Laboratorio 3: Diseño Modular Controlador VGA

Kenneth Hernández, Marco Herrera, and Jasson Rodríguez

Ingeniería en Computadores, Instituto Tecnológico de Costa Rica

Setiembre 2019

1. Comprensión del problema

El problema en desarrollo consiste en el manejo de la lógica de un juego simple como tic tac toe y su interfaz con el usuario, lo que implica la sincronización de las señales que forman parte de la interfaz analógica VGA, pues requiere del manejo de tiempos muy precisos para su correcto funcionamiento.

2. Investigación

1. Investigue sobre el funcionamiento de máquinas de estado finitos. Explique la diferencia entre una máquina de Moore y una de Mealy y muestre la diferencia por medio de diagramas de estados y señales.

   Es una forma de modelar comportamiento, esta contiene estados que son intermediarios entre las entradas y las salidas y representan como su nombre lo dice el estado actual. Máquina de Mealy es un tipo de máquina de estados finitos que genera una salida basándose en su estado actual y sus entradas mientras que Moore solo depende de su estado actual

2. Explique los conceptos de *setup time* y *hold time*. ¿Qué importancia tienen en el diseño de sistemas digitales?

   Setup time es la cantidad de tiempo antes de un ciclo de reloj en la que la entrada debe ser estable para que sea capturada correctamente por un flip flop Hold time es la cantidad de tiempo después de un ciclo de reloj en el que la entrada debe permanecer estable para que sea registrada correctamente en un flip flop. En flip flops modernos es cero o negativo.

3. Investigue sobre el efecto de rebote en señales digitales provenientes de elementos mecánicos (interruptores, por ejemplo). Muestre al menos dos formas de solucionar el efecto de rebote, por medio de circuitos digitales. Al tratarse con botones mecánicos suele ocurrir un fenómeno conocido como rebote de señales digitales el cual significa en muchas ocasiones una detección incorrecta de señales de entrada cuando no las hay. Se detectan múltiples entradas cuando esta es única. Una forma de contrarrestar este efecto es mediante el siguiente circuito, ya que los flip flops conectados en la entrada generan el timing junto a la compuerta XOR, ya que estos se encontraran en un valor exacto y los transmitirán en la salida solo si se mantienen estables durante un tiempo especifico.

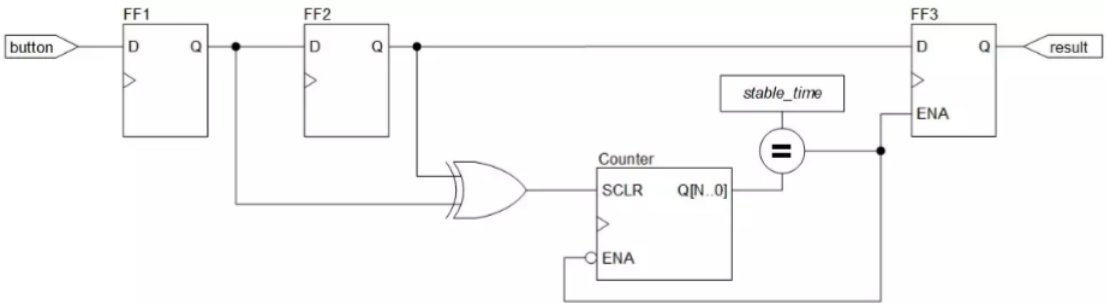

Figura 1: Circuito propuesto para contrarrestar el efecto rebote

4. Investigue sobre las señales involucradas en la sincronización de una interfaz VGA.

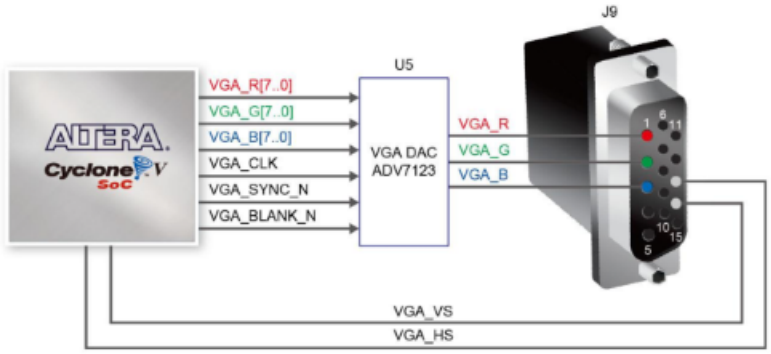

Figure 3-21 VGA Connections between FPGA and VGA

Figura 2: Señales de sincronización VGA (parte 1)

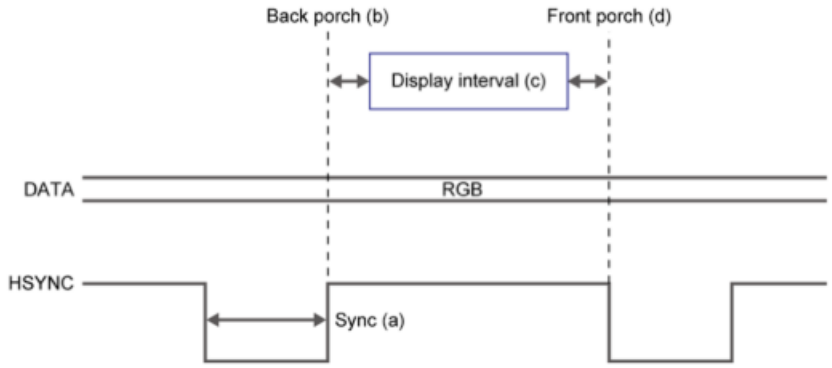

Figure 3-22 VGA horizontal timing specification

Table 3-14 VGA Horizontal Timing Specification

VGA mode		*Horizontal Timing Spec*				
Configuration	*Resolution(HxV)*	*a(us)*	*b(us)*	*c(us)*	*d(us)*	*Pixel clock(MHz)*
VGA(60Hz)	640x480	3.8	1.9	25.4	0.6	25
VGA(85Hz)	640x480	1.6	2.2	17.8	1.6	36
SVGA(60Hz)	800x600	3.2	2.2	20	1	40
SVGA(75Hz)	800x600	1.6	3.2	16.2	0.3	49
SVGA(85Hz)	800x600	1.1	2.7	14.2	0.6	56
XGA(60Hz)	1024x768	2.1	2.5	15.8	0.4	65
XGA(70Hz)	1024x768	1.8	1.9	13.7	0.3	75
XGA(85Hz)	1024x768	1.0	2.2	10.8	0.5	95
1280x1024(60Hz)	1280x1024	1.0	2.3	11.9	0.4	108

Table 3-15 VGA Vertical Timing Specification

VGA mode		*Vertical Timing Spec*				
Configuration	*Resolution(HxV)*	*a(lines)*	*b(lines)*	*c(lines)*	*d(lines)*	*Pixel clock(MHz)*
VGA(60Hz)	640x480	2	33	480	10	25
VGA(85Hz)	640x480	3	25	480	1	36
SVGA(60Hz)	800x600	4	23	600	1	40
SVGA(75Hz)	800x600	3	21	600	1	49
SVGA(85Hz)	800x600	3	27	600	1	56
XGA(60Hz)	1024x768	6	29	768	3	65
XGA(70Hz)	1024x768	6	29	768	3	75
XGA(85Hz)	1024x768	3	36	768	1	95
1280x1024(60Hz)	1280x1024	3	38	1024	1	108

Figura 3: Señales de sincronización VGA (parte 2)

5. Muestre un diagrama de tiempos de las señales de sincronización de VGA para una resolución de 640x480 píxeles.

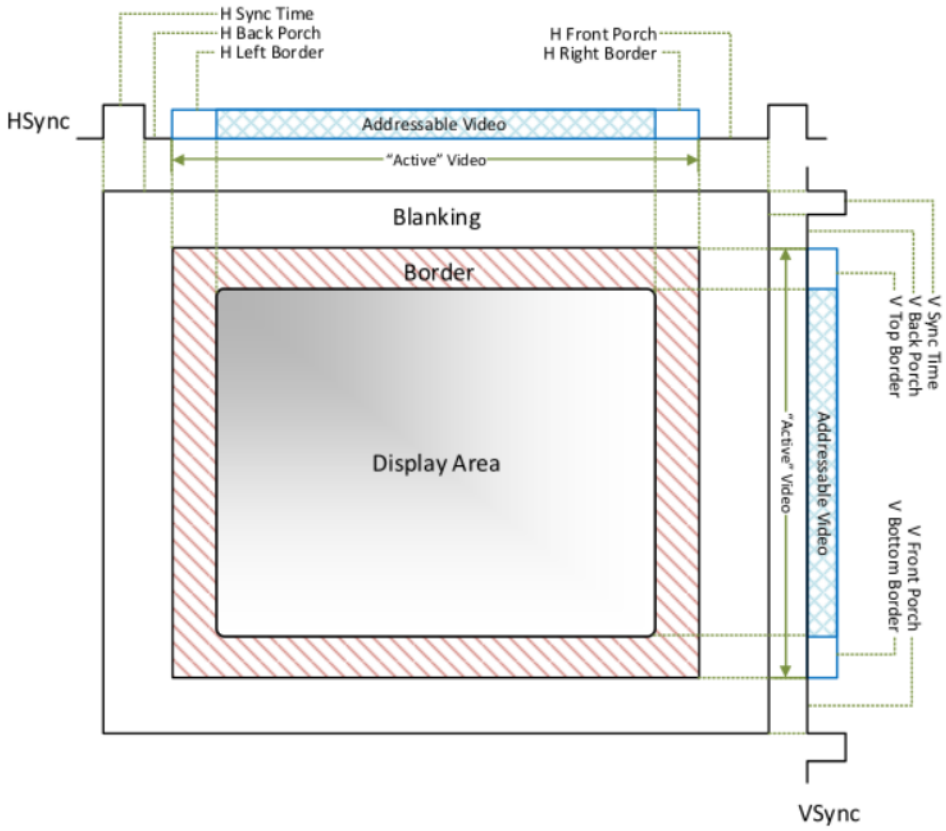

Figura 4: Diagrama de tiempos de sincronización VGA

6. Para la resolución del punto anterior, calcule matemáticamente la frecuencia aproximada de las señales de sincronización vertical y horizontal.

   A partir de las diversas secciones que componen la señal, conocidas como *back porch*, *front porch*, *display interval*, *Sync*. Y conociendo su tamaño en píxeles y la frecuencia del reloj de píxeles, se puede calcular el tiempo que debe estar activo cada una de estas señales y calcular la frecuencia de la señal de sincronización horizontal y vertical. A partir de la siguiente figura se pueden calcular los valores de frecuencia de la tabla #.

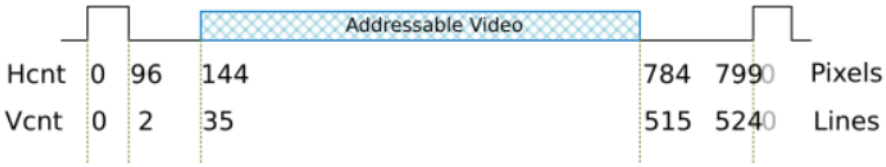

Figura 5: Señales de sincronización horizontal y vertical VGA

7. Proponga un diagrama de bloques que implemente el controlador de VGA. Tenga en cuenta que este será parte de su diseño final, utilizando un modelado de estructura. El controlador cuenta con dos señales de sincronización, una horizontal y otra vertical; estas señales deben activarse en un momento preciso de forma periódica; esto se logra mediante contadores y comparadores. Un contador indica la columna del píxel y si el comparador determina que se encuentra en el valor que debe reiniciar la fila, se genera la señal de sincronización, de forma análoga se genera la señal vertical. Si el contador indica que la fila y la columna están en el rango visible, se envían las señales RGB del píxel. A continuación se muestra una propuesta de implementación por la compañía Diligent.

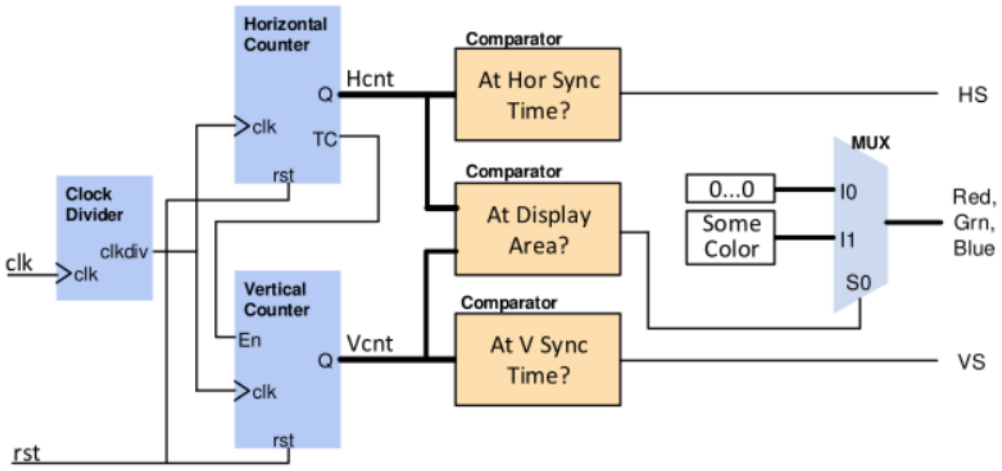

Figura 6: Digrama de bloques de propuesta para un controlador VGA

8. Investigue sobre la técnica de sprites y su aplicación.

   Los sprites son objetos gráficos que pueden ser localizados en cualquier parte de la pantalla, la técnica encontrada es la llamada double buffering, esta técnica tiene un buffer que esta manejando la conexión con la pantalla mientras que el otro buffer carga los sprites necesarios, solo que se necesita el doble de memoria.

3. Estimación de la solución

A partir de un lenguaje de descripción de hardware, describir e implementar una versión funcional del juego tic tac toe y realizar una interfaz de este con un monitor a través de un controlador de VGA.

4. Objetivos de la solución

- Implementar una versión funcional del juego tic-tac-toe.
- Diseñar e implementar un controlador funcional de VGA que cumpla con los requerimientos estándar.
- Utilizar la menor cantidad posible de unidades básicas de la FPGA.

5. Diseño del hardware del sistema

A continuación se presenta una propuesta para la implementación de la solución, se clasifican según su nivel de detalles. Además contienen una breve explicación de cada una de sus entradas y salidas; junto con su funcionamiento general.

5.1. Primer Nivel

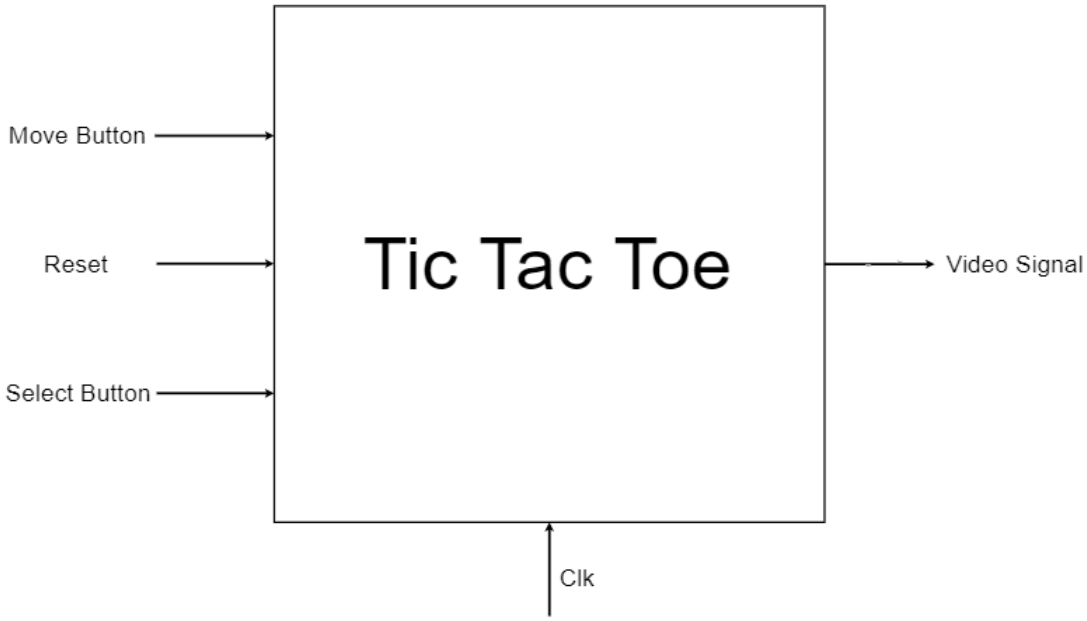

Figura 7: Diagrama de primer Nivel

Objetivo: Juego de Tic Tac Toe capaz de ser controlado mediante dos botones y con salida para un monitor.

Entradas:

- Move Button: Botón utilizado para alternar entre las diferentes posiciones del tablero
- Select Button: Botón utilizado para seleccionar la posición actual del tablero

Salidas

- Video Signal: Salida de video tipo VGA para ser mostrada en un monitor con resolución 640x480 pixeles.

Explicación General: Este modulo representa la funcionalidad total del juego, los dos botones son utilizados de acuerdo a los turnos de cada jugador para seleccionar la siguiente casilla a jugar. Toda la interfaz del juego y los resultados serán mostrados en un monitor conectado a la salida de video del sistema.

5.2. Segundo Nivel

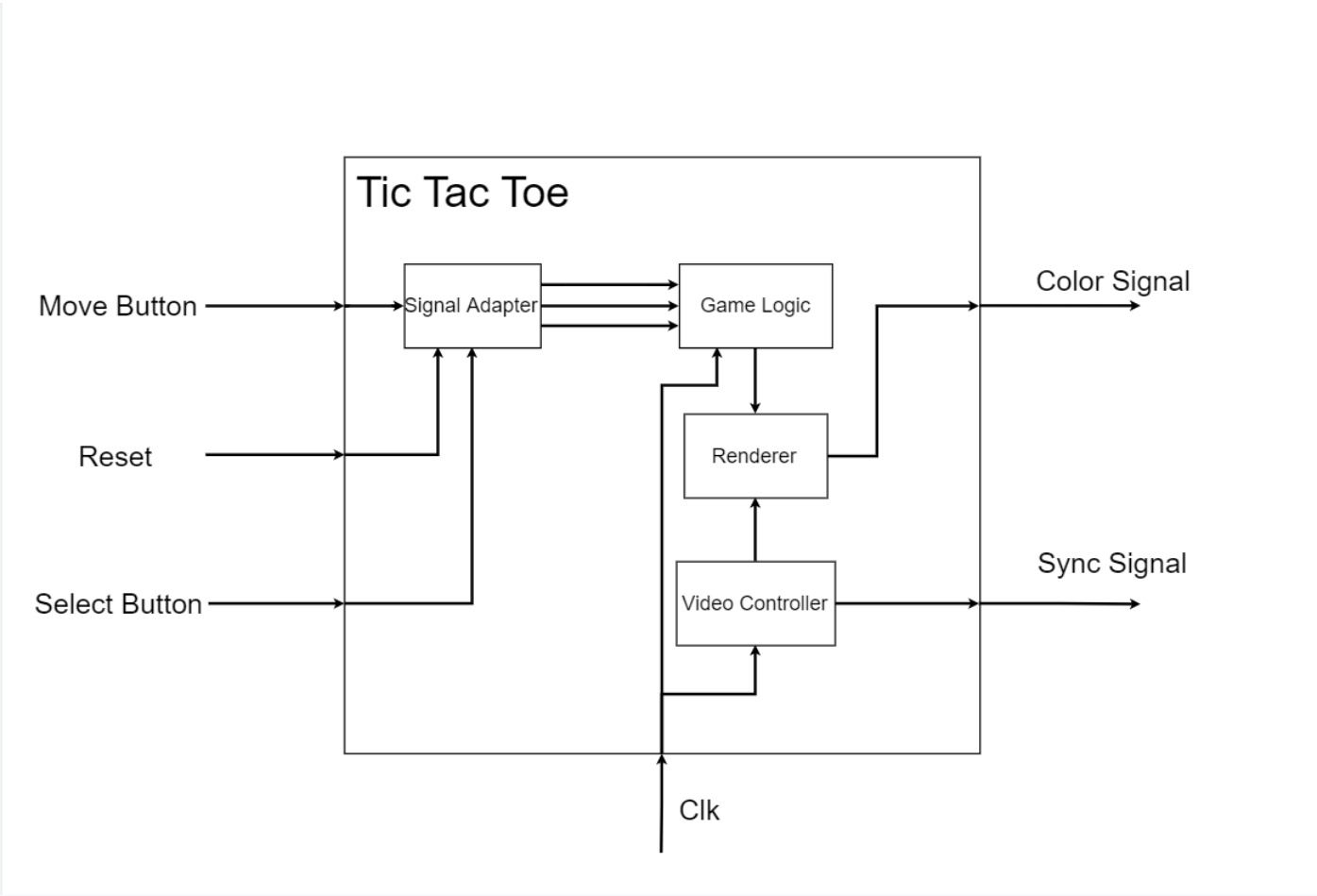

Figura 8: Diagrama de segundo Nivel

5.2.1. Signal Adapter

Objetivo: Acoplar las señales de entrada para que puedan ser utilizadas por el sistema sin detectar selecciones dobles o falsas.

Entradas:

- Move Button: valor lógico controlado por un botón que indica que se debe pasar a la siguiente casilla.
- Select Button: valor lógico controlado por un botón que indica que se debe seleccionar la casilla actual.
- Reset: valor lógico controlado por un botón que indica que se iniciará una nueva partida del juego.

Salidas:

- Move Button: valor lógico estabilizado que indica que se debe pasar a la siguiente casilla.
- Select Button: valor lógico estabilizado que indica que se debe seleccionar la casilla actual.
- Reset: valor lógico estabilizado que indica que se iniciará una nueva partida del juego.

Explicación General: Este módulo representa un acople de las señales provenientes de los botones para su procesamiento, pues al tratarse de botones mecánicos suele ocurrir un fenómeno conocido como rebote el cual

significa en muchas ocasiones una detección incorrecta de señales de entrada cuando no las hay. Este módulo se encarga de contrarrestar este proceso y generar una única señal estable ante la entrada. La salida de este módulo es entrada de la lógica del juego y esencial para su correcto funcionamiento.

5.2.2. Game Logic

Objetivo: Manejar las entradas y los turnos de cada uno de los jugadores.
Entradas:

- Move Button: Botón utilizado para alternar entre las diferentes posiciones del tablero
- Select Button: Botón utilizado para seleccionar la posición actual del tablero

Salidas

- Información relevante para posteriormente dibujar el estado del juego en pantalla.

Explicación General: Este módulo recibe las entradas del jugador actual mediante las cuales se selecciona una casilla para jugar y verifica si la casilla es válida. Después de cada jugada revisa si existe un ganador o avanza al siguiente turno. Además, debe pasar la información necesaria al renderer para poder dibujar el juego en pantalla.

5.2.3. Renderer

Objetivo: Transformar los datos del juego en la imagen que se va a ver en pantalla.
Entradas:

- Estado del juego: Estado actual del juego como el valor de las casillas y un indicador de resultado.

Salidas

- Imagen a color del estado actual del juego.

Explicación General: Este módulo procesa el estado actual del juego de acuerdo a la lógica del juego y genera la imagen que se va enviar al controlador de video para ser mostrada en la pantalla. Además, debe contener los gráficos necesarios para dibujar el juego.

5.2.4. Video Controller

Objetivo: Enviar todos los datos necesarios para dibujar la imagen proporcionada por el renderer en un monitor.
Entradas:

- Imagen que se desea dibujar.

Salidas

- Señal de video capaz de ser entendida por un monitor.

Explicación General: Este módulo es el encargado de tomar la imagen generada por el renderer y sincronizarla pixel por pixel con el monitor a través de un cable de video.

5.3. Tercer Nivel

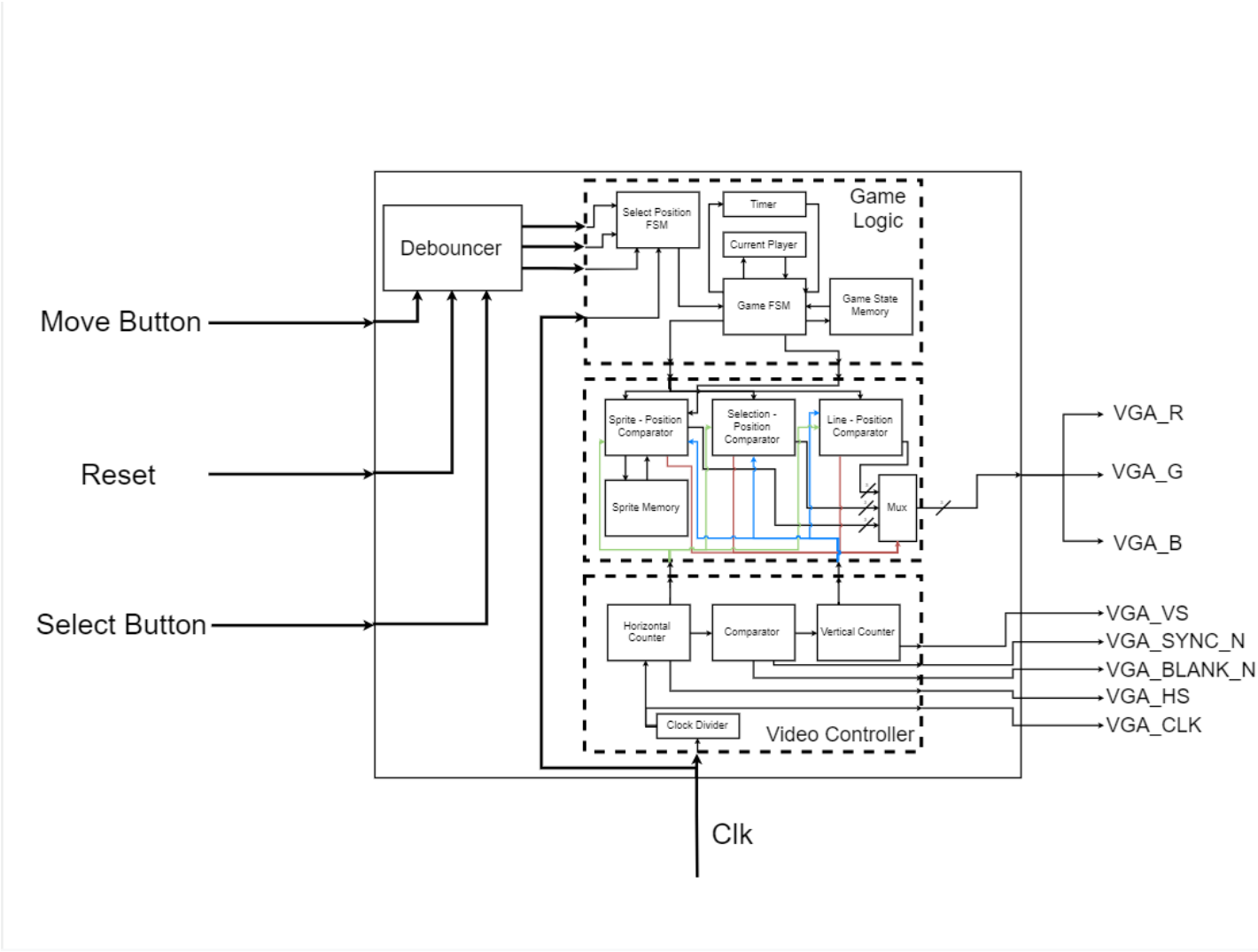

Figura 9: Diagrama de tercer Nivel ALU

5.3.1. Debouncer

Objetivo: Acoplar las señales de entrada para que puedan ser utilizadas por el sistema sin detectar selecciones dobles o falsas.

Entradas:

- Move Button: valor lógico controlado por un botón que indica que se debe pasar a la siguiente casilla.
- Select Button: valor lógico controlado por un botón que indica que se debe seleccionar la casilla actual.
- Reset: valor lógico controlado por un botón que indica que se iniciará una nueva partida del juego.

Salidas:

- Move Button: valor lógico estabilizado que indica que se debe pasar a la siguiente casilla.
- Select Button: valor lógico estabilizado que indica que se debe seleccionar la casilla actual.

- Reset: valor lógico estabilizado que indica que se iniciará una nueva partida del juego.

Explicación General: Este módulo representa un acople de las señales provenientes de los botones para su procesamiento, pues al tratarse de botones mecánicos suele ocurrir un fenómeno conocido como rebote el cual significa en muchas ocasiones una detección incorrecta de señales de entrada cuando no las hay. Este módulo se encarga de contrarrestar este proceso y generar una única señal estable ante la entrada. La salida de este módulo es entrada de la lógica del juego y esencial para su correcto funcionamiento.

5.3.2. Select Position FSM

Objetivo: Permite al usuario moverse a través del tablero y seleccionar la posición que desea marcar.
Entradas:

- MoveButton: esta señal lógica de 1 bit corresponde a la salida de un botón luego de un módulo antirrebote, en su flanco positivo, habilita el contador, lo cual representa el movimiento a la siguiente casilla en el tablero.
- SelectButton: esta señal en su flanco positivo selecciona la casilla que se encuentra marcada; esto se logra al cambiar la salida del multiplexor.
- Reset: esta señal en su flanco positivo indica que se debe iniciar una nueva partida, por lo que reinicia el estado de todos los circuitos.

Salidas:

- Pos: esta señal representa la posición del usuario en el tablero, es un valor binario de 4 bits con valores en decimal desde 0 hasta 8.
- Next_pos: esta señal es igual a la señal de posición anterior con la diferencia de que puede ser desactivada; lo cual implicaría un valor en la salida de 15 (1111). De forma tal que un valor diferente a este significa que ha ocurrido una selección.

Explicación General: Este módulo representa el componente que permite al usuario seleccionar la posición del tablero que desea marcar, se logra mediante una máquina de estado finito, en este caso se implementa mediante un contador simple, cuya salida aumenta con la señal MoveButton, la salida de este componente puede ser seleccionado mediante la señal SelectButton a partir de un multiplexor, cuya salida en caso de no estar seleccionado es 4'b1111. Las salidas de este módulo son entradas para la máquina de estados principal que controla el estado del juego.

5.3.3. Timer

Objetivo: Permite controlar el tiempo que lleva un jugador en su turno, para determinar si este ha excedido el tiempo máximo.
Entradas:

- Reset: esta señal en su flanco positivo indica que se debe iniciar el conteo del timer, pues hubo un cambio de jugador o se inició una nueva partida.
- Clk: esta señal representa un reloj base a partir del cual se calcula .

Salidas:

- time: esta señal indica cuanto tiempo en segundos ha transcurrido desde que se reinició el turno.

Explicación General: Este módulo representa el componente que mantiene el tiempo de un nuevo turno, su implementación se logra mediante un divisor de frecuencia del reloj de entrada.

5.3.4. Current Player

Objetivo: Permite mantener un registro del jugador que se encuentra en turno.
Entradas:

- Toggle: esta señal en su flanco positivo cambia el valor almacenado que indica el jugador que se encuentra en turno.

Salidas:

- player: esta señal indica mediante un valor lógico, cual de los dos jugadores se encuentra en turno.

Explicación General: Este módulo almacena el jugador que se encuentra actualmente en juego, esto se logra mediante un flipflop tipo T. De forma que con una única señal se cambia el jugador en turno.

5.3.5. Game State Memory

Objetivo: Permite mantener un registro estado en que se encuentra la partida actual.
Entradas:

- Address: esta señal selecciona la posición que se desea consultar.

Salidas:

- data: esta señal indica el valor que se encuentra almacenado en el tablero del juego en la posición definida en la entrada.

Explicación General: Este módulo almacena el estado de cada una de las casillas del tablero, indicando por cual jugador se encuentra ocupada o si se encuentra libre. Esto se logra mediante un banco de registros con 9 registros de 2 bits.

5.3.6. Game FSM

Objetivo: Permite comunicar todos los componentes de la lógica del juego entre sí y orquestar su comportamiento.
Entradas:

- Next_pos: esta señal lógica de 4 bits corresponde a la posición seleccionada por el jugador, esta es procesada siempre que sea diferente de 4'b1111.

Señales intermedias:

- player: esta señal indica mediante un valor lógico, cual de los dos jugadores se encuentra en turno.
- time: esta señal indica cuanto tiempo en segundos ha transcurrido desde que se reinició el turno.
- data: esta señal indica el valor que se encuentra almacenado en el tablero del juego en la posición definida en la entrada.

Salidas:

- Wr_fail: esta bandera es activada si el valor de la posición seleccionada (next_pos) se encontraba previamente ocupada. Esta señal detiene el siguiente proceso de verificación de la victoria.
- Game_state: esta señal se encuentra compuesta por una bandera de victoria, que indica si hay un ganador. Además, contiene las posiciones de la tripleta ganadora.

Explicación General: Este módulo representa el componente que procesa la selección de una casilla en el tablero, si la casilla por seleccionar se encuentra ocupada genera una bandera de advertencia, de otra forma modifica el banco de registros, colocando en la posición requerida, el identificador del usuario en turno. Una vez procesada la selección, el comparador se encarga de determinar si la nueva adición genera una victoria, esto se logra al comparar las 8 posibles tripletas ganadoras, y determinar si alguna de estas pertenece a un único jugador, además se encuentra atento ante la señal del timer que controla el turno de cada jugador, y se encarga de modificar el valor almacenado en le flip flop del current player.

A continuación se muestran todos los componentes que forman parte del módulo presentado en el diagrama de segundo nivel como Renderer.

5.3.7. Sprite Memory

Objetivo: Permite almacenar la representación del Sprite de los jugadores.
Entradas:

- Addr: esta señal lógica de 10 bits corresponde a la dirección de la memoria que se desea acceder.

Salidas:

- Data: esta señal corresponde al dato almacenado en la posición seleccionada en addr.

Explicación General: Esta memoria de lectura contiene un Sprite de 32 x 32 pixeles, dado que es una memoria, recibe una dirección de lectura y obtiene el dato almacenado en esta.

5.3.8. Sprite-Position Comparator

Objetivo: Determinar si en la posición entrante se debe dibujar un sprite.
Entradas:

- row: posición vertical en la que se encuentra el pixel a dibujar
- column: posición horizontal en la que se encuentra el pixel a dibujar
- game_state: estado del tablero para así determinar si se debe dibujar un sprite o no.
- data: datos del sprite almacenados en la memoria en la posición estipulada.

Salidas:

- Address: posición del pixel en la memoria, que se desea dibujar.
- RGB: información de la composición del pixel que se dibujará en la pantalla.
- visible: valor lógico de 1 bit que define si en la posición definida se debe dibujar un sprite o no.

Explicación General: Esta componente realiza una comparación de la posición en la que se dibujará un píxel y determina si este posición se encuentra en un rango donde se deba dibujar un sprite

---

5.3.9. Selection-Position Comparator

Objetivo: Determinar si en la posición entrante se debe dibujar un color distinto en el fondo para determinar que se encuentra seleccionado por el usuario.
Entradas:

- row: posición vertical en la que se encuentra el pixel a dibujar
- column: posición horizontal en la que se encuentra el pixel a dibujar

- game_state: estado del tablero para así determinar si se debe dibujar un sprite o no.
- data: datos del sprite almacenados en la memoria en la posición estipulada.

Salidas:

- Address: posición del pixel en la memoria, que se desea dibujar.
- RGB: información de la composición del pixel que se dibujará en la pantalla.
- visible: valor lógico de 1 bit que define si en la posición definida se debe cambiar el color de fondo o no.

Explicación General: Esta componente realiza una comparación de la posición en la que se dibujará un píxel y determina si este posición se encuentra en un rango donde se deba cambiar el fondo del tablero pues el jugador está seleccionando esta posición.

5.3.10. Line-Position Comparator

Objetivo: Determinar si en la posición entrante se debe dibujar una linea del tablero.
Entradas:

- row: posición vertical en la que se encuentra el pixel a dibujar
- column: posición horizontal en la que se encuentra el pixel a dibujar

Salidas:

- RGB: información de la composición del pixel que se dibujará en la pantalla, representa el color de la línea.
- visible: valor lógico de 1 bit que define si en la posición definida se debe dibujar una línea o no.

Explicación General: Esta componente realiza una comparación de la posición en la que se dibujará un píxel y determina si este posición se encuentra en un rango donde equivale a una linea del tablero por lo que se debe cambiar el color por mostrar.

5.3.11. Mux

Objetivo: Decidir cual de las señales será mostrada en pantalla entre sus posibles entradas.
Entradas:

- RGB1: información de la composición del pixel que se dibujará en la pantalla, representa el color del sprite.
- RGB2:información de la composición del pixel que se dibujará en la pantalla, representa el color de la línea.
- RGB3:información de la composición del pixel que se dibujará en la pantalla, representa el color del fondo si este se encuentra seleccionado.

Salidas:

- VGA_R: bus de datos de 8 bits con la intensidad del color rojo del pixel seleccionado.
- VGA_G: bus de datos de 8 bits con la intensidad del color verde del pixel seleccionado.
- VGA_B: bus de datos de 8 bits con la intensidad del color azul del pixel seleccionado.

Explicación General: Este componente es de gran importancia y permite controlar y coordinar la información que está siendo enviada sobre cada pixel, a partir de este componente se conoce la posición que debe tomar cada pixel y además la señal de la fila y columna actual permiten a los demás componentes decidir que deberá mostrarse en dicha posición.

A continuación se muestran todos los componentes que forman parte del módulo presentado en el diagrama de segundo nivel como VGA controller.

5.3.12. Horizontal Counter

Objetivo: Contar el número de columna en que se encuentra el píxel actual.
Entradas:

- pixel_clk: reloj de píxeles, su frecuencia define la velocidad en que se envían los datos de cada uno de estos píxeles.

Salidas:

- VGA_HS: Esta señal es la encargada de la sincronización horizontal.
- column: Este valor numérico indica en qué posición horizontal se encuentra el píxel que se está evaluando.

Explicación General: Este componente es de gran importancia y permite controlar y coordinar de forma horizontal la información que está siendo enviada sobre cada píxel. La señal de sincronización horizontal se activa cada vez que se inicia una nueva fila, además indica cuando se debe reiniciar el conteo de la columna. La salida del valor de la columna, se relaciona con el componente anterior que determina que debe ser dibujado según la posición (fila y columna) dada.

5.3.13. Vertical Counter

Objetivo: Contar el número de fila en que se encuentra el píxel actual.
Entradas:

- column_clk: reloj de columnas, esta señal se activa cada vez que el contador de columnas alcanza un valor específico, lo cual indica que debe haber un cambio de fila.

Salidas:

- VGA_VS: Esta señal es la encargada de la sincronización Vertical.
- row: Este valor numérico indica en qué posición vertical se encuentra el píxel que se está evaluando.

Explicación General: Este componente es de gran importancia y permite controlar y coordinar de forma vertical la información que está siendo enviada sobre cada píxel. La señal de sincronización vertical se activa cada vez que se inicia un nuevo frame, además indica cuando se debe reiniciar el conteo de la fila. La salida del valor de la fila, se relaciona con los componentes comparadores del renderer que determinan que debe ser dibujado según la posición (fila y columna) dada.

5.3.14. Comparator

Objetivo: Generar la señal de conteo de filas a partir del conteo de columnas.
Entradas:

- column_clk: Frequencia de cambio de columna, representa el movimiento horizontal.

Salidas:

- VGA_SYNC_N: esta señal no maneja ninguna señal de control o datos, solo debe ser utilizada durante el proceso que está activa la señal VGA_BLANK_N.
- VGA_BLANK_N: señal lógica que se activa cuando se encuentra fuera del rango visible para que el controlador no intente colocar los colores de esas posiciones.

Explicación General: Este componente realiza una comparación del valor de salida del contador horizontal, y una vez que este alcanza un valor específico, genera una señal que será utilizada para aumentar el contador vertical.

5.3.15. Clock Divider

Objetivo: Disminuir la frecuencia de operación del reloj base del sistema a la frecuencia apropiada para los pixeles.

Entradas:

- clk: Reloj base de aproximadamente 50MHz

Salidas:

- pixel_clk: Reloj de frecuencia apropiada para ser utilizada por el controlador VGA.

Explicación General: Este componente se comporta como un contador, que aumenta su conteo con cada flanco del reloj base, además de esto involucra un comparador encargado de generar una señal luego de una cantidad definida de ciclos, lo cual se representa en la salida como una disminución de la frecuencia del reloj. Basta con definir la cantidad de ciclos que deben transcurrir para modificar la división de la frecuencia.

5.4. Cuarto Nivel

A continuación se muestran todos los componentes que forman parte del módulo presentado en el diagrama de segundo nivel como Game Logic.

5.4.1. Debouncer

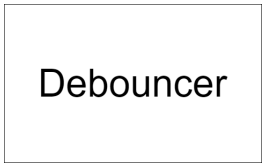

Figura 10: Elemento utilizado para la eliminación del efecto de rebote

El efecto de este bloque se aprecia en la siguiente figura.

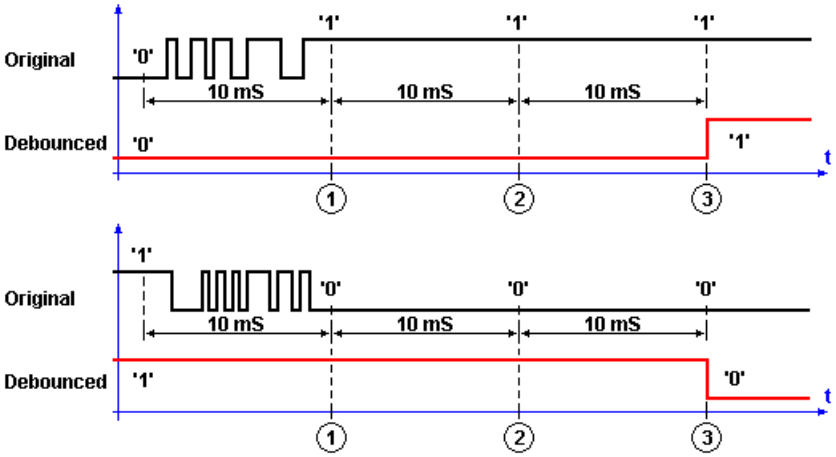

Figura 11: Eliminación del efecto rebote en una señal de ejemplo

Objetivo: Acoplar las señales de entrada para que puedan ser utilizadas por el sistema sin detectar selecciones dobles o falsas.

Entradas:

- Move Button: valor lógico controlado por un botón que indica que se debe pasar a la siguiente casilla.
- Select Button: valor lógico controlado por un botón que indica que se debe seleccionar la casilla actual.
- Reset: valor lógico controlado por un botón que indica que se iniciará una nueva partida del juego.

Salidas:

- Move Button: valor lógico estabilizado que indica que se debe pasar a la siguiente casilla.
- Select Button: valor lógico estabilizado que indica que se debe seleccionar la casilla actual.
- Reset: valor lógico estabilizado que indica que se iniciará una nueva partida del juego.

Explicación General: Este módulo representa un acople de las señales provenientes de los botones para su procesamiento, pues al tratarse de botones mecánicos suele ocurrir un fenómeno conocido como rebote el cual significa en muchas ocasiones una detección incorrecta de señales de entrada cuando no las hay. Este módulo se encarga de contrarrestar este proceso y generar una única señal estable ante la entrada. La salida de este módulo es entrada de la lógica del juego y esencial para su correcto funcionamiento.

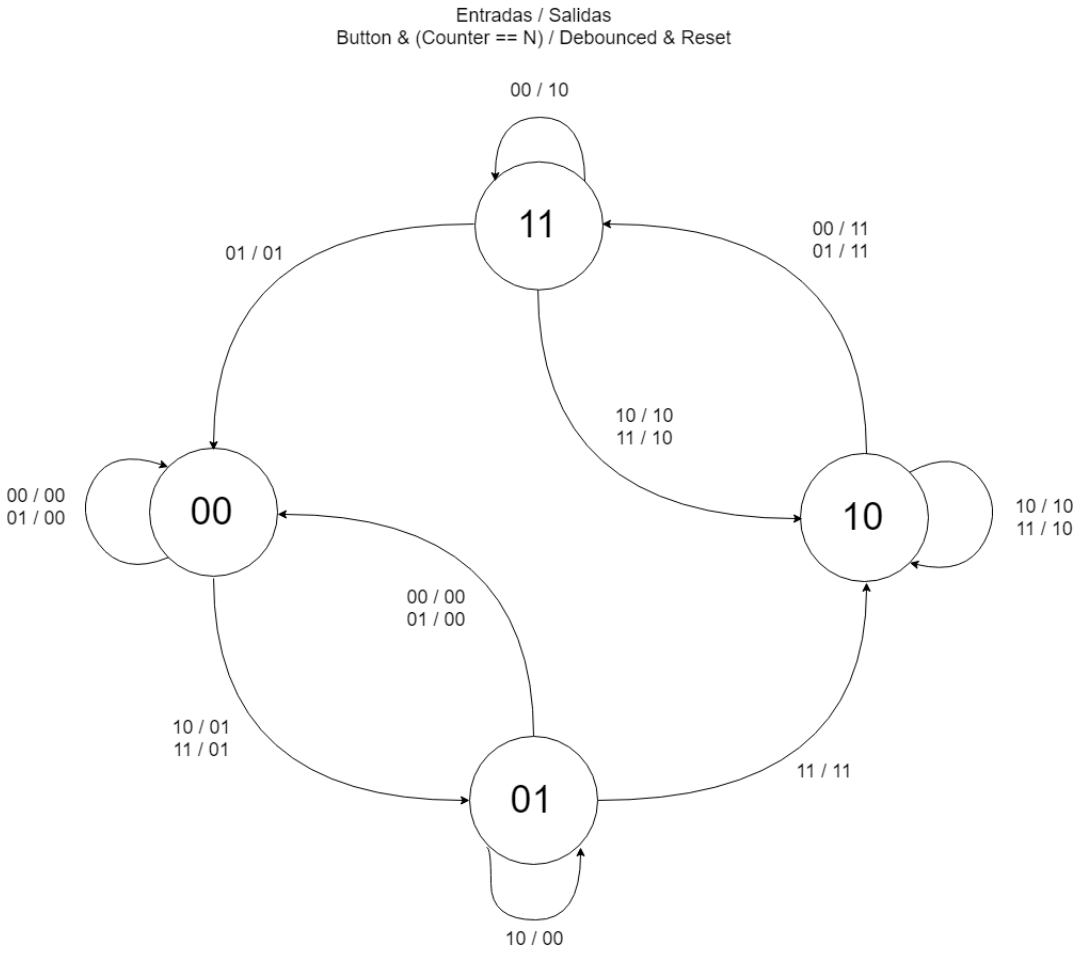

Figura 12: Maquina de estados para módulo antirrebote

La maquina de estados anterior utiliza un contador externo, pero recibe una señal de este que indica si el conteo ha superado un valor específico; además envía a este una señal de reset, para reiniciar el conteo. Las entradas de la máquina de estado, son el botón que se desea estabilizar, y una señal lógica que define si el contador ha alcanzado el valor definido. Las salidas del módulo son el valor de la señal estable y una señal de reset para el contador externo

5.4.2. Select Position FSM

La maquina de estados en cuestión puede ser implementada como un contador simple de 4 bits, pues esta máquina de estados recorre cada uno de los estados de forma ordenada.

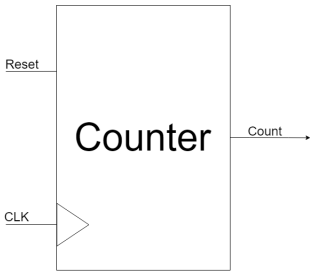

Figura 13: Representación de un contador de 4 bits

Objetivo: Permite al usuario moverse a través del tablero y seleccionar la posición que desea marcar.
Entradas:

- MoveButton: esta señal lógica de 1 bit corresponde a la salida de un botón luego de un módulo antirrebote, en su flanco positivo, habilita el contador, lo cual representa el movimiento a la siguiente casilla en el tablero.
- SelectButton: esta señal en su flanco positivo selecciona la casilla que se encuentra marcada; esto se logra al cambiar la salida del multiplexor.
- Reset: esta señal en su flanco positivo indica que se debe iniciar una nueva partida, por lo que reinicia el estado de todos los circuitos.

Salidas:

- Pos: esta señal representa la posición del usuario en el tablero, es un valor binario de 4 bits con valores en decimal desde 0 hasta 8.
- Next_pos: esta señal es igual a la señal de posición anterior con la diferencia de que puede ser desactivada; lo cual implicaría un valor en la salida de 15 (1111). De forma tal que un valor diferente a este significa que ha ocurrido una selección.

Explicación General: Este módulo representa el componente que permite al usuario seleccionar la posición del tablero que desea marcar, se logra mediante una máquina de estado finito, en este caso se implementa mediante un contador simple, cuya salida aumenta con la señal MoveButton, la salida de este componente puede ser seleccionado mediante la señal SelectButton a partir de un multiplexor, cuya salida en caso de no estar seleccionado es 4'b1111. Las salidas de este módulo son entradas para la máquina de estados principal que controla el estado del juego.

5.4.3. Timer

El cronómetro que contiene el tiempo en turno de un usuario puede implementarse mediante un contador simple.

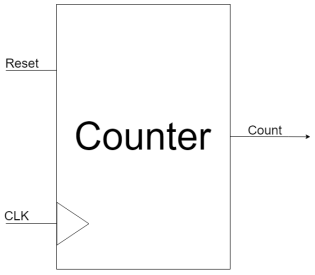

Figura 14: Representación de un cronómetro mediante un contador

Objetivo: Permite controlar el tiempo que lleva un jugador en su turno, para determinar si este ha excedido el tiempo máximo.
Entradas:

- Reset: esta señal en su flanco positivo indica que se debe iniciar el conteo del timer, pues hubo un cambio de jugador o se inició una nueva partida.
- Clk: esta señal representa un reloj base a partir del cual se calcula .

Salidas:

- time: esta señal indica cuanto tiempo en segundos ha transcurrido desde que se reinició el turno.

Explicación General: Este módulo representa el componente que mantiene el tiempo de un nuevo turno, su implementación se logra mediante un divisor de frecuencia del reloj de entrada.

5.4.4. Current Player

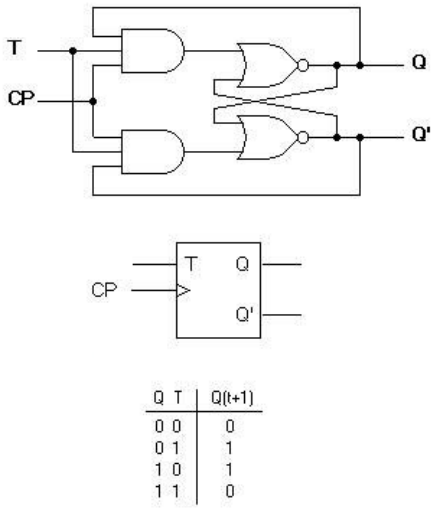

Figura 15: Circuito, símbolo equivalente y tabla de verdad de un flip flop tipo T

Objetivo: Permite mantener un registro del jugador que se encuentra en turno.
Entradas:

- Toggle: esta señal en su flanco positivo cambia el valor almacenado que indica el jugador que se encuentra en turno.

Salidas:

- player: esta señal indica mediante un valor lógico, cual de los dos jugadores se encuentra en turno.

Explicación General: Este módulo almacena el jugador que se encuentra actualmente en juego, esto se logra mediante un flipflop tipo T. De forma que con una única señal se cambia el jugador en turno.

5.4.5. Game State Memory

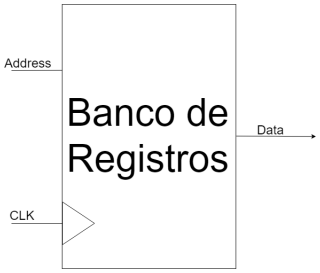

Figura 16: Representación de un banco de registros simple

Objetivo: Permite mantener un registro estado en que se encuentra la partida actual.
Entradas:

- Address: esta señal selecciona la posición que se desea consultar.

Salidas:

- data: esta señal indica el valor que se encuentra almacenado en el tablero del juego en la posición definida en la entrada.

Explicación General: Este módulo almacena el estado de cada una de las casillas del tablero, indicando por cual jugador se encuentra ocupada o si se encuentra libre. Esto se logra mediante un banco de registros con 9 registros de 2 bits.

5.4.6. Game FSM

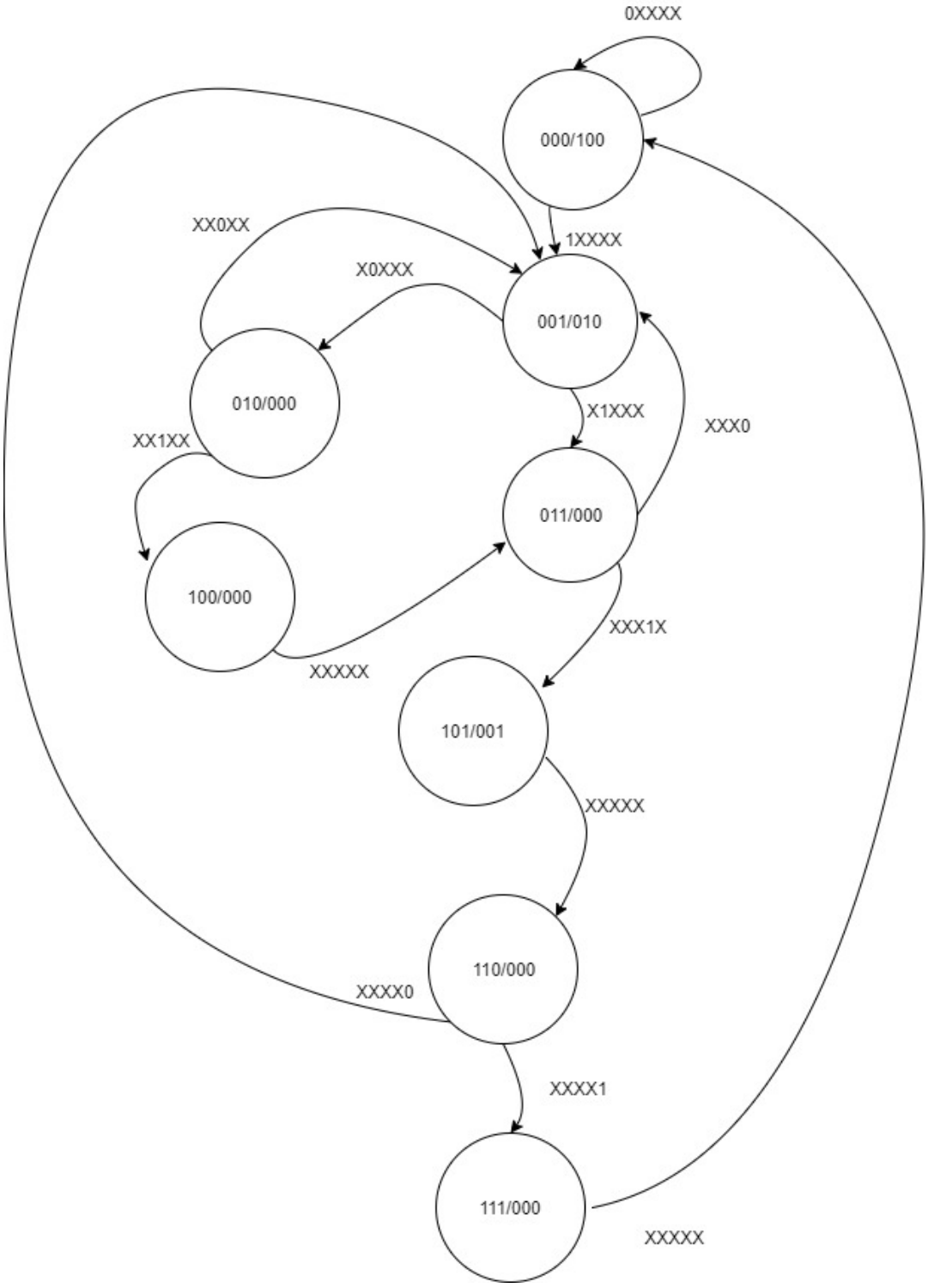

Figura 17: Maquina de estados principal que controla las señales del juego

Objetivo: Permite comunicar todos los componentes de la lógica del juego entre sí y orquestar su comportamiento.

Entradas:

- Next_pos: esta señal lógica de 4 bits corresponde a la posición seleccionada por el jugador, esta es procesada siempre que sea diferente de 4'b1111.

Señales intermedias:

- player: esta señal indica mediante un valor lógico, cual de los dos jugadores se encuentra en turno.
- time: esta señal indica cuanto tiempo en segundos ha transcurrido desde que se reinició el turno.
- data: esta señal indica el valor que se encuentra almacenado en el tablero del juego en la posición definida en la entrada.

Salidas:

- Wr_fail: esta bandera es activada si el valor de la posición seleccionada (next_pos) se encontraba previamente ocupada. Esta señal detiene el siguiente proceso de verificación de la victoria.
- Game_state: esta señal se encuentra compuesta por una bandera de victoria, que indica si hay un ganador. Además, contiene las posiciones de la tripleta ganadora.

Explicación General: Este módulo representa el componente que procesa la selección de una casilla en el tablero, si la casilla por seleccionar se encuentra ocupada genera una bandera de advertencia, de otra forma modifica el banco de registros, colocando en la posición requerida, el identificador del usuario en turno. Una vez procesada la selección, el comparador se encarga de determinar si la nueva adición genera una victoria, esto se logra al comparar las 8 posibles tripletas ganadoras, y determinar si alguna de estas pertenece a un único jugador, además se encuentra atento ante la señal del timer que controla el turno de cada jugador, y se encarga de modificar el valor almacenado en le flip flop del current player.

A continuación se muestran todos los componentes que forman parte del módulo presentado en el diagrama de segundo nivel como Renderer.

5.4.7. Sprite Memory

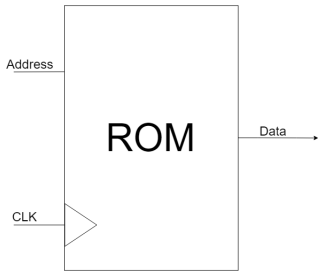

Figura 18: Implementación del almacenamiento del Sprite mediante una ROM

Objetivo: Permite almacenar la representación del Sprite de los jugadores.
Entradas:

- Addr: esta señal lógica de 10 bits corresponde a la dirección de la memoria que se desea acceder.

Salidas:

- Data: esta señal corresponde al dato almacenado en la posición seleccionada en addr.

Explicación General: Esta memoria de lectura contiene un Sprite de 32 x 32 pixeles, dado que es una memoria, recibe una dirección de lectura y obtiene el dato almacenado en esta.

5.4.8. Sprite-Position Comparator

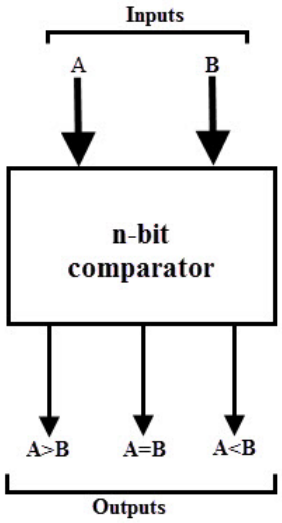

Figura 19: Comparador de Sprite

Objetivo: Determinar si en la posición entrante se debe dibujar un sprite.
Entradas:

- row: posición vertical en la que se encuentra el pixel a dibujar
- column: posición horizontal en la que se encuentra el pixel a dibujar
- game_state: estado del tablero para así determinar si se debe dibujar un sprite o no.
- data: datos del sprite almacenados en la memoria en la posición estipulada.

Salidas:

- Address: posición del pixel en la memoria, que se desea dibujar.
- RGB: información de la composición del pixel que se dibujará en la pantalla.
- visible: valor lógico de 1 bit que define si en la posición definida se debe dibujar un sprite o no.

Explicación General: Esta componente realiza una comparación de la posición en la que se dibujará un píxel y determina si este posición se encuentra en un rango donde se deba dibujar un sprite

---

5.4.9. Selection-Position Comparator

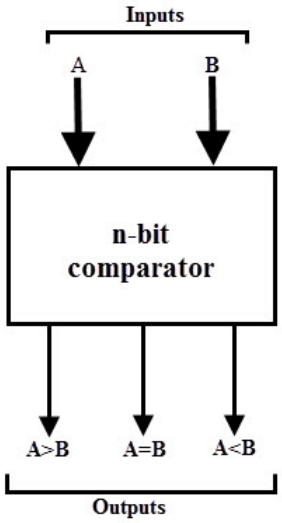

Figura 20: Comparador de selección

Objetivo: Determinar si en la posición entrante se debe dibujar un color distinto en el fondo para determinar que se encuentra seleccionado por el usuario.

Entradas:

- row: posición vertical en la que se encuentra el pixel a dibujar
- column: posición horizontal en la que se encuentra el pixel a dibujar
- game_state: estado del tablero para así determinar si se debe dibujar un sprite o no.
- data: datos del sprite almacenados en la memoria en la posición estipulada.

Salidas:

- Address: posición del pixel en la memoria, que se desea dibujar.
- RGB: información de la composición del pixel que se dibujará en la pantalla.
- visible: valor lógico de 1 bit que define si en la posición definida se debe cambiar el color de fondo o no.

Explicación General: Esta componente realiza una comparación de la posición en la que se dibujará un píxel y determina si este posición se encuentra en un rango donde se deba cambiar el fondo del tablero pues el jugador está seleccionando esta posición.

5.4.10. Line-Position Comparator

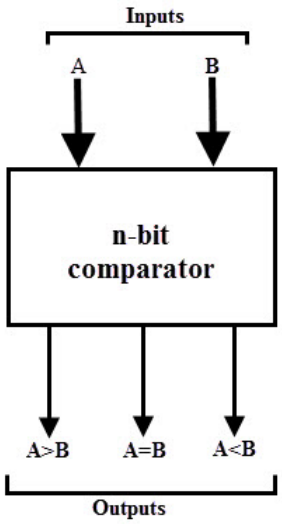

Figura 21: Comparador de linea

Objetivo: Determinar si en la posición entrante se debe dibujar una linea del tablero.
Entradas:

- row: posición vertical en la que se encuentra el pixel a dibujar
- column: posición horizontal en la que se encuentra el pixel a dibujar

Salidas:

- RGB: información de la composición del pixel que se dibujará en la pantalla, representa el color de la línea.
- visible: valor lógico de 1 bit que define si en la posición definida se debe dibujar una línea o no.

Explicación General: Esta componente realiza una comparación de la posición en la que se dibujará un píxel y determina si este posición se encuentra en un rango donde equivale a una linea del tablero por lo que se debe cambiar el color por mostrar.

5.4.11. Mux

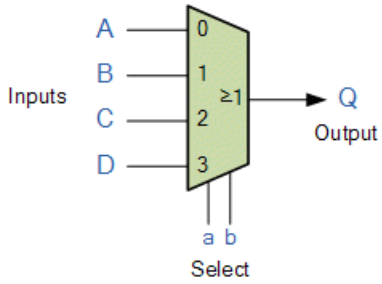

Figura 22: Multiplexor de informaicón del pixel

Objetivo: Decidir cual de las señales será mostrada en pantalla entre sus posibles entradas.
Entradas:

- RGB1: información de la composición del pixel que se dibujará en la pantalla, representa el color del sprite.
- RGB2:información de la composición del pixel que se dibujará en la pantalla, representa el color de la línea.
- RGB3:información de la composición del pixel que se dibujará en la pantalla, representa el color del fondo si este se encuentra seleccionado.

Salidas:

- VGA_R: bus de datos de 8 bits con la intensidad del color rojo del pixel seleccionado.
- VGA_G: bus de datos de 8 bits con la intensidad del color verde del pixel seleccionado.
- VGA_B: bus de datos de 8 bits con la intensidad del color azul del pixel seleccionado.

Explicación General: Este componente es de gran importancia y permite controlar y coordinar la información que está siendo enviada sobre cada pixel, a partir de este componente se conoce la posición que debe tomar cada pixel y además la señal de la fila y columna actual permiten a los demás componentes decidir que deberá mostrarse en dicha posición.

A continuación se muestran todos los componentes que forman parte del módulo presentado en el diagrama de segundo nivel como VGA controller.

5.4.12. Horizontal Counter

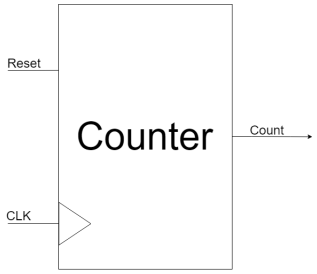

Figura 23: Contador horizontal

Objetivo: Contar el número de columna en que se encuentra el píxel actual.
Entradas:

- pixel_clk: reloj de píxeles, su frecuencia define la velocidad en que se envían los datos de cada uno de estos píxeles.

Salidas:

- VGA_HS: Esta señal es la encargada de la sincronización horizontal.
- column: Este valor numérico indica en qué posición horizontal se encuentra el píxel que se está evaluando.

Explicación General: Este componente es de gran importancia y permite controlar y coordinar de forma horizontal la información que está siendo enviada sobre cada píxel. La señal de sincronización horizontal se activa cada vez que se inicia una nueva fila, además indica cuando se debe reiniciar el conteo de la columna. La salida del valor de la columna, se relaciona con el componente anterior que determina que debe ser dibujado según la posición (fila y columna) dada.

5.4.13. Vertical Counter

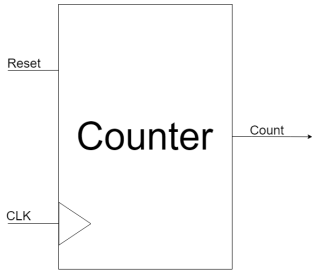

Figura 24: Contador Vertical

Objetivo: Contar el número de fila en que se encuentra el píxel actual.
Entradas:

- column_clk: reloj de columnas, esta señal se activa cada vez que el contador de columnas alcanza un valor específico, lo cual indica que debe haber un cambio de fila.

Salidas:

- VGA_VS: Esta señal es la encargada de la sincronización Vertical.
- row: Este valor numérico indica en qué posición vertical se encuentra el píxel que se está evaluando.

Explicación General: Este componente es de gran importancia y permite controlar y coordinar de forma vertical la información que está siendo enviada sobre cada píxel. La señal de sincronización vertical se activa cada vez que se inicia un nuevo frame, además indica cuando se debe reiniciar el conteo de la fila. La salida del valor de la fila, se relaciona con los componentes comparadores del renderer que determinan que debe ser dibujado según la posición (fila y columna) dada.

5.4.14. Comparator

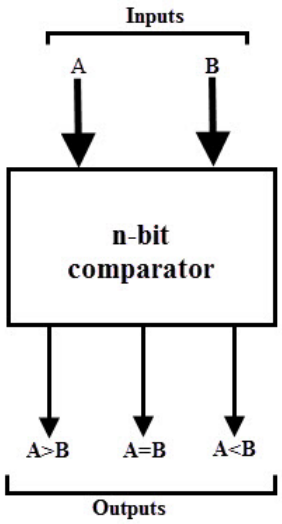

Figura 25: Comparador de columnas

Objetivo: Generar la señal de conteo de filas a partir del conteo de columnas.
Entradas:

- column_clk: Frequencia de cambio de columna, representa el movimiento horizontal.

Salidas:

- VGA_SYNC_N: esta señal no maneja ninguna señal de control o datos, solo debe ser utilizada durante el proceso que está activa la señal VGA_BLANK_N.
- VGA_BLANK_N: señal lógica que se activa cuando se encuentra fuera del rango visible para que el controlador no intente colocar los colores de esas posiciones.

Explicación General: Este componente realiza una comparación del valor de salida del contador horizontal, y una vez que este alcanza un valor específico, genera una señal que será utilizada para aumentar el contador vertical.

5.4.15. Clock Divider

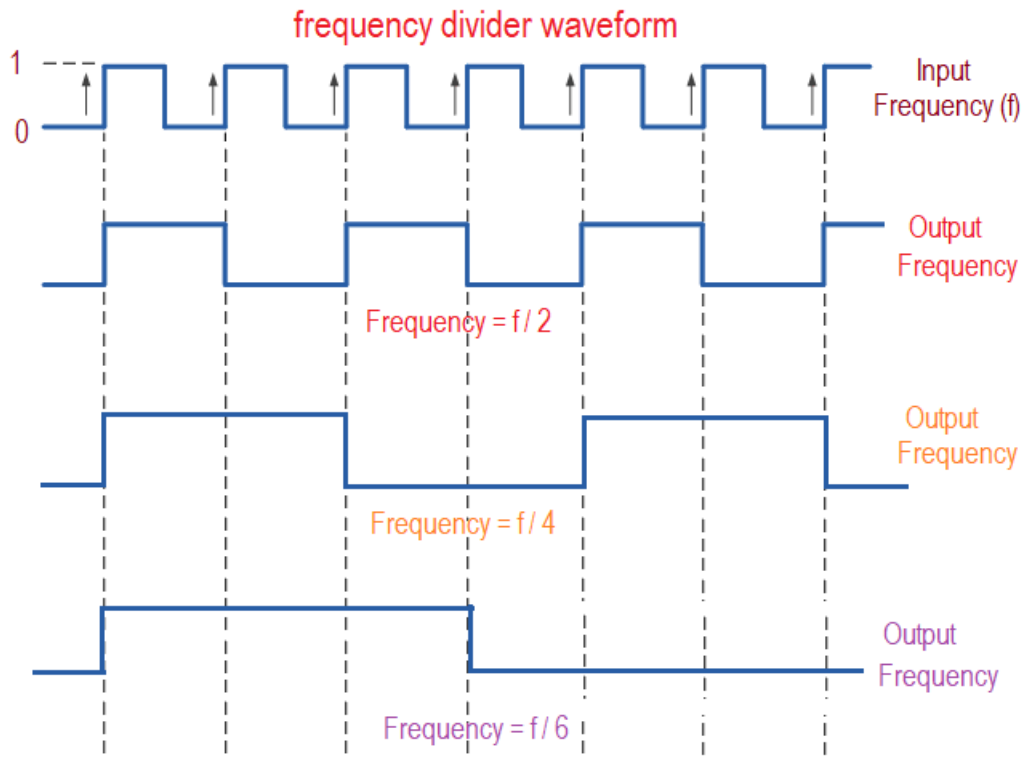

Figura 26: División de frecuencia de un reloj de ejemplo

Objetivo: Disminuir la frecuencia de operación del reloj base del sistema a la frecuencia apropiada para los pixeles.

Entradas:

- clk: Reloj base de aproximadamente 50MHz

Salidas:

- pixel_clk: Reloj de frecuencia apropiada para ser utilizada por el controlador VGA.

Explicación General: Este componente se comporta como un contador, que aumenta su conteo con cada flanco del reloj base, además de esto involucra un comparador encargado de generar una señal luego de una cantidad

definida de ciclos, lo cual se representa en la salida como una disminución de la frecuencia del reloj. Basta con definir la cantidad de ciclos que deben transcurrir para modificar la división de la frecuencia.

5.5. Quinto Nivel

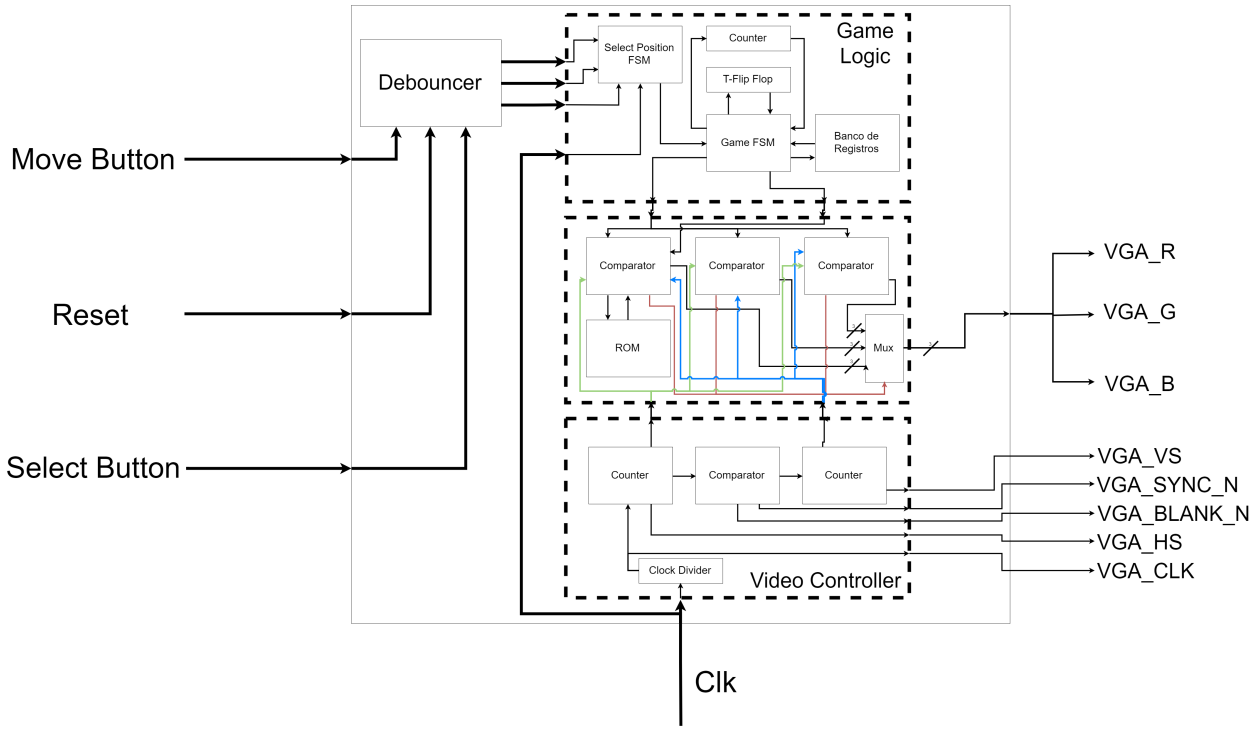

Figura 27: Diagrama de quinto nivel

Referencias

[1] Terasic De1-SoC User Manual

[2] Sarah Harris and David Harris. Digital Design and Computer Architecture: ARM Edition. Morgan Kaufmann, 2015.