ELO212: Laboratorio de Sistemas Digitales
Guía de Actividades 9 (Evaluada)

26 al 30 de junio de 2023

1. Requisitos de entrada.

- Haber **completado y entendido** las actividades de todas las sesiones anteriores.
- Contar con un repositorio personal de módulos diseñados, simulados, y verificados de las sesiones previas.
- Haber estudiado sobre el protocolo UART y revisado las cápsulas de video asociadas a la sesión guiada previa.
- Saber utilizar un analizador lógico.

2. Objetivos

- Continuar con la aplicación de conceptos de diseño y descripción de máquinas de estado.
- Experimentar con protocolos de comunicación serial utilizando una interfaz UART.
- Revisar algunos conceptos de sistemas distribuidos y co-procesadores.
- Aprender técnicas para entender y adaptar códigos de terceros para ser reutilizados en sus diseños.

3. Indicaciones generales.

Esta sesión será evaluada en dos partes: (i) un trabajo previo a la sesión de laboratorio, el cual deberá entregar como informe breve al iniciar la sesión evaluada; y (ii) las actividades prácticas que deberá mostrar al staff junto con responder una interrogación asociada 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 mostrar sus resultados.

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). Desde las sesiones previas también cobran relevancia los diagramas de máquinas 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 a sufrimiento y frustración innecesarios.

3.1. Sobre el proceso de revisión de actividades prácticas.

Asumiendo que, a partir del trabajo realizado en las sesiones previas, ya tienen asimilados los pasos del proceso de diseño y verificación funcional, en esta ocasión nos enfocaremos en la revisión del trabajo asociado a la implementación física. En este contexto, la revisión se basará en ver su sistema funcionando en la tarjeta de desarrollo que cada grupo tiene en su poder. No deberán subir archivos de diseño. Si bien no habrá evaluación mediante simulación exhaustiva, se recuerda que la simulación funcional es una parte fundamental del proceso de diseño y deben estar list@s para mostrarla en caso que se le solicite.

Considere las siguientes indicaciones:

- La revisión y evaluación de cada una de las actividades se realizará en una sola etapa que considera mostrar los sistemas funcionando en la tarjeta de desarrollo y una interrogación asociada a esta. Para obtener una actividad marcada como correcta, el sistema debe funcionar de acuerdo a las especificaciones y debe responder en forma satisfactoria la interrogación.
- 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 estos reportes accesibles 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 tienen 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.
- 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.

- Cualquier falla en el funcionamiento del sistema implementado o la interrogación deja la actividad como incompleta hasta que cumpla con los requisitos. Si no alcanza a completar una actividad dentro de la sesión de su paralelo, entonces contará como atraso y deberá mostrarla posteriormente con la penalización correspondiente. 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 *"que funciona a veces"*.
- Ante cualquier indicación que considere ambigua o que no tenga una especificación explícita, puede tomar las decisiones de diseño que estime conveniente. Estas deberá explicarlas y justificarlas durante la interrogación.

4. Introducción a protocolos de comunicación serial.

En los diseños desarrollados hasta ahora, nos hemos limitado a describir, simular, y analizar sistemas computacionales muy simples cuyos módulos están contenidos en un mismo pedazo de silicio. Sin embargo, en la práctica, la mayoría de los sistemas digitales modernos se basan en enfoques distribuidos en los que múltiples dispositivos físicamente separados intercambian información en forma coordinada para realizar una tarea en conjunto.

El intercambio efectivo de información entre dos entidades requiere que los participantes de la comunicación establezcan una base común de reglas que permitan interpretar correctamente lo que el otro intenta comunicar. Considerando el caso de la comunicación entre seres humanos, estas reglas quedan establecidas en gran parte por el idioma que hablan las partes involucradas, además de otras normas culturales. En el caso de la comunicación entre dispositivos electrónicos, el análogo al idioma son los protocolos de comunicación digital.

En esta sesión experimentaremos con conceptos fundamentales de comunicación digital utilizando una interfaz *Universal Asynchronous Receiver and Transmitter* (UART). La UART se utiliza masivamente para **comunicación serial asincrónica entre dispositivos**. Si bien esta interfaz es relativamente antigua y ofrece una tasa de transmisión de datos bastante baja en comparación a otros protocolos estándar actuales (USB, Ethernet, etc.), su simplicidad y bajo costo la mantienen como una de las interfaces más usadas para aplicaciones embebidas en la actualidad [1]. Por otro lado, a pesar de su simplicidad, los conceptos técnicos para el funcionamiento de la UART proveen una excelente base para asimilar conceptos fundamentales y comenzar a trabajar con protocolos más sofisticados, ya sea en cursos posteriores o para proyectos personales.

Los siguientes enlaces entregan más detalles que complementan la información general revisada en clases previas:

- `https://learn.sparkfun.com/tutorials/serial-communication`
- `http://www.circuitbasics.com/basics-uart-communication`
- `https://en.wikipedia.org/wiki/Universal_asynchronous_receiver/transmitter`

Los detalles técnicos de bajo nivel acerca del funcionamiento interno de la UART se discuten en el material revisado previamente. Los detalles de la implementación del driver de referencia escapan

[1] De hecho, es tan común, que a veces se suele hablar de "comunicación serial" a secas para referirse a la UART.

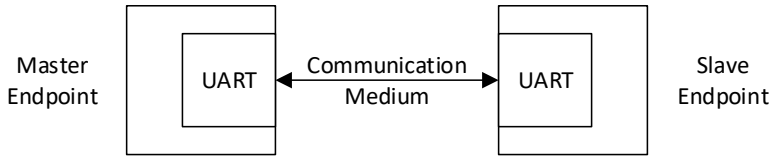

Figura 1: Diagrama de alto nivel de dispositivos distribuidos conectados por interfaz UART

de los alcances de este curso introductorio. Sin embargo, basándonos en los conceptos de modularidad y jerarquía que ya tenemos completamente asimilados a esta altura del curso, sabemos que no es necesario conocer los detalles de implementación de una componente para poder utilizarla, siempre y cuando conozcamos su comportamiento de entrada/salida (modelo de caja negra).

Para esta experiencia se proveerá como base un módulo en (legacy) Verilog de una UART, el cual está disponible en el repositorio Github del curso y se basa en el módulo provisto y descrito siguiente enlace:

`http://www.fpga4fun.com/SerialInterface.html`

Se recomienda revisar con detención el enlace anterior y complementarlo con investigación independiente para tener al menos una noción de como funciona el módulo UART entregado y entender el comportamiento de las señales de entrada y salida. Tratar de hacer ingeniería inversa al diseño es además un excelente ejercicio para repasar conceptos y aprender nuevas técnicas que podrían aplicar en sus propios diseños.

5. Actividades asociadas a la sesión.

Durante esta sesión se agregará soporte para manejo remoto a la calculadora implementada en las sesiones anteriores. Para esto, se agregará una interfaz UART que permita a la calculadora recibir los operandos y operación a realizar desde un dispositivo externo, que permita realizar operaciones sin necesidad de manipular directamente la tarjeta. La Figura 1 muestra un diagrama de alto nivel del sistema, donde se tienen dos dispositivos o *endpoints* conectados por medio de una UART. El setup representa una configuración *master/slave* [2,3,4,5], en la que el *Master Endpoint* (ME) envía datos y comandos al *Slave Endpoint* (SE). En el caso del sistema a implementar en esta sesión, el SE actuará como un co-procesador aritmético que presta servicios al procesador principal (ME). Un co-procesador no puede gatillar acciones por sí solo, sino que debe recibir instrucciones desde un procesador maestro [6].

La Figura 2 presenta un diagrama más detallado de una implementación de la configuración master/slave de la calculadora. El diagrama representa el caso de que que ambos endpoints estén implementados en FPGAs, mostrando algunos de los módulos principales asociados a la comunicación.

[2]Esta terminología va de salida por razones obvias. Sin embargo, al no haber una definición técnica formal alternativa aún, por simplicidad se usará en su contexto netamente técnico. Puede revisar artículos interesantes al respecto.

[3]`https://www.eetimes.com/its-time-for-ieee-to-retire-master-slave/`

[4]`https://www.nytimes.com/2021/04/13/technology/racist-computer-engineering-terms-ietf.html`

[5]`https://tools.ietf.org/id/draft-knodel-terminology-00.html`

[6]https://www.computerhope.com/jargon/m/mathcopr.htm

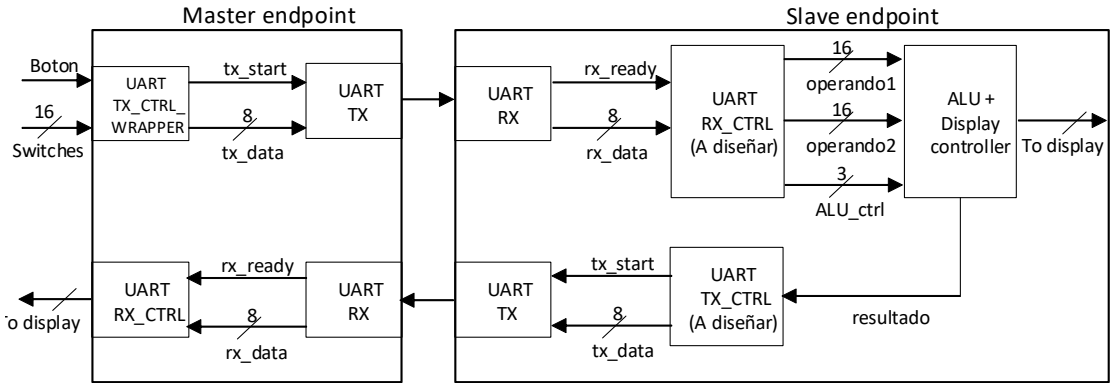

Figura 2: Diagrama interno de lógica de comunicación para nodos maestro y esclavo.

Considerando que el ME está implementado en una tarjeta Nexys4DDR, este lee los estados de switches y botones que especifican los operandos y comando para la operación a realizar, enviándolos por medio de la UART a la ALU implementada en el SE. El SE ejecuta acciones acorde a una secuencia de datos recibidos, enviando un resultado de vuelta al ME cuando se haya recibido cierto patrón de datos. Esto permite al ME usar los servicios del SE para realizar operaciones aritméticas sin necesidad de contar con una ALU en el mismo chip. El diagrama anterior es solo una referencia de alto nivel y por simplicidad omite algunas señales y niveles de jerarquía adicionales dentro de cada bloque.

Es importante notar que para que el esquema anterior funcione, ambos dispositivos deben estar configurados para operar con el mismo protocolo y los mismos parámetros. Además, las interfaces solo reciben y procesan pulsos digitales sin preocuparse de que tipo de dispositivo los está generando. Esto quiere decir que desde el punto de vista del SE, el ME puede ser una FPGA, un laptop, una tarjeta Arduino, o cualquier otro dispositivo, y este respondería de la misma manera sin necesidad de saber el tipo de dispositivo que está haciendo las peticiones ni como esté implementado, siempre que se cumpla el protocolo y las secuencias establecidas. Pueden hacer una analogía cuando hablan por teléfono con algún desconocido y pueden comunicarse perfectamente sin conocer su apariencia física, siempre que hablen el mismo idioma.

El desafío para esta sesión está enfocado en la implementación del SE propuesto en la Figura 2, el cual recibe datos desde el ME y envía un resultado de vuelta al ME. Se asume que para realizar una operación se ha definido una secuencia de comunicación de acuerdo a la configuración polaca inversa, en la que el ME envía el primer operando de 16 bits, luego el segundo operando de 16 bits, y finalmente la operación a realizar en un solo byte. Al recibir la operación, el SE realiza el cálculo y envía un resultado de vuelta al ME, volviendo automáticamente al estado inicial y quedando a la espera de otra operación. Por simplicidad, se asume que el ME siempre envía los datos en la secuencia correcta y el medio de comunicación no tiene ruido. En la práctica, la implementación de un canal de comunicación robusto requiere establecer ciertos acuerdos para verificar que el receptor está disponible para recibir los datos y que estos llegaron sin errores, pero eso es tema de cursos superiores y se omitirá por ahora por simplicidad.

5.1. Archivos de base.

Como base para que trabaje antes de la sesión, en el Github del curso está disponible la carpeta `UART_reference_design` con proyectos y archivos base para que experimente con la UART con anticipación a su sesión evaluada. Esta carpeta contiene los siguientes archivos que puede utilizar para su trabajo, tanto reutilizando módulos para su diseño como para realizar pruebas:

- **Código fuente para descripción de ME**. La carpeta `UART_ME_reference` contiene los códigos para el ME de acuerdo al esquema de la Figura 2. Debe estudiar y entender en detalle este diseño para responder las preguntas del trabajo previo, y además le servirá de base para que desarrolle el SE que será evaluado durante la sesión de laboratorio. El diseño implementa un ME que se conecta a la interfaz UART incorporada por el conector USB (la tarjeta de desarrollo incorpora un chip FTDI que permite convertir las señales recibidas por USB a al protocolo UART). Además, también se pueden utilizar para este propósito los conectores PMOD (En este caso se propone usar el conector JD. Revise el datasheet y el constraint file entregado en el proyecto. Si implementa el diseño provisto y lo carga en su tarjeta, puede implementar la comunicación por medio de los switches y el BTNC de la tarjeta, y utilizar una aplicación de terminal [7] o un analizador lógico para verificar y depurar la comunicación. Mas detalles sobre esto se entregarán en la sesión guiada correspondiente.
- **Bitfile de referencia de SE**. Se provee además el bitfile `SE_reference.bit` como referencia de sistema que debe implementar para su sesión evaluada. Este SE tiene conectados los pines UART al pin de comunicacion que usa el mismo conector USB que utiliza para alimentar y programar la tarjeta (ver datasheet). Puede programar su tarjeta con este bitfile, y podrá enviar comandos por medio de una aplicación terminal desde su computador para controlar la calculadora en forma remota. Debe configurar la terminal a 115200 bauds, 8 bits de datos, sin paridad, y 1 bit de stop. Asegúrese de que está enviando datos en formato hexadecimal (no ASCII). El objetivo para su sesión evaluada es que logre un diseño funcionalmente equivalente al provisto en este bitfile.

Nota: los archivos base provistos presentan diferencias en las especificaciones de la ALU con respecto a las que trabajó en las sesiones previas, tanto en términos de OpCodes y representación de flags. Para efectos de prueba y experimentación con los archivos base, enfóquese en los aspectos de comunicación. Para las actividades a evaluar en esta sesión, debe cumplir con el comportamiento de la calculadora trabajada en las sesiones anteriores y no reproducir el comportamiento de la referencia.

5.2. Trabajo Previo (25 puntos)

Al entrar a la sesión de su paralelo, debe entregar un informe con respuestas a las preguntas planteadas a continuación. Se esperan respuestas breves, pero concisas y contundentes. Sea prolijo en la presentación de su informe. Se descontarán puntos por presentación desprolija (figuras de mala calidad, no indicar los integrantes del grupo ni su paralelo, etc.), mala redacción, faltas gramaticales y de ortografía, etc.

[7] Por ejemplo, pueden utilizar la aplicación HTERM disponible en el siguiente enlace: `http://www.der-hammer.info/terminal/`

Al responder su parte previa para esta sesión considere los siguientes puntos:

- Un diagrama de transición de estados es una especificación técnica de un sistema que puede usarse para prender luces, como también ir en un marcapasos o controlar el lanzamiento y activación de una bomba atómica. En este sentido, no se deben dejar especificaciones al azar o ambiguas asumiendo que el usuario debería ser capaz de interpretarlas. La labor de los profesores y ayudantes es certificar que adquieren habilidades que en su vida profesional pueden tener consecuencias importantes (vean ejemplos de los sistemas Ariane V, Therac-25, Air France Flight 447, etc.), por lo que técnicamente la USM sería imputable si cometen errores de este tipo en el futuro. Puede sonar exagerado, pero es efectivamente así (sino cualquiera podría decir que es ingeniero). En este sentido, se enfatiza que no hay diagramas "casi buenos" o puntos parciales por dibujar "los circulitos y algunas flechas que quedaron para el otro lado, pero esos son pequeños detalles" [8].
- El diagrama de estados contiene toda la información técnica para describir el sistema. Si se pide una explicación breve de la funcionalidad, debe apuntarse a una descripción informativa de alto nivel en lenguaje humano. Esta debe complementar la especificación técnica, no describir letra por letra la misma información que está en el diagrama. Es muy importante que aprendan a comunicarse con gente que no es de su especialidad.

Para responder las siguientes preguntas se recomienda revisar referencias sobre operación de la UART, además de estudiar y entender el código HDL provisto para el ME que se encuentra en la carpeta `UART_reference_design/UART_ME_reference` del Github del curso.

(A) El módulo `UART_tx_control_wrapper` del diseño del ME contiene dos sub-módulos: `tx_control` y `tx_sequence`. Cada uno de estos sub-módulos describe una máquina de estados que interactúa con la otra[9]. Dibuje el diagrama de estados asociada a cada módulo, y explique de manera breve y concisa la funcionalidad de cada máquina . El diagrama debe utilizar los nombres de estados y señales usados en el código. Si no se cumple lo anterior la respuesta no se revisará y no obtendrá puntaje.

(B) Con respecto a lo desarrollado en la pregunta anterior, responda lo siguiente:

- ¿Cuál es la función de la señal `TX_start`?¿Bajo qué condiciones toma valor TRUE y cuantos segundos dura en alto?
- ¿Qué acción se realiza en el estado `REGISTER_DATAIN16` del módulo `TX_control`? ¿Por qué es necesario?
- ¿Qué función cumplen los parámetros `INTER_BYTE_DELAY` y `WAIT_FOR_REGISTER_DELAY`? ¿De que dependen sus valores?
- ¿Qué función cumple la salida busy del módulo `tx_control`?

(C) Considerando la configuración de la Figura 2, si los switches del ME están en la posición SW[15:0] = 0x38FA cuando se presiona el botón para pasar al estado `TX_ALU_CTRL` del módulo `TX_control`, ¿qué valor se transmite por la UART? Dibuje el diagrama de tiempo de los datos transmitidos usando como unidad de tiempo lo necesario para transmitir un bit.

[8] Argumentos tomados de la vida real en versiones pasadas del curso.

[9] Esto también refuerza el concepto de factorización de máquinas de estado, mediante el cual FSM complejas se particionan en múltiples FSM más simples que interactúan entre sí. Nuevamente, modularidad y jerarquía son claves para simplificar los diseños.

(D) Dibuje el diagrama de estados del bloque `UART_TX_CTRL` del SE, el cual deberá implementar y mostrar durante el laboratorio. Considere que el resultado que se transmite de vuelta al ME es de 16 bits.

5.3. Actividades prácticas evaluadas (75 puntos)

Las siguientes actividades están diseñadas para ayudarlo a describir su sistema desde una base simple e ir agregando complejidad a medida que se prueben componentes mas básicas. No se puede revisar una actividad si no se ha completado la anterior. Recuerde que la asignación de puntaje para cada actividad está condicionada a una interrogación que cualquier miembro del grupo debe ser capaz de responder. Debe entender los reportes y warnings generados por la herramienta, y no se revisará ningún diseño que contenga latches.

5.3.1. Máquina de estados base para lógica de recepción de datos (10 puntos)

Implemente el módulo `UART_RX_CTRL` del SE siguiendo como base el diagrama de estados de la Figura 3. El diagrama omite intencionalmente las salidas para cada estado, las cuales deben ser definidas por Ud. de acuerdo a los requerimientos del resto de su sistema. Como pista, se necesitan salidas en los estados `Store` que gatillen el almacenamiento y retención del dato correspondiente en un arreglo de Flip-Flops (como se vio en experiencias anteriores). El estado `Trigger_TX_result` debe generar un pulso que gatille la transmisión del resultado hacia el endpoint maestro. Este pulso debería ser una entrada a la máquina de estados a implementar en la Actividad 5.3.3. Puede agregar más salidas si lo considera necesario. En su descripción debe seguir los nombres de estados incluidos en el diagrama. Si encuentra un error en el diagrama o encuentra necesidad de definir algo adicional, debe consultarlo con el profesor o ayudantes de su paralelo.

En esta actividad se evaluará que la máquina avance por los estados correctamente cada vez que se reciba un nuevo byte desde el ME (puede utilizar su PC para enviar datos por el conector USB hacia la taejta). Agregue lógica simple para mostrar en los LEDs como se mueven los estados a medida que se reciben los datos.

No es necesario implementar la interfaz UART. Simplemente instancie el wrapper entregado en el diseño del ME. Debe ser capaz de identificar cuales son los archivos fuente necesarios y como operan las señales de entrada y salida.

5.3.2. Integración de calculadora con UART para recepción de datos (15 puntos)

Una vez verificado su diseño de la actividad anterior, agregue la lógica necesaria para almacenar los datos recibidos según la secuencia de llegada de datos, junto con los módulos de la ALU y displays en el endpoint esclavo. Cada vez que se recibe un operando, el display debe mostrar inmediatamente el valor recibido **en decimal** (note que los operandos son de 16 bits). Cuando se recibe la operación a realizar, el display debe mostrar automáticamente el resultado de la operación.

La operación a realizar se codifica usando los dos bits menos significativos del byte recibido de acuerdo a los siguientes valores: 00 = NOR bit a bit; 01 = NAND bit a bit; 10 = Suma; 11 = Resta.

5.3.3. Agregar la lógica de transmisión del resultado de la operación (20 puntos)

Agregue la lógica de transmisión del resultado de 16 bits del SE hacia el ME. Esto debe ir acorde al diseño que presentó para la parte previa. Si lo desea, puede usar como referencia la lógica de transmisión del ME (requiere algunas modificaciones).

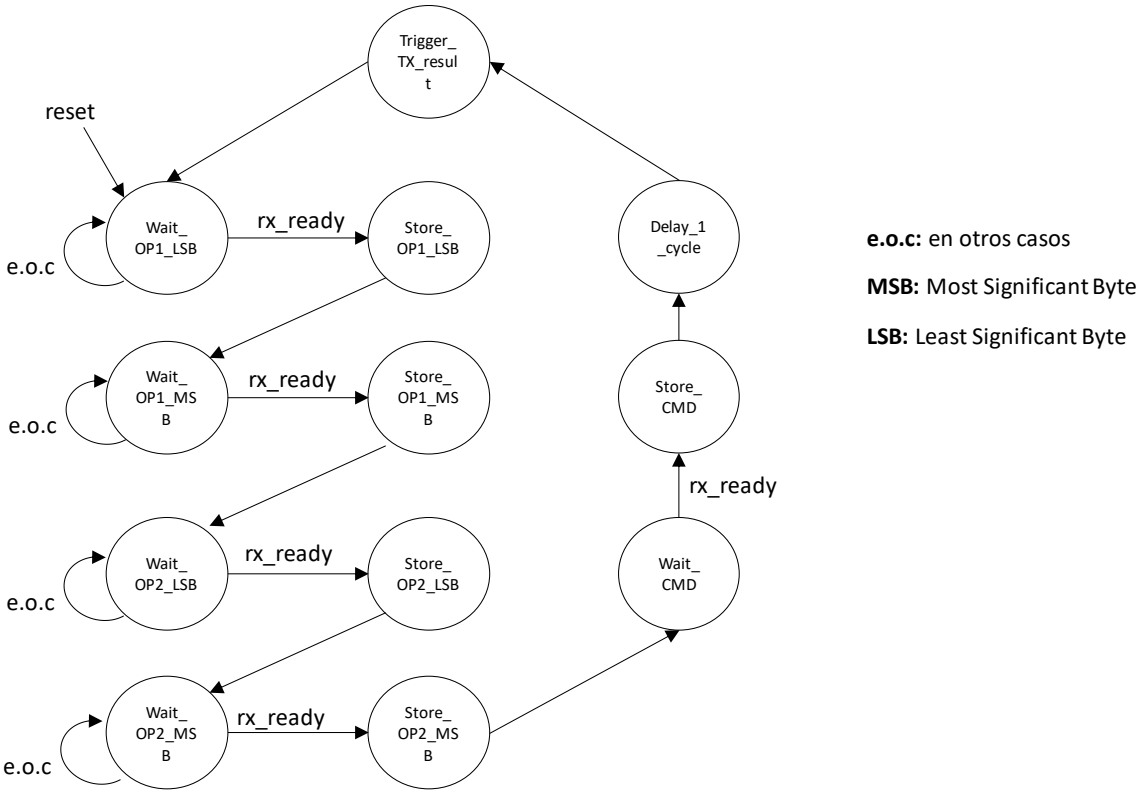

Figura 3: Máquina de estados a implementar para lógica de recepción de datos para la calculadora.

Valide el flujo completo de procesamiento, enviando datos desde su computador por medio del puerto serial virtual, verificando que el resultado calculado en el SE coincide con el recibido en el ME. Muestre su diseño completo y responda las preguntas del profesor o ayudante.

5.3.4. Capturar las señales de comunicación con analizador lógico (30 puntos)

Con su diseño probado, utilice el analizador lógico para capturar las señales digitales de comunicación entre los dispositivos. Para esto, haga un *tapping* de la señales de comunicación serial de recepción y transmisión, replicándolas para que sean también transferidas hacia un puerto PMOD en donde puede conectar el analizador lógico para visualizarlas. Esto también se conoce como un "sniffer" del canal de comunicación.