Fecha de entrega 11 de Diciembre de 2023.

SNAKE

FELIPE MARTINEZ[1], RODRIGO REYES[1]
[1]Pontificia Universidad Católica de Chile (e-mail: felipe.martinez1@uc.cl, rreys@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 presente proyecto consiste en la recreación del juego Snake en la tarjeta BoosterPack MK II. Este juego comprende de una serpiente que va creciendo a medida que consume la manzana puesta en pantalla, tratando de evitar la colisión de la serpiente con los muros y sí misma. Para recrear de manera prolija el juego anterior, se utiliza una máquina de estado que consta de un menú, el juego, la pantalla de perdida, la pantalla de los mejores puntajes y una pantalla de warning de temperatura. Al partir el juego, este parte en el estado de menú, donde se utiliza el acelerómetro para iniciar el juego, al tener que girar la tarjeta hacia adelante. Luego, en el estado del juego, se utiliza el joystick para dar la dirección de movimiento de la serpiente. Además, para que el movimiento de la serpiente se utiliza una interrupción por timer para tener un movimiento constante. Luego, al perder este pasa por unos segundos al estado de perdida antes de pasar al estado de puntajes, donde se mostrará en pantalla los 5 mejores puntajes. Los cuales están guardados en una tarjea SD externa. Finalmente, para volver al estado de menú, se utiliza el mismo método que para pasar de menú a juego. Además, durante todos los estados se puede escuchar una canción, la cual se puede apagar utilizando uno de los potenciómetros.

INDEX TERMS VHDL, maquina de estados, interrupciones, BOOSTERPACK MK II, Vitis, Snake, juegos, tarjeta SD.

I. ARQUITECTURA DE HARDWARE Y SOFTWARE (1 PUNTO)

EL siguiente proyecto recrea el juego Snake en la tarjeta boosterpack MK II, la cual es manejada por la ZYBO-Z7. Para poder lograr recrear al anterior juego mencionado, es necesario crear ,primeramente, el Hardware del juego, para sí luego programar el software y la lógica necesaria para el correcto funcionamiento del juego.

1) Arquitectura de Hardware:

Por el lado del hardware, este proyecto utiliza el procesador ZYNQ, el cual utiliza un AXI Interconnect para conectarse con todos los perifericos de la Booster, con los IPCores propios creados y con el LCD. Los perifericos utilizados son el sensor de temperatura, el sensor de luz, el Joystick, el Acelerometro, el Buzzer y el Potenciometro. El manejo de la musica es mediante 2 IPCores propios llamados Canción y Axi Musica. El primero contiene las notas musicales en una ram, las cuales son enviadas utilizando un divisor de clock al bloque Axi Musica. Luego, el bloque Axi Musica recibe las notas y utilizando una PWM toca las notas en el buzzer. Además, este bloque se conecta al Axi Interconnect para que el usuario pueda settear al comienzo del juego si quiere tener la musica prendida o apagada, ingresando su eleccion desde el PC por comunicación UART. Estos dos IPCores propios, permiten que la musica del juego suene al mismo que tiempo que este, y se utilizan los leds de la zybo para indicar cuando la musica esta pausada. Cabe destacar que si el usuario quisiera prender o apagar la musica en algun momento, esta tambien se puede controlar mediante el potenciometro.

Para las interrupciones, se utilizan 2 AXI Timers, y un bloque GPIO para generar las 2 interrupciones de los timers, y la interrupcion del sensor de luz. Estas interrupciones van concatenadas por un bloque Concat y luego conectadas a la entrada de interrupciones del ZYNQ.

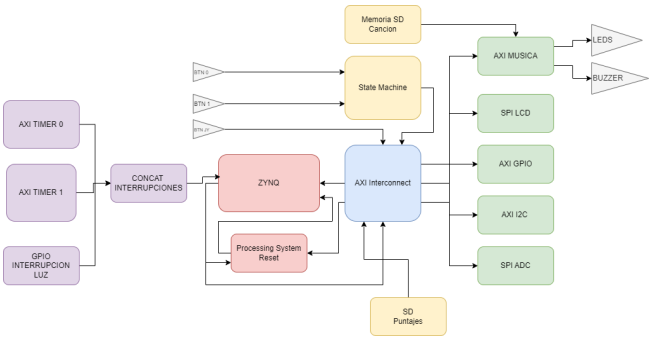

FIGURE 1: Diagrama explicativo del Hardware del Proyecto

2) Arquitectura de Software:

Por el lado del software, primero se inicializan las interrupciones, los GPIOS y todas las comunicaciones necesarias para la correcta interacción con la tarjeta booster. Posterior a esto, se le pide al usuario elegir la dificultad del juego, mediante comunicación UART con el PC, la cual se reflejará en la rapidez en que una interrupción por timer se active, y por ende, en el movimiento de la serpiente en el juego, ya que la velocidad de la serpiente está dada por la interrupción. Además, se pide, con el mismo sistema, si se quiere escuchar o no música durante el juego. No obstante, este se podrá igualmente manejar durante el juego, a través de uno de los potenciómetros.

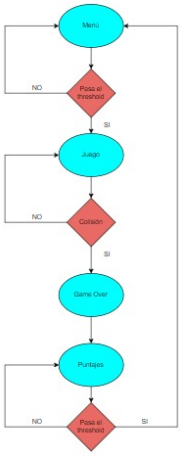

FIGURE 2: Diagrama de flujos de la maquina de estados

Luego de configurar los parámetros iniciales, empieza la máquina de estado que será la que controlará el flujo del programa. Al inicio, se mostrará el menú del juego, pudiendo cambiar al juego si se gira la tarjeta hacia adelante. Dentro del juego, se controla la dirección de la serpiente con el joystick, con el fin de consumir las manzanas que se van mostrando en pantalla, lo que conllevaría al crecimiento de la serpiente. Al consumir una manzana, se obtendría 10 puntos. Además, se obtiene 1 punto por cada segundo sobrevivido, donde el tiempo es calculado con precisión por una interrupción por timer. Para perder, es necesario colisionar con los bordes de la pantalla o con el mismo cuerpo de la serpiente. Al perder, se mostrará por unos segundos la pantalla de pérdida, para después mostrar los 5 puntajes más altos del juego, los cuales están guardados en una tarjeta SD externa. Finalmente, para volver al menú, se tiene que girar la tarjeta hacia al frente.

En adición a todo esto, existe un estado extra de seguridad, el cual se activa cuando la temperatura supera un umbral, y avisa por pantalla la subida de temperatura. También, existe una interrupción de luz, que pasar por un umbral, aparece una manzana extra para consumir.

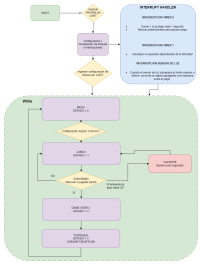

FIGURE 3: Diagrama de flujos del software

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 (100%): *Descripción:* Se utiliza la pantalla LCD y 3 periféricos extras, como lo es el joystick, el acelerometro y el buzzer.

Nivel de Logro: 100%. Completamente logrado, se logra imprimir en pantalla y los 3 periféricos funcionan correctamente.

AO2 (100%): *Descripción:* Se logro utilizar 2 interrupciones generadas por Timers implementados en Vivado, las cuales leen periodicamente el estado de perifericos como el sensor de temperatura y también permitiendo mover el snake y sumarle puntaje al jugador cada segundo, y ademas se implemento la interrupcion del Sensor de Luz, en donde se configura un limite superior y un limite inferior para este.

Nivel de Logro: 100%. Completamente logrado

AO3(100%): *Descripción:* Se crea un IP core que interactua con el buzzer y que interactue con el ZYNQ. Además, este se configura al recibir un valor por el usuario mediante UART. El cual permite poner pausa a la música.

Nivel de Logro: 100%. Completamente logrado, se logra pausar la música, al enviar un 0 por UART.

AC1(100%): *Descripción:* Se utiliza una máquina de estados asociada al funcianamiento de su proyecto.

Nivel de Logro: 100%. Completamente logrado, se logra implementar de forma correcta la máquina de estados.

AC2(100%): *Descripción:* Se utilizan estructuras, arreglos, punteros, funciones y macros a lo largo de todo el código.
Nivel de Logro: 100%. Completamente logrado.

AC3(100%): *Descripción:* Se utilizan 3 periféricos adicionales, el sensor de luz, el sensor de temperatura y el potenciometro.
Nivel de Logro: 100%. Completamente logrado, se logran utilizar los periféricos adicionales sin error algúno. El sensor de luz se utiliza para generar interrupciones, el sensor de temperatura se utiliza para revisar constantemente que la temperatura no sea superior a cierto umbral y la placa no se sobrecaliente, y finalmente el potenciometro es utilizado para poder pausar o ponerle play a la musica.

AC4(100%): *Descripción:* Se logra Bootear la zybo para que quede guardado el código.
Nivel de Logro: 100%. Completamente logrado, se logra ver el código en la zybo, sin tener que cargarlo nuevamente.

AC5(100%): *Descripción:* Se logra utilizar el VIO y la ILA para onteractuar con un modulo y validar que la informacion se envia y se recibe de manera correcta. La diferencia entre la VIO y la ILA radica principalmente en que la VIO es un Virtual Input Output, el cual sirve simplemente para enviar o leer una señal, en cambio la ILA tiene un funcionamiento parecido a un osciloscopio, en donde uno puede ver distintas señales en el tiempo.
Nivel de Logro: 100%. Completamente logrado, se logra utilizar VIO para el reseat de la música e ILA para observar la frecuencia de salida, la PWM del Buzzer y la salida del concat.

AC6(100%): *Descripción:* Se logra tocar múscia mediante el buzzer, durante todo el juego.
Nivel de Logro: 100%. Completamente logrado, se logra escuchar la canción en todo momento.

AC7(100%): *Descripción:* Se utiliza la memoria externa SD para guardar los puntajes de los jugadores.
Nivel de Logro: 100%. Completamente logrado, se logran guardar los 5 mejores puntajes en la SD.

III. RESULTADOS (3 PUNTOS)

Para validar el correcto funcionamiento del IP Core creado, se utiliza la ILA y el VIO. En las siguientes 3 imágenes, se puede ver el valor de la frecuencia enviadas en distintos tiempos.

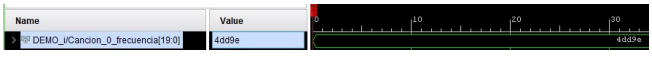

FIGURE 4: frecuancia en tiempo 1

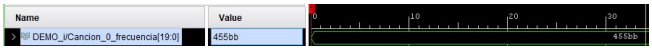

FIGURE 5: frecuancia en tiempo 2

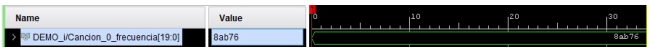

FIGURE 6: frecuancia en tiempo 3

Con esto es evidente que las frecuencias van variando durante el transcurso del tiempo, lo que nos permite evidenciar la creación de la canción por parte del módulo creado. Además, gracias a la VIO, se puede activar el reseat. Al activarlo, el valor de frecuencia se queda en 4dd9e. Lo cual sería lo esperado, ya que la VIO estría reseteando la canción constantemente, y el primer tono tiene de frecuencia 4dd9e.

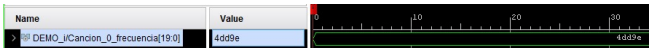

FIGURE 7: Valor de frecuencia con reseat activado

Para ver los resultados del software, se utiliza el Debug y se analiza los valores de las variables y como estas van provocando cambios en el juego. Si bien existen varias variables, se analizan las más importantes. En la siguiente figura, se muestran los valores iniciales de “estado”, el cual es el estado de la máquina, “se_mueve” que es una variable que cambia la interrupción para habilitar el movimiento de la serpiente, “posx” y “posy” siendo las posiciones de cada parte de la serpiente.

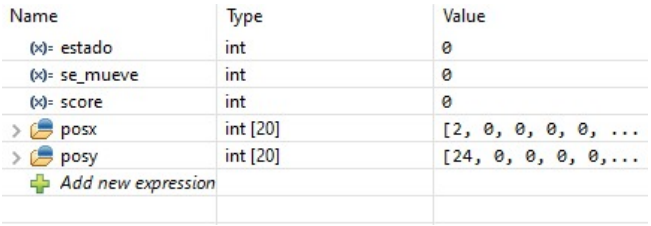

FIGURE 8: Valor iniciales

Luego, en la siguiente figura se puede observar que la variable “se_mueve” se activó, ya que paso el suficiente tiempo para que la interrupción pasara. Sin embargo, ni una posición de la serpiente cambio, debido a que se encuentra en el estado 0, el cual es el estado de menú. Ahora, en la imagen, adyacente a esta última, se puede ver que, si hubo un cambio en la posición de la serpiente, esto debido a que en ese momento se encuentra en el estado 1, siendo esta ya el estado de juego. Además, se percata del cambio en “se_mueve”, ya que el movimiento ya se efectuó. Entonces se tiene que esperar que la interrupción permita la movilidad otra vez.

Name	Type	Value
estado	int	0
se_mueve	int	1
score	int	5
posx	int [20]	[2, 0, 0, 0, 0, ...
posy	int [20]	[24, 0, 0, 0, 0,...
Add new expression		

FIGURE 9: Valor en estado 0

Name	Type	Value
estado	int	1
se_mueve	int	0
score	int	5
posx	int [20]	[4, 0, 0, 0, 0, ...
posy	int [20]	[24, 0, 0, 0, 0,...
Add new expression		

FIGURE 10: Valor en estado 1

Finalmente, en la siguiente figura, se logra ver que estando en el estado 1, las posiciones de la serpiente no se cambian. Esto se debe a que la variable "se_mueve", no se encuentra activada. Impidiendo el cambio de posición, y ayudando a controlar el movimiento de la serpiente de forma constante y ordenada.

Name	Type	Value
estado	int	1
se_mueve	int	0
score	int	23
posx	int [20]	[16, 0, 0, 0, 0,...
posy	int [20]	[24, 0, 0, 0, 0,...
Add new expression		

FIGURE 11: Estado 1 sin permiso de movilidad

IV. RESULTADOS IMPLEMENTACIÓN (0.1 PUNTOS)

La implementación del proyecto en la tarjeta funcionó correctamente. Se logra visualizar y jugar el juego del snake correctamente, ademas se logran visualizar todas las interrupciones y tambien la musica se escucha y se logra pausar como se espera. Se logro implementar correctamente cada etapa de la maquina estados.

Para validar el correcto funcionamiento del IP Core creado, se utiliza la ILA y el VIO. En las siguientes 3 imágenes, se puede ver el valor de la frecuencia enviados en distintos tiempos.

V. CONCLUSIONES(0.2)

- Las interrupciones de los timers funcion acorde a lo esperado. Por un lado se esperaba que la actualizacion del snake del timer 0, se active dependiendo del modo de dificultad escogido por el usuario, el cual se ajusta adecuadamente. Por otro lado, la interrupcion del segundo timer sirve para sumarle puntaje al usuario por cada segundo que persista en el juego. Esto funciona correctamente sumando puntaje cada 1 segundo.
- Los IPCores creados funcionan correctamente y como se espera. El uso del dividor de clock se configuro para que toque un tono distinto cada 0.3 segundos, lo cual ocurre correctamente y va circulando por los distintos tonos presentes en la ram del bloque de manera que suena la musica correctamente. Además, el uso del potenciometro permite controlar correctamente la pausa de la musica, de manera que cuando este se gira completamente antihorario y pasa el umbral de 900, la musica se para, por el contrario la musica se toca.
- Los delays ingresados funionan correctamente. Un ejemplo es en la pantalla de Game Over, en donde se configuro un delay que mantenga el programa en la pantalla de Game Over durante 3 segundos, para que luego pase a el display de los puntajes.
- Los top 5 puntajes funcionan correctamente, en donde se guardan exactamente 5 valores en la tajeta SD y tambien al final del juego se muestran estos top 5 puntajes.

VI. TRABAJOS FUTUROS (0.2)

El proyecto en cuestión tiene varias oportunidades de mejora, como lo son:

- Utilizar la interrupción del sensor de luz para ajustar el backlight del LCD dependiendo en el nivel de luz presente.
- Poder elegir distintas canciones de musica y no solo tener la misma todo el rato.
- Utilizar conexion a internet para poder conectarse a un servidor y competir contra puntajes de otras personas.
- Agregar distintos modos de juego, como por ejemplo que vayan apareciendo más de una manzana a la vez, o que la serpiente pueda pasar de lado a lado sin ningun problema.

REFERENCES

[1] G. O. Young, "Synthetic structure of industrial plastics," in Plastics, 2nd ed., vol. 3, J. Peters, Ed. New York, NY, USA: McGraw-Hill, 1964, pp. 15–64.
[2] W.-K. Chen, Linear Networks and Systems. Belmont, CA, USA: Wadsworth, 1993, pp. 123–135.
[3] J. U. Duncombe, "Infrared navigation—Part I: An assessment of feasibility," IEEE Trans. Electron Devices, vol. ED-11, no. 1, pp. 34–39, Jan. 1959, 10.1109/TED.2016.2628402.
[4] E. P. Wigner, "Theory of traveling-wave optical laser," Phys. Rev., vol. 134, pp. A635–A646, Dec. 1965.
[5] E. H. Miller, "A note on reflector arrays," IEEE Trans. Antennas Propagat., to be published.