Fecha de entrega 09 de Mayo de 2023.

Disclaimer This latex template has been obtained from IEEE ACCESS for educational purpose.

Led topo

IGNACIO BARRIENTOS[1], RODRIGO ZALDIVAR[1]
[1]Pontificia Universidad Católica de Chile (ignacio.barrientos@uc.cl, rodrijzs@uc.cl)

SI 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 tal y como dice su nombre simula el juego clásico de golpear a los topos con el martillo cuando estos se asoman, en este caso se encenderán los leds un tiempo determinado y en ese tiempo debes presionar su botón correspondiente, luego de apretar el botón el led rgb tomara diferentes colores: Verde si le haz dado al correcto, rojo si es erróneo y azul cuando aun no se haya golpeado o no se golpeó. Esto se implemento principalmente con el uso de lógicas secuenciales para distinguir las situaciones mediante condiciones que pueden cumplirse o no, estos posibles casos se dan mediante el cambio de leds con el tiempo y la pulsación de los botones logrando visualizar los resultados con un led rgb.
La implementación de este juego en la tarjeta Zybo Z7 fue exitosa logrando jugar con distintas secuencias de leds que pueden cambiarse con el uso de switches.

INDEX TERMS VHDL, memoria RAM, AXI, código secuencial, concurrente, Zybo Z7-10, programación en hardware, Vivado, Vitis, Xilinx.

I. ARQUITECTURA DE HARDWARE (1 PUNTO)

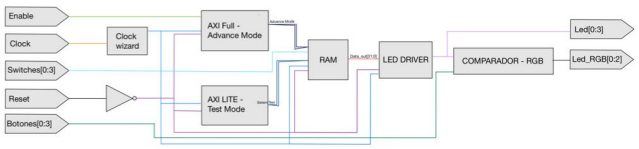

FIGURE 1: Diagrama de bloques.

LA estructura del proyecto corresponde como puede observarse en el diagrama de bloques a 4 etapas principales:

1) Lectura de RAM y comunicación AXI.
2) recepción de datos con temporizador (LED DRIVER).
3) Comparador del valor del led con el ingresado por botón.
4) Mostrar resultados mediante led RGB.

Detallando mas el funcionamiento general tenemos lo siguiente: Lectura de memoria RAM mediante comunicación AXI la memoria RAM dispone de n secuencias o filas de 32 bits estas secuencias pueden ser cambiadas con los switches de manera que al momento de jugar no sea siempre el mismo patrón, luego de esta forma entra al bloque que corresponde a un "Led Driver" y un temporizador al mismo tiempo un "STD LOGIC VECTOR" de 32 bits (secuencia escogida) que contendrá la información a leer, en total de ese vector se tomaran 8 datos de 4 bits, los cuales corresponden a los leds que se deben encender en la secuencia. El temporizador por su parte contara 2.5 segundos lo que corresponde a 312.5 millones de flancos de clock, una vez que termine de contar se pasara al siguiente dato de 4 bits y cambiara el led que se encienda reiniciándose también el contador. Una vez encendidos los leds se envía la señal al bloque rgb, este recibirá la señal enviada por los botones y el valor del led mencionado anteriormente, este dispone de tres estados (Azul, Verde y Rojo) el estado Azul corresponde a que el botón aun no ha sido presionado, el estado Verde corresponde a que el botón ha sido presionado y es el correspondiente al valor del led encendido y por ultimo el estado Rojo corresponde a cuando el botón es presionado y el resultado es incorrecto.

II. ACTIVIDADES REALIZADAS (1.5 PUNTOS)

A01: (0%) No se han utilizado components.
AO2 (100%): El proyecto fue diseñado en base a ciertos bloques los cuales fueron implementados como packages, los utilizados fueron el de comunicación AXI, el led driver que incluye temporizador y el bloque rgb que funciona como comparador y enseña los resultados, estos funcionaron correctamente, cumpliendo las funciones esperadas.
AO3: (100%) El Axi traffic Genetator ATG en modo AXI Lite en Test Mode, se utilizó para cargar 4 vectores de 32 bits, en una memoria ram que se comunicaba con este mismo

protocolo, luego esta ram se conectó con un led driver, que cada 2.5 segundos seleccionaba 4 bits diferentes, donde solo un bit tenia valor 1, lo cual al implementarlo en la tarjeta Zybo, se pudo comprobar que se desplegaba la información correcta. Por otro lado, AXI4 Advance mode, se utilizó como buffer del enable, y verificación de la carga de la información en una ram anexada a esta, para activar la primera ram mencionada. Esto se comprobó con el correcto uso del enable de la ram al mostar los led en la tarjeta zybo.
AC1: (0%) No se implemento una maquina de estados.

AC2: (100%) Los 4 botones y 4 leds son implementados enseñando el led que se debe comparar con el valor del botón presionado, los switches por otro lado corresponden al enable que se utiliza en el AXI, el reset y los últimos 2 corresponden a secuencias de números que serán ingresadas para encender los leds funcionando esto a la perfección.
AC3:(60%) Se utilizan operadores los cuales se utilizan en condiciones IF/ELSIF a lo largo del código, los utilizados son '=, /=, <, >, AND y NOT. En el caso de los atributos solo se ha utilizado el 'EVENT para reconocer flancos de clock.

AC4:(100%) En la implementación del AXI-full se utilizo una variable con la finalidad de que se actualice instantáneamente, esta logro su objetivo.
AC5:(100%) Se ha utilizado código secuencial en diversas ocasiones con el objetivo de actualizar señales cuando existen cambios en una lista de sensibilidad especifica de tal modo que la señal solo se actualice cuando sea de nuestro interés, el código concurrente se utilizo en la comunicación de AXI-LITE con la finalidad de que las señales se actualicen apenas se cumplan ciertas condiciones. De esta forma se logran cambiar la posición de los leds y activar las diferentes etapas del led rgb.
AC6: (0%) No se han utilizado functions ni procedures.
AC7:(100%) Se ha añadido el uso del led RGB, lo cual no ha sido enseñado hasta ahora en el curso, este ha funcionado correctamente para indicar si se ha apretado un botón, si es correcto e incorrecto.

El puntaje asociado a la descripción de las actividades desarrolladas se entrega en la siguiente tabla.

- (2puntos) El alumno logra mostrar el correcto funcionamiento de la ZYBO para las 3 actividades obligatorias y las 7 actividades complementarias.
- (1puntos) El alumno logra mostrar el correcto funcionamiento de la ZYBO para para las 3 actividades obligatorias y las 4-6 actividades complementarias.
- (2 punto) El alumno logra mostrar el correcto funcionamiento de la ZYBO para para las 3 actividades obligatorias y 1-3 actividades complementarias.
- (1 punto) El alumno logra mostrar el correcto funcionamiento de la ZYBO para para al menos 2 de las actividades obligatorias.

III. RESULTADOS DE SIMULACIÓN (3 PUNTOS)

Como se puede ver en la figura 1 y 2, las simulaciones son parecidas, sin embargo presentan diferencias mínimas que

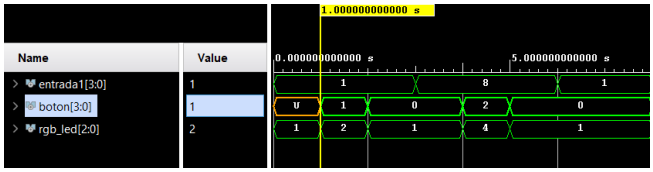

FIGURE 2: Simulación ideal.

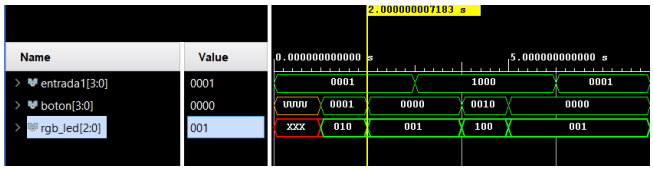

FIGURE 3: Simulación Post-Implementación.

pueden hacer cambios drásticos en el momento de la implementación aunque no es el caso, viendo la simulación post-implementación se podría pensar que en el primer segundo el led rgb no estaría encendido en azul como es debido pero al implementarlo si funciona, al comparar ambas simulaciones viendo los casos limite donde ocurren cambios en la entrada botón veremos que el led rgb en la simulación ideal cambiara al instante en cambio la simulación post-implementación tiene un delay del orden del orden de los nano segundos, lo que en este caso no afecta pero en otros diseños podría causar que una señal no se actualice causando un error al implementarlo.

IV. RESULTADOS IMPLEMENTACIÓN (0.1 PUNTOS)

Al momento de implementar en un comienzo se dieron problemas al momento de unir los packages, ya que se probo anteriormente cada uno por si solo, sin embargo al momento de implementar el sistema completo existieron diversos problemas de sincronización que se pudieron haber mejorado con el uso de una maquina de estados o si se hubieran diseñado todos los bloques en torno al clock de la Zybo z7.
Fue posible darse cuenta también implementando los bloques que se suelen generar problemas cuando se realiza un proceso el cual depende de valores anteriores y se debe ser sumamente minucioso al momento de pensar en todos los casos que pueden ocurrir cuando existen cambios en las señales de la lista de sensibilidad, esto provoco diversos problemas al momento de ejecutar condiciones tipo IF/ELSIF/ELSE, generando que en algunos casos se ingrese a la condición errónea o a varias condiciones a la vez sin desearlo, estos problemas pudieron ser solucionados parcialmente, ya que en muchos casos fue necesario cambiar completamente la estructura del código utilizando otros métodos como "case" o simplemente reescribiendo el código de otra manera.
De igual manera la implementación final cumple su función de manera correcta cumpliendo los objetivos que tenia el juego logrando simularlo de buena forma.

V. CONCLUSIONES(0.2)

- Los resultados de la comunicación AXI mediante AXI-LITE y FULL fueron satisfactorios, logrando comunicar la RAM con los bloques posteriores, estos bloques (LED DRIVER y RGB) lograron funcionar tal y como se esperaba realizando los cambios en los leds en los momentos precisos a simple vista, sin perjudicar la jugabilidad de este proyecto, a nivel de simulaciones existen delays realmente pequeños al momento de cambiar los leds del orden de los nano segundos lo que no afecta en nada al jugador, ya que la ventana del tiempo es de 2.5 segundos por led encendido y si tenemos en cuenta que el delay aproximado es de unos 7 nano segundos según las mediciones, es posible notar que este retardo ni siquiera es un ciclo del clock maestro de la Zybo el cual tiene un periodo de 8 nano segundos de todas formas estos delays pueden clasificarse como naturales ya que las implementaciones ocurren en circuitos físicos de la Zybo por lo que es posible obviar que no son errores de código a menos que este delay sea mayor o los resultados de implementación no sean los esperados.

VI. TRABAJOS FUTUROS (0.2)

Para trabajos futuros este proyecto podría mejorarse añadiendo que al momento de pulsar el botón y se compare los leds cambien una vez se suelte el botón de manera que en un tiempo determinado sea capaz de generar cierto puntaje pudiendo acertar en innumerables veces en este tiempo y descontando puntaje al equivocarse, luego el puntaje calculado como $(aciertos - errores)$ podría ser guardado en una memoria RAM.

Además se podría implementar un método de generación de leds aleatoria de manera automática, dado que en este caso la lectura de la ram tiene secuencias de 32 bits que están prehechas por lo que realmente no son aleatorias.

REFERENCES

[1] IEEE Standard VHDL Language Reference Manual. (2000). Uchicago.edu. https://edg.uchicago.edu/ tang/VHDLref.pdf

[2] Rojas, F. (2023). LAB04: AXI_Traffic_Generation. Documento interno. Sistemas Electrónicos Programables, Sección 1, Pontificia Universidad Católica de Chile. Archivo no publicado.