Laboratorio 04
Lógica Secuencial y Controladores

Joseph David Jimenez Zuñiga, *2016133677, josephdjz@estudiantec.cr,*
Carlos Andrés Mata Calderón, *2019033834, carlos.andres12001@estudiantec.cr,*
Heizel Tatiana Chacón Mora, *2018203793, tati2327@estudiantec.cr*

Resumen—En este proyecto se presenta la implementación de un juego de lógica llamado 2048 en la placa de desarrollo DE1-SoC de Altera. El juego se desarrolló en el lenguaje de System Verilog utilizando una máquina de estados finitos para controlar el flujo del juego y un controlador VGA para mostrar la interfaz gráfica en un monitor. El objetivo del juego es combinar los bloques con números de igual valor para crear un bloque con el equivalente del resultado de la suma de los bloques que se combinaron, esto con el objetivo final de alcanzar el bloque 2048.
Se implementaron varias funcionalidades en el juego, como la capacidad de mover los bloques hacia la izquierda, derecha, arriba o abajo, y la capacidad de detectar cuándo el jugador ha ganado o perdido. El diseño se sintetizó y se simuló en el software Quartus y se verificó en la placa DE1-SoC. Se desarrolló una máquina de estados de seis estados para controlar el movimiento, y un controlador VGA con un sincronizador para generar la señal de vídeo que se muestra en una pantalla. Se logró una velocidad de reloj de 50 MHz y una resolución de pantalla de 640x480 píxeles con una frecuencia de refresco de 60 Hz.
El resultado final fue casi en su totalidad un juego funcional que se puede jugar en la placa de desarrollo, proporcionando una interfaz de usuario visual y una experiencia de juego satisfactoria. Este proyecto demuestra la viabilidad de implementar juegos en una placa de desarrollo FPGA y ofrece una base sólida para futuros proyectos de juegos en este tipo de plataforma.

Index Terms—Combinacional, FPGA, Maquina de Estados, Secuencial, System-Verilog, VGA, Quartus

1. Introducción

La lógica digital es un área fundamental en la electrónica y la informática, y se divide en dos categorías principales: lógica combinacional y lógica secuencial. La lógica combinacional se refiere a circuitos electrónicos que realizan operaciones matemáticas o lógicas en sus entradas y producen un resultado en la salida en función de estas entradas. Por otro lado, la lógica secuencial se basa en circuitos que contienen elementos de memoria, como flip-flops, que les permiten almacenar información y generar una salida en función de su estado actual y entradas. [1] En este contexto, es importante destacar que la lógica combinacional es más simple y eficiente que la lógica secuencial, ya que no tiene en cuenta el tiempo o el estado anterior de las entradas. Además, los circuitos combinacionales no tienen memoria interna, lo que significa que su salida solo depende de las entradas en un momento dado. Esto hace que sean ideales para una variedad de aplicaciones, como el procesamiento de señales, la aritmética y la lógica booleana.

En el diseño y desarrollo de sistemas digitales, los testbench son herramientas fundamentales para verificar el correcto funcionamiento del hardware y software. Por lo tanto, es importante contar con testbench adecuados que cubran una amplia variedad de situaciones para garantizar la robustez y confiabilidad del sistema. En el caso de la lógica combinacional, los testbench pueden ser más sencillos que en la lógica secuencial, debido a la naturaleza de los circuitos, pero siempre es importante contar con pruebas rigurosas para asegurarse de que el sistema funcione correctamente.

La aplicación a implementar en este laboratorio es un juego digital basado en el popular juego 2048. El objetivo del juego es combinar dos bloques del mismo número para crear un bloque con un número mayor. El jugador comienza con dos bloques de valor 2 en una matriz 4x4 y debe mover los bloques hacia arriba, abajo, izquierda o derecha para combinarlos y crear bloques con números más altos. El objetivo final es crear un bloque con el valor de 2048, aunque el usuario también puede elegir un valor más bajo como objetivo.

La mecánica del juego implica la generación aleatoria de bloques de valor 2 o 4 en la matriz, la suma de bloques del mismo valor cuando se mueven en una dirección y la actualización de la puntuación del jugador. Si el jugador llega a la meta establecida, se mostrará un mensaje de victoria en la pantalla. Sin embargo, si ya no existen movimientos posibles, se mostrará un mensaje de derrota en la pantalla.

Para la visualización del juego, se utilizará un controlador VGA que se encargará de generar las señales de sincronización y las señales de color por píxel. El controlador VGA debe diseñarse para una resolución de 640x480 píxeles y se integrará en el controlador principal de la aplicación. El controlador principal se implementará utilizando una máquina de estados finita que permitirá la interacción del usuario con el juego a través de los botones. Además, se mostrará el puntaje del jugador en un display de 7 segmentos.

En resumen, este laboratorio se enfoca en el diseño de un juego 2048 utilizando conceptos de diseño de circuitos digi-

- *Instituto Tecnológico de Costa Rica, Área Académica de Ingeniería en Computadores, CE3201 Taller de diseño digital.*
- *Profesor. Luis Alberto Chavarría Zamora.*

tales. La combinación de los controladores VGA y principal permitirá la visualización y el control del juego, mientras que el diseño de la máquina de estados finita permitirá la interacción del usuario con el juego.

2. SISTEMAS DESARROLLADOS

2.1. Máquina de Estados del Juego 2048

La lógica del juego (módulo game2048) puede describirse eficazmente mediante una máquina de estados, que es un modelo de comportamiento compuesto por estados, transiciones y acciones. En el caso de 2048, la máquina de estados se puede clasificar como una máquina de Mealy, ya que la salida o las acciones realizadas dependen tanto del estado actual como de la entrada (la dirección de desplazamiento elegida por el jugador). [2]

2.1.1. Funcionamiento de la Máquina de Estados

La máquina de estados del juego 2048 se puede describir como sigue:

- **Estado iniciales:** Al inicio del juego, la cuadrícula está casi vacía, ya que solo se generan dos celdas con los números 2 o 4. Este es el estado inicial de la máquina de estados.
- **Estados intermedios:** Cada vez que el jugador realiza un movimiento, el estado de la cuadrícula cambia. Cada posible disposición de la cuadrícula puede considerarse como un estado intermedio.
- **Estado finales:** Si se alcanza el número 2048 en alguna celda, el juego se gana y se llega a un estado final. Alternativamente, si no hay movimientos posibles y la cuadrícula está llena, el juego se pierde, y esto también es un estado final.

2.1.2. Transiciones y Acciones

Las transiciones entre estos estados dependen de las acciones del jugador. Cuando el jugador elige desplazar las celdas en una dirección (arriba, abajo, izquierda o derecha), se realiza una acción que cambia el estado de la cuadrícula. Si este movimiento resulta en el número 2048, se llega a un estado final de victoria. Si el movimiento no puede hacerse porque las celdas no pueden desplazarse en la dirección elegida, el estado de la cuadrícula no cambia.

2.1.3. Submódulos

La máquina de estados del juego 2048 consta de varios submódulos importantes, que incluyen:

- **Manejador de acciones (move and merge tiles):** Este submódulo recibe la entrada del jugador (la dirección de desplazamiento) y realiza la acción correspondiente, es decir, desplaza las celdas en la dirección elegida y combina las celdas adyacentes si tienen el mismo número. Además obtiene el sumando de los números de todas las celdas combinadas durante el juego.
- **Comprobador de espacio en la matriz (can new tile)** Este submódulo se encarga de revisar si existe un 0 en la matriz del juego para agregar una celda en dentro la matriz.
- **Generador de celdas (place random tile):** Este submódulo es responsable de generar una celda con un número 2 o 4 en una posición aleatoria de la cuadrícula vacía después de cada acción del jugador. Para dicha genereacion aleatoria se utiliza otra máquina de estados.
- **Comprobador ganar (check win):** Este submódulo verifica si se ha alcanzado un estado final, es decir, si se ha ganado el juego (se ha formado un número 2048) .
- **Comprobador pierde (check lose):** Este submódulo verifica si se ha alcanzado un estado final, es decir, si se ha perdido el juego (no hay movimientos posibles y la cuadrícula está llena) .

Cada uno de estos submódulos desempeña un papel crucial en la lógica del juego y en la definición del funcionamiento de la máquina de estados.

2.1.4. Referencia a los Diagramas de Estados

Para comprender mejor la lógica del juego 2048 y cómo se puede representar mediante una máquina de estados, se recomienda referirse al Diagrama de Estados del Juego 2048 proporcionado. Este diagrama (Figura 1) muestra todos los posibles estados y transiciones en el juego, lo que puede ayudar a visualizar y entender mejor la dinámica del juego.

Se recomienda también referirse al Diagrama de Estados del Generador de celdas (Figura 2). Para entender la generación de celdas aleatorias en el juego.

2.2. Controlador VGA

El controlador de VGA es un componente fundamental del sistema desarrollado en el laboratorio. Su función es generar las señales de vídeo necesarias para mostrar el juego en la pantalla del monitor, para lo cual debe ser capaz de generar las señales de sincronización horizontal y vertical, así como las señales de color por píxel correspondientes, adaptadas a la resolución de 640x480 píxeles para la que se ha diseñado un flujo que se puede visualizar en la Figura 4.

Con este fin, se ha desarrollado el módulo Synchronizer_VGA, que se vale de varias señales de entrada, como HACTIVE, HFP, HSYN, HBP, VACTIVE, VFP, VSYN y VBP, y produce señales de salida como SYNC_H, SYNC_V, SYNC_B, SYNC_BLANK, CLK_VGA, x e y. Estas últimas permiten identificar la posición de cada píxel en el momento actual, lo que resulta fundamental para dibujar en el área virtual correspondiente y lograr así la sincronización perfecta de las dimensiones del monitor. [1]

Uno de los elementos vitales para el correcto funcionamiento de este módulo es el CLK_VGA. Para controlar esta entrada se ha creado el módulo pll, que recibe el CLK de la FPGA a 50MHz como entrada y proporciona como salida el clk necesario para que la frecuencia que recorre la pantalla sea la adecuada. [3]

Además, se ha diseñado el módulo Paint_VGA, que recibe como entradas game_state, matrix_table, x e y y proporciona como salidas los valores Red, Green y Blue. Estas entradas indican qué figuras deben dibujarse en pantalla en función de ciertas condiciones, como el estado del juego o la posición de las figuras. Los valores de Red, Green y

Blue corresponden al color por píxel que se mostrará en la pantalla del monitor.

Para sincronizar todas las instancias anteriores con el monitor y mostrar las señales de video de manera exitosa, se utiliza el módulo Controller_VGA. Este recibe como entrada la matriz lógica del juego que se debe dibujar y produce como salida la señal RGB necesaria para llevar a cabo la sincronización. Todos estos elementos conforman un controlador de VGA completo y funcional para el sistema de juego 2048. La Figura **??** muestra el módulo controlador de VGA.

2.3. Modelo del juego 2048

El módulo ”game2048.[es] una parte crucial del sistema del juego desarrollado en el laboratorio, ya que se encarga de controlar el estado del juego y la lógica de su funcionamiento. Este módulo recibe varias señales de entrada, como el reloj çlkz la señal de reinicio rst”, que le permiten operar en el tiempo correcto y reiniciarse cuando sea necesario. Además, recibe información importante sobre el número máximo de victorias, la dirección del movimiento y el estado actual del juego. Utilizando esta información, el módulo ”game2048”genera la matriz lógica del juego, la puntuación y el estado del juego, que se utilizan en conjunto con el controlador de VGA para mostrar el juego en la pantalla del monitor, este interaccion se visualiza en la Figura 5. De esta manera, el módulo ”game2048z el controlador de VGA trabajan en conjunto para crear una experiencia de juego completa y satisfactoria.

3. TESTBENCH DE ALGUNOS MÓDULOS

En esta sección, se describirán brevemente los testbenches utilizados para verificar el correcto funcionamiento de los diferentes submódulos del sistema del juego 2048. Cada testbench tiene su propia subsección donde se discutirán las pruebas realizadas y los resultados obtenidos.

3.1. Testbench del módulo game2048

El testbench para el módulo ”game2048”prueba la operación integral del juego. Este módulo es el encargado de controlar el estado del juego y la lógica de su funcionamiento. Por lo tanto, este testbench es esencial para verificar que todas las partes del juego estén funcionando juntas correctamente. (Figura 6 y Figura 7)

3.2. Testbench del Manejador de acciones (move and merge tiles)

Este testbench verifica el correcto funcionamiento del submódulo ”move and merge tiles”, que es responsable de desplazar y combinar las celdas en la dirección elegida por el jugador. Se simula una serie de movimientos en diferentes direcciones y se verifica si las celdas se desplazan y combinan correctamente y si se obtiene la suma adecuada de los números de las celdas combinadas. (Figura 8 y Figura 9)

3.3. Testbench del Comprobador de espacio en la matriz (can new tile)

En este testbench, se examina el submódulo çan new tile”para asegurarse de que identifica correctamente si hay espacio disponible en la matriz del juego para agregar una nueva celda. Se prueban diferentes configuraciones de la matriz, incluidas las matrices vacías, parcialmente llenas y completamente llenas, y se verifica si el submódulo devuelve la información correcta sobre la disponibilidad de espacio en la matriz. (Figura 10 y Figura 11)

3.4. Testbench del Generador de celdas (place random tile)

El testbench para el submódulo ”place random tileçomprueba si el submódulo es capaz de generar celdas con números 2 o 4 en posiciones aleatorias de la cuadrícula vacía después de cada acción del jugador. Se simulan diferentes escenarios de juego y se verifica si las celdas generadas se colocan correctamente en la cuadrícula. (Figura 12 y Figura 13)

3.5. Testbench del Comprobador ganar (check win)

Este testbench examina el submódulo çheck win”para asegurarse de que detecta correctamente si el jugador ha ganado el juego al formar un número 2048 en la matriz. Se prueban diferentes configuraciones de la matriz que contienen el número 2048 y se verifica si el submódulo identifica correctamente el estado ganador. (Figura 14 y Figura 15)

3.6. Testbench del Comprobador pierde (check lose)

Por último, el testbench del submódulo çheck lose”verifica si el submódulo es capaz de identificar correctamente si el jugador ha perdido el juego, es decir, si no hay movimientos posibles y la cuadrícula está llena. Se simulan diferentes escenarios de juego en los que la cuadrícula está completamente llena y no hay movimientos posibles, y se verifica si el submódulo detecta correctamente el estado de pérdida. (Figura 16 y Figura 17)

4. Resultados

El juego implementado cuenta con varios inputs y outputs para una interacción con el juego. En cuanto a los inputs, se encuentran los cuatro botones que permiten mover los números en la cuadrícula hacia la izquierda, derecha, arriba y abajo respectivamente. Además, se incluyen switches para elegir el punto final del juego, que son las elevaciones del número dos. Por último, se cuenta con un display donde se muestra la puntuación del jugador en tiempo real. Estas entradas y salidas permiten al usuario tener un control total del juego y tomar decisiones estratégicas en función de su objetivo final. los cuales pueden ser visualizados en la Figura 18. Por otro lado, la pantalla VGA de 640x480 píxeles permite una visualización clara y detallada de la cuadrícula del juego y su progreso.

La Figura 19 muestra la interfaz final del juego 2048 implementado en la FPGA. En la pantalla VGA de 640x480 píxeles se muestra la cuadrícula del juego en la parte derecha de la pantalla, donde se desarrolla el juego. En este caso, se visualizan todas las fichas disponibles como prueba. Los bloques de números se disponen en la cuadrícula a medida que el juego avanza. En la parte izquierda de la pantalla, se muestra un cuadro que permite al usuario saber si ha ganado, perdido o si puede continuar jugando. La pantalla es clara y fácil de leer, lo que proporciona una experiencia de juego fluida y agradable.

5. Análisis de resultados

El juego se ha programado utilizando una arquitectura de máquina de estados que consta de seis estados diferentes, lo que permite una transición suave entre los estados dependiendo de la entrada del usuario. Además, se ha diseñado un controlador de VGA para proporcionar una experiencia visual agradable al usuario.

En términos de rendimiento, el juego ha demostrado ser capaz de manejar y ejecutar las operaciones requeridas sin ningún problema importante. Sin embargo, se identificó un error en el programa que afectaba el registro de las jugadas cuando se presionaban los botones, lo que provocaba una sobrecarga en el sistema y causaba que el juego se colgara. Esto se podría solucionar mediante la revisión y el ajuste del código para garantizar que solo se registrara una jugada por cada pulsación del botón, pero por problemas de tiempo no ha sido posible su implementación.

En cuanto a la interacción con el usuario, se ha diseñado una interfaz fácil de usar que permite a los usuarios comprender y jugar el juego sin dificultad. La combinación de colores y el diseño de las fichas se han elegido cuidadosamente para proporcionar una experiencia visual atractiva y agradable. Además, el juego una pantalla de visualización de finalización junto con una puntuación que se actualiza en tiempo real en el display para mejorar la experiencia del usuario.

6. Conclusión

La máquina de estados resultó ser una herramienta fundamental en el diseño del juego 2048, permitiendo la implementación de la lógica necesaria para el correcto funcionamiento del mismo. Gracias a la maquina de estados se pudo definir de manera clara y ordenada el comportamiento del juego, controlando los diferentes estados y transiciones entre ellos. La implementación exitosa de la máquina de estados fue clave para lograr un juego funcional y estable, lo que demuestra su importancia en la creación de sistemas digitales complejos.

La implementación del controlador VGA resultó ser una parte fundamental en la visualización del juego. El uso de las señales SYNC_H, SYNC_V, SYNC_B y SYNC_BLANK permitió la generación de una señal VGA correcta y la sincronización adecuada con la pantalla. Además, la resolución de 640x480 píxeles y la tasa de refresco de 60 Hz lograron una imagen fluida y clara del juego. La utilización de un sincronizador adecuado y la configuración de los parámetros de tiempo también fueron esenciales para asegurar la correcta visualización en la pantalla.

El proyecto del juego 2048 ha sido un éxito en términos de su implementación, diseño visual y rendimiento. A pesar del problema identificado con el registro de jugadas, con la solución de este problema se podría garantizar que el juego sea completamente funcional y satisfactorio para el usuario. La combinación de la arquitectura de la máquina de estados y el controlador de VGA ha permitido una experiencia visual agradable y suave para el usuario. En general, el juego 2048 es un proyecto satisfactorio y exitoso que cumple con los objetivos establecidos inicialmente.

REFERENCIAS

[1] S. Harris and D. Harris, *Digital Design and Computer Architecture: ARM Edition*. San Francisco, CA: Morgan Kaufmann Publishers Inc., 2015. ISBN: 978-0-12-800056-4.
[2] D. A. Patterson and J. L. Hennessy, *Computer Organization and Design: The Hardware/Software Interface*. Amsterdam, Netherlands: Elsevier, 2014. ISBN: 978-0-12-407726-3.
[3] P. Horowitz and W. Hill, *The Art of Electronics*. Cambridge, United Kingdom: Cambridge University Press, 2015. ISBN: 978-0-521-80926-9.

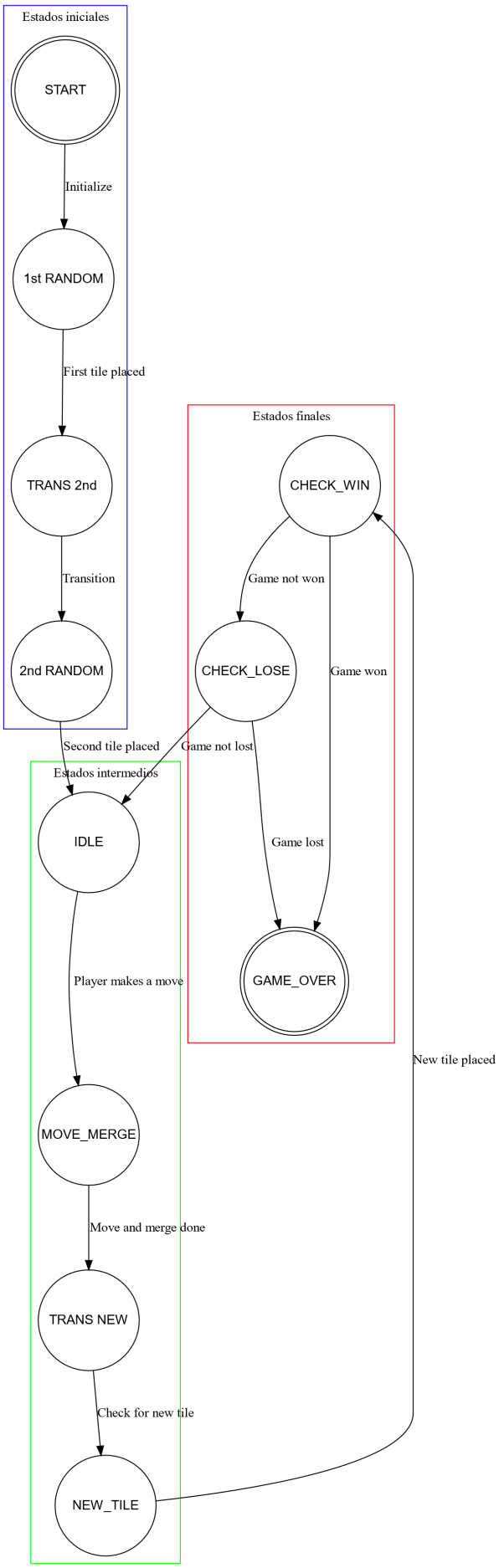

Figura 1. Diagrama de Estados de game2048

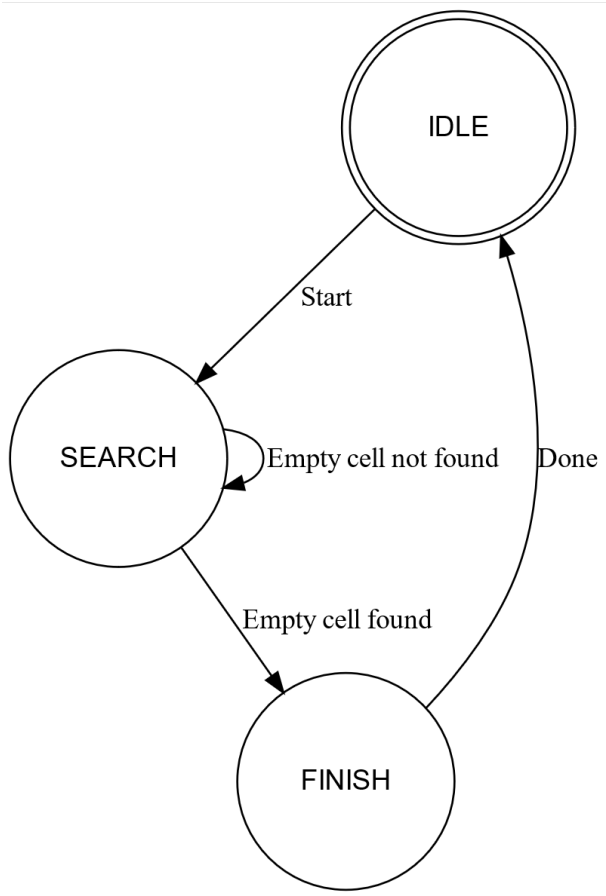

Figura 2. Diagrama de Estados de Generador de celdas

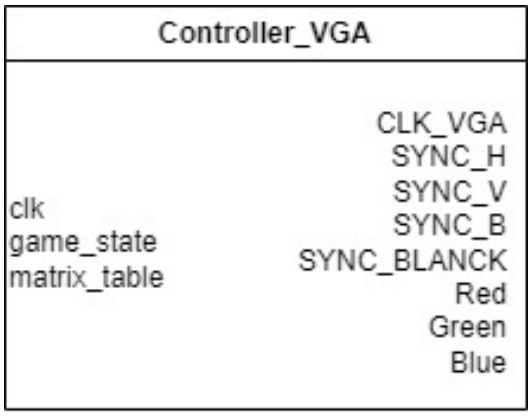

Figura 4. Controlador VGA

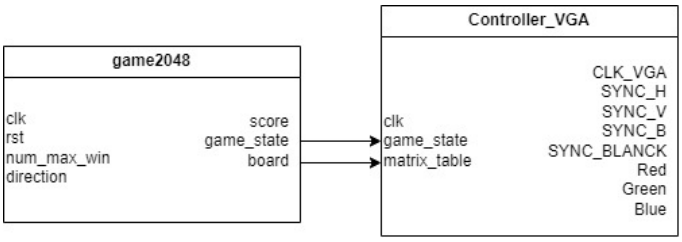

Figura 5. Juego 2048

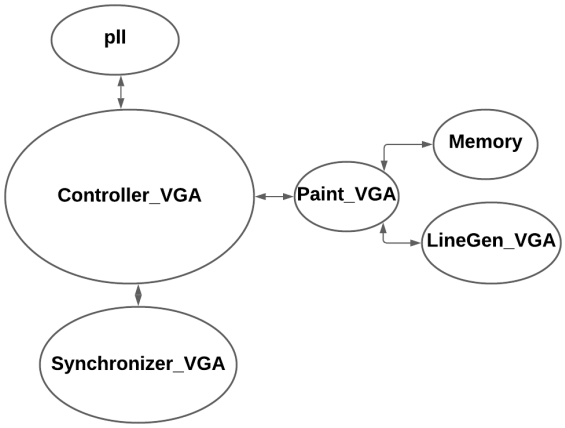

Figura 3. Flujo de Controlador VGA

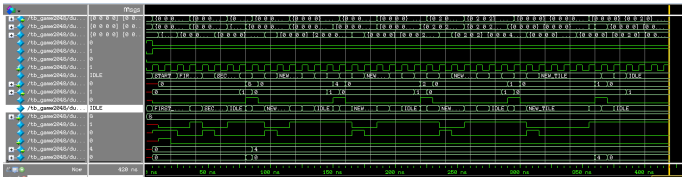

Figura 6. Señales Comprobador del juego

```
VSIM 22> run
# Initial board:
#     0      0      2      2
#
#     0      0      0      0
#
#     0      0      0      0
#
#     0      0      0      0
#
#
#
# Board after move right:
#     0      0      0      4
#
#     0      0      0      0
#
#     0      0      0      2
#
#     0      0      0      0
#
#
#
# Board after move left:
#     4      0      2      0
#
#     0      0      0      0
#
#     2      0      0      0
#
#     0      0      0      0
#
#
#
# Board after move bottom:
#     0      0      0      0
#
#     0      2      0      0
#
#     4      0      0      0
#
#     2      0      2      0
#
#
#
# Board after move top:
#     4      2      2      0
#
#     2      0      0      0
#
#     0      0      0      0
#
#     0      0      0      0
#
#
#
# Board after move top:
#     4      2      2      0
#
#     2      0      0      0
#
#     0      2      0      0
#
#     0      0      0      0
#
#
#
# ** Note: $finish    : C:/Users/carlo/Desktop/random/jjimen
#    Time: 420 ns  Iteration: 0  Instance: /tb_game2048
```

Figura 7. Consola Comprobador del juego

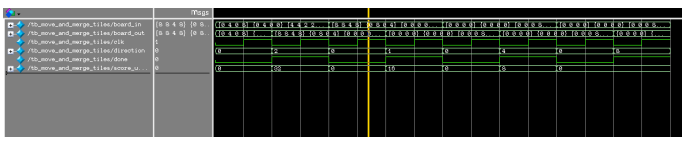

Figura 8. Señales Comprobador mover fichas

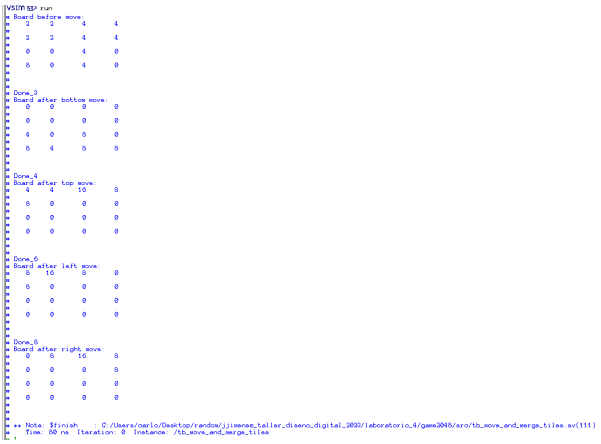

Figura 9. Consola Comprobador mover fichas

Figura 10. Señales Comprobador espacio matriz

```
# Break in Module tb_move_and_merge_tiles at C:/Users/carlo/Desktop/random
VSIM 54> vsim work.tb_can_new_tile
# End time: 22:58:13 on May 15,2023, Elapsed time: 0:03:21
# Errors: 0, Warnings: 0
# vsim work.tb_can_new_tile
# Start time: 22:58:13 on May 15,2023
# Loading sv_std.std
# Loading work.tb_can_new_tile
# Loading work.can_new_tile
```

Figura 11. Consola Comprobador espacio matriz

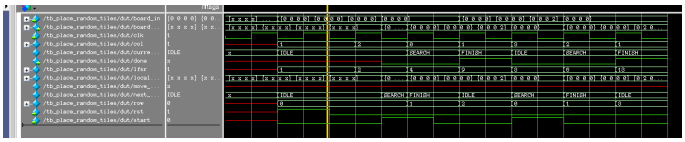

Figura 12. Señales Comprobador generador celdas

Figura 13. Consola Comprobador generador celdas

Figura 14. Señales Comprobador ganar

Figura 15. Consola Comprobador gane

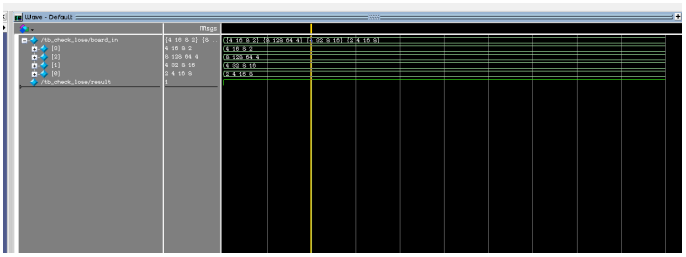

Figura 16. Señales Comprobador pierde

Figura 17. Consola Comprobador pierde

Figura 18. Inputs y outputs del Juego 2048

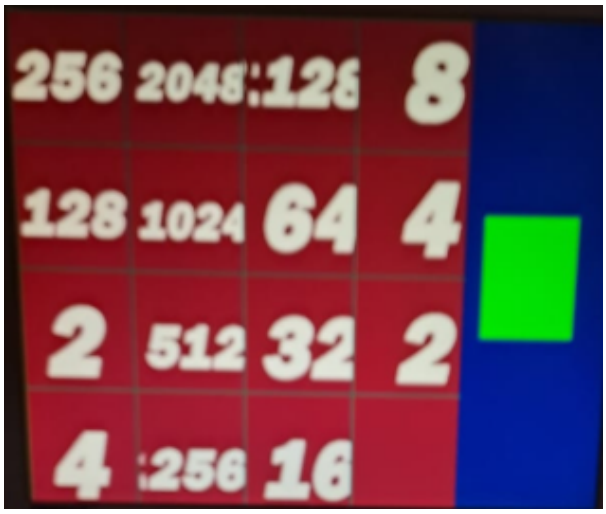

Figura 19. Visualización Juego 2048