic [15:0] DataIn,

  output logic [15:0] ToDisplay,   //valor de salida para el Display
  output logic [ 4:0] Flags,  // {V,C,Z,N,P}
  output logic [ 2:0] Status     // Indica de manera secuencial el estado en el
    que se encuentra
);
```

Asuma que la señal `Undo` también proviene de un pulsador electromecánico similar al de la señal Enter, por lo que debe pasar por un módulo `Level-to-Pulse`.

Modifique su máquina de estados para que, estando en un estado de la notación polaca inversa, sea posible devolverse al estado de ingreso de datos previo. Cada uno de los estados debe mantener la misma funcionalidad especificada en la actividad anterior. Por ejemplo, si la máquina se encuentra en el estado `Entering_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 `Entering_OpA`. Si la máquina se encuentra en el estado `Entering_OpA`, solo es posible moverse al estado `Load_OpA`.

4.3. Agregar interfaz para visualización de números en display. (32 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 Nexys4DDR/NexysA7. Para esto, debe agregar su módulo para el manejo de displays diseñado y verificado en las sesiones previas. Notar que, habiendo seguido un diseño modular, solo necesitaría agregar módulos adicionales a la salida, sin requerir cambios en los diseños de las actividades anteriores.

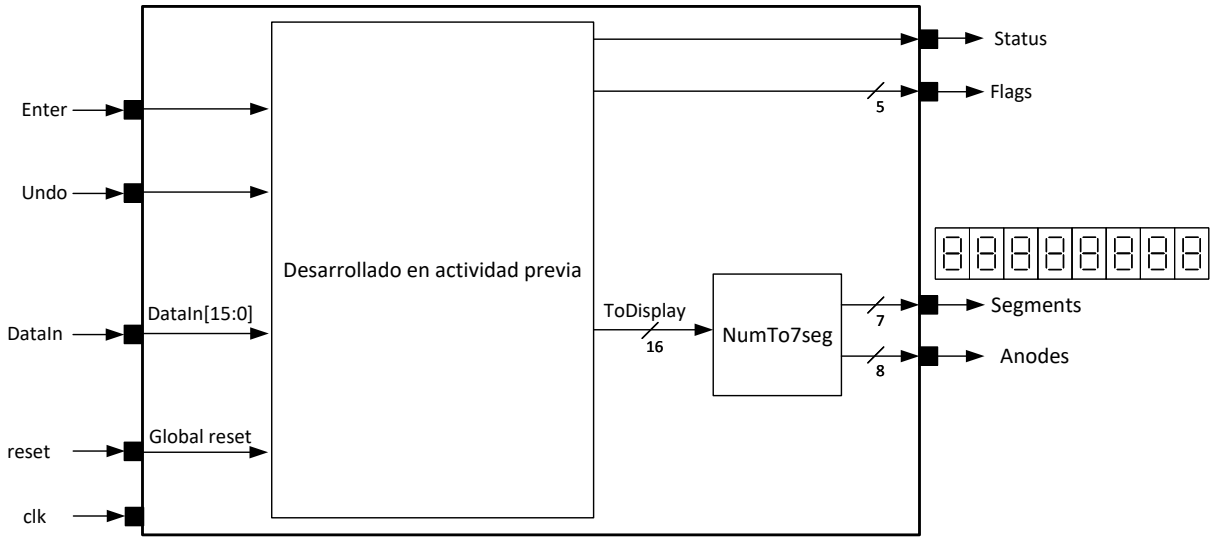

Figura 3: Base de máquina de estados para funcionalidad de calculadora en notación polaca inversa incluyendo valores de salida para cada estado.

Para esta actividad, utilice el siguiente módulo top:

```
module S7_actividad3 #(
  input  logic        clk,
  input  logic        reset,
  input  logic        Enter,
                      Undo,
  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
);
```

En este caso, debe utilizar 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. Por ejemplo, si el resultado es 0x32, entonces el display debe mostrar 0x00000032 (solo el valor numérico, omitiendo el 0x).

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 durante la operación normal.