Sistemas de Tempo Real: FPGAs
2013/14

Práctica 2: Construcción de una ALU simple de 4 bits

Objetivo

En esta práctica se pretende reafirmar los conocimientos adquiridos en la Práctica 1 y lo visto en el tutorial de VHDL de las clases de teoría. Con este propósito se va a diseñar una ALU de 4 bits que debe permitir realizar operaciones sencillas.

Enunciado

Una ALU (*Arithmetic Logic Unit*) es un circuito digital que lleva a cabo operaciones de aritmética entera (sumas, restas, multiplicaciones, divisiones), lógicas (AND, NOT, OR, XOR…) y de desplazamiento de bits. Su estructura puede verse en la Figura 1, donde A y B son los valores de entrada, R es el valor de salida, F indica la operación a realizar y D señala el estatus de la salida (acarreo de salida, overflow, división por cero…).

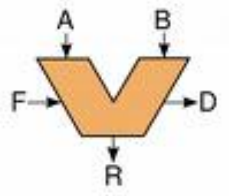

Figura 1: Esquema básico de una ALU.

Se pide:

1) Crear un nuevo proyecto en ISE (p2) con un módulo VHDL que contendrá una arquitectura con todas las funciones requeridas en el apartado 3.

2) Diseñar e implementar la estructura de la ALU (entradas/salidas) teniendo en cuenta que se trabaja con registros de 4 bits.

3) Diseñar e implementar el comportamiento de la ALU para el caso de realizar las operaciones NOT(A), AND, OR, XOR, A + B, A – B y desplazamiento de 1 bit a la izquierda o derecha de la entrada A. Dado que A deberá de ser de tipo STD_LOGIC_VECTOR, se precisará implementar las funciones de desplazamiento (SLL y SRL sólo están implementadas para elementos de tipo BIT_VECTOR). Además, se supondrá que en cada ciclo de reloj sólo se realizará una operación: se leen las entradas, se aplica la operación y se devuelven las salidas correspondientes. En cuanto a la salida D, únicamente se pondrá a 1 en el caso en que haya overflow al realizar una suma.

4) Añadir las librerías IEEE.STD_LOGIC_ARITH y IEEE.STD_LOGIC_UNSIGNED en el caso de que el ISE no las declare al crear el proyecto.

5) Crear un **Test Bench** en el que se pueda verificar el correcto comportamiento de las operaciones realizadas con la ALU. Para realizar esto se utilizará un reloj a 10 MHz. Se debe simular durante al menos 1000 ns, durante los cuales A y B tomarán distintos valores y F variará entre 000 y num_max, donde num_max es la representación en binario del número total de operaciones implementadas para la ALU menos uno.

Para introducir los distintos valores de A, B y F, tras crear el Test Bench, se procederá de la siguiente manera:

- A) Se configurará CLK_period en función de la frecuencia de reloj indicada.

- B) En el proceso *stim_proc*, proceso encargado de inyectar los estímulos en nuestro código vhdl principal, se eliminará el contenido entre el *begin* del proceso y la última sentencia *wait;* (sin eliminar este último wait).

- C) Se rellenará *stim_proc* desde el *begin* con los estímulos oportunos. Por ejemplo, en la Figura 2 puede verse como quedaría *stim_proc* cuando se quiere probar la operación NOT(A) y un AND de A y B. La ejecución de este Test Bench da lugar a la simulación de ISIM que se muestra en la Figura 3.

```
-- Stimulus process
stim_proc: process
begin
   -- insert stimulus here
   wait for 40 ns;
   A <= "1010";
   F <= "000";

   wait for 100 ns;
   A <= "0010";
   B <= "0111";
   F <= "001";


   wait;
end process;
```

Figura 2: Ejemplo de estímulos de entrada.

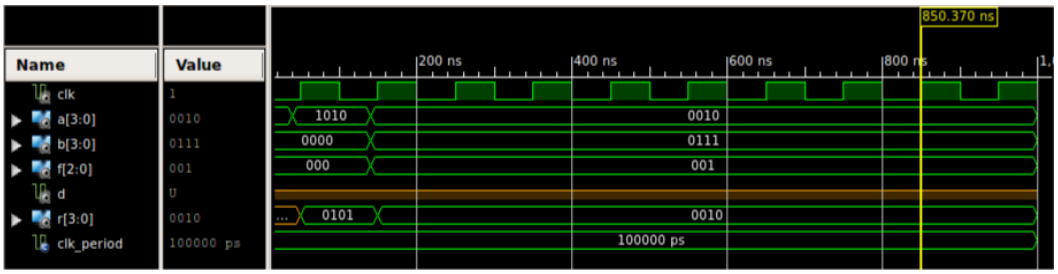

Figura 3: Ejemplo de ejecución del Test Bench de Captura 1.

6) Enviar por correo electrónico (a tiago.fernandez@udc.es) una memoria explicativa indicando cómo se ha realizado el diseño (entradas, salidas, tamaños de los registros…) acompañada del directorio del proyecto comprimido.