

TRABAJO PRÁCTICO IC3 2022 (MEMORIA)

EJERCICIO 1

LISTADO DE FICHEROS DE CÓDIGO:

- entity1.vhd
- architecture1.vhd
- circuit_gates1.vhd
- architecture_1d.vhd
- test_bench1.vhd

SOLUCIONES:

1.a

Las funciones lógicas están simplificadas para que el código VHDL resultante sea más compacto:

$$F = Z\bar{X} + XY \text{ (factor común y complementariedad)}$$

$$G = \bar{X}(Z + \bar{Y}) \text{ (Karnaugh, factor común y complementariedad)}$$

Son el resultado de simplificar las funciones F y G en forma normal disyuntiva. La entity está en el archivo "entity1.vhd". Los nombres de las salidas y entradas declaradas en la cláusula port son los mismos que los del enunciado.

Las funciones F y G son independientes entre sí pero también existe la posibilidad de aprovechar una de las dos para hacerla función de la otra si se ahorran puertas lógicas con ello. Por ejemplo, si en la primera fila de la tabla de verdad G valiera 0, la función G podría haber sido: $G = F\bar{X}$ ahorrando una puerta not y una or.

1.b

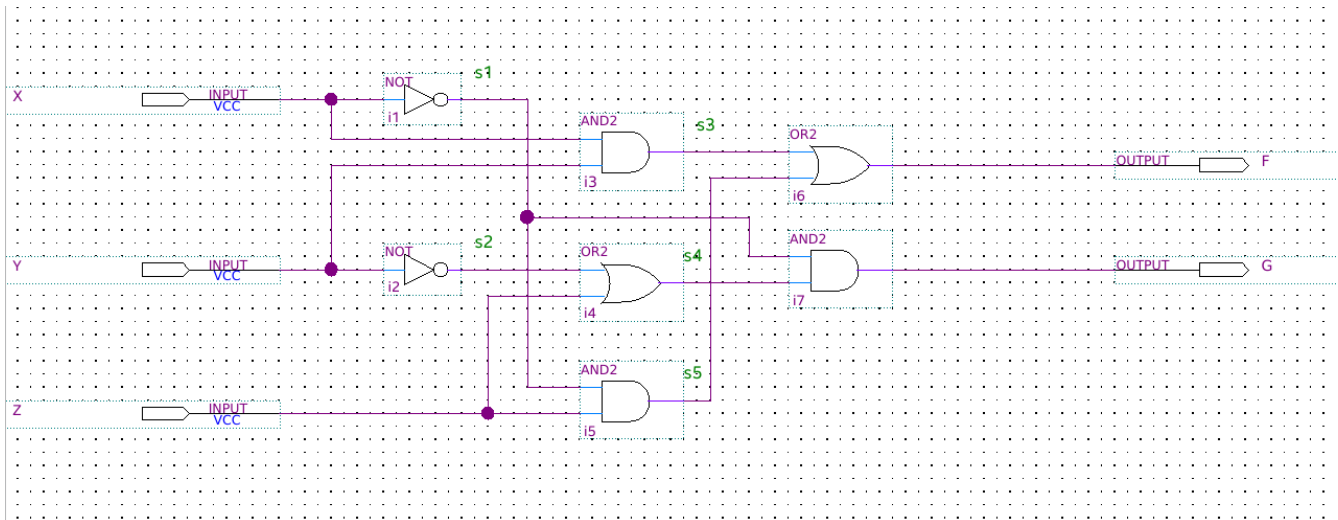
La architecture está en el archivo "architecture1.vhd". El comportamiento está especificado con dos asignaciones concurrentes, una para la salida F y otra para la salida G, traduciendo directamente las dos funciones lógicas anteriores:

$$f \leq (z \text{ and not } x) \text{ or } (x \text{ and } y)$$

$$g \leq \text{not } x \text{ and } (z \text{ or not } y)$$

1.c

El diagrama está hecho con el programa quartus prime de Intel.



Las entity y architecture para las puertas lógicas están en el archivo "circuit_gates1.vhd".

En él está descrito el comportamiento de la puerta not y las puertas and y or de dos entradas usando asignaciones concurrentes con sus correspondientes operadores lógicos. La descripción es directa ya que VHDL provee los operadores not, and y or.

1.d

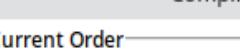
La architecture está en el archivo "architecture_1d.vhd".

Aquí se declaran los componentes del ejercicio 1.c y se instancian concretamente 2 puertas not, 2 or de 2 entradas y 3 and de 2 entradas siguiendo la misma nomenclatura que las etiquetas del diagrama del ejercicio 1.c, cada puerta está etiquetada con i1 hasta i7 y las señales intermedias con s1 a s5. La asignación de las señales en las clausulas port map está hecha siguiendo el diagrama dibujado en el apartado 1.c

1.e

El archivo que implementa el banco de pruebas es "test_bench1.vhd". Como la entity del primer apartado está en un archivo separado se puede reutilizar en este banco de pruebas para probar tanto la architecture del apartado b como la del apartado d, basta con cambiar el orden de compilación de las architectures. En el código se le asigna al vector de test "test_in" cada uno de los posibles valores utilizando un bucle for y convirtiendo la variable i de integer a unsigned y de unsigned a std_logic_vector de tal manera que cuando i es 0 el vector de test es 000 y cuando i es 7 el vector es 111.

A continuación se muestra una captura con los ordenes de compilación. En la izquierda, el último archivo de arquitectura en compilar es architecture1.vhd por lo que ésta será la arquitectura utilizada en la simulación, en el de la derecha el último es architecture_1d.vhd.

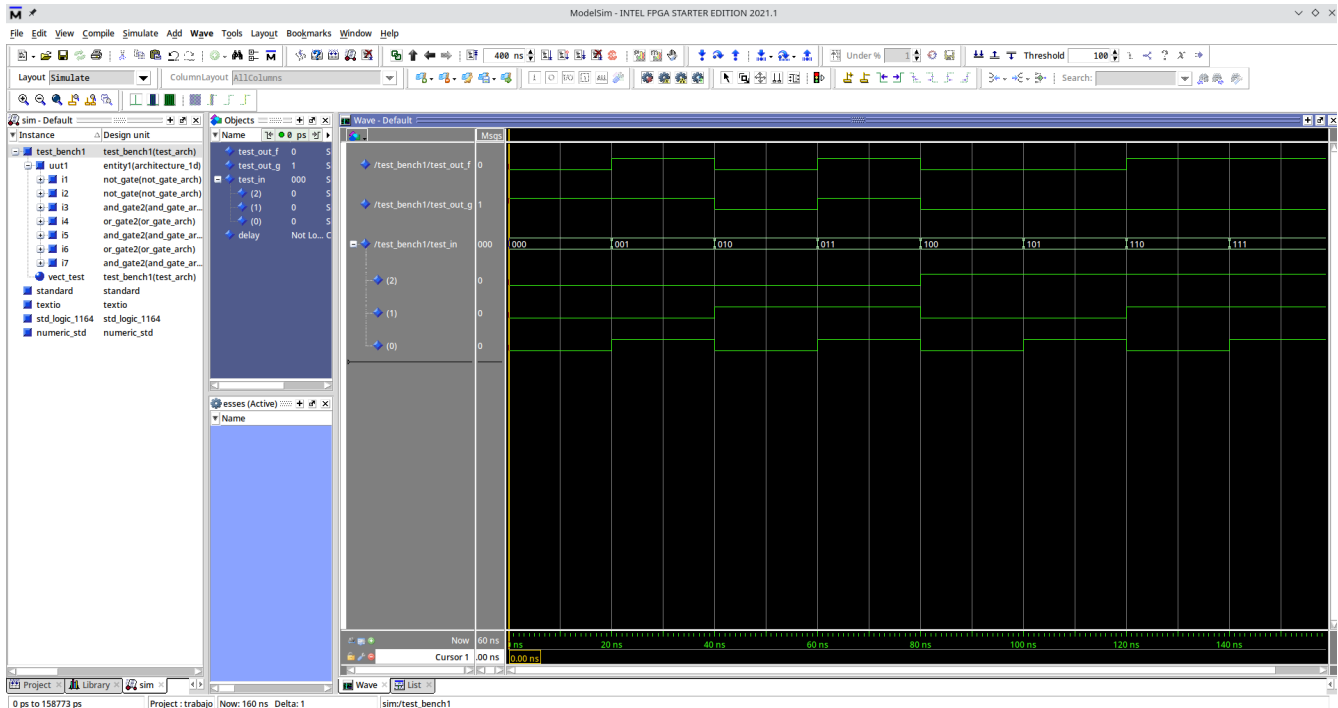


Compile Order

Current Order

- entity1.vhd
- circuit_gates1.vhd
- architecture1.vhd
- architecture_1d.vhd
- test_bench1.vhd

Cronograma usando la architecture del apartado 1d.



EJERCICIO 2

LISTADO DE FICHEROS DE CÓDIGO:

- entity2.vhd
- test_bench2.vhd

SOLUCIONES:

2.a

El archivo con la solución es “entity2.vhd”, contiene tanto la entity como la architecture.

Los puertos res, a y b son de tipo signed porque facilita el calculo del sumador, el resto son std_logic.

Hay 3 señales r, d y z que sirven para hacer los calculos de las funciones logicas para el resultado, el desbordamiento y la señal cero respectivamente.

La señal r es la suma de los operadores a y b, (+1 cuando hay acarreo de entrada).

La función lógica de la señal d vale 1 cuando el signo de a y b es 1 y el del resultado es 0 o cuando el signo de a y b es 0 y el del resultado 1, es decir, cuando los operandos tienen mismo signo y el resultado lo tiene diferente.

La señal z es 1 cuando no hay desbordamiento y el resultado es 0.

El signo es el mismo que el del resultado cuando no hay desbordamiento y el contrario cuando sí lo hay.

2.b

El banco de pruebas está en el archivo "test_bench2.vhd".

Para probar todas las posibles entradas al circuito, la metodología es la misma que en el ejercicio 1. El banco tiene un bucle que va desde el valor máximo posible del vector de test hasta el mínimo. La longitud del vector es $2*N+1$. El bit más significativo es la entrada cin, mientras que los siguientes $2*N$ bits son los operadores a y b en este orden. Al usar el bit más significativo como acarreo de entrada se consigue más legibilidad en el cronograma resultante ya que la mitad de la simulación tiene valor 0 y la otra 1.

A demás del vector de test, para probar las salidas esperadas hay una variable por cada salida dentro de un bloque process llamado vect_test2. También hay variables para los operadores y para contar el número de errores.

Las variables que intervienen en la comprobación del desbordamiento son op_a, op_b y op_r que representan el operando a y b, y el resultado. Primero se convierte cada operando de signed a integer y se almacena en las variables anteriores, después, se suman y se almacena en op_r, de esta manera, al utilizar el tipo integer la operación resulta en un número por encima del rango permitido si hay desbordamiento, siendo el rango permitido $-2^{N/2}$ a $2^{N/2} - 1$ ya que la representación del tipo signed es en complemento a 2. Para que el método sea válido es necesario hacer la simulación con un valor de N que no provoque la superación del rango de un entero.

En la comprobación de las salidas esperadas se muestra un mensaje de error que indica la salida que produce el error y los valores de las señales relevantes.

Cronograma para $N = 4$.

