escuela técnica superior de ingeniería y diseño industrial

UNIVERSIDAD POLITÉCNICA DE MADRID

TRABAJO VHDL:
ASCENSOR

Grupo 09

Álvaro Sánchez Barba, N.º matrícula: 56085

Álvaro Sequeira Bárcena, N.º matrícula: 55471

Sofía Toledo Delgado, N.º matrícula: 56120

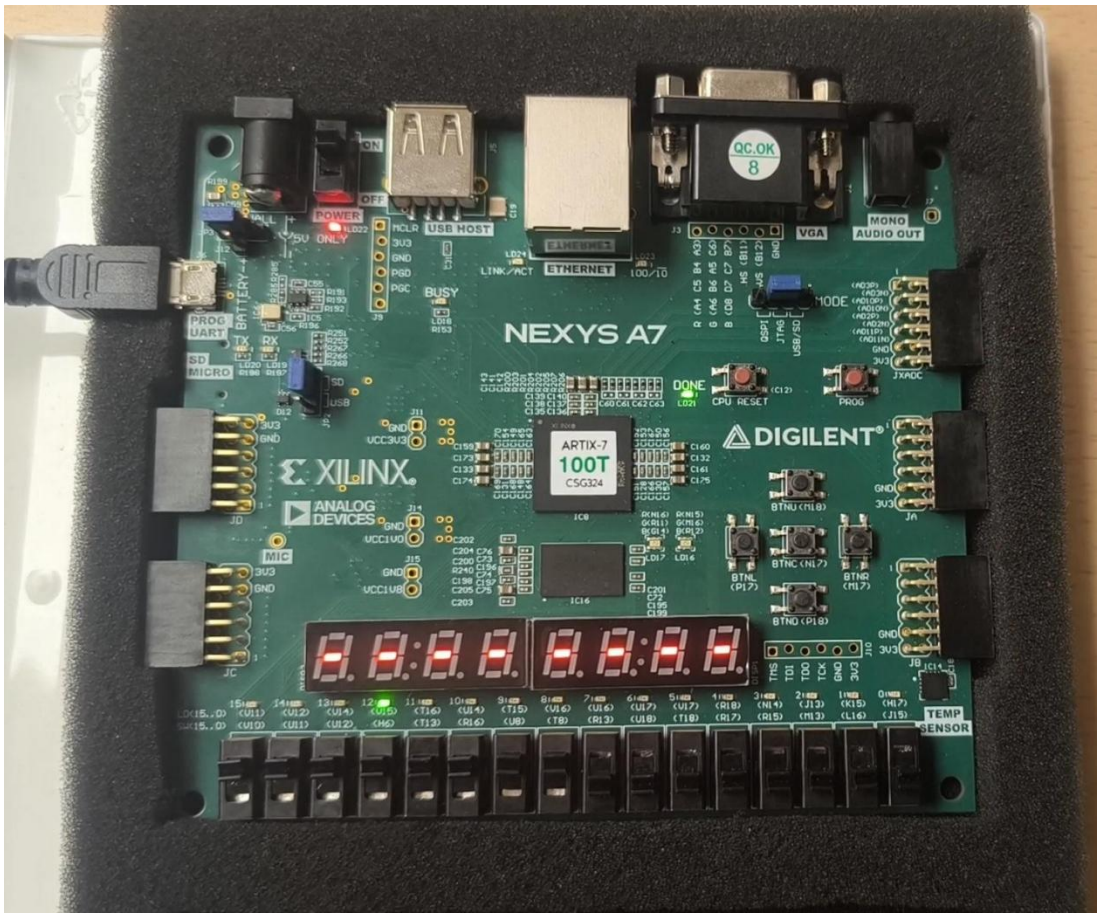
INDICE

INTRODUCCIÓN

En la realización de este trabajo se desarrolla un modelo funcional de un ascensor utilizando VHDL. Para ello, se implementan en Vivado las diferentes funcionalidades del sistema, como la selección de pisos, la gestión de prioridades en las solicitudes y la interacción con los sensores de puertas y posición. Además, se realizarán simulaciones, garantizando que cumpla con el comportamiento del sistema y los requisitos establecidos.

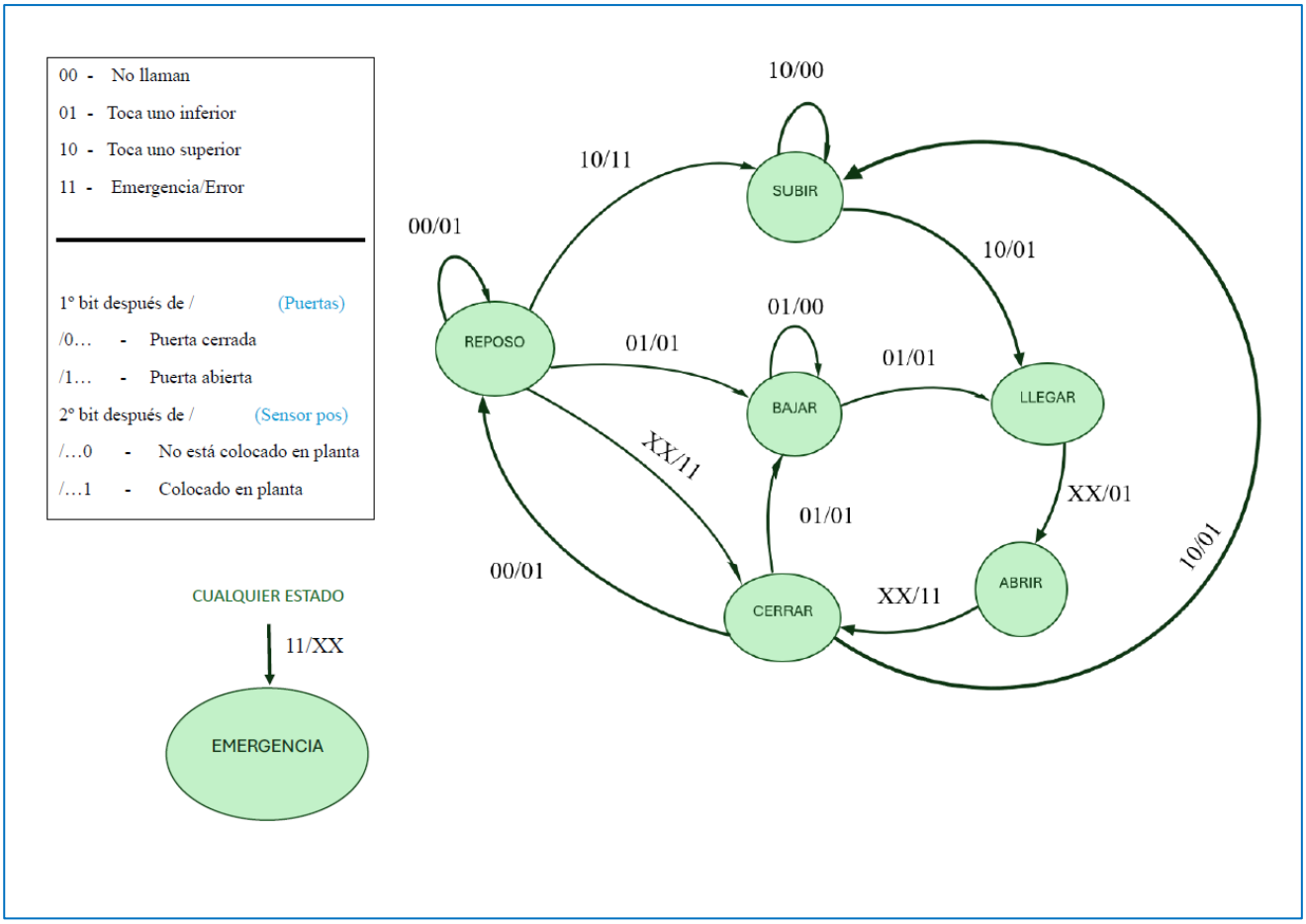
El control de las placas FPGA dadas por el laboratorio mediante la aplicación de Vivado es un aspecto que se practica de manera continua durante el semestre mediante las prácticas de laboratorio.

El objetivo de este trabajo será demostrar el manejo de la herramienta Vivado simulando el comportamiento de un ascensor en la placa Nexys 4-DDR. La placa se comportará de la manera más fiel posible a como lo haría un ascensor. Para demostrar su comportamiento se emplean displays, leds, botones y switches.



ESTRUCTURA

DIAGRAMA DE ESTADOS



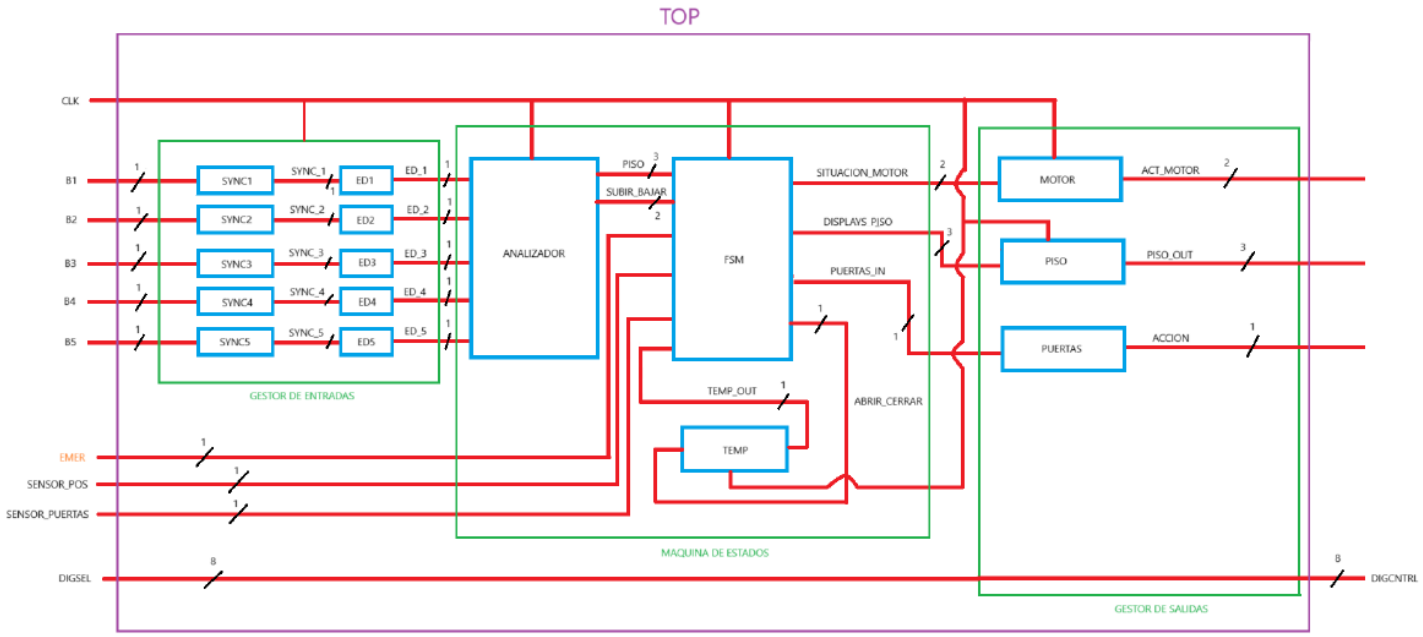
Para simular el comportamiento de un ascensor se emplean los siguientes estados: REPOSO, SUBIR, BAJAR, LLEGAR, ABRIR, CERRAR.

Cuando inicia el programa, el ascensor comienza en REPOSO. Dependiendo de la señal de entrada puede avanzar al estado SUBIR, BAJAR, EMER o quedarse en REPOSO.

En caso de SUBIR o BAJAR, una vez pasa a LLEGAR (que corresponde a llegar al piso indicado), se activa un sensor de posición (que indica que está posicionado en el piso) y se pasa al estado ABRIR. En este estado se abren las puertas, manteniéndose abiertas unos segundos. A continuación, se avanza al estado CERRAR. Desde este estado, depende que lo que se reciba, vuelve a pasar a SUBIR, BAJAR, EMER o, en caso de recibir un “motor stop”, se mantiene en el estado CERRAR.

Independientemente del estado en el que se encuentre, si se activa EMER el ascensor pasa directamente a REPOSO.

DIAGRAMA DE BLOQUES



PROGRAMACIÓN

El funcionamiento del ascensor se organiza en tres grandes bloques: Gestor_entradas, gestor_ascensor, gestor_salidas.

GESTOR_ENTRADAS

En este bloque, como su nombre indica, se gestionan las entradas que corresponden a los botones de los pisos solicitados. Se emplean cinco botones de la placa para representar la llamada de cinco pisos, numerados del 1 al 5. Este bloque se compone de un sincronizador (sync) y un Edge-detector, empleados para detectar flancos y evitar la metaestabilidad, sincronizando señales asíncronas con un reloj.

Las entradas son procesadas asignándoles un valor correspondiente a cada piso, como por ejemplo podría ser: cuando ED_1 =1 (han pulsado el botón del piso 1) se manda a la máquina de estados un 001 para que lo reconozca como “solicitud de llamada del piso 1”. En caso de ser ED_x =0 se manda un “000”, lo que la máquina de estados interpreta como: “no hay ninguna llamada”.

GESTOR_ASCENSOR

En este gran bloque, se reciben las entradas correctamente procesadas y se emiten las "órdenes" de salidas para los bloques del gestor de salidas que se encargan de enviarlos a la placa.

Primero, disponemos de un ANALIZADOR, que se encarga de gestionar todas las entradas que recibe y enviar a la FSM, a qué piso debe ir y si hay que subir o bajar.

Para ello se ha gestionado el código de la siguiente manera: Se ha creado una variable P cuyo valor al inicio del programa es "000". Una vez se gestionan los pisos adquiere el valor del último piso al que ha ido el ascensor. Esta variable es útil para ubicar el ascensor y poder calcular así qué piso se encuentra más próximo y gestionar si el ascensor sube, baja o se mantiene parado.

Todos los pisos recibidos se almacenan en un vector que se recorre continuamente. En caso de sólo haber un piso solicitado lo primero que se hace es compararlo con P para saber si el ascensor sube, baja o se queda en la misma planta. A continuación, se envía esta información junto al piso al gestor de salidas.

En caso de haber varios pisos en el vector, se comparan todos con la variable P, para saber cuál es el más cercano. Una vez encuentra el más cercano detecta si se debe subir o bajar y envía esta información junto al piso al gestor de salidas.

El siguiente bloque es el de la FSM, que es el bloque más importante de todos. Es el encargado de seguir la lógica de procedimiento del ascensor según lo recibido, es decir, se comporta según lo explicado anterioridad en el diagrama de estados.

Por último, otro bloque que compone el gestor_ascensor es el bloque TEMP, que se trata de un temporizador que indica el tiempo que tienen que estar las puertas abiertas cuando llegue al estado ABRIR.

Las salidas emitidas por el bloque FSM van directas al siguiente bloque gestor de salidas, excepto "abrir_cerrar" que es interna y va al propio TEMP.

GESTOR DE SALIDAS

El bloque gestor de salidas se encarga de recibir las salidas del bloque gestor ascensor y procesarlas para adaptarlas a la hora de definir los *constraints* de la placa. Para ello se han creado tres bloques:

MOTOR: Indica la situación en la que se encuentra el motor en cada momento. Para ello hemos utilizado cuatro leds de la placa. Dependiendo de la situación en la que se encuentre el motor se enciende su led correspondiente. Para la situación de emergencia se ha utilizado un led de color rojo.

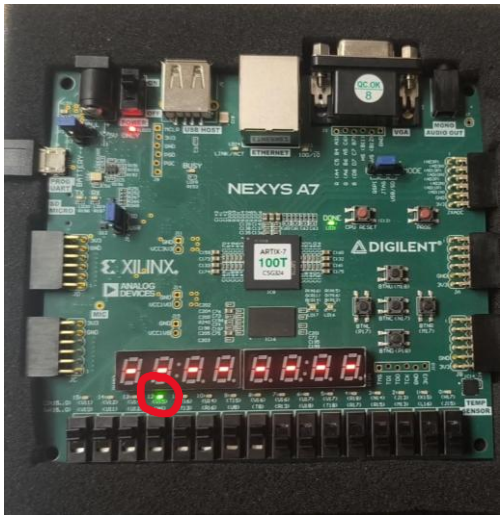


Imagen 1: motor parado

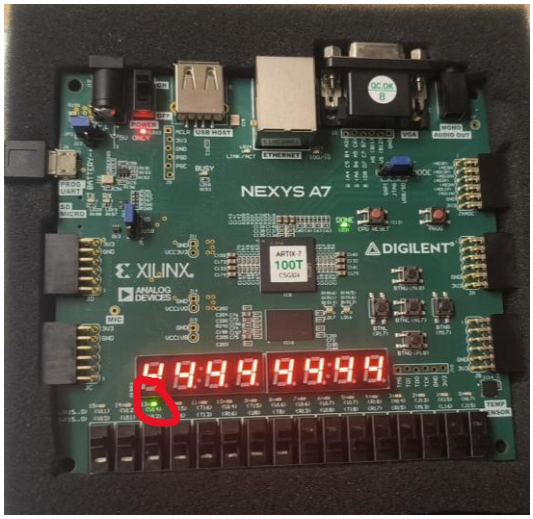


Imagen 2: motor subiendo

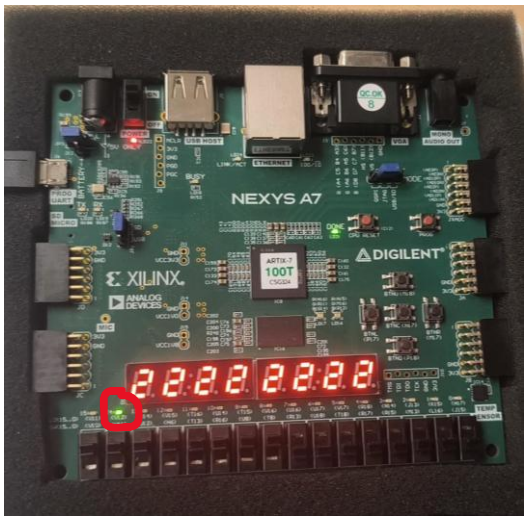


Imagen 3: motor bajando

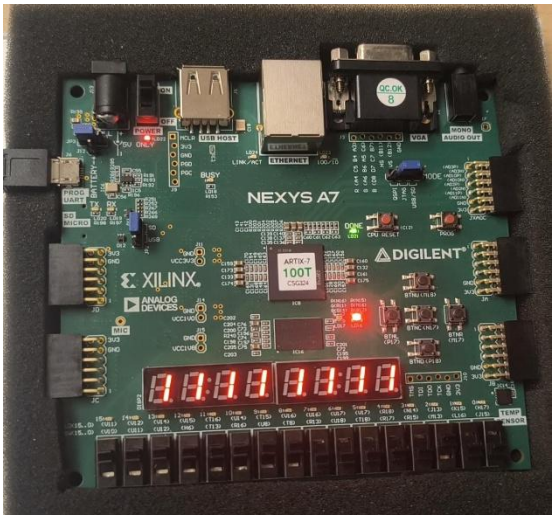
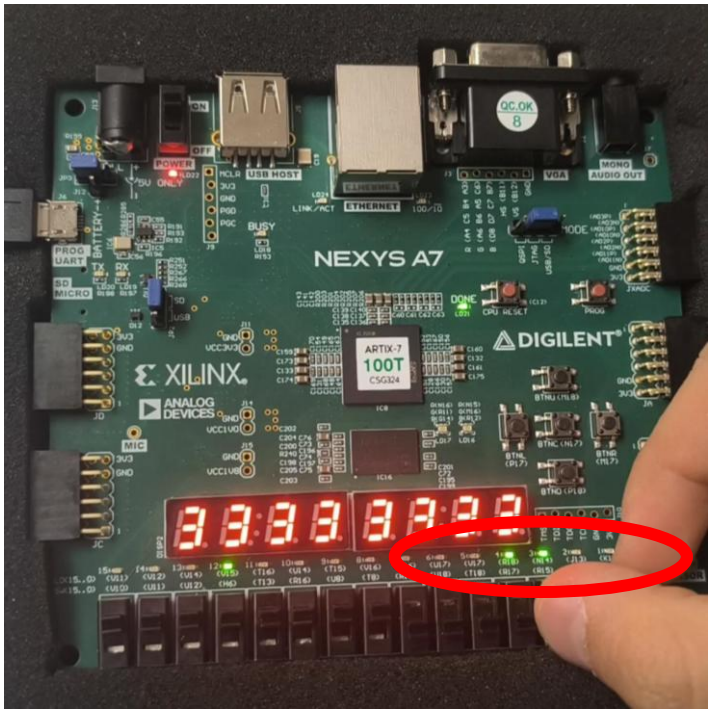
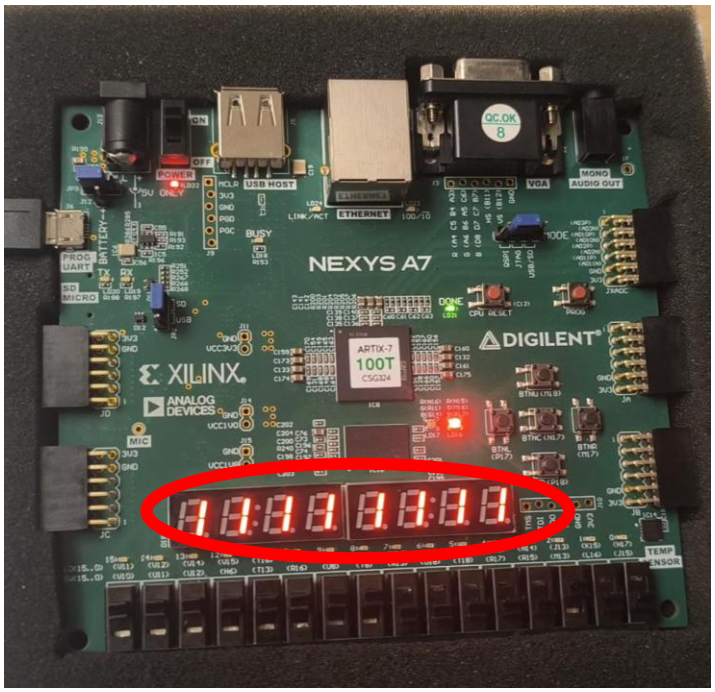


Imagen 4: motor en emergencia

PUERTAS: Recibe si las puertas se abren o están cerradas. En caso de abrirse se emplea un decodificador para convertir el 1 recibido a una secuencia de unos y ceros que apagarán y encenderán seis leds de la placa simulando una apertura de puertas.



PISO: Recibe el número del piso al que se va a desplazar el ascensor y mediante un decodificador se convierte en un vector de siete bits para representarlo en los displays de la placa. Gestiona que los displays muestren los pisos que va alcanzando el ascensor y, una vez haya llegado al piso destino, se muestre ese piso hasta que se solicite una nueva llamada.



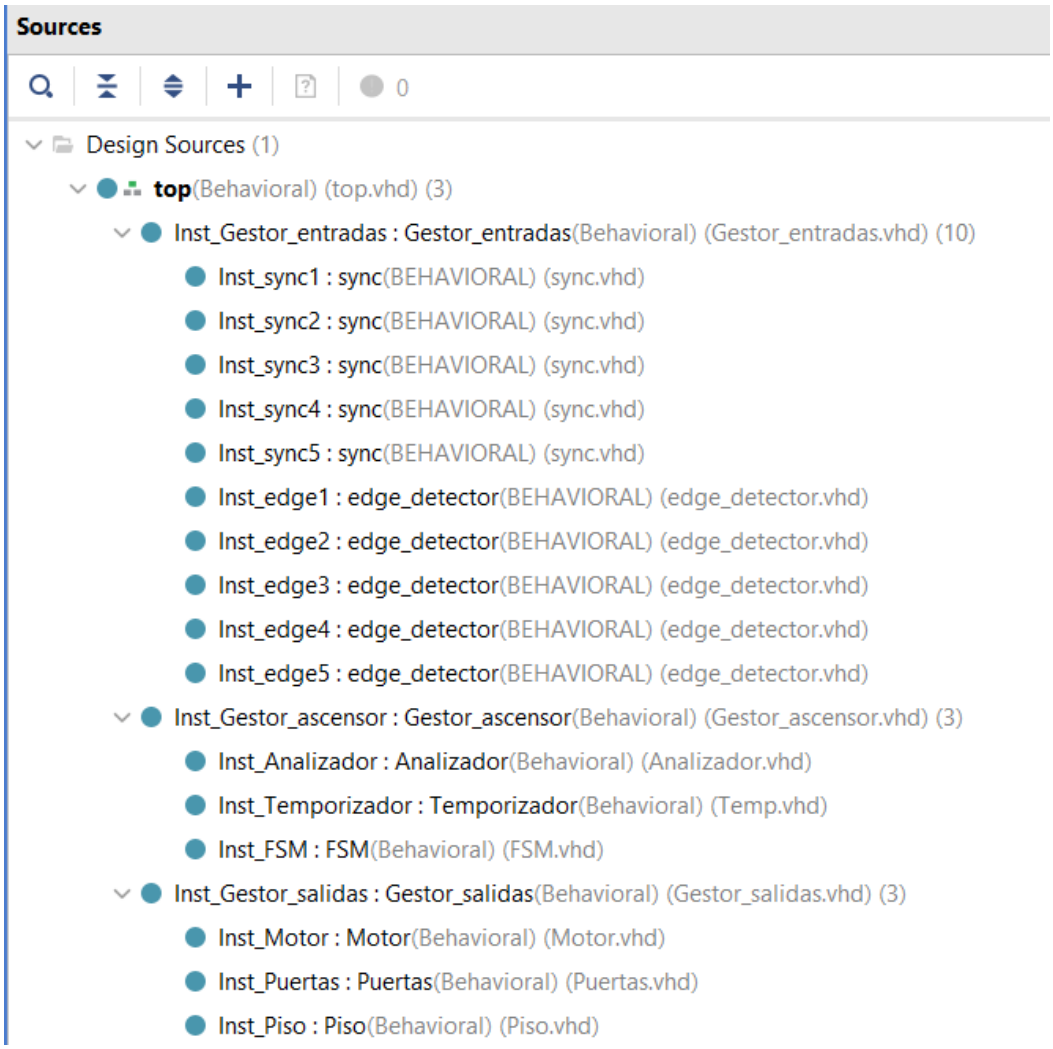
SENSORES

El proyecto cuenta con tres señales externas, un sensor de posición, un sensor de puertas y un sensor de emergencia.

Se han definido como switches, cada uno en su correspondiente en la placa.

- Sensor de posición: Cuando se activa su switch representa que el ascensor se encuentra en la posición deseada. Por ejemplo, si queremos subir del piso 2 al piso 4, el motor se encontrará subiendo hasta que esté activa esta señal, y entonces recibirá que ha llegado y pasará a motor parado.
- Sensor de puertas: Cuando se activa su switch representa que las puertas se pueden abrir porque se dan las condiciones adecuadas.
- Emergencia: Cuando se activa su switch representa que el ascensor ha entrado en situación de emergencia, pasando el motor a situación de emergencia.

SOURCES



En el source *top* se instancian los 3 grandes bloques: gestor de entradas, gestor de salidas y gestor ascensor.

Dentro de cada gran bloque, vienen instanciados los bloques adyacentes a ese bloque:

- En gestor de entradas: *sync* y *Edge detector*.
- En gestor de ascensor: *analizador, fsm* y *temporizador*.
- En gestor de salidas: *motor, piso* y *puertas*.

TESTBENCH

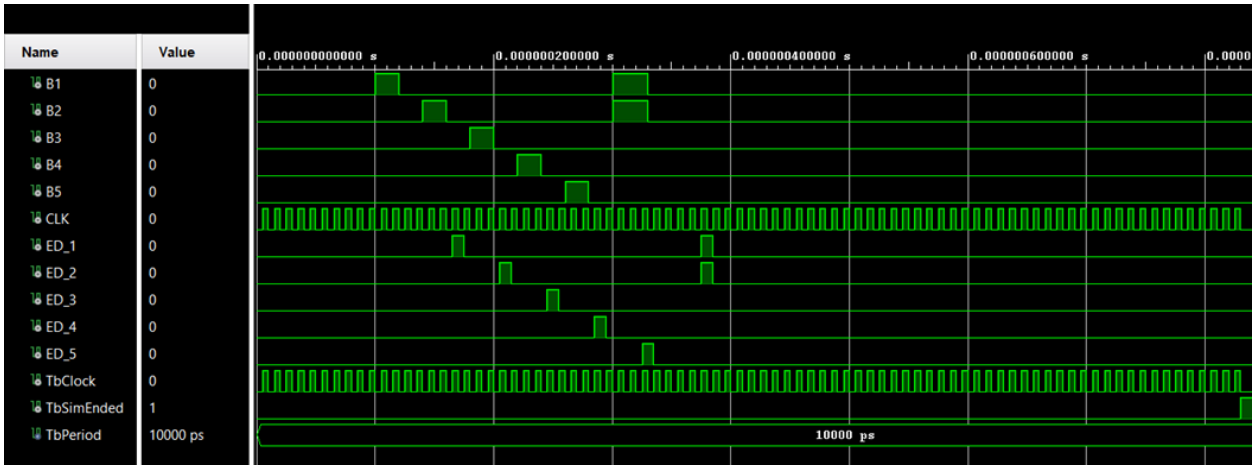
Se han realizado testbench correspondientes a cada bloque para comprobar la lógica de programación conforme realizábamos el trabajo para asegurar el correcto funcionamiento de cada uno y corregir posibles errores de programación.

A raíz de esto se han encontrado varios fallos que no nos permitían observar el comportamiento de los bloques:

- Lista de sensibilidad de la FSM: Al principio programamos una lista de sensibilidad que contenía todas las señales asíncronas como son los sensores de posición y de puertas, etc. Esto impedía ver su funcionamiento y descubrimos que incluyendo únicamente el CLK era suficiente y solucionábamos el problema.

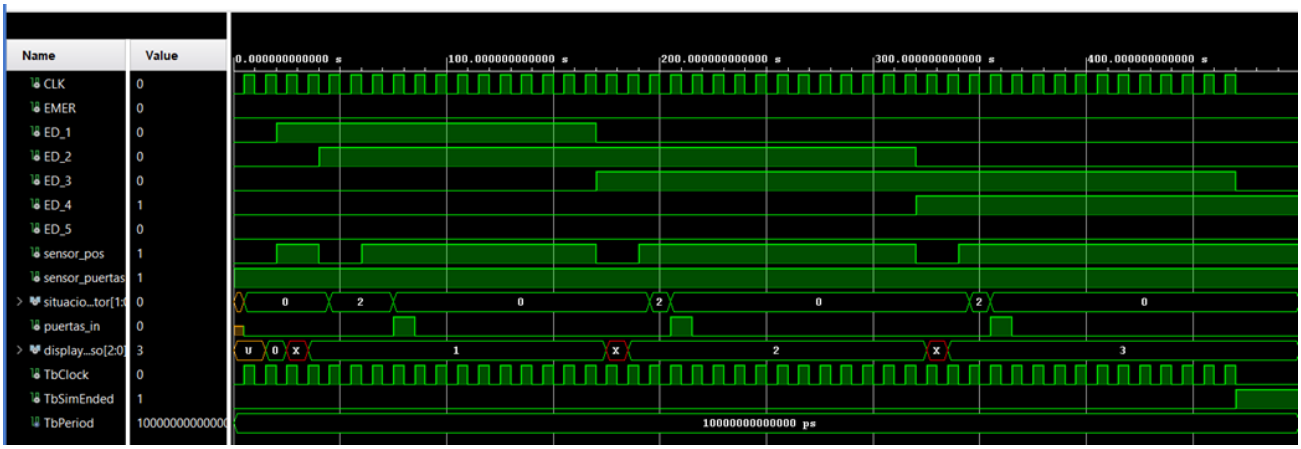
- Edge detector y CLK en la simulación: Tenemos definido el edge detector de manera que cuando recibe un único flanco positivo (100) envía un '1'. A simular con un TbPeriod demasiado pequeño recibía varios flancos positivos, lo que en el testbench se observaba una X y no el verdadero valor de la salida.

Testbench de gestor de entradas:



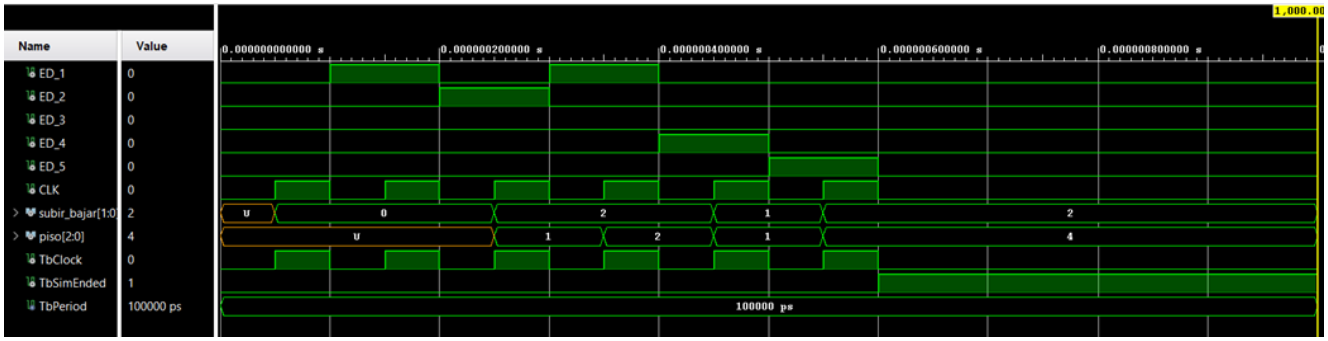
En este testbench podemos ver como al principio activamos los botones y, con un pequeño delay, se activan los Edge detector (ED_X). Lo que nos indica que se procesan de manera correcta.

Testbench de gestor de ascensor:



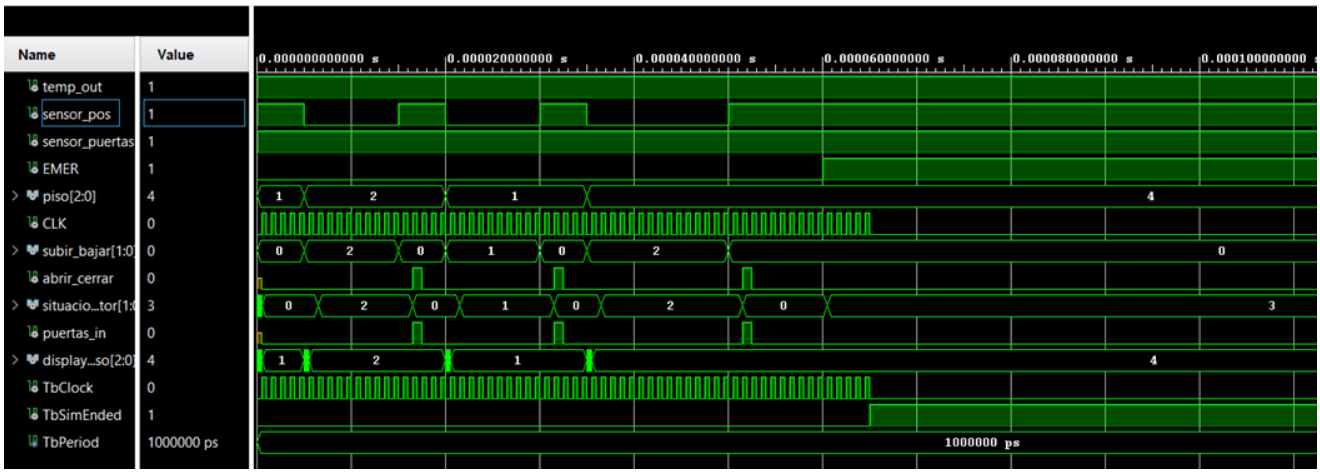
En este testbench la puesta a '1' de cada ED_X representa la llamada de un piso. Observamos el comportamiento de las salidas en función de estas entradas junto a los sensores. Situación_motor inicialmente se muestra con un valor 0 "motor parado" y como se cambia del piso 1 al 2 pasa a valer '2' (10), lo que representa "subir". También se observa el piso en los displays. Al pulsar emer, situación_motor pasa a valer 3 (11), lo que representa "motor en emergencia".

Testbench de analizador:



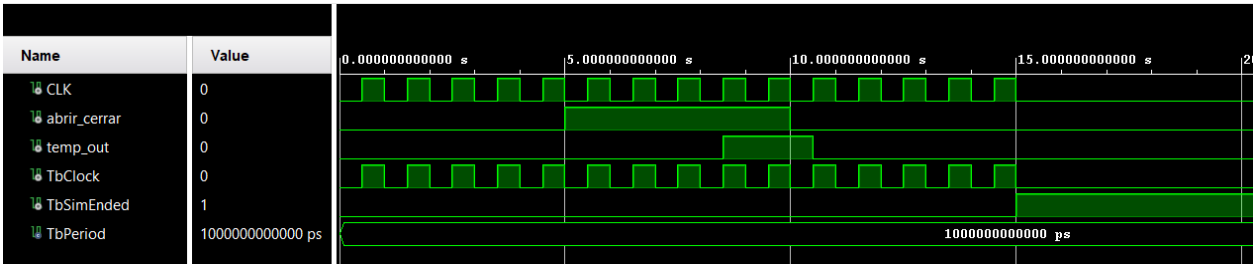
En este testbench, se reciben los Edge detectors (ED_X) como entradas. Según esas entradas, situación motor y piso que son sus salidas, actúan en consecuencia de manera correcta. Las salidas "subir_bajar" (que indica si se debe subir o bajar según el piso del que hayan llamado) y "piso" (indica al piso al que hay que ir) van directamente a FSM y ese bloque las procesa.

Testbench de FSM:



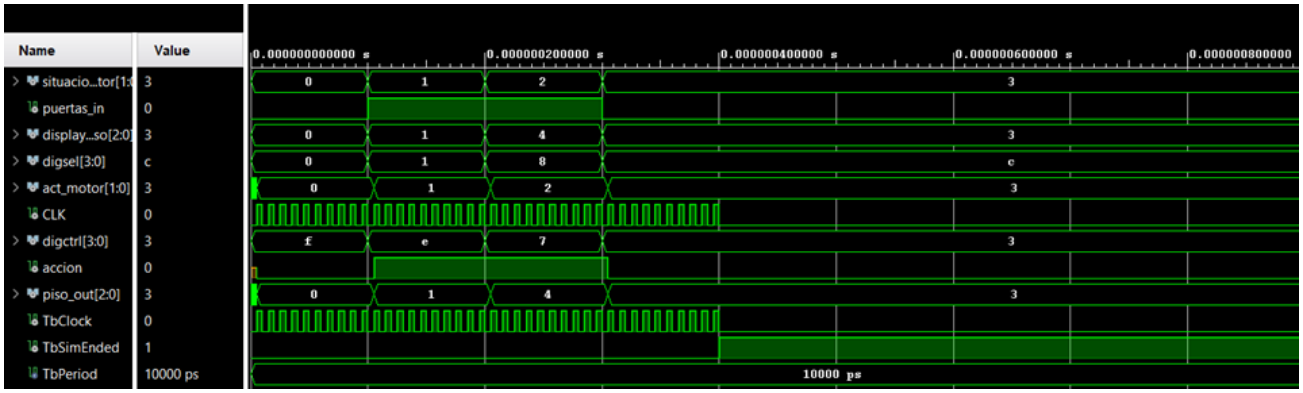
En este testbench corresponde al de la máquina de estados y en él se puede apreciar que según le van llegando los pisos a los que se quiere ir, cambia situación_motor dependiendo del piso en el que estaba, a 2 (10, que significa subir) y a 01 que es bajar, siempre que sensor_pos este a 0 que quiere decir que aún no ha llegado al piso. Cuando está en 1, como ha llegado, el motor se para (pasa a 00), se activa la apertura de puertas (puertas_in a 1) y activa también una señal a nivel alto de abrir_cerrar que será la que active el flanco del temporizador.

Testbench de temporizador:



Este testbench es de un temporizador de 3 segundos utilizado para esperar desde que se abre las puertas. Se utiliza para ello una activación de una variable bandera que es abrir_cerrar que activa el temporizador y, hasta después de 3 segundos, no se activa una variable temp_out para indicar que ha llegado al final de la cuenta y que la valore la fsm para que pueda cambiar la señal de cerrar las puertas.

Testbench de gestor de salidas:



En este testbench de las salidas, podemos ver cómo, en función de situación motor, que llega desde el FSM y de puertas_in, acción, act_motor y displays_piso actúan en consecuencia, mostrando lo que hacen las puertas, mostrando lo que hace el motor y, por último, mostrando en que piso nos encontramos.