Pontificia Universidad Católica de Chile
Escuela de Ingeniería
Departamento de Ingeniería Eléctrica
IEE2473 – Laboratorio de electrónica analógica y digital

Experiencia 1

Grupo 6
Matías Moreno - Santiago Larraín

1. Desarrollo de la Experiencia.

1.1. Trabajo analógico

1.1.1. Acondicionamiento de Señal.

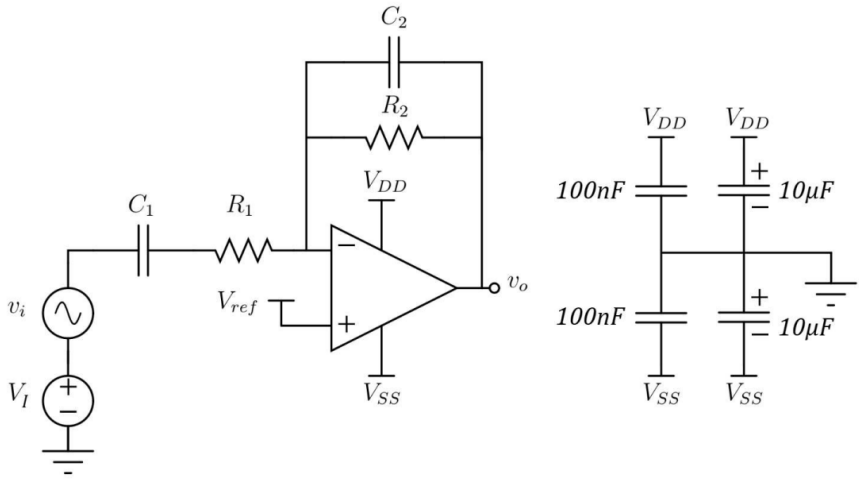


Figura 1: Descripción de imagen

1. El circuito de la figura 1 permite fijar el nivel DC a la salida y la ganancia. Encuentre una expresión para v_o en función de v_i y V_{ref}.

   Analizando el circuito con el principio de superposición, resulta una parte AC y otra DC. Al considerar solo la componente DC el capacitor $C1$ y $C2$ quedan abiertos, por lo que funciona como un buffer y resulta lo siguiente.

   $$v_{o(DC)} = V_{ref} \tag{1}$$

   Al considerar solo la parte AC, el voltaje V_{ref} y V_I quedan en cortocircuito a tierra, los resistores y capacitores se trabajan como impedancias y queda un Opamp en configuración inversora. Desarrollando resulta lo siguiente.

   $$v_o = V_{ref} - v_i \frac{j\omega R_2 C_1}{(j\omega C_2 R_2 + 1)(j\omega C_1 R_1 + 1)} \tag{2}$$

2. Utilizando el resultado encontrado en 1, encuentre una expresión para la ganancia del circuito en la banda de interés en función de R1 y R2. ¿C1 opera como circuito abierto o cortocircuito?¿C2 opera como circuito abierto o cortocircuito?.

   Antes de hacer un análisis en la banda de interés, se debe entender que el capacitor C1 sirve para el filtrado de bajas frecuencias y el C2 para el filtrado de las altas frecuencias. Por lo tanto, antes de la primera frecuencia de corte, ambos capacitores están en circuito abierto, y la ganancia es nula (solo está presente el valor V_{ref} a la salida). Al aumentar la frecuencia y entrar en la banda de interés, el capacitor C1 entra en cortocircuito, pero el capacitor C2 sigue en circuito abierto, por lo que se obtendría una ganancia asociada a la configuración inversora del opamp solo considerando las resistencias. Finalmente, cuando las frecuencias son mayores a la frecuencia de corte dada por C2 ambos capacitores entran en corto circuito produciendo una ganancia nula nuevamente.

   Con lo anterior descrito es posible decir que cuando el circuito opera en la banda de interés la ganancia es la siguiente.

$$v_o = -\frac{R_2}{R_1} \tag{3}$$

   Además, el capacitor C1 operaria en cortocircuito, y el C2 en circuito abierto.

3. Utilizando el resultado encontrado en 1 y considerando a C2 como circuito abierto, obtenga una expresión para la fase en función de R1, C1 y la frecuencia.

   Si se considera la forma compacta de la expresión en 1 resulta lo siguiente.

$$v_o = -v_i \frac{R_2 // \frac{1}{sC_2}}{R_1 + \frac{1}{sC_1}} \tag{4}$$

   Debido a que el capacitor C2 se considera como circuito abierto, la impedancia es infinita por lo que solo queda la resistencia R_2 en la rama de realimentación del opamp. Por lo tanto, resulta la siguiente expresión.

$$v_o = -v_i \frac{R_2}{R_1 + \frac{1}{sC_1}} \tag{5}$$

   Reemplazando $s = 2\pi f j$ se obtiene lo siguiente

$$v_o = -v_i \frac{j2\pi f R_2 C_1}{j2\pi f C_1 R_1 + 1} \tag{6}$$

   Para encontrar la fase, hay que considerar que se tiene un cero en el origen, un polo en una frecuencia de corte y el signo negativo le agrega -180 a la fase. Por lo tanto, la fase resulta de la siguiente forma.

$$\phi = -\frac{\pi}{2} - \arctan(2\pi f C_1 R_1) \tag{7}$$

4. Utilizando el resultado encontrado en 1 y considerando a C1 como cortocircuito, obtenga una expresión para la fase en función de R2, C2 y la frecuencia.

   Si se considera la forma compacta de la expresión en 1 resulta lo siguiente.

$$v_o = -v_i \frac{R_2 // \frac{1}{sC_2}}{R_1 + \frac{1}{sC_1}} \tag{8}$$

   Debido a que el capacitor C1 se considera como cortocircuito, quedaría el opamp en configuración inversora considerando la impedancia de la rama que no realimenta como solamente la resistencia R1. Con lo anterior, la expresión sería la siguiente.

$$v_o = -v_i \frac{R_2 // \frac{1}{sC_2}}{R_1} \tag{9}$$

   Reemplazando $s = 2\pi f j$ resulta lo siguiente.

$$\frac{v_o}{v_i} = -\frac{R_2}{R_1} \frac{1}{j2\pi f C_2 R_2 + 1} \tag{10}$$

   Para encontrar la fase, hay que considerar que se tiene un polo en una frecuencia de corte y el signo negativo le agrega -180 a la fase. Por lo tanto, la fase resulta de la siguiente forma.

$$\phi = -\pi - \arctan(2\pi f C_2 R_2) \tag{11}$$

5. Investigue la diferencia entre capacitores electrolíticos y cerámicos. ¿Cuáles son más útiles para C1 y C2?.

   Según electropreguntas [1] los capacitores electrolíticos sirven para aplicaciones de baja frecuencia debido a que tienen una mayor capacitancia, comienzan desde los μF y pueden llegar hasta los mF. En cambio, los capacitores cerámicos sirven para aplicaciones de alta frecuencia, ya que poseen capacitancias mucho más bajas, del orden de los μF hasta los pF. Debido a lo anterior mencionado, sería conveniente que el capacitor C1 sea electrolítico para lograr filtrar bajas frecuencias, y que el capacitor C2 sea cerámico para lograr filtrar altas frecuencias.

6. Utilizando las expresiones encontradas en 2, 3 y 4, calcule el valor de R1, R2, C1 y C2 para que la señal acondicionada cumpla con las especificaciones.

   Primero se debe encontrar cual es la ganancia en pasabanda del circuito, para ello se debe considerar que la señal de entrada y salida del circuito deben ser la siguiente manera.

$$V_{in}(t) = 3.5 \sin(wt) + 6.5 \tag{12}$$

$$V_{out}(t) = 1.65 \sin(wt) + 1.65 \tag{13}$$

Debido a las expresiones anteriores es posible decir que el valor de V_{ref} es 1.65 y el valor de V_I es 6.5. Según el valor de v_o encontrado en la pregunta 1, es posible decir que el valor de V_I no afecta a la salida. Por lo tanto, la ganancia en pasabanda viene dada por la siguiente expresión.

$$\frac{v_{out}}{v_{in}} = \frac{1.65}{3.5} = 0.471 \tag{14}$$

Para determinar los valores de las resistencias, arbitrariamente se elige el valor de $10k[\Omega]$ para R1. Según esto, y el valor de la ganancia obtenido en la pregunta 2, la resistencia R2 debe valer $4.7k[\Omega]$.

Para determinar los valores de los capacitores se debe considerar la especificación de fase. La condición que debe cumplirse es que en la frecuencia $1[Hz]$ la fase debe ser $\phi = -165°$. Además, en la frecuencia de $400[Hz]$ la fase debe ser $\phi = -195°$. Para cumplir las siguientes relaciones, se deben despejar las siguientes ecuaciones.

$$\frac{\pi}{2} - \frac{11\pi}{12} = -\arctan(2\pi C_1 \cdot 10k) \tag{15}$$

$$\pi - \frac{13\pi}{12} = -\arctan(2\pi 400 C_2 \cdot 4.7k) \tag{16}$$

Los valores resultantes son los siguientes.

$$C_1 = 59[\mu F] \tag{17}$$

$$C_2 = 22[nF] \tag{18}$$

$$R_1 = 10[k\Omega] \tag{19}$$

$$R_2 = 4.7[k\Omega] \tag{20}$$

7. Con los valores obtenidos, calcule las frecuencias de corte del circuito acondicionador.

   Las frecuencias de corte son las siguientes.

$$f_{c1} = \frac{1}{2\pi R_1 C_1} = 0.27[Hz] \tag{21}$$

$$f_{c2} = \frac{1}{2\pi R_2 C_2} = 1539[Hz] \tag{22}$$

8. Haga una simulación AC con los valores obtenidos en 6 y compare con los valores calculados teóricamente.

   Para el caso de la frecuencia de corte 1 es exactamente 0.27[Hz] al igual que el valor obtenido anteriormente.

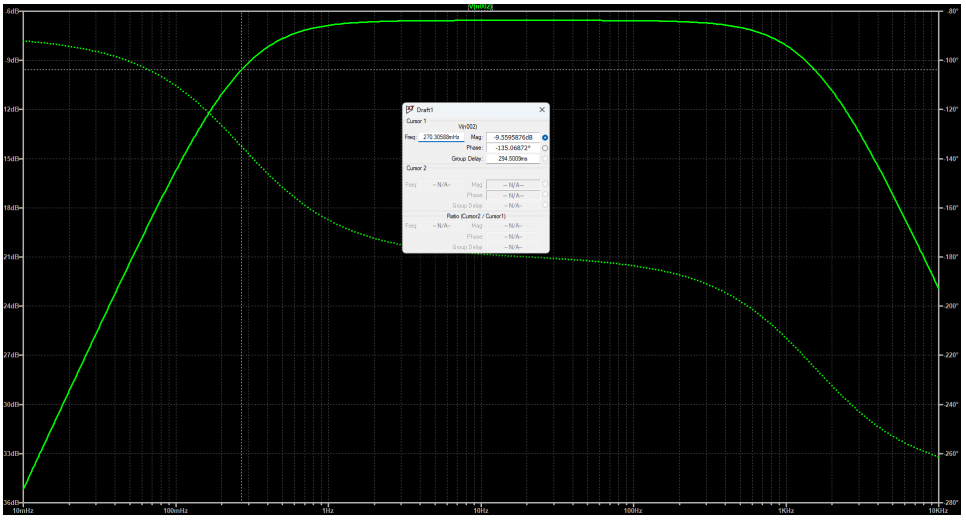


Figura 2: Gráfico Bode f_{c1}

Para el caso de la frecuencia de corte 2 es 1.533[kHz] muy similar al obtenido anteriormente.

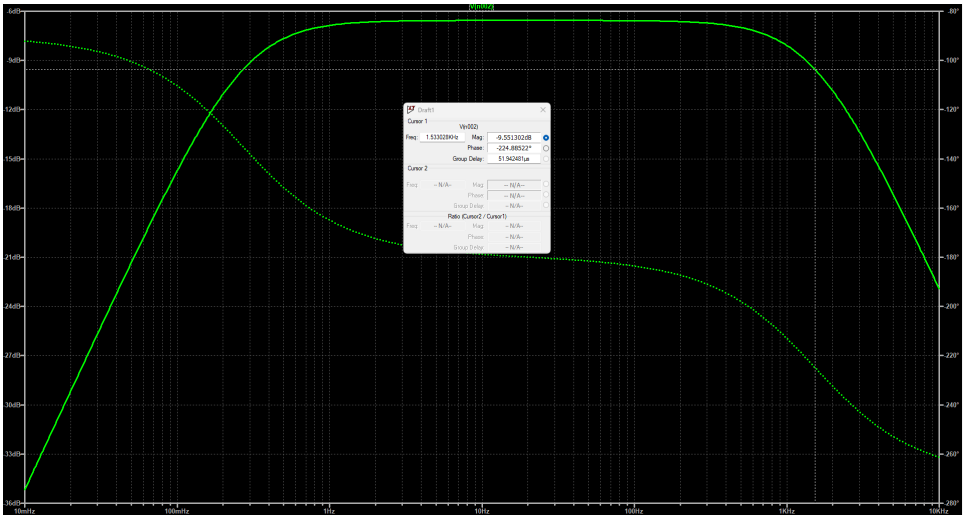


Figura 3: Gráfico Bode f_{c2}

9. Haga una simulación transiente, graficando la señal de entrada V_i y de salida V_o para:

   En todos los gráficos la entrada está graficada de color azul y la salida de color verde.

   a) Entrada nula

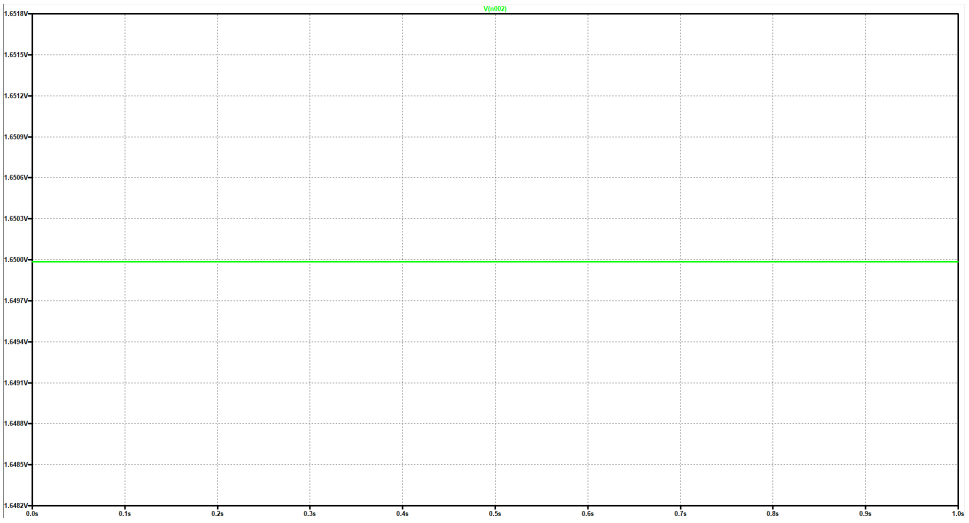


Figura 4: V_o con entrada nula

b) Entrada fija de 6.5V

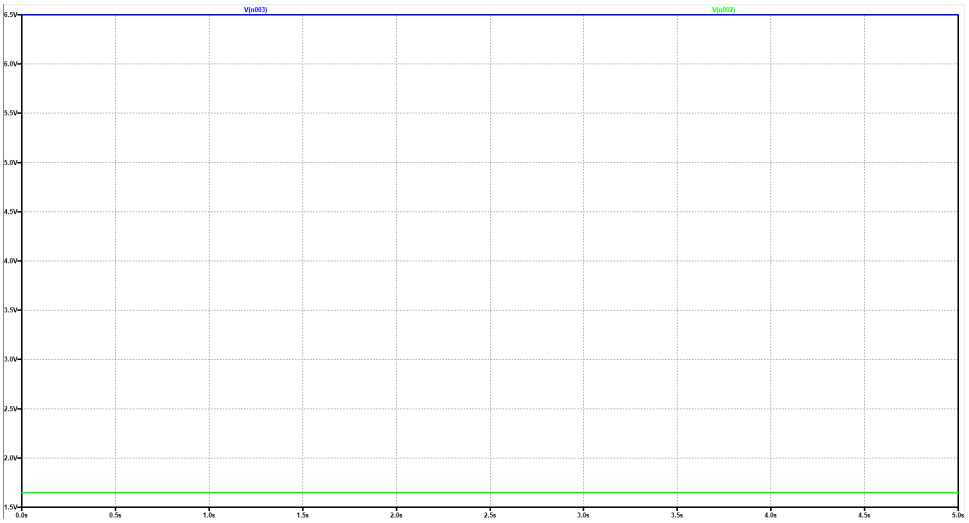


Figura 5: V_o con entrada fija de 6.5[V]

c) Entrada sinusoidal de 1 Hz, 7Vpp y componente continua de 6.5[V].

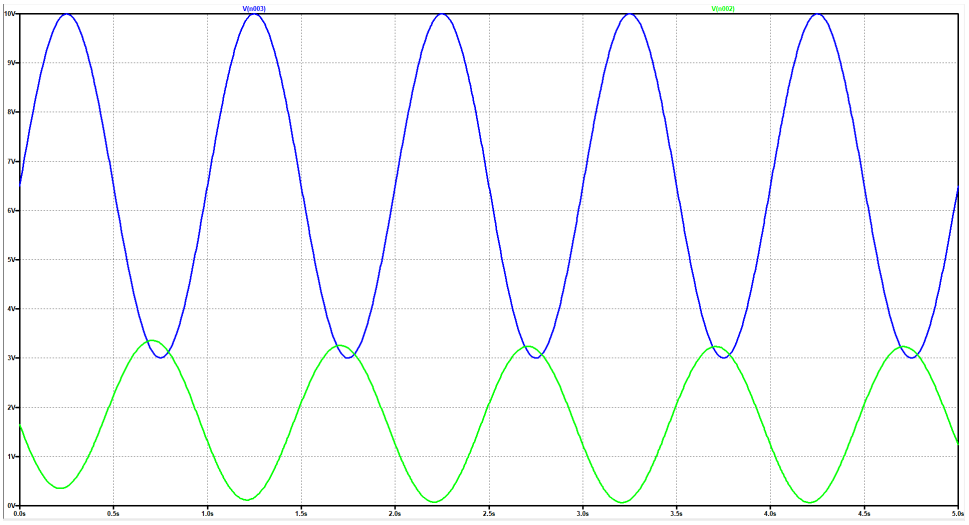


Figura 6: V_o con entrada de 1Hz, 7Vpp, y fija de 6.5[V]

d) Entrada sinusoidal de 1 Hz, 1Vpp y componente continua de 6.5[V].

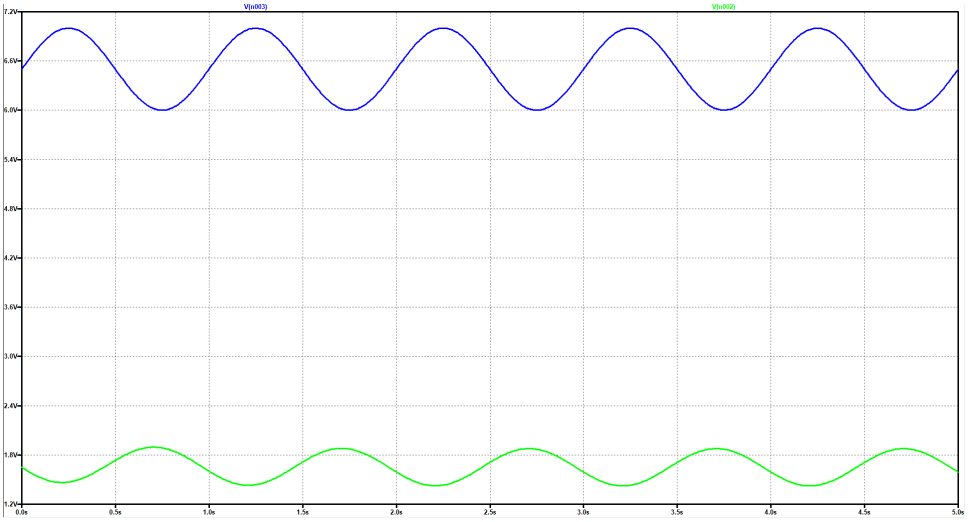


Figura 7: V_o con entrada de 1Hz, 1Vpp, y fija de 6.5[V]

e) Entrada sinusoidal de 20 Hz, 7Vpp y componente continua de 6.5[V].

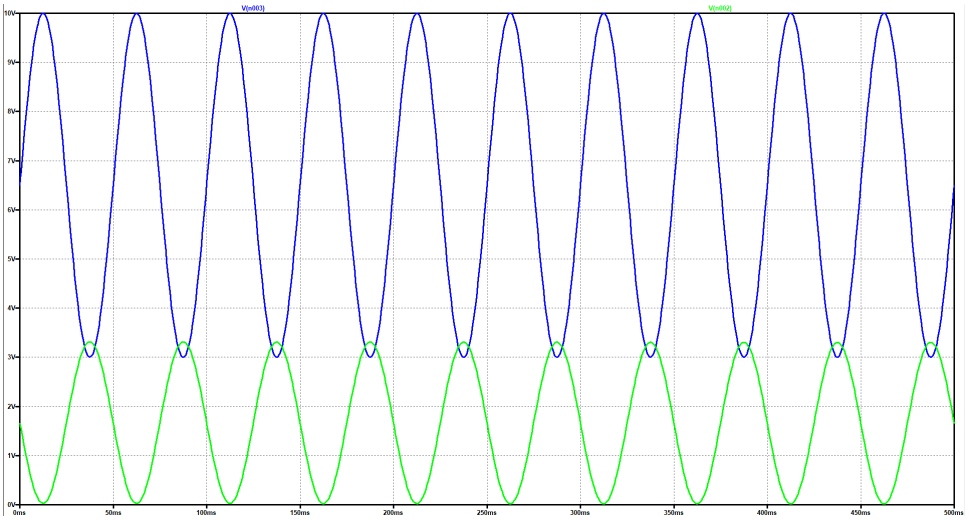


Figura 8: V_o con entrada de 20Hz, 7Vpp, y fija de 6.5[V]

f) Entrada sinusoidal de 20 Hz, 1Vpp y componente continua de 6.5[V].

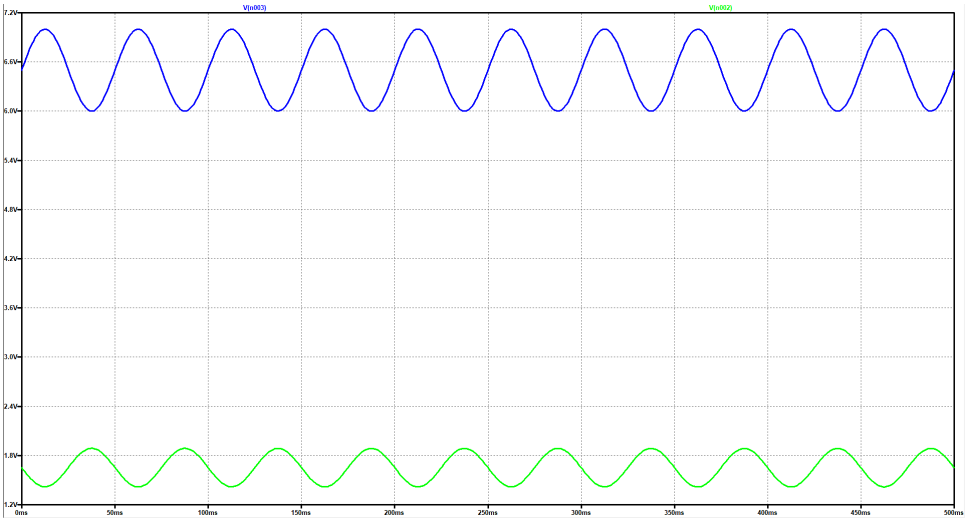


Figura 9: V_o con entrada de 20Hz, 1Vpp, y fija de 6.5[V]

g) Entrada sinusoidal de 400Hz, 7Vpp y componente continua de 6.5[V].

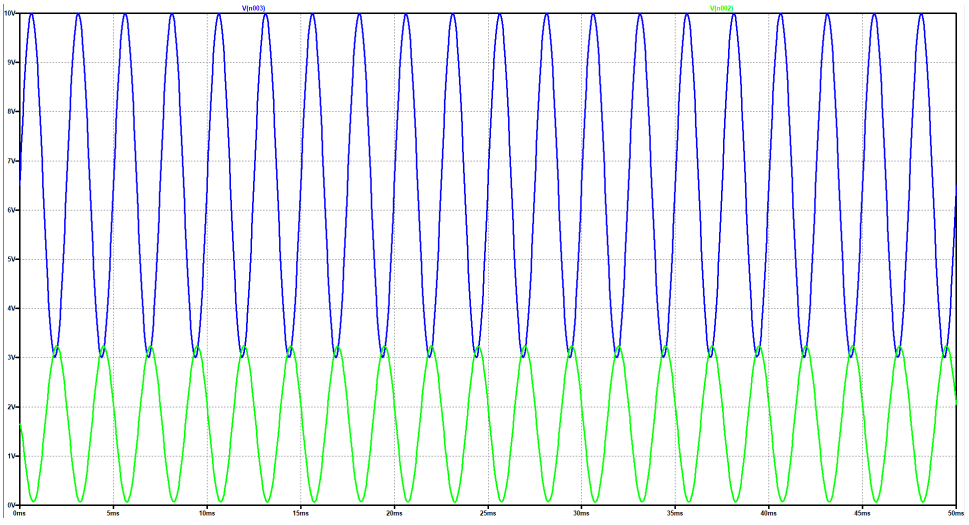


Figura 10: V_o con entrada de 400Hz, 7Vpp, y fija de 6.5[V]

h) Entrada sinusoidal de 400Hz, 1Vpp y componente continua de 6.5[V].

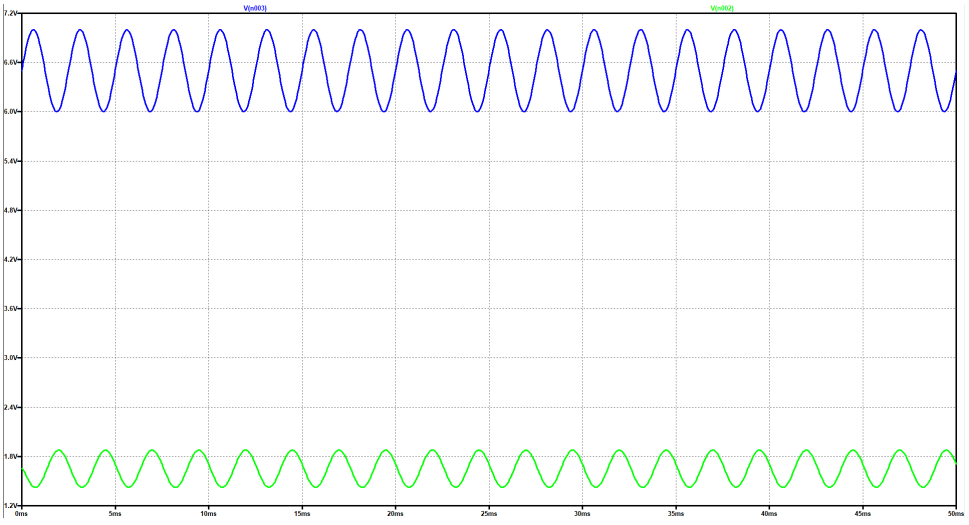


Figura 11: V_o con entrada de 400Hz, 1Vpp, y fija de 6.5[V]

i) Mida el desfase relativo entre la señal de entrada y la señal de salida de los puntos (c), (e) y (g).

A continuación se muestran 2 tipos de desfases por cada ítem. El primero es el desfase real de la señal de entrada con la de salida. El segundo es el desfase de la señal de entrada con la de salida desfasada en 180, esto se hace con el objetivo de observar si se cumplen los 15° máximos de desfase admisibles según las especificaciones.

$$\phi_{(c)real} = 167.2^\circ$$
$$\phi_{(c)virtual} = 12.8^\circ$$
$$\phi_{(e)real} = 179.25^\circ$$
$$\phi_{(e)virtual} = 0.75^\circ$$
$$\phi_{(g)real} = 194.5^\circ$$
$$\phi_{(g)virtual} = 14.5^\circ$$

1.2. Trabajo digital

1.2.1. MSP

1. Investigue sobre las protecciones (si aplica) y los rangos de voltaje admisibles del ADC integrado en el microcontrolador.

   La protección que trae la MSP es de tipo ESD, esta sirve para proteger a la placa de la estática. Esto lo realiza mediante diodos de protección, los cuales aguantan como máximo una descarga de 1000[V] si viene de un humano, o 250[V] si viene de un aparato electrónico [2].

   Para el caso de los rangos de voltaje, se tiene que se pueden tomar 2 referencias internas (1.5, o 2.5), se puede conectar una referencia externa V_{cc}, o se pueden ajustar 2 valores arbitrarios definidos (VREF+ y VREF-) para ser los rangos de valores en los pines A4 y A3.

2. Investigue cuáles son las ventajas y desventajas de los protocolos de comunicación serial y paralelo. ¿Qué pines pueden ser utilizados para los protocolos serial?

   El protocolo de comunicación serial comprende un solo hilo de dato, en el cual se transmiten las señales de a un bit a la vez. Las comunicaciones serie tienen una línea para recibir información (RX) y otra para enviarla (TX). Dentro de sus desventajas está tener una comunicación más lenta, pues para transmitir un bus de X datos, necesitará enviar secuencialmente X señales más señales de control, como bit de inicio, parada y paridad. Por otro lado, sus ventajas consideran una mayor capacidad de traslado a distancias largas y no tiene problema con la sincronización de datos.

   El protocolo de comunicación paralela, a diferencia del serial, comprende múltiples hilos de información. Si se quieren enviar datos de ocho bits simultáneamente, serán necesarios ocho hilos de comunicación. El mayor problema que podría ocurrir es que los datos no se reciban simultáneamente, resultando en errores. Además de esta desventaja, este protocolo de comunicación no es preferible para distancias largas. Su ventaja con respecto al protocolo de comunicación serial vendría a ser una comunicación más rápida, pues envía múltiples

datos de forma simultánea. [3]

En la tarjeta MSP430 se tienen los pines P1.1 y P1.2, los cuales pueden ser utilizados para la comunicación serial por medio del protocolo UART; el pin P1.1 sería el canal TX y el pin P1.2 el canal RX.

1.2.2. FPGA

1. Realice un diagrama de bloques, explicitando las entradas y salidas, así como el tamaño de los buses de datos utilizados. Explique que realiza cada bloque.

   Para procesar la información del voltaje a mostrar en el display de la Basys3, la FPGA debe procesar un bus serial de entrada y entregar 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 12.

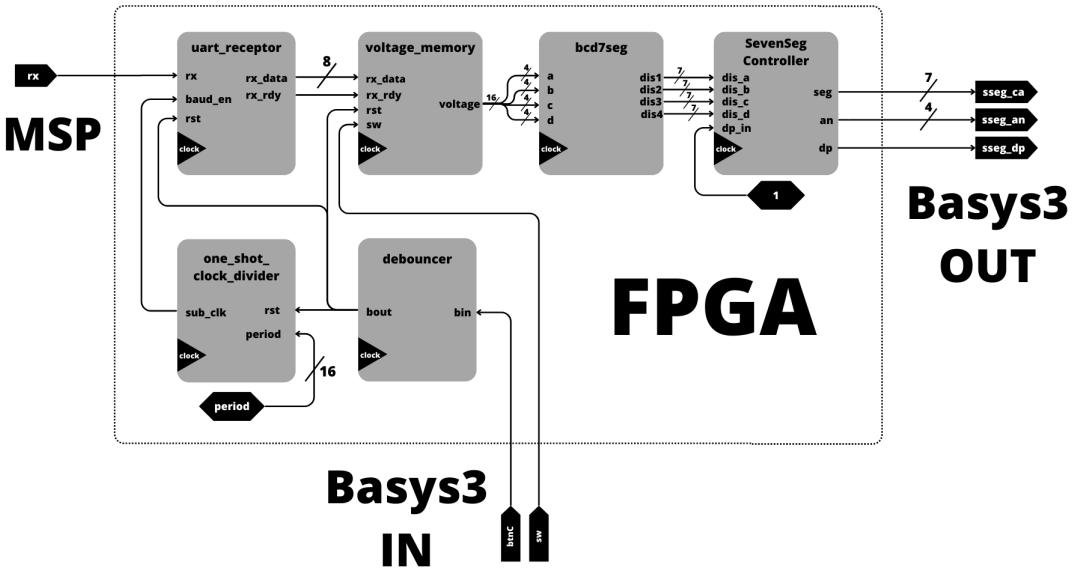


Figura 12: Diagrama de bloques FPGA

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

a) **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 evalúe 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 13 y el código tenía errores, por lo que se tuvo que alterar en una gran porción.

El código final corresponde al de las Figuras 16 y 17. Es importante resaltar que a modo de recibir bien los bits 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, por lo que se guarda el valor de la posición número 7).

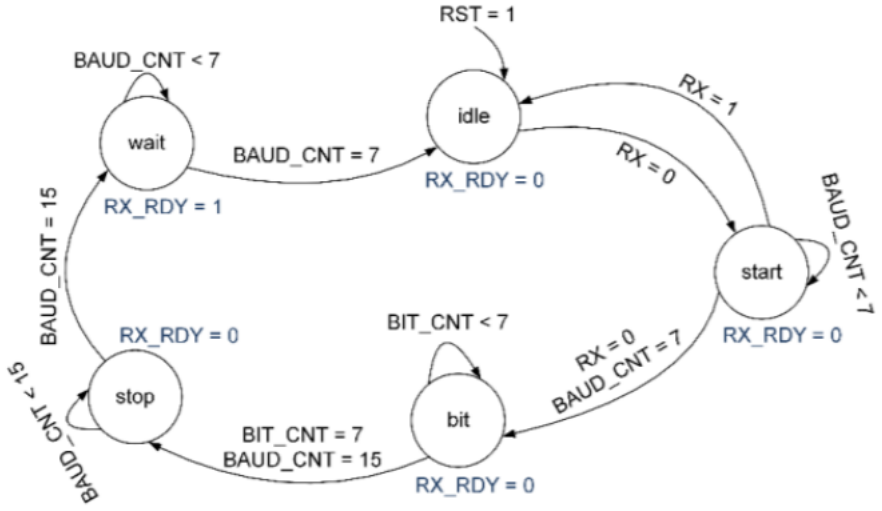


Figura 13: FSM receptor

b) **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. También 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 cuatro dígitos del voltaje (cada uno de 4 bits). En caso de resetear el código, $V1$ y $V2$ vuelven a 0. En la Figura 14 se muestra el código implementado para este módulo.

c) **Bcd7seg**
Corresponde a un módulo extraído de un proyecto realizado el segundo semestre del año 2021 en el curso IEE2713 Sistemas Digitales por Santiago Larraín. Este módulo permite convertir un valor numérico entero unitario (entre 0 y 9) en un código de siete bits para el display de siete segmentos de la Basys3.

d) **SevenSegController**
También fue extraído de la fuente mencionada anteriormente y la función de este módulo es recibir los valores de los displays (cuatro dígitos 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.

e) **One shot clock divider**
Este módulo es una variación del módulo clock divider 15, el cual con una señal de reloj y un periodo determinado, emite una señal de reloj *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, en cada periodo tiempo, sirviendo así como señal de actualización del módulo uart receptor.

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

2. Investigue las 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 descoordinada. Para resolver esto, se puede definir una frecuencia de comunicación en *bits por segundo [bps]* 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 considerablemente mayor al 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, si se quiere encender y apagar un led a distintas frecuencias para observar cambios, se necesita tener un periodo cercano al que procesa el ojo humano para poder distinguir entre ambos estados.

Una última ventaja de ejemplo 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.

3. Escriba un código y simule un divisor de reloj. El código del divisor de reloj se muestra en la Figura 15, el *testbench* se encuentra en la figura 18, y la simulación en la figura 19

4. Escriba un código y simule algún protocolo de comunicación serial.

   Para simular el funcionamiento de la comunicación serial con el protocolo UART, se diseñó un archivo top.v (figura 20) 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 {0, tipo (1 bit), posición (2 bits), número (4 bits)} y en la visualización se incluyen las variables internas de la memoria de voltaje $V1$ y $V2$ en formato hexadecimal para visualizar correctamente lo que el display mostraría.

   El testbench del código UART se encuentra en las figuras 21 y 22. Las simulaciones se encuentran en las figuras 23, 24 y 24.

Bibliografía

[1] *¿Cuáles son las diferencias entre los capacitores cerámicos, electrolíticos y de película delgada?* https://electropreguntas.com/cuales-son-las-diferencias-entre-los-capacitores-ceramicos-electroliticos-y-de-pelicula-delgada/.

[2] *ESD Diode Current Specification.* https://www.ti.com/lit/an/slaa689b/slaa689b.pdf?ts=1692361347647&.

[3] *Velocidad de escritura serie y paralelo.* https://massive.io/es/transferencia-de-archivos/velocidad-de-escritura-en-serie-vs-paralelo/.

2. Anexos

2.1. Códigos

```
22
23      module voltage_memory(
24          input clk,
25          input sw,
26          input rst,
27          input rx_rdy,
28          input [7:0] rx_data,
29          output [15:0] voltage
30          );
31
32          reg [15:0] V1;
33          reg [15:0] V2;
34
35          // Save Values
36          always @(posedge clk) begin
37              if (rst) begin
38                  V1 = 0;
39                  V2 = 0;
40              end
41              else if (rx_rdy)
42                  case(rx_data[7:4])
43                      4'b0000 : V1[3:0] = rx_data;
44                      4'b0001 : V1[7:4] = rx_data;
45                      4'b0010 : V1[11:8] = rx_data;
46                      4'b0011 : V1[15:12] = rx_data;
47                      4'b0100 : V2[3:0] = rx_data;
48                      4'b0101 : V2[7:4] = rx_data;
49                      4'b0110 : V2[11:8] = rx_data;
50                      4'b0111 : V2[15:12] = rx_data;
51                      default : begin
52                                V1 = 16'b1001100001110101;
53                                V2 = 16'b1001100001110101;
54                                end
55                  endcase
56          end
57
58          assign voltage = sw ? V2 : V1;
59      endmodule
60
```

Figura 14: Código voltage memory

```
22
23  module clock_divider(
24      input clk, rst,
25      input [15:0] period,
26      output reg sub_clk = 0
27      );
28
29      reg [16:0] counter = 0;
30
31      always @(posedge clk) begin
32          if (rst) begin
33              counter <= 17'd0;
34              sub_clk = 1'b0;
35          end
36          else if (counter < period/2-1) begin
37              counter <= counter + 1;
38          end
39          else begin
40              sub_clk = ~sub_clk;
41              counter <= 0;
42          end
43      end
44  endmodule
45
46  module one_shot_clock_divider(
47      input clk, rst,
48      input [15:0] period,
49      output reg sub_clk = 0
50      );
51
52      reg [16:0] counter = 0;
53
54      always @(posedge clk) begin
55          if (rst) begin
56              counter <= 0;
57              sub_clk = 0;
58          end
59          else if (counter < period-1) begin
60              counter <= counter + 1;
61              sub_clk = 0;
62          end
63          else begin
64              sub_clk = 1;
65              counter <= 0;
66          end
67      end
68  endmodule
69
```

Figura 15: Código clock divider

```
22
23  module uart_receptor(
24      input clk,
25      input rst,
26      input baud_en,
27      input rx,
28      output reg rx_rdy,
29      output reg [7:0] rx_data
30      );
31
32      // ESTADOS
33      parameter IDLE =  3'b000;
34      parameter START = 3'b001;
35      parameter BIT =   3'b010;
36      parameter STOP =  3'b011;
37      parameter WAIT =  3'b100;
38
39      reg [2:0] state, nextstate;
40      reg [2:0] bit_cnt;
41      reg [3:0] baud_cnt;
42
43      // Actualización del estado
44      // Ojo: Esta parte es secuencial
45      // BIT_CNT es un contador, por lo que también debe cambiar
46      // de manera secuencial
47      always @(posedge clk) begin
48          if (rst) begin
49              nextstate <= IDLE;
50              baud_cnt <= 0;
51              bit_cnt <= 0;
52              rx_rdy = 0;
53              rx_data = 8'd0;
54          end
55          else begin
56              state <= nextstate;
57              if (baud_en) begin
58                  baud_cnt <= baud_cnt + 1;
59                  if (baud_cnt == 15 && state == BIT) begin
60                      rx_data[bit_cnt] = rx;
61                      bit_cnt <= bit_cnt + 1;
62                  end
63              end
64          end
65      end
66
67      // Logica para siguiente estado
68      // Ojo: Esta parte es combinacional
69
70      always @(posedge clk) begin
71          case(state)
72          IDLE: begin
73              if (baud_en && rx == 0) begin
74                  nextstate = START;
75                  baud_cnt <= 0;
76                  bit_cnt <= 0;
77              end
78              else nextstate = IDLE;
79          end
80
81          START: begin
82              if (baud_cnt < 7) nextstate = START;
83              if (baud_cnt == 7 && rx == 0) begin
84                  nextstate = BIT;
85                  baud_cnt <= 0;
86              end
87              if (rx == 1) nextstate = IDLE;
88          end
89
```

Figura 16: Código UART parte 1

```
89
90          BIT: begin
91              if (bit_cnt <= 7) nextstate = BIT;
92              if (baud_cnt == 15 && bit_cnt == 7) begin
93                  nextstate = STOP;
94                  baud_cnt <= 0;
95              end
96          end
97
98          STOP: begin
99              if (baud_cnt < 15) nextstate = STOP;
100             else begin
101                 nextstate = WAIT;
102                 baud_cnt <= 0;
103                 bit_cnt <= 0;
104             end
105         end
106
107         WAIT: begin
108             if (baud_cnt < 7) nextstate = WAIT;
109             else begin
110                 nextstate = IDLE;
111                 baud_cnt <= 0;
112             end
113         end
114         endcase
115     end
116
117     always @(posedge clk) begin
118         case(state)
119             IDLE: begin
120                 rx_rdy = 0;
121             end
122             START: begin
123                 rx_rdy = 0;
124             end
125             BIT: begin
126                 rx_rdy = 0;
127             end
128             STOP: begin
129                 rx_rdy = 0;
130             end
131             WAIT: begin
132                 rx_rdy = 1;
133             end
134          endcase
135     end
136
137
138 endmodule
139
```

Figura 17: Código UART parte 2

```
22
23      module clock_divider_tb;
24          reg clk;
25          reg rst;
26          reg [15:0] period = 11'd20; // T/2
27          wire sub_clk;
28
29          clock_divider uut(clk, rst, period, sub_clk);
30
31          initial begin
32              clk = 0;
33              forever clk = #(1) ~clk;
34          end
35
36          initial begin
37              rst = 1;
38              #10 rst = 0;
39              #1000;
40          end
41      endmodule
42
```

Figura 18: Testbench clock divider

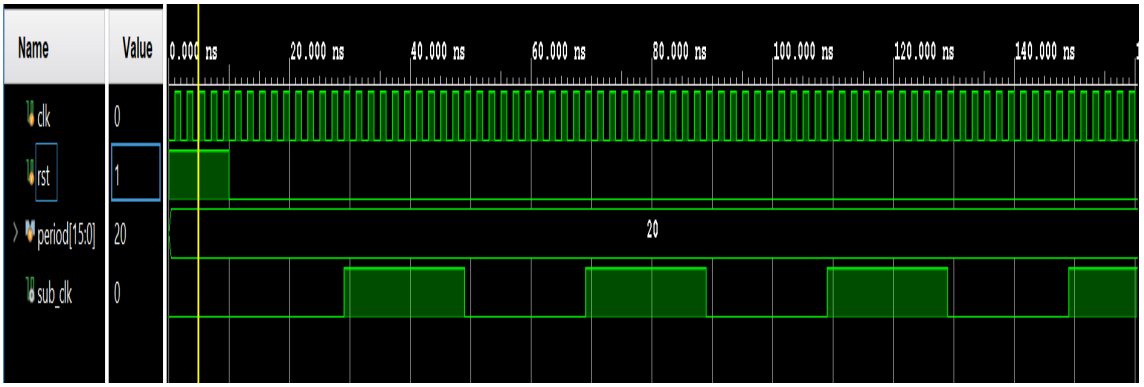


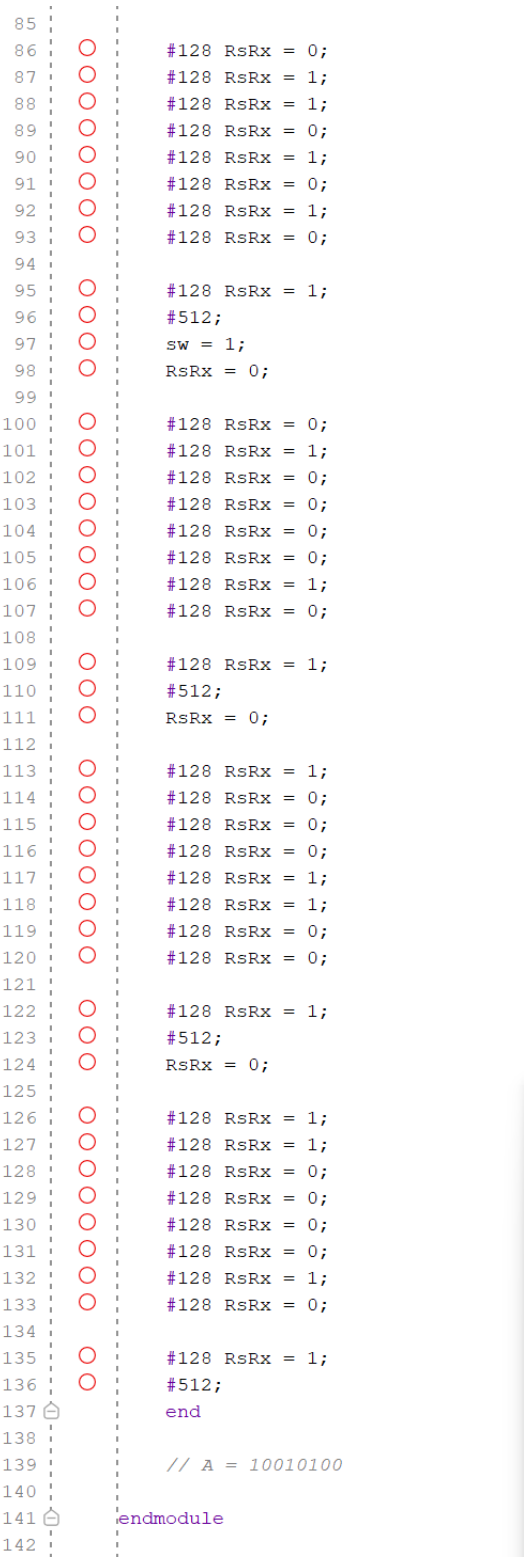
Figura 19: Simulación clock divider

```
22
23  module top(
24      input clk,
25      input sw,
26      input rst,   //btnC
27      input RsRx,
28      output [15:0] voltage
29      //output [6:0] sseg_ca,
30      //output [3:0] sseg_an,
31      //output sseg_dp
32      );
33
34      parameter [15:0] period = 16'd4; // f = 100 MHz clock -> period = f/baud_rate * 1/16
35
36      //wire rst_db;
37      wire baud_en;
38      wire rx_rdy;
39      wire [7:0] rx_data;
40      //wire [15:0] voltage;
41      //wire [6:0] dis1, dis2, dis3, dis4;
42
43      //debouncer rst_debouncer(clk, btnC, rst_db);
44
45      // Receptor
46      one_shot_clock_divider os_cd(clk, rst, period, baud_en);
47      uart_receptor uart_rx(clk, rst, baud_en, RsRx, rx_rdy, rx_data);
48      voltage_memory v_mem(clk, sw, rst, rx_rdy, rx_data, voltage);
49
50      // Display
51      //bcd7seg bcd_7seg(voltage[15:12], voltage[11:8], voltage[7:4], voltage[3:0], dis1, dis2, dis3, dis4);
52      //SevenSegController(clk, 1'b1, dis1, dis2, dis3, dis4, sseg_ca, sseg_an, sseg_dp);
53
54  endmodule
55
```

Figura 20: Código TOP.v

```
22
23  module uart_tb;
24      reg clk;
25      reg sw;
26      reg rst;
27      reg RsRx;
28      wire [15:0] voltage;
29      //wire rx_rdy;
30      //wire [7:0] rx_data;
31
32      top uut(clk, sw, rst, RsRx, voltage);
33
34      initial begin
35          clk = 0;
36          forever clk = #(1) ~clk;
37      end
38
39      initial begin
40      rst = 1;
41      RsRx = 1;
42      sw = 0;
43      #20 rst = 0;
44      #20;
45      RsRx = 0;
46
47      #128 RsRx = 1;
48      #128 RsRx = 1;
49      #128 RsRx = 1;
50      #128 RsRx = 0;
51      #128 RsRx = 1;
52      #128 RsRx = 1;
53      #128 RsRx = 0;
54      #128 RsRx = 0;
55
56      #128 RsRx = 1;
57      #512;
58      RsRx = 0;
59
60      #128 RsRx = 1;
61      #128 RsRx = 0;
62      #128 RsRx = 0;
63      #128 RsRx = 0;
64      #128 RsRx = 0;
65      #128 RsRx = 1;
66      #128 RsRx = 0;
67      #128 RsRx = 0;
68
69      #128 RsRx = 1;
70      #512;
71      RsRx = 0;
72
73      #128 RsRx = 0;
74      #128 RsRx = 1;
75      #128 RsRx = 0;
76      #128 RsRx = 0;
77      #128 RsRx = 1;
78      #128 RsRx = 0;
79      #128 RsRx = 0;
80      #128 RsRx = 0;
81
82      #128 RsRx = 1;
83      #512;
84      RsRx = 0;
85
```

Figura 21: Testbench UART parte 1



```
85
86      #128 RsRx = 0;
87      #128 RsRx = 1;
88      #128 RsRx = 1;
89      #128 RsRx = 0;
90      #128 RsRx = 1;
91      #128 RsRx = 0;
92      #128 RsRx = 1;
93      #128 RsRx = 0;
94
95      #128 RsRx = 1;
96      #512;
97      sw = 1;
98      RsRx = 0;
99
100     #128 RsRx = 0;
101     #128 RsRx = 1;
102     #128 RsRx = 0;
103     #128 RsRx = 0;
104     #128 RsRx = 0;
105     #128 RsRx = 0;
106     #128 RsRx = 1;
107     #128 RsRx = 0;
108
109     #128 RsRx = 1;
110     #512;
111     RsRx = 0;
112
113     #128 RsRx = 1;
114     #128 RsRx = 0;
115     #128 RsRx = 0;
116     #128 RsRx = 0;
117     #128 RsRx = 1;
118     #128 RsRx = 1;
119     #128 RsRx = 0;
120     #128 RsRx = 0;
121
122     #128 RsRx = 1;
123     #512;
124     RsRx = 0;
125
126     #128 RsRx = 1;
127     #128 RsRx = 1;
128     #128 RsRx = 0;
129     #128 RsRx = 0;
130     #128 RsRx = 0;
131     #128 RsRx = 0;
132     #128 RsRx = 1;
133     #128 RsRx = 0;
134
135     #128 RsRx = 1;
136     #512;
137     end
138
139     // A = 10010100
140
141 endmodule
142
```

Figura 22: Testbench UART parte 2

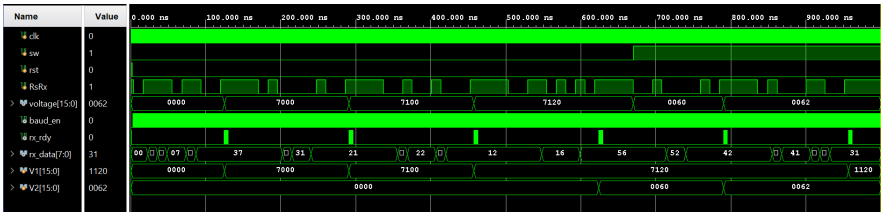


Figura 23: Simulación UART parte 1

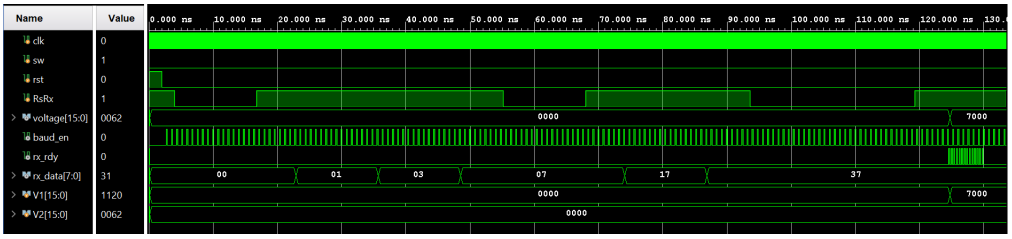


Figura 24: Simulación UART parte 2

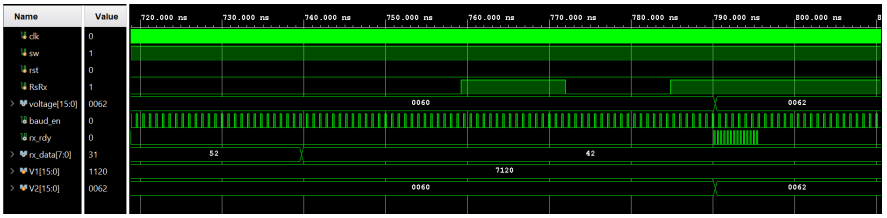


Figura 25: Simulación UART parte 3