Fecha de entrega 10 de Mayo de 2023.

Simon Says

ARACELLY CID H.[1], ISAÍAS NAVARRO Q.[2]
[1]Pontificia Universidad Católica de Chile (e-mail: aracelly.cid@uc.cl, isaas.navarro@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 Simon Says corresponde a un juego de memoria que cuenta con 8 modalidades de juego. Cada una de ellas está asociada a una posición de los switches de la Zybo Z7 10, donde al elegir una modalidad comienza una secuencia de prendido y apagado de leds que el jugador debe memorizar para poder recrear la misma secuencia apretando los botones de la tarjeta en el mismo orden. Si el jugador lo hace de forma correcta se prende un led verde, lo cual indica que el jugador gano el juego, en caso contrario, se prende una luz roja y el jugador pierde esta partida. Para poder realizar esta implementación, se programaron bloques en VHDL, los cuales corresponden a una máquina con 5 estados posibles, llamada Simon says, bloques de Debouncer para estabilizar las señales al apretar cada botón en el juego, una memoria Ram donde se encuentran registrados los modos de juego y protocolos de comunicación Axi lite y Axi full para poder controlar los modos de juego. Finalmente, se obtuvo como resultado una correcta implementación de Simon Says, funcionando en todas sus modalidades de juego listo para ser usado por cualquier persona.

INDEX TERMS VHDL, Zybo Z7 10, Máquina de estados, Simon Says, Memoria Ram, AXI Traffic Generator, ATG Test Mode, ATG Advance Mode, AXI Lite, AXI , Debouncer.

I. ARQUITECTURA DE HARDWARE (1 PUNTO)

LA arquitectura de hardware se explica mediante el siguiente diagrama de bloques representativo:

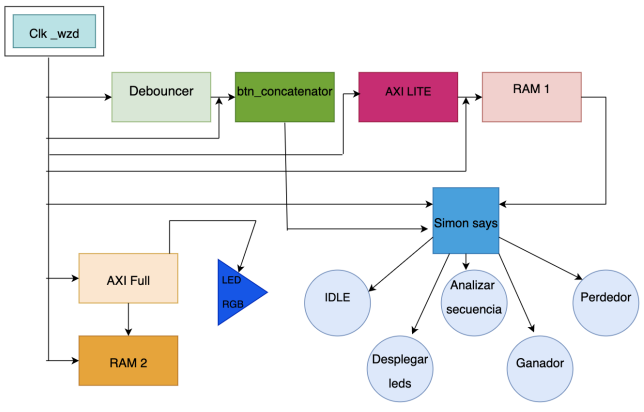

FIGURE 1: Diagrama explicativo Simon Says.

Este hardware se encuentra gobernado por un clocK wizard, el cual ayuda a sintonizar todos los clock utilizados para los diferentes bloques y obtener una correcta secuencia de implementación. Además, se tienen 4 bloques de Debouncer, los cuales ayudan a estabilizar la señal de cada boton cuando éste es apretado eliminando los posibles "glitches" que puedan existir. Estos bloques de Debouncer son conectados al bloque Btn_concatenator, el cual tiene separado cada uno de los botones que pueden ser apretados en el juego, siendo este bloque conectado a la máquina de estados Simon Says. Ésta última, posee 5 estados para la operación del juego, los cuales corresponden : IDLE, Desplegar leds, Analizar secuencia, Ganador y Perdedor. Estos estados se complementan para poder determinar si se gana o se pierde la partida. También, se tienen los bloques de comunicación Axi, los cuales corresponde a Axi lite y Axi full, cada uno acompañado de su respectiva RAM para el procesamiento de las datas (secuencias de los leds). Ram 1 entrega el registro de las secuancias de los leds, según la modalidad de juego elegida a la máquina de estados Simon Says. Finalmente, se tiene el bloque de Axi full el cual ayuda al despliegue de un led RGB.

II. ACTIVIDADES REALIZADAS (1.5 PUNTOS)

Descripción:

AO1 (66%): Se logró incorporar dos Components dentro de la Entity global al interconectar todos los bloques de nuestro proyecto.

AO2 (100%): Se logró implementar más de 3 Package diferentes en este proyecto las cuales corresponden a: Simon says, Debouncer, Boton_concatenator, AXI_lite, AXI_full, RAM y clk_wzd. Estos bloques fueron interconectados entre

si para la implementación total de este proyecto, pudiendo ser sintetizados sin problemas.

AO3 (100%): Se implementó la comunicación Axi con un IP core esclavo y un ATG maestro en Test Mode, es decir, comunicación Axi Lite, a partir de la cual se tienen las secuencias de los leds para cada modo de juego, lo cual se configuró mediante los archivos.coe de control, address, data y mask. Finalmente, se implementó la comunicación Axi con un IP-core esclavo y un ATG maestro en Advance Mode para el desplegamiento de un led RGB.

AC1 (100%): Se generó una máquina de estados para este proyecto llamada "Simon says", cuya arquitectura está conformada por codigo secuencial y concurrentes. Dentro de este último se encuentran 5 estados denominados: IDLE, Desplegar_leds, Analizar_secuencia, Ganador y Perdedor, los cuales permiten el despliegue de la secuencia de juego de leds e indicar si el jugador gano o perdió la partida.

AC2 (100%): Se utilizaron 4 leds, los cuales se despliegan para cada secuencia de juego. También se utilizaron los 4 botones de la tarjeta para para recrear la secuencia desplegada por los leds. Por último, se utilizaron los 4 switches para elegir la secuencia que desplegaran posteriormente los leds. Estas implementaciones se logran en su totalidad en el desempeño del juego.

AC3 (50%): En este apartado solo se logran utilizar 5 operadores, los cuales vienen dados por comparación, igual, NOT, AND, adición, multiplicación y se pueden encontrar a lo largo del código de la máquina de estados Simon Says.

AC4 (100%): Se utilizan variables en el código de la máquina de estados "Simon_Says", las cuales corresponden a "timer", "led_counter", "player_counter". Sus funcionalidades respectivas corresponden a que los leds puedan mantenerse encendidos por 1[s] cuando la variable "timer" sea igual a TIMER_LEDS y las últimas dos variables corresponden a contadores que ayudan al despliegue de los leds y de los botones contando cada uno hasta 5.

AC5 (100%): Se utilizó codigo secuencial y concurrente en la máquina de estados. El código secuencial se puede evidenciar en la primera parte de la máquina de estados con el primer Process, cuya lista de sensibilidad es solo el clock (clk) y se va ejecutando con el resto del código. Por otro lado, el código concurrentre se puede evidenciar en la última parte de la máquina de estados los cuales se encuentran después del término del Process. También se puede encontrar código concurrente en la implementación de la RAM1.

AC6 (0%): No se implementó ninguna función ni procedures.

AC7 (100%): Se implementaron leds RGB. Verde para el estado ganador y Rojo para el estado Perdedor. Por último, se implementó un led morado con ayuda de la comunicación Axi full.

III. RESULTADOS DE SIMULACIÓN (3 PUNTOS)

A continuación se muestran las simulaciones para la identificación de Delays. Es posible observar un retardo de 7 ns entre el clock y la señal de activación de ese mismo flanco.

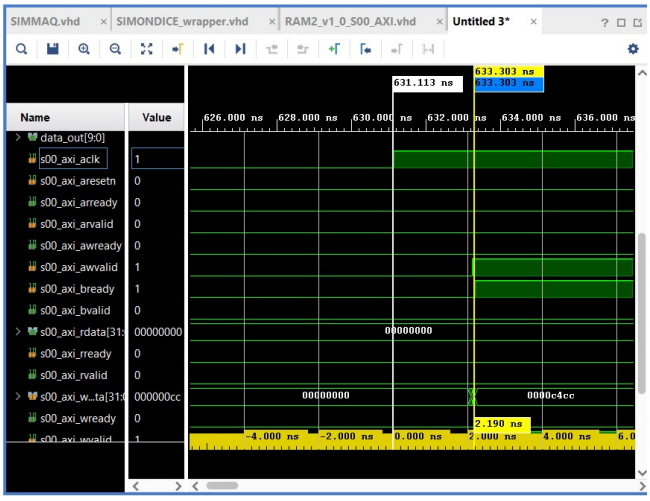

FIGURE 2: Simulación para determinación de Delays.

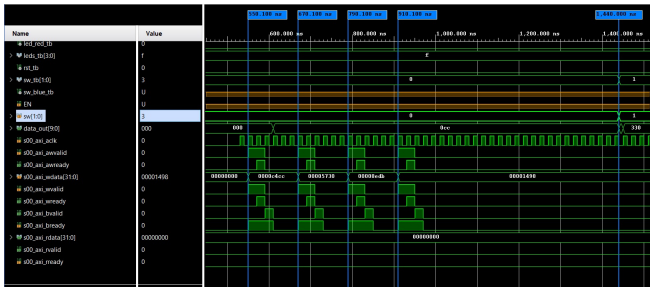

FIGURE 3: Simulación despliegue de datos.

Por otro lado, cabe destacar que en la máquina de estados se recurrió a la utilización de señales, ya que se necesitaba su respectiva actualización después de cada flanco de subida de clock y no de forma instantánea que sería lo que se tendría si se ocupara una varible. Esto debido a la implementación de una máquina de estados.

Por último, se realizó una simulación del protocolo AXI en su versión test mode para el envío de la secuencia de leds (data) a la máquina de estados. Esto se realizó como bloque individual con el despliegue de un led para notar el correcto despliegue de éste. Posteriormente, se implementó en el proyecto con la interconexión de todos los bloques.

IV. RESULTADOS IMPLEMENTACIÓN (0.1 PUNTOS)

En general, se obtuvo una implementación conformada por una Máquina de estados "Simon Says", la cual después de varios intentos logró su funcionalidad. Principalmente, se obtuvieron problemas con el estado del Despliegue de leds, ya que en un principio se desplegaban a una velocidad muy rápida, donde los cambios se hacían imperceptibles al ojo humano. Por lo tanto, para poder solucionar esto, se debió implementar un Timer que mantenga el led prendido cuando sea igual a "Timer_leds", el cual corresponde a una frecuencia de prendido de 125000000, es decir, de 1[s].

Por otro lado, se obtuvieron varios problemas de imple-

mentación del ATG Test Mode, ya que se configuró de forma errónea el address.coe sin considerar el tamaño de la memoria y la sintaxis que cuenta con una asignación de 4 en 4 cada dirección. Esto se solucionó redefiniendo el archivo address.coe de manera correcta. Por ultimo, en la implementación total del proyecto se tuvieron errores como la mala interconexión de los diferentes bloques utilizados, lo cual se corrigió por inspección de cada una de las conexiones.

V. CONCLUSIONES(0.2)

Enumere las conclusiones más importante de su proyecto. Recuerde que:

- La cantidad de modalidades de juego esperadas eran 8, según la combinación de switch elegida lo cual se ve reflejado con el despliegue de la secuencia de leds en cada modo.
- Se esperaba que los 4 botones de la tarjeta imitaran la secuencia de leds mostrada al apretarlos, lo cual se estabilizó ppor medio de un Debuncer para eliminar ciertos glitches de la señal.
- En cada modalidad de juego, en donde se despliega una secuencia de prendido y apagado de leds, se esperaba que cada led se mostrara en 1 [s] y se logró que se mostrara cada uno a esa velocidad.
- Se esperaba desplegar el Led RGB verde para indicar que un jugador ganó la partida y se logró desplejar este led en color verde cada vez que un jugador completaba de forma correcta la secuencia con los botones de la tarjeta. De forma análoga, ocurre lo mismo en el caso perdedor, pero con el led RGB en rojo.
- Se esperaba guardar cada secuencia de leds de la modalidad de juego elegida en una Memoria RAM para poder indicarle como entrada a la Máquina de estados que secuencia se debe desplegar, lo cual se realizó por medio del protocolo ATG en Test Mode mediante los 4 archivos.coe de control, address, data y mask con las isntrucciones respectivas en cada caso.
- Se esperaba la implementación AXI full, la cual se implementó para el despliegue de un led RGB de color morado, con un correcto uso del flujo de datos.

VI. TRABAJOS FUTUROS (0.2)

Este proyecto admite la integración de nuevas versiones. Se podría modificar cada modalidad de juego agregándole más de una secuencia de prendido y apagado de leds y un sistema contador de puntaje, es decir, al elegir una modalidad de juego con un switch, se muestre una secuencia de leds y si el jugador la recrea de forma correcta con los botones, se muestre otra secuencia, pero con una mayor velocidad y así sucesivamente hasta que el jugador pierda o se complete la cantidad total de secuencias. Sumado a lo anterior, en relación a el sistema contador de puntaje, se podría ir guardando los aciertos del jugador y que éste se compare con el puntaje de otro jugador para elegir como ganador al que tenga un mayor puntaje.

Técnicamente, esta nueva modalidad de juego se podría llevar a cabo implementando un bloque divisor de clock para ir mostrando a diferentes velocidades cada secuencia de leds. También, se podría modificar la máquina de estados, con un estado donde se comparen los puntajes de cada jugador, los cuales pueden ser escritos en una memoria y de esta forma se elija el mayor puntaje como ganador. Por otro lado, se podría agregar otro estado que cuente el tiempo que se demora cada jugador en realizar la secuencia, solo si la completa entera de forma correcta y que elija como ganador el que la realiza en un menor tiempo.

REFERENCES

[1] Xilinx. (2021). PG125 - AXI Traffic Generator [Documentation]. Retrieved May 9, 2023, from https://docs.xilinx.com/v/u/en-US/pg125-axi-traffic-gen.
[2] Xilinx. (2021). AXI Interconnect [Documentation]. Retrieved May 9, 2023, from https://docs.xilinx.com/r/en-US/pg059-axi-interconnect/Introduction.
[3] Doulos. (n.d.). VHDL Testbench Creation Using Perl [Blog post]. Retrieved May 9, 2023, from https://www.doulos.com/knowhow/perl/vhdl-testbench-creation-using-perl/.
[4] FPGA4Student. (2017, August 15). VHDL code for debouncing buttons on FPGA [Blog post]. Retrieved May 9, 2023, from https://www.fpga4student.com/2017/08/vhdl-code-for-debouncing-buttons-on-fpga.html.
[5] Tang, J. (n.d.). VHDL Reference Guide [PDF]. Retrieved May 9, 2023, from https://edg.uchicago.edu/ tang/VHDLref.pdf.