Universidad de Costa Rica
Escuela de Ingeniería Eléctrica
Circuitos Digitales II
Prof.Jorge Soto
II Ciclo 2020
IE-0523

Tarea #7

1. Diagrama y Diseño

El diseño de la tarea consiste en un bus de datos con parámetro de tamaño BUS_SIZE, el cual posee palabras por dentro del parámetro de tamaño WORD_SIZE. Como se observa en la figura 1. Se posee una salida de control correspondiente a una bitwise OR de los bits de cada palabra el bus. Además para la salida de datos se invierte el orden de las palabras del bus, como se ve en la figura 2.

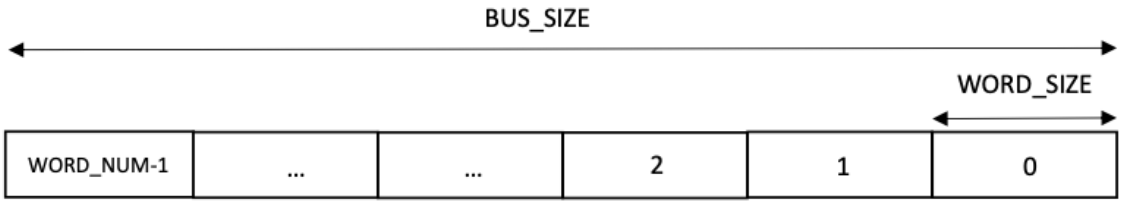

Figura 1: Bus de datos

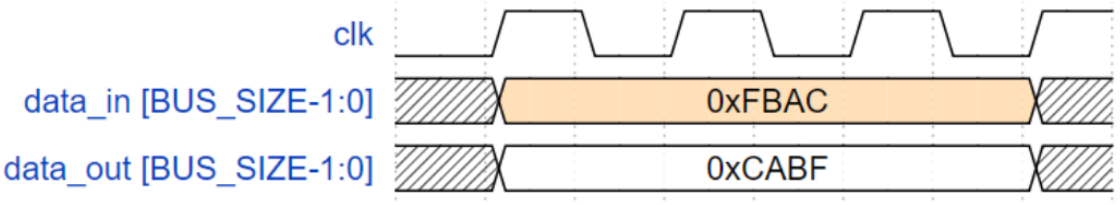

Figura 2: Salida de datos

En cuanto a la maquina de estados, se tiene como requisito que cada paquete contenga en la palabra más significativa (la primera de derecha a izquierda) una F, de lo contrario será un error F_ERR .Además que la palabra menos significativa (la ultima de derecha a izquierda) debe seguir el orden cronológico empezando por 0, de lo contrario será un error SEQ_ERR. Dicho esto, se cuentan con 4 estados donde RESET es cuando el reset está en bajo. FIRST_PKT es cuando ingresa el primer dato correcto y que posee su ultima palabra en 0. REG_PKT es cuando después del estado de FIRST_PKT, que se siguen ingresando datos correctos, siguiendo el orden cronológico en la palabra final. F_ERR es cuando se presenta el error de la palabra más significativa sin el valor de F. Finalmente SEQ_ERR es cuando se presenta el error de la palabra menos significativa no sigue la secuencia correspondiente.

En caso de que se presenten ambos errores de manera simultanea, el estado podrá ser F_ERR o SEQ_ERR. En la figura 3 del diagrama de tiempos se muestran las transiciones de estado y salidas esperadas dado el paquete recibido.

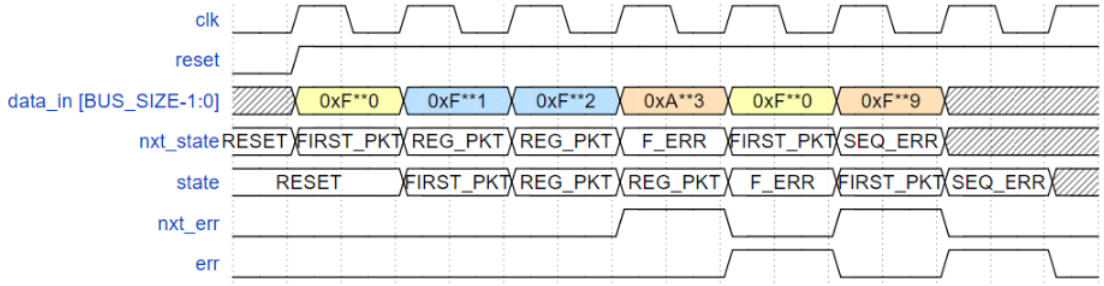

Figura 3: Diagrama de tiempos esperado

2. Plan de Pruebas

Para corroborar el óptimo funcionamiento del trabajo se crea un probador, el cual posee una estructura similar al de la figura 3, con la diferencia que se hace la adición de más señales al final de data_in para probar más casos. Entre lo que se pretende notar en la simulación, primeramente esta la verificación que cuando se esté en reset con señal baja, el data_out posea a la salida 0, el estado se encuentre en 0(estado RESET) y no se reporten errores.Se pretende observar la buenas transiciones de estados, además que si se pase a algún estado de error que esta señal se levante. Observar que todos los cambios sucedan en flancos crecientes. Verificar la equivalencia de la señal de próximo estado o próximo error. Además se requiere comprobar que la salida de control sea una OR de cada palabra el bus. Para observar el comportamiento se pretende utilizar la herramienta gráfica de gtkwave.

3. Instrucciones de utilización de la simulación

Para la automatización se crea el siguiente makefile:

```
all: yosys sed iverilog gtkwave
iverilog:
        iverilog -o prueba.vvp bancopruebas.v  cmos_cells.v
        vvp prueba.vvp

gtkwave:
        gtkwave probador.vcd

yosys:
        yosys -s box_bus_script.ys
        yosys -s m_d_e_script.ys

sed:
        sed -i 's/box_bus/box_bus_sintetizado/' box_bus_sintetizado.v
        sed -i 's/maquina_de_estados/maquina_de_estados_sintetizado/'
            maquina_de_estados_sintetizado.v
```

Para utilizar el makefile primeramente hay que dirigirse en la terminal a la carpeta que contiene estos archivos.Se utliza el comando **make yosys** para sintetizar el modelo conductual del bus y la maquina de estados. Posteriormente mediante el comando **make sed** se modifica el nombre de los modulos al sintetizado.Después utilizando el comando **make iverilog** se crea el archivo **probador.vcd**. Una vez creado el archivo probador.vcd correspondiente, se utiliza el comando **make gtkwave** y se abre el programa gtkwave, se seleccionan las señales que se quieren observar y finalmente se presenta la simulación deseada. En caso de utilizar el comando **make all** se realizan juntos todos los comando necesarios para la síntesis y gráficas.

4. Ejemplos de los resultados:

En la figura 4 se muestra la simulación con ayuda de la herramienta gtkwave. Se observa en este el modelo conductual y el sintetizado. Para la selección de estados, se utilizó la configuración one hot, por lo que el estado RESET es 0, FIRST_PKT es 1, REG_PKT es 2, F_ERR es 4 y SEQ_ERR es 8.

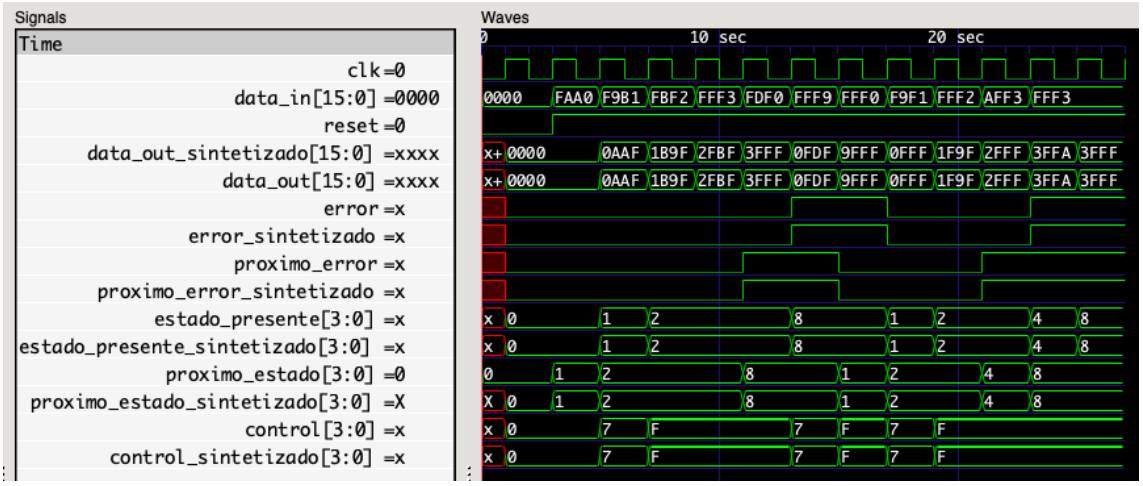

Figura 4: Resultados de simulación

5. Análisis y conclusiones

Analizando la figura 4,se puede observar que se cumple con todos los objetivos. Se empieza observando que todos los cambios están sincronizados de acuerdo al flanco creciente.Se sigue con la salida data_out donde comparando con la entrada data_in se puede apreciar que esta salida invierte el orden de las palabras en el bus de datos tal como se solicita como por ejemplo al inicio se tiene FAA0 y en la salida saca 0AAF. De acuerdo a la salida de control se puede observar que este posee un bit mapeado para cada palabra del bus, donde dicho bit corresponde a un bitwise OR de los WORD_SIZE bits de cada palabra el bus, como por ejemplo el primer dato de salida corresponde a 0AAF el cual sería 0111 que equivale a 7 . Con respecto al reset se observa que mientras esté en bajo, todas las señales permanecen en 0. Comparando el error con el próximo_error se muestra que ambas señales son equivalentes, con la diferencia que el error va atrasado un ciclo de reloj, lo mismo sucede

para el estado_presente y próximo_estado. Con respecto a las transiciones de la maquina de estados, se observa que primeramente se está en el estado RESET debido a que el reset se encuentra en bajo, cuando este sube dado que entra un dato válido FAA0 y pasa a FIRST_PKT. Posteriormente llega otro dato válido F9B1 y pasa a REG_PKT, donde se mantiene en ese estado porque recibe FFF3. Hasta que llega un dato inválido FDF0 debido a que no sigue la secuencia entonces pasa al estado SEQ_ERR.Después de que se detecta un error, la secuencia vuelve a iniciar en cero, por ende cuando le llega el dato FFF0 vuelve a su estado de FIRST_PKT, posteriormente con el ingreso de otro dato válido F9F1 se mueve al estado REG_PKT. Después vuelve a aparecer un error AFF3, pero en este caso es un error de que la palabra más significativa no posee una F, por ende se pasa al estado F_ERR.

Se puede concluir dado el análisis, que se cumplen todos los objetivos y se garantiza el óptimo funcionamiento del código. Cabe mencionar que fue fundamental el uso de genvar, generate y el for loop para el manejo de la estructura de esta tarea, que consistía en tener cajas(donde van las letras) dentro de un paquete(el bus de datos. Además para las salidas próximas fue fundamental el uso de always @(*), el cual hasta el momento no se le había dado mucho uso en las tarea anteriores. En la maquina de estados se destaca el uso de cases para manejar de una manera más sencilla cada estado en que se encuentre, además que para asignar los valores a estos estado se recomienda la codificación One Hot, dado que se evitan problemas.

6. Distribución del tiempo invertido en la tarea

En total se duró 8 horas con realizando la tarea

- Buscar información: 2 minutos ya que todo lo que se necesito fue repasar las clases y ejemplos anteriores
- Estudiando la información tomó 3 horas entender que era exactamente lo que se pedía
- Ejecutando lo que decidió hacer y probándolo se tomó 4 horas.
- Realizar el reporte tomó 1 hora.