Fecha de entrega 8 de julio de 2024.

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

Conway's game of life

JORGE APUD[1], MARTÍN ATRIA[1]
[1]Pontificia Universidad Católica de Chile (e-mail: japud@uc., martin.atria@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 Este proyecto consiste en implementar un simulador de *Conway's Game of Life*, un juego de automatas celulares donde la vida de cada celula se encuentra unicamente determinada por sus celulas vecinas y ella misma, esto permite generar un sistema *Turing complete* por lo que con las condiciones iniciales correctas se es posible programar cualquier cosa dentro del juego. El proyecto presenta una grilla de celulas a traves de la pantalla LCD, permite cambiar el estado inicial de las celulas y muestra una pantalla final cuando todas las celulas han muerto. Todas las interacciones con el juego se realizan mediante distintos perifericos de la tarjeta Booster.

INDEX TERMS Conway's game of life, C, VHDL, Turing complete, sensores.

I. ARQUITECTURA DE HARDWARE Y SOFTWARE (1 PUNTO)

EL hardware del proyecto consiste en primer lugar en conectar los perifericos de la tarjeta Booster al procesador zynq, para esto utiliza los IP cores: *AXI QUAD SPI*, *AXI IIC* y *AXI GPIO*. Utiliza dos bloques SPI para conectar la pantalla LCD y el conversor analogo - digital, este segundo se utiliza para las medidas del acelerometro, los potenciometros y el microfono. El bloque IIC se utiliza para conectar el sensor de luz. Utiliza dos bloques GPIO para conectar los botones y parametros de la pantalla. Todos estos bloques se utilizan para convertir su señal respectiva en protocolo AXI y asi poder conectarse al puerto AXI del procesador.
Por otra parte estan los bloques de interrupción, para esto utiliza dos *AXI TIMER INTERRUPT* y el bloque *AXI IIC* nuevamente, estos bloques se unen a un bloque concatenador que permite juntar sus salidas en el puerto de interrupciones del procesador.
Por ultimo el hardware tiene una conexion desde el procesador a un IP core propio llamado *New cancion* el que maneja la reproducción de la musica del juego en conjunto con un bloque basado en el bloque buzzer desarrollado en el curso, este se diferencia del bloque original del curso en que no requiere de un enable para sonar, suena siempre que recibe una pwm, el bloque *New cancion* se comunica con el procesador por medio de AXI.

El flujo del programa consta de 4 etapas principales:

- Menu Principal
- Modo edición

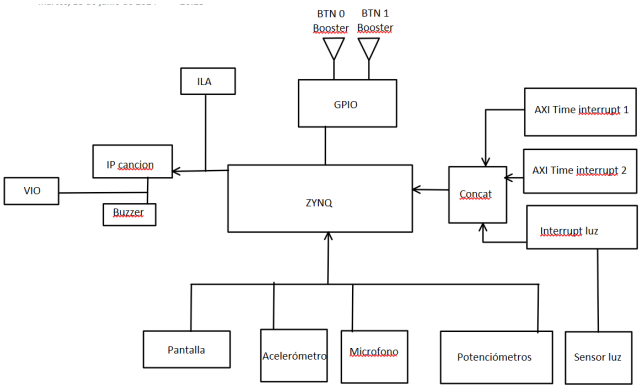

FIGURE 1: Diagrama Hardware

- Modo simulación
- Modo Game Over

La primera vez que se está en el modo **Menu Principal** se configura el juego mediante la consola serial de vitis, esta permite elegir la cancion a reproducir cuando termine el juego, despues esta es solo una pantalla de espera. Para pasar de esta a modo edición se tiene que hacer un ruido por sobre un umbral en el microfono, el valor del umbral se determinó de manera en que un soplido fuerte en el microfono sea suficiente.

En el modo **edición** se permite cambiar las condiciones iniciales del juego, cambiando las celulas muertas por vivas y viceversa. Para elegir que celda cambiar se utilizan los

potenciometros y para alternar el estado de la celda escogida se utiliza un boton de la tarjeta Booster. La celda en la que se está actualmente se ve de color azul mientras que el resto de las celdas se ven de color blanco si estan vivas y negro si estan muertas, si se desea resetear el estado a todas las celulas en blanco se puede girar la tarjeta para estar verticalmente y el acelerometro borrará todas las celdas. Para pasar de este modo al modo de simulación se debe apretar el otro boton de la tarjeta Booster.

En el modo **simulación** se visualiza la evolución del juego, aqui se puede ver como van cambiando las iteraciones del juego en tiempo real, para pausar el juego se debe tapar el sensor de luz y esto generará una interrupción que detendra la simulación. Esta parte del codigo está muy optimizada por lo que el codigo corre más rapido de lo que se puede percibir por el ojo humano, es por esto que se tiene un delay en los calculos de la logica interna del juego. Mientras existan celulas vivas la simulación seguira corriendo, pero apenas mueran todas las celulas cambia al modo Game Over.

En el modo **Game Over** se presenta una pantalla de Game Over, se toca la musica configurada en el Menu Principal y se guardaria en la tarjeta SD el numero de iteraciones transcurridas en ese juego, esto no se logró desarrollar por completo. Una vez que termina la musica este modo termina y vuelve al modo Menu Principal.

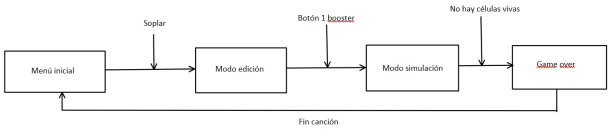

FIGURE 2: Diagrama Software

II. ACTIVIDADES REALIZADAS (1.5 PUNTOS)

AO1 (100%): *Descripción:* El proyecto utiliza la pantalla para presentar el desarrollo del juego, utiliza el sensor de luz para generar un interrupt, el acelerometro para resetear el modo edición y el microfono para pasar del Menu Principal a modo edición. Se cambian los registros del sensor de luz para configurar el interrupt. *Nivel de Logro:* 100%. Completamente logrado.

AO2 (100%): *Descripción:* Se generan dos interrupts por timer en vivado, uno maneja la lectura del conversor analogo digital y el otro maneja la lectura de los botones de la tarjeta, estos se separan porque los botones deben leerse a una menor frecuencia para evitar leer el jitter. Se utiliza un interrupt del sensor de luz en el modo simulación. *Nivel de Logro:* 100%. Completamente logrado.

AO3: (100%): *Descripción:* Se crea el IP core **New canción**, este bloque se comunica con el procesador por medio de AXI para ser configurado. Esta configuración se logra por medio del terminal serial, donde se solicita la secuencia de notas para reproducir. *Nivel de Logro:* 100%. Completamente logrado.

AC1: (100%) *Descripción:* Se crea una maquina de estados que maneja el flujo general del programa, esta cambia el modo en el que se encuentra el juego. *Nivel de Logro:* 100%. Completamente logrado.

AC2: (100%) *Descripción:* Se utilizan estructuras para manejar las celulas vivas y muertas en cada iteración y poder comparar cuales cambian para manejar estas en la pantalla, arreglos para recibir la información por UART, punteros dentro de la función *InterruptSystemSetup*, se crean y utilizan multiples funciones y se utilizan macros para definir las direcciones de los IP cores relacionados a la musica y para redefinir los ID de los interrupts por nombres más simples. *Nivel de Logro:* 100%. Completamente logrado.

AC3: (100%) *Descripción:* Se utilizan los potenciometros para escoger una celda en el modo edición y se utilizan los botones para multiples funcionalidades dependiendo del estado como se explicó en la arquitectura de software. *Nivel de Logro:* 100%. Completamente logrado.

AC4: (100%) *Descripción:* Se realiza el boot de la Zybo. *Nivel de Logro:* 100%. Completamente logrado.

AC5: (100%) *Descripción:* Se utiliza VIO para visualizar la frecuencia que se le envia al bloque new buzzer desde el bloque new cancion e ILA para visualizar la transacción por AXI del ciclo de trabajo del bloque buzzer. *Nivel de Logro:* 100%. Completamente logrado.

AC6: (100%) *Descripción:* Se utiliza el buzzer para reproducir una canción escogida por el usuario por medio de UART al comienzo del juego. *Nivel de Logro:* 100%. Completamente logrado.

AC7: (30%) *Descripción:* Se tienen las funciones para escribir el numero de iteraciones en una tarjeta SD pero no se ejecutan.

Se escribe un archivo en la tarjeta SD que guarda la cantidad de iteraciones que duró el juego. *Nivel de Logro:* 30%. Parcialmente logrado.

III. RESULTADOS (3 PUNTOS)

Se valida el funcionamiento del IP core new cancion revisando con vio su salida de frecuencia, en la siguiente imagen se puede observar la frecuencias que se envia para el buzzer.

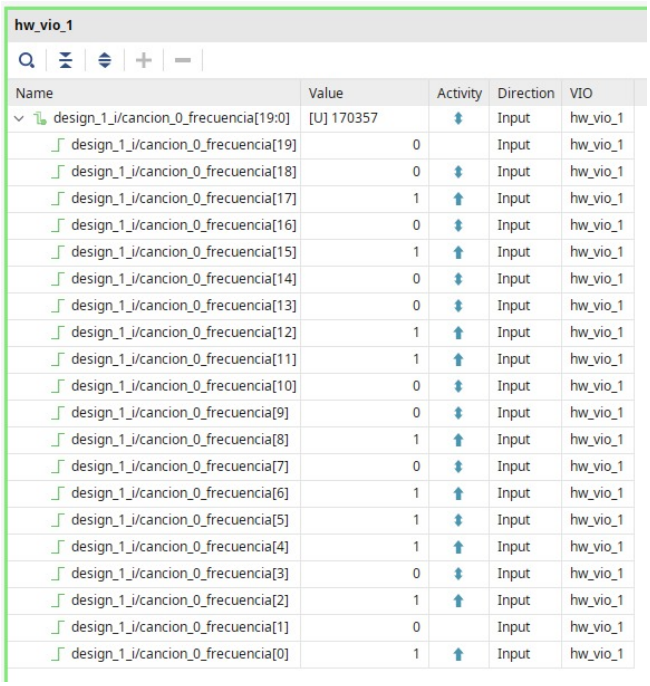

FIGURE 3: VIO

En las siguientes imagenes se puede ver como utilizando la herramienta de *debug* de Vitis, es posible capturar una interrupción, en este caso la interrupción generada por el sensor de luz, y ver qué es lo que pasa con la variable que está configurando. En el caso de este proyecto, cuando existe un bajo nivel de luz, el juego se pone en pausa.

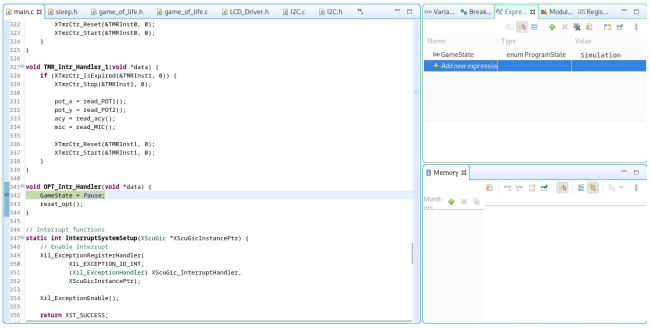

FIGURE 4: Interrupt luz 1

La primera figura muestra la captura inmediatamente después de capturar la interrupción

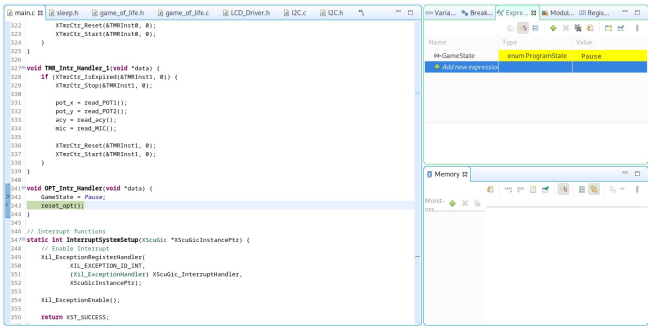

FIGURE 5: Interrupt luz 2

La segunda figura muestra la línea donde se cambia el estado del juego

Se confirma el correcto funcionamiento del hardware utilizando ILA para revisar la conexion AXI desde el procesador al buzzer, donde se utiliza AXI awvalid para iniciar la captura de ILA. En la siguiente imagen se puede visualizar el ciclo de trabajo que se le envia al bloque buzzer, se ve que este tiene un valor de 1, valor correspondiente a lo enviado desde el procesador.

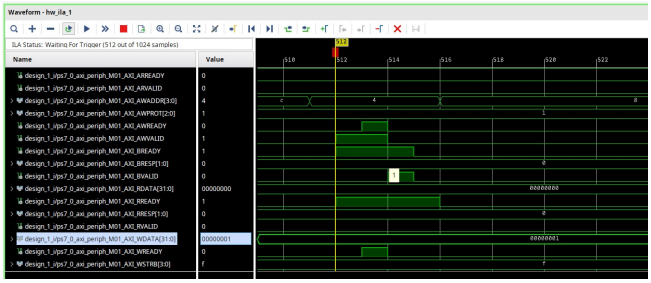

FIGURE 6: ILA

IV. RESULTADOS IMPLEMENTACIÓN (0.1 PUNTOS)

Nuestra implementación fue exitosa, logramos desarrollar casi todos los objetivos que planteamos inicialmente. Logramos presentar la logica completa del juego y la transición entre modos funciona a la perfección. Logramos utilizar los perifericos de maneras creativas y en partes cruciales del juego. Lo unico que nos faltó integrar fue el uso de la tarjeta SD.

V. CONCLUSIONES(0.2)

- Se logró alta eficiencia en el codigo, el juego logra correr hasta 100+FPS, la unica restricción es el tiempo de refresco de la pantalla.
- Se logró una precision temporal adecuada en el manejo de la canción, se logra medir dentro del codigo el retraso de reproducción de la canción por el tiempo de envio de todos los datos y la inicialización del buzzer.

VI. TRABAJOS FUTUROS (0.2)

Como funciones pendientes, se podrían guardar configuraciones iniciales del juego en diferentes archivos de la tarjeta

SD, luego con un menú, se podría seleccionar la configuración inicial y editarla para no tener que dibujar toda la configuración inicial.

REFERENCES

[1] IGOR PERALTA. (2022, 16 noviembre). Video Resumen Ayudantía 10 - SD [Vídeo]. YouTube. https://www.youtube.com/watch?v=n5zFwmG9dhw
[2] Vipin Kizheppatt. (2020, 15 mayo). Reading and Writing from SD Card (Binary files) [Vídeo]. YouTube. https://www.youtube.com/watch?v=i_zWX0mBt1k
[3] Wikipedia contributors. (2024, 2 julio). Conway's Game of Life. Wikipedia. https://en.wikipedia.org/wiki/Conway