Fecha de entrega 09 de mayo de 2023.

Display para un ascesor en VHDL

BENJAMÍN REBOLLEDO[1], IGNACIO WANG 2[1]
[1]Pontificia Universidad Católica de Chile (e-mail: berebolledo@uc.cl, ignacio.wang@uc.cl)

NO Autorizo que mi proyecto (tal como ha sido entregado, sin nota ni comentarios de evaluación) sea publicado en un repositorio para pueda servir de guía y ser mejorado en proyectos de futuros estudiantes.

Este proyecto ha sido desarrollado bajo el curso IEE2463: Sistemas Electrónicos Programables.

ABSTRACT El proyecto simula el funcionamiento de un ascensor de 4 pisos, representados por los leds de la fpga, al que se le entrega un piso a la vez, al que se va a dirigir desde el piso en que se encuentra, donde la dirección del piso es proporcionada por los switches de la tarjeta ZYBO. Para su correcto funcionamiento se utilizó una máquina de estados, para asignar los pisos y realizar distintas acciones, archivos previamente cargados con los pisos del ascensor comunicados con los protocolos de AXI, todos estos bloques interconectados en un diseño de bloques. Los resultados generados por el programa implementado son acorde al funcionamiento de un ascensor que se le entrega una información a la vez, sin embargo, uno de los desafíos que se tiene en comparación a un ascensor real es la múltiple selección de pisos.

INDEX TERMS VHDL, ascensor, máquina de estados, AXI Lite, AXI Advance.

I. ARQUITECTURA DE HARDWARE (1 PUNTO)

LA estructura del proyecto se puede modelar con el siguiente diagrama de bloques que simplifica la explicación del flujo de datos del proyecto:

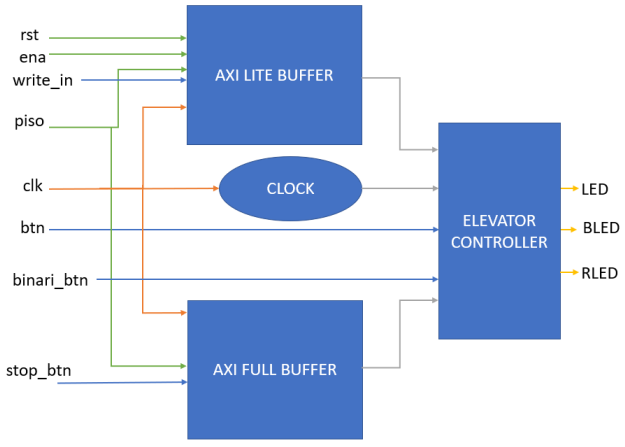

FIGURE 1: Diagrama de bloques de proyecto.

El proyecto tiene como inputs las variables que se presentan a la izquierda, donde las que se conectan a los bloques con líneas verdes corresponden a los datos generados por switches, con líneas azules se generan por botones y con líneas rojas son señales correspondiente al reloj interno de la ZYBO.

Los inputs uusados para el proyecto se describen de la siguiente manera:

- pisos: corresponde a un vector de largo 2, definido por los primeros 2 switches de la tarjeta, por lo que generaría un vector binario, definiendo los pisos del sistema como: piso 0 = "00", piso 1 = "01", piso 2 = "10", piso 4 = "11".
- ena: corresponde a un switch que permite el flujo de información de los switches con el bloque axi lite buffer.
- rst: este switch cuando se encuentra encendido impide que se realicen acciones durante su ejecución en el caso del bloque de axi lite buffer.
- write_in: este es un botón que registra las distintas entradas del sistema, lo que permite manejar los archivos COE del protocolo axi lite. Dichos registros, al apretar write_in, se ejecuta un código descrito en la RAM de axi lite, dicha RAM esta relacionada con una varible tipo array que guarda las entradas para luego ser llamadas.
- btn: este botón permite que los datos guardados en la RAM se usen para
- stop_btn: permite que el ascensor se "detenga" durante el movimiento del mismo.
- binari_btn: al presionar dicho botón, muestra el piso en el que está detenido el ascensor en binario.
- clk: corresponde al reloj interno de la FPGA, dicho reloj se usa en los bloques de comunicación axi y en el divisor de clock que maneja el bloque del controlador.

Los bloques expuestos en el diagrama simplificado tienen las siguientes descripciones:

- AXI LITE BUFFER: corresponde a un conjunto de bloques de IP que se utilizan para manejar los datos de los distintos archivos COE. Como ya se describió previamente, los datos que entran en esta parte corresponden a diversas configuraciones en la botonera de los pisos, lo que se lee en el programa del módulo generado de AXI Lite, lo que realiza la acción de buscar cierto dato en el archivo data.coe, dependiendo de la instrucción, en este archivo data.coe se encuentran los pisos del ascensor. Ese piso se manda al módulo elevator controller. Adicionalmente rst y ena son reguladores de las acciones (permiten o no las acciones) que se realizan en el módulo de axi lite. Interiormente posee un ATG en modo test mode conectado a un IPcore con entrada AXI, a su vez para su correcta implementación el programa generó dos bloques que sirven para manejar las variables ena y rst, con un clk_wizard para el funcionamiento en el tiempo del bloque AXI LITE BUFFER. Para hacer este módulo de bloques nos basamos en el funcionamiento de la ayudantía 4, con respecto a este tema.
- AXI FULL BUFFER: Al igual que en el modulo lite, utilizamos un AXI lite para interpretar archivos coe y enviarlos a un AXI advance. Este saca un dato, el cual procesado por un módulo con entrada de un botón, es capaz de obtener un dato lógico para el ,manejo al interior del ascensor.
- CLOCK: Un divisor de clock, que permite regular el tiempo entre los estados del ascensor.
- ELEVATOR CONTROLLER: Este módulo lo que hace es a traves de una máquina de estados asigar los distintos pisos a los leds, de forma de que cada led representa un piso del ascensor.

Entre los outputs que se tienen sería los siguientes:

- LEDs: representan en que piso se encuentra el ascensor dependiendo de los registros que se le mande.
- BLED: es el color azúl del sexto led de la ZYBO, que indica que está subiendo el ascensor.
- RLED: es el color rojo en el sexto led que representa que el ascensor esta bajando.

Por último, nos gustaría indicar que gran parte de nuestros códigos fueron escritos con ayuda de chatgpt, especialmente a la hora de resolver errores de sintaxis o de funcionamiento.

II. ACTIVIDADES REALIZADAS (1.5 PUNTOS)

Describa el nivel de avance que obtuvo en cada una de las 10 actividades planteadas. Sea breve y guíese por el siguiente formato:

AO1 (66.6%): *Utilice components para integrar al menos tres componentes dentro de otra entity.:* El código cumple con las funciones que habíamos planteado en nuestro proyecto las cuales son: el código principal del ascensor, y un divisor de clock, que permita regular la velocidad a la que se cambien los estados.

AO2 (100%): *Genere al menos 3 diferentes packages en vivado block design para incorporar sus códigos. Utilice parametos genéricos para la configuración de sus packages.* El código cumple con 3 packages generados por nosotros. De acuerdo a la arquitectura presentada en la sección I, esto se implementó en los bloques elevator_controler, que funciona como el ascensor por medio de máquina de estados y el clok_div, que funciona como divisor de clock para el programa. *Nivel de Logro:* 66.6% Parcialmente logrado, el código implementado ha compilado, se logra simular luego de la implementación, se logra generar bitstram y cargarlo en la ZYBOZ7, sin emargo, no posee la cnatidad total de IPCores requeridos.

AO3 (50%): *AO3: Utilice comunicación AXI para comunicar: i) al menos un IP core esclavo creado por ud con un ATG maestro en Test Mode y ii) al menos un IP-core esclavo creado por ud. con un ATG maestro en Advance Mode.* Pudimos implementar el IP core esclavo con un ATG maestro en Test mode, sin embargo,no pudimos implementar en advance mode.

AC1 (100%): *Genere una máquina de estados que esté asociada a parte o a el total de su proyecto.* El ascensor funciona con una máquina de estados de 4 estados en total.

AC2 (100%): *Utilice los 4 botones, 4 swiches y 4 leds de la ZYBOZ7 vistos en el curso* Se utilizaron 3 de los 4 botones, la implementación de los switches fue completa, al igual que los leds.

AC3 (0%): *Utilice al menos 5 operadores y 5 atributos diferentes..*

AC4 (0%): *Utilice variables para actualizar valores instantaneamente en alguna parte de la lógica programada.*

AC5 (100%): *Utilice código secuencial y código concurrente y evidencie claramente la diferencia de funcionamiento de ambos tipos de códigos.* Como está descrita la máquina de estados se utilizao códigos seuneciales y concurrentes en el trabajo realizado.

AC6 (0%): *Haga uso de functions y procedures, donde el uso de ambas rutinas explicite sus diferencias principales* No se pudo implementar el uso de functions o procerdures.

AC7: (0%) *Implemente alguna funci on que no se haya presentado en el curso. Esto puede ser activar algún hardware integrado en la ZYBOZ7, implementar la interconexi on entre el PL y alg un perif erico elemento del zynq (memorias, EMIO, DMA, snoop control, etc), utilizar una variante de AXI no visto en clases, etc.* No se pudo implementar una función no vista en clases.

III. RESULTADOS DE SIMULACIÓN (3 PUNTOS)

Lamentablemente no pudimos realizar las simulaciones.

IV. RESULTADOS IMPLEMENTACIÓN (0.1 PUNTOS)

En cuanto a la implementación del programa se pudo realizar en la tarjeta ZYBO10, pudo mostrar la subida y bajada de los distintos pisos en la que se les asignaba por medio de AXI lite, se muestran los distintos colores de los leds al

subir y bajar. Uno de los problemas de la implementaión del programa es al darle más de un piso.

V. CONCLUSIONES(0.2)

- Algo que se pudo apreciar en la implementación es el hecho de que los datos tenían un delay muy grande al momento de dar más de una dirección.
- En cuanto a la recepción de datos de parte de los distintos botones que se tenían en acción se pudo observar un delay en el caso de setear más de uno, esto debido a que la máquina de estado priorizaba las acciones que estaban anteriormente.

VI. TRABAJOS FUTUROS (0.2)

Uno de los desafios más grandes que tenemos que resolver en la implementación es el uso de más de una dirección mandada a los protocolos y su implementación en el elevador.

Otro posible trabajo a futuro, es la implementación de una botonera externa para los pisos del ascensor, al igual que la implementación de botoneras para cerrar y abrir puertas y una señal de emergencia.