Contador Gray

Gabriel Fernando Araya Mora B80525

Escuela de Ingeniería Eléctrica, Universidad de Costa Rica

Circuitos Digitales II (IE-0523)

---

1. Tiempo de ejecución de la tarea:

- **Buscar información**: Siempre antes de empezar cualquier trabajo es importante tomarse un tiempo prudente para buscar información e investigar acerca del tema. Esto es especialmente cierto cuando se trata de programar. Para la búsqueda de información, se revisó la documentación, y la teoría detrás del contador Gray. La búsqueda de información me llevó una hora.
- **Usando la información**: Encontrados varios ejemplos en la web de cómo implementar un contador gray, se comparó para saber cual se ajustaba más al dado por el profesor en el enunciado de la tarea. Hecho esto entonces se modifica el código conductual encontrado para que cumpla con todos los requisitos para la tarea. Se agrega un checker automático el cual se va a encargar de comparar las señales de salida flanco a flanco y va a decir dónde no son iguales, encontrando el error de una forma automática. La elaboración del código para esta tarea me tomó al rededor de 2 horas.
- **Elaboración del reporte**: La elaboración del reporte me tomó aproximadamente una hora.
- **Total:** De lo anterior se saca que el tiempo total para la realización de la tarea es de 4h.

2. Contador Gray

El código Gray es un tipo especial de código binario que no es ponderado (los dígitos que componen el código no tienen un peso asignado). Su característica es que entre una combinación de dígitos y la siguiente, sea esta anterior o posterior, solo hay una diferencia de un dígito.

2.1. Aplicaciones:

- **Minimización de circuitos Booleanos**: El código Gray se utiliza para la minimización de circuitos lógicos en el uso de métodos como Mapa de Karnaugh.
- **Corrección de errores**: En comunicación digital el código Gray tiene un aporte importante a la hora de enviar y recibir datos de forma correcta.

3. Descripción arquitectónica o diagrama del circuito.

En la siguiente tarea no se pidió hacer la síntesis del módulo conductual, ya que el objetivo era encontrar el error en el archivo estructural dado por el profesor al compararlo con una descripción completamente conductual que se sabe que está bien hecha. Además el checker se encarga de hacer el trabajo sucio ya que cuando se pone en cero indica una discrepancia entre ambas descripciones.

El siguiente código es el contador Gray implementado de forma conductual, cabe decir que este contador es de cinco bits nada más.

```
module contador_gray_cond(
            input clk,
            input reset_L,
            input enable,
            output reg [4:0] salida_gray);
        reg [4:0] contador_bin;
        always @(posedge clk)begin
        if (reset_L == 0 || enable == 0)begin
            contador_bin <= 0;
            salida_gray <= 0;
        end
        else if (reset_L == 1 && enable == 1)begin
            contador_bin = contador_bin + 1;
            salida_gray [4] <= contador_bin[4];
            salida_gray [3] <= contador_bin[4] + contador_bin[3];
            salida_gray [2] <= contador_bin[3] + contador_bin[2];
            salida_gray [1] <= contador_bin[2] + contador_bin[1];
            salida_gray [0] <= contador_bin[1] + contador_bin[0];
            end
        end
endmodule
```

4. Plan de pruebas

Para este plan de pruebas primero se realizó un código a nivel de conductual sin importar las limitaciones de este ya que no fue sintetizado. Este código conductual sirvió para comprobar el correcto comportamiento del contador Gray dado en el enunciado.

Seguidamente se creó un testbench en el cual se agregaron las señales de entrada y se simuló por bastante tiempo el funcionamiento de los módulos conductual y estructural con el fin de encontrar algún error en el diseño estructural proporcionado por el profesor.

El plan de pruebas se resume en dejar al checker hacer su trabajo y encontrar donde ambas salidas fueran diferentes. Esto se espera que se vea como un cero en la salida del checker indicando que hay una discrepancia.

5. Istrucciones para ejecutar el programa.

Para ejecutar el programa, estando en el directorio **src** y abriendo una terminal en esta dirección basta con escribir ***make final***.

```
final:
        iverilog BancoPrueba.v
        ./a.out
        rm a.out
        gtkwave prueba_1.vcd
```

6. Ejemplos de los Resultados.

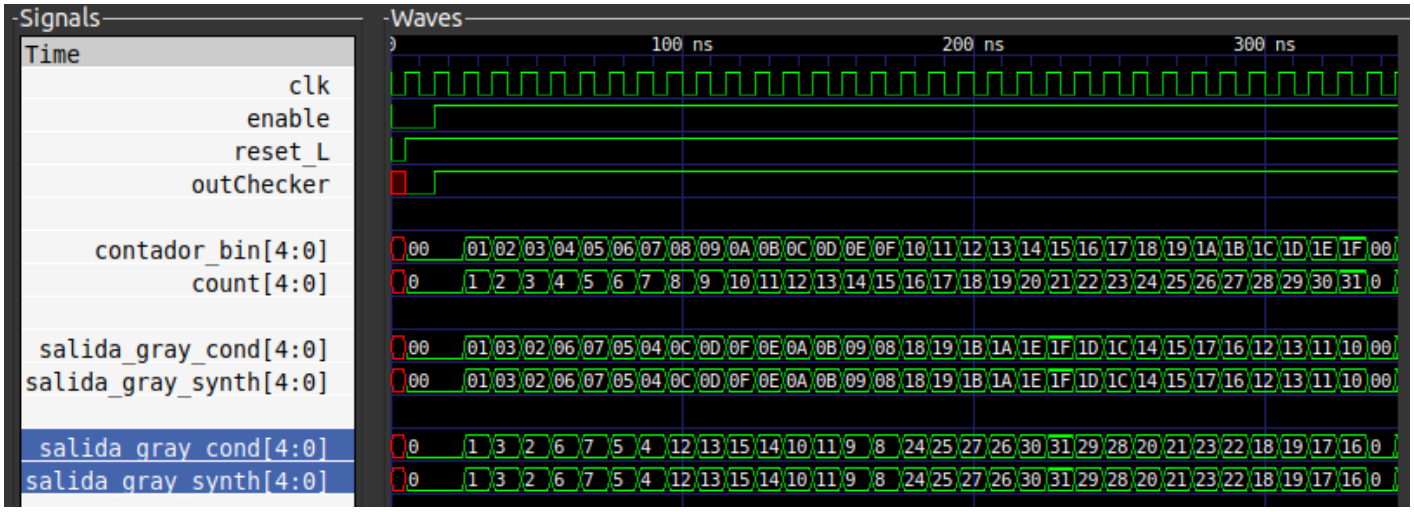

Figura 1: Salidas el contador Gray conductual y estructural con buen funcionamiento. (Creación Propia)

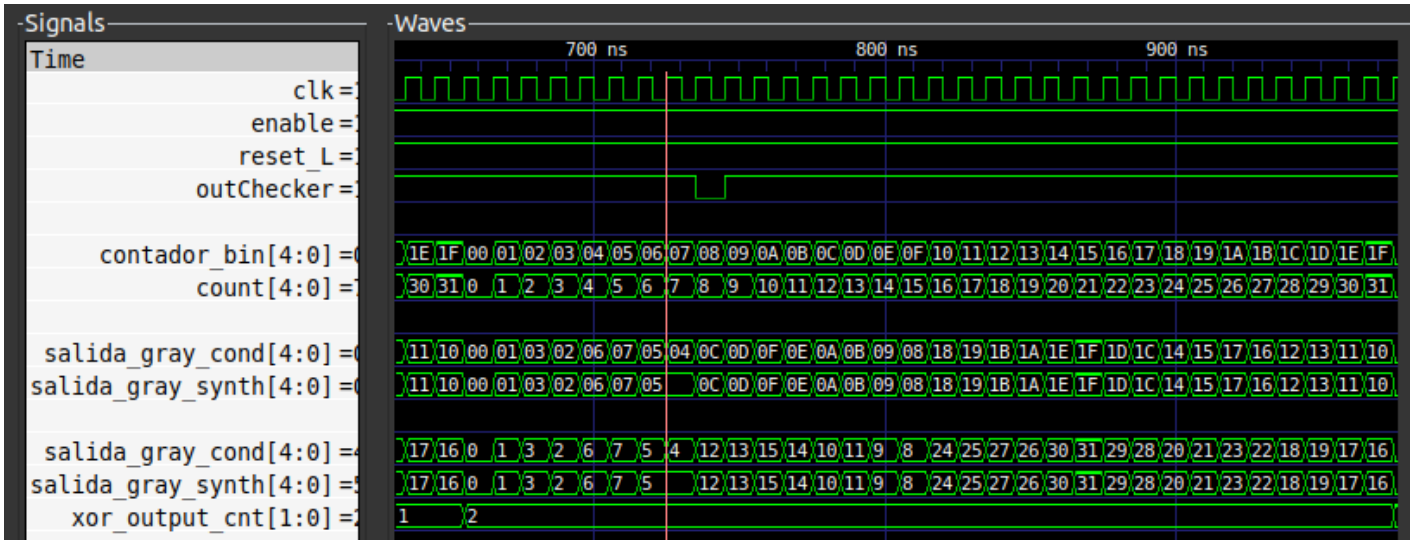

Figura 2: Salidas del contador en la tercera iteración donde se encuentra el error. (Creación Propia)

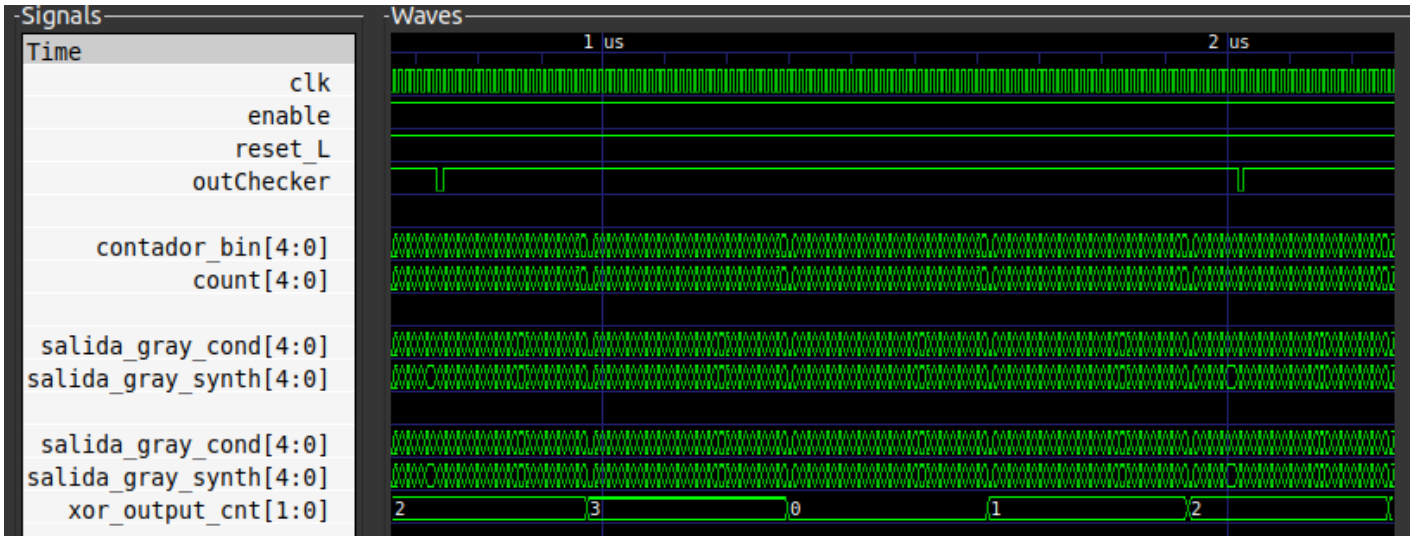

Figura 3: Salidas de comprobación que el error ocurre en la tercer iteración siempre. (Creación Propia)

7. Análisis y conclusiones:

Esta tarea cuenta con un "Catch", ya que el error se encuentra hasta la tercera iteración del conteo, por lo que si solo se simula para la primera iteración va a parecer que todo está bien; sin embargo, para asegurar el correcto funcionamiento del programa se deben tomar en cuenta todos los casos posibles, en este caso el contador hace 3 iteraciones para que vuelva a empezar a contar. Es por esto que entonces al verificar el funcionamiento se debe asegurar que las tres iteraciones de cuentas funcionen bien, y para este caso se encuentra en la tercera el error, donde la descripción estructural se salta el conteo del 4.