UNIVERSIDAD CARLOS III DE MADRID
ESCUELA POLITÉCNICA SUPERIOR

Dpto. de TECNOLOGÍA ELECTRÓNICA

INGENIERÍA TÉCNICA INDUSTRIAL
ESPECIALIDAD ELECTRÓNICA INDUSTRIAL

PROYECTO FIN DE CARRERA

DISEÑO E IMPLEMENTACION EN FPGA DEL JUEGO DE LAS DAMAS

AUTOR: MANUEL RECAS PEREZ

TUTOR: MARTA PORTELA GARCIA

A mis padres y mi hermana.

Agradecimientos

En primer lugar, me gustaría mostrar mi agradecimiento a mi tutora de proyecto, Marta Portela, por su gran ayuda y paciencia a lo largo del proyecto.

A mis compañeros de universidad, Pablo, Borja, Tamara, Francisco y un sin fin de nombres que me han ayudado a acabar la carrera, pasando momentos inolvidables.

A mis amigos del barrio, con los que he compartido durante todos estos años buenos y malos momentos y que siempre han estado ahí.

A los distintos foros de electrónica donde multitud de personas brindan sus conocimientos de forma desinteresada y sin otro propósito que el de la divulgación científica y técnica.

Por ultimo un agradecimiento especial a mis padres y hermana que me han soportado y apoyado en todas mis decisiones, y que sin ellos no habría llegado hasta el fin.

ÍNDICE

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

1 Introducción

1.1 Introducción

Este proyecto tiene como objetivo principal el estudio de viabilidad de un sistema digital para su uso en la asignatura “Laboratorio de Microelectrónica” de Ingeniería de Telecomunicaciones. Dicha asignatura aborda de una manera práctica el diseño de un circuito integrado de aplicación específica de complejidad media en alto nivel. Para ello los alumnos deben desarrollar el diseño especificado en el lenguaje de descripción hardware VHDL que aprendieron en una asignatura anterior. Finalmente el sistema debe implementarse en una plataforma hardware con una FPGA (Field Programmable Gate Array).

Tradicionalmente, los proyectos a realizar en dicha asignatura han consistido en sistemas que implementaban algún tipo de juego como El juego de la Oca o Las Cuatro en Raya. En este proyecto se desea desarrollar un sistema digital que implemente el juego de Las Damas para evaluar su dificultad y posible uso en la asignatura.

Las damas es un juego de mesa estratégico de dos jugadores, que se juega sobre un tablero de 64 casillas. Cada jugador comienza con doce piezas simples o peones, y el objetivo es capturar todas las piezas del oponente.
En este proyecto, se aborda el diseño hardware del juego de las damas. Para llevarlo a cabo se han tenido en cuenta una serie de especificaciones impuestas por el propio juego como son las reglas, y otras relacionadas con la implementación del sistema que se han fijado durante la realización del proyecto como el uso de un teclado pasivo para la introducción de posiciones, o el uso de una pantalla VGA para visualizar las partidas.

1.2 Objetivos

Como se señaló en el apartado anterior, este proyecto tiene como objetivo principal el estudio de viabilidad de un sistema digital para su uso en la asignatura “Laboratorio de Microelectrónica” impartida durante el cuarto curso de la titulación de Ingeniería de Telecomunicaciones. Este objetivo general puede desglosarse en los siguientes puntos:

- Especificar y fijar los componentes hardware a utilizar para la implementación final del circuito digital.
- Realizar el diseño e implementación del sistema propuesto, partiendo de unas especificaciones generales junto con las fijadas en el punto anterior para su realización.
- Evaluar la dificultad del diseño final así como de las distintas partes que lo componen teniendo en cuenta el esfuerzo realizado en su desarrollo durante la realización del proyecto, el conocimiento previo que tienen los alumnos de la asignatura cuando la cursan, la limitación del tiempo para el desarrollo del sistema (30 horas de laboratorio) y el número de personas que conformaran los grupos.
- Finalmente, se desea realizar una propuesta de trabajo en función de las conclusiones obtenidas en el punto anterior.

1.3 Estructura del documento

A continuación se describe la estructura del presente documento.

Capítulo 1 "Introducción", se exponen los objetivos principales y la estructura del mismo.

El capítulo 2 "Especificaciones del diseño", comienza con una breve introducción a la historia de las damas, y los fundamentos del juego junto con las reglas que se han tenido en cuenta para la realización del proyecto. Aquí, también hablaremos de los recursos hardware utilizados a lo largo del proyecto.

En el capítulo 3 "Descripción del diseño", se explica el funcionamiento interno de cada bloque o módulo.

En el capítulo 4 "Evaluación del circuito", se muestran los datos obtenidos de las distintas simulaciones para verificar el correcto funcionamiento del diseño.

Las conclusiones a las que se han llegado se muestran en el capítulo 5. Mientras que los posibles trabajos futuros y líneas de investigación que se pueden seguir, se muestran en el capítulo 6.

Al final del documento, se incluyen la bibliografía utilizada.

2 ESPECIFICACIONES DEL DISEÑO

2.1 Juego de las damas

No está claro el origen del antiquísimo juego de tablero de las damas, si bien algunos historiadores se aventuran a afirmar que el juego de las damas tiene sus raíces en el Egipto de los faraones, otros que en el imperio romano; pero aparte de estas especulaciones lo real es que España es la primera nación donde se publica un "Libro de Damas", concretamente en 1547, mientras que en Francia la primera publicación que presenta cierta concreción con el juego, aparece en 1668. Ver referencia [11]

La variante española, es considerada como la más antigua de las variantes vivas del juego de damas. Es la practicada en toda la Península Ibérica, en el Norte de África y en muchos países de Sudamérica y Centroamérica.

El juego de las damas evolucionó de forma distinta en diferentes lugares del mundo dando lugar a muchas variantes muy diferenciadas como las Damas Turcas, Rusas, Internacionales, etc., cada una de ellas con sus matices y reglas diferentes, aunque con características comunes.

2.2 Requisitos específicos

Las “damas” es un juego de estrategia, que se juega sobre un tablero de 64 casillas de las cuales sólo se usan 32. Cada jugador cuenta inicialmente con 12 fichas de un mismo color, con dichas fichas tiene que intentar capturar las del contrario siguiendo una serie de reglas que veremos a continuación.

2.2.1 Tablero y piezas

El juego de las damas se lleva a cabo sobre un damero de 8x8 con esquina inferior derecha de color blanco y 12 peones por jugador (blancos y negros respectivamente para cada uno de ellos) situados sobre las casillas blancas de las tres primeras filas de cada jugador. El desarrollo del juego se realiza por lo tanto sólo sobre las casillas blancas. Ver Figura 2.1.

Cuando un peón llegue a la primera fila del bando contrario, se corona, convirtiéndose en dama, la cual se distingue del resto de piezas por la colocación de otro peón del mismo color sobre el que llega a la primera fila.

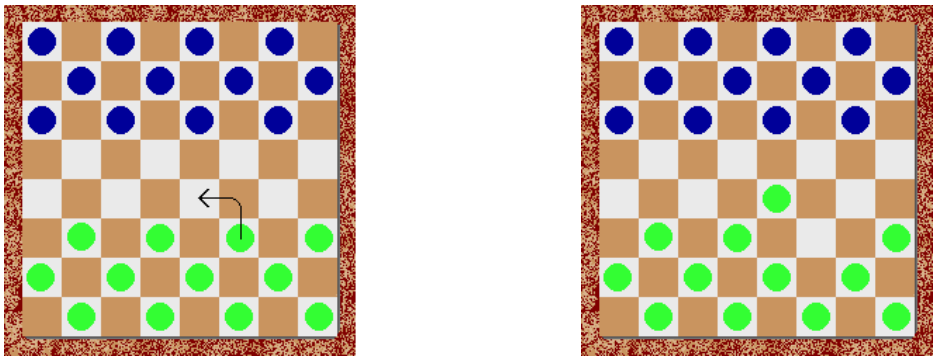
En el proyecto, las fichas son de color azul (negras) y verde (blancas) para una mejor visualización en la pantalla VGA.

Figura 2.1 Tablero y Piezas

2.2.2 Movimientos

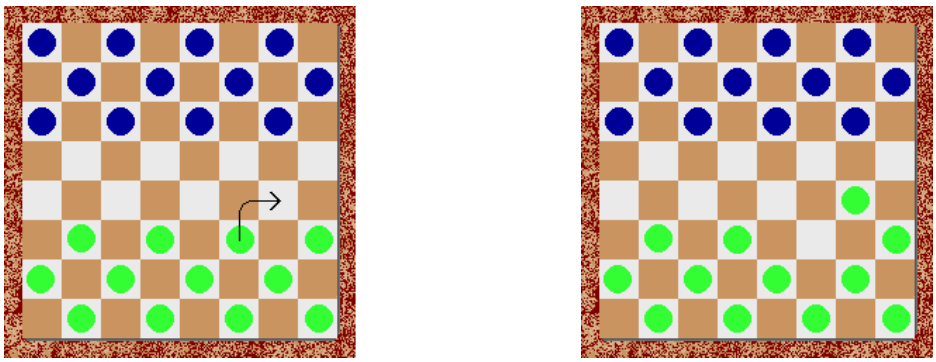
El juego lo comienzan los peones de color blanco, en nuestro caso serian las de color verde. Avanzan en diagonal, desplazándose una casilla. Como podemos ver en las

figuras 2.2 y 2.3, dónde se muestra el desplazamiento simple de un peón a izquierdas y a derechas respectivamente.



a) b)

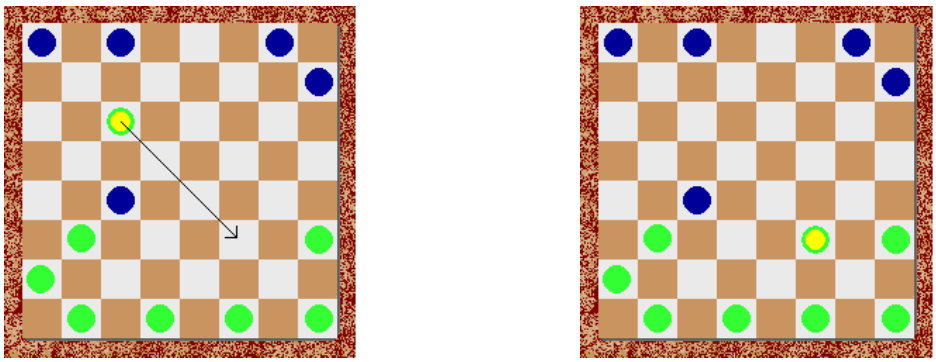
Figura 2.2 Movimiento simple a izquierda. a) Posición inicial. b) Posición final tras el movimiento



a) b)

Figura 2.3 Movimiento simple a derecha. a) Posición inicial. b) Posición final tras el movimiento

El movimiento de las damas es distinto al de los peones. Para desplazarse, las damas avanzan el número de casillas que se desee tanto hacia delante como hacia atrás y siempre en diagonal. La Figura 2.4 muestra un ejemplo del movimiento de una dama.

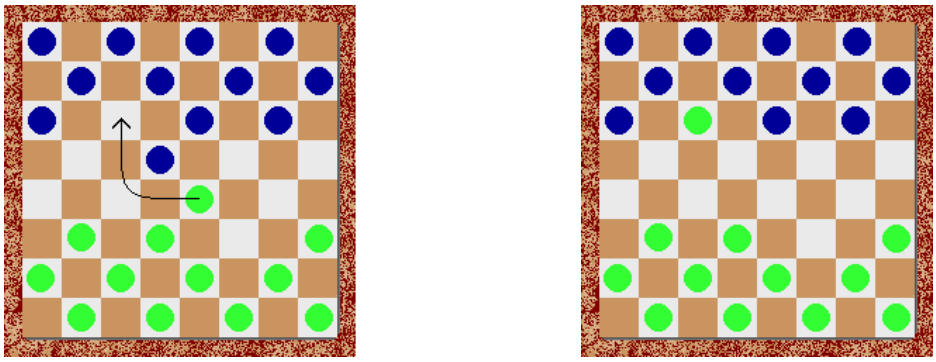


a) b)

Figura 2.4 Movimiento simple con la dama. a) Posición inicial. b) Posición final tras el movimiento

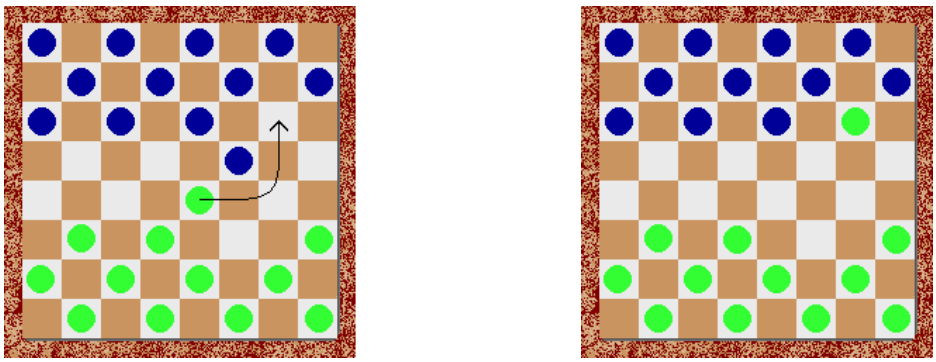
2.2.3 Capturas

El peón captura en diagonal sólo hacia delante (en sentido a la posición de salida del adversario), saltando por encima de la ficha contraria que va a ser capturada, cayendo sobre la casilla inmediatamente detrás de ésta (en el sentido de la captura), y siempre que la ficha que se captura esté en una casilla adyacente al capturado, y que la casilla inmediatamente detrás de éste esté libre para que acabe el movimiento. La Figura 2.5 muestra un ejemplo de captura simple realizada por un peón hacia la izquierda, y la Figura 2.6 hacia la derecha.



a) b)

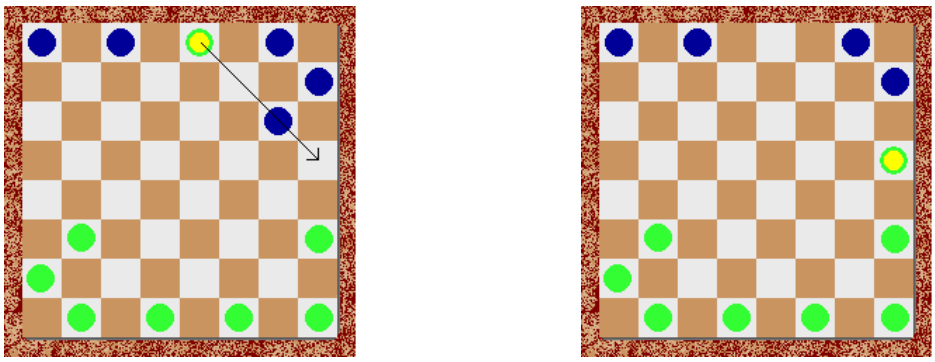
Figura 2.5 Captura simple a izquierdas. a) Posición inicial. b) Posición final tras el movimiento



a) b)

Figura 2.6 Captura simple a derechas. a) Posición inicial. b) Posición final tras el movimiento

La captura con dama es igual que con peón pero, como en el caso de los movimientos, la dama tiene más poder que los peones. La dama puede capturar tanto hacia adelante como hacia atrás y la ficha a capturar no tiene por qué estar en una casilla adyacente mientras esté en la misma diagonal y sin piezas entre la dama que realiza la captura y la pieza capturada. El movimiento tampoco tiene por qué terminar en la casilla inmediatamente detrás de la ficha capturada. Lo que nunca puede hacer la dama es saltar por encima de sus propias piezas, y nunca puede capturar dos piezas contrarias situadas en casillas adyacentes entre sí. La Figura 2.7 muestra un ejemplo de una captura simple realizada por una dama.

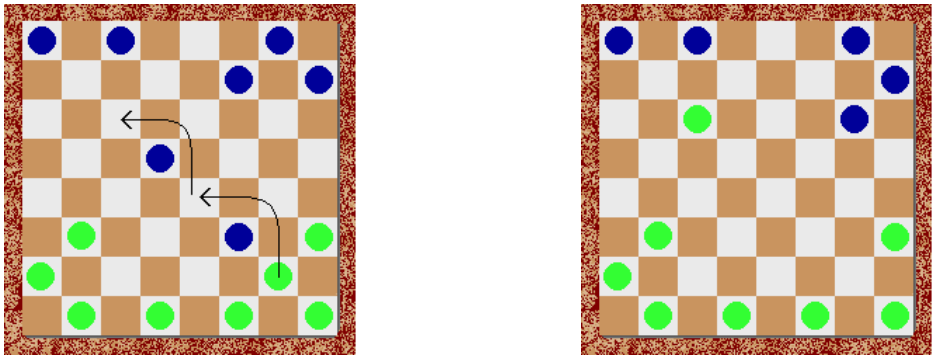


a) b)

Figura 2.7 Captura simple con dama. a) Posición inicial. b) Posición final tras el movimiento

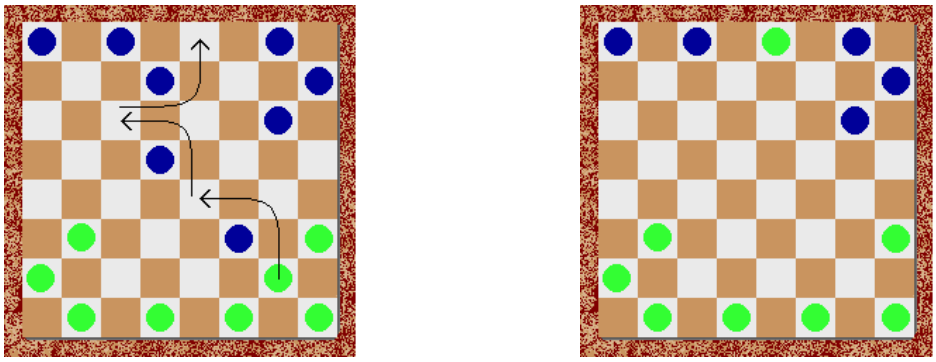
Tanto la dama como el peón, si tras una captura, la pieza en cuestión estuviera en situación de realizar una nueva, ésta se llevará a cabo de forma encadenada, y así

sucesivamente mientras se diera tal circunstancia de poder seguir capturando. Su movimiento y su turno terminan cuando ya no hay más piezas para capturar. Las figuras 2.8 y 2.9 muestran ejemplos de capturas múltiples realizadas por un peón. En la Figura 2.8 las fichas capturadas están en la misma diagonal, mientras que la Figura 2.9 muestra un ejemplo en el que la trayectoria de la ficha que captura no es recta.



a) b)

Figura 2.8 Captura múltiple en una misma diagonal. a) Posición inicial. b) Posición final tras el movimiento



a) b)

Figura 2.9 Captura múltiple en zig-zag. a) Posición inicial. b) Posición final tras el movimiento

2.2.4 Otras reglas

A continuación se describen las reglas:

-Si se toca una pieza y se puede jugar, tiene que ser jugada.

- Obligación de captura a la cantidad: En su turno si un jugador dispone de dos o más opciones de captura, deberá optar por aquella que capture mayor número de piezas contrarias. La captura es obligatoria, es decir, si al llegar el

turno de un jugador, una o más de sus piezas estuvieran en situación de realizar capturas, será obligatorio mover ésta o una de estas piezas y realizar tal captura, no pudiendo optar por mover una pieza que no esté en situación de realizar captura.

- Obligación de captura a la calidad: Si en el caso anterior con dos o más piezas se capturara el mismo número de piezas contrarias, deberá optar por mover aquella con la que capture piezas de mayor valor (la mayor cantidad de damas posibles).

- Soplar: Cuando una ficha en su turno tiene la posibilidad de realizar una captura pero no la lleva a cabo, se retiraría dicha ficha del juego. Esta regla no se utiliza en los torneos.

2.3 Recursos hardware

En el laboratorio docente, los alumnos disponen de tarjetas de experimentación de Digilent Inc, las cuales llevan incorporadas una FPGA XC3S200 de la familia Spartan-3. Con una de estas placas se ha desarrollado nuestro proyecto junto con una pantalla VGA para visualizar el tablero y un teclado para la introducción de los movimientos.

2.3.1 Placa FPGA

Una FPGA (Field Programmable Gate Arrays) es un dispositivo lógico programable por el usuario, compuesto por bloques configurables comunicados entre si. Por lo tanto lo que programamos en una FPGA son los conmutadores que sirven para realizar las conexiones entre los diferentes bloques, más la configuración de los bloques.

Las FPGA pueden ser reprogramadas (aunque en función de la tecnología utilizada para su implementación hay algunos tipos que sólo pueden programarse una única vez). En la mayoría de las FGPA la configuración es volátil (característica que también depende de la tecnología de implementación) por lo que si hay un corte de energía se tiene que volver a reprogramar.

Existen varios tipos de FPGA en el mercado que difieren principalmente en su arquitectura interna, lo que también define su programación y eficiencia (retardos, área/volumen y costos).

La arquitectura Spartan-3 se basa en 3 bloques funcionales programables, estos son:

1. CLBs (Configurable Logic Blocks) que contienen tablas de búsqueda (look-up tables, o LUTS) para implementar elementos lógicos (como funciones booleanas) y de almacenamiento (como flip-flops o latches).
2. Bloques de entrada y salida (IOB, Input/Output Block) que controlan los datos entre los pines de entrada/salida y la lógica interna. Cada IOB es bidireccional y soporta operaciones de tercer estado.
3. RAM para almacenamiento de bloques de datos de 18-Kbit.

La Figura 2.10 resume la interconexión de estos bloques funcionales. Una descripción detallada se encuentra en la hoja de datos proporcionada por el fabricante [5].

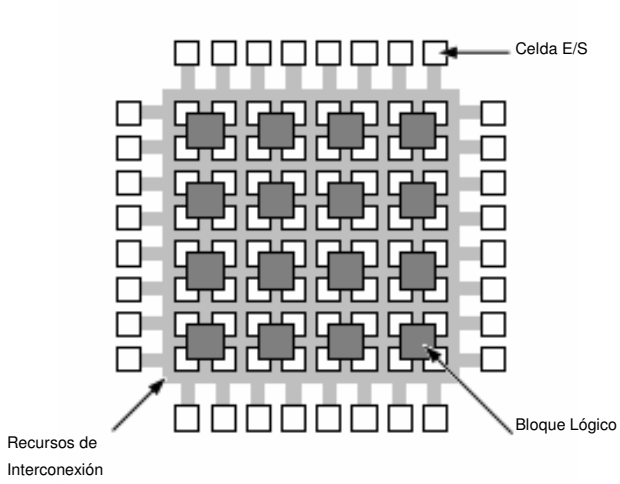


Figura 2.10 Arquitectura Interna de una FPGA de la Familia Spartan-3 de Xilinx

Para la realización del proyecto se usará una tarjeta de evaluación Spartan-3 de Digilent, que contiene una FPGA Xilins XC3S200, como vemos a continuación en la Figura 2.11.

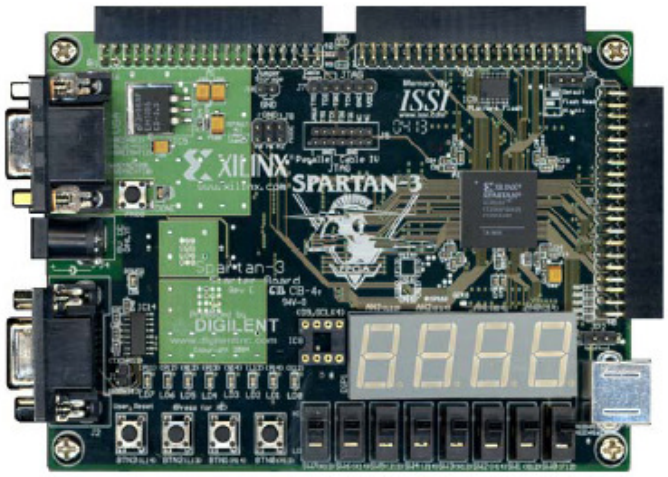


Figura 2.11 Placa de evaluación

Su diagrama de bloques es el siguiente, ver Figura 2.12.

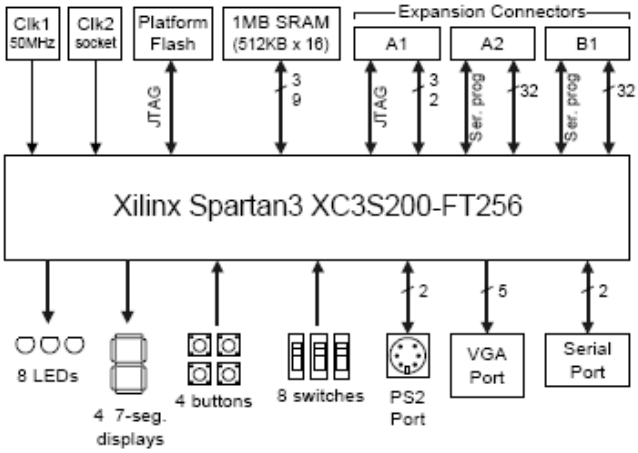


Figura 2.12 Diagrama de bloques de la placa de evaluación

A partir de los dispositivos e interfaces disponibles en la placa de evaluación que se va a utilizar se realizó la elección de los elementos necesarios para la implementación del sistema:

- Puerto VGA: Para Visualizar en una pantalla VGA el tablero y por lo tanto el desarrollo del juego.
- Pulsador: Utilizaremos un pulsador para el reseteo del sistema (Reset).
- Interruptores: Se utilizaran cuatro interruptores, uno para la puesta en marcha (PM), otro para cuando se quiere volver a comer con una dama (comer doble) y

por ultimo dos interruptores, uno por jugador, para indicar tablas (Tablas J1 y Tablas J2).

- Conector de expansión: Utilizaremos 8 pines para comunicarnos con el teclado. Cuatro para las filas y otras cuatro para las columnas. Se decidió utilizar un teclado para indicar las coordenadas de origen y destino de las fichas en el tablero.

A continuación veremos la situación de estos elementos y su conexión a la placa de experimentación, Figura 2.13.

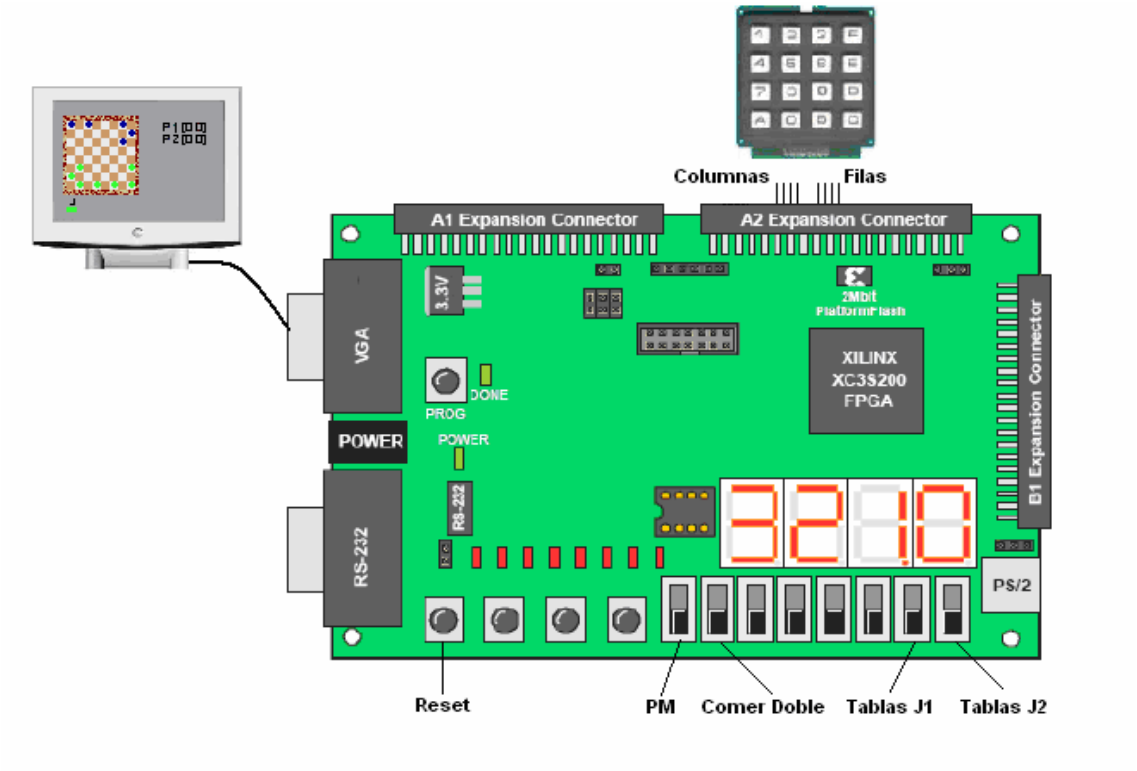


Figura 2.13 Conexionado placa de experimentación

2.3.2 Teclado

El teclado utilizado para este proyecto es un teclado pasivo matricial, el cual ofrece en ocho líneas toda la información necesaria para identificar cual de las dieciséis teclas de que dispone ha sido pulsada.

Para el proyecto solo se han utilizado once teclas (los números del cero al nueve y la letra 'A' que la usaremos como un "enter" cuando se introduzcan los valores de las coordenadas).

Brevemente, pasamos ahora a describir las características más relevantes. El esquema de este tipo de teclados se presenta en la siguiente figura:

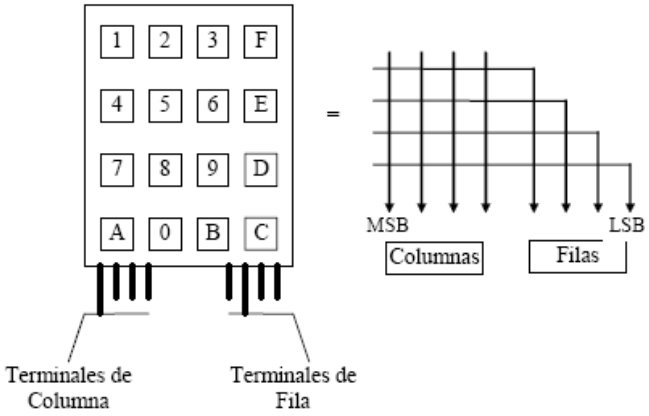


Figura 2.14 Teclado

Como se puede observar, de las ocho líneas que ofrece el teclado, cuatro dan acceso a las columnas (pines 7 a 4), y cuatro dan acceso a las filas (pines 3 a 0). Es importante destacar que el hecho de pulsar una tecla sólo implica que se cortocircuita la línea de la fila con la línea de la columna correspondiente. Es por ello por lo que se ha diseñado un modulo llamado 'tecla', el cual tiene un papel activo en la lectura, realizando las siguientes tareas:

- Identificar la tecla pulsada
- Evitar los rebotes

2.3.3 VGA

Para dibujar una imagen en la pantalla, hay que realizar un barrido de líneas, empezando por la esquina superior izquierda y barriendo hacia la derecha y hacia abajo. Es decir, el monitor va dibujando los datos en líneas horizontales, y después de cada línea, es necesario dar un pulso a la señal de sincronismo horizontal HSYNC (Figura 2.15), y cuando están dibujadas todas las líneas horizontales, hay que dar un pulso a la señal de sincronismo vertical VSYNC (Figura 2.16).

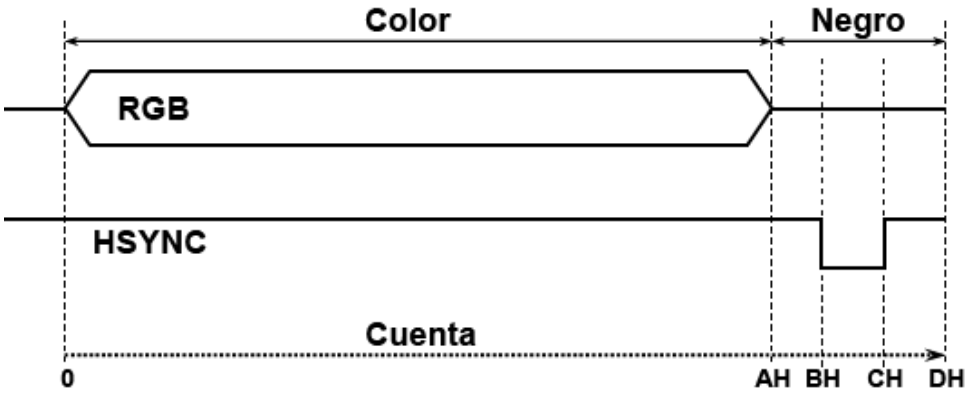


Figura 2.15 Sincronismo horizontal mediante la señal HSYNC

Para implementar el barrido de cada línea, se utilizará un contador (contx), que se reiniciará al comienzo de cada una, y que permitirá temporizar la señal HSYNC.

El proceso de barrido de línea, se repite tantas veces como líneas haya en la pantalla, y después se procede a realizar el sincronismo vertical (conty). La señal VSYNC, tiene una forma equivalente a HSYNC, pero el pulso se produce tras haberse barrido todas las líneas horizontales.

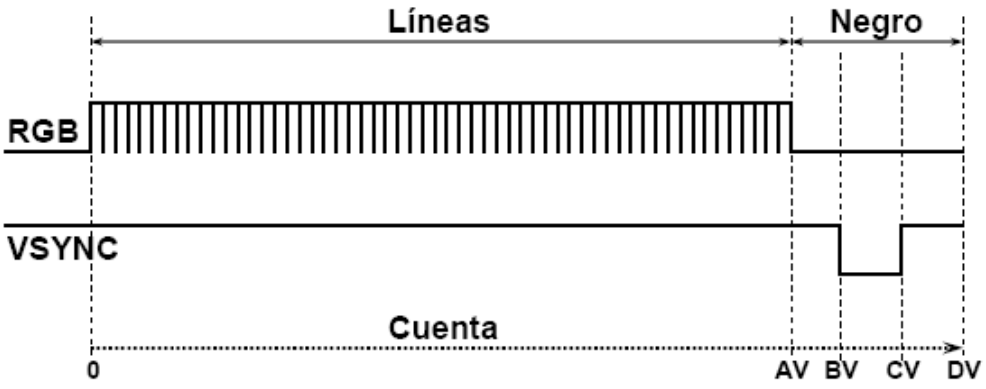


Figura 2.16 Sincronismo vertical mediante la señal VSYNC

La resolución que se va a utilizar será de 640x480, con una frecuencia de actualización de 60Hz. Para esta resolución y suponiendo una frecuencia de reloj de 25MHz, los tiempos correspondientes a cada evento y sus valores se muestran en la Tabla 1.

Horizontal			Vertical		
	Tiempo	**Puntos a recorrer**		**Tiempo**	**Líneas a recorrer**
AH	25,60 us	640	**AV**	15,360 ms	480
BH	26,16 us	654	**BV**	15,680 ms	490
CH	30,08 us	752	**CV**	15,744 ms	492
DH	32,00 us	800	**DV**	16,672 ms	521

Tabla 1 Registro de tiempos

Puesto que la placa Xilinx, tiene un reloj de 50 MHz y los cálculos de tiempo están realizados para 25MHz, el contador utilizado para implementar el sincronismo horizontal sólo deberá activarse uno de cada dos ciclos de reloj, y ello se lleva acabo en el modulo 'div_frecuencia' (Véase el capítulo 3).

3 Descripción del diseño

Antes de comenzar con la descripción del circuito se van a comentar, las especificaciones concretas establecidas que deben implementarse.

El tablero del juego de las damas consta de ocho filas y ocho columnas. Para indicar la casilla de origen o la de destino a la cual nos queremos mover, se ha decidido utilizar el teclado para introducir las coordenadas (fila, columna) siguiendo la numeración de la Figura 3.1.

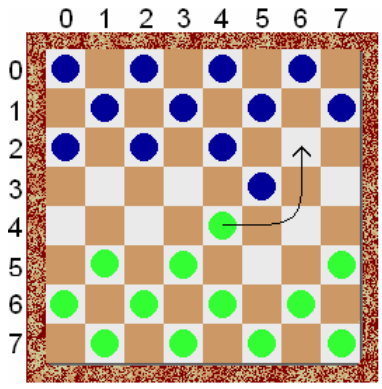


Figura 3.1 Tablero con coordenadas

A la hora de introducir dichas coordenadas hay que tener en cuenta que se debe pulsar la letra 'A' del teclado para confirmar el valor pulsado, es decir, la secuencia a la hora de introducir datos seria le siguiente:

Coordenada origen: 4 + A + 4 + A

Coordenada destino: 2 + A +6 + A

El funcionamiento del juego es el siguiente:

Primeramente el jugador del turno correspondiente introducirá las coordenadas de la ficha a desplazar, si esta ficha no tiene la posibilidad de moverse, se ha elegido

una ficha que no es del turno o se trata de una casilla vacía, el sistema no aceptará el valor y habrá que volver a introducir nuevas coordenadas.

Una vez aceptada la coordenada origen se introducirá la coordenada destino y el sistema verificará que el movimiento pedido es factible, si no lo fuera, habría que volver a introducir solo la coordenada destino, ya que una de las reglas es que ficha tocada ficha que hay que mover.

A la hora de realizar capturas múltiples en diferentes direcciones con una dama, hay que ir indicando los movimientos intermedios y activar el interruptor de comer múltiple. Ver Figura 3.2.

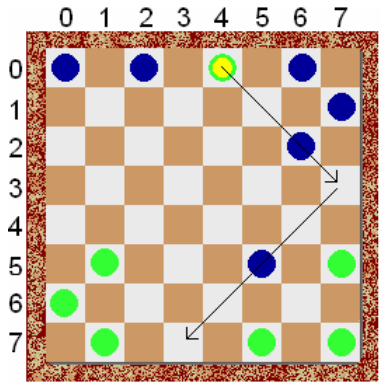


Figura 3.2 Captura múltiple con dama

La secuencia de entrada de datos para realizar el movimiento mostrado en la Figura 3.2 sería:

Coordenada origen: 0 + A + 4 + A

Coordenada destino: 3 + A + 7 + A (*)

Coordenada origen: la introduce el sistema →(3, 7)

Coordenada destino: 7 + A + 3 + A

(*) Hay que pulsar el interruptor de captura múltiple, antes de pulsar por segunda vez la letra ‘A’ en la coordenada destino.

La victoria se alcanza cuando uno de los jugadores ha capturado todas las fichas del contrincante.

Por ultimo comentar, que cuando ambos jugadores acuerdan terminar en tablas, cada uno debe pulsar su interruptor de tablas y al finalizar el movimiento del turno correspondiente aparecerán dos iconos para indicar que la partida ha terminado en tablas.

3.1 Sistema completo

En este apartado se describe en detalle la jerarquía del sistema completo y su interfaz. El sistema completo engloba a todos los componentes que han sido necesarios para la realización de este juego, alguno de estos bloques ya sea de forma parcial o total se pueden utilizar en la realización de otros proyectos de similares características.

3.1.1 Interfaz

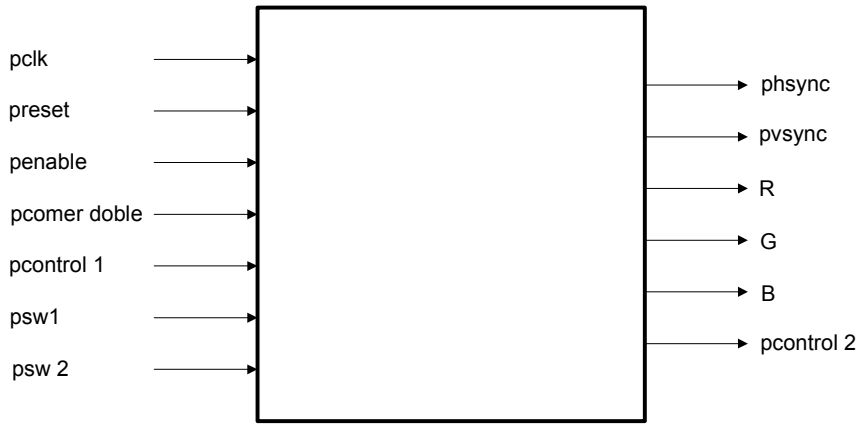


Figura 3.3 Interfaz Sistema Completo

Entradas:

-pclk: Señal de reloj.

-preset: Reinicio del sistema.

-penable: Interruptor de marcha.

-pcomer_doble: Interruptor para comer múltiple con la dama.

-pcontrol1: Valor de la columna pulsada (4 bits).

-psw1: Interruptor de tablas para jugador 1.

-psw2: Interruptor de tablas para jugador 2.

Salidas:

-phsync: Salida para el sincronismo horizontal.

-pvsync: Salida para el sincronismo vertical.

-R: Salida para el color rojo.

-G: Salida para el color verde.

-B: Salida para el color azul.

-pcontrol2: Valor de fila a comprobar (4 bits).

3.1.2 Diagrama de bloques

La Figura 3.4 muestra el diagrama de bloques del sistema desarrollado. El sistema está compuesto de cinco bloques claramente diferenciados:

- El bloque *VGA* se encarga de controlar la interfaz con la pantalla para representar la imagen necesaria.
- El bloque *Tablero* es el módulo que define qué debe representarse en cada momento en la pantalla.
- El bloque *Tecla* se corresponde a la parte del circuito encarga de decodificar el teclado matricial y evitar el efecto de posibles rebotes en las pulsaciones de las teclas.
- El bloque *RAM* se encarga de almacenar el contenido del tablero en cada momento.
- El bloque *Árbitro* controla el juego.

Una descripción detallada de cada bloque se realiza en los siguientes apartados.

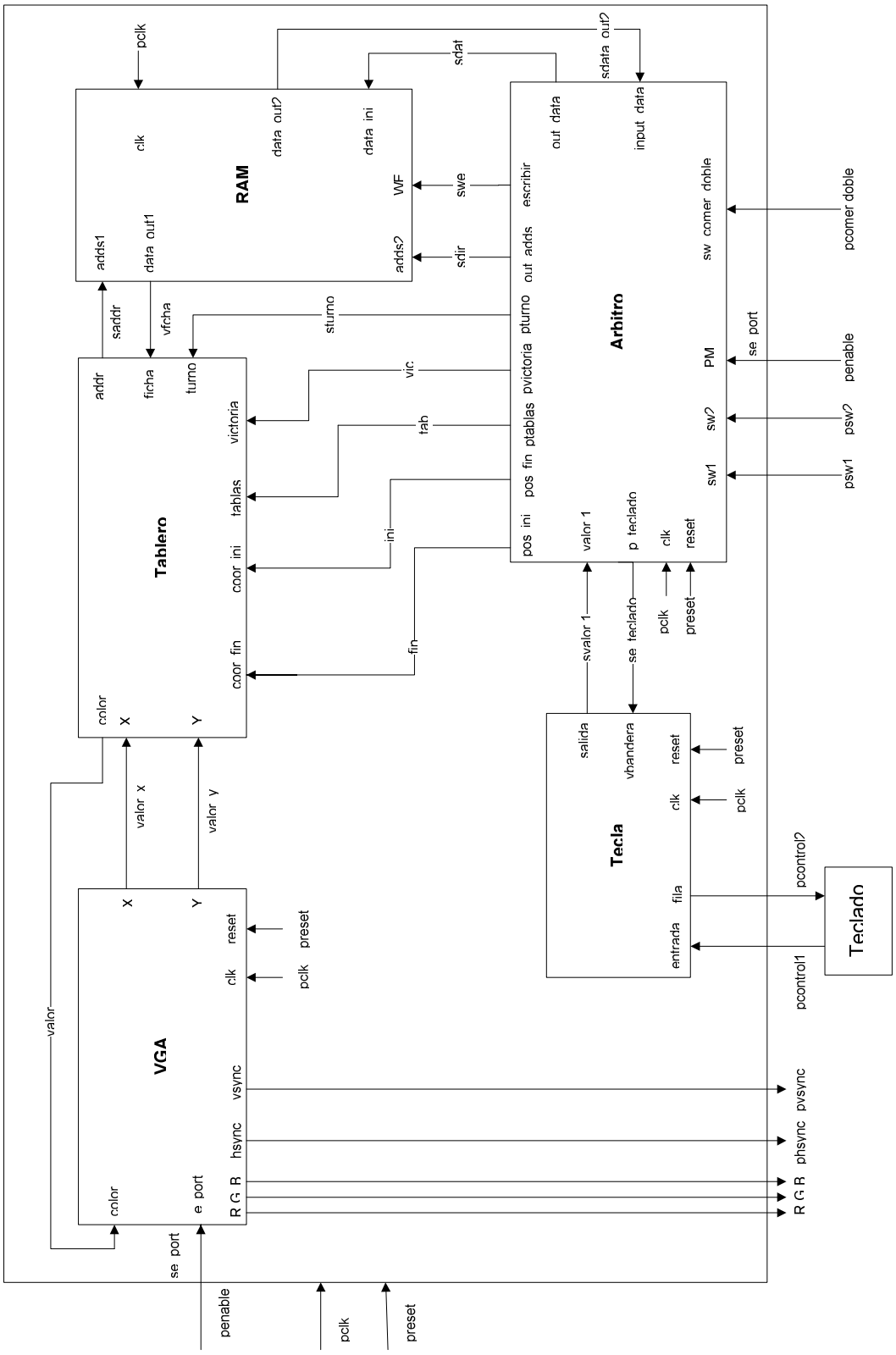


Figura 3.4 Diagrama de bloques del Sistema Completo

3.2 Árbitro

El bloque árbitro es el motor principal del juego, con él controlamos al resto de módulos, exceptuando al de VGA que realiza tareas independientes en paralelo. Este módulo está compuesto por cinco máquinas de estado, ver Figura 3.6, una principal o máquina padre y cuatro máquinas secundarias, o máquinas hijo.

La máquina padre, es la encargada de regular las principales acciones del Árbitro, y el nombre que se le a ha dado es 'fsm1'.

Las máquinas hijo, se encargan de distintas verificaciones, tales como comprobar que la ficha seleccionada es correcta o que el movimiento pedido por el jugador es factible. De las cuatro máquinas, hay dos dedicadas (fsm2 y fsm3) a verificaciones que hacen referencia a los peones y otras dos (fsm4 y fsm5) que son para las damas.

3.2.1 Interfaz

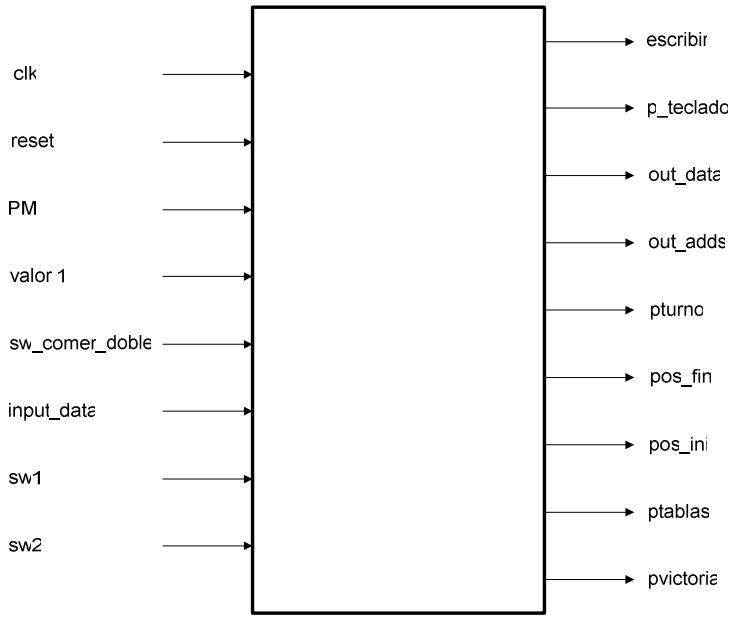


Figura 3.5 Interfaz Árbitro

Entradas:

-clk: Señal de reloj.

-reset: Reinicio del sistema.

-PM: Pulsador de marcha.

-input_data: Valor de la casilla (3bits).

-valor1: Informa de que tecla ha sido pulsada (4 bits).

-sw1: Indica que el jugador 1 quiere tablas.

-sw2: Indica que el jugador 2 quiere tablas.

-sw_comer_doble: Informa de captura múltiple con una dama.

Salidas:

-out_data: Valor que se le va a dar a la casilla (3bits).

-out_adds: Vector de dirección de la casilla a modificar o consultar (6 bits).

-escribir: Permite la escritura en el bloque RAM.

-p_turno: Informa del turno.

-p_teclado: Permite la lectura de la tecla pulsada.

-p_tablas: Informa a 'tablero' que la partida ha concluido en tablas.

-p_victoria: Informa a 'tablero' que la partida ha concluido con victoria de alguno de los dos jugadores.

-pos_ini: Valor coordenada origen (6 bits).

-pos_fin: Valor coordenada destino (6 bits).

3.2.2 Diagrama de bloques

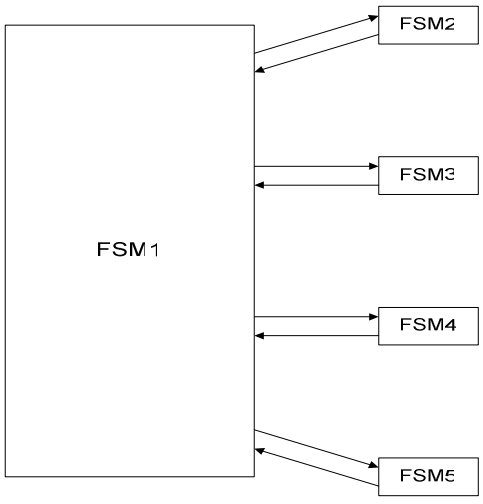


Figura 3.6 Diagrama de bloques Máquina Principal

3.2.3 Componentes

3.2.3.1 FSM1

El módulo 'fsm1' es la máquina de estados principal, la cual se encarga de controlar la evolución del juego, ya sea comunicándose con otras máquinas de estado pidiéndoles

información, (es decir, preguntándoles si las coordenadas introducidas por el jugador son correctas o no lo son), o comprobando si la partida ha finalizado tras unas tablas o una victoria de alguno de los jugadores.

3.2.3.1.1 Interfaz

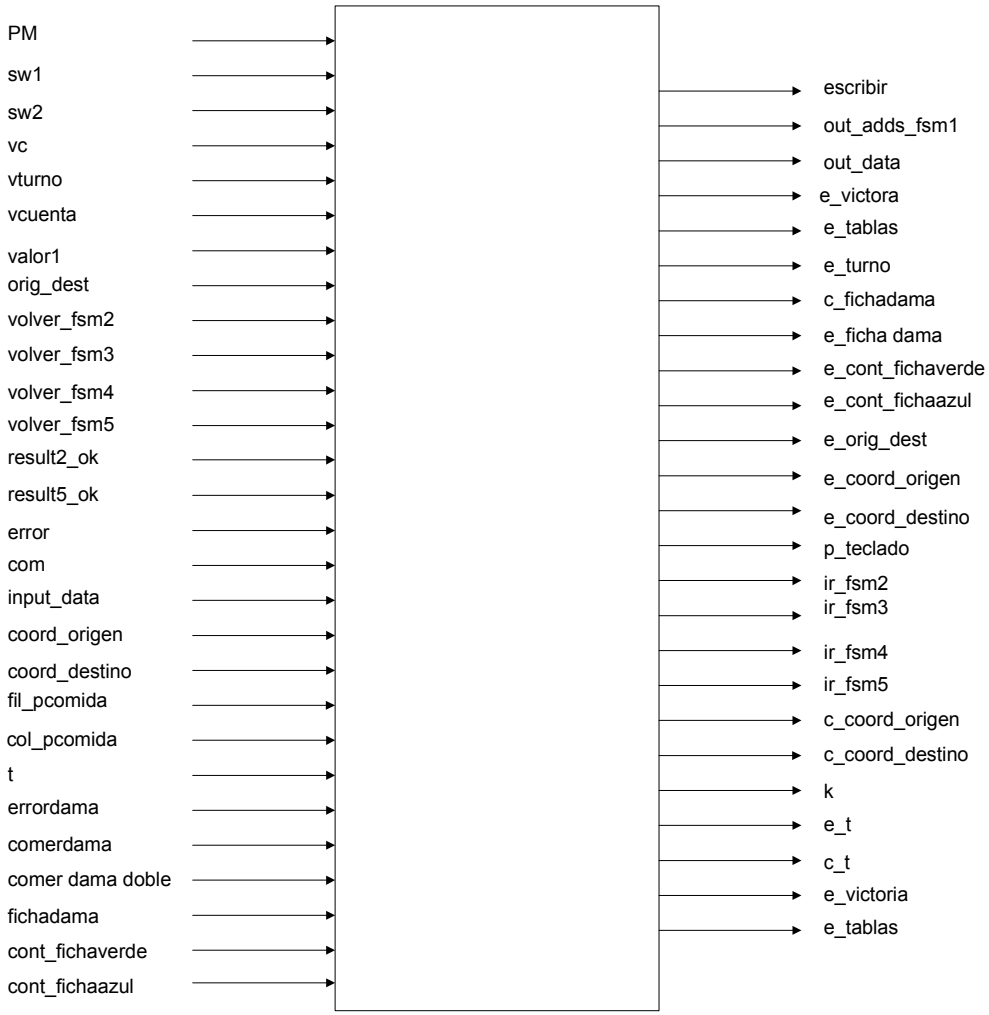


Figura 3.7 Interfaz FSM1

Entradas:

-PM: Pulsador de marcha del sistema.

-clk: Señal de reloj.

-reset: Reinicio del sistema.

-sw1: Interruptor de jugador 1 para aceptar tablas.

-sw2: Interruptor de jugador 2 para aceptar tablas.

-sw_comer_doble: Interruptor para indicar que se desea hacer un movimiento de captura múltiple con la dama.
-input_data: Informa del contenido de la casilla (3 bits).
-valor1: Informa del valor de la tecla pulsada (4 bits).
-vc: 'flag' de fin de cuenta de la señal vcuenta (se han recorrido todas las casillas del tablero).
-vcuenta: Contador de casillas.
-vturno: Indica el turno (ficha verde=1, ficha azul=0).
-orig_dest: Selector de coordenada (origen=0, destino=1).
-volver_fsm2: Señal que indica la finalización de operaciones en la máquina secundaria.
-volver_fsm3: Señal que indica la finalización de operaciones en la máquina secundaria.
-volver_fsm4: Señal que indica la finalización de operaciones en la máquina secundaria.
-volver_fsm5: Señal que indica la finalización de operaciones en la máquina secundaria.
-result2_ok: Resultado de 'fsm2' (ficha correcta=1, ficha incorrecta=0), con ella terminan todas las comprobaciones.
-result5_ok: Resultado de 'fsm5' (ficha correcta=1, ficha incorrecta=0), con ella terminan todas las comprobaciones.
-error: Informa de si el movimiento con el peón se ha podido realizar (movimiento erróneo=1, movimiento factible=0).
-com: Informa de si el peón en su movimiento a realizado alguna captura (captura si=1, captura no=0).
-coord_origen: Valor de fila y columna de la posición origen (6bits).
-coord_destino: Valor de fila y columna de la posición destino (6bits).
-fil_pcomida: Vector que indica la fila, de las fichas capturadas (9bits).
-col_pcomida: Vector que indica la columna, de las fichas capturadas (9bits).
-t: Contador de fichas eliminadas.
-errordama: Informa de si el movimiento con la dama se ha podido realizar (movimiento erróneo=1, movimiento factible=0).
-comerdama: Informa de si la dama en su movimiento a realizado alguna captura (captura si=1, captura no=0).
-comerdamadoble: Informa de que la dama puede volver a capturar.
-fichadama: informa que la ficha es una dama.

-cont_fichaverde: Contador de las fichas verdes.

-cont_fichaazul: Contador de las fichas azules.

Salidas:

-out_adds_fsm1: Vector de dirección de la casilla a modificar o consultar (6 bits).

-out_data: Valor que se le va a dar a la casilla (3bits).

-escribir: Habilita la escritura en la RAM

-e_victorias: Se activa cuando la partida ha concluido y hay victoria por parte de alguno de los jugadores.

-e_tablas: Se activa cuando la partida ha concluido con resultado de tablas.

-e_turno: para cambiar de turno.

-c_fichadama: Pone a cero la señal 'fichadama'.

-e_fichadama: Pone a uno la señal 'fichadama'.

-e_cont_fichaverde: Resta uno al valor de 'cont_fichaverde'.

-e_cont_fichaazul: Resta uno al valor de 'cont_fichaazul'.

-e_orig_dest: Señal de habilitación del registro orig_dest, que almacena si las coordenadas introducidas se corresponden con la posición de origen o la de destino.

-e_coord_origen: Carga el valor de fila o columna origen en la señal 'coord origen'.

-e_coord_destino: Carga el valor de fila o columna destino en la señal 'coord destino'.

-c_coord_origen: Pone a cero a la señal 'coord_origen'.

-c_coord_destino: Pone a cero a la señal 'coord_destino'.

-k: Selector de fila/columna (k=1 "fila", k=0 "columna").

-ir_fsm2: Señal para activar 'fsm2'.

-ir_fsm3: Señal para activar 'fsm3'.

-ir_fsm4: Señal para activar 'fsm4'.

-ir_fsm5: Señal para activar 'fsm5'.

-pteclado: permite recoger datos del bloque 'tecla'.

-e_t: Habilita el contador 't'.

-c_t: Deshabilita al contador 't'.

3.2.3.1.2 Descripción detallada

Esta máquina está compuesta por 18 estados y su funcionalidad es la siguiente:

-Reposo: es el estado inicial del sistema y su función, es la de esperar a que se active el interruptor de inicio de partida o pulsador de marcha.

-Inicio: el circuito permanece en este estado hasta que se rellene la memoria RAM con los datos de inicio de partida. Es decir, se van a colocar las fichas en su posición inicial dentro del tablero, que se consigue gracias a un contador utilizado para ir casilla por casilla del tablero. Cuando lleguemos a la última casilla, activaremos un flag indicando que la introducción de datos en la memoria se ha terminado.

-Turno: se utiliza para cambiar de jugador y para comprobar si la partida ha finalizado ya sea por una victoria o por un empate.

-Introducción de coordenadas: Esta función engloba a los estados, e1, e2, e3 y e4.

Los estados e1 y e3, se usan para recoger los valores de fila y columna de las coordenadas origen o destino. Dichos estados sólo recogen valores del cero al siete, por lo tanto, cualquier otro valor o tecla pulsada, no se tendrá en cuenta. Los otros dos estados, e2 y e4, se usan para confirmar una entrada, como un “intro” en un teclado convencional de ordenador. La tecla usada para dicho fin es la letra ‘A’.

-Estado e5: En él se elige a qué máquina de estados secundaria se va a ir.

-Estados e6, e7, e8 y e9: Son los encargados de llamar a las máquinas hijo del sistema y de comprobar los valores que estas devuelven.

-Estado e10: Borra la ficha de su posición de origen. Para ello le mandamos a la memoria de juego (a partir de ahora utilizaremos la palabra RAM para referirnos a dicha memoria) la posición y el valor de ‘casilla vacía’.

-Estado e11: Se convierte al peón que ha llegado a la primera fila del bando contrario en dama.

-Estado e12: Se encarga de escribir en la ‘RAM’ la nueva posición que va a ocupar la ficha después de realizar el movimiento.

-Estado e13: Se comprueba si la ficha ha realizado alguna captura en el transcurso del movimiento, si es así avanza al estado ‘e14’, si no salta a ‘turno’ para cambiar de jugador.

-Estado e14: Borra las fichas que se han capturado. Para ello se asigna el valor de casilla vacía a la dirección de la ficha comida en la RAM. Una vez borradas todas las fichas capturadas, hay que tener en cuenta lo siguiente:

-Si la ficha con la que hemos comido es una dama, y hay posibilidades de volver a comer (ver fsm4) entonces nos vamos al estado 'e1' para seguir jugando con esa ficha, y si no, nos vamos al estado 'turno' para cambiar de jugador.

-Si la ficha con la que jugamos es un peón, siempre nos vamos a 'turno'.

-Estado fin_partida: Se llega a él cuando hay un vencedor o cuando ha habido tablas.

3.2.3.2 FSM2

Máquina de estados secundaria, que se ocupa de verificar que la coordenada origen introducida por el jugador es correcta. Para ello, comprobamos que las coordenadas introducidas corresponden a una casilla que tiene ficha, y que esta pertenece al jugador que tiene el turno. A continuación, pasamos a la verificación de movimiento, que consiste en comprobar que la ficha seleccionada tiene algún movimiento factible, ya sea desplazándose a una casilla adyacente, o capturando una ficha del contrincante.

3.2.3.2.1 Interfaz

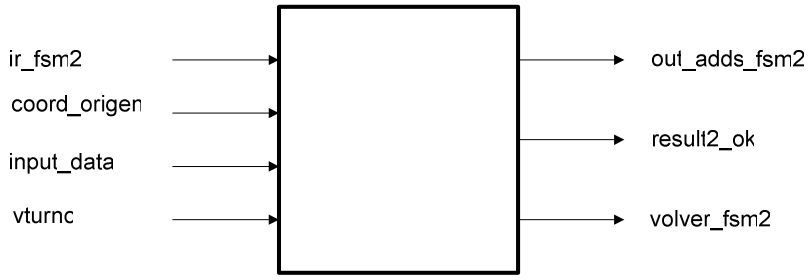


Figura 3.8 Interfaz FSM2

Entradas:

-ir_fsm2: Señal para activar a fsm2.

-coord_origen: Valor de fila y columna de la posición inicial (6 bits).

-input_data: Informa del contenido de la casilla (3 bits).

-vturno: Informa que jugador tiene el turno.

Salidas:

-out_adds_fsm2: Vector de dirección de la casilla a consultar (6 bits).

-result2_ok: Resultado de las comprobaciones.

-volver_fsm2: Indica el fin de las comprobaciones.

3.2.3.2.2 Descripción detallada

-Estado rep2: Es el estado de reposo de 'fsm2'.

-Estado comp0: Comprueba que las coordenadas introducidas son correctas, es decir, verifica que dichas coordenadas pertenecen a una casilla del tablero ocupada por un peón del turno correspondiente. Si esto no se diera, volveríamos a 'fsm1' para que el jugador pueda introducir nuevos valores.

-Estados comp1, comp2, comp3 y comp4: Se utilizan para simular posibles movimientos de la ficha elegida, si hay algún movimiento factible se informa de ello a 'fsm1' para que se introduzcan las coordenadas de destino y si no hay ningún movimiento factible, se regresa a 'fsm1' para que el jugador pueda introducir nuevamente unas coordenadas de origen.

Los estados comp1 y comp2 se ocupan de los desplazamientos simples, y comp3 y comp4 de las capturas.

-Estados fin_false y fin_true: Los utilizamos para volver a la máquina padre e informar de los resultados obtenidos en las diversas verificaciones realizadas.

3.2.3.3 FSM3

FSM3 es una máquina de estados secundaria que se encarga de verificar que las coordenadas de destino introducidas por el jugador son correctas, es decir, comprueba que el movimiento pedido para la ficha elegida anteriormente es válido. Para ello distinguiremos entre varios movimientos:

-Movimiento simple: desplazamiento de una ficha a una casilla adyacente vacía.

-Captura simple: desplazamiento de dos casillas, comiéndose a una ficha contraria.

-Captura múltiple: Desplazamiento de una ficha realizando capturas simples de forma consecutiva, sólo se podrá hacer un máximo de tres capturas.

Para realizar las capturas, se han creado dos vectores que indican el tipo de movimiento. Uno para el de tres capturas (vector_3mov) de tres bits y otro para el de dos capturas (vector_2mov) de dos bits. Cada bit indica el tipo de movimiento, es decir, si el desplazamiento es hacia la izquierda o hacia la derecha. Esto se representa de la siguiente manera:

-0→ movimiento a la derecha.

-1→ movimiento a la izquierda.

En las siguientes tablas, se indican los movimientos posibles para realizar una captura, según el vector.

En el caso de capturas triples la ficha se moverá siempre seis filas, y dependiendo de las coordenadas finales de la ficha, se desplazará dos columnas (en zig-zag) o seis columnas (en diagonal). Ver Tabla 2.

TRES capturas (6 filas)			
Movimiento Izquierda		Movimiento Derecha	
2 Columnas	(110)	2 Columnas	(001)
	(101)		(010)
	(011)		(100)
6 Columnas	(111)	6 Columnas	(000)

Tabla 2 Triple captura

En la Tabla 3, se ve que en las capturas dobles los movimientos serán siempre de 4 filas y al igual que en la anterior tabla, según las coordenadas finales, nos moveremos cuatro columnas o ninguna.

DOS capturas (4 filas)					
Movimiento Izquierda		Movimiento Derecha		Movimiento Centro	
4 Columnas	(11)	4 Columnas	(00)	0 Columnas	(10), (01)

Tabla 3 Doble captura

3.2.3.3.1 Interfaz

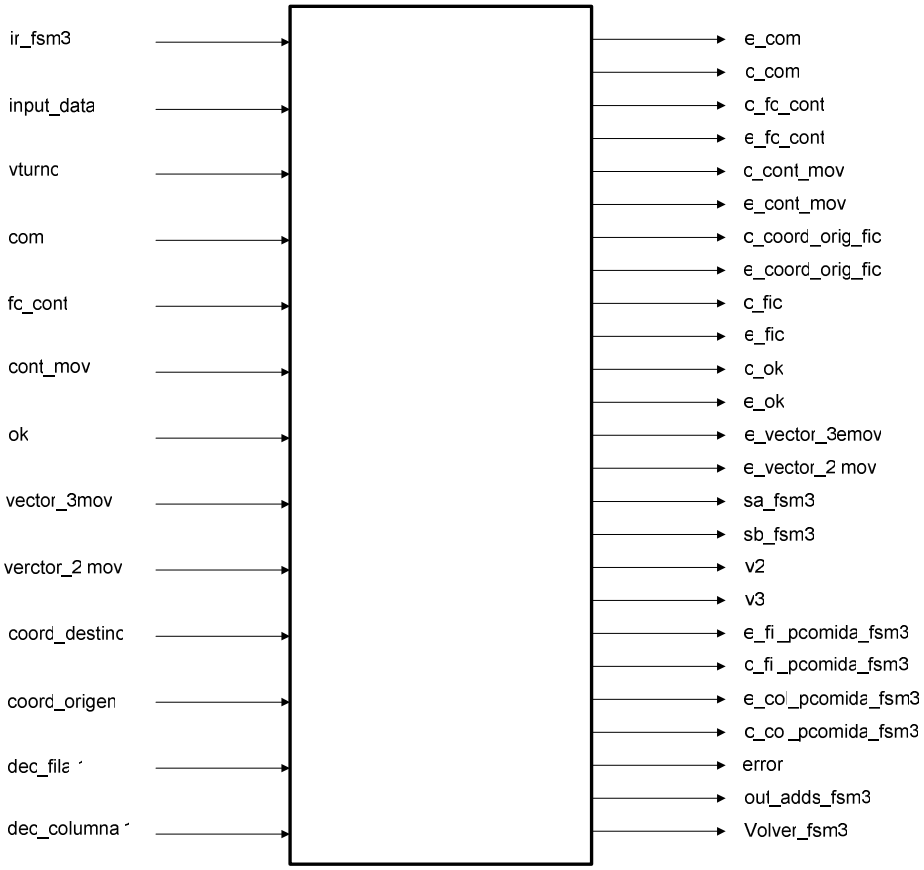


Figura 3.9 Interfaz FSM3

Entradas:

-ir_fsm3: Señal para activar a fsm3.

-input_data: Informa del contenido de la casilla (3 bits).

-vturno: Informa de qué jugador tiene el turno.

-coord_origen: Valor de fila y columna de la posición inicial (6 bits).

-coord_destino: Valor de fila y columna de la posición final (6 bits).

-dec_fila1: Valor decimal de fila de la señal 'coord_orig_fic'.

-dec_columna1: Valor decimal de columna de la señal 'coord_orig_fic'

-com: Se activa cuando se produce una captura.

-fc_cont: contador que usamos para recorrer los vectores (vector_3mov y vector_2mov).

-cont_mov: Contador de trayectorias. Ver Tabla 5.

-ok: movimiento de captura intermedio correcto.

-vector_3mov: Vector de movimiento para captura triple. Ver Tabla 2.

-vector_2mov: Vector de movimiento para captura doble. Ver Tabla 3.

Salidas:

-out_adds_fsm3: Vector de dirección de la casilla a consultar (6 bits).

-error: informa a fsm1 de que la dirección de destino no es correcta.

-e_com: Señal de habilitación del biestable 'com' que almacena si el movimiento es de captura.

-c_com: Inicializa a 0 el biestable 'com'.

-c_fc_cont: Inicializa la cuenta del biestable 'fc_cont'.

-e_fc_cont: Habilita la cuenta del biestable 'fc_cont'.

-c_cont_mov: Inicializa a 1 la cuenta del biestable 'cont_mov'.

-e_cont_mov: Habilita la cuenta del biestable 'cont_mov'.

-c_coord_orig_fic: Inicializa el registro 'coord_orig_fic' almacenando el valor de 'coord_origen'.

-e_coord_orig_fic: Habilita el registro 'coord_orig_fic' almacenando el valor de 'fic'.

-c_fic: Inicializa a 0 el biestable 'fic'.

-e_fic: Habilita el registro 'fic' almacenando el valor concatenado de 'sa' y sb'.

-c_ok: Inicializa a 0 el biestable 'ok'.

-e_ok: Señal de habilitación del biestable 'ok' que almacena si el desplazamiento intermedio realizado es correcto.

-e_vector_3mov: Habilita el registro 'vector_3mov' almacenando el valor de 'v3'.

-e_vector_2mov: Habilita el registro 'vector_2mov' almacenando el valor de 'v2'.

-sa_fsm3: valor de fila de la ficha a comer (3 bits).

-sb_fsm3: valor de columna de la ficha a comer (3 bits).

-v2: Vector para doble captura.

-v3: Vector para la triple captura.

-e_fil_pcomida_fsm3: guarda el valor de fila de la ficha capturada por la dama.

-c_fil_pcomida_fsm3: borra el valor de fila de la ficha capturada por la dama.

-e_col_pcomida_fsm3: guarda el valor de columna de la ficha capturada por la dama.

-c_col_pcomida_fsm3: borra el valor de columna de la ficha capturada por la dama.

-volver_fsm3: Indica el fin de las comprobaciones.

3.2.3.3.2 Descripción detallada

La máquina de estados fsm3 está compuesta por 17 estados.

-Estado rep3: Estado en el que permanece 'fsm3' hasta que es activada por 'fsm1'.

-Estado m1: Se comprueba que la casilla elegida como destino esté vacía, y para ello, mandamos las coordenadas destino a la RAM, y ésta, nos devuelve el dato de la casilla. Si la casilla está vacía continuamos avanzando por los estados para comprobar que el movimiento es factible, por el contrario, si la casilla estuviera ocupada por alguna ficha no hace falta el seguir comprobando, ya que el movimiento va a ser imposible y por lo tanto, tendremos que volver a 'fsm1' para que el jugador pueda introducir nuevas coordenadas de destino.

-Estado m2: Se usa para averiguar el tipo de movimiento. Para ello, restamos la fila origen a la fila destino en el turno de las fichas verdes, y viceversa en el turno de las fichas azules. Este valor, se lo asignamos a la variable 'fil_res' y según el valor de ésta, el movimiento será distinto; ver Tabla 4.

fil_res	Tipo de movimiento
(001)	Simple (m3)
(010)	Captura simple (m4)
>010 y fil_res(0)=0	Capturas múltiples (m5)
Cualquier otro valor	Movimiento erróneo (m14)

Tabla 4 Tipo de movimientos

-Estado m3: Se emplea para el movimiento simple. Este estado se alcanza sólo si la casilla destino está vacía, por lo tanto en este estado sólo se comprueba si el movimiento pedido es hacia la izquierda o hacia la derecha.

-Estado m4: Se utiliza cuando el movimiento implica realizar una captura simple. Para ello, verificamos la dirección del movimiento, es decir, miramos si el desplazamiento es hacia la izquierda o hacia la derecha, esta comprobación es igual que en el estado anterior, únicamente cambiamos el valor del sustraendo o del sumando por dos, ya que el desplazamiento es de dos casillas. Una vez que sabemos la dirección y sentido del movimiento, comprobamos que en la casilla contigua a la de origen, hay una ficha del contrincante. Si esto es así, el movimiento es correcto y se regresa a 'fsm1' para

ejecutar el movimiento, si no también se regresa a 'fsm1' pero para pedir una nueva dirección de destino ya que esta no ha sido correcta.

-Estado m5: Se calcula el número de capturas que se van a dar a lo largo del movimiento pedido y de establecer una ruta para realizar dichas capturas.

El número de capturas y las posibles rutas, las podemos ver a continuación en la Tabla 5.

Nº Capturas	Fila/Columna	Contador (cont_mov)	Turno	Vector	Estado futuro
3	l=6 y j1=2	1	1	v3="110"	m6
			0	v3="001"	
		2	1	v3="101"	m6
			0	v3="010"	
		3	1	v3="011"	m6
			0	v3="100"	
		>3	-	-	m13
3	l=6 y j2=2	1	1	v3="001"	m6
			0	v3="110"	
		2	1	v3="010"	m6
			0	v3="101"	
		3	1	v3="100"	m6
			0	v3="011"	
		>3	-	-	m13
3	l=6 y j1=6	1	1	v3="111"	m6
			0	v3="000"	
		>1	-	-	m13
3	l=6 y j2=6	1	1	v3="000"	m6
			0	v3="111"	
		>1	-	-	m13
2	l=4 y j1=4	1	1	v2="11"	m7
			0	v2="00"	
		>1	-	-	m13
2	l=4 y j2=4	1	1	v2="00"	m7
			0	v2="11"	

		>1	-		m13
2	l=4 y j2=0	1	-	v2=”10”	m7
		2	-	v2=”01”	m7
		>2	-	-	m13

Tabla 5 Nº de capturas

En la Tabla 5 se muestra la siguiente información:

-Nº de capturas: Nos informa del número de fichas que van a ser comidas.

-Fila/Columna:

l: Número de filas que se desplazará el peón en la captura.

j1 y j2: Número de columnas que se moverá el peón en la captura.

-Contador: Esta columna depende del valor de la señal ‘cont_mov’.Dicha señal, es utilizada para saber que ruta se va a comprobar.

-Turno: Indica a que jugador le toca.

1→ Fichas verdes.

0→ Fichas azules.

-Vector: Nos informa del camino que puede recorrer el peón seleccionado.

1→ Movimiento hacia la izquierda.

0→ Movimiento hacia la derecha.

-Estado futuro: Indicamos a qué estado vamos a ir, una vez realizadas todas las comprobaciones anteriores.

-Estado m6: Se encarga de comprobar la dirección del siguiente movimiento intermedio del que consta la captura, en este caso triple. Ver Tabla 5.

-Estado m7: Tiene la misma funcionalidad que el estado anterior, pero ahora es para las trayectorias de dos capturas. Ver Tabla 5.

-Estados m8 y m9: Se emplean para realizar el movimiento hacia la izquierda. Primero, comprobamos que el desplazamiento no se salga del tablero. Seguidamente, se comprueba que dos casillas mas adelante no haya ninguna ficha; para ello mandamos la dirección de la casilla al bloque ‘RAM’ y éste nos devolverá el dato de

dicha casilla. Si la casilla estuviera vacía, se guarda el valor de dicha dirección para volver a usarlo en el siguiente movimiento, como coordenada origen.

A continuación, se comprueba que en la casilla contigua a la de origen hay una ficha del contrincante. Para ello realizamos las mismas operaciones que antes, es decir, le mandamos la dirección de dicha casilla al bloque 'RAM', y éste nos devuelve el valor de la casilla. Si ese valor es el de una ficha contraria, el movimiento será correcto y guardaremos la dirección de la ficha comida para su posterior borrado en 'fsm1'.

-Estados m10 y m11: La funcionalidad de estos dos estados, es la misma que la de 'm8' y 'm9', pero difieren en que se utilizan para el movimiento hacia la derecha.

- Estado m12: Comprueba que el movimiento intermedio a derechas o izquierdas ha sido correcto.

- Estado m13: Se encarga de mirar la señal que nos informa de si el movimiento pedido por el jugador es factible o no, es decir, si las coordenadas de destino eran correctas o no.

-Estados m14, m15 y m16: Los utilizamos para volver a la máquina padre e informar de los resultados obtenidos en las diversas verificaciones realizadas.

-m14: informa de que las coordenadas destino no son correctas.

-m15: informa de que las coordenadas destino son correctas y que el movimiento realizado ha sido un movimiento simple.

-m16: informa que las coordenadas destino son correctas y que al menos ha habido una captura en el transcurso del movimiento.

3.2.3.4 FSM4

FSM4 es una máquina de estados secundaria que se encarga de verificar la posición de destino introducida por el jugador, cuando la ficha movida es una dama.

Se divide en dos bloques, uno donde comprobamos que la coordenada destino introducida por el jugador es correcta (n1, n2, n3, n4, n5 y n6), y otro donde se verifica si la dama puede volver a comer (n7, n8, n9, n10, n11, n12 y n13).

3.2.3.4.1 Interfaz

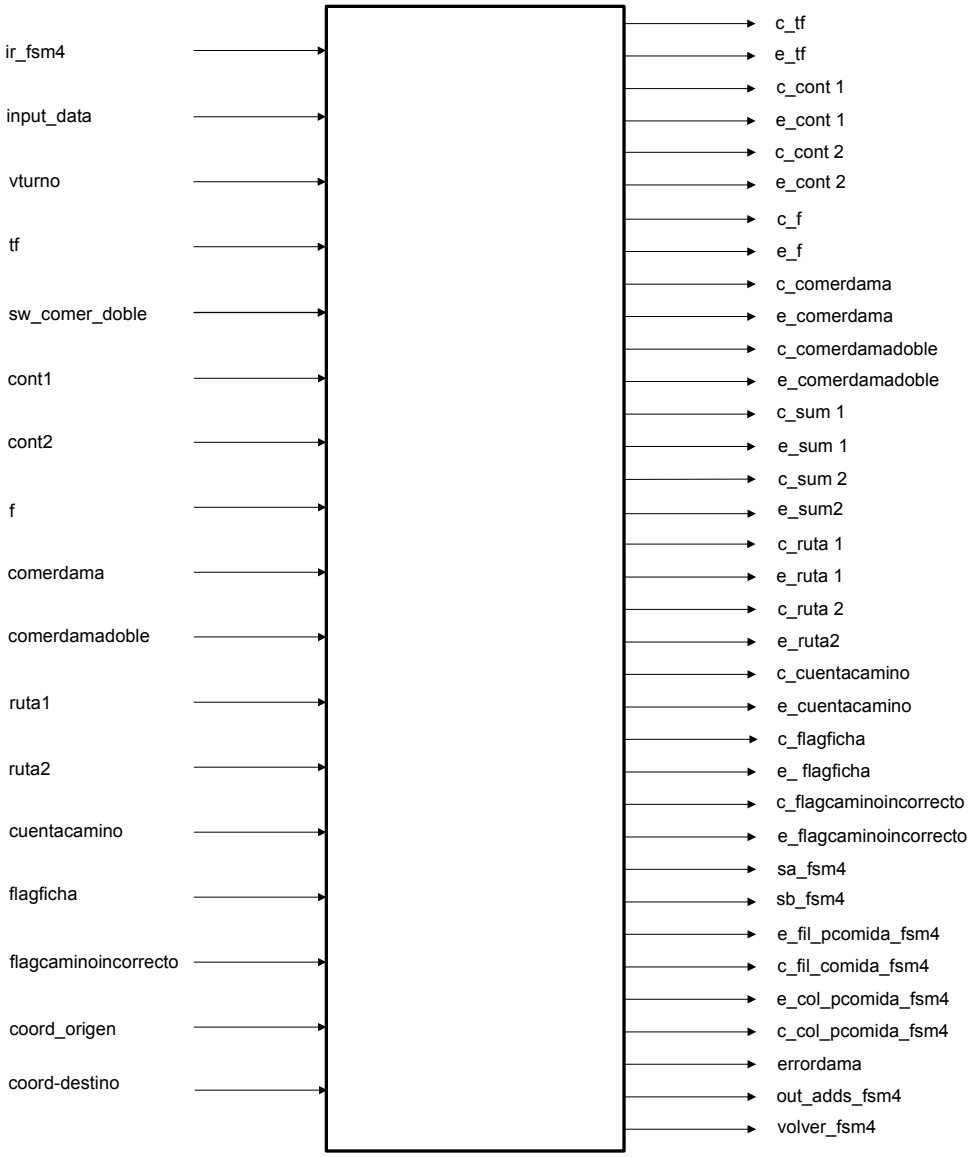


Figura 3.10 Interfaz FSM4

Entradas:

-ir_fsm4: Señal para activar a fsm4.

-input_data: Informa del contenido de la casilla (3 bits).

-vturno: Informa que jugador tiene el turno.

-sw_comer_doble: Informa de que el jugador pretende volver a realizar una captura con la dama.

-tf: Informa del tipo de movimiento (tf=1 arriba, tf=0 abajo).

-cont1: Contador para las filas en los movimientos a comprobar.

-cont2: Contador para las columnas en los movimientos a comprobar.

-comerdama: Se activa cuando la dama realiza un movimiento de captura.

-comerdamadoble: valor de fila de la ficha a capturar (3 bits).

-ruta1: Informa de la ruta tomada por la ficha (fila).

-ruta2: Informa de la ruta tomada por la ficha (columna).

-cuentacamino: La usamos para indicar que ruta vamos a tomar.

-flagficha: Se activa cuando hay fichas de turno contrario consecutivas.

-flagcaminoincorrecto: indica que la ruta por la que vamos no es correcta.

-coord_origen: Valor de fila y columna de la posición inicial (6 bits).

-coord_destino: Valor de fila y columna de la posición final (6 bits).

Salidas:

-out_adds_fsm4: Vector de dirección de la casilla a consultar (6 bits).

-c_tf: Inicializa a 0 el biestable 'tf'.

-e_tf: Señal de habilitación del biestable 'tf' que almacena el tipo de movimiento.

-c_cont1: Guarda el valor de fila de la coordenada origen.

-e_cont1: Suma o resta uno al valor de 'cont1'.

-c_cont2: Guarda el valor de columna de la coordenada origen.

-e_cont2: Suma o resta uno al valor de 'cont2'.

-c_comerdama: Inicializa a 0 el biestable 'comerdama'.

-e_comerdama: Señal de habilitación del biestable 'comerdama' que almacena si el movimiento realizado por la dama es de captura.

-c_comerdamadoble: Inicializa a 0 el biestable 'comerdamadoble'.

-e_comerdamadoble: Señal de habilitación del biestable 'comerdamadoble' que almacena que la dama va a realizar una captura múltiple.

-errordama: Movimiento no permitido.

-c_ruta1: Inicializa a 0 el biestable 'ruta1'.

-e_ruta1: Habilita el biestable 'ruta1'.

-c_ruta2: Inicializa a 0 el biestable 'ruta2'.

-e_ruta2: Habilita el biestable 'ruta2'.

-c_sum1: Inicializa a 0 el biestable 'sum1'.

-e_sum1: Habilita el biestable 'sum1'.

-c_sum2: Inicializa a 0 el biestable 'sum2'.

-e_sum2: Habilita el biestable 'sum2'.

-c_cuentacamino: Inicializa a 0 el biestable 'cuentacamino'.

-e_cuentacamino: Habilita el biestable 'cuentacamino'.

-c_flagficha: Inicializa a 0 el biestable 'flagficha'.

-e_flagficha: Pone a uno 'flagficha'.

-c_flagcaminoincorrecto: Pone a cero ‘flagcaminoincorrecto’.

-e_flagcaminoincorrecto: Pone a uno ‘flagcaminoincorrecto’.

-sa_fsm4: valor de fila de la ficha a comer (3 bits).

-sb_fsm4: valor de columna de la ficha a comer (3 bits).

-c_fil_pcomida_fsm4: Borra el valor de fila de la ficha capturada por la dama.

-e_fil_pcomida_fsm4: guarda el valor de fila de la ficha capturada por la dama.

-c_col_pcomida_fsm4: Borra el valor de columna de la ficha capturada por la dama.

-e_col_pcomida_fsm4: guarda el valor de columna de la ficha capturada por la dama.

-volver_fsm4: Indica el fin de las comprobaciones.

3.2.3.4.2 Descripción detallada

Para la realización de esta máquina se han necesitado 17 estados.

-Estado rep4: Se encarga de averiguar el tipo de movimiento (ascendente o descendente) que va a realizar la dama, teniendo en cuenta las coordenadas de origen y las de destino. En este estado permaneceremos, hasta que ‘fsm1’ active el flag para poder empezar con las comprobaciones de la coordenada destino.

-Estado n1: Una vez conocido el sentido del movimiento (ascendente o descendente), comprobamos la dirección que va a tomar (derecha o izquierda). Estos valores se ven reflejados en la Tabla 6.

Ruta 1	Ruta 2	Movimiento	Dirección
0	0	Ascendente	Izquierda
0	1	Ascendente	Derecha
1	0	Descendente	Derecha
1	1	Descendente	Izquierda

Tabla 6 Rutas

-Estado n2: Se utiliza para simular el avance de una casilla en la dirección y sentido que acabamos de calcular.

-Estado n3: Chequea el valor de la casilla a la que hemos avanzado de forma ficticia. Para ello, mandamos el valor de las coordenadas de la nueva casilla a la 'RAM' para leer la información de dicha casilla. Dependiendo de este valor se realizará una transición a uno de estos estados: 'n4', 'n5' o 'n15'.

-Estado n4: Se guardan los valores de las coordenadas fila y columna de la posible ficha a capturar, y activamos un flag para indicar que acabamos de pasar por una casilla que tenía una ficha del oponente. Esto lo hacemos para evitar pasar por encima de dos fichas contrarias colocadas consecutivamente.

-Estado n5: Comprueba si la dama puede realizar un movimiento de captura.

-Estado n6: Se comprueba si el movimiento ha finalizado, es decir, si se ha llegado a la casilla destino. Una vez realizada esta comprobación, se verifica si el jugador pretende volver a comer con dicha ficha a través del interruptor "comer_doble", siempre y cuando se den las condiciones necesarias.

-Estado n7: Calcula las posibles rutas que pueda tomar la dama para volver a comer, teniendo en cuenta que el nuevo desplazamiento nunca se puede dar en la misma dirección del movimiento anterior, ver Tabla 7.

	sum1	sum2	Señal 'cuentacamino'	Desplazamiento
ruta1=ruta2	1	0	0	Descendente derecha
	0	1	1	Ascendente derecha
ruta1≠ruta2	1	1	0	Descendente Izquierda
	0	0	1	Ascendente Izquierda

Tabla 7 Desplazamientos posibles para una dama

-Estado n8: Chequea si la ruta elegida en el estado anterior se va a poder tomar o no.

-Estado n9: Simula el avance de una casilla siguiendo la ruta establecida en 'n7', para ello actualizamos los contadores.

- Estado N10: Se comprueba el valor de la casilla a la que se ha avanzado en el estado anterior. Dependiendo de dicho valor se ira a un estado u a otro.

-Estado n11: Informa al sistema de que la ruta establecida no es correcta y que por lo tanto tenemos que tomar otra ruta alternativa si es que la hubiera.

-Estado n12: Se activa un flag para indicar que el camino tomado es incorrecto, al intentar pasar por encima de dos piezas del contrincante que estaban situadas consecutivamente.

-Estado n13: Comprueba si la dama seleccionada ha podido volver a comer en la ruta establecida.

-Estados n14, n15 y n16: En esta máquina de estados, hay tres estados de fin;

-n15: se usa para indicar que las coordenadas de destino no son validas y que hay que volver a introducirlas.

-n14: informa de que el movimiento pedido es correcto y que no hay posibilidad de seguir jugando con esa pieza.

-n16: al igual que en el anterior estado, comunica que el movimiento ha terminado satisfactoriamente, pero con la salvedad, de que en este caso la dama va a poder realizar otro movimiento de captura, y para realizar este otro movimiento, se volverán a introducir nuevas coordenadas de destino, ya que las de origen quedarán preestablecidas.

3.2.3.5 FSM5

FSM5 es una máquina de estados secundaria, que tiene como misión, verificar la coordenada origen introducida por el jugador, cuando la ficha es una dama. Para que dicha coordenada sea correcta, tiene que cumplirse que en la casilla seleccionada, haya una dama que sea del turno correspondiente (estado g1). Una vez verificado esto, se comprueba si hay algún movimiento factible en alguna dirección (estados g2, g3, g4, g5, g6, g7, g8, g9).

3.2.3.5.1 Interfaz

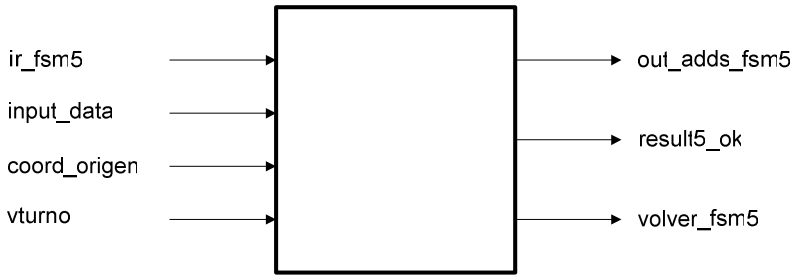


Figura 3.11 Interfaz FSM5

Entradas:

-ir_fsm5: Señal para activar a fsm5.

-coord_origen: Valor de fila y columna de la posición inicial (6 bits).

-input_data: Informa del contenido de la casilla (3 bits).

-vturno: Informa que jugador tiene el turno.

Salidas:

-out_adds_fsm5: Vector de dirección de la casilla a consultar (6 bits).

-result5_ok: Resultado de las comprobaciones.

-volver_fsm5: Indica el fin de las comprobaciones.

3.2.3.5.2 Descripción detallada

Máquina secundaria compuesta por 12 estados que a continuación pasamos a describir.

-Estado rep5: Espera a que 'fsm1' le llame para iniciar las comprobaciones.

-Estado g1: Comprueba que la dama seleccionada es del turno actual. Dicha verificación, se realiza mandando al bloque de la 'RAM' la dirección de la casilla que ocupa dicha ficha en el tablero, y ésta nos devolverá la información de dicha casilla.

-Estados g2, g4, g6 y g8: Se usan para realizar la comprobación de un movimiento simple, es decir, el desplazamiento a una casilla contigua en diagonal. Para la realización de esa verificación seguimos los siguientes pasos:

- Calculamos la dirección de la casilla a la que queremos desplazarnos.

- Mandamos al bloque RAM la dirección anteriormente obtenida.

- La RAM nos devuelve el valor de la casilla. Si está vacía, significará que la dama seleccionada tiene al menos un movimiento factible, y volveremos a 'fsm1' para

introducir las coordenadas de destino. Por el contrario, si la casilla está ocupada por una ficha del contrincante, nos iremos a los estados que se encargan de verificar si se puede realizar la captura, y si la casilla está ocupada por una ficha del mismo turno, entonces nos vamos al siguiente estado de movimiento simple.

-Estados g3, g5, g7 y g9: En ellos comprobamos si hay posibilidad de realizar una captura simple. Para lo cual realizamos los siguientes pasos:

- Calculamos la dirección de la casilla a la que queremos desplazarnos.
- Mandamos al bloque RAM la dirección anteriormente obtenida.
- La RAM nos devuelve el valor de la casilla. Si está vacía, el movimiento es factible. Si no es así, pasamos al siguiente estado para buscar un movimiento simple en otra dirección. Si se llega al estado 'g9' y no hay movimiento factible entonces pasamos al estado 'fin_error'.

- Estados fin_error y fin_ok: Son los encargados de informar a la máquina principal de los resultados obtenidos.

3.3 Tablero

Es el que se encarga de indicar al modulo 'VGA' el color del píxel que se va a representar por pantalla.

3.3.1 Interfaz

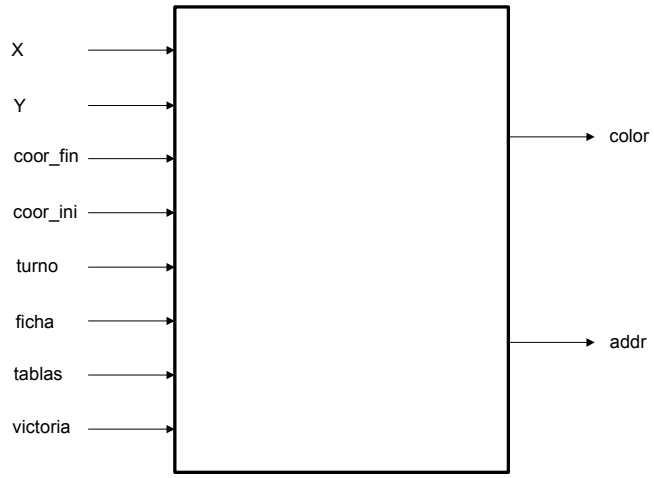


Figura 3.12 Interfaz Tablero

Entradas:

-x: Indica el píxel del eje x (10 bits).

-y: Indica el píxel del eje y (10 bits).

-coord_ini: Valor coordenada origen (6 bits).

-coord_fin: Valor coordenada destino (6 bits).

-turno: Indica a que jugador le toca.

-ficha: Valor de la casilla a representar (3 bits).

-tablas: Indica que la partida ha terminado en tablas.

-victoria: Indica que la partida ha terminado con la victoria de uno de los dos jugadores.

Salidas:

-color: Informa del color a representar (3 bits).

-addr: Dirección de una casilla del tablero (6 bits).

3.3.2 Descripción detallada

El módulo tablero es el encargado de controlar la interfaz grafica del sistema, por lo tanto, es el que se ocupa de representar en la pantalla los diferentes movimientos que se realizan a lo largo de una partida en el tablero, así como la introducción de las coordenadas, el turno del jugador al que le toca mover o la representación de fin de partida.

Para realizar todo lo anteriormente citado se ha dividido la pantalla en cuatro zonas y se han utilizado siete procedimientos para manejarlas.

-Zona 1 (Tablero): Situado ligeramente a la izquierda del centro de la pantalla. Para su representación utilizamos tres procedimientos, uno de ellos, 'PintaFicha', que se ocupa de indicar el color del píxel a representar según el valor que tenga la señal 'ficha'. Ese valor le obtenemos con el segundo procedimiento, 'Fil_Col', que tiene como misión traducir la coordenada del píxel en un valor de fila y columna, que se mandara a la 'RAM', para que esta nos devuelva el valor de dicha casilla (ver apartado 3.5), y ese valor será el de ficha. El último procedimiento utilizado es 'PintaNúmero' y con él representamos los números que aparecen alrededor del tablero que facilitan al jugador la entrada de coordenadas.

-Zona 2 (Turno): En este apartado también se han utilizado dos procedimientos, uno para representar la letra “J” (de jugador) ‘PintaLetra’, y otro para representar un rectángulo del color del turno correspondiente, ‘PintaTurno’.

Ambas representaciones están situadas debajo del tablero y a su izquierda.

-Zona 3 (Coordenadas): Están situadas a la derecha del tablero y con ellas se van a representar las posiciones de origen y destino de la ficha.

En cada inicio de jugada se resetean los valores de las coordenadas y el valor por defecto que aparece por pantalla será el de cero.

Para la representación de las coordenadas se han utilizado los procedimientos ‘PintaNúmero’ y ‘PintaLetra’.

-Zona 4 (Victoria y Tablas): En esta última zona se representa el final de partida, para ello se han diseñado unos símbolos a través de los procedimientos ‘PintaVictoria’ y ‘PintaTablas’. Dichos símbolos se representan debajo del tablero y a su derecha.

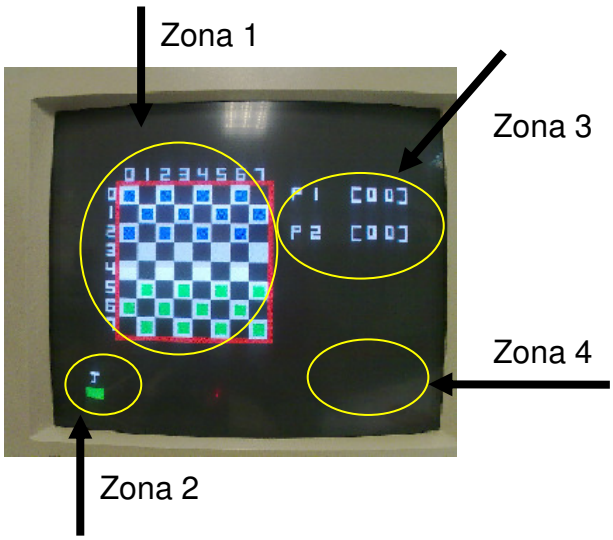


Figura 3.13 Imagen real

3.4 VGA

El objetivo de este módulo, es controlar una pantalla VGA. Dicha pantalla, se controla a través de tres señales analógicas, una especifica la intensidad de los colores (Rojo, Verde, Azul, (RGB)) y las otras dos el sincronismo horizontal y vertical. Ver apartado 2.3.3.

3.4.1 Interfaz

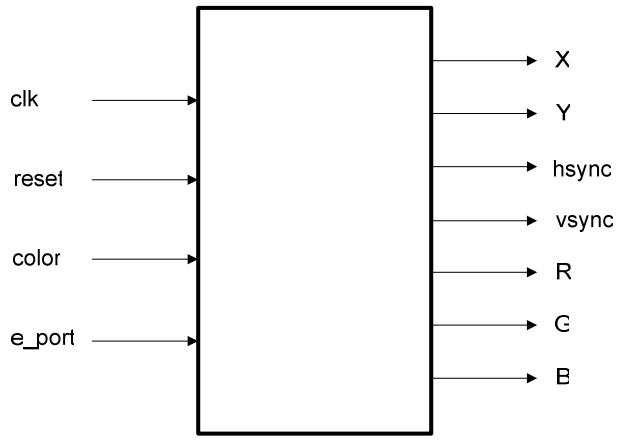


Figura 3.14 Interfaz VGA

Entradas:

-clk: Señal de reloj.

-reset: Reinicio del sistema.

-color: es el código del color que se da al píxel que se está procesando.

-e_port: Habilita al bloque.

Salidas:

-x: valor del píxel en el eje x (10 bits).

-y: Valor del píxel en el eje y (10 bits).

-hsync: Salida de sincronismo horizontal.

-vsync: Salida de sincronismo vertical.

-R: Salida para el color rojo.

-G: Salida para el color verde.

-B: Salida para el color azul.

3.4.2 Diagrama de bloques

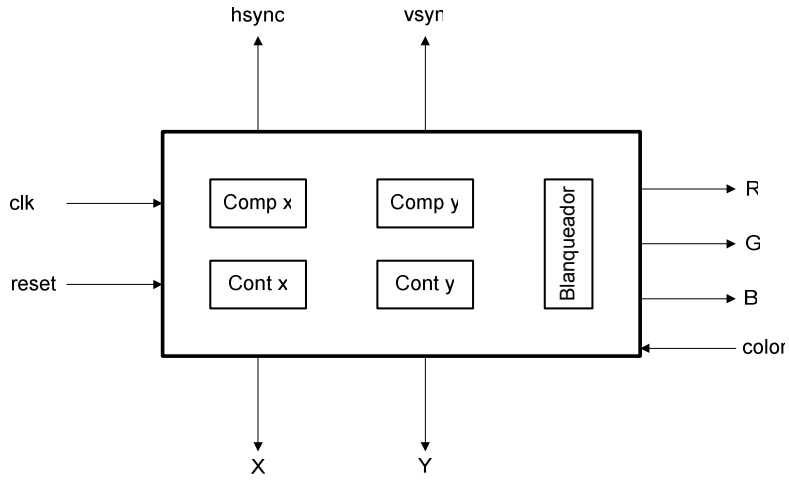


Figura 3.15 Diagrama de bloques VGA

3.4.3 Descripción detallada

El circuito consta de dos contadores, dos comparadores y un bloque blanqueador necesario para el sincronismo ver apartado 2.3.3.

Los contadores (contx, conty) generan las coordenadas X e Y, y se implementa de la siguiente manera:

- *Contx:* está formado por dos componentes (cont y div_frecuencia), dichos bloques forman el 'contador2', ver Figura 3.16. El componente 'div_frecuencia' se utiliza para dividir la frecuencia de cuenta a la mitad según lo explicado en el apartado 2.3.3. El componente 'cont', es un contador de pulsos, y en este caso nos indica el valor de la coordenada X.

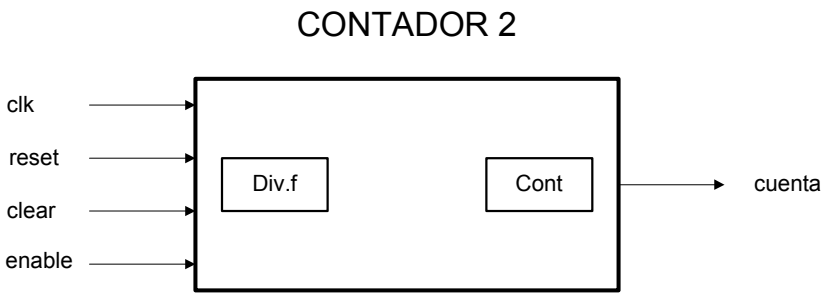


Figura 3.16 Contador x

Las entradas y salidas del contador 2 se describen a continuación.

Entradas:

-clk: Señal de reloj.

-reset: Reinicio del sistema.

-clear: Pone a cero la salida.

-enable: activa al contador.

Salidas:

-cuenta: Valor del contador (10 bits).

- Conty: está compuesto únicamente por el componente 'cont', el cual, tiene como misión contar el número de líneas verticales de la pantalla (valor de Y).

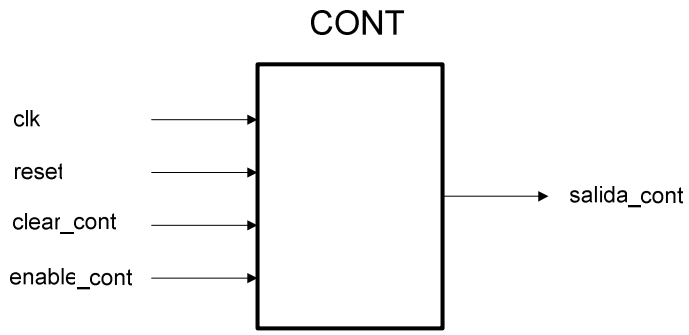


Figura 3.17 Contador y

A continuación describimos las entradas y salidas del contador y.

Entradas:

-clk: Señal de reloj.

-reset: Reinicio del sistema.

-clear_cont: Pone a cero el valor de 'salida_cont'

-enable_cont: Suma uno al valor de 'salida_cont'

Salidas:

-salida_cont: Valor del contador (10 bits).

Los comparadores son idénticos y su funcionamiento interno es el siguiente: comprueban una coordenada que les es enviada del contador correspondiente y

generan tres señales: la señal de sincronismo (HSYNC o VSYNC), otra que indica el final del ciclo (que se usara para reiniciar el contador correspondiente o para habilitar otro contador), y por último la de blanqueo, que indicará al bloque blanqueador que debe eliminar los colores de las salidas R, G y B.

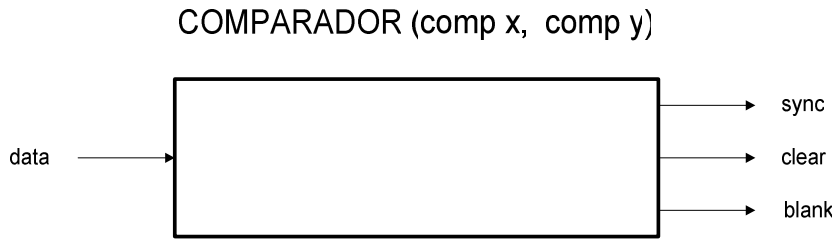


Figura 3.18 Comparador

Las entradas y salidas de los comparadores son las siguientes.

Entradas:

-data: Recoge los valores de 'salida_cont' (10 bits).

Salidas:

-sync: Valor de sincronismo.

-clear: Indica final de ciclo.

-blank: Activa al bloque blanqueador.

3.5 Memoria de juego: Bloque RAM

Es el módulo de memoria del sistema. En él se guardan las posiciones que ocupan las fichas en cada momento del juego.

3.5.1 Interfaz

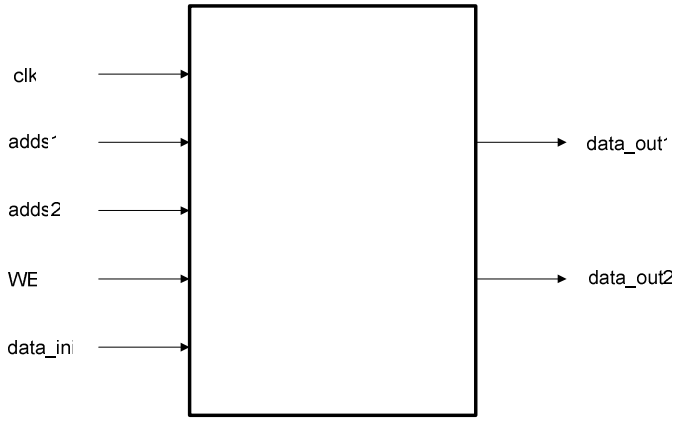


Figura 3.19 Interfaz del bloque RAM

Las entradas y salidas de la RAM se describen a continuación.

Entradas:

-clk: Señal de reloj.

-data_in: Valor casilla (3 bits).

-adds1: Coordenadas desde el bloque ‘tablero’ (3 bits).

-adds2: Coordenadas desde el bloque ‘árbitro’ (3 bits).

-WE: Permite la escritura.

Salidas:

-data_out1: Informa del valor de ficha al ‘tablero’ (3 bits).

-data_out2: Informa del valor de ficha al ‘árbitro’ (3 bits).

3.5.2 Descripción detallada

El módulo ‘RAM’ está unido a otros dos módulos, ‘tablero’ y ‘árbitro’. Con el primero, la comunicación es constante y únicamente de lectura, es decir, el tablero le manda la dirección de la casilla a representar y la ‘RAM’ le devuelve el valor de dicha casilla. Gracias a esta comunicación constante y asíncrona se puede visualizar en tiempo real las jugadas.

La comunicación con el ‘árbitro’ es distinta ya que no es constante y puede ser tanto de lectura como de escritura. No es constante, porque solamente se comunican en ciertos momentos de cada jugada, como por ejemplo, cuando comprobamos las coordenadas introducidas, o cuando se realiza una captura. Cuando se realiza una o varias capturas, el bloque ‘árbitro’ habilita el puerto de entrada ‘WE’ del bloque ‘RAM’

para que este guarde en memoria la nueva posición que va a ocupar la ficha en el tablero y se eliminen de la memoria la o las fichas capturadas.

La introducción de datos sólo se da en tres casos, cuando se inicia la partida (colocación de las fichas en su posición inicial), cuando hay capturas y cuando un peón se convierte en dama.

Cada casilla puede tener cinco valores, que son:

-Casilla vacía: (000)

-Ficha verde: (011)

-Ficha azul: (001)

-Dama verde: (111)

-Dama azul: (101)

Este módulo está diseñado de la siguiente manera:

Un array que va de cero a sesenta y tres. Cada elemento del vector hace referencia a una casilla del tablero, como podemos ver en la Tabla 8.

0	1	2	3	4	5	6	7
8	9	10	11	12	13	14	15
16	17	18	19	20	21	22	23
24	25	26	27	28	29	30	31
32	33	34	35	36	37	38	39
40	41	42	43	44	45	46	47
48	49	50	51	52	53	54	55
56	57	58	59	60	61	62	63

Tabla 8 Número de casilla

Cada elemento de dicho vector, guarda el valor de la casilla a la que está direccionando, esto lo podemos ver en la Tabla 9, que sería un ejemplo de un inicio de partida.

001	000	001	000	001	000	001	000
000	001	000	001	000	001	000	001
001	000	001	000	001	000	001	000
000	000	000	000	000	000	000	000
000	000	000	000	000	000	000	000
000	011	000	011	000	011	000	011
011	000	011	000	011	000	011	000
000	011	000	011	000	011	000	011

Tabla 9 Valor de casilla

3.6 Teclado

Este último módulo, lo usamos para poder comunicarnos con un teclado físico, por lo tanto, su misión es decodificar las teclas pulsadas y mandar dicha información al árbitro.

3.6.1 Interfaz

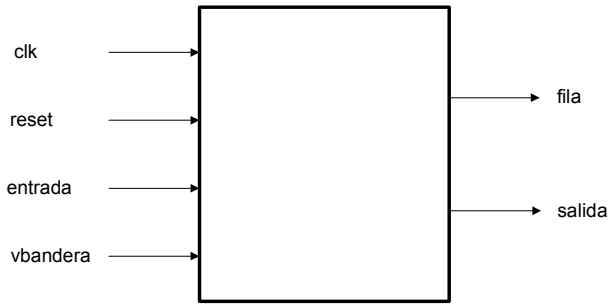


Figura 3.20 Tecla

Entradas:

-clk: Señal de reloj.

-reset: Reinicio del sistema.

-entrada: Valor de la columna pulsada (4 bits).

-vbandera: Permite la recogida de teclas.

Salidas:

-fila: Valor de la fila a comprobar (4 bits).

-salida: Valor de la tecla pulsada (4 bits).

3.6.2 Descripción detallada

Como ya se adelantó en el capitulo 2, este módulo tiene como misión:

• Identificar la tecla pulsada

• Evitar los 'rebotes'

Para llevar acabo estas dos premisas, se ha diseñado una pequeña máquina de estados, que se encarga de muestrear el valor de la tecla pulsada que es entrada del árbitro.

En los estados 'f1', 'f2', 'f3 y 'f4', se utiliza un contador para evitar posibles rebotes. Este contador actúa como un temporizador y su misión es, realizar una espera de 5 ms antes de empezar a comprobar qué tecla ha sido pulsada.

A la hora de identificar la tecla pulsada hay que tener en cuenta que se ha utilizado una lógica activa a nivel bajo, por ello a la hora de averiguar la tecla pulsada se irán recorriendo las diferentes filas en busca de un cero en alguna de las columnas del teclado. Ver Tabla 10.

Tecla pulsada	Fila + Columna	Salida
0	1110 1011	0000
1	0111 0111	0001
2	0111 1011	0010
3	0111 1101	0011
4	1011 0111	0100
5	1011 1011	0101
6	1011 1101	0110
7	1101 0111	0111
A	1110 0111	1000

Tabla 10 Teclado

Para que se pueda mandar el valor de la tecla pulsada al 'árbitro', se ha tenido que muestrear al menos dos veces dicho valor de forma consecutiva. Esta comprobación se realiza en el estado 'hay tecla'.

4 Evaluación del circuito

4.1 Pruebas en el hardware

Se han realizado diferentes comprobaciones del sistema que pasaremos a detallar a continuación.

En primer lugar se diseñó un circuito que controlase una pantalla VGA. Para poder comprobar que el diseño de la VGA funcionaba, se realizaron varias simulaciones, algunas de ellas se pueden ver en el siguiente punto, en el apartado de VGA.

Una vez realizado el modulo de VGA, se pasó al diseño del tablero para verificar que efectivamente se podía dibujar una imagen en la pantalla.

Después de esto faltaba la comunicación entre el jugador y el sistema realizado, para lo cual se optó por un teclado pasivo matricial. A la hora de implementarlo se tuvo que tener muy en cuenta el problema de los rebotes, ya que, este percance retraso el avance del proyecto. Para evitarlo se introdujeron un par de contadores, uno para retrasar las lecturas del teclado y otro para comparar valores, si ambas cosas se daban correctamente, el valor de salida pertenecía al de la tecla pulsada.

A continuación se pasó a realizar diferentes pruebas al sistema ya completo, y son las siguientes:

-Movimientos simples de peones.

-Limites del tablero (movimientos erróneos).

-Capturas simples y múltiples con peones.

-Hacer dama.

-Movimiento de la dama hacia delante y hacia detrás de varias casillas en la misma diagonal.

-Capturas simples y múltiples con una dama.

-Comprobación de movimientos erróneos, como puedan ser el desplazamiento de una ficha a una casilla que no esta en la misma diagonal, mover un peón hacia atrás, seleccionar un peón que no es de tu turno o seleccionar un peón que esta ahogado (no tiene movimiento).

-Finalización de una partida tanto por victoria como por tablas.

-También se hicieron pruebas para mejorar la colocación y presentación de los diferentes elementos que se iban a representar por pantalla.

4.2 Simulaciones

Durante la ejecución del proyecto se han ido realizando varias simulaciones donde se ven los pasos que se toman a la hora de efectuar un movimiento, o como usamos los distintos bloques pertenecientes al sistema.

A continuación, vamos a nombrar las distintas simulaciones que se han llevado a cabo:

- Para la VGA, se han probado la parte de sincronismo tanto vertical como horizontal, y los distintos módulos (contadores, divisor de frecuencia y comparadores) que se han utilizado para el diseño de esta.

- Para el tablero, se han realizado algunas simulaciones para comprobar las comunicaciones que hay entre los distintos bloques con él, como pueda ser la finalización de una partida o la representación de las coordenadas introducidas por el jugador.

- En el caso de la RAM sí se realizaron varias simulaciones para comprobar la introducción de datos, tanto en el inicio de partida, como durante la partida; así como la salida de información hacia el árbitro. También en estas simulaciones se comprobó el funcionamiento asíncrono de la RAM.

- Para acabar, el módulo árbitro ha sido donde más simulaciones se han realizado, puesto que es el módulo que controla el juego. Se simularon tanto

movimientos factibles como movimientos incorrectos para comprobar su robustez. Algunos de estos movimientos incorrectos serian, seleccionar una ficha que no se pueda mover o una ficha que no es de tu turno, querer avanzar con un peón dos casillas sin realizar una captura, moverse por fuera del tablero, pasar por encima de fichas propias, etc. Estas verificaciones conllevaban implícitamente las comprobaciones de comunicación entre los distintos bloques.

Sólo se han plasmado algunas de las muchas simulaciones que se han llevado a cabo y que pasamos a detallar a continuación.

Empezamos con el bloque RAM, que se encarga de guardar la información de cada casilla de nuestro tablero y la de darnos dicha información. Para este bloque se han realizado un par de simulaciones:

- En la primera, ver Figura 4.1, se ve como se introducen los datos de inicio de partida a través de las señales ‘adds2’ y ‘data_in’.

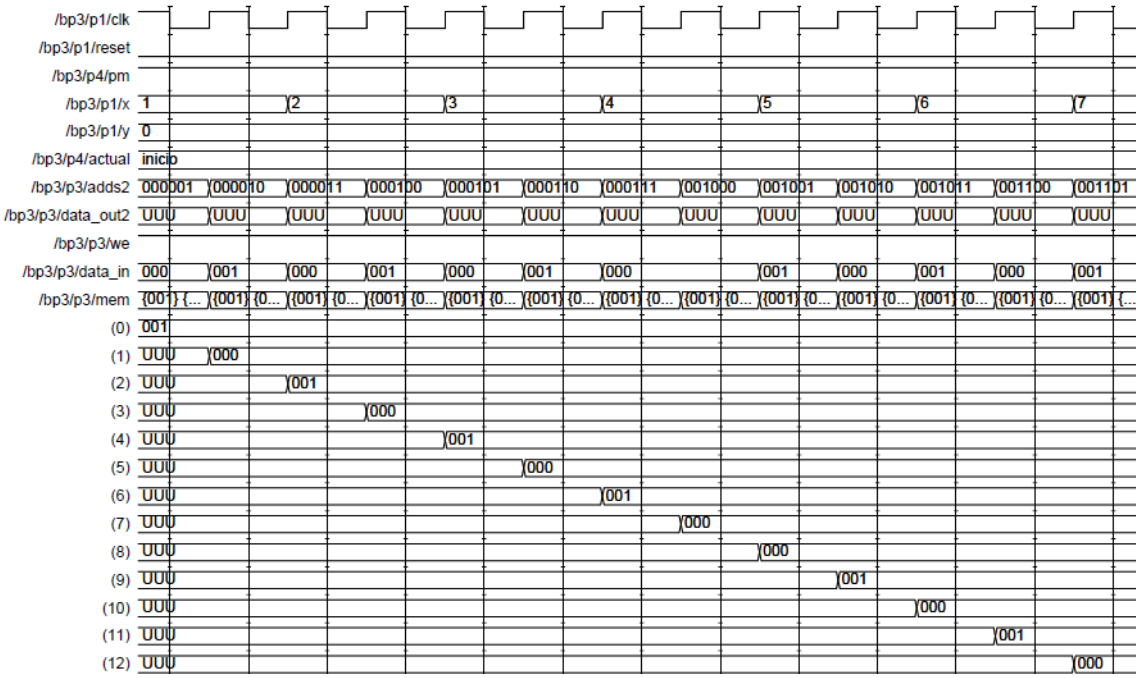


Figura 4.1 Simulación RAM 1

- La segunda simulación, ver Figura 4.2 nos permite ver dos momentos (e5 y e6) del juego donde se pide información de una determinada casilla, ya que dependiendo de ese valor se realizaran unas operaciones u otras. Para ello se han utilizado las señales ‘adds2’ (dirección de RAM) y ‘data_out2’ (datos de salida de RAM).

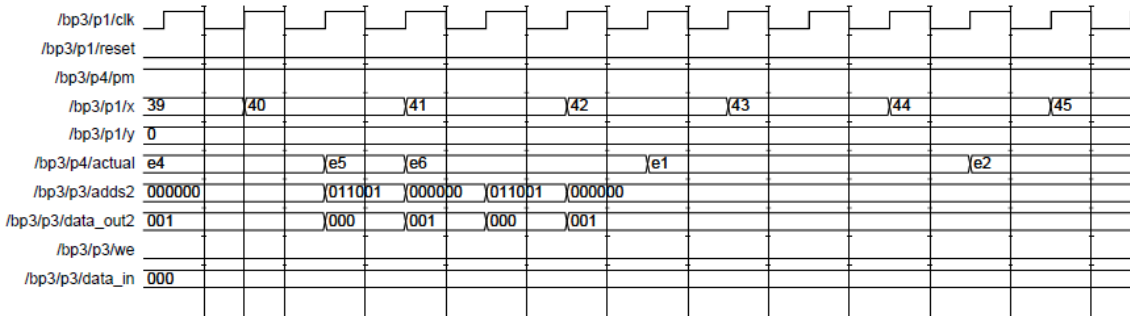


Figura 4.2 Simulación RAM 2

El siguiente par de simulaciones pertenecen al bloque VGA y en ellas vamos a ver un cambio de línea horizontal y un cambio de pantalla.

- En esta primera simulación, se puede apreciar el momento del cambio de línea horizontal a través de la señal ‘x’ y con ello el aumento del valor de línea vertical con la señal ‘y’. Ver Figura 4.3.

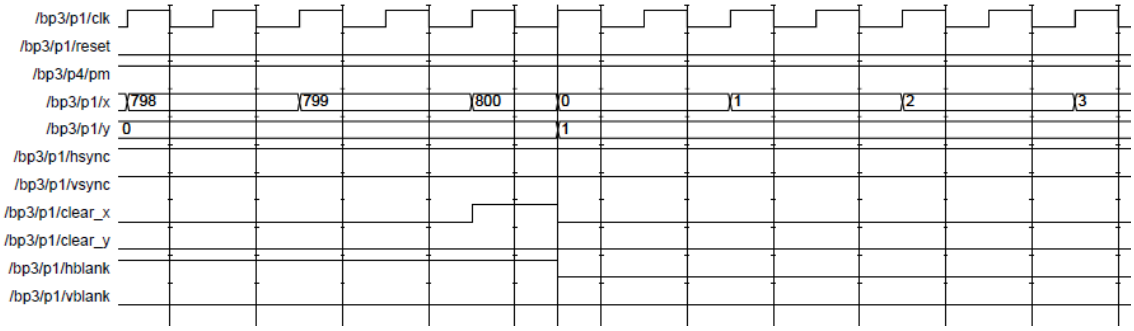


Figura 4.3 Simulación VGA 1

- En la segunda se observa el cambio de pantalla o lo que es lo mismo, la finalización de una pantalla y el comienzo de otra nueva. Esto lo podemos ver en los valores de las señales ‘x e y’ y en la señal ‘vsync’. Ver Figura 4.4.

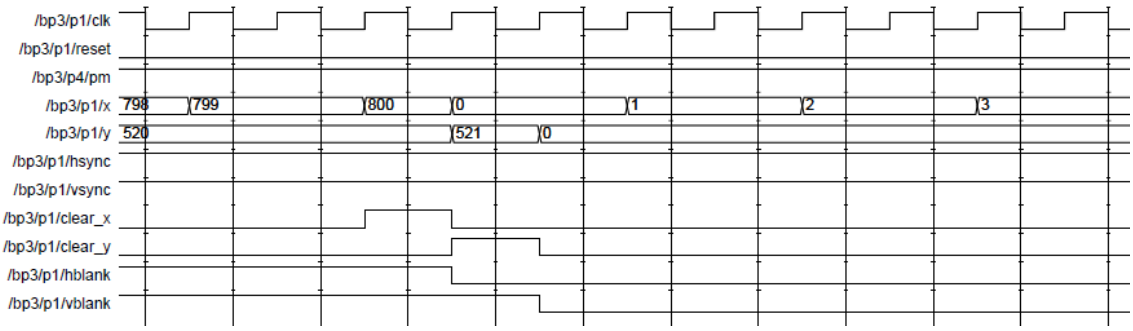


Figura 4.4 Simulación VGA 2

Una vez vistas las simulaciones de los bloques de VGA y RAM, pasaremos a ver las del árbitro, las cuales engloban los diferentes tipos de movimientos que se pueden realizar durante una partida. Para un mayor entendimiento, sólo se han representado las señales más importantes a la hora de realizar el movimiento especificado.

Antes de pasar a ver las simulaciones del árbitro, reseñar que para simular la entrada de los valores de fila y columna se ha usado la señal ‘valor1’. Los valores de esta señal van del cero al ocho, siendo los siete primeros utilizados para la introducción de filas y columnas, y el valor ocho para la tecla ‘A’ o ‘intro’. Ver el apartado 2.3.2 Teclado.

Para concluir con esta introducción, comentar que para describir cada simulación se muestra una ventana principal de simulación que ilustra el conjunto del movimiento, y varias ventanas de simulación secundarias que son ampliaciones de la principal, gracias a las cuales se puede ver mas en detalle los cambios de valor de las señales, así como los estados por los que se pasa.

- **Movimiento simple del peón**

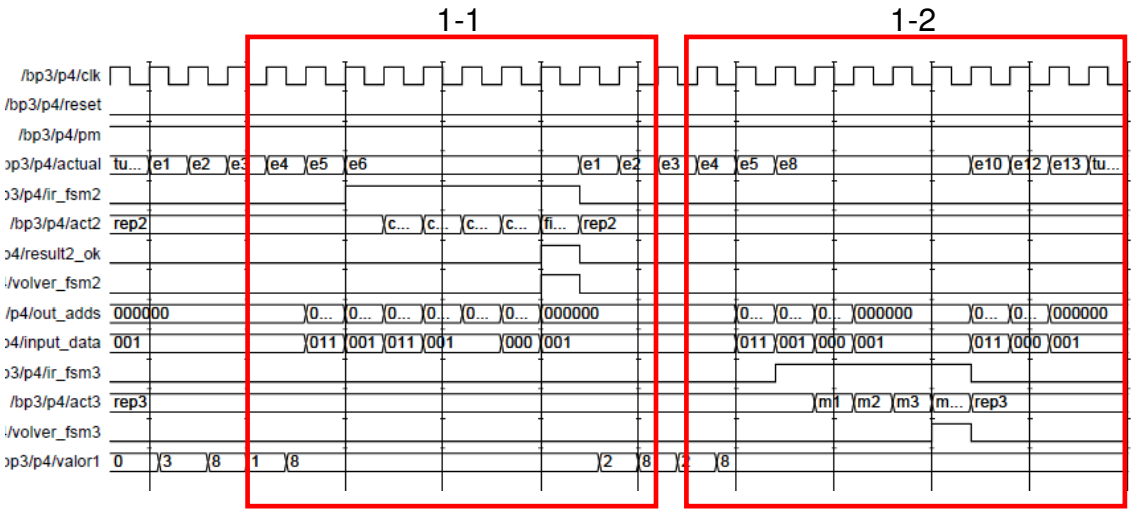


Figura 4.5 Simulación Árbitro 1

Para la realización de este movimiento usamos las máquinas secundaria ‘fsm2 y fsm3’ además de la máquina principal ‘fsm1’. En estas tres simulaciones se puede apreciar como se pide información de una determinada casilla usando la señal ‘out_adds’ (dirección de la RAM) y a continuación se recibe la información a través de la señal ‘input_data’ (datos de entrada del árbitro) proveniente del bloque RAM.

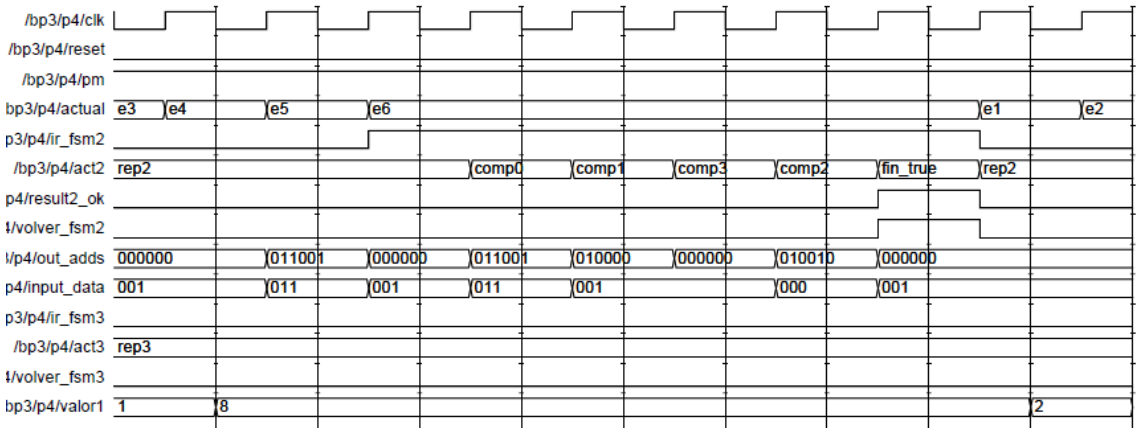


Figura 4.6 Simulación Árbitro 1-1 (fsm2)

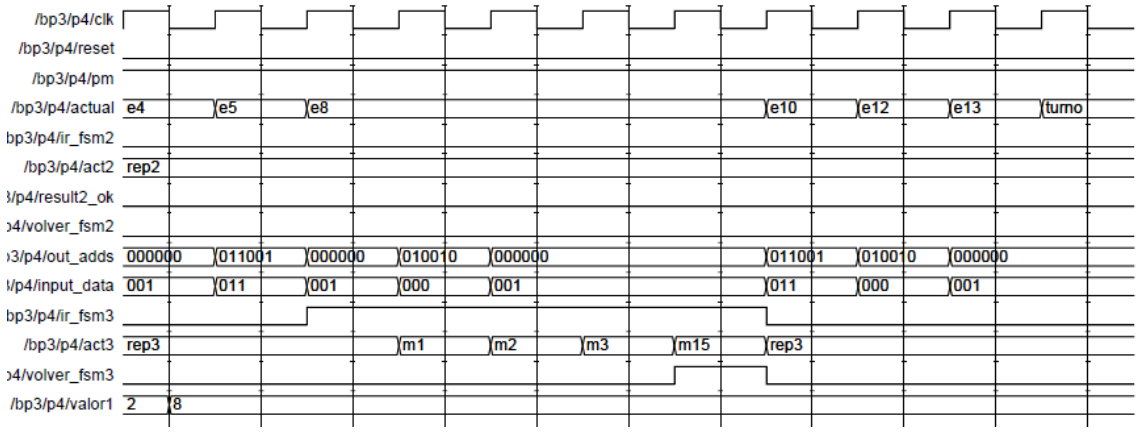


Figura 4.7 Simulación Árbitro 1-2 (fsm3)

- **Captura simple de un peón**

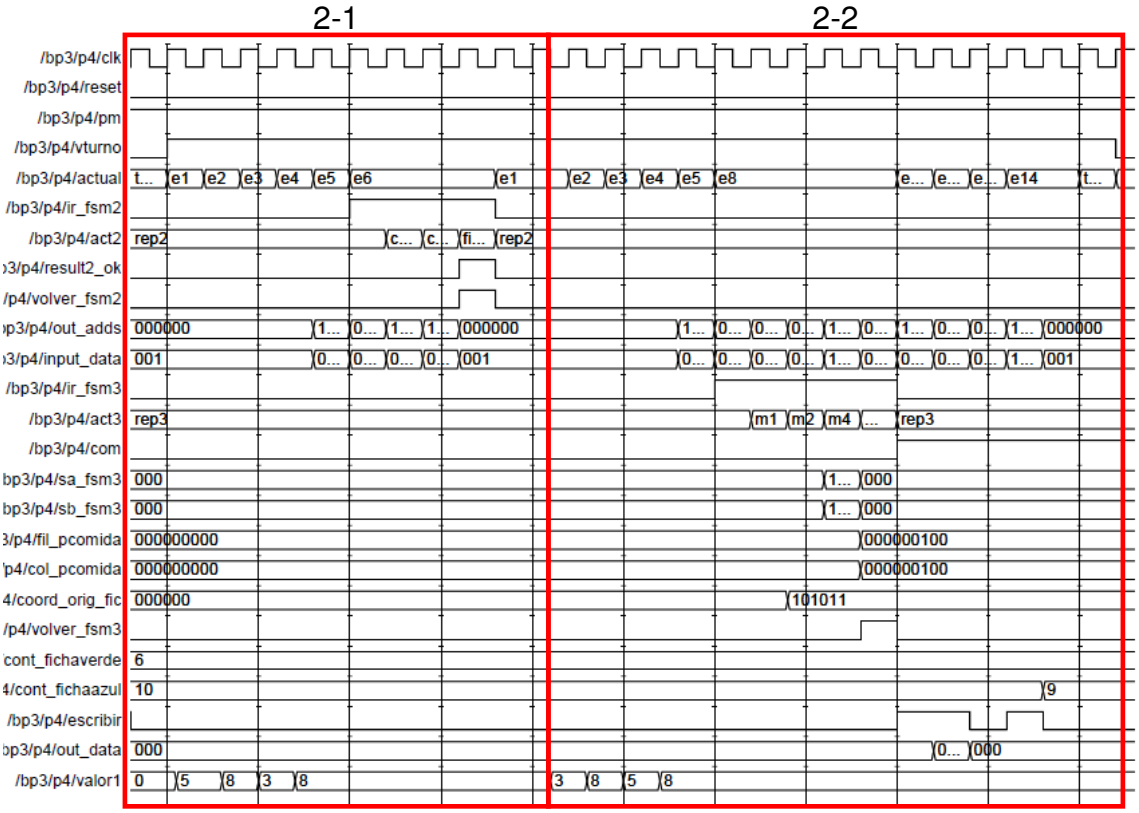


Figura 4.8 Simulación Árbitro 2

En esta ocasión volvemos a utilizar las mismas máquinas de estado que en el caso anterior, pero introduciendo algunas señales que son pertinentes para el movimiento. En la Figura 4.10 se pueden observar los cambios de valor que sufren estas nuevas señales a lo largo de 'fsm3 y fsm1'. Recordar que las señales 'fil_pcomida y col_pcomida' se usan para guardar el valor de la coordenada de la ficha capturada para su posterior eliminación. Dicha eliminación la vemos reflejada en el valor del contador de fichas azules, que desciende en uno al final del turno.

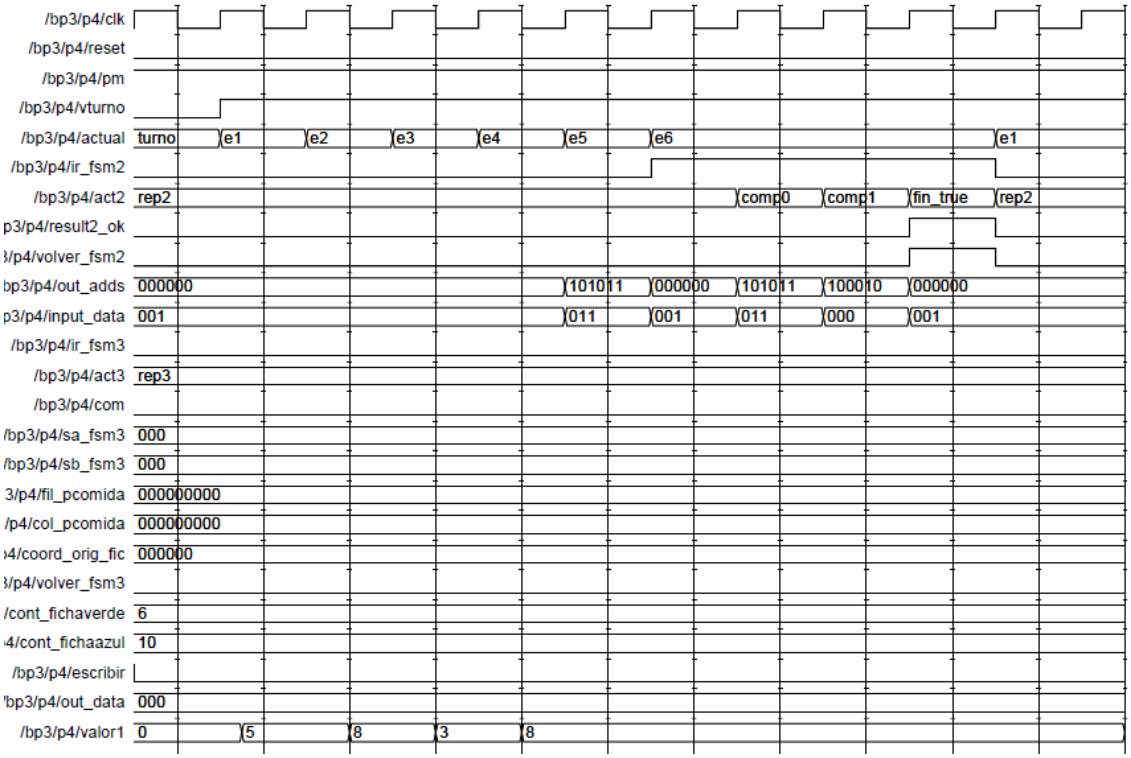


Figura 4.9 Simulación Árbitro 2-1 (fsm2)

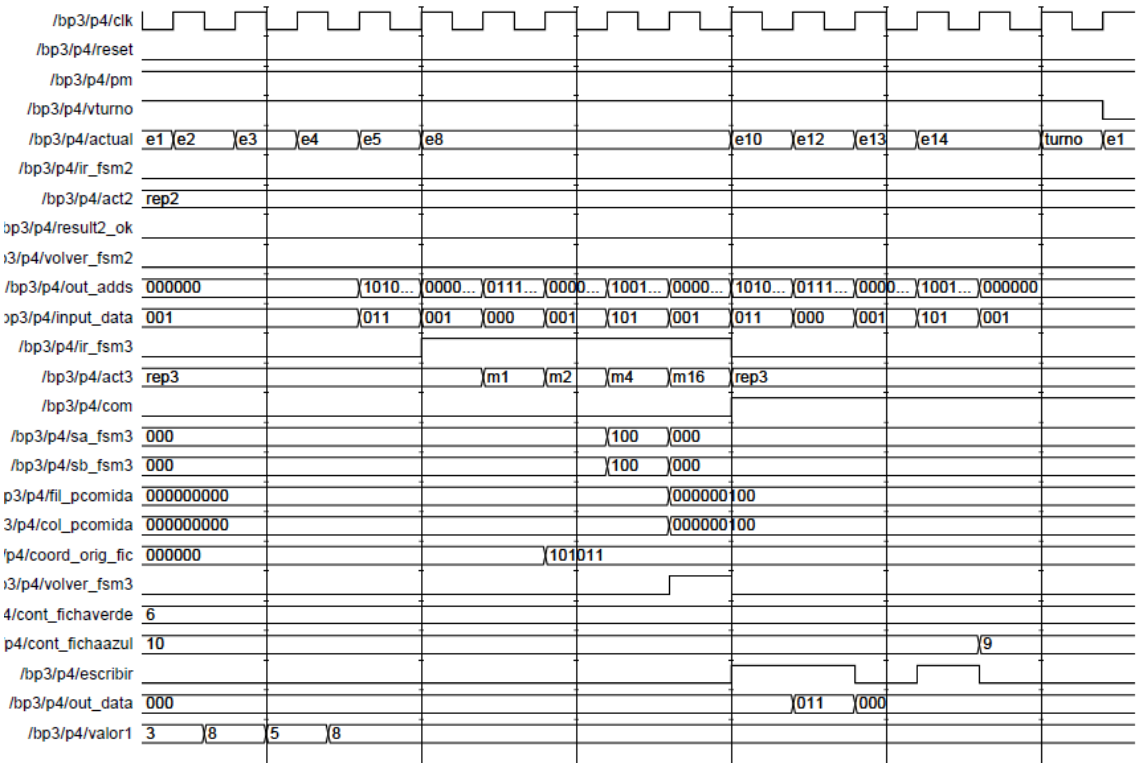


Figura 4.10 Simulación Árbitro 2-2 (fsm3)

- **Captura múltiple peón**

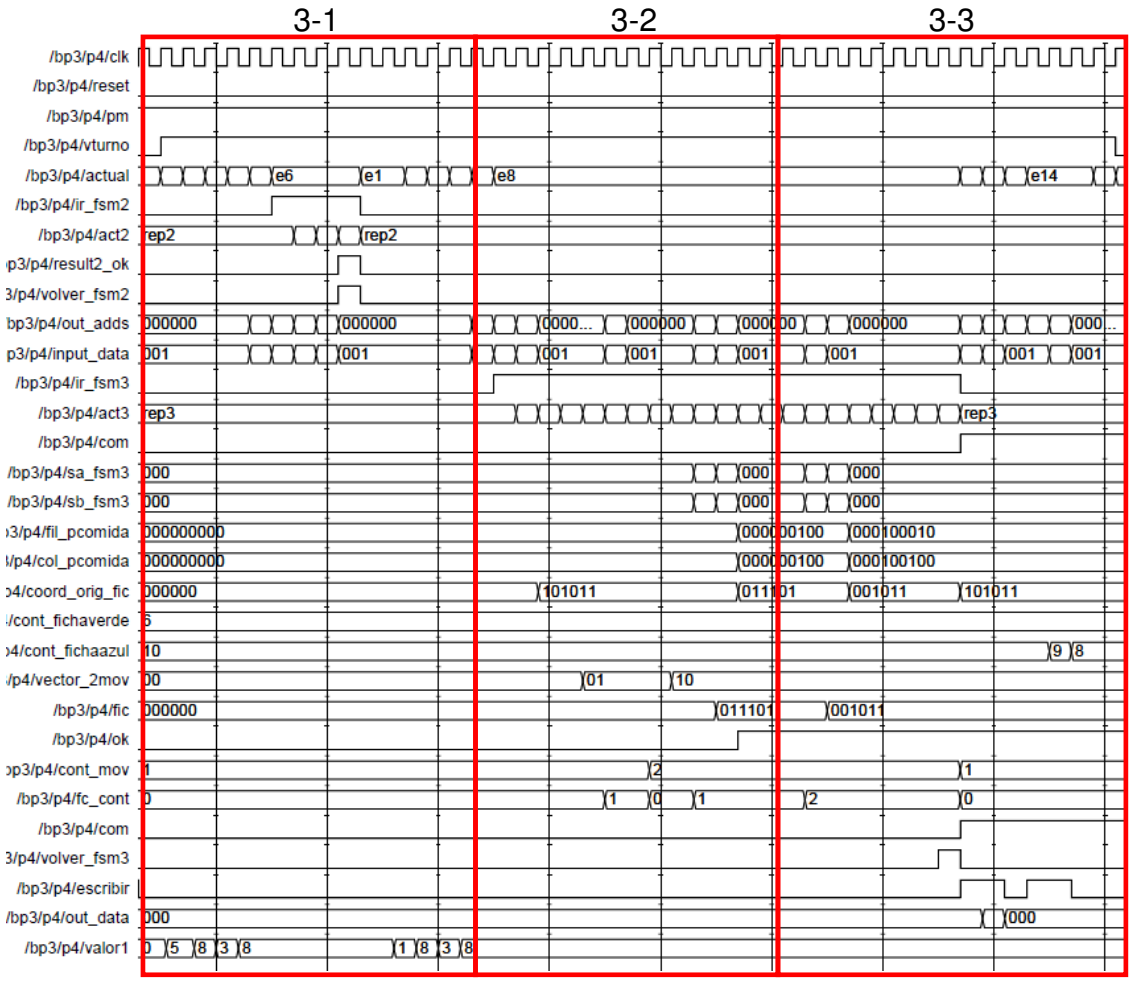


Figura 4.11 Simulación Árbitro 3

Para una mejor visualización del movimiento se ha dividido la simulación en tres. En la primera se observa como se recogen los valores de las coordenadas origen y destino y entre medias se llama a 'fsm2' para comprobar la coordenada origen. En las dos siguientes se ven las comprobaciones que se realizan en 'fsm3', donde se aprecian los distintos caminos 'vector_2mov' que tiene que comprobar la máquina de estados para dar el visto bueno al movimiento.

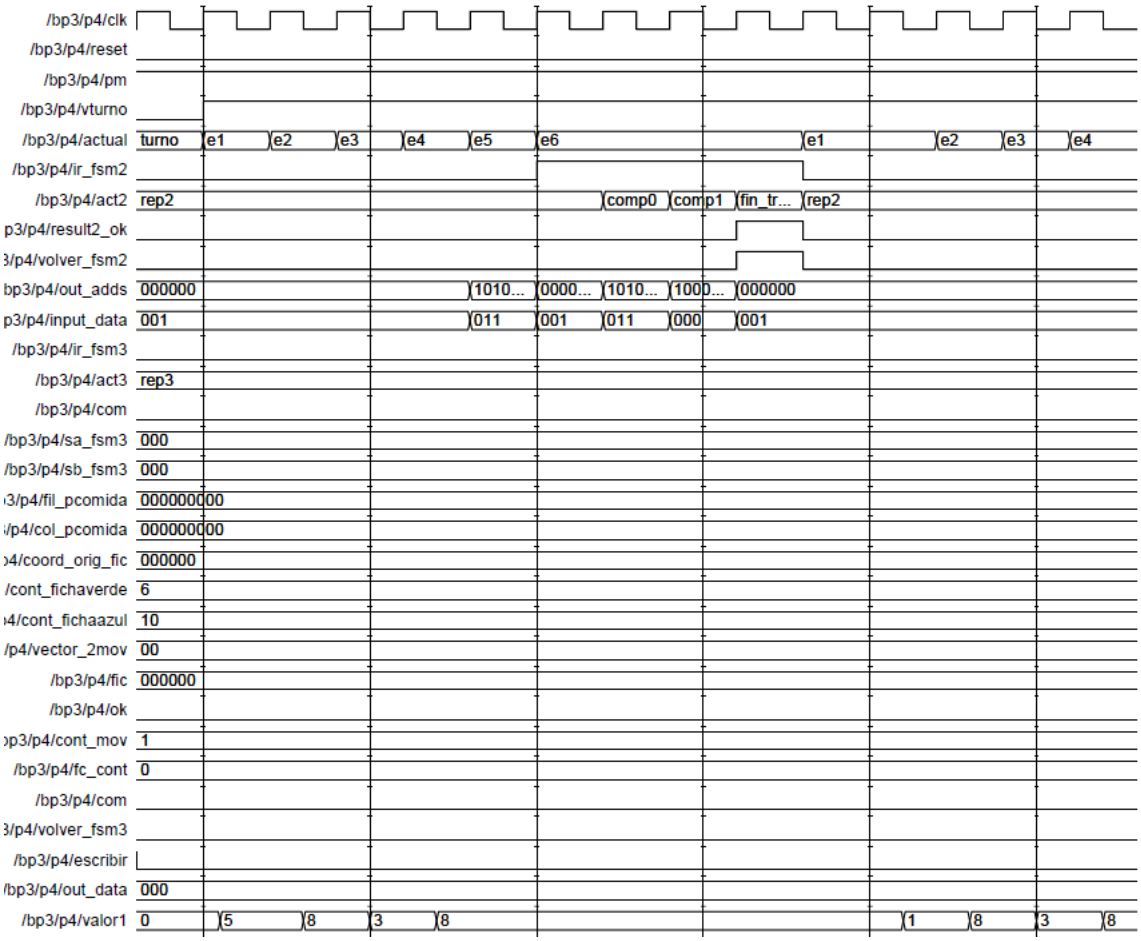


Figura 4.12 Simulación Árbitro 3-1 (fsm2)

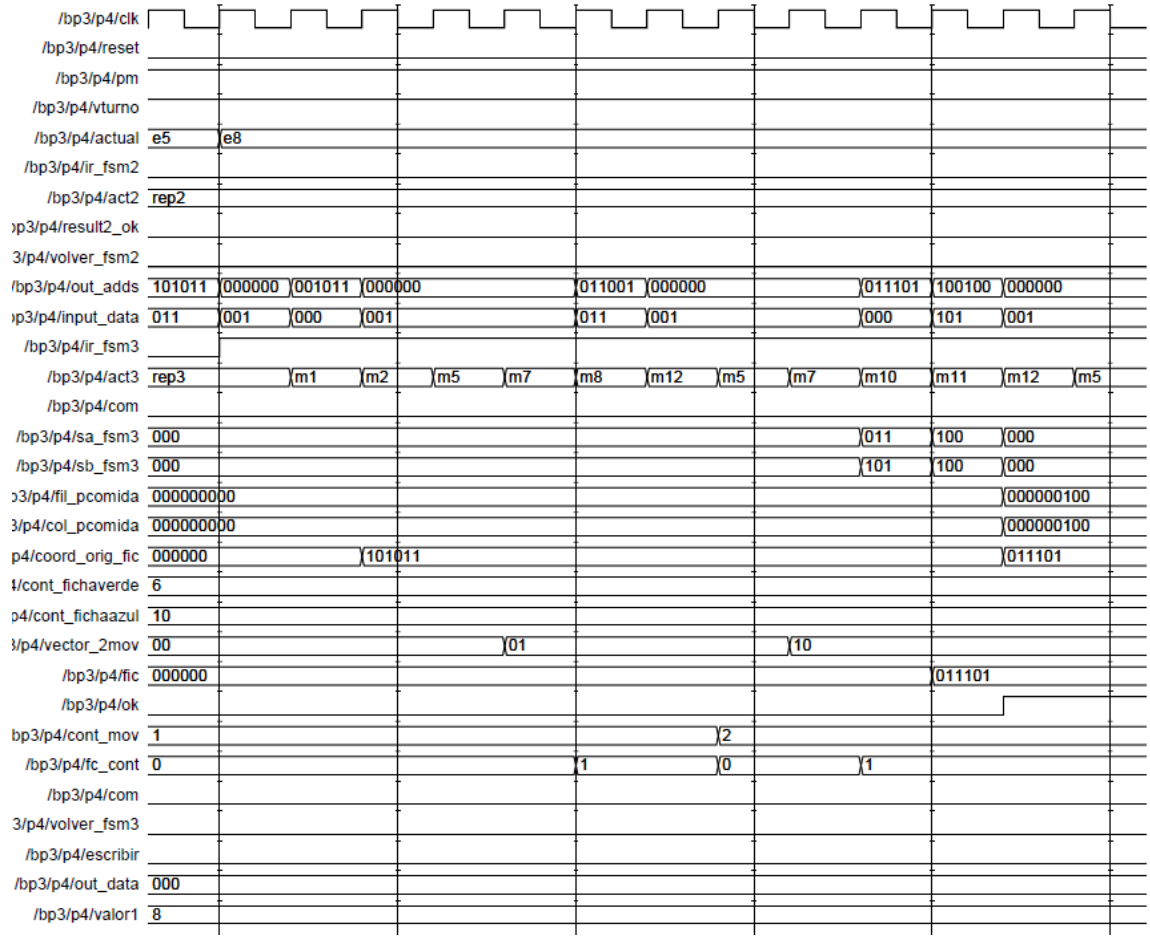


Figura 4.13 Simulación Árbitro 3-2 (fsm3)

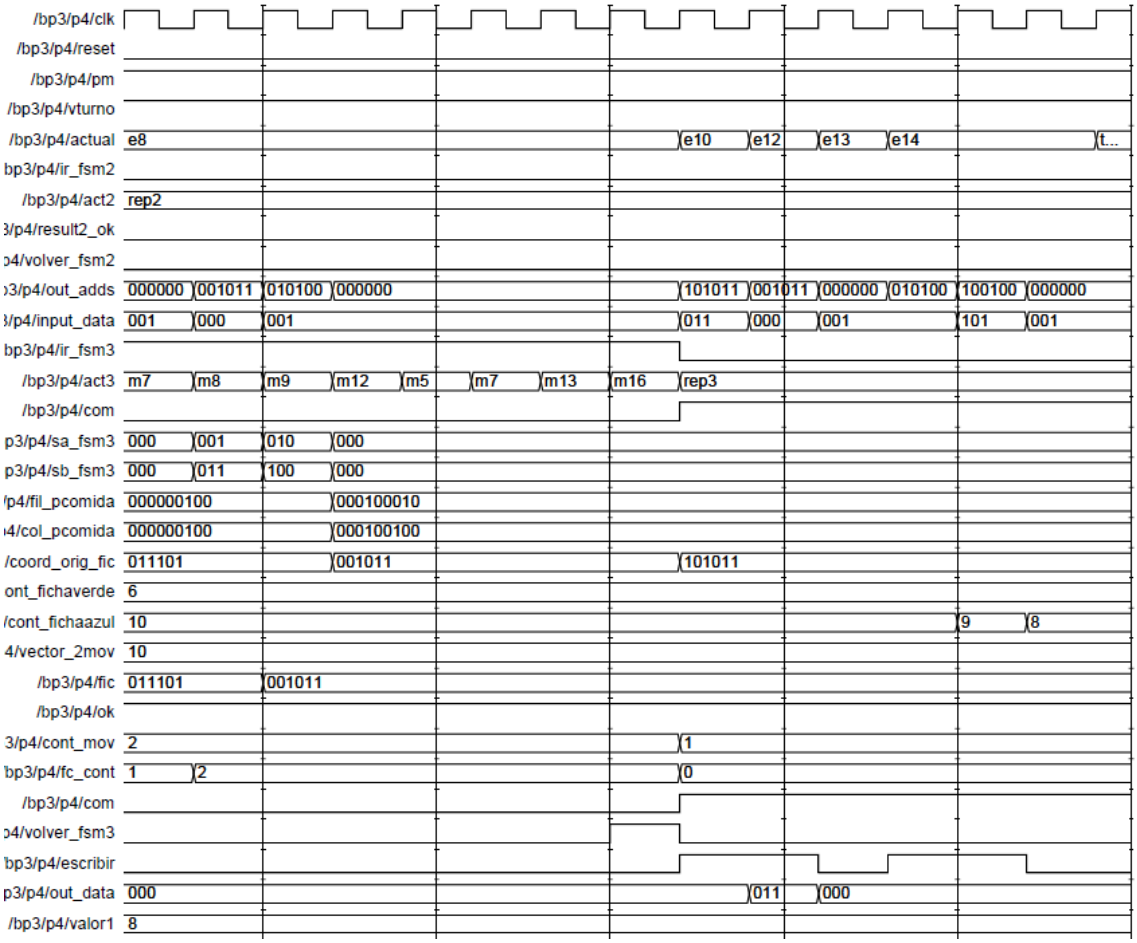


Figura 4.14 Simulación Árbitro 3-3 (fsm3)

- **Movimiento simple con una dama**

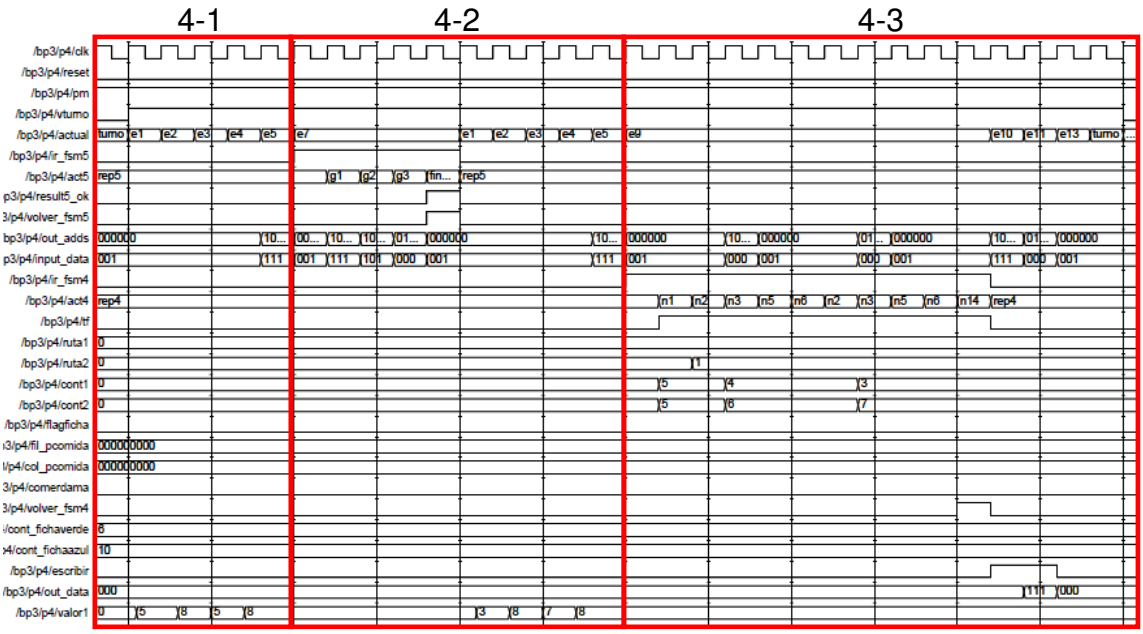


Figura 4.15 Simulación Árbitro 4

Para la realización de este movimiento se utilizan las máquinas de estado secundarias ‘fsm4 y fsm5’, ya que estas son las destinadas para los movimientos que se realizan con damas. Al igual que en el caso del movimiento simple del peón que se ha visto anteriormente, las señales más importantes son ‘out_adds e input_data’, junto con ‘cont1 y cont2’ que nos indican las coordenadas de las casillas por las que pasamos, antes de llegar al destino.

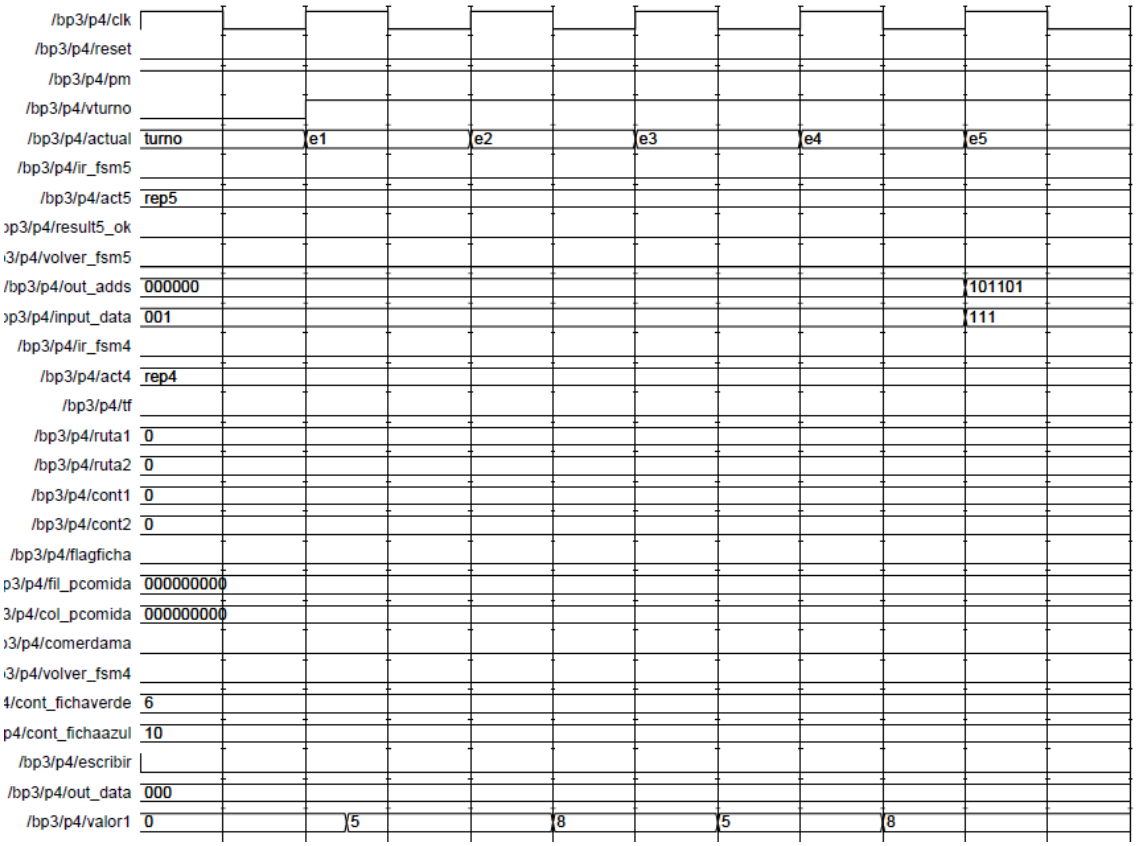


Figura 4.16 Simulación Árbitro 4-1 (fsm5)

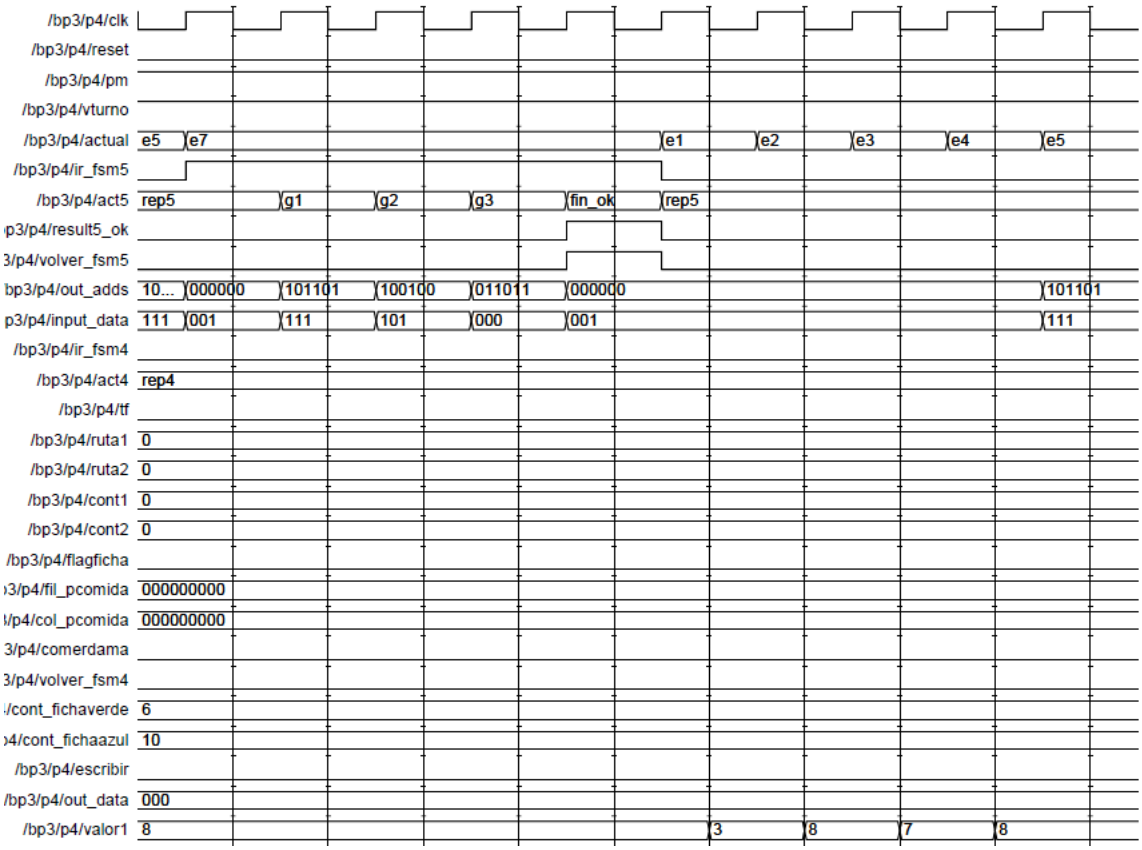


Figura 4.17 Simulación Árbitro 4-2 (fsm4)

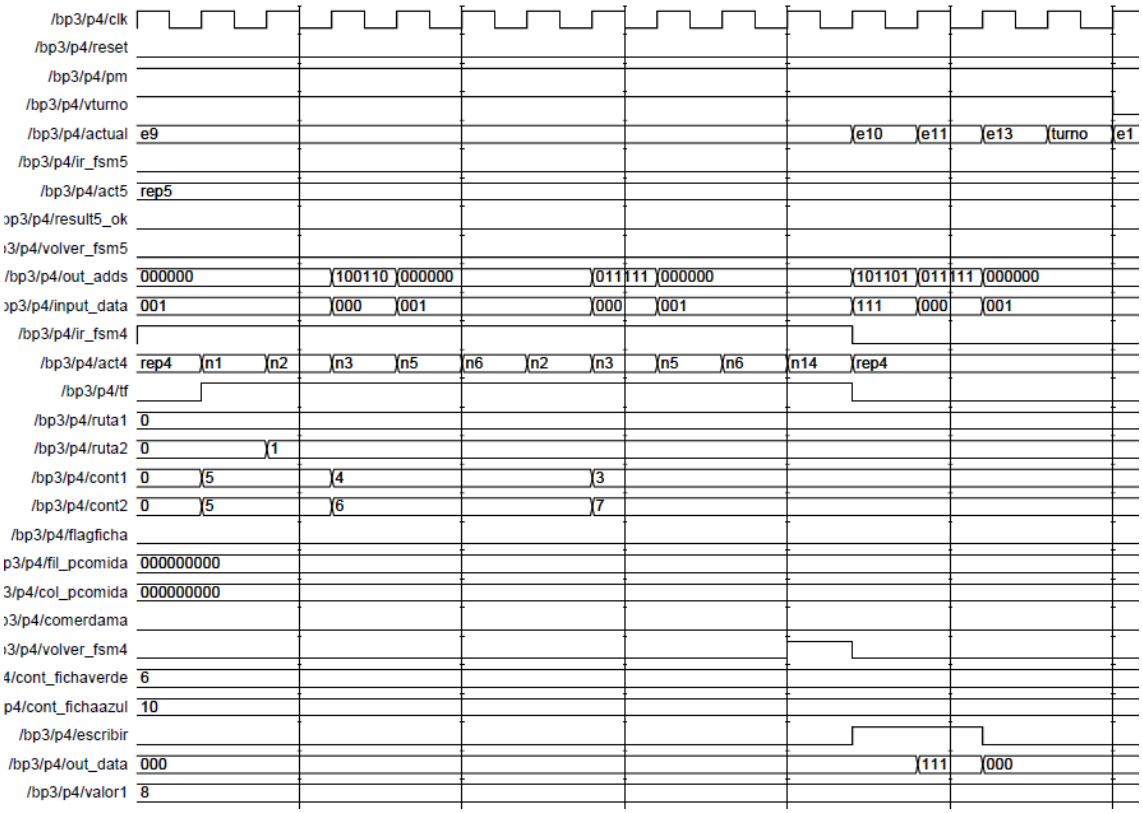


Figura 4.18 Simulación Árbitro 4-3 (fsm4)

- **Captura simple con una dama**

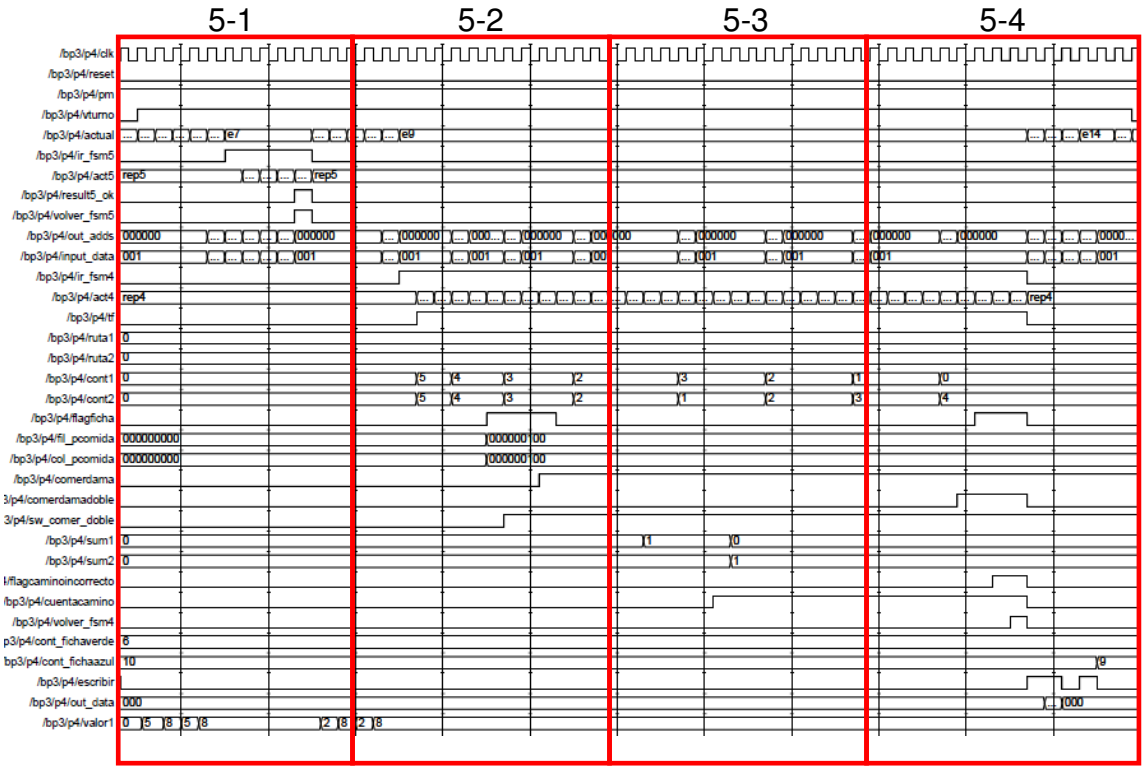


Figura 4.19 Simulación Árbitro 5

En esta última simulación se ha recreado la captura de una ficha con una dama con intención de volver a comer pero sin conseguirlo, es decir, se ha activado el interruptor de comer múltiple 'sw_comer_doble' pero la dama no puede realizar dicha operación porque no hay ningún camino que se lo permita, por lo tanto el movimiento termina con la captura de una única ficha y sin opción de volver a realizar otra captura.

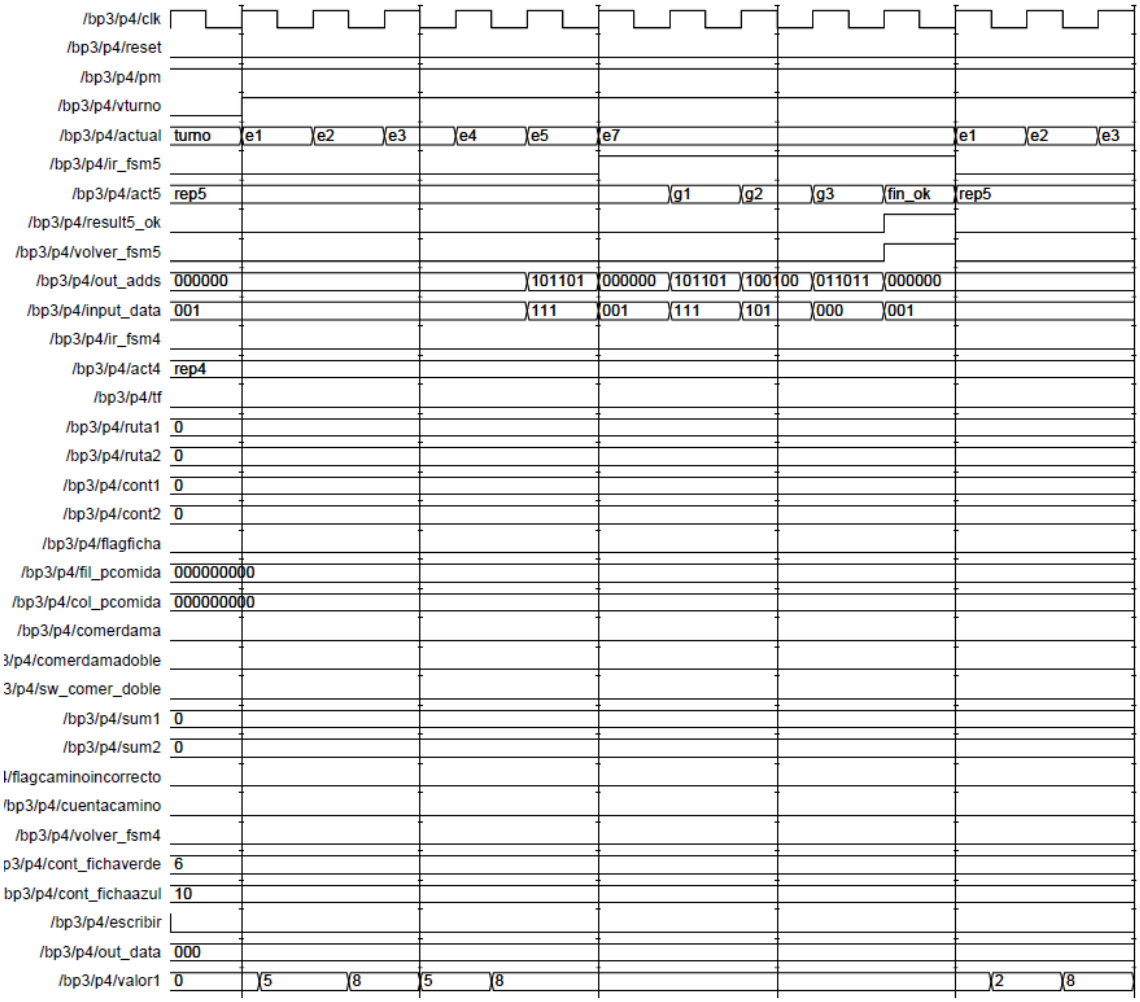


Figura 4.20 Simulación Árbitro 5-1 (fsm5)

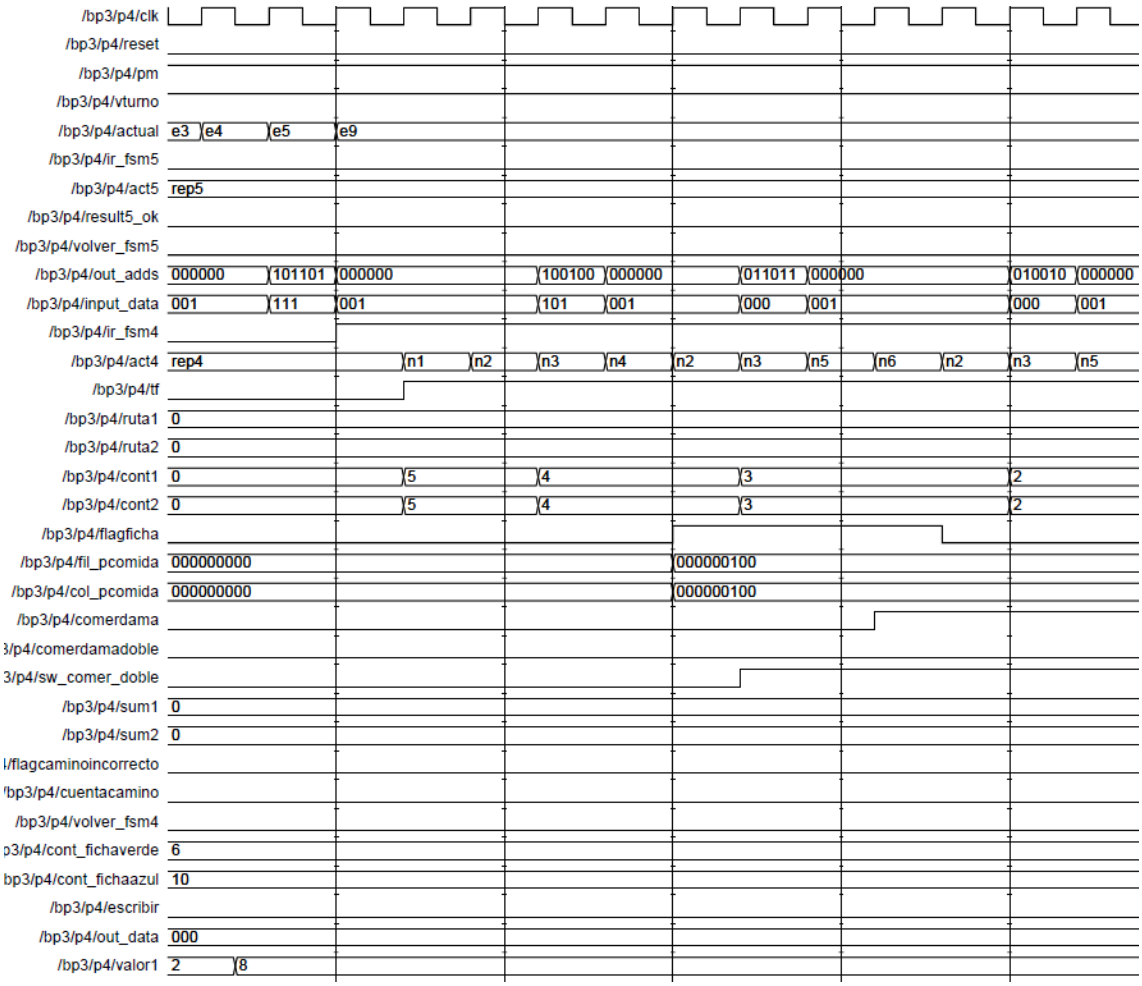


Figura 4.21 Simulación Árbitro 5-2 (fsm4)

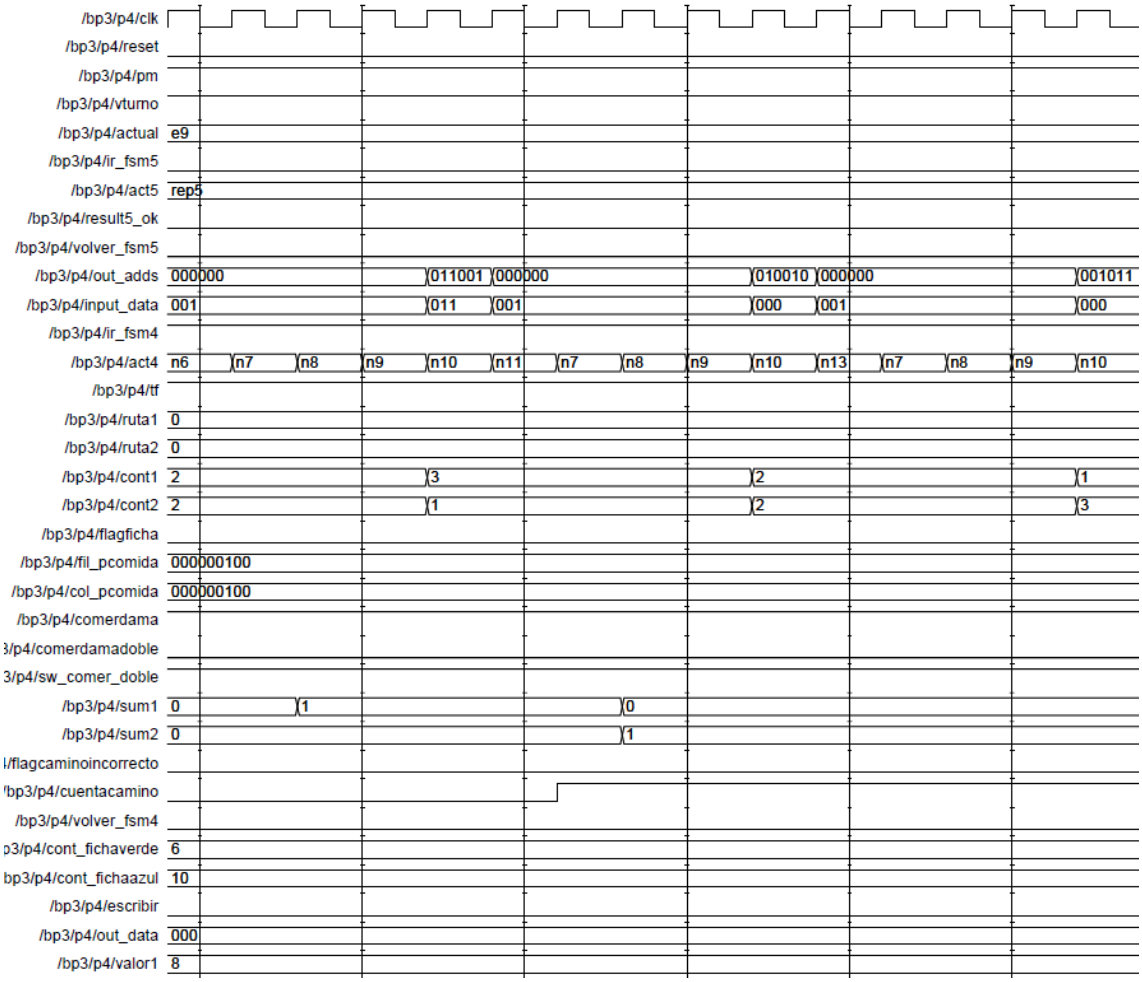


Figura 4.22 Simulación Árbitro 5-3 (fsm4)

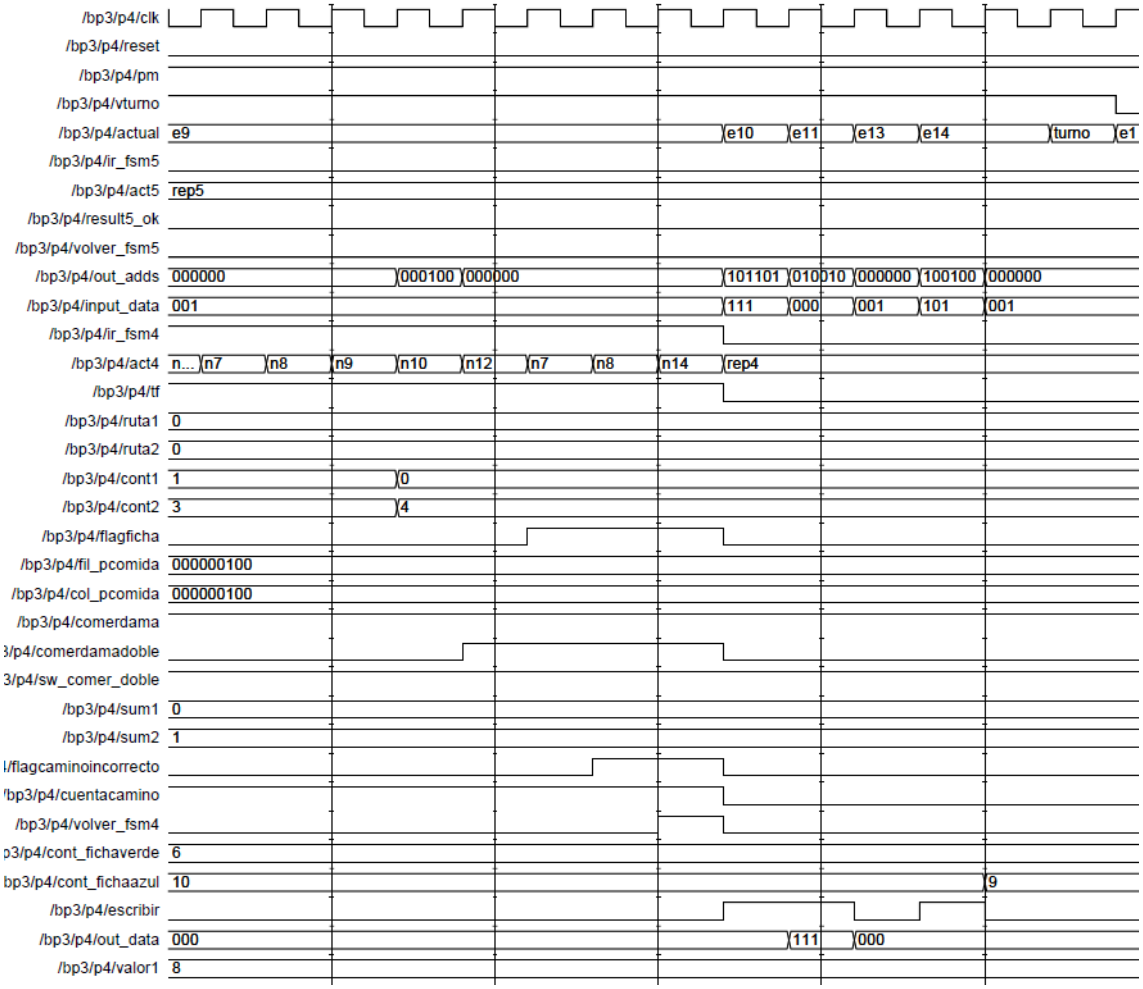


Figura 4.23 Simulación Árbitro 5-4 (fsm4)

Para concluir, la simulación de captura múltiple con una dama no se ha plasmado en la memoria porque era demasiado grande, pero se ha dejado en formato digital en el archivo (Árbitro6.pdf). Comentar también que todas las simulaciones están en formato digital para una mejor visualización si fuera pertinente.

5 Presupuesto

A continuación se recogen los costes globales de la realización de éste Proyecto Fin de Carrera. Ver Tabla 11.

Fases	Horas
Documentación	200
Desarrollo del sistema propuesto y pruebas de funcionamiento	500
Redacción de la memoria del proyecto	210

Tabla 11 Fases del proyecto

Teniendo en cuenta que el Colegio Oficial de Ingenieros Técnicos establece una tarifa de 24 €/hora, el coste ascendería a 21.840 €.

El resto de gastos se recogen en la Tabla 12.

Concepto	Importe (€)
Ordenador de sobremesa	900
Placa de evaluación	80
Teclado matricial	5,21
Costes indirectos	1500

Tabla 12 Costes de material

A partir de los datos obtenidos, el presupuesto total asciende a 24.325,21 € sin IVA y 28.704 € con IVA.

6 Conclusiones

El principal objetivo de este proyecto consistía en implementar un juego de mesa, en este caso las damas, usando una FPGA, y evaluar su complejidad. Dichos objetivos se han alcanzado satisfactoriamente.

Después de realizar el diseño e implementarlo, se realiza la siguiente propuesta de trabajo, teniendo en cuenta tres cosas: Nº de personas que forman el grupo de trabajo, tiempo para llevarla a cabo y conocimientos previos del alumno en la asignatura. Se proponen dividir la realización del sistema en una parte básica obligatoria y otra opcional:

- Trabajo básico: Representar el tablero, indicar el cambio de turno, movimiento simple del peón, captura simple del peón, transformar un peón en dama y que esta se mueva como el peón, detección de fin de partida, ya sea por victoria o por tablas, indicar a través de los leds de la placa de simulación las coordenadas introducidas, implementar las funciones de teclado, tablero, VGA y RAM.
- Trabajo opcional (mejoras): Y para las ampliaciones se sugiere, que un peón pueda realizar capturas múltiples, que la dama pueda desplazarse varias casillas en un sólo movimiento y que pueda realizar capturas tanto múltiples como con desplazamiento (ver Figura 3.2) y representar por pantalla las coordenadas introducidas.

La realización de este proyecto me ha permitido ampliar mis conocimientos sobre el lenguaje VHDL, siendo este un campo de aplicación amplio y con grandes perspectivas de futuro.

7 Trabajos futuros

Como desarrollos futuros pueden incluirse las siguientes líneas de estudio:

- Implementar la función de soplar. Aunque, es una regla que no se utiliza en los torneos porque hay árbitros, sí que sería interesante el implementarla ya que es una regla muy extendida entre jugadores no profesionales.
- Captura a la calidad y a la cantidad. Son reglas que no se han implementado por su complejidad. Además, ambas reglas deben implantarse a la vez, ya que una complementa a la otra.
- Mejorar el movimiento de captura múltiple con la dama. Es decir, que cuando la dama realice capturas múltiples en diferentes direcciones, no haga falta pulsar un interruptor para la realización del movimiento.

Bibliografía

MANUALES

[1] Luis Mengibar y Fernando Casado “Manual Xilins ISE”, UC3M

[2] Mario García y Fernando Casado “Manual de laboratorio de Microelectrónica”. UC3M, 2007

LIBRO

[3] Smith Douglas “HDL CHIP DESIGN”,1998

DOCUMENTOS

[4] Guillermo Carpintero, Susana Patón, Marta Portela “Sistemas digitales basados en microprocesadores”, UC3M, 2009.

[5] Xilinx “User guide Spartan-3”, 2005

[6] J.I Artigas y L.A Barragán "Diseño con FPGA". Universidad de Zaragoza 2007 (http://diec.unizar.es/~barragan/docto_archivos/T3.pdf)

[7]M.L.López Vallejo y J.L. Ayala Rodrigo “FPGA: nociones básicas e implementación”, UPM, 2004. (http://www.lsi.die.upm.es/~marisa/docencia/fpga_a2_2004.pdf)

DIRECCIONES DE INTERNET

[8] http://www.terra.es/personal2/jlgsanz/damas/reglas.htm

[9]http://www.dma.fi.upm.es/docencia/primerciclo/matrecreativa/Juegos/damas/data/objetivo.html

[10] http://www.ludoteka.com/damas.html

[11] http://www.damasweb.com/spanish/

[12]http://www.digilentinc.com/Products/Detail.cfm?Nav1=Products&Nav2=Programmable&Prod=S3BOARD