Máquinas de estados

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 materia de digitales I para las máquinas de estados. Aproximadamente una hora tomó la búsqueda de información.
- **Usando la información**: Primero se diseña el bus de datos el cual es capaz de invertir las palabras. Para esto se usa un ciclo for, y en el fondo es exactamente igual que los multiplexores de tareas pasadas. Después para la máquina de estados es importante diseñar usando cases cada estado de forma individual. La elaboración de la 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. Descripción arquitectónica o diagrama del circuito.

Para esta tarea se pide realizar una máquina de estados y un multiplexor con parámetros. Primero para el multiplexor, se debe de tomar una señal de entrada de tamaño **BUS SIZE**, el cual posee palabras de tamaño **WORD SIZE**.Además se tiene una señal de control, la cual corresponde a un **OR** (bitwise) de las distintas palabras del bus de datos. En la siguiente figura se muestra la salida esperada y dada en el enunciado de la tarea:

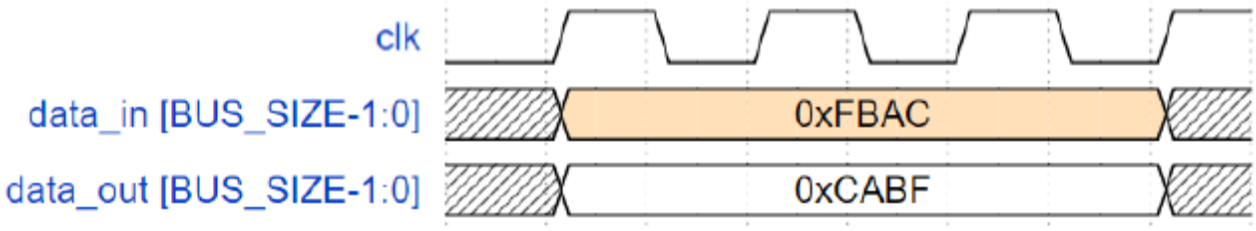

Figura 1: Salida esperada (Enunciado de la Tarea)

finalmente para la máquina de estados, se tomó como base las señales dadas en el enunciado de esta tarea donde se puede ver que la señal se encuentra en el estado **RESET** hasta que esta señal pase de cero a uno. Hecho esto, la máquina pasa al estado **FIRST_PKT** siempre y cuando se cumpla que las especificaciones de la palabra más significativa del bus esté compuesta por unos así como la última siga la secuencia.

Si esto se sigue cumpliendo, el estado se mantendrá en **REG_PKT**, hasta que este se encuentre o con un error en la condición de la primera palabra del bus y llegue a **F_ERR** o bien un error en la secuencia, lo que hace que la máquina se mueva al estado de **SEQ_ERR**. Ahora si la máquina se encuentra en alguno de los estados de error, y se tiene un dato en la entrada que cumple con las condiciones de inicio, se devuelve al primer estado **FIRST_PKT**.

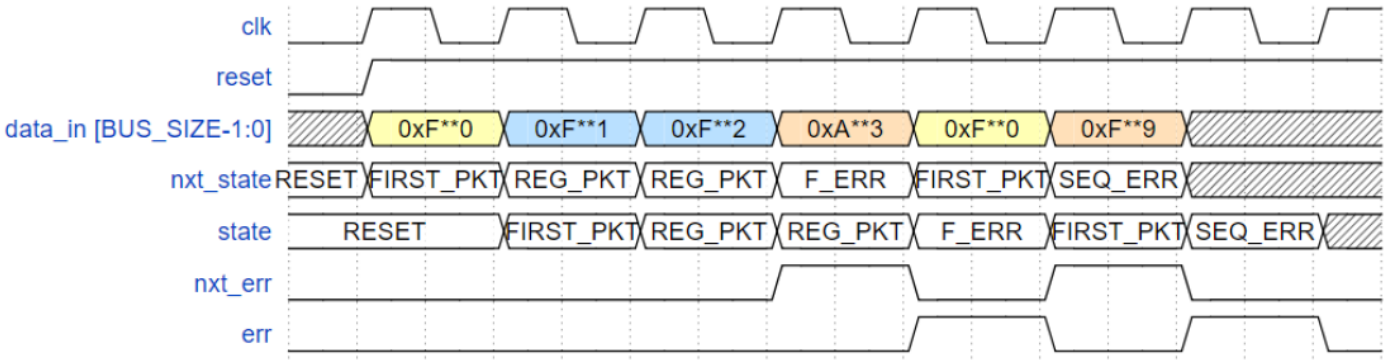

Figura 2: Salida esperada de la máquina de estados. (Enunciado de la Tarea)

3. Plan de pruebas

Para esta sección se implementó una lógica la cual generando las señales dadas en la figura anterior, se obtengan la salidas así como se presentan en la figura dada en el enunciado, aparte es importante recordar que se presenta la señal de control.

Se espera ver un correcto comportamiento de las señales, tomando en cuenta las transiciones de estados próximos y estados actuales, también la señal de error actual y próxima. Es importante notar que los parámetros se utilizan para hacer un cambio en el tamaño en los buses de datos y tener una mayor flexibilidad a la hora de realizar pruebas.

También se realizará una descripción estructural de los módulos para comprobar su funcionamiento a nivel fisico.

4. 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 Banco_Prueba.v
        ./a.out
        rm a.out
        gtkwave tarea7.vcd
```

5. Ejemplos de los Resultados.

Es importante tomar en cuenta que para evitar problemas con transiciones no deseadas se utilizó una asignación *one hot*, con la cual se espera no tener problemas.

Los valores que se le dieron a los estados son los siguientes:

- RESET = 4'b0000 = 1
- FIRST_PKT = 4'b0001 = 2
- REG_PKT = 4'b0010 = 3
- F_ERR = 4'b0100 = 4
- SEQ_ERR = 4'b1000 =8

La siguiente figura muestra las salidas para el multiplexor parametrizado:

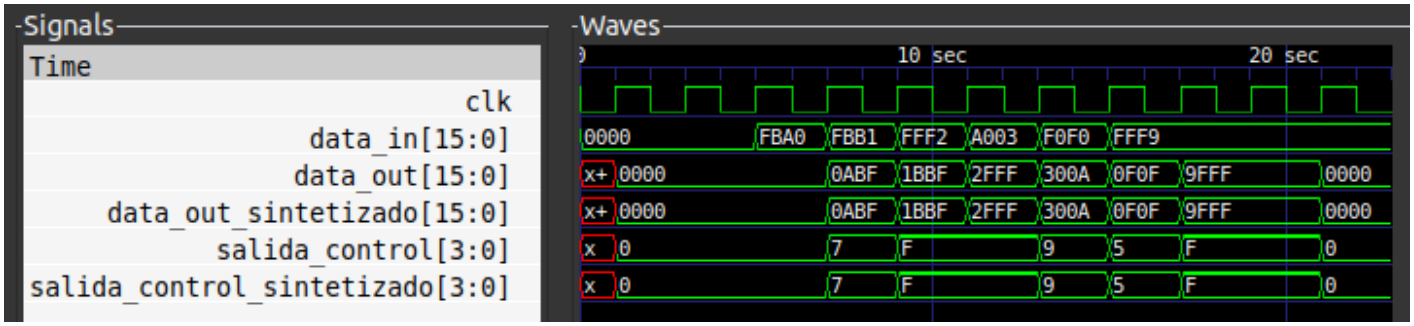

Figura 3: Salidas para el multiplexor (Creación Propia)

La siguiente figura muestra las salidas para la máquina de estados:

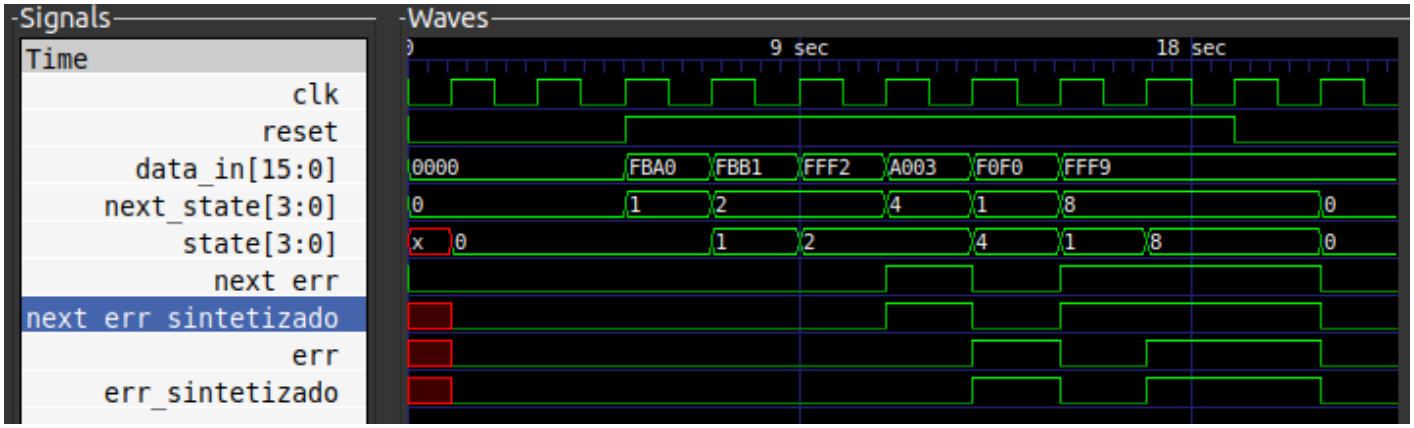

Figura 4: Salidas de la máquina de estados (Creación Propia)

6. Análisis y conclusiones:

De la figura 4 podemos notar un correcto funcionamiento del diseño, como podemos notar la señal está volteada y esta se transmite hasta un ciclo después debido al flanco del reloj, seguido a esto podemos ver que la señal de control genera una correcta salida del ORWISE para estas señales, podemos ver que la primera señal de salida es de 0ABF, el ORWISE de esta bus es de 0111 lo cual es equivalente en hexadecimal a 7, lo que nos permite demostrar el correcto funcionamiento de este módulo. Con respecto a la figura 5 podemos ver el funcionamiento de la máquina de estados ası como de las señales de error que representan la salida que esta genera. Primero podemos notar que la máquina se encuentra en el estado 0 el cual representa al RESET como se mencionó anteriormente, hasta que llega un conjunto de datos que posee las caracterısticas de diseño y para al estado 1, donde seguido se reciben dos señales que siguen cumpliendo los criterios de funcionamiento, por lo que se mantiene en el estado 2 por dos ciclos más. Después se recibe una señal que no cumple con los criterios de la primera letra, entonces se activa el estado 4, para después recibir una entrada válida y comenzar la secuencia. Por último podemos encontrar una señal que no cumple con el criterio de la secuencia por lo que se va a llegar al estado 8. De lo anterior podemos ver el correcto funcionamiento de la máquina de estados tanto sintetizada como conductual ya que estas generan las mismas señales de error las cuales se levantan en los estados 4 y 8. Por último, es importante notar que el diseño estructural realizado con yosys poseıa un diseño bastante diferente a lo planteado conductualmente, esto debido a que yosys sintetiza al máximo cada diseño conductual, pero es importante mostrar que las señales de error son iguales para ambos diseños a pesar de sus diferencias internas. Gracias a esta tarea se pudo entender mejor los conceptos y el uso de genvar y generate ası como el diseño de una máquina de estados lo que fue de gran ayuda y un reto diferente a las demás tareas.