Diseño y Codificación de un Operador Lógico en Lenguaje Descriptivo de Hardware.

Vargas Meza Aldo Alexandro, 213495653

Desde 1970, la complejidad de los circuitos electrónicos ha ido creciendo exponencialmente con los años, eventualmente los diseñadores empezaron a requerir de un más alto nivel en las descripciones de hardware que fuera capaz de trabajar a una lógica compleja, sin estar atados a lo que establecía una tecnología tales como la CMOS o la BJT. De esta manera los Hardware Desing Language fueron creados para hacer posible el diseño de circuitos con un alto nivel de abstracción, y con la posibilidad de incluir en los modelos características propias de los circuitos electrónicos, tales como los flujos de datos y su variación en el tiempo.

I. Introducción

El lenguaje descriptivo de hardware, llevó el diseño de dispositivos electrónicos a otro nivel completamente más detallado y profesional, dotando a los ingenieros y diseñadores de los mismos de herramientas "finas" para la especialización de componentes.

En este trabajo se ejemplificará la lógica básica que sigue un programa pequeño en lenguaje descriptivo de hardware, el diseño se basa en el funcionamiento de un selector **[SEL]** de 4 bits, que cambia la operación lógica aplicada a las dos entradas **[A], [B]** realizada por el programa, entregando un resultado a la operación en una salida de 4 bits **[OUT]** que se ejecutara siempre y cuando la variable enable **[ON]** se encuentre activada.

```
module logicas(
  input wire [3:0]A,  //Entradas
  input wire [3:0]B,
  input wire [3:0]sel,  //Selector
  input wire on,      //Enable
  output reg [3:0]out //Salida
);
```

Imagen 01 "Señales Entrada y Salida"

II. Lógica de Ejecución

Al ejecutarse el programa, el primer paso esencial para el funcionamiento del mismo, es la condicional para la variable ON que se debe cumplir siendo esta estado alto. Si esta condicional no se cumple, la salida OUT será 0.

Si la condicional ON==1 se cumple, el programa cae a un proceso *always* , sensible a los cambios que puedan ocurrir en A, B o SEL. Dentro del proceso, un ciclo case se encarga de evaluar a SEL para llevar a cabo la operación lógica programada.
El selector ha sido programado para que cada uno de sus bits por separado de la función lógica deseada.

La combinación 0001 en el selector es la función lógica AND:

A	B	Salida
0	0	0
0	1	1
1	0	1
1	1	1

Imagen 02 "Tabla verdad AND"

La combinación 0010 en el selector es la función lógica OR:

A	B	Salida
0	0	0
0	1	1
1	0	1
1	1	1

Imagen 03 "Tabla verdad OR"

La combinación 0100 en el selector es la función lógica NAND:

A	B	Salida
0	0	1
0	1	1
1	0	1
1	1	0

Imagen 04 “Tabla verdad NAND”

La combinación 1000 en el selector es la función lógica XOR:

A	B	Salida
0	0	0
0	1	1
1	0	1
1	1	0

Imagen 05 “Tabla verdad XOR”

Una vez el selector obtenga un resultado correcto, ejecuta la operación lógica con los 4 bits de cada entrada A y B, y los deposita en la variable en el registro OUT.
Este proceso se repite cada que alguna variable de entrada cambia su valor

III. Compilación

Mediante el uso del software QuestaSim 64 bits Version 10.2c se codificó el programa, y se llevó a cabo una compilación de sintaxis y de errores lógicos, pasando esta exitosamente. El archivo que contiene el código del programa, se encuentra en formato .v dentro del proyecto asignado.

IV. Código

```
1   //Aldo Alexandro Vargas Meza
2   //Actividad 3
3
4   module logicas(
5     input wire [3:0]A,  //Entradas
6     input wire [3:0]B,
7     input wire [3:0]sel,  //Selector
8     input wire on,       //Enable
9     output reg [3:0]out //Salida
10  );
11
12  always @ (sel or A or B)
13    begin
14      if (on==1)
15        case (sel)
16           4'b0001:  //AND
17            out= A&B;
18           4'b0010:  //OR
19            out= A|B;
20           4'b0100:  //NAND
21            out= ~(A&B);
22           4'b1000:  //XOR
23            out= A^B;
24            default
25            out=0;
26         endcase
27      else
28        out=0;
29    end
30  endmodule
```

Imagen 06 “Código en Verilog”

V. Conclusion

El diseño de hardware fue revolucionado en el momento que se eliminaron las fronteras que aislaba a los mismos diseñadores. Con el lenguaje descriptivo, la lógica de sus procesos y la sencillez de los programas es posible desarrollar aplicación grandes que dependen de pequeñas contribuciones, así como el programa que se desarrolló en esta práctica, que depende de una serie de condicionales para poder funcionar. Es grato y emocionante tener la oportunidad de aprender el diseño a este nivel.

VI. Bibliografía

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/