|  |
| --- |
| Controlador ascensor 7 pisos |
| Controlador de ascensor con placa NEXUS DDR |
|  |
| Un ascensor o elevador es un sistema de transporte vertical, diseñado para mover personas u objetos entre los diferentes niveles de un edificio o estructura. Está formado por partes mecánicas, eléctricas y electrónicas que funcionan en conjunto para ponerlo en marcha |
| https://tse4-2.mm.bing.net/th?id=OIP.k6ODSbxHFHnSvnJXMo8wdgHaFj&pid=Api |

1. **Objetivo del proyecto**

El presente Proyecto tiene por objeto el diseño del controlador de un ascensor único de una vivienda de 7 pisos capaz de acudir al piso solicitado. El ascensor debe cerrar sus puertas comprobando que no hay ningún objeto en la trayectorias de cierre, trasladarse hasta su destino, abrir puertas y mantenerse en estado de espera hasta que un botón de destino sea pulsado de nuevo.

1. **Descripción**

El diseño del controlador corresponde a un ascensor capaz de desplazarse a través de 7 pisos.

Está formado por 8 módulos programados en VHDL que han sido implementados en una placa de desarrollo NEXUS 4 DDR. Hemos tratado de elaborar un diseño funcional que se asemeje lo máximo a un controlador real en el que se puedan implementar nuevas funciones con facilidad.

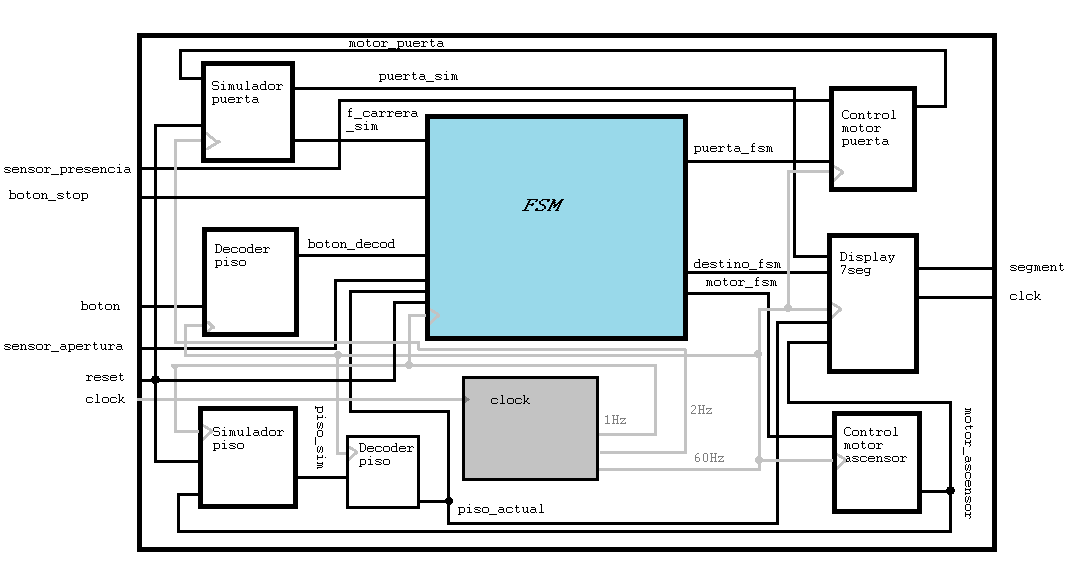
El controlador trabaja como un sistema que se mantiene en periodo de espera hasta que alguno de los botones de destino es pulsado. Cuando alguno de los pulsadores es accionado, el ascensor comprueba si el piso solicitado corresponde con el actual, en cuyo caso no realizará ningún movimiento y permanecerá en estado de espera nuevamente. Si el botón pulsado es distinto al piso actual, se acciona el sistema de cerrado de puertas. Se comprueba que no hay ningún objeto en la trayectoria de las puertas y si no es así comienza a cerrarse. Si durante el proceso de cierre el sensor de presencia detecta algún objeto, las puertas vuelven a abrirse y el ascensor se coloca en estado de espera de nuevo.

Una vez cerradas las puertas el ascensor se desplaza hasta el piso solicitado donde permanecerá cerrado hasta recibir la señal de nivel para asegurar que la puerta no se abrirá hasta estar en la posición correcta. Tras comprobar que se encuentra al nivel se abren las puertas y el ascensor pasa a estado de espera hasta recibir una nueva llamada.

El ascensor consta de un botón de stop que detiene los motores del ascensor sin abrir las puertas siempre que éste esté presionado.

**Diagrama de bloques**

Debido a la necesidad de probar el funcionamiento del controlador sin poder instalarlo en un sistema real decidimos implementar un simulador de puertas que se encargan de proporcionar las señales que deberían obtenerse de los sensores situados en las puertas para comprobar su posición. Y por otro lado un simulador de piso que proporciona la información sobre el piso en el que se encuentra el ascensor en sustitución de los sensores situados en cada planta del edificio.



* 1. FSM

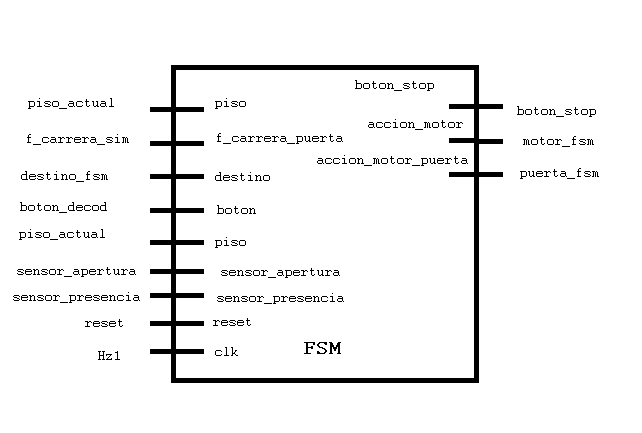
La máquina de estados se encarga de coordinar el funcionamiento del conjunto de módulos a través de las entradas externas y de las señales recibidas del resto de módulos.

A la FSM llegan las siguientes señales:

* **boton\_decod**: señal de los botones de entrada de los 7 pisos codifica por el decoder\_piso
* **sensor\_apertura**: señal que posibilita la apertura de puertas y determina que la puerta del ascensor ha alcanzado la posición correcta del piso solicitado
* **boton\_stop**: señal de stop que se acciona dentro del ascensor y que lo mantiene parado mientras esté pulsado.
* **f\_carrera\_sim**:señal de 2 bits que llega del simulador de puertas con su estado.
* **Piso\_actual**: señal de 2 bits que llega del decodificador de piso con la información de la posición actual del ascensor.
* **reset:** entrada para resetear el módulo.
* **Hz60**: señal de entrada contador proveniente de módulo Clock.

Hacia el exterior de la FSM salen 3 señales:

* **puerta\_fsm**: señal que se dirige al controlador de puerta para indicar apertura, cierre o stop a los motores de las puertas
* **destino\_fsm**: señal que se dirige al display con el piso de destino.
* **motor\_fsm**: señal que se dirige al controlador del motor del ascensor para indicar subida, bajada o stop.



* 1. Simulador\_puerta

El simulador puerta simula el comportamiento de las puertas de una instalación real de un ascensor. La información sobre el estado de las puertas debería llegar a la FSM a través de unos sensores. Como no es así, generamos esa señal a través de la señal **motor\_puerta** que la obtenemos del módulo control\_motor\_puerta que proporciona la información sobre el movimiento de los motores de apertura de la puerta.

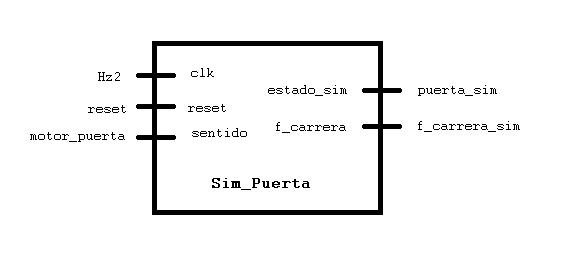
Simulador\_puerta tiene como salidas las señales:

**puerta\_puerta:** señal de 2bits que llega hasta el display para mostrar el cierre de puertas

**f\_carrera\_sim:** señal de 2 bits que llega hasta la FSM como señal de los finales de carrera al abrir y cerrar

**reset:** entrada para resetear el módulo.

**Hz2**: señal de entrada contador proveniente de módulo Clock.



* 1. Simulador piso

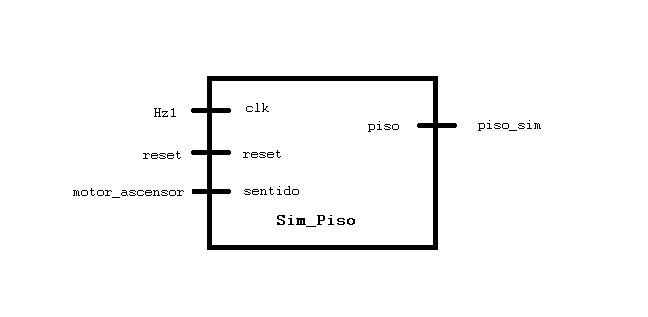
El simulador piso simula el comportamiento del ascensor. Proporciona las señales que deberían adquirirse de los sensores físicos que detectan la posición del ascensor en un sistema real.

La señal **motor\_ascensor** llega al simulador de piso, una señal de 2 bits (subiendo, bajando y stop) con la información sobre el movimiento de los motores de subida y bajada.

Simulador\_puerta tiene como salida **piso\_sim**, una señal de 7 bits, uno por cada piso del edificio que llega hasta el módulo decoder\_piso.

**reset:** entrada para resetear el módulo.

**Hz1**: señal de entrada contador proveniente de módulo Clock.



* 1. Decodificador piso

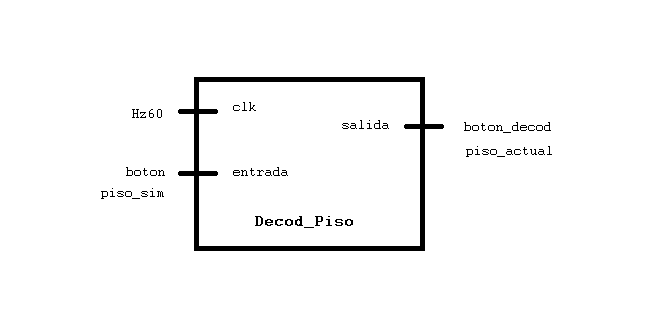
El modulo decodificador piso tiene dos instancias.

Por un lado recibe la entrada **botón** (7 bits, uno por cada piso) con la información del piso de destino. Genera la señal **botón\_decod** con la misma información en 3 bits y la envía a la FSM.

Por otro lado recibe la señal **piso\_sim**, que proviene del simulador de piso (7 bits, uno por cada piso) y genera la señal **piso\_actual**, 3 bits que se envían al display para mostrar el piso actual y a la FSM para que la máquina de estados sepa la posición del ascensor en cada momento.

**reset:** entrada para resetear el módulo.

**Hz60**: señal de entrada contador proveniente de módulo Clock.



* 1. Control motor puerta

El módulo control puerta se encarga de gestionar la activación de los motores de apertura de las puertas del ascensor.

Tiene como entradas las señales:

**Sensor\_presencia**: señal de 1 bit que llega a través del sensor de presencia

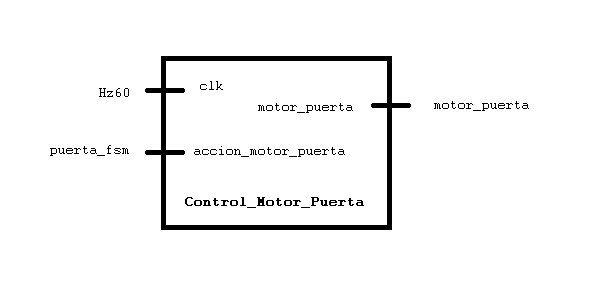
**Puerta\_fsm**:señal de 2 bits con la información sobre la dirección en la que deben moverse las puertas o si deben permanecer quitas.

Tiene como salida la señal:

**Motor\_puerta:** señal que se envía hasta el simulador puerta con la información del estado de los motores para que éste pueda generar su simulación.

**reset:** entrada para resetear el módulo.

**Hz60**: señal de entrada contador proveniente de módulo Clock.



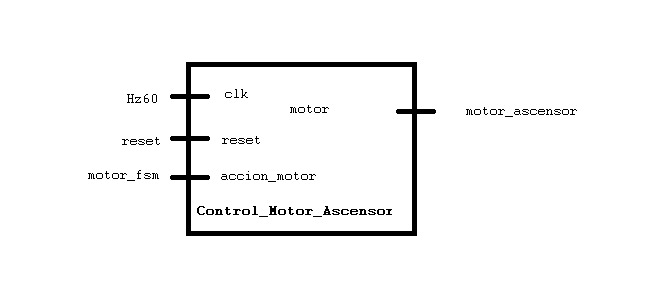
* 1. Control motor ascensor

El módulo control ascensor está pensado para poder añadir funciones con facilidad. En la actualidad se encarga de controlar la subida, bajada y stop del ascensor a través de la señal **motor\_fsm** de 2 bits que proviene de la FSM.

Como salida se genera la señal **motor\_ascensor** de 2 bits con el sentido del movimiento del motor que es enviada al display y al simulador de piso.

**reset:** entrada para resetear el módulo.

**Hz60**: señal de entrada contador proveniente de módulo Clock.



* 1. Clock divider

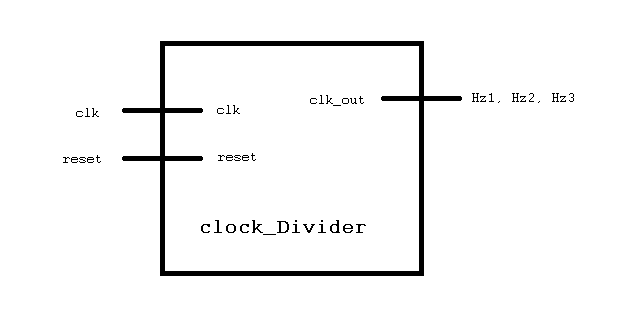
Se trata de un módulo encargado de adapatar la frecuencia de la placa FPGA a la frecuencia que necesitan los distintos módulos.

Recibe como entrada la señal de **clk** proveniente de la FPGA.

Genera las señales:

**Hz1:** señal de reloj que utilizan la FSM, el simulador de puerta y el simulador de piso

**Hz60**: señal de reloj que utilizan el control de motor ascensor, el control de motor de puerta, el display y el decoder piso.



* 1. Display

Se trata de 8 displays de 7 segmentos sincronizados a través de una bandera que activa cada uno de los displays cuando hay un flanco de reloj.

El display recibe 4 señales:

**Puerta\_sim**: señal con que proviene del simulador puerta. 2 bits con la información del estado de las puertas.

**Destino\_fsm**: señal que llega de la FSM con el piso que debe alcanzar el ascensor

**Piso\_actual**: señall que llega del decodificiador de piso con el piso en el que se encuentra el ascensor en cada instante.

**Motor\_ascensor**: señal con la dirección de desplazamiento de los motores del ascensor

**reset:** entrada para resetear el módulo.

**Hz60**: señal de entrada contador proveniente de módulo Clock.

Los displays se distribuyen de la siguiente manera:

Displays 1-4

Simulan la apertura y cierre de puertas mendiante una animación que imita el movimiento de las puertas.

Display 5

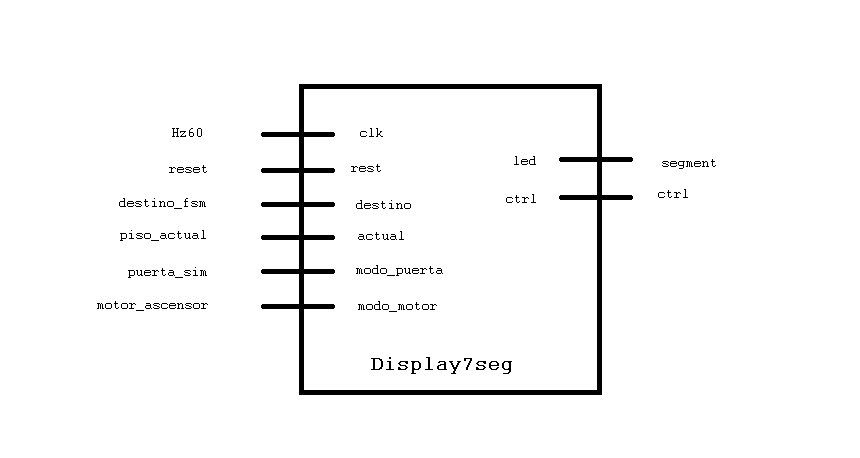
Muestra el piso actual del ascensor

Displays 6-7

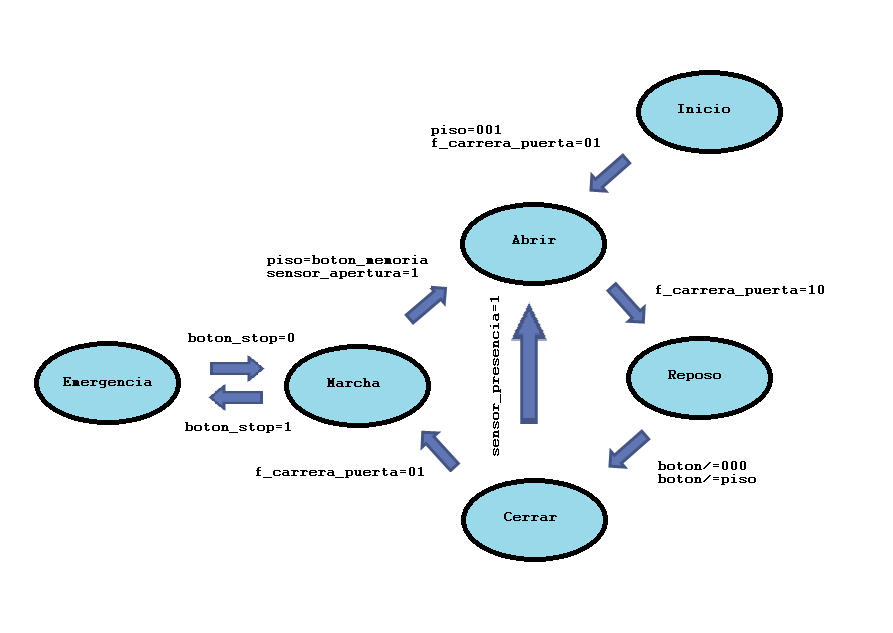
Muestran una flecha de subida o bajada o una línea horizontal en función del estado de los motores.

Display 8

Muestra el piso solicitado hasta el cual debe desplazarse el ascensor



1. **Diagrama de estados**



1. **Conclusión**

Tras ver las características mínimas del trabajo nos propusimos el reto de añadir algunas características extras para obtener un controlador más cercano a un ascensor real. Enfocamos el trabajo como un proyecto con flexibilidad para ser desarrollado en profundidad elaborando una estructura modular bien organizada y definida con un código limpio y fácilmente legible.

Para el desarrollo del código en VHDL hemos utilizado el programa Sublimecomo editor de texto debido a sus numerosas herramientas. Además hemos utilizado Github para facilitar la coordinación entre los miembros del grupo y combinar nuestros avances en el trabajo.

El problema más grande ha sido tener que sincronizar diferentes versiones de vivado en nuestros respectivos PCs con el mismo repositorio a la hora de realizar los testbenchs. Para solucionarlo, hemos migrado nuestro programa a uno nuevo en el que no están los testbench incluidos directamente en el proyecto de Xillinx, pero están agregados a una carpeta externa al proyecto. También nos han surgido dificultades con la implementación de la placa debido al poco tiempo que hemos dispuesto de ella.

Ha resultado un proyecto muy interesante que desarrollar que nos ayudado a entender mejor la programación VHDL así como conocimientos adquiridos en otras asignaturas.