1. Diagrama de Bloques: Para procesar la información del voltaje a mostrar en el display de la Basys3, la FPGA debe procesar un bus serial de entrada y sacar valores para los displays, utilizándose un switch para indicar si se quiere visualizar el voltaje real de la señal medida o el voltaje adaptado por el circuito amplificador. Para lograr esto, se diseñó el diagrama de bloques de la **FIGURA X.**

**Figura Diagrama\_Bloques\_FPGA**

En el diagrama propuesto se utilizan distintos bloques, cada uno con una tarea específica.

* **Uart\_receptor:** Es el bloque más importante, pues el encargado de recibir correctamente la información del bus serial proveniente de la MSP en su entrada **rx**. Funciona bajo el protocolo UART con 1 bit de inicio, 8 bits de datos y 1 bit de término, sin bit de paridad. Además, el bus serial no trabaja a la frecuencia del reloj de la FPGA, por lo que necesita un divisor de reloj en forma de one-shot en su entrada **baud\_en**; el próposito de que esta entrada sea de tipo one-shot es permitir que se evalue cuando su valor sea 1 y no cuando este tenga un *rising edge.* Además, cuanta con una forma de resetearse externamente. Sus salidas son la información del bus serial en forma de vector de 8 bits **rx\_data** y una señal **rx\_rdy** para indicar que se terminó de recibir la información. Este módulo se construyó sobre un módulo de comunicación UART construido el año 2021 en el curso IEE2713 Sistemas Digitales en el Taller 13 por Santiago Larraín; el módulo está basado sobre una máquina de estados finitos, mostrada en la **FIGURA X** y el código tenía errores, por lo que se tuvo que alterar en una gran proporción. El código final corresponde al de la **FIGURA X.** Es importante resaltar que a modo de recibir bien el bit de entrada, se tiene una tasa de refresco 16 veces más rápida que la tasa de envío del transmisor; esto permite sobre muestrear la entrada y que no haya duda de esta, rescatando el valor del medio medio de cada bit (un bit de entrada es dividido en 16 trozos por la lectura 16 veces más rápida.

**Foto FSM\_receptor**

**Foto codigo\_uart\_1 y codigo\_uart\_2**

* **Voltage\_memory:** Este módulo maneja el vector de 8 bits recibido. En su entrada recibe el bus de largo 8 rx\_data que contiene la información de un número del 0 al 9 en los primeros 4 bits, luego su posición en el display en los siguientes dos bits y por último un bit que indica si pertenece a un voltaje u otro; el último bit no tiene uso. Tmbn entra la señal rx\_rdy, que indica cuándo se puede actualizar un nuevo valor. Este módulo tiene dos variables internas (V1 y V2) que serán las que guarden los dos voltajes y mediante un multiplexor gobernado por un switch (sw) se elige entre las dos variables anteriores para asignársela al valor de voltaje a mostrar en pantalla. El método de codificación es hexadecimal, por lo que las variables internas V1 y V2 y la salida voltage son de 16 bits, representando a los 4 digitos del voltaje (cada uno de 4 bits). En caso de resetear el código, V1 y V2 vuelven a 0. En la **FIGURA X** se muestra el código implementado para este módulo.

**Foto codigo\_voltage\_memory**

* **Bc7seg:** Corresponde a un módulo extraído de un proyecto reslizado el segundo semestre del año 2021 en el curso IEE2713 Sistemas Digitales por Santiago Larraín. Este módulo permite convertir un valor numérico unitario (entre 0 y 9) en un código de 7 bits para el display de siete segmentos.
* **SevenSegController:** También fue extraído de la fuente mencionada anteriormente y la función de este módulo es recibir los valores de los displays (4 digitos y un punto) y sacarlos de la FPGA hacia el display de siete segmentos de manera secuencial y rápida, pues este display no funciona de forma paralela, sino que muestra un valor tras otro de forma muy rápida, aparentando estar todo encendido a la vez para el ojo humano. Sus salidas corresponden a sseg\_ca que es el valor del número, sseg\_an que indica la posición y sseg\_dp que indica si encender o no el punto.
* **One\_shot\_clock\_divider:** Este módulo es una variación del módulo **clock\_divider**, el cual con una señal de reloj y un periodo determinado, emite una señol de clock *period* veces más larga que la señal de reloj de entrada. La versión modificada y utilizada permite emitir un pulso que dura un tiempo del reloj de entrada, pero cada *period* tiempo, sirviendo así como señal se actualización del módulo **uart\_receptor**.

**Foto codigo\_clock\_divider**

* **Debouncer:** Permite eliminar saltos o cambios erráticos al presionar un botón.

1. Ventajas de utilizar un divisor de reloj

Utilizar un divisor de reloj significa trabajar a una frecuencia más baja a la del reloj base del componente circuital. Si se intentan comunicar dos componentes distintos, ya sea el mismo componente, pero distinto distribuidor u otro dispositivo propiamente tal, lo más probable es que trabajen a frecuencias distintas, resultando en una comunicación des coordinada. Para resolver esto, se puede definir una frecuencia de comunicación en bit por segundo y a partir de ella, hacer divisores de reloj en ambos dispositivos tal que ambos tengan un nuevo reloj con la misma frecuencia para realizar comunicaciones coordinadas. Otra ventaja de hacer un divisor de reloj es que a la hora de interactuar con personas, la frecuencia de trabajo del dispositivo es brutalmente mayor a la tiempo de respuesta de un humano, por lo que se pueden hacer grandes divisores de reloj que permitan al operador ver funcionalidades del circuito o interactuar con él; por ejemplo, encender y apagar un led a distintas frecuencias para observar cambios necesita tener un periodo cercano al que procesa el ojo humano para poder distinguir entre ambos estados. Por último, otra ventaja sería utilizar distintos relojes para distintos módulos que tienen que trabajar a distintas frecuencias, entonces en vez de que cada uno cuente con su reloj, se pueden diseñar sub relojes para ellos.

1. Código y simulación de divisor de reloj

**Foto simulacion\_clock\_divider**

**Foto testbench\_clock\_divider**

1. Código y simulación del protocolo de comunicación serial

Para simular el funcionamiento de la comunicación serial con el protocolo UART, se diseñó un archivo top.v que incluye a los módulos **one\_shot\_clock\_divider**, **uart\_receptor** y **voltage\_memory**. De esta forma se podrá visualizar cómo van quedando guardados los valores recibidos y la actualización de ellos, además de qué ocurre con la señal de salida al cambiar el switch.En la simulación del testbench se envían varios datos de la forma {número, posición, tipo, 0} y en la visualización se incouyen las variables internas de la memoria de voltaje V1 y V2 en formato hexadecimal para visualizar correctamente lo que el display mostraría.

**Foto codigo\_top**

Fotos Simulacion\_UART\_1-2-3

Fotos testbench\_uart\_1-2