Diseño de un Sistema de Conteo de 8 bits en Lenguaje Descriptivo de Hardware

Vargas Meza Aldo Alexandro, 213495653

El control de las variables, y el monitoreo de las salidas es una parte muy importante en el funcionamiento de un sistema, ya que esta parte es el último filtro de funcionamiento antes de que el circuito lleve a cabo la tarea para la cual fue diseñado. El uso de variables "temporales" es muy importante, porque sacrificando pequeños fragmentos de memoria, podemos optimizar un funcionamiento partiendo de una simulación.

I. Introducción

El programa a llevar a cabo, es un contador de 8 bits, con un reloj digital, una señal de reset y una señal "enable" que controlará la salida del sistema, entre tomar el valor del contador en ese momento, o en tener un valor de ceros.

II. Composición del código

El archivo fuente solo consta de un módulo, que es el del contador, puesto que se le agrego a este módulo una salida extra a las dos prediseñadas, que son la del estado actual y la del estado siguiente. Esta salida nueva, quedará aparte de las salidas de estados, y fungirá como un registro que controlado por la señal de enable, será el valor actual del contador, o un valor de ceros.

```
module contador(
  input wire clk,
  input wire reset,
  input wire enable,
  output reg[7:0]old_valor,    //El valor actual
  output reg[7:0]new_valor,   //El proximo valor
  output reg[7:0]out          //Salida del sistema
);
```

Imagen 01"Modulo contador"

El código contiene dentro del módulo 3 procesos always con diferencias en sus listas de sensibilidad y en las variables manejadas. El primero toma en cuenta el estado del clk y del botón de reset, en caso de sensibilizar estas variables, evalúa que el botón de reset esté desactivado, para ingresar el new_valor dentro de old_valor, que es 1 en el primer estado.
En caso de estar activado el botón de reset, old_valor=0, y new_valor=1.

```
always @(posedge clk or posedge reset)
  begin
    if(reset)
      old_valor=0;
    else
      old_valor=new_valor;
  end
```

Imagen 02"Always 01"

El siguiente proceso Always, encapsula solo a old_valor que cambiará cuando el botón de reset esté desactivado. El valor de new_valor, es siempre old_valor+1, por lo tanto es el que inicia el conteo.

```
always@(old_valor)
  new_valor= old_valor +1;
```

Imagen 03"Always 02"

El tercer proceso Always, finaliza el modulo con la condicional del enable, que se encargará de sacar old_valor por la salida del sistema out, esto solo si la señal enable está en estado lógico alto, de no ser así, la salida será cero.

```
always@(new_valor)
  begin
    if(enable)
      out= old_valor;
    else
      out=0;
  end
endmodule
```

Imagen 04"Always 3"

III. Simulación

Para simular, primero es necesario llevar a cabo la compilación del código y de cada uno de los módulos, así como una compilación completa de cada uno.
El software utiliza un compilador propio, que hace un chequeo de sintaxis y una vez compilados todos los módulos involucrados es posible llevar la simulación.
En este diseño, solo fue necesaria la implementación de un módulo, por lo tanto de un solo archivo para la compilación.

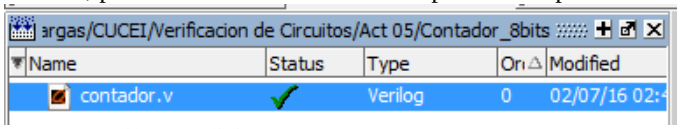

Imagen 06 "Archivos compilados"

Una vez compilado el archivo, se procede a la simulación de las siguientes variables añadidas a la curva de comportamiento.

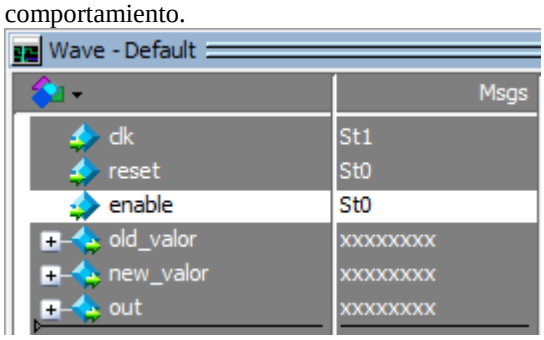

Imagen 07"Variables en la curva"

Las variables son modificadas para ser reconocidas como tal en el simulador, por ejemplo forzando sus estados para accionar el comportamiento que se necesitaría, tal como en la vida real.
El Clk se debe configurar como reloj digital, esto se hace en la ventana de la simulación, justo antes de dar el salto al siguiente estado en el tiempo.

IV. Comprobación de datos

Una vez simulado, es posible checar que los datos que salen sean los esperados, a nivel bit por bit.

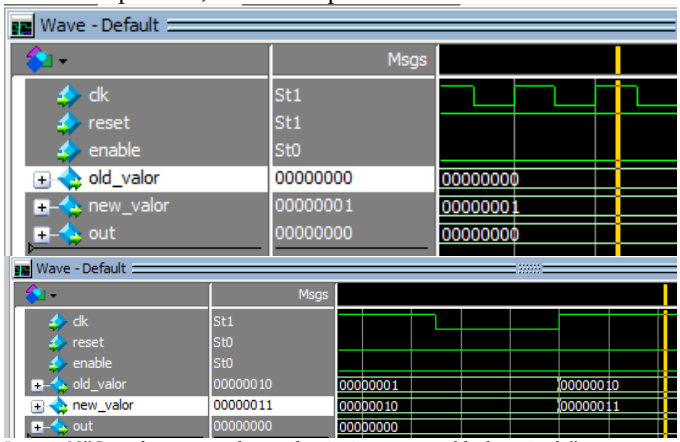

Imagen 08"Contadores en estado actual y siguiente, con enable desactivado"

En la imagen anterior se puede apreciar que ambos estados (actual y siguiente) se manejan completamente independientes de la salida del sistema, la cual al no estar activada la señal enable, conserva su valor de 0. En el momento que la señal enable se hace alta, la salida toma el valor del contador actual, y empieza a cambiar conforme el valor mismo, para regresar a ceros si el enable se desactiva.

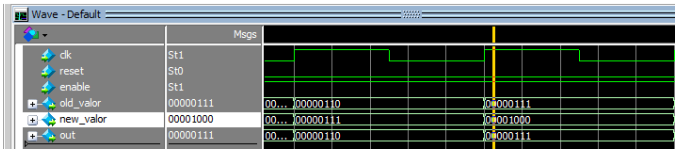

Imagen 09"Contadores en estado actual y siguiente, con enable activado"

"

V. Conclusión

El control que se maneja en este circuito me parece esencial para comprender el comportamiento de sistemas más avanzados.
Así como el uso de clk en un sistema, abrió la oportunidad para que tuviera que investigar al respecto, y aprender un poco más de cómo se comportan los componentes que creemos comprender en algún punto, pero siempre mejora un poco la comprensión de los sistemas avanzados, el diseño básico, y el control básico en un diseño.

VI. Bibliografía

Asic World. (s.f.). Obtenido de http://www.asic-world.com/verilog/operators1.html

ECCE. COLORADO. (s.f.). Obtenido de Colorado Education: http://ecee.colorado.edu/~ecen5837/cadence/problem_images/synth_code2.JPG

IUMA oficial. (s.f.). Obtenido de http://www.iuma.ulpgc.es/~nunez/clases-FdC/verilog/Verilog%20Tutorial%20v1.pdf

Tamu.EDu. (s.f.). Obtenido de http://students.cs.tamu.edu/tanzir/csce350/reference/verilog.html

Verilog Oficial Site. (s.f.). Obtenido de http://www.verilog.com/

Wikibooks. (s.f.). Obtenido de https://es.wikibooks.org/wiki/Programaci%C3%B3n_en_Verilog/