UNIVERSIDAD TECNOLÓGICA DE BOLÍVAR

FACULTAD DE INGENIERÍAS

Título: DISEÑO DE UN NÚCLEO DE PROPIEDAD INTELECTUAL PARA COMBINAR VIDEO DVI E INFORMACIÓN GRÁFICA HMI

Autor: ALFREDO ENRIQUE PONCE IGLESIAS

Jurado

Jurado

Director: Ph.D. Juan Carlos Martínez Santos

Cartagena, Noviembre de 2015

DISEÑO DE UN NÚCLEO DE PROPIEDAD INTELECTUAL PARA COMBINAR VIDEO DVI E INFORMACIÓN GRÁFICA HMI

ALFREDO ENRIQUE PONCE IGLESIAS

Director:

Ph.D. JUAN CARLOS MARTÍNEZ SANTOS

Universidad Tecnológica de Bolívar
Facultad de Ingenierías
Programa de Ingeniería Eléctrica
Cartagena

Noviembre de 2015

DISEÑO DE UN NÚCLEO DE PROPIEDAD INTELECTUAL PARA COMBINAR VIDEO DVI E INFORMACIÓN GRÁFICA HMI

ALFREDO ENRIQUE PONCE IGLESIAS

Trabajo de grado para optar al título de

Magíster en Ingeniería Electrónica

Director:

Ph.D. JUAN CARLOS MARTÍNEZ SANTOS

Universidad Tecnológica de Bolívar
Facultad de Ingenierías
Cartagena

Noviembre de 2015

Resumen

La Armada Nacional de Colombia ha incursionado en el desarrollo de Sistemas de Armas para sus unidades a flote, los cuales permiten controlar con precisión armas utilizadas para la defensa de la soberanía. Uno de los procesos de diseño de mayor importancia es el desarrollo de la Interfaz Hombre-Máquina, la cual permite al ser humano interaccionar con el sistema; por tal razón, dentro de los requisitos a satisfacer están poseer la mayor claridad en los despliegues, así como brindar comodidad y facilidad en su operación. Mediante este trabajo se logró implementar un sistema gráfico basado en un núcleo de propiedad intelectual, desarrollado para arreglos de compuertas programables en campo, utilizando un esquema de arquitectura combinada con módulos desarrollados en lenguaje de descripción de hardware y procesador embebido que ejecuta programas desarrollados en lenguaje secuencial.

Agradecimientos

Sin lugar a dudas, en primera instancia quiero agradecer a Dios que siempre me ilumina y a mi amadísima esposa por su férrea paciencia y apoyo permanente, que me impulsaron de manera decisiva para superar los momentos difíciles que nunca faltaron en el desarrollo del trabajo; seguidamente, quiero agradecer al Ingeniero Cristian Jiménez Barrera, ese gran amigo que durante el desarrollo del Sistema "Barracuda" siempre aportó ideas fundamentales para el desarrollo de este trabajo; al Ingeniero Agenor Polo Zabaleta, quien de manera permanente me colaboró en el análisis y desarrollo de esta monografía y del trabajo. A mis padres y en especial a mi papá que siempre me dio voz de aliento para alcanzar la culminación del trabajo; finalmente, al profesor Juan Carlos Martínez Santos por haber sido un excelente guía en la visión global del trabajo de grado y por su acertada dirección y permanente compromiso con los objetivos del presente trabajo.

Índice general

Lista de Figuras

Lista de Tablas

Capítulo 1.
Introducción

En el presente capítulo se hace una ambientación en los diferentes conceptos relativos al manejo de interfaces de video aplicables a Sistemas de Armas, mostrando argumentos que llevan al planteamiento de soluciones basadas en dispositivos de lógica programable, a la vez que se expone la pregunta de investigación, los objetivos generales y específicos, de manera que se conforme un apropiado preámbulo que permita acometer el desarrollo de un diseño que cumpla con el propósito del trabajo de grado.

1.1. Cómo leer el presente trabajo

Para una adecuada comprensión del presente trabajo, se usan las siguientes convenciones:

- Los módulos del diseño se citan entre <>: <Nombre_módulo>.
- Las señales se citan en letra cursiva : *Nombre_de_señal*.
- Las siglas de citan en letra mayúscula y se encuentran desglosadas en el capítu-

lo de siglas: NOMBRE_SIGLA.

- La referencia más usada es ***IP core***, la cual se cita en letra cursiva resaltada.

1.2. Antecedentes

Tanto la industria como las entidades militares requieren hacer uso de tecnologías que permitan operar sistemas para cumplir con misiones específicas. Particularmente, la Armada Nacional debió enfrentar el desarrollo de una plataforma HMI que permitiera visualizar toda la información de sus sistemas de combate, en una manera amable, práctica y con protocolos de comunicación sencillos de utilizar, basado en la experiencia obtenida en el desarrollo de Sistemas de Armas para unidades a flote desde el año 1999. En un Sistema de Armas típico, debe desplegarse a los usuarios, al menos, la siguiente información:

- Estado de armas y sensores: ángulos de operación, estado de alistamiento, alarmas, alertas.

- Estado de comunicaciones de módulos operativos: Ethernet, RS232, RS422, RS485, I2C, SPI, PSP.

- Video de equipos de detección: cámaras, directores de tiro, radares.

De la misma manera, deben apoyar la gestión de operación, permitiendo la utilización de funcionalidades como:

- Dispositivos apuntadores: ratón.
- Captura de datos desde dispositivos e/s: palancas de control (joysticks), pantallas táctiles.
- Generación de retículos de apuntamiento.
- Generación de menús de diálogo y de configuración.
- Almacenamiento de datos operativos.
- Almacenamiento de datos de configuración.
- Conversiones de protocolos de comunicación.

En busca del brindar a los sistemas las mencionadas funcionalidades, se ha acometido el desarrollo de una solución basada en núcleo de propiedad intelectual que permita cumplir en una sola tarjeta electrónica las tareas de HMI típicas de los Sistemas de Armas.

1.3. Marco teórico

El diseño de interfaces hombre-máquina (HMI) ocupa un importante lugar en el extenso mundo del desarrollo de soluciones integrales para la industria y, particularmente, de los Sistemas de Armas. El diseño de soluciones en arreglos de compuertas programables en campo (FPGAs) bajo lenguaje de descripción de hardware (HDL) es una tendencia que se encuentra a la vanguardia del diseño digital [1, 2]. Un núcleo

de propiedad intelectual (***IP core*** Intellectual Property Core) es un bloque de lógica que se utiliza para ser sintetizado en una FPGA o para producir un circuito integrado de propósito específico (ASIC) como parte de un producto. Al ser elementos esenciales para ser re-utilizados en diseño, los ***IP core***s forman parte de la creciente tendencia de la industria en Automatización de Diseño Electrónico (EDA) debido al uso repetido de componentes previamente diseñados. Idealmente, un ***IP core*** debe ser completamente portátil, es decir, capaz de ser utilizado por cualquier metodología de diseño o producto de tecnología de diferentes fabricantes [3]. Los receptores/transmisores asíncronos universales (UARTs), unidades de procesamiento central (CPUs), los controladores Ethernet, interfaces PCI son todos ejemplos de ***IP core***s [4].

La flexibilidad y capacidad (tanto en densidad lógica como en velocidad) que ofrecen los dispositivos de lógica programable de tipo FPGA actualmente, permite implementar diseños de alto desempeño. Dos de los principales fabricantes de estos dispositivos, Altera y Xilinx, han venido promoviendo desde hace algunos años el uso de sus respectivos microprocesadores soft-core (Nios y Microblaze), que pueden ser sintetizados empleando los recursos lógicos de la FPGA. Recientemente también han desarrollado chips híbridos que integran procesadores hard-core de alto rendimiento tipo ARM, contando con interfaces de alta velocidad para permitir la interacción de la FPGA con el microprocesador dentro del chip. El hecho de poder contar con

uno o más procesadores de propósito general dentro del chip, sean hard o soft, ha permitido que en los diseños basados en FPGA pueda hacerse uso de la programación de hardware con lenguajes HDL (Verilog [5], VHDL [6]) y la programación de software mediante lenguajes de medio/alto nivel (C, C++) para procesadores. Debido a lo anterior, resulta natural emplear metodologías de diseño hardware/software, donde de acuerdo a la aplicación específica, se decide cuales funcionalidades de un sistema serán implementadas en hardware mediante HDL y cuáles serán implementadas en software ejecutado desde el microprocesador. Normalmente, para llevar a cabo esta partición hardware/software se tiene en cuenta la complejidad, secuencia, paralelismo y restricciones de tiempo características de cada proceso.

A través del presente trabajo se implementó un núcleo de propiedad intelectual que permite cumplir con las funciones de despliegue de la información en pantalla que requiere un Sistema de Armas estándar, para el cual, al menos, se necesita contar con las siguientes capacidades:

- Captura y despliegue de una señal de video DVI procedente de sensores y equipos asociados al Sistema de Armas.

- Despliegue de información alfanumérica, en forma de letras, números y símbolos.

- Despliegue de mapas de bits de variados tamaños y colores.

- Despliegue de ventanas de usuario, para organizar temáticamente la información del sistema.

- Despliegue de figuras (líneas, cajas, vectores, círculos y arcos).

- Despliegue de un dispositivo apuntador (ratón).

- Despliegue de video en un monitor dotado de puerto VGA y/o DVI.

1.4. Objetivos

Para el presente proyecto, se planteó como pregunta de investigación: ¿que pasaría si no se desarrolla una interfaz hombre máquina más avanzada para ser utilizada en los Sistemas de Armas de la Armada Nacional? La respuesta se desglosó así:

- No se lograría la velocidad ni la claridad en el despliegue de información procedente de un complejo conjunto de equipos asociados a los Sistemas de Armas.

- No se tendría propiedad intelectual de la solución en HMI, que limitaría el avance en el desarrollo de los Sistemas de Armas y el escalamiento del mismo en requerimientos futuros.

- Los equipos asociados a los Sistemas de Armas tendrían que ser de menores prestaciones técnicas, lo cual limitaría su eficiencia como un conjunto funcional.

- Los costos asociados al desarrollo del Sistema de Armas se verían impactados negativamente por el uso de tecnologías que paulatinamente van cayendo en desuso u obsolescencia.

Consecuentemente se plantearon los siguientes objetivos:

a. General.

Disponer de un sistema HW/SW para combinar información HMI con una imagen procedente de una fuente de video, que mejore las características funcionales de calidad de despliegue, velocidades y usabilidad aplicado en los Sistemas de Armas "Escorpión" de la Armada Nacional.

b. Específicos

- Obtener el control maestro del ***IP core*** mediante la utilización de lenguaje Verilog para el manejo de procesos secuenciales, en el desarrollo del módulo.
- Implementar el ***IP core*** en una tarjeta dotada de las capacidades mínimas de referencia, con el fin de evaluar su desempeño.
- Comparar el desempeño entre el ***IP core*** implementado en una tarjeta de desarrollo y el esquema de despliegue actualmente en uso en el Sistema de Armas "Escorpión" de la Armada Nacional.
- Generar la documentación para el uso del ***IP core***.

Capítulo 2.
Preliminares: el sistema "Escorpión"

La Armada Nacional desarrolló entre los años 2.002 y 2.004 el Sistema de Armas "Escorpión", a través del cual se logró controlar armas y sensores en sus buques fluviales. En este capítulo se presenta la configuración del esquema de video utilizado y se exponen las restricciones que éste presenta en términos de capacidades y versatilidad en su uso, que representa el punto de partida para el desarrollo del presente trabajo.

2.1. Funcionamiento general

El sistema escorpión está conformado por un conjunto de equipos cuya misión final es proveer medios de defensa a las unidades fluviales. Está conformado por tres sub-sistemas funcionales:

- Control. Ubicado en la consola de operación, está dotado de palancas de mando, botones y elementos programables que procesan las órdenes del operador al sistema.

- Motorización. Corresponde al conjunto de elementos electro-mecánicos que

traducen las órdenes de control en movimiento mecánico: rotación de la casamata, elevación del arma y cámara, sistema de cargue del arma.

- Gráfico. Corresponde a los elementos asociados a la captura de imágenes y despliegue de información visual al operario.

- Armamento.

De manera generalizada, el sistema tiene el siguiente ciclo de funcionamiento:

- A través del muestreo de la palanca de mando y botones, el sub-sistema de control recibe las diferentes órdenes que el operario del sistema quiere dar al mismo: movimiento transversal de la casamata, elevación del arma y cámara, acercamiento/alejamiento/enfoque de la imagen, cargue y disparo del arma.

- El sub-sistema de control procesa la información y la convierte al protocolo utilizado por los servo motores del sub-sistema de motorización, al igual que las órdenes a la cámara y arma.

- El sub-sistema gráfico despliega la imagen procedente de la cámara (sensor principal del sistema) ubicada en la casamata y recibe del sub-sistema de control las órdenes de generación de información en pantalla.

- El sub-sistema gráfico genera los diferentes caracteres en pantalla que indican el estado operativo del sistema.

- El operario se retroalimenta del estado del área de operaciones a través del subsistema gráfico y decide qué nuevas órdenes transferirá al sistema de control a través de la palanca de mando y los botones, cerrando así el ciclo descrito.

2.2. Esquema de video

La interfaz gráfica HMI del Sistema "Escorpión" está conformada por los siguientes elementos (fig. 2.1):

A. Sistema de cámara combinada de visión diurna e infrarroja.

B. Tarjetas OSD XBOB-3.

C. Tarjeta de consola.

D. Monitores de video.

A continuación se hace una explicación más detallada de estos componentes.

A. Cámara combinada de visión diurna e infrarroja.

Posee dos salidas de video compuesto NTSC; representa el medio para visualizar el área de operaciones y debe permitir la detección de los blancos y su reconocimiento a una distancia mayor a la del alcance máximo del arma del sistema, lo cual determina su mínimo campo de visión FOV (Field Of View).

B. Tarjetas OSD XBOB-3 [7].

Generan los despliegues en pantalla (fig. 2.2) y están dotadas de una interfaz

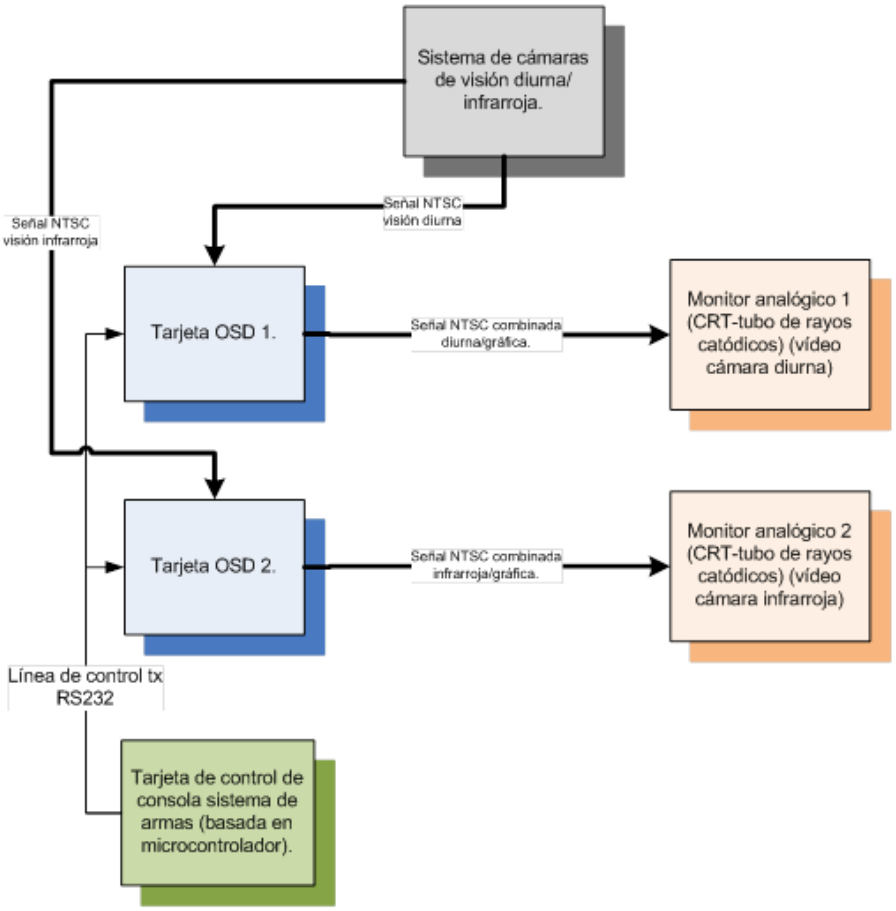

Figura 2.1: Esquema de video Sistema de Armas "Escorpión"

RS232, además de una entrada y una salida de video compuesto. Se utilizan dos tarjetas en el Sistema puesto que se manejan dos señales de video (diurno e infrarrojo), cada una de los cuales alimenta con señal de video su monitor asociado; proveen el medio para superponer a las imágenes de las cámaras el conjunto de caracteres alfanuméricos que constituyen la información que se entrega al operario para la explotación de las capacidades del Sistema de Armas.

Las tarjetas OSD están dotadas de tres memorias [7]:

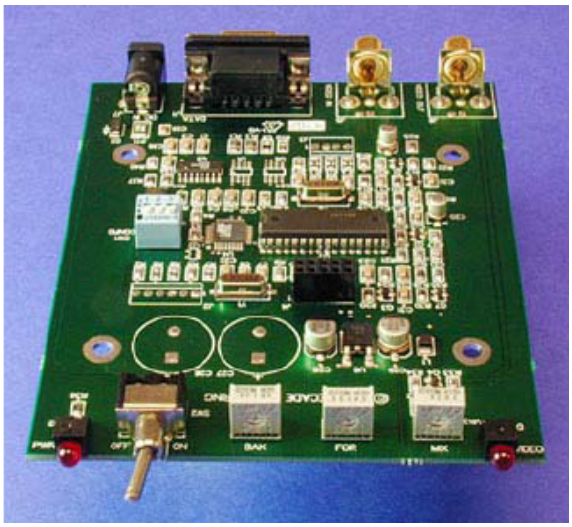

Figura 2.2: Tarjeta OSD XBOB-3

- Una memoria RAM de despliegue de 680 posiciones de 8 bits, en una distribución de 40 columnas por 17 filas. En cada posición de esta memoria se almacena el código ASCII del caracter que se desea desplegar.

- Una memoria generadora de caracteres CGROM (Character Generator Read Only Memory) de 256 posiciones (fig. 2.3). En cada posición de esta memoria, el fabricante ha almacenado un mapa de 12x13 bits que contiene los píxeles de cada caracter de las diferentes fuentes de letras disponibles. Por tratarse de una ROM, estos mapas no pueden modificarse.

- Una memoria generadora de caracteres CGRAM (Character Generator Random Access Memory) de 64 posiciones (fig. 2.4). Al arranque de la tarjeta, ésta almacena en cada posición de la CGRAM mapas de 12x13 bits de caracteres especiales; el usuario puede, a través de comandos seriales, ha-

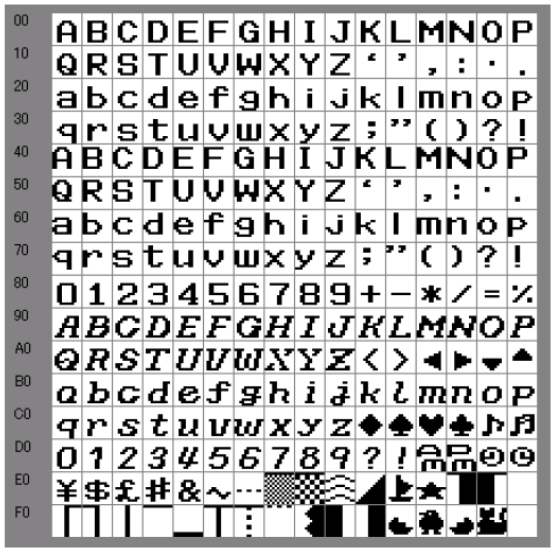

Figura 2.3: Mapas en memoria CGROM de caracteres XBOB-3

cer uso de los mismos y/o modificarlos para generar otros símbolos en el despliegue .

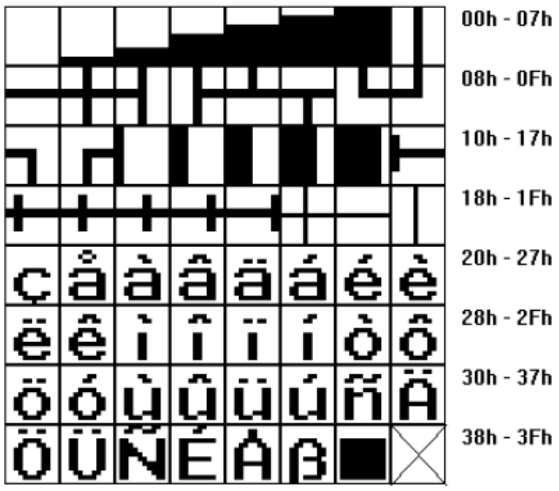

Figura 2.4: Mapas por defecto en memoria CGRAM de caracteres XBOB-3

Para modificar el despliegue que se realiza en pantalla, debe cargarse en la me-

moria RAM de despliegue, a través de un conjunto de comandos transferidos por el puerto RS232, la posición en fila, columna y el código ASCII de cada caracter a desplegar. Por cada posición de RAM de despliegue, la tarjeta OSD extrae de las memorias CGROM ó CGRAM (de acuerdo al código de caracter almacenado) el mapa de bits que conforma el caracter, lo cual permite un manejo de 320 caracteres (256 fijos y 64 modificables).

Teniendo en cuenta el arreglo de 40 columnas x 17 filas de la memoria RAM de despliegue y considerando que cada posición de un caracter en pantalla la ocupa un mapa de 12x13 píxeles, la tarjeta OSD maneja una resolución de (40*12) x (17*13) = 480 x 221 píxeles.

Figura 2.5: Despliegue Sistema de Armas “Escorpión”

El Sistema de Armas actualmente despliega la siguiente información (fig. 2.5): posición angular del arma bajo control, posición angular del sistema de cámaras,

velocidad de movimiento del arma y la cámara, estado de los sensores de distancia al objetivo, estado de la condición de puntería, información de la operación de los lentes de las cámaras, retículo de puntería, información relativa a los motores que mueven las armas, fecha y hora local, cronómetro de operación y menús de configuración del sistema.

C. Tarjeta de consola.

Es una tarjeta basada en un microcontrolador de 8 bits, que genera los comandos seriales (transfiriéndolos a través de la interfaz RS232) para el control de las tarjetas OSD. A través de esta interfaz, la consola ordena qué caracteres deben desplegarse en los monitores y en qué posiciones.

D. Monitores de video WV-CM1020 de 9" [8].

Son dos monitores, cada uno de los cuales despliega la señal que contiene el video compuesto de su correspondiente cámara, combinada con la información alfanumérica que superpone el Sistema de Armas a través de cada tarjeta OSD, información que es idéntica para ambos monitores de video.

Los monitores, las tarjetas OSD y la tarjeta de control de la consola se encuentran ubicados al interior de la consola de control del sistema, que a su vez se encuentra en el interior del buque; la cámara combinada se encuentra ubicada en el exterior del mismo.

2.3. Restricciones

La configuración descrita en la sección 2.2, a pesar de su funcionalidad, presenta las siguientes limitaciones básicas:

A. En lo que respecta a las tarjetas OSD:

- Los despliegues sólo pueden generarse encasillando un caracter en cada una de las 40 columnas x 17 filas, lo cual limita la capacidad de generar gráficos complejos (fig. 2.6). Generar estos gráficos implica un proceso de codificación en el cual el mapa principal a desplegar debe convertirse a un conjunto de mapas reducidos de 12x13 píxeles, los cuales deben cargarse uno por uno en la CGRAM para posteriormente ordenar su despliegue en pantalla mediante la transferencia del código de dirección y posición en la memoria RAM de despliegue de cada una de las posiciones que conforman el mapa general.
- La máxima resolución (480x221 píxeles) que es posible obtener en la generación de caracteres en las tarjetas OSD se encuentra por debajo, inclusive, de la resolución VGA (640x480 píxeles).
- Toda la gestión de despliegue se maneja a través del puerto serial, lo cual determina una ruta crítica en la velocidad de refresco y una limitante en la calidad, cantidad y dinámica de despliegues en pantalla.

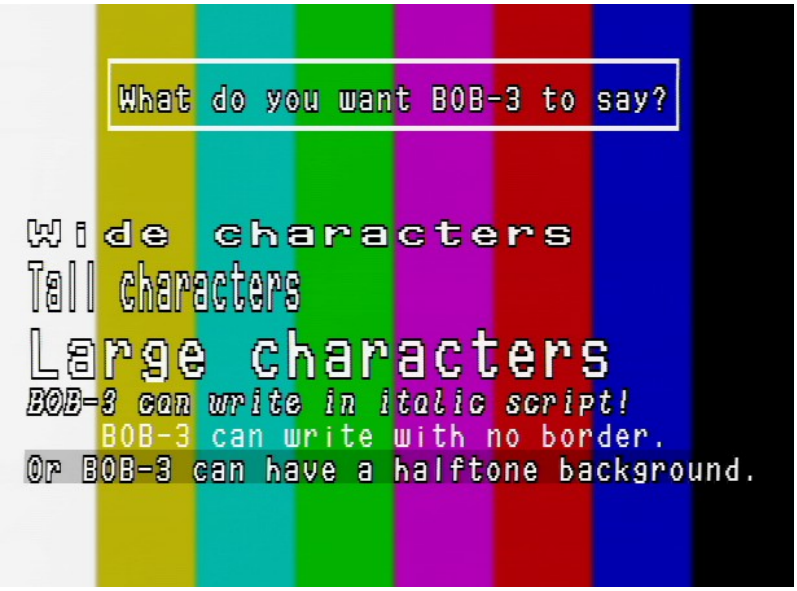

Figura 2.6: Despliegue tarjeta OSD XBOB-3

B. Debe existir una tarjeta OSD por cada cámara (visión diurna e infrarroja) y cada monitor, puesto que en un solo monitor no puede desplegarse la imagen de las dos cámaras al mismo tiempo, lo cual ha implicado el uso de la configuración descrita anteriormente.

C. En el aspecto físico, el volumen que ocupa cada monitor está determinado en mayor parte por la profundidad (309 mm), más que por el ancho o el alto del mismo (220 mm) [8]. El motivo fundamental radica en el uso de pantalla de tubo de rayos catódicos (CRT Cathode Ray Tube), lo cual tiene implicaciones en ubicación, diseño de consolas, consumo de energía y costos de adquisición por su reducido uso en el mercado actual.

Capítulo 3.
Materiales y métodos

En este capítulo se describe el diseño general y detallado del ***IP core***, el cual deriva de las consideraciones en configuración y restricciones del sistema "Escorpión" expuestas en el capítulo 2. Cada sección consta de diagramas de bloques y flujo que ilustran en detalle el funcionamiento del sistema completo y por módulos principales; al final del capítulo se abordan en detalle los diferentes criterios de verificación del trabajo.

3.1. Criterios de diseño planteados

Analizadas la configuración y restricciones asociadas al video del Sistema "Escorpión", se evidenció que para lograr un mejor desempeño del Sistema de Armas hacia el usuario debían alcanzarse tres objetivos principales: aumentar la resolución de la imagen, ofrecer mejores funcionalidades de interfaz gráfica que incorporaran al Sistema mayor dinamismo en el despliegue de la información y desarrollar la solución en una única tarjeta que además pudiera ser utilizada en otras funcionalidades del Sistema.

Una vez determinadas las necesidades se llegó a la conclusión que podría aplicarse como método para implementar el sistema una tarjeta que estuviera dotada de un dispositivo capaz de ser re-programado y que ante todo pudiera escalarse con el paso del tiempo, acorde a las mejoras que fueran requeridas en las nuevas versiones de los Sistemas de Armas. Esto determinó la conveniencia y viabilidad en el uso de dispositivos de lógica programable que permitieran favorecer tal objetivo, llevando al uso de FPGAs en lugar de dispositivos ASICs, teniendo en cuenta que a través de los primeros se puede llegar a versiones robustas del proyecto, con tiempos de desarrollo y relación costo beneficio altamente favorables para la Armada Nacional.

Se pasó por una versión intermedia implementada en una tarjeta de desarrollo Altera DE2-70 [9] donde se logró alcanzar una resolución de 1280x960 píxeles y una profundidad de colores de 8 bits/píxel, versión que sirvió como base para los Sistemas de Armas "Arpón", desarrollados durante los años 2.009 a 2.010 (actualmente en funcionamiento), para finalmente, acorde a los nuevos requerimientos del Sistema de Armas "Barracuda", implementar la interfaz en una tarjeta de desarrollo Altera DE2-115 [10] con las especificaciones mencionadas en el presente trabajo (fig. 3.1).

3.2. Descripción general del sistema

La figura 3.2 ilustra el diagrama de diseño del sistema, cuya arquitectura se determina por las funcionalidades que ha de cumplir, como se describió en el capítulo 1.

Consta de un sistema en chip programable (SOPC) sintetizado en una FPGA,

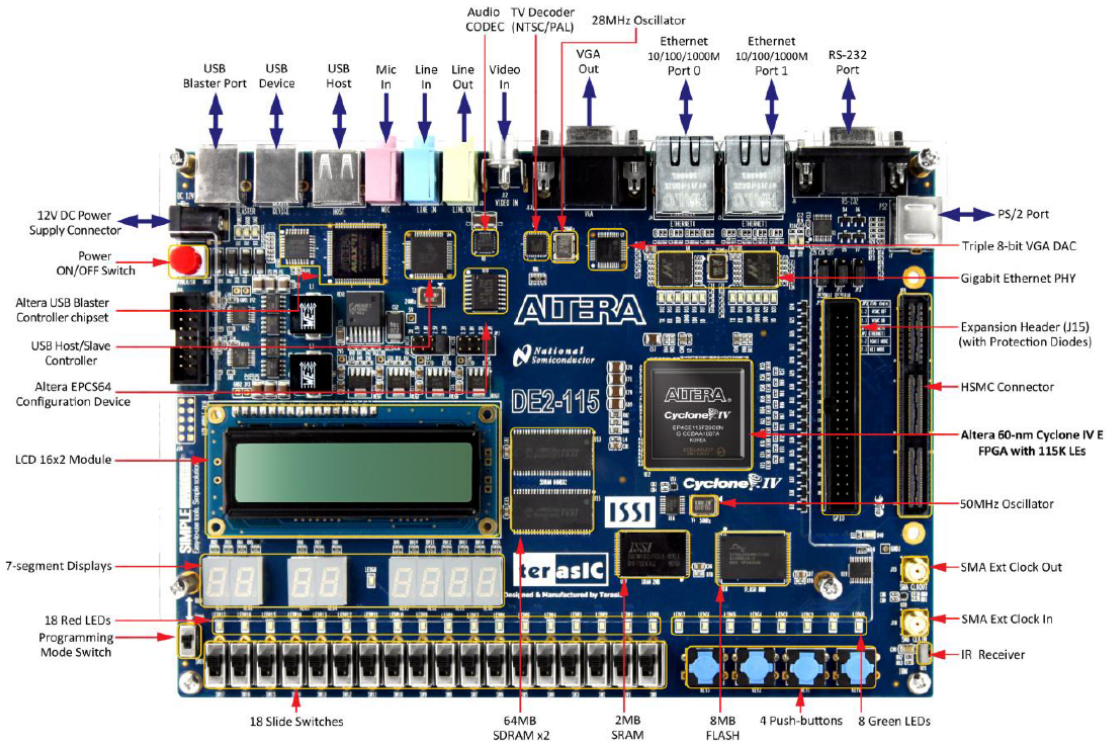

Figura 3.1: Tarjeta Altera Terasic DE2-115

y tiene como función principal combinar con información OSD la señal de video DVI procedente de sensores y equipos asociados. En este caso particular, dicha información está compuesta bien sea de caracteres, figuras geométricas y/o mapas de bits. Además, ya que el diseño debe formar parte de un sistema mayor (Sistema de Armas), se han proveído interfaces de comunicación que permiten tener control remoto del ***IP core***: interfaz de comunicación serial RS-232 e interfaz de comunicación Ethernet. Todas las funcionalidades gráficas fueron implementadas mediante el lenguaje de descripción de hardware Verilog [5] y se ha recurrido al procesador soft-core embebido Nios II para implementar un módulo TCP/IP en lenguaje C con

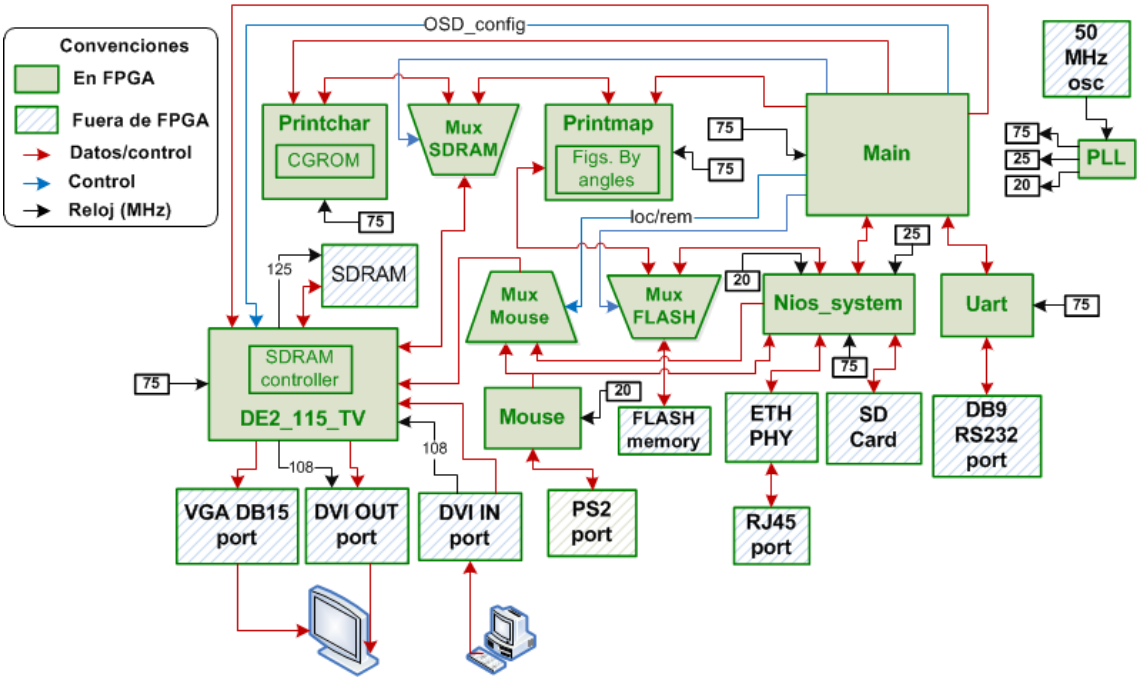

Figura 3.2: Diagrama de bloques ***IP core***

el propósito de disponer de un medio de transferencia de órdenes a través de bus Ethernet.

Se dieron los siguientes pasos para determinar los recursos que el sistema requiere:

A. Cálculo del tamaño mínimo de memoria de video y tipo de memoria.

B. Cálculo del tamaño mínimo de memoria no volátil para almacenamiento de mapas de bits.

C. Determinación de recursos adicionales.

Estos pasos son detallados a continuación.

A. Cálculo del tamaño mínimo de memoria de video y tipo de memoria.

Teniendo en cuenta que una de las funcionalidades que debe tener el ***IP core*** es la de desplegar imágenes procedentes de radares de navegación, los cuales, en los modelos más usados en la Armada de Colombia manejan resoluciones de 1280x1024, se escogió dicha resolución como especificación del sistema, lo cual implica disponer de un espacio de almacenamiento volátil mínimo de 1'310.720 posiciones de memoria por cada OSD. Considerando que por cada OSD se desea manejar una profundidad de colores de al menos 15 bits/píxel más 1 bit/píxel para la función de parpadeo, equivale al mismo número de posiciones en una memoria con bus de datos de 16 bits; en consecuencia, para manejar dos OSDs en el sistema, el tamaño mínimo de memoria de video es de 2,56 MB, necesidad que se ubica dentro del rango bajo de las memorias SDRAM.

B. Cálculo del tamaño mínimo de memoria no volátil para almacenamiento de mapas de bits.

   Se determinó que el sistema pudiera desplegar mapas con tamaños desde 2x2 hasta 640x640 píxeles, lo cual conforma el requerimiento estándar de mapas que requiere el Sistema de Armas "Barracuda" desarrollado por la Armada Nacional. Dado que muchas de las tarjetas de desarrollo basadas en FPGA existentes en el mercado que poseen memoria FLASH externa incorporan al menos 8 MB de capacidad, se ha destinado un espacio máximo de memoria de 7 MB, que permitiría almacenar hasta 520 mapas de 84x84 píxeles (6,99 MB).

C. Determinación de recursos adicionales.

Acorde al diagrama del sistema, se requieren los siguientes recursos adicionales:

- MAX RS232 y puerto DB9.
- Convertidor DAC para salida de video VGA (RGB) y puerto DB15.
- Módulo de E/S video DVI.
- Puerto PS/2 para ratón.
- Chip PHY para Ethernet, y puerto RJ45.
- Ranura para tarjeta SD.

Haciendo un estudio de opciones comerciales en tarjetas de desarrollo basadas en FPGA, se seleccionó como plataforma para implementar el sistema la tarjeta Altera Terasic DE2-115 [10], la cual cuenta con recursos suficientes para cumplir los requerimientos del sistema: FPGA Altera Cyclone IV EP4CE115 con 114.480 elementos lógicos, dotada de memoria RAM embebida de 3.888 Kb y oscilador local de cristal de 50 MHz; en almacenamiento cuenta con una memoria estática (SRAM) de 2 MB con bus de datos de 16 bits, dos memorias sincrónicas dinámicas (SDRAM) de 64 MB c/u con bus de datos de 16 bits, una memoria no vólatil de 8 MB (FLASH) con bus de datos de 8 bits y ranura para memoria externa SD; en conectividad cuenta con chip de capa física Gigabit Ethernet PHY 10/100/1000 Mbps con puerto RJ45, transreceptor MAX232 con puerto DB9, convertidor DAC VGA con conector DB15

y conector PS/2 para dispositivo apuntador (ratón). Para contar con entrada y salida DVI, se determinó utilizar la tarjeta DVI-HSMC Card de Terasic la cual se conecta al puerto HSMC (High Speed Mezzanine Card) [11] de la tarjeta DE2-115 [10].

Como ambiente de programación se usó, para el desarrollo de los módulos HDL-Verilog [5] el ambiente integrado de desarrollo Quartus II V12.1 [12] y para la programación del procesador embebido el ambiente de desarrollo integrado Eclipse V12.1 para el procesador Nios II [13].

3.3. Descripción detallada del sistema

En la figura 3.2 puede observarse el diagrama general de bloques y la interacción entre los distintos módulos del sistema, a saber:

- Módulo <Main> (3.3.1).
- Módulo <Printchar> (3.3.2).
- Módulo <Printmap> (3.3.3).
- Módulo <DE2_115_TV> (3.3.4).
- Módulo <Nios_system> (3.3.5).
- Módulo <Uart> (3.3.6).
- Módulo <Mouse> (3.3.7).

Cada uno de estos módulos es explicado en detalle a continuación.

3.3.1. Módulo <Main>

Se encarga de arbitrar el funcionamiento general del ***IP core*** (fig. 3.3), para lo cual se implementaron los siguientes procesos concurrentes:

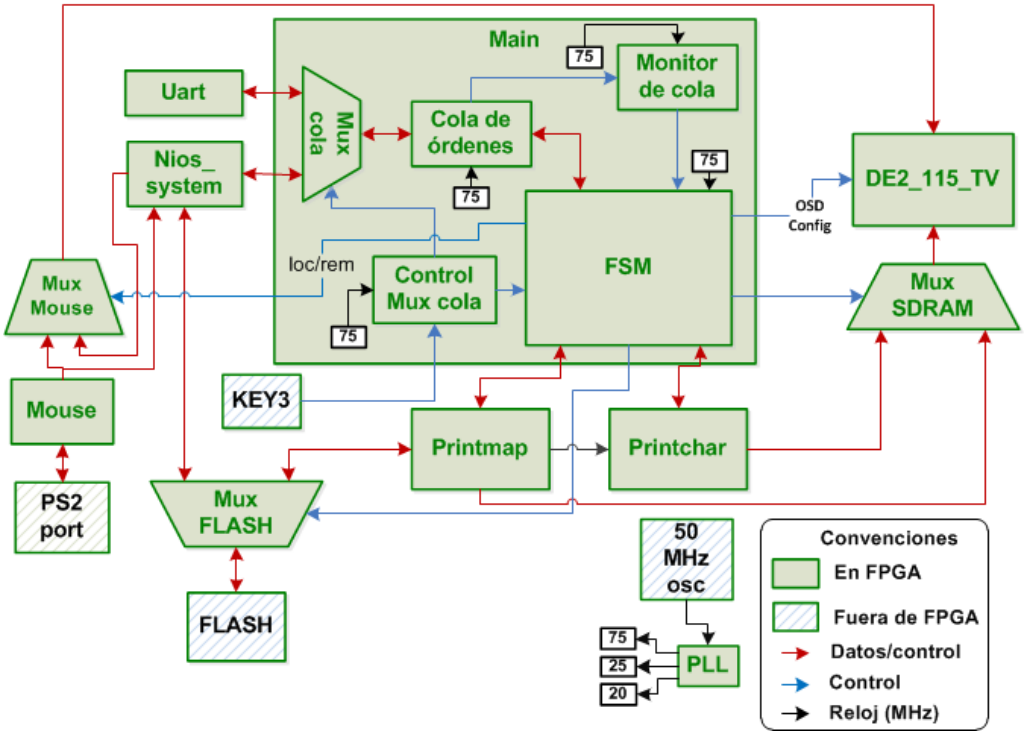

Figura 3.3: Diagrama de bloques <Main>

- Multiplexor de cola de órdenes.
- Monitor de cola de órdenes.
- Escritura/lectura cola de órdenes.
- FSM principal.

Estos procesos son explicados en detalle a continuación.

- Multiplexor de cola de órdenes.

  Este proceso selecciona (entre los módulos <Nios_system> y <Uart>) la fuente de datos que escribe en la cola de órdenes. La selección cambia de un modo al otro cada vez que se detecta que el usuario mantiene presionado el botón KEY3 de la tarjeta por más de 2 segundos (fig. 3.4). Por defecto, el sistema inicia recibiendo las órdenes del módulo <Nios_system>.

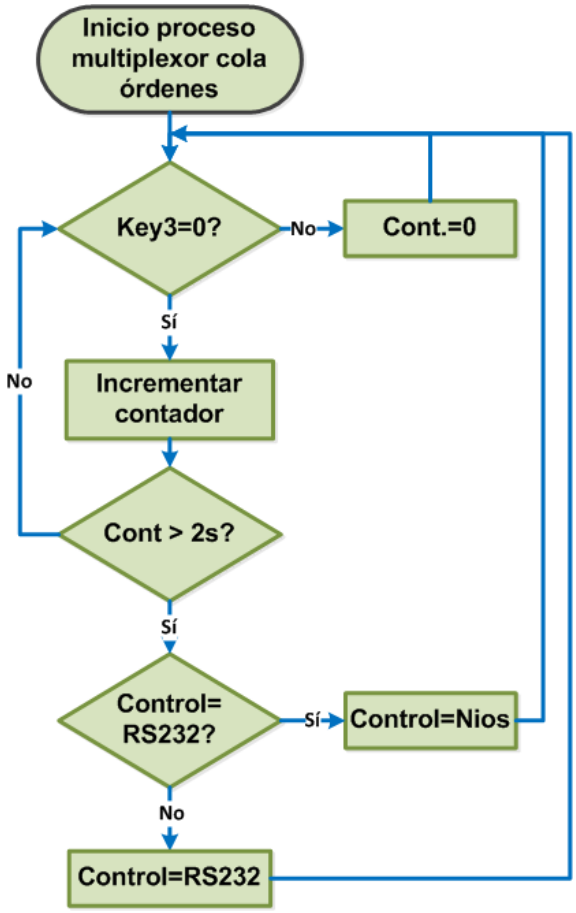

Figura 3.4: Multiplexor de cola de órdenes <Main>

- Monitor de cola de órdenes.

Informa al módulo que envía las órdenes al sistema (<Nios_system> o <Uart>) los momentos en los cuales la carga de la cola sobrepasa el 75 % y cuando se vuelve menor del 25 % (fig. 3.5), con el fin evitar que se sobreescriban órdenes que aún no se han cumplido en el módulo <Main>.

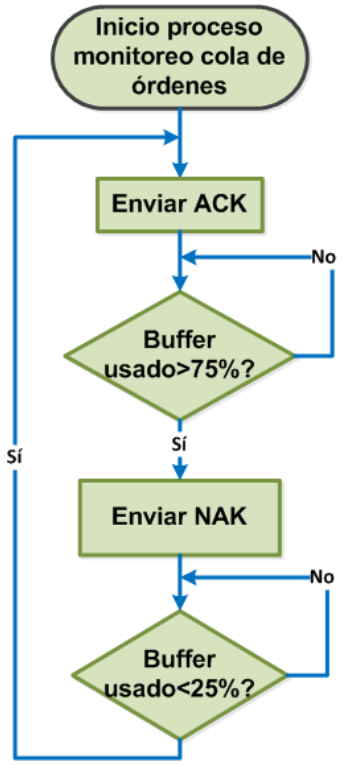

Figura 3.5: Monitor de cola de órdenes <Main>

- Escritura/lectura cola de órdenes.

  La cola de órdenes es de tipo FIFO y está basada en una memoria RAM (embebida en la FPGA) de doble puerto, uno de escritura y otro de lectura; cada una maneja su propio puntero de dirección: puntero de carga (upload) en el puerto de escritura y puntero de descarga (drop) en el puerto de lectura. Si el control

de la cola esta asignado a <Nios_system> y éste requiere transferir una orden, se genera una secuencia de arbitramento entre <Nios_system> y <Main> para escribir en la cola. Por cada byte almacenado se incrementa el puntero de upload. Por el lado del <Main> (drop), éste evalúa si el drop_pointer es diferente al upload_pointer, descargando una por una las órdenes. Por cada byte descargado se incrementa el drop_pointer. El proceso de descarga genera pausa cuando el drop_pointer se hace igual al upload_pointer (fig. 3.6). El proceso es idéntico cuando las órdenes provienen del módulo <Uart>.

- FSM principal.

  Representa el proceso central del módulo; decodifica las órdenes de la cola y a su vez ordena al módulo correspondiente la ejecución de la tarea. A medida que se descargan los bytes uno a uno de la cola de órdenes, la FSM valida que el comando cumpla con el protocolo establecido (apéndice B) el cual se diseñó bajo el criterio de uso de caracteres del código ASCII, de manera que, adicionalmente, pudieran transferirse órdenes a través de un programa de terminal de datos; si no lo cumple, lo descarta y genera un mensaje de error; si el comando es válido, genera una notificación de protocolo correcto y determina el recurso a asignar. Los comandos según su tipo pueden requerir el uso, bien sea del módulo <Printchar>, del módulo <Printmap> o de ambos, como en el caso de comandos de ventana, o en su defecto puede tratarse de un comando

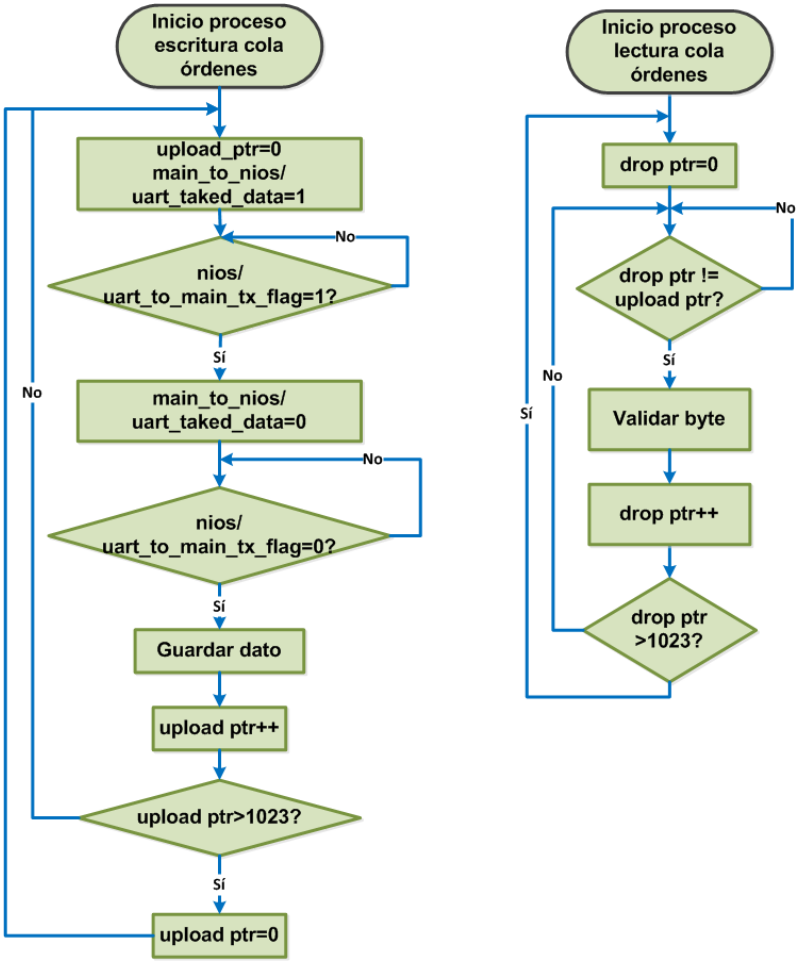

Figura 3.6: Escritura/lectura cola de órdenes <Main>

de configuración del entorno del ***IP core*** (ej. configuración de OSD o asignación de la memoria FLASH compartida). Cada vez que se requiere el uso del módulo <Printchar> o <Printmap>, la FSM verifica que éste se encuentre disponible, le asigna el recurso SDRAM por medio del multiplexor del mismo y le envía una orden para ejecución de la tarea (fig. 3.7).

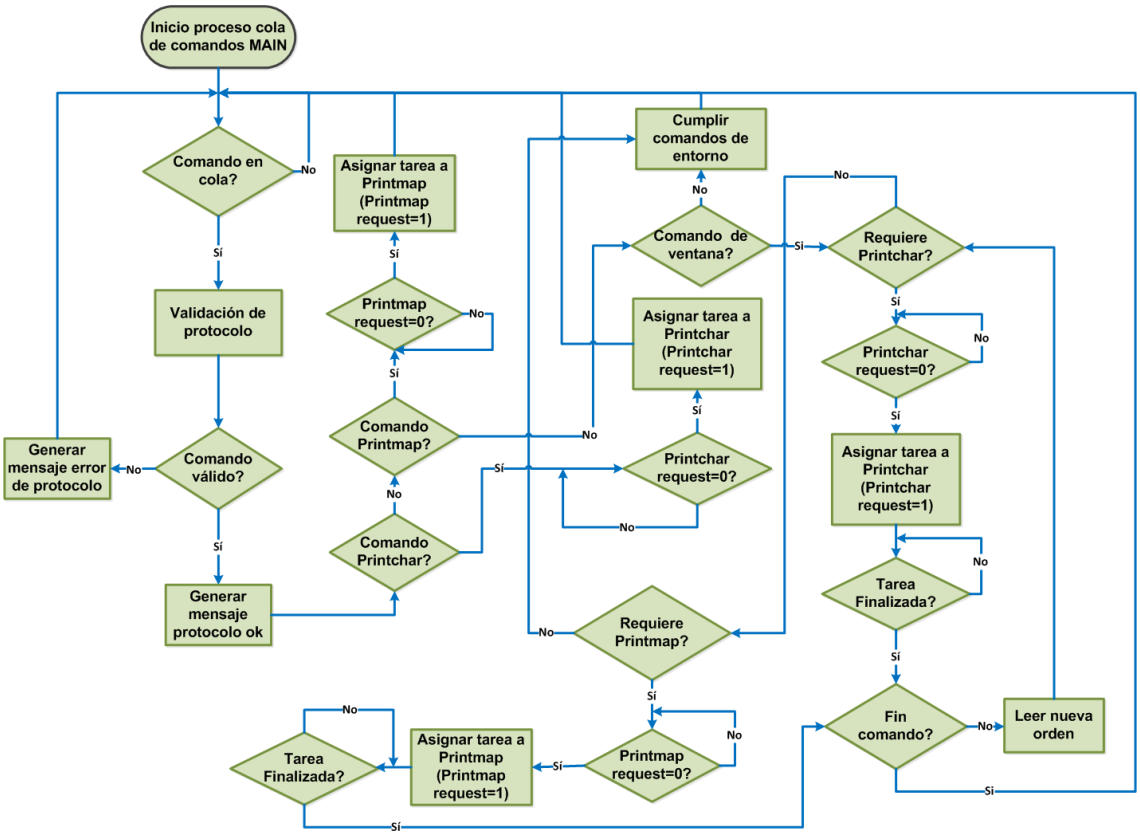

Figura 3.7: Manejo cola de órdenes <Main>

El módulo <Main> también controla la configuración de un dispositivo apuntador, de manera que a través del protocolo (apéndice B) puede seleccionarse mouse inactivo/activo, local o remoto. Para el modo local, la posición del puntero será la que salga del módulo <Mouse>; para modo remoto, la posición del puntero será la que envíe el módulo <Nios_system>. La posición del puntero es un dato a ser utilizado por el módulo <DE2_115_TV>, generando el dibujo del puntero acorde al barrido de la escritura de píxeles en la pantalla.

3.3.2. Módulo <Printchar>

Genera los despliegues de caracteres en pantalla (figs. 3.8 y 3.9) de tipo normal, con opciones de distintos tamaños y colores generados a partir de la fuente base de caracteres del sistema. Soporta también efectos de resaltado de caracteres y parpadeo. Está controlado por una FSM que mantiene monitoreo de la orden de ejecución de una tarea por parte del módulo <Main>, compartiendo con éste la señal de módulo listo *oReady*. Cuando <Main> debe ordenar a este módulo que imprima una letra, a través de las líneas de control indica cuál es el tipo de letra, número, posición y color de la misma. <Printchar> recibe de <Main> la orden de inicio a través de la señal *iStart*; la FSM indica a <Main> que recibió la orden desactivando su señal *oReady*, la cual vuelve a activar una vez termine su tarea, dando por finalizado el proceso. Los mapas de letras están cargados en dos memorias CGROM, la primera para los caracteres sin borde (organizada en 224 caracteres de 8x8 píxeles cada uno, 1 bit/píxel, 1,75 KB) y la segunda para caracteres con borde (organizada en 224 caracteres de 10x10 píxeles cada uno, 2 bits/píxel, 5,46 KB). La FSM extrae una fila de píxeles, donde cada píxel activo lo almacena en la memoria de salida con el color correspondiente (el cual viene dado por las señales de entrada *iColor* [14:0] e *iBackground_color* [14:0], tabla 3.2), y cada píxel inactivo en 0. Una vez la ráfaga de la fila esta cargada, la transfiere a la memoria SDRAM cuando su cola de datos tiene el espacio suficiente para recibir dicha ráfaga. Además, el módulo

<Printchar> ejecuta el rellenado (borrado) de pantalla.

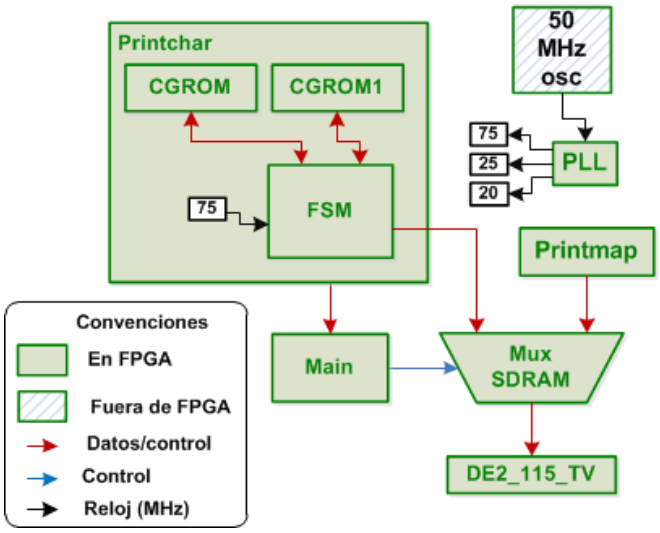

Figura 3.8: Diagrama de bloques <Printchar>

3.3.3. Módulo <Printmap>

Genera despliegues de mapas de bits, caracteres de tipo especial y figuras geométricas como líneas, cajas, vectores, círculos y arcos (figs. 3.10 y 3.11). Para la generación de figuras geométricas se ha implementado el algoritmo de Bresenham [14] (algoritmo que permite la generación de circunferencias y líneas, utilizando técnicas de distibución de píxeles en segmentos, basado totalmente en matemática de números enteros; de esta manera se evita el uso de funciones trigonométricas y matemática de punto flotante, reduciendo drásticamente el tiempo de procesamiento y la complejidad del algoritmo generador). Para las figuras angulares (vector y arco) fue necesario recurrir a las funciones seno y coseno con el fin de calcular las coordenadas (x,y) de inicio y fin de la figura (el resto del proceso se hace usando algoritmo de

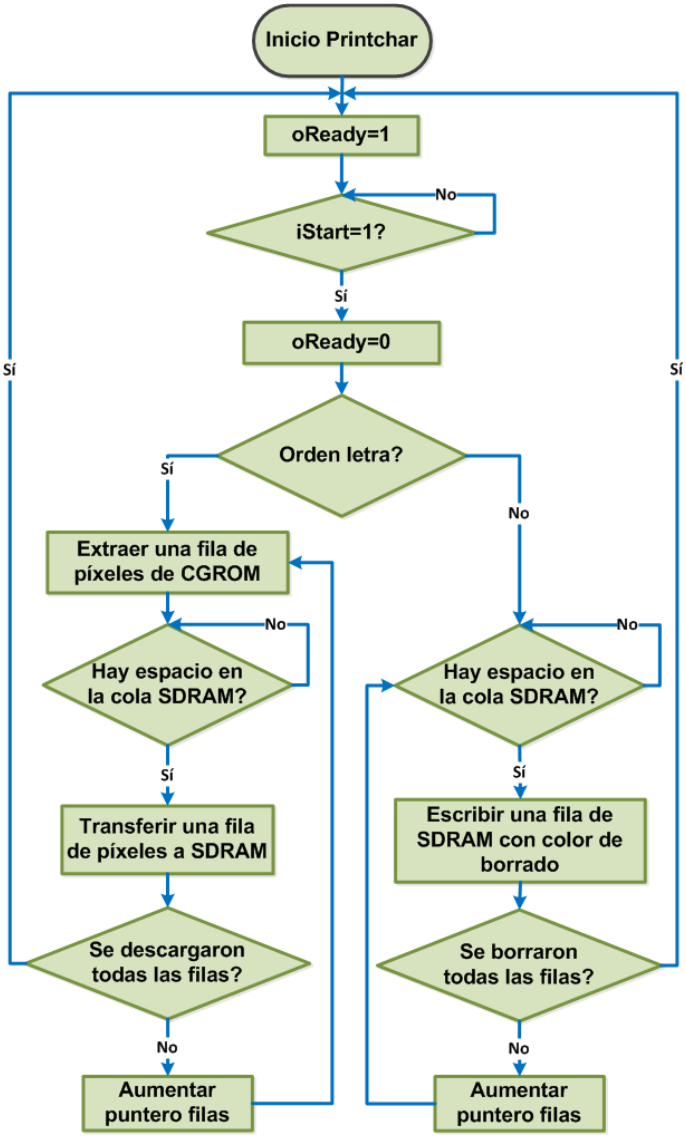

Figura 3.9: Diagrama de flujo <Printchar>

Bresenham [14]). Con el propósito de obtener el seno y el coseno de manera eficiente, se implementó una tabla de datos digitales (LUT) donde se tuvo en cuenta:

- cos(x) = sen(90 - x), por tanto, sólo es necesario tabular el seno.
- Amplificar valores del seno por 256 para trabajar con enteros de 8 bits en

lugar de punto flotante. De esta manera se representa el rango [0:1] a través del rango [0:255].

- Se reduce la tabla al considerar sólo los ángulos de 0 a 90 grados. Esto es válido porque en este rango la magnitud del seno asume todos los valores posibles. El seno de los ángulos fuera de este rango puede obtenerse mediante identidades trigonométricas sencillas utilizando ángulos dentro del rango.

Como ejemplo de la implementación seno y coseno, se requiere saber el coseno de 77°. De acuerdo con la identidad, cos (77)=sen(90-77)=sen(13)=0,2249. Acudiendo a la tabla implementada mediante LUT:

```
reg [7:0] SIN_BY_256_LUT_DATA;

  always begin

    case (angle_to_sin_by_256_lut)
      1: SIN_BY_256_LUT_DATA <= 8'd4;
      2: SIN_BY_256_LUT_DATA <= 8'd8;
      3: SIN_BY_256_LUT_DATA <= 8'd13;
      4: SIN_BY_256_LUT_DATA <= 8'd17;
      5: SIN_BY_256_LUT_DATA <= 8'd22;
      6: SIN_BY_256_LUT_DATA <= 8'd26;
      7: SIN_BY_256_LUT_DATA <= 8'd31;
      8: SIN_BY_256_LUT_DATA <= 8'd35;
      9: SIN_BY_256_LUT_DATA <= 8'd40;
      10: SIN_BY_256_LUT_DATA <= 8'd44;
      11: SIN_BY_256_LUT_DATA <= 8'd48;
      12: SIN_BY_256_LUT_DATA <= 8'd53;
      13: SIN_BY_256_LUT_DATA <= 8'd57;
```

Se obtiene así como resultado el número 57. Al dividir este valor por 256, se obtiene 0,2226, el cual es un valor muy cercano al correspondiente. De esta manera se tratan los valores de estas funciones como un valor entero amplificado 256 veces.

El módulo se compone internamente por una FSM principal y el módulo auxiliar <figures_by_angles> utilizado para generar las figuras angulares. La operación del módulo se inicia al recibir un nivel activo en la señal *iStart* desde el módulo <Main>. Luego de recibir esta señal, <Printmap> desactiva su señal *oReady* dando a entender que ha empezado la ejecución de la orden; luego, la FSM determina si la tarea solicitada es del tipo mapa de bits o figura geométrica. Si la tarea es desplegar un mapa de bits o un caracter de texto especial, lee la información de contexto del mapa en la memoria FLASH paralela, con la cual obtiene el puntero inicial de los píxeles, ancho y alto del mapa. En caso de que haya espacio para desplegar el mapa en la posición de pantalla solicitada, lee las filas del mapa de la memoria FLASH y las almacena en la memoria FIFO del controlador SDRAM a partir de la dirección que corresponde a la posición inicial del mapa en pantalla, hasta que termina con todos los píxeles. Si la tarea es dibujar una figura tipo línea, caja o círculo, las posiciones de los píxeles se generan una a una usando el algoritmo de Bresenham [14] y se depositan en la memoria FIFO del controlador SDRAM hasta completar la figura. En el caso de las figuras angulares vector y arco, las posiciones de los píxeles son generadas con la ayuda del submódulo <figures_by_angles> donde se

usa la tabla de 8 bits de la función seno y luego se almacenan los píxeles uno a uno en la memoria FIFO del controlador SDRAM. Al final de cada tarea, el módulo <Printmap> regresa su señal *oReady* al modo activo.

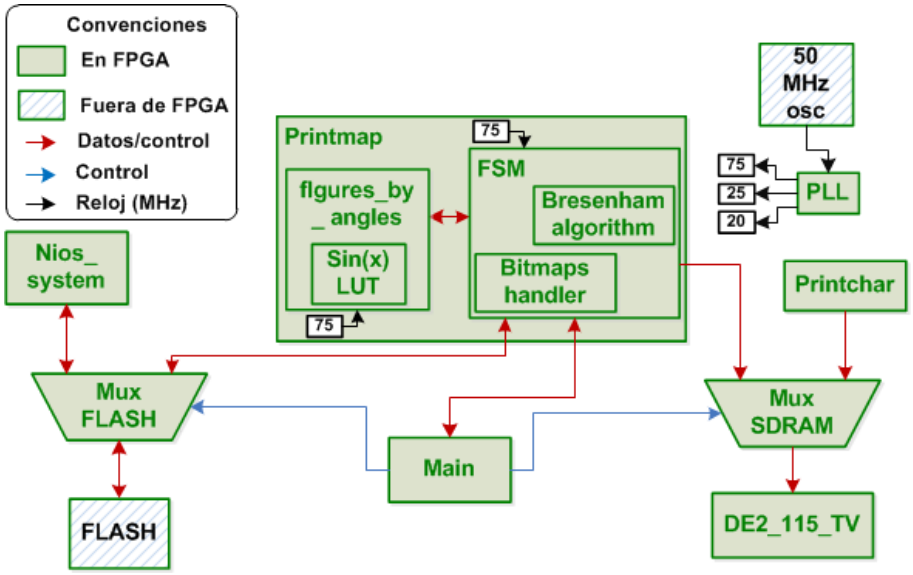

Figura 3.10: Diagrama de bloques <Printmap>

3.3.4. Módulo <DE2_115_TV>

Controla la memoria SDRAM (memoria externa a la FPGA de 128 MB la cual está distribuida en dos chips de 64 MB con bus de datos de 16 bits c/u) en la cual se encuentran almacenados los píxeles OSD. Además genera la imagen y combina los píxeles OSD con los de la señal DVI_IN. Para llevar a cabo estas funciones el módulo depende a su vez de tres submódulos principales (figs. 3.12 y 3.13):

- Módulo <SDRAM_controller>.
- Módulo <Comparador Mezclador>.

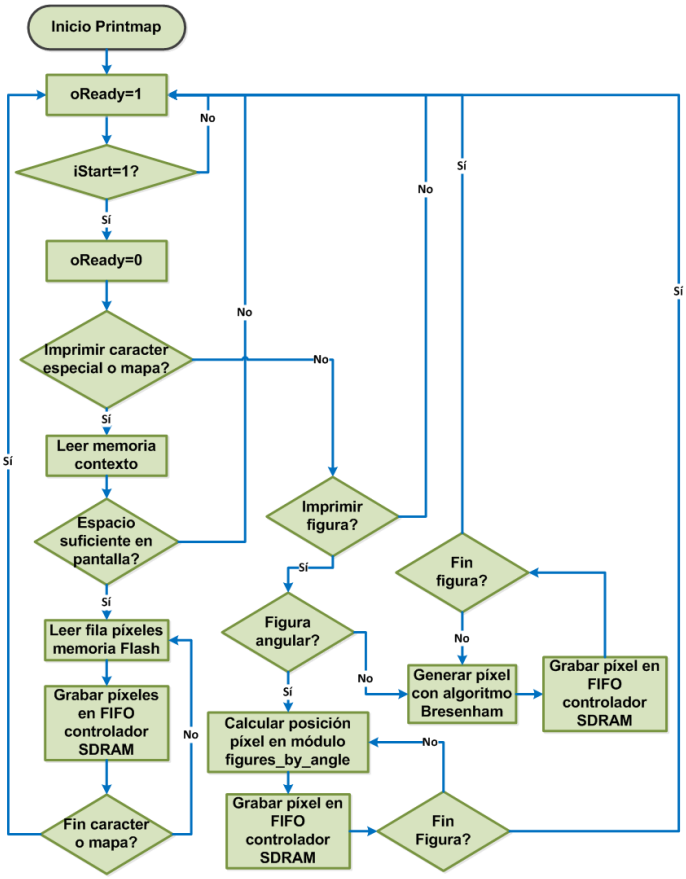

Figura 3.11: Diagrama de flujo <Printmap>

- Módulo <VGA_Controller>.

Estos módulos son explicados en detalle a continuación.

- Módulo <SDRAM_controller>.

  Gestiona la lectura y escritura de la memoria de video, para lo cual cuenta con dos memorias embebidas RAM tipo FIFO para los diferentes usuarios de la SDRAM. Los usuarios corresponden a los módulos <Printchar> y <Printmap> (los cuales descargan ráfagas de píxeles acorde a las zonas de

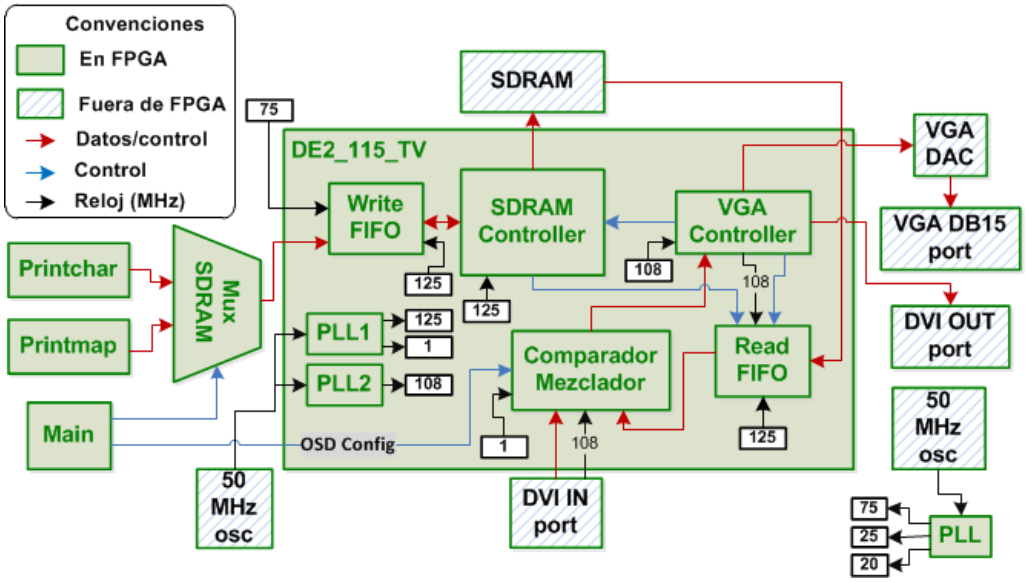

Figura 3.12: Diagrama de bloques <DE2_115_TV>

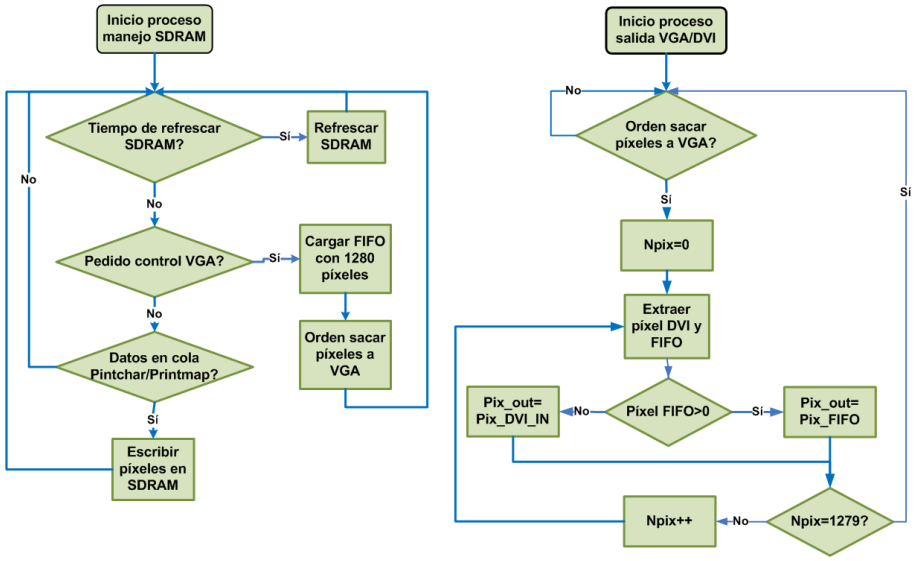

Figura 3.13: Diagrama de flujo <DE2_115_TV>

memoria de video que han de modificarse) y el módulo <VGA_Controller>, quien genera la imagen de salida del sistema haciendo pedidos a este módulo cada vez que necesita una nueva fila de píxeles. Cuando no hay pedidos

del <VGA_Controller>, <SDRAM_controller> gestiona la escritura de píxeles OSD, los cuales se acumulan en una cola FIFO de entrada. Al tratarse la SDRAM de una memoria dinámica, las posiciones de memoria deben refrescarse periódicamente para preservar la integridad de los datos almacenados. Para los chips de memoria utilizados en el proyecto (ISSI IS42S16320B [15]), el requerimiento es de al menos 8.192 refrescos por cada 64 ms. Para no trabajar al límite, se fijó el controlador para que ordenara a la memoria SDRAM 8.602 refrescos por cada 64 ms, equivalente a un refresco cada 7,44 us (el refresco es un comando que se envía a la memoria a través de la interfaz física de la misma).

- Módulo <Comparador Mezclador>.

  Recibe cada píxel procedente de la memoria FIFO de lectura y lo compara con el píxel que está ingresando desde la entrada DVI_IN, para determinar el píxel resultante que saldrá finalmente a la pantalla. Si el píxel de OSD es mayor que cero se despliega éste, de lo contrario se despliega el píxel de DVI_IN. Si la señal OSD_config indica que el despliegue OSD está inactivo, siempre saldrá el píxel de DVI_IN sin efectuar comparaciones.

- Módulo <VGA_controller>.

  Descarga los píxeles acumulados en la memoria FIFO de lectura (en los tiempos precisos del estándar SXGA [16]) y los recibe filtrados por el módulo

<Comparador Mezclador>, generando los tiempos y señales de sincronización necesarias para el despliegue de la imagen completa.

3.3.5. Módulo <Nios_system>

La fig. 3.14 ilustra el módulo, el cual, a través de la ejecución de un programa desarrollado en lenguaje C (no se hizo uso de sistemas operativos en tiempo real), se encarga de la comunicación Ethernet por medio de un stack TCP/IP software desarrollado para el procesador Nios y un controlador de acceso al medio TSE-MAC (Triple Speed Ethernet-Media Access Controller), el cual provee la interfaz con la capa física; este sub-sistema también es responsable de gestionar el inventario de mapas de bits, los cuales son actualizados a través de una tarjeta de memoria SD y almacenados para su uso en una memoria FLASH paralela externa a la FPGA. La comunicación Ethernet tiene dos propósitos: recibir los comandos enviados remotamente al ***IP core*** de gráficos HMI y las coordenadas (x,y) para el despliegue del puntero del dispositivo apuntador (mouse) en pantalla cuando se trabaja el esquema de mouse remoto. El diagrama en fig. 3.15 ilustra cómo es el proceso que gestiona la comunicación remota con el ***IP core*** de gráficos HMI. Para llevar a cabo esto, el sistema emplea el modo servidor TCP/IP y habilita el puerto 3.500 para aceptar conexiones. Una vez se establece la conexión, el procesador NiosII verifica si hay paquetes nuevos en la cola de recepciones. El manejo de dicha cola se hace a través de dos punteros: un puntero de carga llamado rx_upload_ptr y otro de descar-

ga llamado rx_drop_pointer. Al inicio del proceso ambos punteros son iguales, pero rx_upload_ptr incrementa su valor en 1 cada vez que recibe un paquete de datos.

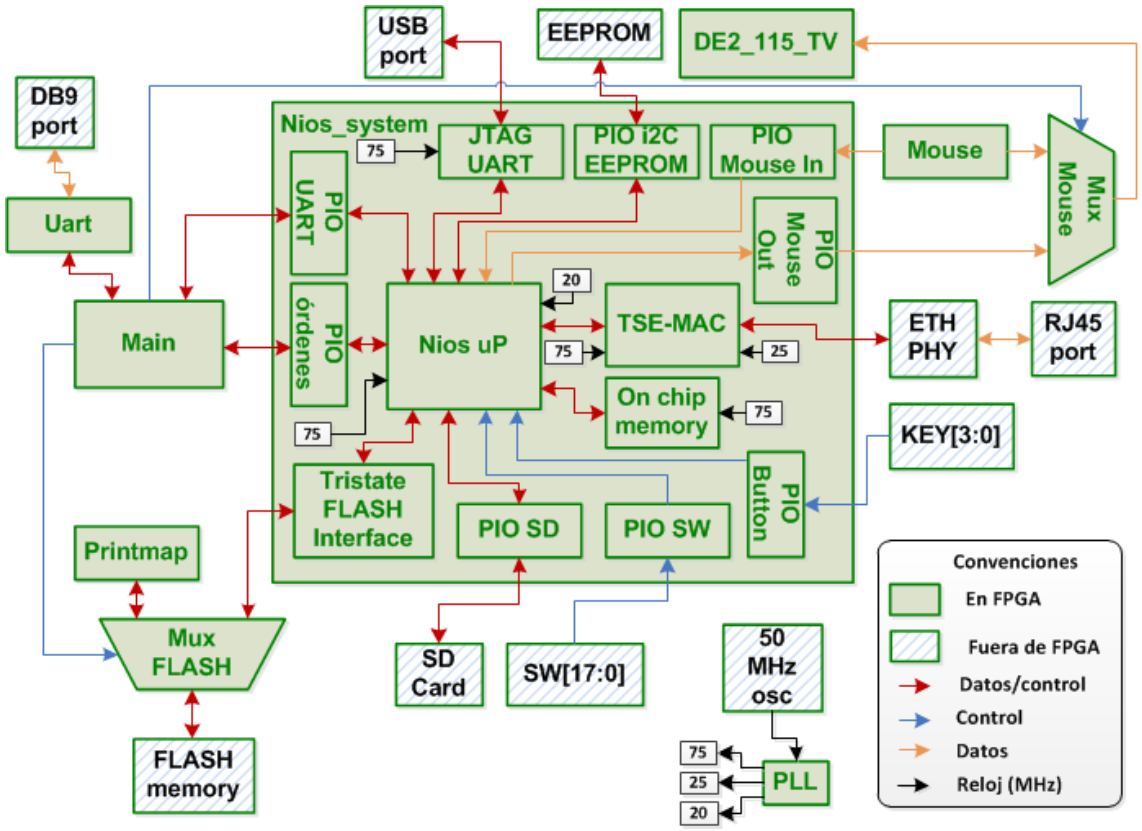

Figura 3.14: Diagrama de bloques SOPC <Nios_system>

El proceso que atiende el socket HMI detecta que hay paquetes en la cola de recepción al evaluar que rx_upload_ptr es diferente de rx_drop_ptr; los paquetes son descargados uno a uno de la cola al tiempo que rx_drop_ptr se incrementa hasta igualar el valor de rx_upload_ptr. A cada paquete se le efectúa la suma de comprobación (checksum); si el paquete pasa el chequeo, su contenido es transferido al ***IP core*** a través de la PIO que se conecta con la cola de órdenes; de lo contrario, el paquete

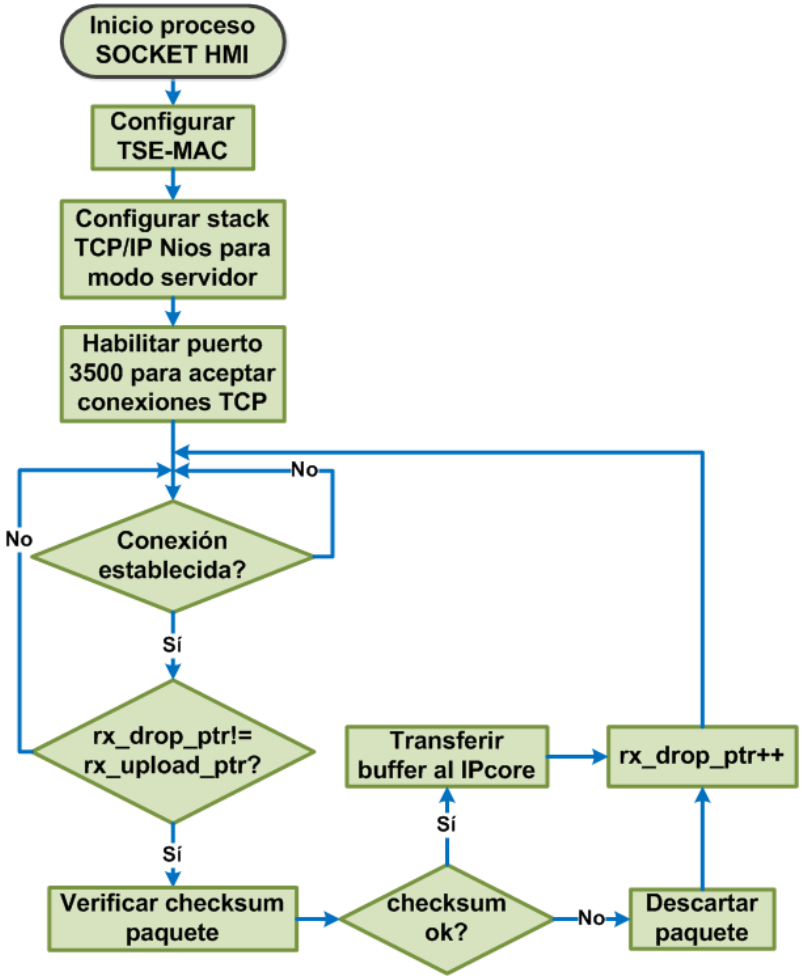

Figura 3.15: Proceso socket HMI <Nios_system>

es descartado. El socket de mouse remoto tiene un manejo similar al de HMI, pero no se realiza suma de verificación (ver fig. 3.16). Una vez se recibe la información de las coordenadas (x,y) del puntero, éstas se transfieren al módulo <DE2_115_TV> a través de la PIO de salida correspondiente.

El proceso de actualización de mapas de bits se describe en la fig. 3.17. Inmediatamente se enciende el sistema, una rutina de software en el procesador Nios

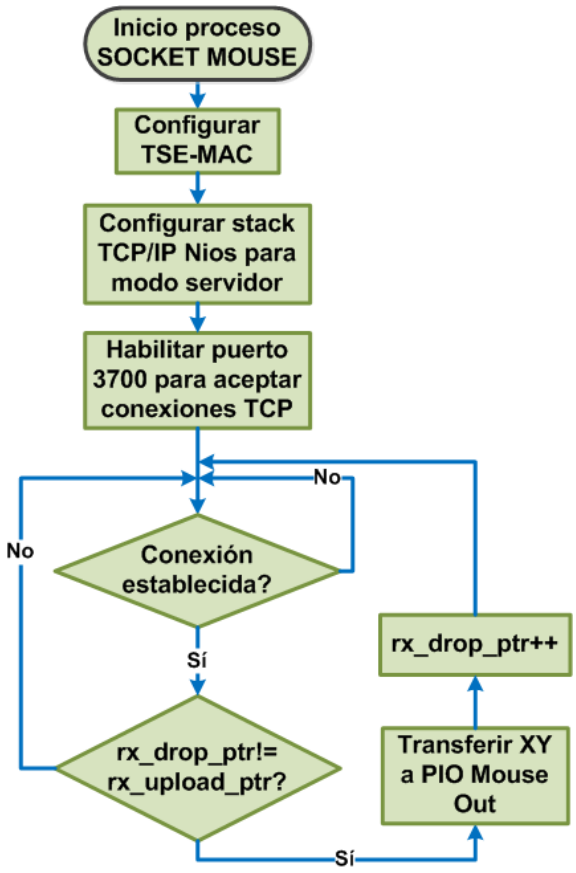

Figura 3.16: Proceso socket mouse <Nios_system>

verifica si la tarjeta SD ha sido insertada; en caso negativo, el proceso finaliza y no ejecutará actualización de mapas hasta que el sistema sea encendido nuevamente o sea reiniciado; por otro lado, si la memoria SD fue insertada y está formateada en FAT, se realiza una exploración de los archivos presentes en el directorio raíz. Para que un archivo sea tenido en cuenta como mapa de bits debe cumplir con el formato de nombre específico "mapa %u.bmp", donde %u es el número de mapa (map_id), el cual debe ser menor que el valor definido en la macro MAX_MAPS (1.000 para este caso). Si cumple con el formato de nombre, el mapa es marcado como candidato

para descarga. Como requisito de seguridad para el proceso, se determinó que para realizar un procedimiento de actualización de mapas, la memoria SD insertada debe contener el archivo "mapa1.bmp", por tanto, si este archivo no está presente, se da por terminada la actualización. Cumplidos los primeros pasos de validación, el procesador continúa haciendo un barrido entre todos los valores de map_id válidos, es decir 0≤map_id<MAX_MAPS, y en cada caso evalúa si fue marcado y si el mapa fue previamente grabado en la memoria externa FLASH, lo cual se verifica revisando el área de contexto correspondiente a dicho mapa. La memoria de contexto es una porción de la memoria FLASH (los primeros 32KB), que han sido reservados para contener información básica sobre cada mapa que podría ser inscrito. El contexto de cada mapa tiene un tamaño de 32 bytes, por lo que puede almacenarse en esta zona de memoria el contexto de hasta 1.024 mapas. La información de contexto especificada por cada mapa comprende: map_id (2 bytes), dirección de inicio (3 bytes), dirección de fin (3 bytes), fecha de modificación (7 bytes), ancho (2 bytes), alto (2 bytes) y 13 bytes disponibles para uso futuro (expansión del ***IP core***).

Dependiendo si el mapa fue marcado candidato y si ha sido previamente grabado, puede ocurrir uno de los siguientes casos:

- Mapa no presente en memoria SD (no marcado) pero ha sido previamente grabado en memoria FLASH. El contexto del mapa es borrado de la memoria FLASH, lo que equivale a borrar el mapa de la memoria, quedando no dispo-

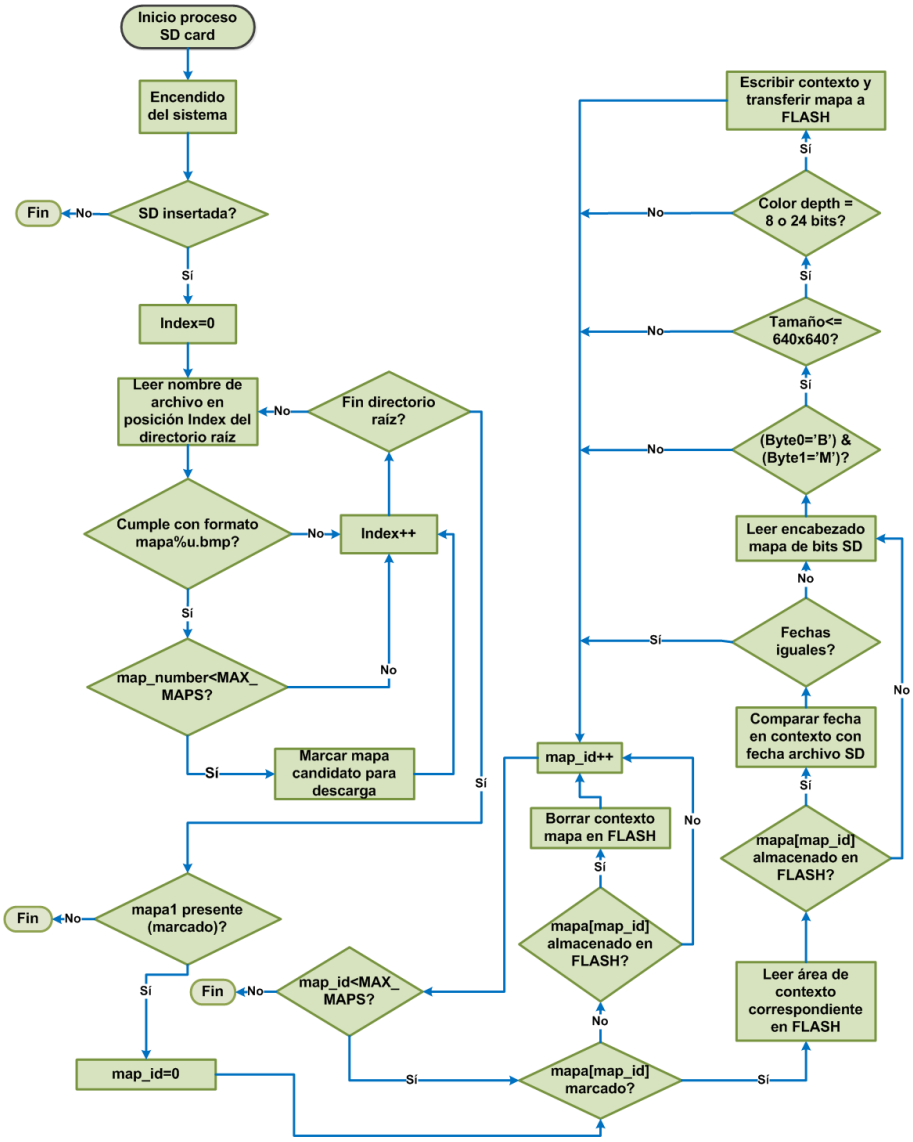

Figura 3.17: Proceso manejo memoria SD <Nios_system>

nible para uso y la zona de memoria ocupada podrá ser usada por otro mapa de igual o menor tamaño.

- Mapa marcado y no ha sido grabado previamente en memoria FLASH. Se graba el contexto del mapa en la posición correspondiente y seguidamente

se graba el contenido del mismo a partir de una zona disponible, la cual se determina haciendo una búsqueda entre los contextos de los mapas grabados previamente para encontrar qué mapa válido posee la dirección de fin más alta; entonces el nuevo mapa se graba a partir de la posición siguiente a este valor.

- Mapa marcado y previamente grabado en memoria FLASH. En este caso, primero se determina si la fecha de modificación de archivo almacenada en la zona de contexto de la memoria FLASH coincide con la fecha de modificación del archivo en la memoria SD; de ser así no se realiza ningún cambio; en caso contrario, se procede a verificar si el tamaño del nuevo mapa puede ser almacenado en la región que se le había asignado anteriormente. Si el mapa no cabe en la región previamente asignada, entonces se debe asignar una nueva región dentro de la memoria FLASH, lo cual se hace como se describió en el caso anterior.

Para implementar el módulo <Nios_system> se usó la herramienta Qsys [17] (parte del IDE Quartus II V12.1 de Altera) que permite generar SOPC; se incluyeron los siguientes recursos para su adecuado funcionamiento (fig. 3.14): memoria Onchip RAM, que conforma la memoria de programa y datos del procesador embebido; esta memoria está contenida dentro de la FPGA como recurso del chip; temporizadores (timers) y manejo de interrupciones, cuya misión es controlar procesos que deban

temporizarse al interior del procesador; controlador Ethernet MAC que conforma la interfaz para la LAN, y señales de datos y control para la memoria SD, que la conectan físicamente con el procesador, permitiendo que a través de ésta puedan actualizarse los mapas de bits que se almacenan en la memoria paralela FLASH.

3.3.6. Módulo <Uart>

Provee el medio para transferir las órdenes en forma serial desde un dispositivo externo RS232, con la posibilidad de configurar varias velocidades de transferencia en baudios. El usuario externo debe cumplir con el protocolo que se ha implementado para este proyecto (apéndice B) para lograr acceder a las funciones gráficas del ***IP core*** de manera adecuada.

3.3.7. Módulo <Mouse>

Gestiona el control de un dispositivo apuntador tipo PS/2; si está conectado, ejecuta la secuencia de inicialización del mismo y mantiene actualizado en sus salidas la posición (x,y) del puntero. A través del multiplexor de Mouse, <Main> decide qué coordenadas llegan al módulo generador de píxeles, bien sean procedentes del módulo <Mouse> local o del módulo <Nios_system> desde un dispositivo remoto. El protocolo de operación del ***IP core*** (apéndice B) describe el comando para definir el modo de operación del dispositivo apuntador.

3.4. Señales del sistema

3.4.1. Señales módulo <Main>

Nombre	Tipo	Descripción	Proced/destino
oPage_select	Salida	Indicador de la OSD a ser escrita	Printchar, Printmap
oGconfig [4:0]	Salida	Indicador de configuración de gráficos (prioridades OSDs y entrada de video on/off)	DE2_115_TV
iDvirx_on	Entrada	Indicador de señal de video de entrada estable	DE2_115_TV
oMouse_on	Salida	Indicador de mouse encendido/apagado	DE2_115_TV
oMouse_mux	Salida	Selector de mouse local/remoto	De2_115_ipcore
oSdram_selmux [1:0]	Salida	Selector de módulo de escritura de SDRAM	Sdram_fifo_mux
oBlock_to_fill [6:0]	Salida	Selector de área de pantalla a colorear	Printchar
oFill_memories_req	Salida	Indicador de requerimiento llenado de pantalla	Printchar
oColor_to_fill [14:0]	Salida	Indicador de color de llenado de pantalla	Printchar
oRow_printchar [9:0]	Salida	Indicador de fila de inicio de escritura de caracter	Printchar
oColumn_printchar [10:0]	Salida	Indicador de columna de inicio de escritura de caracter	Printchar
oLetter_to_print [7:0]	Salida	Indicador de código de letra a imprimir	Printchar
oLetter_size [4:0]	Salida	Tamaño de la letra a imprimir	Printchar
oLetter_type	Salida	Tipo de letra (normal o especial) a imprimir	Printchar
oInv	Salida	Indicador de modo de letra inverso	Printchar

Continúa en la página siguiente.

Nombre	Tipo	Descripción	Proced/destino
oColor [14:0]	Salida	Indicador de color de letra (normal o especial) a imprimir	Printchar, Printmap
oBackground_color [14:0]	Salida	Indicador de color de fondo de letra (normal o especial) a imprimir	Printchar, Printmap
oBlink	Salida	Indicador de letra o figura a imprimir intermitente/estática	Printchar, Printmap
oStart_printchar	Salida	Orden de iniciar impresión de caracter o llenado de pantalla	Printchar
iReady_printchar	Entrada	Indicador de proceso en ejecución	Printchar
oFill_nrows [9:0]	Salida	Indicador de número de filas de relleno	Printchar
oFill_ncolumns [10:0]	Salida	Indicador de número de columnas de relleno	Printchar
oFill_by_area_enable [10:0]	Salida	Indicador de requerimiento de relleno de área	Printchar
oFill_borders_sizes [7:0]	Salida	Indicador del tamaño de cada borde de relleno	Printchar
oFill_border_color [14:0]	Salida	Indicador del color de los bordes de relleno	Printchar
oRow_printmap [9:0]	Salida	Indicador de fila de inicio de escritura de mapa	Printmap
oColumn_printmap [10:0]	Salida	Indicador de columna de inicio de escritura de mapa	Printmap
oMap_letter_to_print [10:0]	Salida	Indicador de número de mapa o letra especial a imprimir	Printmap
oRead_context_en	Salida	Orden de leer el contexto de un mapa	Printmap
oLetter_border_enable	Salida	Orden de imprimir letra con/sin borde	Printmap
iMap_n_columns [9:0]	Entrada	Indicador del número de columnas que tiene un mapa o letra especial	Printmap
Continúa en la página siguiente.			

Nombre	Tipo	Descripción	Proced/destino
iMap_n_rows [9:0]	Entrada	Indicador del número de filas que tiene un mapa o letra especial	Printmap
oStart_printmap	Salida	Orden de iniciar impresión de mapa, caracter especial o figura	Printmap
iReady_printmap	Entrada	Indicador de proceso en ejecución	Printmap
iSucess_printmap	Entrada	Indicador de proceso exitoso	Printmap
oFigure_x1 [10:0]	Salida	Indicador de coordenada X de inicio de una figura	Printmap
oFigure_y1 [9:0]	Salida	Indicador de coordenada Y de inicio de una figura	Printmap
oFigure_x2 [10:0]	Salida	Indicador de coordenada X de término de una figura	Printmap
oFigure_y2 [9:0]	Salida	Indicador de coordenada Y de término de una figura	Printmap
oFigure_radius [9:0]	Salida	Indicador de radio de un arco o círculo	Printmap
oFigure_color [14:0]	Salida	Indicador de color de una figura	Printmap
oFigure_type [2:0]	Salida	Indicador de tipo de figura a generar (línea, círculo, caja, vector, arco)	Printmap
oFigure_angle0 [8:0]	Salida	Indicador de ángulo de inicio de un arco o ángulo de un vector	Printmap
oFigure_angle1 [8:0]	Salida	Indicador de ángulo de término de un arco	Printmap
oFlash_mux	Salida	Selector de usuario de memoria FLASH (Printmap o Nios_system)	De2_115_ipcore
oNios_tx_empty	Salida	Indicador de dato serial de Nios a Main libre	Nios_system
iNios_tx_data [7:0]	Entrada	Dato de comunicación serial Nios a Main	Nios_system
Continúa en la página siguiente.			

Nombre	Tipo	Descripción	Proced/destino
iNios_tx_flag	Entrada	Indicador de dato serial Nios a Main listo para ser leído	Nios_system
oNios_tx_ack	Salida	Indicador de cola de datos seriales Nios a Main está cargada en menos del 25 %	Nios_system
oNios_tx_nak	Salida	Indicador de cola de datos seriales Nios a Main está cargada en más del 75 %	Nios_system
oNios_rx_data [7:0]	Salida	Dato de comunicación serial Main a Nios	Nios_system
oNios_rx_flag	Salida	Indicador de dato serial Main a Nios listo para ser leído	Nios_system
iNios_rx_taked_data	Entrada	Indicador de dato serial de Main a Nios ha sido leído	Nios_system
oMain_to_nios_flag	Salida	Indicador de dato paralelo Main a Nios listo para ser leído	Nios_system
oMain_to_nios_data [7:0]	Salida	Dato de comunicación paralela Main a Nios	Nios_system
iNios_taked_data	Entrada	Indicador de dato paralelo Main a Nios ha sido leído	Nios_system
iNios_to_main_flag	Entrada	Indicador de dato paralelo Nios a Main listo para ser leído	Nios_system
iNios_to_main_data [7:0]	Entrada	Dato de comunicación paralela Nios a Main	Nios_system
oMain_taked_data	Salida	Indicador de dato paralelo Nios a Main ha sido leído	Nios_system
oDebug_counter [31:0]	Salida	Cantidad de ciclos de reloj que tomó la última función ordenada a Printchar o Printmap	Nios_system

Continúa en la página siguiente.

Nombre	Tipo	Descripción	Proced/destino
oDebug_toggle_flag	Salida	Indicador que una función de Printchar o Printmap ha sido terminada	Nios_system
iUart_tx_empty	Entrada	Indicador de dato de salida de Uart ha sido transmitido	Uart
oUart_tx_data [7:0]	Salida	Dato de salida a ser transmitido por la Uart	Uart
oUart_tx_flag	Salida	Orden de transmitir un dato por la Uart	Uart
iUart_tx_overrun	Entrada	Indicador que a la Uart se le han transferido más datos que los que tiene libre la cola	Uart
iUart_tx_ack	Entrada	Indicador de cola de datos de la Uart está cargada en menos del 25 %	Uart
iUart_tx_nak	Entrada	Indicador de cola de datos de la Uart está cargada en más del 75 %	Uart
iUart_rx_data [7:0]	Entrada	Dato de entrada de la Uart a Main	Uart
iUart_rx_flag	Entrada	Indicador que existe un dato de recepción de la Uart listo para ser leído	Uart
oUart_rx_taked_data	Salida	Indicador de dato procedente de la Uart ha sido leído	Uart
oRs232_or_eth_ctrl	Salida	Indicador a la Uart de la fuente de control del ***IP core***	Uart
iCLK	Entrada	Señal de reloj (75Mhz)	Pll
iRST_N	Entrada	Señal de reset del sistema	DE2_115_TV

Tabla 3.1: Señales ***IP core*** desde/hacia el módulo Main

3.4.2. Señales módulo <Printchar>

Nombre	Tipo	Descripción	Proced/destino
iPage_select	Entrada	Indicador de la OSD a ser escrita	Main
iBlock_to_fill [6:0]	Entrada	Indicador de área de pantalla a colorear	Main
iFill_memories_req	Entrada	Indicador de requerimiento llenado de pantalla	Main
iColor_to_fill [14:0]	Entrada	Indicador de color de llenado de pantalla	Main
iRow_printchar [9:0]	Entrada	Indicador de fila de inicio de escritura de caracter	Main
iColumn_printchar [10:0]	Entrada	Indicador de columna de inicio de escritura de caracter	Main
iLetter_to_print [7:0]	Entrada	Indicador de código de letra a imprimir	Main
iLetter_size [4:0]	Entrada	Tamaño de la letra a imprimir	Main
iLetter_type	Entrada	Tipo de letra a imprimir	Main
iInv	Entrada	Indicador de modo de letra inverso	Main
iColor [14:0]	Entrada	Indicador de color de letra a imprimir	Main
iBackground_color [14:0]	Entrada	Indicador de color de fondo de letra a imprimir	Main
iBlink	Entrada	Indicador de letra a imprimir intermitente/estática	Main
iStart_printchar	Entrada	Orden de iniciar impresión de caracter o llenado de pantalla	Main
oReady_printchar	Salida	Indicador de proceso en ejecución	Main
iFill_nrows [9:0]	Entrada	Indicador de número de filas de relleno	Main
iFill_ncolumns [10:0]	Entrada	Indicador de número de columnas de relleno	Main
iFill_by_area_enable	Entrada	Indicador de requerimiento de relleno de área	Main
Continúa en la página siguiente.			

Nombre	Tipo	Descripción	Proced/destino
iFill_borders_sizes [7:0]	Entrada	Indicador del tamaño de cada borde de relleno	Main
iFill_border_color [14:0]	Entrada	Indicador del color de los bordes de relleno	Main
iRow_printmap [9:0]	Entrada	Indicador de fila de inicio de relleno	Main
iColumn_printmap [10:0]	Entrada	Indicador de columna de inicio de relleno	Main
oData_to_write [17:0]	Salida	Dato a ser cargado en la memoria FIFO de escritura del controlador SDRAM	Sdram_fifo_mux
oWr_req	Salida	Habilitador de escritura a la memoria FIFO de escritura del controlador SDRAM	Sdram_fifo_mux
iWr_used [9:0]	Entrada	Indicador del número de posiciones utilizadas de la memoria FIFO de escritura del controlador SDRAM	DE2_115_TV
iCLK	Entrada	Señal de reloj (75Mhz)	Pll
iRST_N	Entrada	Señal de reset del sistema	DE2_115_TV

Tabla 3.2: Señales ***IP core*** desde/hacia el módulo Printchar

3.4.3. Señales módulo <Printmap>

Nombre	Tipo	Descripción	Proced/destino
iPage_select	Entrada	Indicador de la OSD a ser escrita	Main
iRow_printmap [9:0]	Entrada	Indicador de fila de inicio de mapa o figura	Main
iColumn_printmap [10:0]	Entrada	Indicador de columna de inicio de mapa o figura	Main
iMap_letter_to_print [10:0]	Entrada	Indicador de número de mapa o letra especial a imprimir	Main
iRead_context_en	Entrada	Orden de leer el contexto de un mapa	Main
Continúa en la página siguiente.			

Nombre	Tipo	Descripción	Proced/destino
iMap_n_columns [9:0]	Entrada	Indicador del número de columnas que tiene un mapa o letra especial	Main
iMap_n_rows [9:0]	Entrada	Indicador del número de filas que tiene un mapa o letra especial	Main
iStart_printmap	Entrada	Orden de iniciar impresión de mapa, caracter especial o figura	Main
oReady_printmap	Salida	Indicador de proceso en ejecución	Main
oSucess_printmap	Salida	Indicador de proceso exitoso	Main
iBlink	Entrada	Indicador de letra o figura a imprimir intermitente/estática	Main
iColor [14:0]	Entrada	Indicador de color de letra a imprimir	Main
iLetter_border_enable	Entrada	Orden de imprimir letra con/sin borde	Main
iBackground_color [14:0]	Entrada	Indicador de color de fondo de letra a imprimir	Main
iFigure_x1 [10:0]	Entrada	Indicador de coordenada X de inicio de una figura	Main
iFigure_y1 [9:0]	Entrada	Indicador de coordenada Y de inicio de una figura	Main
iFigure_x2 [10:0]	Entrada	Indicador de coordenada X de término de una figura	Main
iFigure_y2 [9:0]	Entrada	Indicador de coordenada Y de término de una figura	Main
iFigure_radius [9:0]	Entrada	Indicador de radio de un arco o círculo	Main
iFigure_color [14:0]	Entrada	Indicador de color de una figura	Main
iFigure_type [2:0]	Entrada	Indicador de tipo de figura a generar (línea, círculo, caja, vector, arco)	Main
Continúa en la página siguiente.			

Nombre	Tipo	Descripción	Proced/destino
iFigure_angle0 [8:0]	Entrada	Indicador de ángulo de inicio de un arco o ángulo de un vector	Main
iFigure_angle1 [8:0]	Entrada	Indicador de ángulo de término de un arco	Main
ioFlash_dq [7:0]	Entrada/ salida	Bus de datos de memoria FLASH	Mux FLASH
oFlash_addr [22:0]	Salida	Bus de dirección de memoria FLASH	Mux FLASH
oFlash_ce_n	Salida	Habilitador de chip de memoria FLASH	Mux FLASH
oFlash_we_n	Salida	Habilitador de escritura memoria FLASH	Mux FLASH
oFlash_oe_n	Salida	Habilitador de salida activa memoria FLASH	Mux FLASH
oData_to_write [17:0]	Salida	Dato a ser cargado en la memoria FIFO de escritura del controlador SDRAM	Sdram_fifo_mux
oWr_req	Salida	Habilitador de escritura a la memoria FIFO de escritura del controlador SDRAM	Sdram_fifo_mux
iWr_used [9:0]	Entrada	Indicador del número de posiciones utilizadas de la memoria FIFO de escritura del controlador SDRAM	DE2_115_TV
iCLK	Entrada	Señal de reloj (75Mhz)	Pll
iRST_N	Entrada	Señal de reset del sistema	DE2_115_TV

Tabla 3.3: Señales ***IP core*** desde/hacia el módulo Printmap

3.4.4. Señales módulo <DE2_115_TV>

Nombre	Tipo	Descripción	Proced/destino
iClock_50	Entrada	Señal de reloj desde el oscilador local (50Mhz)	Osc. local
oVga_R [7:0]	Salida	Color rojo VGA	VGA DAC
oVga_G [7:0]	Salida	Color verde VGA	VGA DAC
oVga_B [7:0]	Salida	Color azul VGA	VGA DAC
Continúa en la página siguiente.			

Nombre	Tipo	Descripción	Proced/destino
oVga_blank_n	Salida	Señal de blanqueo VGA	VGA DAC
oVga_clk	Salida	Señal de reloj VGA	VGA DAC
oVga_hs	Salida	Pulso sincronía horizontal VGA	VGA DB15
oVga_vs	Salida	Pulso sincronía vertical VGA	VGA DB15
iDvi_rx_clk	Entrada	Señal de reloj de video de entrada (108MHz)	HSMC-DVI
iDvi_rx_d [23:0]	Entrada	Bus de datos de video de entrada	HSMC-DVI
iDvi_rx_de	Entrada	Habilitador de píxeles de video de entrada	HSMC-DVI
iDvi_rx_hs	Entrada	Pulso de sincronización horizontal de video de entrada	HSMC-DVI
iDvi_rx_vs	Entrada	Pulso de sincronización vertical de video de entrada	HSMC-DVI
oDvi_tx_clk	Salida	Señal de reloj de video de salida (108MHz)	HSMC-DVI
oDvi_tx_d [23:0]	Salida	Bus de datos de video de salida	HSMC-DVI
oDvi_tx_de	Salida	Habilitador de píxeles de video de salida	HSMC-DVI
oDvi_tx_hs	Salida	Pulso de sincronización horizontal de video de salida	HSMC-DVI
oDvi_tx_vs	Salida	Pulso de sincronización vertical de video de salida	HSMC-DVI
oDram_addr [12:0]	Salida	Bus de direcciones SDRAM	Memoria SDRAM
ioDram_dq [31:0]	Entrada/ salida	Bus de datos SDRAM	Memoria SDRAM
oDram_ba [1:0]	Salida	Selector de banco SDRAM	Memoria SDRAM
oDram_cke	Salida	Habilitador de reloj SDRAM	Memoria SDRAM
oDram_clk	Salida	Señal de reloj SDRAM (108MHz)	Memoria SDRAM
oDram_cs_n	Salida	Selector de chip SDRAM	Memoria SDRAM

Continúa en la página siguiente.

Nombre	Tipo	Descripción	Proced/destino
oDram_dqm [3:0]	Salida	Habilitador/inhibidor de escritura por bytes individuales SDRAM	Memoria SDRAM
oDram_cas_n	Salida	Señal de direccionamiento en columna SDRAM	Memoria SDRAM
oDram_ras_n	Salida	Señal de direccionamiento en fila SDRAM	Memoria SDRAM
oDram_we_n	Salida	Habilitador de escritura SDRAM	Memoria SDRAM
oRST_N	Salida	Señal para reinicio de los diferentes módulos del sistema	Todos los módulos
iCmd_fifo_data [17:0]	Entrada	Bus de datos de cola de comandos a la memoria SDRAM	Sdram_fifo_mux
iCmd_fifo_wrreq	Entrada	Indicador de requerimiento de escritura de comandos a la memoria SDRAM	Sdram_fifo_mux
oCmd_fifo_wrused [9:0]	Salida	Indicador del número de posiciones utilizadas de la memoria FIFO de comandos del controlador SDRAM	Sdram_fifo_mux
iCmd_fifo_wrclk	Entrada	Señal de reloj de memoria FIFO de comandos del controlador SDRAM	Sdram_fifo_mux
oFifo_rd_data [15:0]	Salida	Bus de datos de cola de lectura a la memoria SDRAM	Sdram_fifo_mux
iFifo_rd_rdreq	Entrada	Indicador de requerimiento de lectura a la memoria SDRAM	Sdram_fifo_mux
oFifo_rd_rdused [9:0]	Salida	Indicador del número de posiciones utilizadas de la memoria FIFO de lectura de memoria SDRAM	Sdram_fifo_mux
iFifo_rd_rdclk	Entrada	Señal de reloj de memoria FIFO de lectura de memoria SDRAM	Sdram_fifo_mux
Continúa en la página siguiente.			

Nombre	Tipo	Descripción	Proced/destino
iGconfig [4:0]	Entrada	Indicador de configuración de gráficos (prioridades OSDs y entrada de video on/off)	Main
oDvirx_on	Salida	Indicador de señal de video de entrada estable	Main
iMouse_x [10:0]	Entrada	Posición X del puntero del mouse	Mouse MUX
iMouse_y [9:0]	Entrada	Posición Y del puntero del mouse	Mouse MUX
iMouse_on	Entrada	Indicador de mouse encendido/apagado	Main

Tabla 3.4: Señales ***IP core*** desde/hacia el módulo DE2_115_TV

3.4.5. Señales módulo <Nios_system>

Nombre	Tipo	Descripción	Proced/destino
iCLK	Entrada	Señal de reloj maestro (75Mhz)	Pll
iRST_N	Entrada	Señal de reset del sistema	DE2_115_TV
iTse_mac_rgmii_in [3:0]	Entrada	Datos de recepción ETH PHY	ETH PHY
oTse_mac_rgmii_out [3:0]	Salida	Datos de transmisión ETH PHY	ETH PHY
iTse_mac_rx_control	Entrada	Control de datos de recepción ETH PHY	ETH PHY
oTse_mac_tx_control	Salida	Control de datos de transmisión ETH PHY	ETH PHY
iTse_mac_tx_clk	Entrada	Reloj de transmisión ETH PHY	ETH PHY
iTse_mac_rx_clk	Entrada	Reloj de recepción ETH PHY	ETH PHY
oTse_mac_ena_10	Salida	Selector de modo 10 Mbps ETH PHY	ETH PHY
oTse_mac_eth_mode	Salida	Habilitador de modo 10/100 Mbps ETH PHY	ETH PHY
Continúa en la página siguiente.			

Nombre	Tipo	Descripción	Proced/destino
ioFlash_dq [7:0]	Entrada/ salida	Bus de datos de memoria FLASH	Mux FLASH
oFlash_addr [22:0]	Salida	Bus de dirección de memoria FLASH	Mux FLASH
oFlash_ce_n	Salida	Habilitador de chip de memoria FLASH	Mux FLASH
oFlash_we_n	Salida	Habilitador de escritura memoria FLASH	Mux FLASH
oFlash_oe_n	Salida	Habilitador de salida activa memoria FLASH	Mux FLASH
iClk_20	Entrada	Señal de reloj módulo SD (20Mhz)	Pll
oSd_clk	Salida	Señal de reloj memoria SD	Memoria SD
iSd_wp_n	Entrada	Señal de protección a la escritura memoria SD	Memoria SD
ioSd_cmd	Entrada/ salida	Señal de comando memoria SD	Memoria SD
ioSd_dat [3:0]	Entrada/ Salida	Señal de datos memoria SD	Memoria SD
iNios_tx_empty	Entrada	Indicador de dato serial de Nios a Main libre	Main
oNios_tx_data [7:0]	Salida	Dato de comunicación serial Nios a Main	Main
oNios_tx_flag	Salida	Indicador de dato serial Nios a Main listo para ser leído	Main
iNios_tx_ack	Entrada	Indicador de cola de datos seriales Nios a Main está cargada en menos del 25 %	Main
iNios_tx_nak	Entrada	Indicador de cola de datos seriales Nios a Main está cargada en más del 75 %	Main
iNios_rx_data [7:0]	Entrada	Dato de comunicación serial Main a Nios	Main
Continúa en la página siguiente.			

Nombre	Tipo	Descripción	Proced/destino
iNios_rx_flag	Entrada	Indicador de dato serial Main a Nios listo para ser leído	Main
oNios_rx_taked_data	Salida	Indicador de dato serial de Main a Nios ha sido leído	Main
iMain_to_nios_flag	Entrada	Indicador de dato paralelo Main a Nios listo para ser leído	Main
iMain_to_nios_data [7:0]	Entrada	Dato de comunicación paralela Main a Nios	Main
oNios_taked_data	Salida	Indicador de dato paralelo Main a Nios ha sido leído	Main
oNios_to_main_flag	Salida	Indicador de dato paralelo Nios a Main listo para ser leído	Main
oNios_to_main_data [7:0]	Salida	Dato de comunicación paralela Nios a Main	Main
iMain_taked_data	Entrada	Indicador de dato paralelo Nios a Main ha sido leído	Main
iDebug_counter [31:0]	Entrada	Cantidad de ciclos de reloj que tomó la última función ordenada a Printchar o Printmap	Main
iDebug_toggle_flag	Entrada	Indicador que una función de Printchar o Printmap ha sido terminada	Main
oSel_uart_bps [2:0]	Salida	Selector de tasa de transferencia en Uart	Uart
iFlash_mux	Entrada	Selector de usuario de memoria FLASH (Printmap o Nios_system)	Main
iMouse_x [10:0]	Entrada	Posición X del puntero del mouse local	Mouse
iMouse_y [9:0]	Entrada	Posición Y del puntero del mouse local	Mouse
iMouse_buttons [2:0]	Entrada	Estado de los 3 botones del mouse local	Mouse

Continúa en la página siguiente.

Nombre	Tipo	Descripción	Proced/destino
oMouse_x [10:0]	Salida	Posición X del puntero del mouse remoto	Mouse MUX
oMouse_y [9:0]	Salida	Posición Y del puntero del mouse remoto	Mouse MUX

Tabla 3.5: Señales ***IP core*** desde/hacia el módulo Nios_system

3.4.6. Señales módulo <Uart>

Nombre	Tipo	Descripción	Proced/destino
oTx_out	Salida	Salida serial al MAX232	MAX232
oUart_tx_empty	Salida	Indicador de dato de salida de Uart ha sido transmitido	Main
iUart_tx_data [7:0]	Entrada	Dato a ser transmitido por la Uart	Main
iUart_tx_flag	Entrada	Orden de transmitir un dato por la Uart	Main
oUart_tx_overrun	Salida	Indicador que a la Uart se le han transferido más datos que los que tiene libre la cola	Main
oUart_tx_ack	Salida	Indicador de cola de datos de la Uart está cargada en menos del 25 %	Main
oUart_tx_nak	Salida	Indicador de cola de datos de la Uart está cargada en más del 75 %	Main
iRx_in	Entrada	Entrada serial desde el MAX232	MAX232
oUart_rx_data [7:0]	Salida	Dato recibido por la Uart hacia Main	Main
oUart_rx_flag	Salida	Indicador que existe un dato de recepción de la Uart listo para ser leído	Main
iUart_rx_taked_data	Entrada	Indicador de dato procedente de la Uart ha sido leído	Main
iRx_buffer_flush	Entrada	Indicador a la Uart que debe limpiar datos en cola	Main

Continúa en la página siguiente.

Nombre	Tipo	Descripción	Proced/destino
iRs232_or_eth_ctrl	Entrada	Indicador a la Uart de la fuente de control del ***IP core***	Main
iSel_uart_bps [2:0]	Entrada	Selector de tasa de transferencia en Uart	Nios_system
iCLK	Entrada	Señal de reloj (75Mhz)	Pll
iRST_N	Entrada	Señal de reset del sistema	DE2_115_TV

Tabla 3.6: Señales ***IP core*** desde/hacia el módulo Uart

3.4.7. Señales módulo <Mouse>

Nombre	Tipo	Descripción	Proced/destino
oMouse_x [10:0]	Salida	Posición X del puntero del mouse local	Mouse MUX
oMouse_y [9:0]	Salida	Posición Y del puntero del mouse local	Mouse MUX
oMouse_buttons [2:0]	Salida	Estado de los 3 botones del mouse local	Nios_system
ioPs2_dat	Entrada/ salida	Línea de datos PS/2	Puerto PS/2
ioPs2_clk	Entrada/ salida	Línea de reloj PS/2	Puerto PS/2
iCLK	Entrada	Señal de reloj (20Mhz)	Pll
iRST_N	Entrada	Señal de reset del sistema	DE2_115_TV

Tabla 3.7: Señales ***IP core*** desde/hacia el módulo Mouse

3.5. Verificación de dominios de reloj y recursos

Dentro del proceso de diseño se tuvieron en cuenta las frecuencias de los diferentes dominios de reloj que el sistema requiere; podemos ver en la tabla 3.8 que todos los dominios del reloj tienen Fmax suficientemente por encima de su requerimiento, a excepción del reloj pll[0] de 75MHz, lo cual ocurre debido a la existencia de variadas operaciones que requieren un ciclo de reloj en los módulos que utilizan este reloj, y que de acuerdo a los tiempos de estabilización de datos existentes entre los buses de entrada/salida, generan rutas críticas. Es de anotar que los relojes DVI_RX_CLK y pll2[1] de 108MHz también están cerca en Fmax de su requerimiento, sin embargo, al ser frecuencias impuestas por el estándar SXGA [16], no se requiere obtener mejoras sobre las mismas para trabajos futuros. Refiérase a la figura 3.2 y a los diagramas de bloques en las sub-secciones 3.3.1 hasta 3.3.7 para identificar la distribución de los dominios de reloj.

Como técnica para evitar meta-estabilidades en el sistema (estados no estables en la salida de los registros) cuando se requiere conectar señales cuyos dominios de reloj son diferentes, se incorporaron dos registros (flip-flops) entre dichas señales, los cuales funcionan bajo el dominio de reloj de la señal de destino. De igual forma, para hacer uso de memorias RAM embebidas que son escritas por un proceso que funciona bajo diferente dominio de reloj que el proceso que las lee, se utilizaron memorias de doble puerto, cada uno bajo el dominio de reloj del bus de datos que

le corresponde, evitando así las mencionadas meta-estabilidades.

Clock	**Uso**	**Frecuencia**	**Fmax**	**Fmax/Frec**
DVI_RX_CLK	<DE2_115_TV>	108,0 MHz	113,4 MHz	1,05
pll[0]	<Main>, <Printchar>, <Printmap>, <DE2_115_TV>, <Nios_system>, <Uart>	75,0 MHz	86,5 MHz	1,15
pll[2]	<Nios_system>	25,0 MHz	43,7 MHz	1,74
pll[4]	<Mouse>, <Nios_system>	20,9 MHz	124,3 MHz	5,94
pll1[1]	<DE2_115_TV>	125,0 MHz	188,0 MHz	1,50
pll1[2]	<DE2_115_TV>	125,0 MHz	150,6 MHz	1,20
pll1[3]	<Comparador Mezclador>	1,0 MHz	99,6 MHz	99,60
pll2[1]	<DE2_115_TV>	108,0 MHz	113,3 MHz	1,04

Tabla 3.8: Relojes y Fmax sistema completo

- Módulo <DE2_115_TV> (tabla 3.9):

  - 108 MHz. Reloj para la generación de los tiempos de sincronización de video; representa una imposición del estándar de video SXGA [16].

  - 125 MHz. Reloj de operación de la memoria SDRAM, cuya frecuencia se seleccionó con el fin de mantener una mayor velocidad en las transferencias de ráfagas hacia las memorias FIFO que en las lecturas de las mismas (108 MHz para generación de video). Para seleccionar 125 MHz se tuvo en cuenta que no se superara la máxima frecuencia (143 MHz)

Clock	Uso	Frecuencia	Fmax	Fmax/Frec
DVI_RX_CLK	<Comparador Mezclador>	108,0 MHz	113,4 MHz	1,05
pll1[1]	<SDRAM_controller>	125,0 MHz	173,7 MHz	1,38
pll1[2]	Read FIFO, Write FIFO	125,0 MHz	151,0 MHz	1,20
pll1[3]	<Comparador Mezclador>	1,0 MHz	143,5 MHz	143,50
pll2[1]	<VGA_controller>	108,0 MHz	118,1 MHz	1,09
pll2[3]	<VGA_controller>	108,0 MHz	152,7 MHz	1,41

Tabla 3.9: Relojes y Fmax módulo <DE2_115_TV>

de reloj especificada para el modelo de memoria instalada en la tarjeta (IS42S16320B-7TL [15]).

- Módulos <Main>, <Printchar>, <Printmap> y <Uart> (tablas 3.10, 3.11, 3.12 y 3.13). Para estos módulos no hay imposiciones en la frecuencia a ser utilizada, ya que las funciones que deben cumplir no se relacionan a la extracción en tiempos precisos de píxeles para la salida de video, sino a la modificación de píxeles del espacio de memoria OSD, que pueden cargarse en la medida de la disponibilidad que tenga el controlador SDRAM para descargar la memoria FIFO asociada a los módulos <Printchar> y <Printmap>. Se seleccionó como frecuencia 75 MHz, que ajusta perfectamente a las restricciones del sistema completo (Fmax de operación), correspondiente a 86,59 MHz (tabla 3.8).

- Módulo <Mouse> (tabla 3.14). Para este módulo se seleccionó 20 MHz teniendo en cuenta que debe manejar un dispositivo externo de baja frecuencia

Clock	Frecuencia	Fmax	Fmax/Frec
pll[0]	75,0 MHz	92,9 MHz	1,23

Tabla 3.10: Relojes y Fmax módulo <Main>

Clock	Frecuencia	Fmax	Fmax/Frec
pll[0]	75,0 MHz	155,1 MHz	2,06

Tabla 3.11: Relojes y Fmax módulo <Printchar>

de operación (mouse o dispositivo apuntador).

- Módulo <Nios_system> (tabla 3.15):

  - 75 MHz. Reloj principal del procesador. Se determinó usar la misma frecuencia del módulo <Main>, con el fin de evitar metaestabilidades, puesto que el procesador debe dar órdenes a dicho módulo.
  - 25 MHz. Reloj para operación de módulo Ethernet del sistema, en el modo 100 Mbps.
  - 20 MHz. Reloj para el módulo de manejo de la memoria SD.

Durante la programación de los módulos, se seleccionaron los principales (primera jerarquía, tabla 3.16) y se compilaron individualmente con el fin de establecer las

Clock	Frecuencia	Fmax	Fmax/Frec
pll[0]	75,0 MHz	93,4 MHz	1,24

Tabla 3.12: Relojes y Fmax módulo <Printmap>

Clock	Frecuencia	Fmax	Fmax/Frec
pll[0]	75,0 MHz	212,7 MHz	2,83

Tabla 3.13: Relojes y Fmax módulo <Uart>

Clock	Frecuencia	Fmax	Fmax/Frec
pll[4]	20,9 MHz	231,1 MHz	11,05

Tabla 3.14: Relojes y Fmax módulo <Mouse>

restricciones en sus dominios de reloj, para finalizar compilando el proyecto completo. Puede observarse que el módulo que utiliza la mayor cantidad de recursos es <Nios_system>, teniendo en cuenta que en su diseño se utilizaron varios ***IP cores*** del entorno de la herramienta Qsys [17], entre los cuales cabe destacar la TSE que utiliza 6.025 elementos lógicos (17,38 % del diseño), seguido del procesador NiosII que utiliza 2.594 elementos lógicos (7,48 % del diseño) de la FPGA para ser implementado. De igual manera, puede observarse que entre los módulos desarrollados a través de Verilog [5], el módulo <Main> requiere la mayor cantidad de recursos, debido a su condición de módulo principal del sistema, al tener que manejar varios procesos concurrentes de control del sistema como un todo, seguido del módulo

Clock	Uso	Frecuencia	Fmax	Fmax/Frec
pll[0]	Nios processor	75,0 MHz	92,9 MHz	1,23
pll[2]	TSE_MAC_RX	25,0 MHz	163,7 MHz	6,54
	TSE_MAC_TX	25,0 MHz	56,0 MHz	2,24
pll[4]	SD	20,9 MHz	121,2 MHz	5,79

Tabla 3.15: Relojes y Fmax módulo <Nios_system>

<Printmap> por tener la mayor carga en la creación de mapas y figuras.

Teniendo en cuenta la capacidad en elementos lógicos (114.480) de la FPGA Altera Cyclone IV EP4CE115 de la tarjeta , el sistema utiliza solamente el 30,27 % de los mismos, lo cual deja abierta la opción de aumentar las capacidades del sistema al tener libre el 69,73 % de la capacidad de la FPGA.

Módulo	**Elementos lógicos**	**Porcentaje**
Uart	216	0,62 %
Mouse	308	0,88 %
Otros módulos	537	1,54 %
Printchar	932	2,68 %
DE2_115_TV	2.117	6,10 %
Printmap	3.578	10,32 %
Main	5.104	14,72 %
Nios_system	21.869	63,09 %
de2_115_ipcore(todo)	**34.661**	**100,00 %**
de2_115_ipcore vs Cyclone IV	**34.661/114.480**	**30,18 %**

Tabla 3.16: Compilación de2_115_ipcore completo

Capítulo 4.
Resultados

En este capítulo se efectúan medidas de desempeño del producto final, con el fin de contar con herramientas que permitan determinar si los objetivos del trabajo fueron alcanzados.

4.1. Pruebas de desempeño

4.1.1. Tiempos de ejecución de funciones

Con el objetivo de efectuar mediciones de desempeño del sistema, se desarrolló un programa en el procesador NiosII que ordena diferentes comandos al ***IP core*** a través de las líneas paralelas de comunicación con el módulo <Main>, con el fin de medir los tiempos de ejecución de las diferentes funciones. A su vez, el módulo <Main> posee una sección de depuración (fig. 4.1) dotada de un contador de ciclos del reloj maestro, que inicia cuando <Main> ordena la operación del módulo <Printchar> o <Printmap> para cada comando, y detiene su conteo al finalizar la tarea del módulo. Este contador está conectado a un puerto de entrada en el procesador, de manera que puede capturarse la cantidad de ciclos de reloj; al divi-

dir este valor por la frecuencia del reloj maestro del sistema (75 MHz), se obtienen medidas de tiempo para cada función en particular. Mediante la tabla 4.1 se hace un reporte de desempeño generado por el procesador a través del terminal de stdout del procesador (JTAG UART).

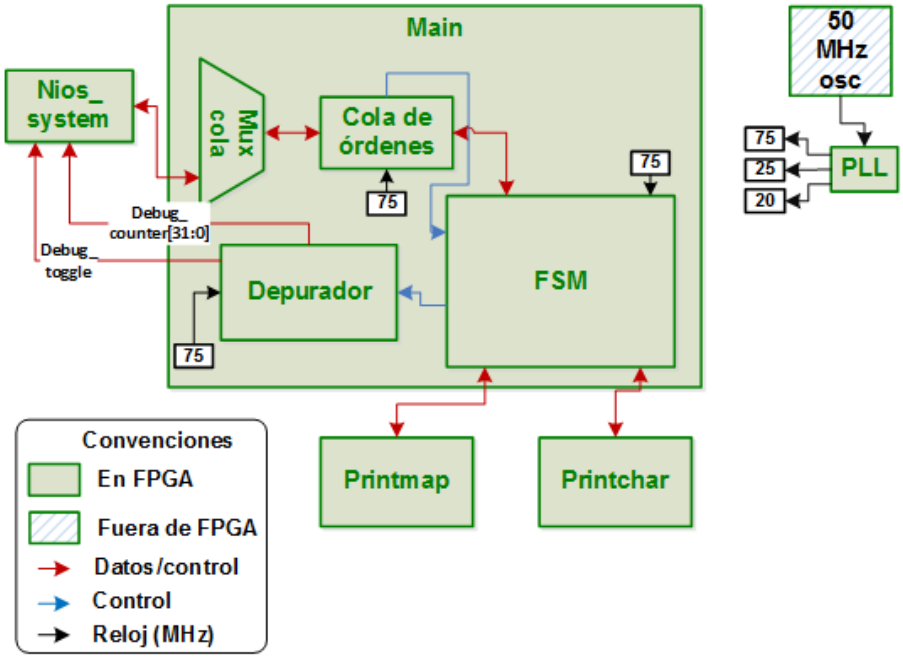

Figura 4.1: Esquema de medición de tiempos de funciones ***IP core***

4.1.2. Perfil de desempeño

Con el fin de determinar la frecuencia de ocurrencia de funciones del ***IP core*** y poder determinar su relación con el tiempo que cada función consume para su ejecución, se desarrolló un programa (fig. 4.2) en el procesador NiosII que decodifica las órdenes que son enviadas al módulo <Main> cuando el ***IP core*** está conectado al Sistema de Armas "Barracuda", pudiendo determinar el número de veces que se

Función	Comando	Cic. reloj	Tiempo
Borrado de pantalla	Borrar pantalla: “{d64”	2’419.934	32,266 ms
Impresión mapa grande (320x320)	Posición, número de mapa: “{p04800512{m103”	1’233.309	16,440 ms
Impresión de ventana	Posición, imprimir ventana: “{p04240100{wpf00”	610.443	8,139 ms
Impresión de caja	Color, esquina inicio caja + esquina fin caja: “{cf0f{fb0000000012791023”	93.209	1,243 ms
Impresión círculo de 500 píxeles de radio	Color, posición de círculo + radio de círculo: “{cfff{fc06400512500”	55.561	0,741 ms
Impresión de arco	Color, punto central del arco + radio arco + ángulo inicio arco + ángulo fin arco : “{c999{fa06400700300045135”	30.560	0,407 ms
Impresión línea esquina a esquina	Color, inicio de línea + fin de línea: “{cf00{fl0000000012791023”	21.510	0,287 ms
Impresión mapa pequeño (18x20)	Posición, número de mapa: “{p04800512{m007”	4.605	0,061 ms
Impresión de vector	Color, punto inicio vector + largo vector + ángulo vector : “{c0f0{fv06400512400047”	2.065	0,027 ms
Impresión letra más grande normal	Tamaño letra, imprimir letra: “{S24A”	1.167	0,016 ms
Impresión letra pequeña normal	Tamaño letra, imprimir letra: “{S00A”	111	0,001 ms
Impresión letra especial calibri 10	Tamaño letra, imprimir letra: “{t10W”	52	0,001 ms

Tabla 4.1: Tiempos de ejecución de las funciones del ***IP core***

cumple cada función durante un tiempo de operación del Sistema de Armas que también es contabilizado por el programa. Se escogieron dos escenarios de operación del Sistema de Armas: el primero operando en el modo de vigilancia y el segundo operando de forma combinada entre el modo configuración y vigilancia, generando

los gráficos de número de ocurrencias por función de las figs. 4.3 y 4.4; las funciones que mayor tiempo consumen son las de menor frecuencia de ocurrencia.

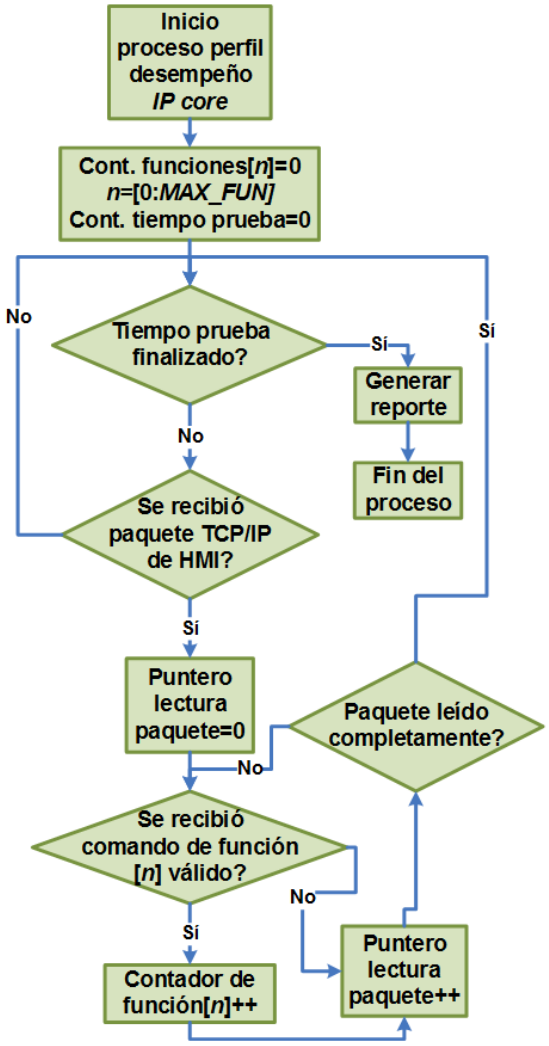

Figura 4.2: Diagrama de flujo para perfil de desempeño ***IP core***

Mediante las tablas 4.2 y 4.3 se presentan los datos del tiempo total de cada función; puede observarse que dentro de las exigencias de operación del Sistema "Barracuda", los tiempos recesivos del ***IP core*** son mucho mayores que los tiempos

activos del mismo, lo cual demuestra que su diseño es perfectamente adecuado a las necesidades de un Sistema de Armas, es decir, las funciones se cumplen en tiempos muy cortos en relación a la carga impuesta por el sistema que utiliza el ***IP core***, que en el primer modo no superó el 1 % y en el segundo no superó el 2 % del tiempo de operación del sistema; también cabe mencionar que la función de borrado de pantalla, a pesar de ser la de mayor carga en tiempo de ejecución, fue la que menos se ejecutó y en consecuencia ocupó la menor porción de tiempo durante la prueba.

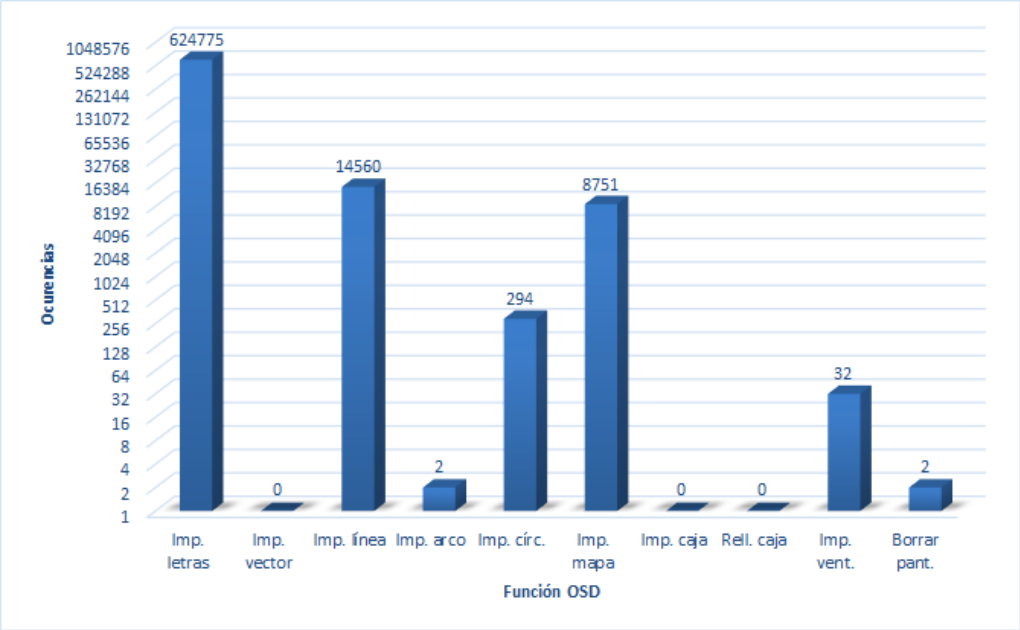

Figura 4.3: Perfil de desempeño ***IP core*** modo vigilancia, para el Sistema de Armas "Barracuda" (tiempo de prueba 11' 37")

Función	Ocurrencias	Tiempo/función	Tiempo total	Porcentaje
Impresión letra	624.775	0,001 ms	624,78 ms	0,090 %
Impresión vector	0	0,027 ms	0,00 ms	0,000 %
Impresión línea	14.560	0,287 ms	4.178,72 ms	0,600 %
Impresión arco	0	0,407 ms	0,00 ms	0,000 %
Impresión círculo	294	0,741 ms	217,85 ms	0,031 %
Impresión mapa	8.751	0,061 ms	533,81 ms	0,077 %
Impresión caja	0	1,243 ms	0,00 ms	0,000 %
Impresión ventana	32	8,139 ms	260,45 ms	0,037 %
Borrado pantalla	2	32,266 ms	64,53 ms	0,009 %
Subtotal funciones			5.880,14 ms	0,844 %
Tiempo de prueba			697.000,00 ms	100,000 %
Tiempo inactividad			691.119,86 ms	99,156 %

Tabla 4.2: Perfil de desempeño ocurrencias/tiempo ***IP core*** modo vigilancia (tiempo de prueba 11' 37")

4.2. Análisis de resultados

Es muy significativa la mejoría que se ha alcanzado con el desarrollo de este núcleo de propiedad intelectual. Los Sistemas de Armas navales Colombianos anteriormente disponían de una HMI muy limitada en los despliegues gráficos. El uso de tarjeta genéricas para la combinación de video e información OSD, tales como la XBOB-3 [7], limitaban la capacidad a sólo poder visualizar caracteres en 8 colores, con una resolución de 480x221 píxeles y con salida de video análogo compuesto; en contraste, ahora se tiene la capacidad de generar no sólo caracteres sino también figuras geométricas, todo esto a una resolución SXGA (1280x1024 píxeles) [16] y a través de interfaces VGA y DVI (fig. 4.5). No obstante, a pesar de todas las ventajas mencionadas, quizás la más importante es que ahora se dispone de un sistema que

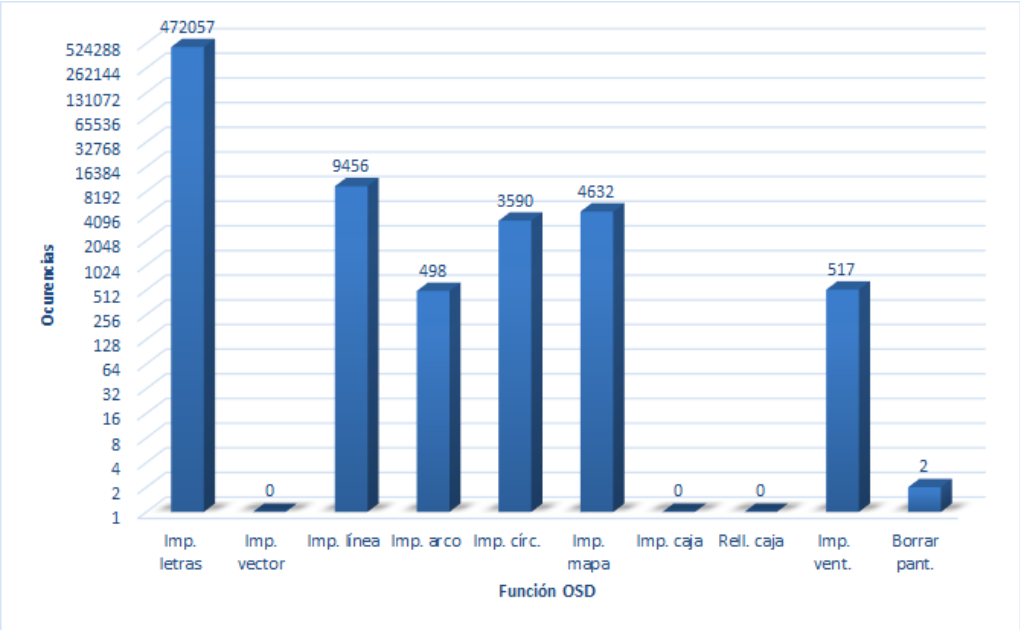

Figura 4.4: Perfil de desempeño ***IP core*** modo configuración/vigilancia, para el Sistema de Armas "Barracuda" (tiempo de prueba 9' 52")

puede ser modificado para satisfacer requerimientos futuros, basado en la capacidad que tienen las FPGAs de ser re-programadas en cualquier momento.

En la tabla 4.4 pueden observarse varios criterios de comparación en especificación y desempeño del ***IP core*** frente a la tarjeta de video XBOB-3 [7], donde se evidencia que el sistema desarrollado cumple con los requisitos previstos para el desarrollo del proyecto y supera en todas las funciones al esquema de video del Sistema de Armas "Escorpión".

Función	Ocurrencias	Tiempo/función	Tiempo total	Porcentaje
Impresión letra	472.057	0,001 ms	472,06 ms	0,080 %
Impresión vector	0	0,027 ms	0,00 ms	0,000 %
Impresión línea	9.456	0,287 ms	2.713,87 ms	0,458 %
Impresión arco	498	0,407 ms	202,69 ms	0,034 %
Impresión círculo	3590	0,741 ms	2.660,19 ms	0,449 %
Impresión mapa	4.632	0,061 ms	282,55 ms	0,048 %
Impresión caja	0	1,243 ms	0,00 ms	0,000 %
Impresión ventana	517	8,139 ms	4.207,86 ms	0,711 %
Borrado pantalla	2	32,266 ms	64,53 ms	0,011 %
Subtotal funciones			10.603,75 ms	1,791 %
Tiempo de prueba			592.000,00 ms	100,000 %
Tiempo inactividad			581.396,25 ms	98,209 %

Tabla 4.3: Perfil de desempeño ocurrencias/tiempo ***IP core*** modo configuración/vigilancia (tiempo de prueba 9’ 52”)

Figura 4.5: Despliegue ***IP core*** en una consola del Sistema de Armas "Barracuda"

Función	***IP core***	XBOB-3
Salidas de video:		
DVI	✓	
VGA	✓	
Compuesto		✓
Entradas de video:		
DVI	✓	
Compuesto		✓
Interfaces de control:		
LAN	✓	
RS232	✓	✓
Soporte en TCP/IP	✓	
Número de capas OSD	2	1
Resolución de pantalla OSD	1280x1024	480x221
Profundidad de colores OSD	32.768	8
Tiempo mínimo de transferencia de 1 byte a la cola de órdenes	0,1 us	65,1 us
Soporte en manejo de mapas	✓	
Descarga de mapas soportados en sistema de archivos FAT	✓	
No encasillamiento de caracteres	✓	
Soporte en figuras líneas, circulos, cajas, arcos	✓	
Soporte en manejo de ventanas	✓	
Soporte en manejo de dispositivo apuntador (ratón)	✓	

Tabla 4.4: Comparación ***IP core*** (sistema "Barracuda") Vs tarjeta XBOB-3 (sistema "Escorpión")

Capítulo 5.
Conclusiones y recomendaciones

5.1. Conclusiones

1. La utilización de técnicas de diseño digital y diseño híbrido HW/SW para dispositivos tipo FPGA permiten implementar soluciones robustas en el área de desarrollo de núcleos de propiedad intelectual para manejo y generación de video. Esto permite concluir que el objetivo general del presente trabajo ha sido alcanzado, consistente en "disponer de un sistema HW/SW para combinar información HMI con una imagen procedente de una fuente de video, que mejore las características funcionales de calidad de despliegue, velocidades y utilización aplicado en los Sistemas de Armas "Escorpión" de la Armada Nacional".

2. El control maestro de un ***IP core*** puede desarrollarse como un módulo descrito mediante HDL, que permitió en este proyecto no depender del procesador embebido para desarrollar el intérprete principal de la cola de órdenes del sistema, sino que su implementación se hizo mediante máquinas de estados

finitos desarrolladas en lenguaje de descripción de hardware, lo cual llevó al cumplimiento del objetivo específico "Obtener el control maestro del ***IP core*** mediante la utilización de lenguaje Verilog para el manejo de procesos secuenciales, en el desarrollo del módulo".

3. Las tarjetas de desarrollo basadas en FPGAs permiten incorporar una amplia gama de núcleos de propiedad intelectual, de manera que las funciones de dicho ***IP core*** puedan contar con una plataforma adecuada para ser implementadas físicamente, lo cual permitió cumplir con el objetivo específico "Implementar el ***IP core*** en una tarjeta dotada de las capacidades mínimas de referencia, con el fin de evaluar su desempeño".

4. Las capacidades con que cuenta el esquema de video del Sistema de Armas "Escorpión" de la Armada Nacional han sido superadas con los resultados del presente trabajo, lo cual permitirá su utilización en mejoras al mismo y a Sistemas de Armas de mayor complejidad. De esta manera se da por cumplido el objetivo específico "Comparar el desempeño entre el ***IP core*** implementado en una tarjeta de desarrollo y el esquema de despliegue actualmente en uso en el Sistema de Armas "Escorpión" de la Armada Nacional".

5. Todo núcleo de propiedad intelectual debe ir acompañado de una documentación para su adecuado uso, objetivo específico que se cumplió presentando la hoja de características técnicas del sistema en el apéndice A y el protocolo de

interfaz de usuario en el apéndice B.

6. Como técnica de diseño digital es recomendable limitar los dominios de reloj que se utilizan, de manera que la mayor parte de los módulos puedan acogerse a un mismo dominio, a menos que restricciones asociadas a estándares internacionales impongan utilizar variados dominios, como ha ocurrido en el presente trabajo.

5.2. Recomendaciones

El presente trabajo incluyó el diseño y montaje del ***IP core***. Se plantean las siguientes actividades como trabajo futuro:

1. Desarrollo de una simulación modular e integral para garantizar la máxima robustez posible, técnica utilizable en sistemas desarrollados en HDL.

2. Ampliar la capacidad del sistema implementando soporte en resoluciones de video superiores.

3. Probar el ***IP core*** en otros dispositivos FPGA con tarjetas de desarrollo diferentes a las utilizadas en el presente trabajo, con el fin de medir y/o aumentar la capacidad de ser usado el proyecto en un esquema multi-plataforma.

Bibliografía

[1] S. Palnitkar, *Verilog HDL: a guide to digital design and synthesis*. Prentice Hall, 2 ed., 2003.

[2] J. F. Toledo-Moreo and et al, "Fpga-based architecture for the real-time computation of 2-d convolution with large kernel size," *Journal of Systems Architecture*, vol. 58, no. 8, pp. 277–285, 2012.

[3] B. Krill and et al, "An efficient fpga-based dynamic partial reconfiguration design flow and environment for image and signal processing ip cores," *Signal Processing: Image Communication*, vol. 25, no. 5, pp. 377–387, 2010.

[4] A. Dollas, I. Ermis, I. Koidis, I. Zisis, and C. Kachris, "An open tcp/ip core for reconfigurable logic, in field-programmable custom computing machines," *Annual IEEE Symposium on IEEE Computer Society*, pp. 297–298, 2005.

[5] IEEE Xplore Digital Library, "1364-2005 - IEEE Standard for Verilog Hardware Description Language." http://ieeexplore.ieee.org/xpl/articleDetails.jsp?

arnumber=1620780&filter%3DAND%28p_Publication_Number%3A10779%29, 2015. [En línea; consultado 02-Octubre-2015].

[6] IEEE Xplore Digital Library, "1076-2008 - IEEE Standard VHDL Language Reference Manual." http://ieeexplore.ieee.org/xpl/login.jsp?tp=&arnumber=4772740&url=http%3A%2F%2Fieeexplore.ieee.org%2Fstamp%2Fstamp.jsp%3Ftp%3D%26arnumber%3D4772740, 2015. [En línea; consultado 02-Octubre-2015].

[7] Decade engineering, "XBOB Video OSD / On Screen Display Board." http://www.decadenet.com/XBOB/XBOB.html, 2015. [En línea; consultado 22-Septiembre-2015].

[8] Panasonic Corporation of North America, "Color monitor WV-CM1020." http://www.panasonic.com/ph/business/business-solution/security-system/viewing-screen/wv-cm1020.html, 2015. [En línea; 22-Septiembre-2015].

[9] Terasic, "Altera DE2-70 Board." http://www.terasic.com.tw/cgi-bin/page/archive.pl?Language=English&CategoryNo=183&No=226, 2015. [En línea; consultado 22-Septiembre-2015].

[10] Terasic, "Altera DE2-115 Development and Education Board." http://www.terasic.com.tw/cgi-bin/page/archive.pl?Language=English&CategoryNo=139&No=502, 2015. [En línea; consultado 22-Septiembre-2015].

[11] Altera Corporation, "High Speed Mezzanine Card (HSMC) Specification." https://www.altera.com/content/dam/altera-www/global/en_US/pdfs/literature/ds/hsmc_spec.pdf, 2015. [En línea; consultado 10-Octubre-2015].

[12] Altera Corporation, "Quartus II Subscription Edition Software." http://www.altera.com/products/software/quartus-ii/subscription-edition/qts-se-index.html, 2015. [En línea; consultado 9-Febrero-2015].

[13] Altera Corporation, "Nios II Embedded Design Suite." https://www.altera.com/products/design-software/embedded-software-developers/nios-ii-eds.html#SBTE, 2015. [En línea; consultado 9-Febrero-2015].

[14] Steve Cunningham, "The Ubiquitous Bresenham Algorithm as a Basis for Graphics Interpolation Processes." https://www.cs.csustan.edu/~rsc/sdsu/Interpolation.pdf, 2015. [En línea; consultado 02-Octubre-2015].

[15] Integrated Solution Silicon INC, "IS42S16320B." http://www.issi.com/WW/pdf/42S16320B-86400B.pdf, 2015. [En línea; consultado 26-Septiembre-2015].

[16] Seconds Ltd., "VESA Signal 1280 x 1024 @ 60 Hz timing." http://tinyvga.com/vga-timing/1280x1024@60Hz, 2015. [En línea; consultado 26-Septiembre-2015].

[17] Altera Corporation, "Qsys - Altera's System Integration Tool." https://www.altera.com/products/design-software/fpga-design/quartus-ii/

quartus-ii-subscription-edition/qts-qsys.html, 2015. [En línea; consultado 22-Septiembre-2015].

Glosario

IP core Intellectual Property Core. 13, 15, 18, 29, 31–33, 36, 40, 51, 52, 55, 58, 63, 65, 67, 70, 73, 74, 79, 81–93, 108, 114, 116

Siglas

ASCII American Standard Code for Information Interchange. 23, 25, 39, 114

ASIC Application Specific Integrated Circuit. 15, 30

CGRAM Character Generator Random Access Memory. 23–25, 27

CGROM Character Generator Read Only Memory. 23–25, 42

CPU Central Processing Unit. 15

CRT Cathode Ray Tube. 28

DAC Digital to Analog Converter. 34, 67, 68

DVI Digital Visual Interface. 16, 17, 31, 34, 35, 68, 86, 90, 105, 107, 108

EDA Electronic Design Automation. 15

FAT File Allocation Table. 54, 106

FIFO First Input First Output. 38, 46–48, 50, 65, 67, 69, 76, 77

FOV Field Of View. 21

FPGA Field Programmable Gate Array. 14–16, 30, 33, 34, 38, 47, 51, 57, 79, 80, 87, 91–93

FSM Finite State Machine. 36, 39, 40, 42, 46

HDL Hardware Description Language. 14, 16, 35, 91, 93

HMI Human Machine Interface. 13, 14, 17, 18, 21, 51–53, 86, 91, 107

HSMC High Speed Mezzanine Card. 35, 68

I2C Inter Integrated Circuit. 13

IDE Integrated Development Environment. 57

JTAG Joint Test Action Group. 82

LAN Local Area Network. 58, 90

LUT Look Up Table. 44, 45

MAC Media Access Controller. 51, 58

NTSC National Television System Committee. 21

OSD On Screen Display. 21–23, 25–28, 31, 33, 40, 47, 50, 59, 64, 65, 70, 77, 86, 90, 105–108, 114

PCI Peripheral Component Interconnect. 15

PIO Parallel Input Output. 52, 53

PSP Paralell Slave Port. 13

RAM Random Access Memory. 23, 25, 27, 34, 38, 48, 57, 75

RGB Red Green Blue. 34, 107, 110, 112

ROM Read Only Memory. 23

SD Secure Digital. 34, 51, 54–58, 71, 78

SDRAM Synchronous Dynamic Random Access Memory. 33, 34, 40, 42, 46–48, 50, 65, 67–69, 76, 77

SOPC System on a Programmable Chip. 30, 57

SPI Serial Peripheral Interface. 13

SRAM Static Random Access Memory. 34, 108

SXGA Super eXtended Graphics Array. 50, 75, 76, 86

TCP/IP Transmission Control Protocol/Internet Protocol. 31, 51, 90

TSE Triple Speed Ethernet. 51, 79

UART Universal Asynchronous Receiver/Transmitter. 15, 82

VGA Video Graphics Array. 17, 27, 34, 48–50, 67, 68, 86, 90, 105

Apéndices

Capítulo A.
Características *IP core*

Implementación:	Tarjeta ALTERA TERASIC DE2-115.
FPGA:	Cyclone IV (EP4CE115).
Alimentación:	12 VDC.
Consumo:	445 mA (5,34 W).
Control:	TCP/IP (100 Mbps): Puerto 3.500 (servicio OSD). Puerto 3.700 (servicio MOUSE REMOTO). RS232: (+/- 10VDC). 9.600, 19.200, 38.400, 76.800 y 115.200 bps.
Frecuencia operación núcleo:	75 Mhz.
Uso elementos lógicos:	34.661 (30,18 %).
Uso pines:	425 (80 %).
Uso Memoria embebida:	345 KB (71 %).
Entrada de video:	DVI-D.
Salidas de video:	DVI-D, VGA.
Resolución de video:	1280x1024, 32.768 colores, 60 Hz.

Número de OSDs:	2
Funciones de impresión:	Borrado de pantalla: llenado de pantalla completa ó 64 franjas de 16 líneas c/u; el color puede seleccionarse. Letras normales: (8x8, 9x9, 9x10, 10x10); los tamaños pueden escalarse hasta el triple en ancho y hasta cinco veces en alto. Letras Calibri 9 hasta 16. Mapas de bits: hasta 1.000 mapas, tamaño mínimo 2x2, tamaño máximo 640x640. Líneas: máxima separación en eje X=1.280, eje Y=1.024. Cajas: máximo ancho 1.280, máximo alto 1.024. Vectores: máximo largo 999. Círculos: máximo radio 999. Arcos: máximo radio 999. Ventanas: máximo ancho 1.280, máximo alto 1.024.
Medio de actualización de mapas:	memoria SDCARD, sistema de archivos FAT.

Capítulo B.
Protocolo *IP core*

NÚCLEO DE PROPIEDAD INTELECTUAL PARA COMBINAR VIDEO DVI E INFORMACIÓN GRÁFICA HMI.

1. PROTOCOLO COMÚN A FUNCIONES DE GRÁFICOS Y CARACTERES.

1.1. Borrar pantalla (delete, Delete).

{d _ _ ó {D _ _

_ _ : bloque de 16 líneas de alto a borrar [00 a 63]. [64] borra la pantalla completa. El borrado se hace sobre la OSD activa, haciendo uso del comando {o.

1.2. Fijar color de borrado.

{k _ _ _

_ _ _ : color (RGB) de borrado [0x000 a 0xFFF].

1.3. Seleccionar página OSD a escribir (output).

{o _

_ : página de OSD. 0 (pág. 0), 1 (pág. 1).

1.4. Configuración gráfica (graphics).

{g _ _ _

_ : OSD activa. 0 (ninguna), 1 (pág. 0), 2 (pág. 1), 3 (págs. 0 y 1).

_ : OSD prioritaria. 0 (prioridad pág. 0), 1 (prioridad pág. 1).

_ : entrada de video activa. 0 (ninguna), 1 (composite), 2 (DVI).

2. PROTOCOLO DE GENERACIÓN DE GRÁFICOS.

2.1. Asignar control memoria Flash al procesador Nios.

{X

Nota: se debe tener precaución con esta orden ya que se trata de un recurso compartido entre el módulo Printmap y el procesador Nios.

2.2. Asignar control memoria flash al ***IP core***.

{x

Nota: se debe tener precaución con esta orden ya que se trata de un recurso compartido entre el módulo Printmap y el procesador Nios.

2.3. Asignar control memoria SRAM al procesador Nios.

{Y

Nota: se debe tener precaución con esta orden ya que se trata de un recurso compartido entre el módulo Tv_decoder y el procesador Nios.

2.4. Asignar control de memoria SRAM al ***IP core***.

{y

Nota: se debe tener precaución con esta orden ya que se trata de un recurso compartido entre el módulo Tv_decoder y el procesador Nios.

2.5. Configuración modo mouse (Mouse).

{M _

_ : 0 (deshabilitar mouse), 1 (mouse local), 2 (mouse remoto).

Mouse local: en esta configuración el dispositivo apuntador debe estar conectado al puerto PS/2 de la tarjeta.

Mouse remoto: en este modo, el procesador Nios recibe la posición del puntero a través de un socket TCP/IP.

2.6. Determinar la posición actual de impresión de mapas y ventanas (position).

{p _ _ _ _ _ _ _ _

_ _ _ _ : coordenada de columna (x) [0000 a 1279].

_ _ _ _ : coordenada de fila (y) [0000 a 1023].

2.7. Imprimir un mapa de bits (map).

{m _ _ _

_ _ _ : número de mapa a imprimir [000 a 999].

2.8. Borrar mapa (erase).

{e _ _ _

_ _ _ : número de mapa a borrar [000 a 999].

2.9. Determinar el color del trazado de figuras de línea, caja, vector, círculo y arco (color).

{c _ _ _

_ _ _ : color (RGB) [0x000 a 0xFFF].

2.10. Imprimir una figura de línea, caja, vector, círculo o arco (figure).

{fl _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ : para trazado de línea (line)

{fb _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ : para trazado de caja (box)

_ _ _ _ : coordenada de columna (x) de inicio de la figura [0000 a 1279].

_ _ _ _ : coordenada de fila (y) de inicio de la figura [0000 a 1023].

_ _ _ _ : coordenada de columna (x) de término de la figura [0000 a 1279].

_ _ _ _ : coordenada de fila (y) de término de la figura [0000 a 1023].

{fv _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ : para trazado de vector (vector)

_ _ _ _ : coordenada de columna (x) de inicio del vector [0000 a 1279].

_ _ _ _ : coordenada de fila (y) de inicio del vector [0000 a 1023].

_ _ _ : largo del vector [002 a 999].

_ _ _ : ángulo del vector [000 a 359].

{fc _ _ _ _ _ _ _ _ _ _ _ _ : para trazado de círculo (circle)

_ _ _ _ : coordenada de columna (x) del centro del círculo [0000 a 1023].

_ _ _ _ : coordenada de fila (y) del centro del círculo [0000 a 1023].

_ _ _ : radio del círculo [002 a 999].

{fa _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ : para trazado de arco (arc)

_ _ _ _ : coordenada de columna (x) del centro del arco [0000 a 1023].

_ _ _ _ : coordenada de fila (y) del centro del arco [0000 a 1023].

_ _ _ : radio del arco [002 a 999].

_ _ _ : ángulo de inicio del arco [000 a 359].

_ _ _ : ángulo de término del arco [000 a 359].

3. PROTOCOLO DE GENERACIÓN DE CARACTERES.

3.1. Determinar la posición actual de impresión de caracteres (Position).

{P _ _ _ _ _ _ _ _

_ _ _ _ : coordenada de columna (x) [0000 a 1279].

_ _ _ _ : coordenada de fila (y) [0000 a 1023].

Cada vez que se digita un caracter, la posición actual de despliegue de caracteres se incrementa tantas columnas como píxeles de ancho representa el caracter previamente impreso. El núcleo inicia con la posición [0000 0000] por defecto (esquina superior izquierda).

3.2. Determinar el color de caracteres a imprimir (Color).

{C _ _ _

_ _ _ : color (RGB) [0x000 a 0xFFF].

Todo caracter que se digite en adelante será desplegado con este color, hasta que se modifique mediante un nuevo comando de color. El núcleo inicia con el color [0xFFF] por defecto (caracter blanco).

3.3. Fijar color de fondo para caracteres (background).

{b _ _ _

_ _ _ : color (RGB) de fondo [0x000 a 0xFFF].

3.4. Activar caracteres intermitentes (Blink) o fijos (Fixed).

{B : selección caracteres intermitentes (Blink).

{F : selección caracteres fijos (Fixed).

Todo caracter que se digite en adelante será desplegado de manera intermitente [B] o fija [F] hasta que se modifique mediante un nuevo comando de este tipo.

3.5. Seleccionar fuente base para generación de caracteres tipo "printchar" normales (Type).

{T _

_ : 0 (fuente 8x8), 1 (fuente 9x9), 2 (fuente 9x10), 3 (fuente 10x10).

3.6. Determinar el tamaño de los caracteres a imprimir tipo "printchar" nor-

males (Size).

{S _ _

_ : ancho del caracter: [0]: 8 píxeles, [1] 16 píxeles, [2] 24 píxeles.

_ : alto del caracter: [0]: 8 píxeles, [1] 16 píxeles, [2] 24 píxeles, [3] 32 píxeles, [4] 40 píxeles.

Todo caracter que se digite en adelante será desplegado con este tamaño, hasta que se modifique mediante un nuevo comando de tamaño.

3.7. Activar caracteres inversos (Inverted) o normales (Normal) tipo "printchar" normal.

{I : selecciona caracteres invertidos (Inverted).

{N : selecciona caracteres normales (Normal).

Todo caracter que se digite en adelante será desplegado de manera invertida [I] o normal [N] hasta que se modifique mediante un nuevo comando de este tipo.

3.8. Seleccionar fuente base para generación de caracteres tipo "special" (type).

{t _ _

_ : tamaño caracter [0 a 7].

_ : habilitar/deshabilitar borde de caracter. 0 (deshabilitar), 1

(habilitar).

3.9. Imprimir caracter.

Sólo debe enviarse el código ASCII de la letra que se desea imprimir. Debe tenerse en cuenta que si se encuentra transfiriendo a la cola de datos los caracteres que conforman un comando (es decir, precedidos de {), dichos caracteres serán considerados parte del comando y no se imprimirán en la OSD.

4. PROTOCOLO DE FUNCIONES DE VENTANAS.

Para facilitar y acelerar el diseño gráfico haciendo uso del ***IP core***, el sistema permite la creación de elementos compuestos tipo VENTANA. Dichos elementos incorporan en un solo contenedor varios de los elementos básicos que pueden ser generados con el ***IP core***. La siguiente es una lista de las características soportadas en el entorno de ventanas:

- Máximo 20 ventanas.
- Color de borde para la ventana.
- Color de la barra de título de la ventana.
- Texto de título de la ventana (hasta 32 caracteres).
- Color de título de la ventana.
- Tamaño de la ventana (hasta 1279x1023).

- La posición de los elementos definidos dentro de una ventana es relativa al vértice superior izquierdo (origen) de la ventana.
- Hasta 50 elementos de texto por ventana. Cada elemento puede tener máximo 16 caracteres.
- Hasta 25 mapas de bits asociados a una ventana.
- Hasta 25 figuras rectangulares asociadas a una ventana. Cada elemento puede ser configurado en color y tamaño.
- Hasta 25 líneas asociadas a una ventana. Cada elemento puede ser configurado en color, longitud y orientación (horizontal/vertical).

4.1. Creación de nueva ventana.

{wcf _ TITULO_VENTANA}

_ _ : identificador de la ventana [00 a 19].

_ _ _ _ : ancho en píxeles de la ventana [0000 a 1279].

_ _ _ _ : alto en píxeles de la ventana [0000 a 1023].

_ _ _ : color de la barra de título [0x000 a 0xFFF].

_ _ _ : color de texto del título [0x000 a 0xFFF].

_ _ _ : color de fondo de la ventana [0x000 a 0xFFF].

_ _ _ : color de borde de la ventana [0x000 a 0xFFF].

TITULO_VENTANA: texto de título de la ventana.

4.2. Impresión de ventana.

{wpf _ _

_ _ : identificador de la ventana destino [00 a 19].

Nota: la posición absoluta del origen de la ventana en la pantalla estará dada por la última posición establecida en el apuntador del módulo printmap, el cual es fijado a través del comando {p.

4.3. Borrado de ventana.

{wef _ _

_ _ : identificador de la ventana destino [00 a 19].

Nota: no es necesario especificar la posición de borrado con {p para este comando ya que la posición de impresión de la ventana está almacenada en la memoria del ***IP core***.

4.4. Eliminar ventana.

{wkf _ _

_ _ : identificador de la ventana destino [00 a 19].

Nota: este comando elimina la información de creación de la ventana. Luego de ejecutado no se cumplirá ningún comando que se refiera a dicha ventana, hasta crearla nuevamente.

4.5. Comandos de creación y modificación de elementos de texto en ventanas.

4.5.1. Creación elemento de texto.

Dependiendo si el texto creado es tipo normal “printchar” o tipo

"special" (mapa de bits), el comando puede tomar 2 formas:

4.5.1.1. Creación de texto tipo normal.

{wct _ _ _ _ p _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _

_ _ : identificador de la ventana destino [00 a 19].

_ _ : identificador del elemento de texto [00 a 49].

p : indicador letra "printchar" normal

_ : habilitar/deshabilitar estilo invertido. 0 (deshabilitar), 1 (habilitar).

_ : ancho de caracter. Se aceptan valores de ancho como fueron definidos para el comando {S_ _, [0 a 2].

_ : alto de caracter. Se aceptan valores de alto como fueron definidos para el comando {S_ _, [0 a 4].

_ : tipo de caracter del modo normal ("printchar"). Sirve para especificar la fuente base para la generación de caracteres.

_ _ _ : color de caracter [0x000 a 0xFFF].

_ _ _ : color de fondo de caracter [0x000 a 0xFFF].

_ _ _ _ : coordenada de posición X relativa al origen de la ventana [0000 a 1279].

_ _ _ _ : coordenada de posición Y relativa al origen de la ventana [0000 a 1023].

4.5.1.2. Creación de texto tipo “special” (caracteres mapa de bits).

{wct _ _ _ _ s _ _ _ _ _ _ _ _ _ _ _ _ _ _ _

_ _ : identificador de la ventana destino [00 a 19].

_ _ : identificador del elemento de texto [00 a 49].

s : indicador letra “special”

_ : tamaño de caracter “special” [0 a 7].

_ : habilitar/deshabilitar borde en caracter “special”. 0 (deshabilitar), 1 (habilitar).

_ _ _ : color de caracter [0x000 a 0xFFF].

_ _ _ : color de fondo de caracter [0x000 a 0xFFF].

_ _ _ _ : coordenada de posición X relativa al origen de la ventana [0000 a 1279].

_ _ _ _ : coordenada de posición Y relativa al origen de la ventana [0000 a 1023].

4.5.2. Escritura de elemento de texto.

{wwt _ _ _ _ TEXTO}

_ _ : identificador de la ventana destino [00 a 19].

_ _ : identificador del elemento de texto de destino [00 a 49].

TEXTO: cadena de texto a imprimir (máximo 16 caracteres).

4.5.3. Borrado de elemento de texto.

{wet _ _ _ _

_ _ : identificador de la ventana destino [00 a 19].

_ _ : identificador del elemento de texto de destino [00 a 49].

4.5.4. Eliminar elemento de texto.

{wkt _ _ _ _

_ _ : identificador de la ventana destino [00 a 19].

_ _ : identificador del elemento de texto de destino [00 a 49].

4.6. Comandos de creación y modificación de elementos de mapas de bits en ventanas.

4.6.1. Creación de elemento mapa de bits.

{wcm _ _ _ _ _ _ _ _ _ _ _ _

_ _ : identificador de la ventana destino [00 a 19].

_ _ : identificador del elemento de mapa de bits [00 a 24].

_ _ _ _ : coordenada de posición X relativa al origen de la ventana [0000 a 1279].

_ _ _ _ : coordenada de posición Y relativa al origen de la ventana [0000 a 1023].

4.6.2. Escritura de elemento mapa de bits.

{wwm _ _ _ _ _ _ _

_ _ : identificador de la ventana destino [00 a 19].

_ _ : identificador del elemento de mapa de bits [00 a 24].

_ _ _ : identificador de mapa de bits [000 a 999].

4.6.3. Borrado elemento mapa de bits.

{wem _ _ _ _

_ _ : identificador de la ventana destino [00 a 19].

_ _ : identificador del elemento de mapa de bits [00 a 24].

4.6.4. Eliminar elemento mapa de bits.

{wkm _ _ _ _

_ _ : identificador de la ventana destino [00 a 19].

_ _ : identificador del elemento de mapa de bits [00 a 24].

4.7. Comandos de creación y modificación de elementos tipo rectángulo en ventanas.

4.7.1. Creación elemento rectángulo.

{wcb _

_ _ : identificador de la ventana destino [00 a 19].

_ _ : identificador del elemento rectángulo [00 a 24].

_ _ _ : color de rectángulo [0x000 a 0xFFF].

_ _ _ _ : ancho de rectángulo [0000 a 1279].

_ _ _ _ : alto de rectángulo [0000 a 1023].

_ _ _ _ : coordenada de posición X relativa al origen de la ventana

[0000 a 1279].

_ _ _ _ : coordenada de posición Y relativa al origen de la ventana

[0000 a 1023].

4.7.2. Borrado elemento rectángulo.

{web _ _ _ _

_ _ : identificador de la ventana destino [00 a 19].

_ _ : identificador del elemento rectángulo [00 a 24].

4.7.3. Eliminar elemento rectángulo.

{wkb _ _ _ _

_ _ : identificador de la ventana destino [00 a 19].

_ _ : identificador del elemento rectángulo [00 a 24].

4.8. Comandos de creación y modificación de elementos tipo línea en ventanas.

4.8.1. Creación elemento línea.

{wcl _

_ _ : identificador de la ventana destino [00 a 19].

_ _ : identificador del elemento línea [00 a 24].

_ _ _ : color de línea [0x000 a 0xFFF].

_ : orientación trazado de línea. 0 (horizontal), 1 (vertical).

_ _ _ _ : longitud de línea [0000 a 1279].

_ _ _ _ : coordenada de posición X relativa al origen de la ventana

[0000 a 1279].

_ _ _ _ : coordenada de posición Y relativa al origen de la ventana

[0000 a 1023].

4.8.2. Borrado elemento línea.

{wel _ _ _ _

_ _ : identificador de la ventana destino [00 a 19].

_ _ : identificador del elemento línea [00 a 24].

4.8.3. Eliminar elemento línea.

{wkl _ _ _ _

_ _ : identificador de la ventana destino [00 a 19].

_ _ : identificador del elemento línea [00 a 24].