Farid Santiago Carvajal Morales

RESUMEN PROYECTO 1 Y PROYECTO 2

RESUMEN PROYECTO 1:

En el proyecto se requiere tener compuertas a partir de otras, llegando así a un encapsulamiento de las anteriores por decirlo así, por ende se toma como primitiva la Compuerta Nand, de la cual se obtendrán las siguientes compuertas.

-Compuerta OR:

Para esta compuerta se analiza la compuerta NAND sus entradas y salidas, por lo cual se llega a la conclusión que se necesitaría negar sus entradas y luego volverlas a negar para que se convierta en una compuerta or.
A NAND A = A´.
B NAND B = B´.
Luego de esto se tiene que unir las salidas de A y B para que se nieguen 2 veces y por álgebra booleana de la suma que representa a la compuerta OR.
A´ NAND B´ = (A´*B´)´=A+B
luego se hace su respectiva comprobación en el programa ModelSim, comprobando con su tabla de verdad.

-Compuerta NOT

Como se explicó anteriormente, en esta compuerta se hizo el mismo análisis de la compuerta or, a lo cual se llegó a la conclusión de que si se tiene una sola entrada y se la une con la otra entrada de la compuerta nand se tendrá la compuerta Not así:
A NAND A = A'.
luego de esto con el programa ModelSim se comprobó su funcionamiento, comprobando sus 2 casos y obteniendo un resultado favorable

-Compuerta AND

para esta compuerta como las anteriores se analizó y se llega a la conclusión de que si se hace uso de una Nand con las 2 entradas, se obtendrá la multiplicación negada de estas así:
A NAND B = (A*B)´
si después de esto se vuelve a colocar la salida de dicha nand en otra nand pero colocando las 2 entradas de esta la misma salida de la otra se negara otra vez, por lo cual hará que se tenga una compuerta AND así:
(A*B)´ NAND (A*B)'= A*B.
Luego de esto se simula sus respectivos casos y se comprueba con la tabla de verdad de dicha compuerta.

-Compuerta XOR

para esta compuerta se tiene los conocimientos de como hacer la NOT,AND Y OR, por lo cual basándose en las anteriores explicaciones solo se tomara el uso como compuertas pero sabiendo como se usaron anteriormente, para esta compuerta como las que siguen, entonces con ayuda de 2 señales se tiene que:

n0 = x and (not y);
n1 = (not x) and y;
por lo cual hasta el momento obtendremos 2 and, con las mismas 2 entradas pero diferentes entradas negadas, luego necesitamos unir estas 2 and, para lo cual se hace uso de la compuerta or, y así teniendo:
f = n0 or n1;
así se obtiene que sus entradas si son iguales se tenga un 0 de lo contrario 1, como anteriormente se usó el programa ModelSim para comprobar su respectivo funcionamiento.

- MULTIPLEXOR

en este caso se conoce que este multiplexor debería tener varias entradas y una salida, con la cual es seleccionada por las entradas de selección, por lo cual se tiene que para este proyecto se cuente con un multiplexor de 3 entradas por 1 salida, digo 3 por lo que una de ellas va a ser la de selección, se define sus entradas y salidas en la entidad correspondiente.
Luego de esto se hace uso de la arquitectura en el código, donde se presentaron 2 señales, las cuales ayudarán al momento de unir todas las compuertas que se usarán.
con ayuda de la tabla de verdad se pudo ver cuando el multiplexor dejaba pasar una entrada, por lo cual se analizó las tablas de verdad de and y or, llegando a la conclusión de que se necesitaria 2 and,1 not y una or, de la siguiente manera:
sel = selección, x = entrada, y= entrada , f= salida
x1 = señal de ayuda, x1 = (not sel) and x,
x2 = señal de ayuda, x2 = sel and y,
f = x1 or x2
por lo cual al negar la selección y ponerla con x en una and, depues hacer una and entre sel ,y ,al tener al final que unirlas con un or, se tiene el esperado multiplexor que varía de acuerdo a la selección, que mostrara cual es la posición con la cual debe tener su salida.
Para los casos siguientes se tendrá en cuenta que (x,y,sel) serán entradas y (f) las salidas, a menos que se indique lo contrario.

-DEMULTIPLEXOR

Este toma lo contrario al multiplexor, puesto que este tiene pocas o una entrada y tiene muchas salidas, en este caso en particular se tomará con 2 entradas (x,sel), una entrada y una de selección, con 2 salidas (f,g);
como con el multiplexor se basó en las tablas de verdad para llegar a hacer el demultiplexor, en este caso las salidas casi toman el lugar de las señales en el multiplexor, teniendo la siguiente manera:
f = (not sel) and x,
g = sel and x,
teniendo así al demultiplexor, luego de esto se entró al programa Modelsim, donde se toman señales como si fueran pruebas para modelar el circuito, por lo cual se declaran los componentes, se toman los mismos nombres de las variables de entradas y salidas de las compuertas a testear, luego se crean las señales a testear tiene que ser la misma cantidad de entradas como de señales del código del demultiplexor, luego de esto se crea un mapeo en el cual el programa toma los puertos de otra entidad para poderlos mapear en la entidad correspondiente del test,
Luego de esto se toma una instancia llamada dut1, la cual se usa para conectar los puertos del multiplexor con las señales declaradas en el test.

ya casi por acabar se toma la función:
Stimulus : process
el cual se usa para comprobar bajo ciertas condiciones si el código funciona, esto siguiendo una secuencia de datos.
Después se establecen valores y se comprueba con la tabla de verdad para poner los valores de entrada y salida como de selección y verificar al final si el circuito funciona.
los pasos mencionados en la parte de ModelSim, describen a la mayoría de los siguientes circuitos y circuitos anteriores por lo cual a los demás circuitos solo se especificara que cambio tuvieron en la parte del testeo mas no desde un principio.

-Not16
para este caso y la mayoría de los que se tienen a continuación se hace uso de un bus de datos, que en este caso se representa de manera de vectores en el código, por lo cual se define una entrada y una salida pero como vectores de 16 bits, por lo cual se toma en cuenta el 0, por lo tanto va del 0 hasta el 15. Como siguiente paso se realiza la asignación de cada entrada con su respectiva salida negada del vector de salida.
por último se testeo, como antes se explicó se realizaron los pasos para el código de testeo, teniendo en cuenta que las entradas y salidas son vectores, por lo cual se las puso así, además de que a las señales igual se las uso como vector y se las inicializan en 0 en todos sus 16 bits.

-AND16
Para este caso es muy similar que con el not, sus 2 entradas se las asignó como vectores de 16 bits, al igual que su única salida, luego de esto en la estructura se asignó a cada entrada su respectiva salida, además de que se llamó al componente de la compuerta principal AND (descrita anteriormente), esto también se hizo en la compuerta Not16 y se hará en las compuertas Or16 y multiplexor16 ,, además de llamarla por la posición del bit del vector como en la compuerta Not16, teniendo así que no hacer el proceso de la compuerta AND con las nand, dado que el programa lo hará por nosotros.
de otra forma de ver es que la compuerta AND está encapsulada y la llamamos cuando queramos para que funcione y no necesitaríamos hacer el circuito uno por uno, lo cual ayuda mucho al hacer el código.
Para el testeo se hizo lo mismo que con el Not 16, solo que usando las asignaciones correspondientes para la and.

-OR16
con esta compuerta se hace uso de las 2 entradas y una salida, cada una como vector de 16 bits y como se ha mostrado y especificado en las compuertas AND16 Y NOT16 se hace su mismo proceso de llamado de las compuertas anteriores, pero en este caso será la Compuerta Or y se la usará con cada posición del vector de entrada y salida.
lo mismo que en los 2 anteriores casos sucede en este caso para el testeo.

-MUX16
como en los 3 últimos casos, se llama a su entidad correspondiente siendo el caso de Multiplexor antes descrito y se le asigna a cada posición de los vectores de entrada y salida sus respectivas entradas y salidas para el Multiplexor, esto como en la parte del testeo que se explicó en el Demultiplexor, en cual se llama a la instancia y se desarrolla un proceso.

Por último para el testeo se realizó como en los 3 anteriores casos y luego se camilo y ejecutó.

-**OR8WAY**

como se explicó anteriormente con el Or16, en este caso se hará lo mismo pero ya no con 16 bits, si no con 8, por lo cual va desde el 0 hasta el 7 en el vector, dado que cuenta el número 0, pero en este caso, en la estructura se tendrá que x,y (vectores de 8 bits de entradas) los cuales se negras y se unirán con una compuerta nand, por cada posición del vector, formando así un bus de datos a la salida, donde se llegará a una salida de 8 bits.
En el testeo se toma en cuenta esto y se cambia sus entradas y salidas por 8 bits, además de asignar los 8 bits a las señales.

-**Mux8Way16 y Mux4Way16**

como se explicó en anteriormente el multiplexor necesita una o varias entradas de selección para el caso de mux8way16 necesitará de 3 entradas puesto que representar las 8 entradas en binario, y para el caso de mux4way16 necesitaría de 2 entradas pues representaría a las 4 entradas en binario, luego de esto se tiene que crear sus respectivas entradas y salidas, siendo de 16 bits en ambos casos y teniendo una sola salida para ambos, pero con la pequeña diferencia de que en el Mux4way16 tendrá 4 entradas y en el otro caso 8.
en la parte de las estructuras, se tiene que se realizará con un Proceso y con Case, en el cual determinamos para cada valor de la selección su respectiva salida, dejando pasar una entrada a la salida, por lo mismo se la asigna, por ejemplo siendo la selección 00 en el Mux4way dejaría pasar la entrada w, por lo cual f tomaría los valores de w.es lo mismo para los 2 casos de Mux4way y Mux8way.
En la parte del testeo, se deberá realizar la comprobación basándose en la selección y determinando que entrada dejaría pasar para convertirse en la salida de 16 bits.

-DMux4Way y DMux8Way

Estos circuitos son similares en el proceso de explicar cómo se desarrollaron como el Mux4way y ,Mux8way, con las notables diferencias que en este caso se tendrá la selección como entrada y una entrada en particular y teniendo 8 y 4 salidas, con 3 entradas de seleccion 2 entradas de seleccion para DMux8way y Dmux4way respectivamente, ahora se hizo el uso de IF y dependiendo del valor de selección se fue asignando el valor de las salidas, teniendo como ejemplo que para w entrada y las 3 entradas de selección en 0, la entrada que dejara salir será la primera salida, o de otra forma la salida F1, para el caso de DMux8way, para el testeo es muy parecido en cuanto a los 2 circuitos anteriormente explicados.

En este documento se realiza el resumen de las compuertas y circuitos posteriores, teniendo en cuenta el diseño de cada uno de ellos por lo cual para entender mejor y de manera óptima se deberá ir leyendo el resumen y ubicando el código del cual se está resumiendo.

PROYECTO 2 ALU

En este proyecto se realiza primero un medio sumador en el cual se usa de forma práctica y sencilla una compuerta and y una xor, también se realizó su análisis de tabla de verdad como se mencionó en el anterior proyecto, teniendo hecho el medio sumado, hace falta completar el sumador, para ello se usaron 2 sumadores en además de otra entrada puesto que para que sea un sumador completo deberá sumar 3 bits para llevar un acarreo mayor al

del semisumador, para esto se realizó el sumador completo, luego se procedió a comprobarlo con un testeo, por consiguiente ya se tiene gran parte para llevar a cabo la ALU, por ende se procede a hacer el Add16 el cual básicamente es un sumador completo que está hecho de medios sumadores, la diferencia radica en que este se hace para sumar por decirlo así 2 trenes de 16 bits, ya no solo 3 bits como en el sumador completo, para esto se usaron vectores y se adaptaron siguiendo las instrucciones del repositorio de GitHub, en el cual básicamente la tercera entrada para la posicion 0 del vector debe ser igualada con 0 puesto que al sumar 2 bits debe dar como maximo 10, no llegara a exceder, sin embargo al pasar a 3 bits como 11 y mas el que se lleva de acarreo se tendra que igualar la tercera entrada en acarreo para que se lleve una correcta suma.
Luego de esto se implementa el último circuito antes de llegar a ALU, donde estarán todos los anteriores circuitos que se mencionaron, este es el Inc16, en el cual básicamente incrementa 1 de 1 en uno por cada suma que de, dando así un incremento que hará que en su salida sea diferente a la entrada por un bit de diferencia que empezó al inicio, esto gracias a que una de las entradas se la inicializo en la primera posición del vector en un valor verdadero o por decirlo de otra forma con un 1 binario.
por último se tiene la creación de un ALU la cual es el principal motivo por el cual se crearon los anteriores circuitos, en esta debe tener cada uno de los anteriores procesos y por ende sus funciones, ya que esta ALU es la principal para un computador o cualquier dispositivo electrónico que tenga algún procesador, por lo cual se definieron muchas entradas y se hicieron los respectivos cambios en la arquitectura según el repositorio y se esperaba emplear cada una de las compuertas y circuitos usados, para luego llegar a poderse usar de manera práctica, esto llamando a sus instancias y creando uno por uno sus respectivos procesos y además de esto probando con los test bench cada uno de los circuitos que fueron mostrados anteriormente, para llegar así a una combinación total de los anteriores circuitos