Fecha de entrega 9 de mayo de 2023.

Implementación del clásico juego Simón Dice en Zybo Z7-10

MARTÍN ÁLVAREZ[1], FRANCISCO HERNÁNDEZ[1]
[1]Pontificia Universidad Católica de Chile (e-mail: martin.alvarez@uc.cl, francisco.hernandez@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 consiste en una implementación del clásico juego simón dice, donde el usuario leerá una secuencia de patrones de leds encendidos en los cuatro leds de la zybo. Por cada nivel que avance el usuario, la secuencia aumentará en 1 valor, hasta llegar a un nivel máximo correspondiente a un patrón de 15 valores de leds. Mediante el diseño de un sistema de comunicación AXI, se implementa un nivel de dificultad para el usuario. El nivel de dificultad, el cuál lo podrá manejar el usuario en cualquier punto del juego, indica el máximo que puede alcanzar la cadena de bits. Para los archivos .coe con los cuales se ha diseñado e implementado el proyecto, los máximos posibles de secuencias son, 1, 3, 7 y 15.

INDEX TERMS Simon Dice, VHDL, maquina de estados, Comunicación AXI, etc.

I. ARQUITECTURA DE HARDWARE

EL primer módulo con el cuál nos encontramos al programar la Zybo es el módulo de Inicio. Este módulo se compone de una máquina de tres estados: incio, carga y juego. Esto nos permite mantenernos en el estado de carga para poder seleccionar el largo total de la secuencia de leds que vamos a tener que ingresar en el último nivel del juego. mediante el accionar de los switches 0 y 1 de nuestra Zybo. Según la selección de switches que realicemos, esta nos indicará el slot de la memoria RAM_PUNTAJES que queremos estamos leyendo, entregándole esta información al módulo TOP. Una vez que hayamos cargado la información necesaria, podemos subir el switch 3 modificando el estado de inicio a carga para luego pasar al estado juego y otorgar el valor de la señal estado a 1, permitiendo comenzar con el juego de Simón Dice.

Además de comenzar con el módulo Inicio, también están funcionando los módulos de RAM PUNTAJES, cuyo comportamiento fue descrito anteriormente, y RAM ADVANCE, utilizando comunicación AXI LITE y AXI FULL respectivamente. El módulo RAM ADVANCE tiene la señal salida la cual toma el valor de uno en caso de que el LSB de la data que esté leyendo desde el ATG en modo advance sea 1, habilitando de esta manera la escritura de los datos del ATG en modo Test Mode en RAM PUNTAJES, información que, como se comentó anteriormente, nos permite definir el largo total de la secuencia de nuestro juego. Es importante destacar que para realizar la comuniación AXI lite, se implementó un ATG en test mode con AXI Lite en conjunto con la RAM PUNTAJES, mientras que para la comunicación AXI Full se utilizó un driver ATG con conexión AXI Lite conectado mediante un módulo AXI Smart Connect a una DUT ATG Advance mode la cual a su vez se conectó con el periférico en AXI Full RAM ADVANCE.

Luego pasamos al bloque principal de nuestro proyecto que se centra en el módulo TOP, el cual contempla toda la lógica programable correspondiente a nuestro modo de juego de "Simón dice". El funcionamiento de este módulo se encuentra regido por una máquina de estados de 8 estados las cuales nos permiten el correcto desarrollo a lo largo del juego, destacando los estados de generación de la secuencia de leds a mostrar, recepción del input del jugador a través de los botones, comparando este resultado con la secuencia generada, entre otros estados de nuestro juego. El módulo TOP utiliza e instancia en su interior tres distintos componentes: debounce, single_pulse_detector y rand_gen. Este último componente se encarga de generar de manera pseudo aleatoria la secuencia de leds que se irán encendiendo a medida que vayamos avanzando los niveles mientras que los primeros dos componentes se encargan de captar cuando cualquiera de los 4 botones es accionado de manera estable para luego entregar una señal de salida como respuesta del jugador. Por lo tanto, al iniciar el juego se nos muestra tan solo un led. Luego, el jugador deberá tocar el botón

correspondiente a ese led para pasar al segundo nivel. Una vez pasado al segundo nivel, la zybo procede a mostrar dos leds (ya que estamos en el nivel 2) y así sucesivamente hasta llegar a una secuencia de 15 leds encendidos en orden pseudo aleatorio. Destacar que el módulo TOP está fuertemente influenciado en una cátedra realizada por la california state university del curso ECE 524 [1].

Finalmente, contamos con otros dos módulos auxiliares en nuestra implementación: un módulo NOT, un Processor System Reset y un módulo multiplexor. El módulo NOT nos permite tomar la entrada del rst, proveniente del switch 2 de la zybo, para luego negar esta señal y asignarla a los módulos que requieren una señal de rst negada el cuál toma las entradas de los leds del módulo de inicio y el módulo TOP y, según sea el valor de la señal estado proveniente del módulo inicio, asigna el valor de salida a los LEDs de la Zybo.

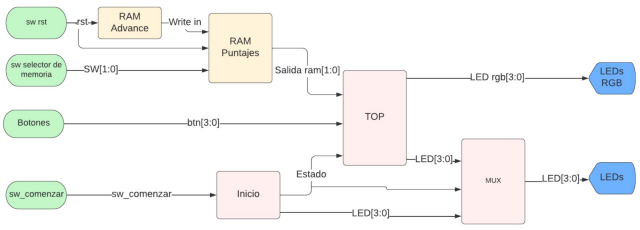

FIGURE 1: Diagrama de nuestro proyecto

II. ACTIVIDADES REALIZADAS

AO1 (100%): *Descripción:* El código cumple con las funciones que habíamos planteado en nuestro proyecto las cuale son: "Utilice components para integrar al menos tres componentes dentro de otra entity".

De acuerdo a la arquitectura presentada en la sección I, esto se implementó en el bloque TOP, implementando los componentes debounce, single_pulse_detector y rand_gen cuya finalidad se encuentra descrita tanto en el código como en la sección I. *Nivel de Logro:* 100%. Completamente logrado, el código implementado ha compilado, se logra simular luego de la implementación, se logra generar bitstram y cargarlo en la ZYBOZ7.

AO2 (100%): El código cumple con las funciones que habíamos planteado en nuestro proyecto las cuales son: "Genere al menos 3 diferentes packages en vivado block design para incorporar sus códigos. Utilice parámetros genéricos para la configuración de sus packages". Esto se implementó dentro del block desing axi_ram en donde se incluyen los 5 packages implementados en el proyecto los cuales son: TOP, Inicio, Mux, Ram puntajes y Ram Advance, cuyas finalidades se encuentran descritas en la sección anterior.*Nivel de Logro:* 100%. Completamente logrado, el código implementado ha compilado, se logra simular luego de la implementación, se logra generar bitstram y cargarlo en la ZYBOZ7.

AO3: El código cumple con las funciones que habíamos planteado en nuestro proyecto las cuales son: "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."

Esto se implementó dentro del block desing axi_ram en donde podemos encontrar la RAM Puntajes, la cual se comunica a través de AXI Lite con un ATG en Test Mode, mientras que por otro lado podemos encontrar la estructura Driver-DUT-RAM la cual se compone de un ATG test mode, un ATG en Advance mode y una ram AXI Full, que recibe las intrucciones del DUT y cuyas finalidades se encuentran descritas en la sección anterior.*Nivel de Logro:* 100%. Completamente logrado, el código implementado ha compilado, se logra simular luego de la implementación, se logra generar bitstram y cargarlo en la ZYBOZ7.

AC1: El código cumple con las funciones que habíamos planteado en nuestro proyecto las cuales son: "Genere una máquina de estados que esté asociada a parte o a el total de su proyecto." Esto se implementó tanto en el módulo TOP donde podemos encontrar una máquina de estados de 8 estados, así como también en el módulo Inicio, en el cuál encontramos una máquina de estados de 3 estados, cuyas finalidades se encuentran descritas en la sección anterior.*Nivel de Logro:* 100%. Completamente logrado, el código implementado ha compilado, se logra simular luego de la implementación, se logra generar bitstram y cargarlo en la ZYBOZ7.

AC2: El código cumple con las funciones que habíamos planteado en nuestro proyecto las cuales son: "Utilice los 4 botones, 4 swiches y 4 leds de la ZYBOZ7 vistos en el curso". Esto se implementó en nuestro proyecto ya que los switches 0 y 1 son utilizados para definir qué slot de la RAM estamos leyendo, mientras que el switch 2 se utiliza como rst y, finalmente, el switch 3 se utiliza como switch de inicio para nuestro juego. Por otro lado, los leds son utilizados tanto por el módulo inicio como también por el módulo TOP para mostrar los lets de la secuencia del juego, utilizando incluso el led rgb disponible en la tarjeta. *Nivel de Logro:* 100%. Completamente logrado, el código implementado ha compilado, se logra simular luego de la implementación, se logra generar bitstram y cargarlo en la ZYBOZ7.

AC3: El código cumple con las funciones que habíamos planteado en nuestro proyecto las cuales son: "Utilice al menos 5 operadores y 5 atributos diferentes.". Esto se implementó a lo largo de todos los módulos desarrollados en este proyecto: TOP, MUX, INCIO, NGATE y ambos módulos de RAM. En particular podemos encontrar los siguentes operadores: asignación (<=, :=, =>), lógicos (NOT), aritméticos (+, **), de comparación, de concatenación. Por otro lado, los atributos que utilizamos son los atributos para una señal como lo es el atributo EVENT. *Nivel de Logro:* 60%. Parcialmente logrado, el código implementado ha compilado, se logra simular luego de la implementación, se logra generar bitstram y cargarlo en la ZYBOZ7.

AC4: El código cumple con las funciones que habíamos planteado en nuestro proyecto las cuales son: "Utilice variables para actualizar valores instantáneamente en alguna parte de la lógica programada.". Esto fue implementado en el módulo Inicio, donde se implementó la variable "estado_presente", la cual nos permite verificar en "tiempo real" el estado de nuestra máquina de estados en la cuál nos encontramos presente para, en caso de estar en el estado de carga, comenzar el contador que nos permite pasar al estado siguiente de juego. ". *Nivel de Logro:* 100%. Totalmente logrado, el código implementado ha compilado, se logra simular luego de la implementación, se logra generar bitstram y cargarlo en la ZYBOZ7.

AC5: El código cumple con las funciones que habíamos planteado en nuestro proyecto las cuales son: " Utilice código secuencial y código concurrente y evidencie claramente la diferencia de funcionamiento de ambos tipos de códigos. ". Esto fue implementado en el módulo Inicio, donde podemos observar la división del código secuencial y concurrente con la creación de la máquina de estados dividia en dos secciones: la upper section con el código concurrente y la lower section con el código secuencial. *Nivel de Logro:* 100%. Totalmente logrado, el código implementado ha compilado, se logra simular luego de la implementación, se logra generar bitstram y cargarlo en la ZYBOZ7.

AC6: El código no cumple con las funciones que habíamos planteado en nuestro proyecto las cuales son: " Haga uso de functions y procedures, donde el uso de ambas rutinas explicite sus diferencias principales". *Nivel de Logro:* 0%. No implementado.

AC7: El código cumple con las funciones que habíamos planteado en nuestro proyecto las cuales son: "Implemente alguna función que no se haya presentado en el curso. Esto puede ser activar algún hardware integrado en la ZYBOZ7, implementar la interconexión entre el PL y algún periférico". Esto fue implementado en el block desing de nuetro proyecto donde se utilizan los módulos AXI Smart Connect, el cual nos permite conectar nuestra RAM AXI Full con el DUT y a su vez nos conecta el DUT con el ATG en Test mode o Driver. *Nivel de Logro:* 100%. Totalmente logrado, el código implementado ha compilado, se logra simular luego de la implementación, se logra generar bitstram y cargarlo en la ZYBOZ7.

III. RESULTADOS IMPLEMENTACIÓN

La implementación de nuestro proyecto se dividió en cuatro distintas etapas. La primera etapa se centró principalmente en la ideación de nuestro proyecto, de tal manera de poder definir cuál sería el diseño que queríamos implementar de tal manera de cumplir tanto con las actividades obligatorias como también las actividades complementarias. Una vez que definimos el funcionamiento de nuestro módulo TOP, entonces procedimos a pasar a la segunda etapa: la implementación de la comunicación AXI lite y AXI full. En esta etapa surgieron bastantes complicaciones al implementar de manera correcta nuestro AXI Lite, como por ejemplo, poder coordinar la información de los archivos COE para que sean escritos cuando la salida del módulo RAM en AXI Full tome el valor de 1.

Una vez solucionado este problema y logrando la correcta implementación de los módulos mencionados, nos enfrentamos a la problemática de agregar las actividades complementarias faltantes, solucionando este problema con la implementación del módulo Inicio.

IV. CONCLUSIONES

Se ha logrado implementar de manera exitosa un sistema de comunicación AXI para el juego Simón Dice en la placa Zybo. Gracias a la implementación de los protocolos AXI Lite y AXI4 en Test Mode y Advanced Mode, respectivamente, ha sido psible manejar el nivel máximo de longitud de la cadena de LEDs del juego, lo que ajusta el nivel de dificultad a este y entrega una mayor interacción por parte del usuario. Se han utilizado módulos de RAM para almacenar los datos necesarios del juego, como la longitud de la secuencia de LEDs.

V. TRABAJOS FUTUROS

En relación a la implementación del juego de Simón dice en la zybo, se plantean distintas mejoras las cuales podrían ser agregadas en un futuro. Se plantea lo siguiente:

- Si bien con la implementación actual ya es posible optar por un nivel de dificultad mediante la limitación del largo de la secuencia de leds en el nivel máximo del juego, otra forma interesante de abarcar la dificultad del juego es mediante un divisor de clock que sea capaz de, mediante el manejo de switch's, regular la velocidad a la cual se visualiza la secuencia de leds. La idea de esto sería abarcarlo mediante comunicación AXI Lite, en Test Mode, tal cual en la actual implementación se maneja la secuencia de leds, pero para manejar la el clock que llega al módulo top.
- La máquina de estados podría ser ajustada para que inmediatamente cuando se comete un error como input, aparezca la luz roja del rgb y el usuario pierda. En la implementación actual se debe esperar a que el usuario ingrese un input equivalente al largo de la cadena visualizada para mostrar que el usuario se ha equivocado, Para esto la máquina de estados debe ser cambiada, haciendo que apenas se encuentre un valor incorrecto, se pase al siguiente estado.
- Con tal de realizar un juego más completo, se le podría agregar un buzzer a la zybo, con tal que cuando el usuario se equivoque, este suene. Para esto, se debe realizar la correcta conexión el archivo de constraint de la zybo, y hacer que este dispositivo suene cuando se entra al estado de input incorrecto, el cual en el código esta definido como INCORR INP.

REFERENCES

[1] Saba Janamian. (2023, March 13). 011 ECE524 Lab Simon Game with VHDL [Video]. YouTube. https://www.youtube.com/watch?v=KbViJ7HRddM

[2] Santifs. (s.f.). GitHub - santifs/simon-game-vhdl: VHDL game that displays incremental random sequences on an LED Matrix by creating a finite state machine and implementing RAM and ROM models. GitHub. https://github.com/santifs/simon-game-vhdl

[3] Instructables. (2017). VHDL Game: LED - Button Push Game Instructable. Instructables. https://www.instructables.com/VHDL-game-LED-Button-Push-Game-Instructable/