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

Tarea #9

1. Diagrama y Diseño

El diagrama se muestra en la figura 1. El cual está compuesto de 2 módulos, empezando por sum_pipe. El sum_pipe posee la unión de 3 módulos, etapa 1 , etapa de transición y etapa2, siendo la etapa 1 la que posee las entradas y la etapa2 la que produce la salida. En la etapa 1 se suman las partes menos significativas de ambos buses y los almacene en un floop, además de igual manera almacenar el acarreo de dicha suma. Posteriormente la etapa de transición se utiliza para que se cargue correctamente la información a la etapa 2. Finalmente en la etapa 2 se suman las partes más significativas además del acarreo de la suma anterior y se almacenan en otro floop. El siguiente módulo diseñado corresponde al sumador el cual posee el módulo sumador, mencionado anteriormente, y el módulo identificador_completo. Dicho módulo corresponde a la unión de los módulos identificador e identificador2. En el módulo de identificador, se pasa la señal de idx a idx_d, esto almacenándolo en un floop, y en el módulo de identificador2 se hace lo mismo pero para pasar de idx_d a idx_dd.

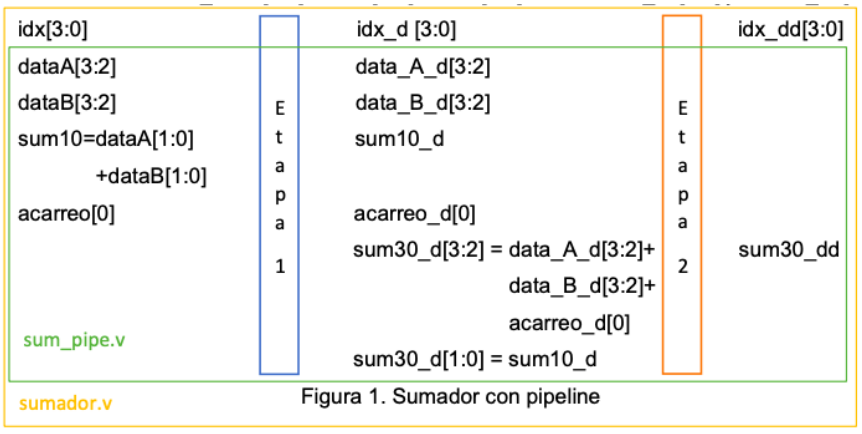

Figura 1: Diagrama de sumador

2. Plan de Pruebas

Para corroborar el óptimo funcionamiento del trabajo se crea un probador y un banco de pruebas para el módulo sumador, ya que este contiene al módulo sumador y al identificador_completo. Entre lo que se pretende notar en la simulación, primeramente esta la verificación que cuando se esté en reset con señal baja, la señal sum30_dd posea a la salida 0. En caso que la señal de reset esté en alto, comprobar que la señal sum30_dd después de

dos ciclos de transición(entre floop 1, el de transición y floop 2) corresponda a la suma entre las entradas A y B. Además de verificar el comportamiento esperado en los módulos de etapa 1 y etapa 2. En el de etapa 1 se pretender revisar que se sumen los bits menos significativos y se almacene en sum10, además que de poseer acarreo esta señal se levante. En cuanto a la segunda etapa se pretende revisar que efectivamente se sumen los 2 bits más significativos con el acarreo, la cual dé a la salida el resultado de la suma deseada. Se pretende observar que todos los cambios sucedan en flancos crecientes. Se espera que la señal de identificador idx, corresponda a idx_dd y que tarde la misma cantidad que sum30_dd para así garantizar que no haya problema con retardos, ni frecuencias. Por último se requiere asegurar que el modelo sea sintetizable, por lo que se utiliza la herramienta yosys para su respectiva síntesis. 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:

```
1  all: yosys sed iverilog gtkwave
2  iverilog:
3          iverilog -o prueba.vvp BancoPruebas.v  cmos_cells.v
4          vvp prueba.vvp
5
6  gtkwave:
7          gtkwave probador.vcd
8  yosys:
9          yosys -s sumador.ys
10 sed:
11         sed -i "s/etapa1/etapa1_synth/g" sumador_synth.v
12         sed -i "s/etapatransicion/etapatransicion_synth/g" sumador_synth.v
13         sed -i "s/etapa2/etapa2_synth/g" sumador_synth.v
14         sed -i "s/identificador/identificador_synth/g" sumador_synth.v
15         sed -i "s/identificador2/identificador2_synth/g" sumador_synth.v
16         sed -i "s/identificadortransicion/identificadortransicion_synth/g"
               sumador_synth.v
17         sed -i "s/identificadortransicion2/identificadortransicion2_synth/g"
               sumador_synth.v
18         sed -i "s/identificador_completo/identificador_completo_synth/g"
               sumador_synth.v
19         sed -i "s/sum_pipe/sum_pipe_synth/g" sumador_synth.v
20         sed -i "s/sumador/sumador_synth/g" sumador_synth.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. Posteriormente mediante el comando **make sed** se modifican los nombres de los módulos a su nombre sintetizado. 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.

4. Ejemplos de los resultados:

En la figura 2 se obtiene el comportamiento del módulo del sumador. Primeramente de acuerdo a las señales sum30_dd y idx_dd, se muestra un retraso de 4 ciclos de reloj antes de mostrar sus valores respectivos.Seguidamente de acuerdo a la señal sum30_dd se puede observar que su valor inicial es 0, posteriormente saca la señal deseada de la suma de 7 y 3 que corresponde a A en hexadecimal, después el resultado de D y 1 que corresponde a E y así consecutivamente. En cuanto a la señal idx_dd se puede apreciar que esta refleja la salida de idx 4 ciclos después producto de los retrasos de los floops. Además de ambas señales se observa un comportamiento idéntico con su versión sintetizada.

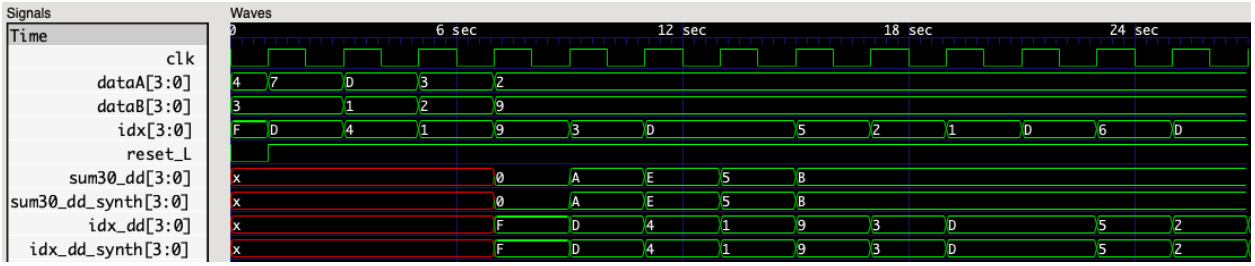

Figura 2: Comportamiento del módulo sumador

En la figura 3 se puede observar que dado que se suman los bits menos significativos de 7 y 3, osea 11 + 11 en binario, esto da un resultado de 10, con un acarreo. Tal como se muestra en la señal sum10 y acarreo en alto dos ciclos después de que la señal de reset se levanta. Posteriormente se realiza la suma de los bits menos significativos de D y 1 , osea 01 + 01 en binario, esto da un resultado de 10, sin un acarreo, y sigue el comportamiento para las demás señales.

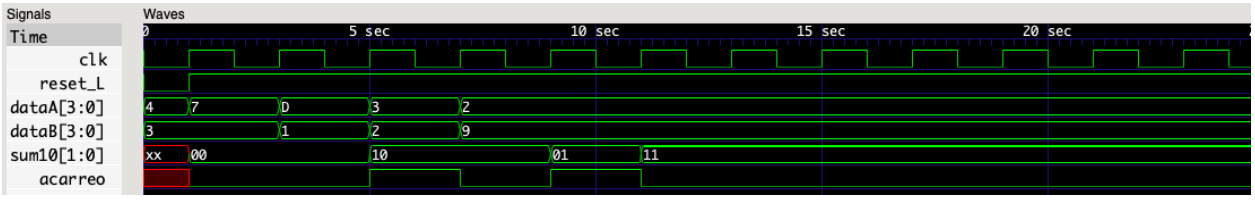

Figura 3: Comportamiento del módulo etapa 1

En la figura 4 se puede observar como sus entradas poseen retardos producto de la transición ente floops, además que a parte de la señal de clk, todas las entradas corresponden a las salidas de la etapa 1 mostrada en la figura 3, pasando estas por una etapa de transición. Además que se obtiene el resultado deseado de la suma de A y B a la salida de este, 4 ciclos después.

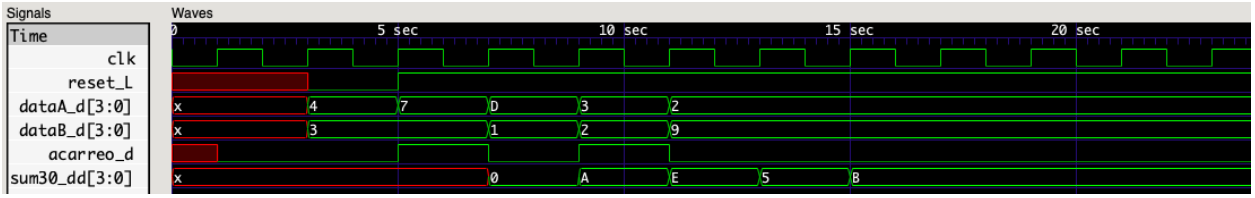

Figura 4: Comportamiento del módulo etapa 2

5. Análisis y conclusiones

Analizando la figura 2 se puede afirmar que la síntesis funciona de correctamente, esto debido a que tal como se muestra en esta figura, las salida del modelo conductal se comporta idénticamente al diseño estructural.Con respecto a estas señales de salida, sum30_dd y idx_dd en la figura 2, se nota que poseen un retraso de 4 ciclos de reloj antes de mostrar valores es de esperar,esto es dado a los retardos que produce cada floop. El valor inicial de la señal sum30_dd corresponde a 0, debido a que el reset empieza en 0, y se refleja 4 ciclos después debido a los retrasos de los floops. Se tarda tantos ciclos debido a que se está pasando la información de un floop a otro, donde para que se llegue a carga correctamente los datos del floop de la etapa 1 en la etapa 2 fue necesario implementar una etapa de transición, la cual hace que se retrase más pero logra el resultado deseado.Observando la señal de identificación idx, se puede comprobar que el circuito no posee errores en retardos ni frecuencias, dado que esta señal cambia 4 ciclos después a como es de esperar, debido a que sigue la misma ruta que el sum_pipe. Analizando la etapa 1, se tiene que esta a diferencia de las otras señales dura simplemente 2 ciclos, esto es debido a que en esta parte solo se ha utilizado un floop, además que cumple con lo correspondido de sumar los bits menos significativos de las señales A y B, además de almacenar el acarreo y pasarlos a la etapa 2, donde después de cargar los datos correctamente, este posee de salida el resultado de la suma.

En conclusión se puede decir que se obtuvo un buen diseño de sumador, ya que el respectivo cumple con los objetivos planteados, tanto para su diseño conductual como estructural. Se encontró que las salidas poseen un atraso del de 4 ciclos de reloj con respecto a las entradas, producto de los retrasos que producen los floops. Además se garantizó que no se poseen problemas de retrasos ni frecuencias, debido a que la señal de identificación reacciona de manera similar a la de la salida sum30_dd, esto se obtuvo variando constantemente el valor de esta señal. Se descubrió que al utilizar un pipeline y asegurarse que se carguen los datos de manera correcta entre floops, hacer etapas de transición facilita el proceso, sin embargo aumenta un poco el retraso.Se puede concluir de igual manera, la gran ventaja que se da al realizar un sumador con la técnica del pipeline, esto debido a que a pesar de que aumenta la latencia, se mejora significativamente el throughput. Además al hacer estas distinciones por etapas, hace mucho más sencillo buscar pulgas y posibles errores que presente ya que se sabe que es en una etapa en especifico, caso que no se podría notar de manera tan fácil si se realiza todo junto en un mismo módulo, aparte que ordena y facilita la comprensión del código.

6. Distribución del tiempo invertido en la tarea

En total se duró 5 horas con realizando la tarea

- Buscar información: 0 minutos
- Estudiando la información tomó 40 minutos, repasando la clase
- Ejecutando lo que decidió hacer y probándolo se tomó 3 horas con 20 minutos.
- Realizar el reporte tomó 1 hora.