UNIVERSIDAD DE CHILE
FACULTAD DE CIENCIAS FíSICAS Y MATEMÁTICAS
DEPARTAMENTO DE INGENIERÍA ELÉCTRICA

TRANSMISIÓN INALÁMBRICA DE IMÁGENES MÉDICAS PARA UN ECÓGRAFO ULTRA-PORTÁTIL

MEMORIA PARA OPTAR AL TÍTULO DE
INGENIERO CIVIL ELÉCTRICO

NICOLÁS IGNACIO HASBUN AVENDAÑO

PROFESOR GUÍA:
MANUEL DUARTE MERMOUD

MIEMBROS DE LA COMISIÓN:
VADER JOHNSON VERA
ANDRÉS CABA RUTTE

SANTIAGO DE CHILE
2018

RESUMEN DE LA MEMORIA PARA OPTAR
AL TÍTULO DE INGENIERO CIVIL ELÉCTRICO
POR: NICOLÁS IGNACIO HASBUN AVENDAÑO
FECHA: 2018
PROF. GUÍA: MANUEL DUARTE MERMOUD

TRANSMISIÓN INALÁMBRICA DE IMÁGENES MÉDICAS PARA UN ECÓGRAFO ULTRA-PORTÁTIL

Taote es un dispositivo de ultrasonido portable completamente diseñado y construido en los laboratorios de Ingeniería Eléctrica de la Universidad de Chile. Está orientado al uso en etapas tempranas de diagnóstico como una herramienta para médicos generales y como un dispositivo que permite ofrecer exámenes de ultrasonido en lugares de difícil acceso. Actualmente en etapa de comercialización como un producto terminado, se busca flexibilizar el uso del dispositivo e integrarlo con tecnologías portátiles y móviles modernas.

En el presente trabajo se procede a analizar y comparar diversas técnicas de envío y almacenamiento de imágenes digitales para sistemas embebidos utilizando enlaces inalámbricos orientado a la implementación en Taote con un especial énfasis en el cumplimiento de condiciones que garanticen el resguardo de la capacidad de diagnóstico médico para exámenes de ultrasonido. El trabajo se realizó ocupando entre otras herramientas, plataformas modernas de Altera, lenguajes de descripción de hardware, programación de rutinas por software en el lenguaje C y módulos de comunicación inalámbrica. En particular, se emplearon sistemas de circuitos **System on a Chip** (SoC) en conjunto con el uso de circuitos digitales re-programables **Field Programmable Gate Array** (FPGA). El uso de ambas tecnologías conforman los chips SoC-FPGA desarrollados bajo la línea de modelos de Altera Cylone V SoC — que se ocupan para este trabajo —, los cuales permiten juntar el mundo de los circuitos lógicos programables, con la flexibilidad y rapidez de desarrollo otorgada por los procesadores.

Los primeros resultados consistieron en la implementación de un módulo de comunicación serial en FPGA, que se integra específicamente para que el procesador del SoC tome control del módulo y permita ejecutar rutinas de software para la transacción de datos por cable. Este sistema de comunicación se re-utiliza de forma satisfactoria en el desarrollo de rutinas de transmisión y recepción inalámbrica de imágenes para las tecnologías de Bluetooth y WiFi.

Para el procesamiento de imágenes se presentó un sistema en FPGA con proyección digital de imagen a pantalla según una serie de datos almacenados en memoria que conforman un mapa de bits llamado **framebuffer**. Este esquema permitió crear rutinas en el procesador del SoC para tener acceso y modificar el **framebuffer** de modo que se crea un sistema de proyección de video completo controlado por software capaz de realizar capturas de pantalla y compresión de imágenes por JPEG para un posterior envío por transmisión inalámbrica.

Finalmente se presenta un sistema SoC-FPGA con proyección a pantalla, captura, compresión y envío inalámbrico de imágenes por Bluetooth. Este sistema conjunto satisface las necesidades de conectividad y almacenamiento de imágenes requeridas en un equipo moderno de ultrasonido al mismo tiempo que otorga un diseño versátil, robusto, escalable y modular lo que abre una puerta a futuros desarrollos que exploten la tecnología utilizada.

A la pepa,
quien me enseñó que uno si puede cambiar el mundo.

Tabla de Contenido

Índice de Tablas

Índice de Ilustraciones

Capítulo 1

Introducción

1.1. Motivación

Taote es un dispositivo de ultrasonido orientado al uso en etapas de pre-diagnóstico como una herramienta que busca complementar el examen médico de rutina. Es un proyecto que nace desde los laboratorios de Ingeniería Eléctrica de la Facultad de Ciencias Físicas y Matemáticas de la Universidad de Chile de la mano de los académicos e investigadores Manuel Duarte, Carlos Conca, Nicolás Beltrán, Vader Johnson, Rodrigo Maureira y el médico radiólogo John Mackinnon.

Figura 1.1: Ecógrafo Ultra-portátil Taote.

El equipo visto en la Figura 1.1, se caracteriza por ser un equipo autónomo y portable, de un costo competitivo en el mercado e innovador en el sector chileno. Actualmente se cuenta con unidades desplegadas en la «Universidad del Desarrollo» para los estudiantes de la Escuela de Medicina. Se busca con esto que los estudiantes de primeros años se familiaricen con el dispositivo e introducirlo al mercado del mismo modo en que el estetoscopio irrumpió en su momento. **Taote** se orienta como una herramienta complementaria para uso por médicos

generales, sin entrar a competir con equipos especializados diseñados para radiólogos.

En etapa de comercialización, su principal ventaja es la portabilidad con características que le permiten estar preparado incluso para acceder a zonas remotas, con independencia de baterías y acceso a lentes **HMD** (Head Mounted Display) especializados. En este contexto nace la necesidad de dotar al equipo funcionalidades nuevas, tanto de almacenamiento como de conexiones inalámbricas. Se busca flexibilizar el uso del aparato e integrarlo con tecnologías portátiles y móviles modernas que forman parte del día a día de los usuarios como lo son los teléfonos inteligentes, dispositivos táctiles y los computadores personales.

Buscando alcanzar estos niveles de conectividad, el trabajo estará orientado a generar la capacidad de capturar y transmitir imágenes a un dispositivo móvil externo ocupando tecnologías modernas y al alcance de los usuarios. Esto se traduce en hacer uso de plataformas de desarrollo recientes y el uso de sistemas de comunicación extendidos en la industria como lo son las tecnologías de **Bluetooth** y **WiFi**.

1.2. Objetivos

1.2.1. Objetivo General

Diseñar e implementar un sistema digital de transmisión inalámbrica de imágenes, orientado al uso de diagnóstico médico con ultrasonido para el equipo **Taote**.

1.2.2. Objetivos Específicos

- Estudiar arquitectura interna y recursos existentes en el equipo Taote para incluir funcionamiento de transmisión de imágenes.
- Generar un sistema de proyección digital de imágenes almacenado en memoria.
- Implementar sistemas de compresión de imágenes y manejo de archivos.
- Analizar alternativas para envío inalámbrico de imágenes.
- Implementar un sistema de comunicación para transmisión inalámbrica.
- Implementar un sistema de captura de imagen en pantalla, compresión y envío inalámbrico de imágenes funcionando en conjunto.
- Entregar una **propuesta de solución** para ser implementada en el equipo considerando posibles problemas y limitaciones.

1.3. Metodología

El trabajo se realiza ocupando los ambientes de desarrollo de Altera entregados en su software «Quartus Prime» [16]. Este entorno contiene las herramientas necesarias para pro-

gramar a nivel de hardware, software y hacer que los desarrollos conversen entre sí.

El envío de imágenes se trabajará a partir de una plataforma de desarrollo **DE-10 Nano** [13] en conjunto con una tarjeta de expansión **RFS daughter card** [32]. Estas plataformas constituyen los elementos básicos necesarios para realizar un prototipo funcional de envío de imágenes inalámbricas.

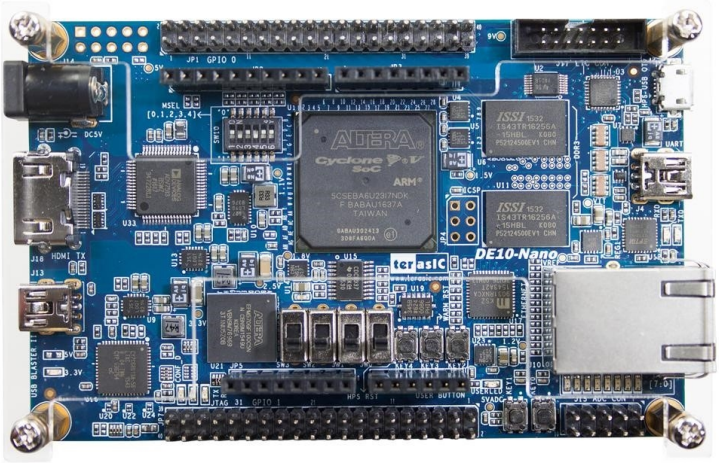


Figura 1.2: Plataforma Altera DE-10 Nano.

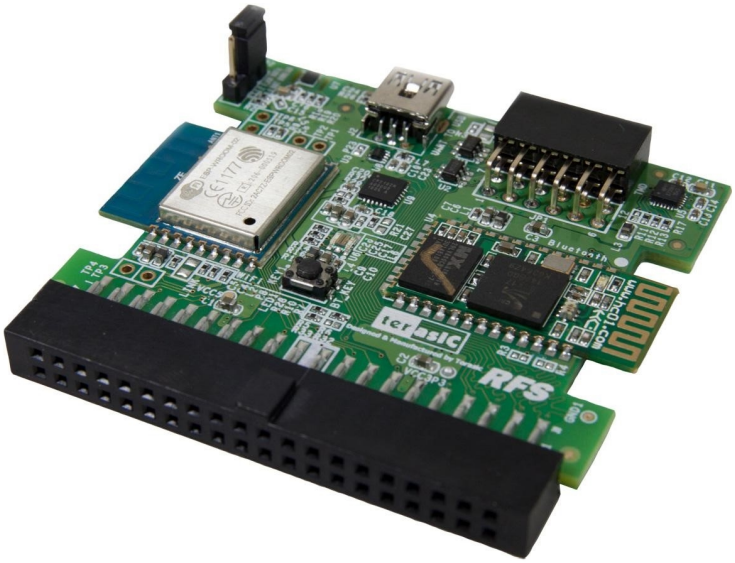


Figura 1.3: Tarjeta de expansión RFS Daughter Card.

La plataforma **DE-10 Nano** vista en la Figura 1.2 tiene como principales características:

- Chip Cyclone V (FPGA + SoC).
- Salida HDMI.
- Distintos pines para expansión y señales I/O.
- Ranura para tarjeta microSD.

- Capacidad de arranque con Linux.

La tarjeta de expansión **RFS Daughter Card** vista en la Figura 1.3 tiene como principales características:

- Comunicación Wifi 2.4 GHz.
- Comunicación Bluetooth 2.0.
- Acelerómetro, giroscopio y magnetómetro.
- Sensor de luz.
- Sensor de humedad y temperatura.

El trabajo se divide en 3 módulos principales. Uno es el hacer uso de los recursos de memoria disponibles en el equipo para proyectar y almacenar un **frame** de video. Para esto puede usarse las **memorias DDR3** disponibles en la plataforma tanto como las memorias **Block RAM** internas del chip **FPGA**.

La segunda parte del trabajo consiste en tomar imágenes en mapa de bits y efectuar un algoritmo de compresión para luego transmitir estos datos por **Bluetooth o WiFi**. Se estudian algoritmos de compresión tanto para envío de imagen única como para secuencias de video. La compresión debe utilizarse en el contexto médico del equipo como herramienta para diagnóstico. En este sentido se realiza un estudio de soluciones aplicables a este caso particular, dado que una excesiva compresión de las imágenes puede entorpecer la labor médica.

Se utilizan imágenes de prueba iniciales pre-cargadas en la plataforma de desarrollo desde las cuales se hace el trabajo de compresión y transmisión como punto inicial. Las soluciones que sean adecuadas para el contexto del proyecto serán implementadas en las plataformas de desarrollo mostradas en las Figuras 1.2 y 1.3.

Teniendo estos dos módulos con implementaciones satisfactorias se procede al punto 3 en donde se diseña un prototipo con los módulos funcionando en simultáneo. Un esquema de esto puede verse en la Figura 1.4 A partir de estos resultados y de la información recabada se generará una propuesta de solución para ser presentada al proyecto del **Ecógrafo Taote**.

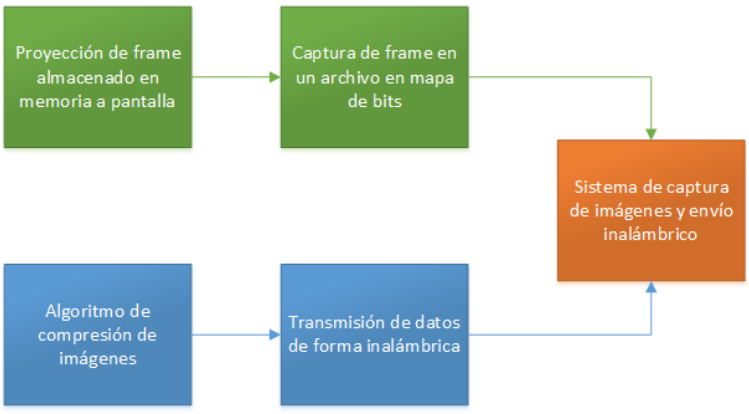


Figura 1.4: Esquema de trabajo simplificado.

1.4. Estructura de la Memoria

El presente trabajo de memoria se ha organizado de forma que mejore la narrativa del lector. Los trabajos realizados no se implementan necesariamente en el orden que son presentados si no que buscan generar una estructura adecuada para que la lectura sea más cómoda.

En el Capítulo 2 se realiza una revisión de los distintos métodos actuales para realizar transferencia inalámbrica de imagen y video digital. Se realiza además un rápida revisión al funcionamiento interno del hardware de **Taote** donde se observa el uso de chips FPGA como núcleo del sistema. El Capítulo se cierra con un alcance sobre consideraciones de integridad de señales y restricciones de frecuencia presentes en los circuitos digitales y en particular en el uso de FPGA.

Las implementaciones realizadas se detallan en el Capítulo 3. Los temas están tratados de forma incremental, y se inicia con una revisión a la plataforma de desarrollo **DE10 - Nano** y los módulos de comunicación inalámbrica a utilizar. Para la transacción de datos se trabaja con una implementación de protocolo serial en FPGA que será la base de las comunicaciones utilizadas. La proyección de video, captura y compresión de imágenes también son tratadas en este capítulo que propone un sistema escalable en base al uso de un búfer en memoria. Este capítulo se cierra con los sistemas de captura de imagen y transmisión funcionando en conjunto.

El Capítulo 4 detalla pruebas realizadas al sistema. Se analiza la cadena de módulos utilizados en la transmisión inalámbrica para encontrar el límite de la tasa de datos del sistema. En la arquitectura propuesta, las proyecciones a pantalla están a cargo de rutinas de software por lo que se analiza los recursos de procesamiento utilizados que permiten evaluar cuan práctico resulta este sistema para crear interfaces fluidas. Se agrega además algunas diferencias observadas en el laboratorio de rendimiento de módulos Bluetooth pertenecientes al mismo modelo.

La propuesta final presentada al equipo de **Taote** se expone en el Capítulo 5. Acá se propone y argumenta según lo expuesto en los capítulos anteriores una actualización al sistema que permita dotar de mayores recursos a **Taote** pero que continua con la misma línea de trabajo y tecnologías usadas en la actualidad.

Capítulo 2

Antecedentes

A continuación se realiza una recopilación de antecedentes recabados en pos de orientar el trabajo a presentar de acuerdo a las tecnologías usadas actualmente. Como se ha mencionado se guarda cuidado en obtener información de cuan efectivas han sido distintos métodos al aplicarse a contextos de diagnóstico médico.

Además se presenta un esquema de funcionamiento interno del **equipo Taote**. Esto debe ser considerado pues la solución que se presenta considera los recursos actuales insuficientes, pero expone una propuesta acorde al marco tecnológico del equipo.

2.1. Transmisión Inalámbrica de Imágenes

La transmisión inalámbrica de imagen se trabaja en opciones tanto de **video** en tiempo real como en **envío único de imagen**. Para la opción de envío de **video** se considera conexiones inalámbricas en **2.4GHz, 5GHz y 60GHz** de acuerdo a los estándares de conexión inalámbrica 802.11n, 802.11ac y 802.11ad. Por otro lado para la opción de **envío único de imagen** se maneja principalmente la opción del uso de tecnología **Bluetooth** dada su cercanía al mundo portátil y móvil al que se apunta el uso del producto.

En el pasado se han implementado soluciones analógicas [29] por video compuesto **NTSC** ocupando la banda libre de 5.8GHz. Esta solución no ha funcionado de forma satisfactoria en el equipo **Taote** por cuanto está muy sujeta a ruido tanto externo como del mismo equipo. Dado estas dificultades se busca soluciones en el mundo digital con mayor resiliencia al ruido electromagnético.

2.1.1. Imagen Única

Compresión de Imagen

Para el envío de Imagen Única se trabaja desde una publicación científica que considera la compresión de un **frame** de video usando codificación **JPEG** para el envío de imágenes médicas de ecografía [28]. Este formato es un método de compresión con pérdidas y en esta publicación se analiza el efecto de distintos niveles de compresión para saber cuanto es el máximo aceptable para los propósitos de diagnóstico médico requeridos.

La compresión **JPEG** es un algoritmo con estándar ISO desarrollado por el grupo «Joint Photographic Experts Group» en los años 80. Tiene gran aceptación en aparatos de telemedicina y en particular en los llamados sistemas PACS (Picture archiving and communication system) dado que la reducción en el tamaño original de datos permite almacenar una gran cantidad de información. Este algoritmo basa su uso en la transformada DCT (Discrete Cosine Transform) y separa la imagen en bloques de 8 x 8 pixeles. Se realiza de este modo un análisis espectral de los bloques que permiten eliminar las frecuencias con menor impacto en el bloque analizado. El funcionamiento en detalle de este algoritmo se puede observar en [33] y un esquema general se ve en la Figura 2.1.

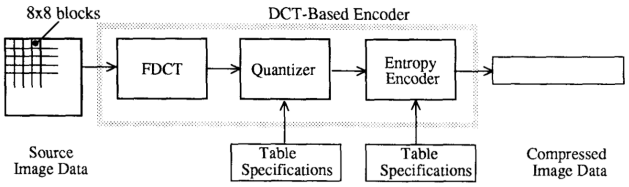


Figura 2.1: Bloques de compresión para JPEG. Fuente: [33].

En la publicación [28], se evalúa el impacto que tiene la pérdida de información dada por la compresión **JPEG** para el diagnóstico médico en una evaluación radiólogos que analizan las imágenes de ecografía comprimidas para tener una visión de ojo experto los cuales evaluan el rendimiento obtenido para distintos grados de compresión utilizados. Las conclusiones obtenidas son las siguientes:

- Compresión **JPEG** a escala 10:1 se consideró de **buena calidad** para diagnóstico médico. En efecto se prefirió la imagen comprimida que la original, probablemente debido a la remoción de ruido.
- Compresión **JPEG** a escala 12:1 a 15:1 se consideró de **buena calidad** para diagnóstico médico. Sin embargo en este caso se prefiere la imagen original sobre la comprimida
- Compresión **JPEG** a escalas mayores de 15:1 se consideró de **mala calidad** para diagnóstico médico.

La compresión **JPEG** preferida de 10:1 corresponde a un *factor de calidad* de aproximadamente $70 \sim 75\,\%$. La mejora en calidad de imagen debido a la reducción de ruido debe considerarse como un efecto positivo para los propósitos del equipo.

Una implementación de compresión de imágenes con JPEG y transmisión inalámbrica puede verse en [21]. El trabajo se implementa en plataformas FPGA usando Bluetooth 2.0 en base a un perfil llamado SPP que permite realizar transacciones sencillas por protocolo UART. Si bien la implementación no está orientada a imagen médica, sirve de referencia los recursos en términos de compuertas lógicas utilizados y como orientación para el trabajo a presentar.

2.1.2. Envío de Video

Se han estudiado distintos métodos de codificación de video para su uso en el proyecto. Las codificaciones de video analizadas corresponden a los estándares **MPEG2, MPEG4, H.263, H.264 y HEVC** estudiados en el ámbito de la ecografía en los trabajos [26], [25] y [27].

En particular se ha considerado una extensión de la codificación **H.264** según el trabajo visto en [26]. Se busca tener una buena protección a los errores en el canal de transmisión, mientras que al mismo tiempo se tiene la consideración del **ancho de banda** disponible tanto como **condiciones variables** (retrasos, perturbaciones por desfase, pérdida de paquetes, corrupción de datos, etc). Para lograr esto se tuvieron consideraciones especiales como el **área de interés** o **ROIs**. De este modo se reutiliza el códec **H.264** con una alta compresión en áreas estáticas de la pantalla o que no aporten al diagnóstico médico, mientras que se mantiene flexible para áreas sensibles donde se necesite mayor detalle.

En los últimos años se ha ocupado un nuevo códec llamado **High Efficiency Video Coding** o **HVEC** por parte del mismo autor [27]. Resulta ser un códec que reduce de forma significativa el flujo de datos en comparación al códec **H.264** antes señalado y frente a otros más antiguos como **MPEG2** y **MPEG4**. El códec promete ser un importante avance por tanto además de la eficiencia en ancho de banda mantiene una capacidad de transmitir video a altas resoluciones. De este modo la capacidad de entregar imágenes aptas para el diagnóstico médico tienen muy buenas chances.

Una comparativa de rendimiento de códecs para una ventana de 560 x 416 puede verse en la Figura 2.2. Acá se utiliza la métrica de **PSNR** (Peak signal-to-noise ratio) como una medida de calidad de imagen y según la publicación [26] valores mayores a 30,5 dB han reportado otorgar la calidad apta para diagnóstico médico.

El ruido **«speckle»**, es un ruido granuloso natural que ocurre en las imágenes de ecografía como parte del proceso de toma de muestras que ha sido estudiado ampliamente en [22]. Aplicando un filtro de **«despeckle»** previo a la utilización del códec HEVC se han obtenido aún mejores resultados por cuanto la reducción de este efecto disminuye las perturbaciones de altas frecuencias en la imagen.

Una comparativa de la relevancia de aplicar el filtro despeckle se ve en la Tabla 2.1, donde se aprecia como la reducción en la tasa de transmisión requerida tiene un impacto significativo. A modo de resumen se puede señalar que el códec HEVC presenta una eficiencia hasta 33,2 % mejor en comparación al códec H.264, y la combinación de HVEC + Filtro Despeckle hasta

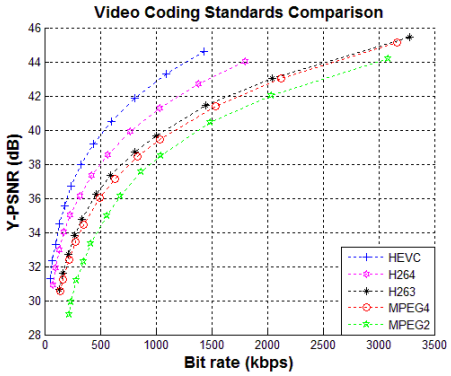


Figura 2.2: Comparación de eficiencia de códecs de video. Fuente: [27].

Tipo de Filtro Despeckle	Ahorro en transmisión de datos	
	vs H264 Original	**vs HEVC Original**
DsFlsmv	39.2 %	43.6 %
DsFhmedian	32.5 %	34.1 %
DsFsrad	23.4 %	23.5 %

Tabla 2.1: Códecs de video al aplicar filtro despeckle. Fuente: [27].

un 39,2 %.

Cabe tener en consideración que el mismo autor señala en [27] que se deben hacer estudios más exhaustivos para analizar el rendimiento del códec HEVC en transmisiones inalámbricas para evaluar la recuperación frente a caídas del canal y la resiliencia general al error de datos propios de las transmisiones inalámbricas. Sin embargo, los resultados resultan prometedores en especial considerando el efecto de la aplicación de un filtro despeckle en la reducción de las tasas de transmisión.

2.2. Hardware Taote

En esta sección se realiza un breve análisis al funcionamiento interno del ecógrafo ultra-portátil **Taote**. Se pone especial énfasis a la tecnología utilizada y a los recursos existentes en el equipo, de modo que las soluciones propuestas continúen en esta misma línea de desarrollo.

Taote basa su arquitectura en el uso de chips «FPGAs» (Field Programable Gate Array), los cuales contienen un arreglo de elementos lógicos programables por lenguajes de descripción de Hardware (en adelante **HDL**, por Hardware Description Language). Algunos lenguajes **HDL** de amplio uso en la actualidad son *Verilog, VDHL y SystemVerilog*. Este tipo de arquitectura contiene varias ventajas entre las cuales se puede mencionar la mejora en el **«time to market»** del producto, la eficacia para procesar datos en paralelo y la capacidad de introducir cambios/mejoras al equipo luego de tener los circuitos ya impresos en **«PCB»**.

Un estudio completo se puede observar en [20], donde se menciona el impacto que ha tenido

esta tecnología desde los años 90 para el rápido desarrollo de nuevos sistemas de hardware. Se expone además la arquitectura interna que componen los chips «**FPGA**» y las etapas de diseño típicas para este tipo de tecnología.

En particular **Taote** está compuesto de dos tarjetas electrónicas principales etiquetadas como **Front-End** (en adelante FE) y **Central Logic Unit** (en adelante CLU). Cada tarjeta contiene sus propios chips **FPGA** que interactúan con distintos periféricos y se comunican entre sí por 4 conectores: PBUS, DBUS, IBUS1, IBUS2. Esto se observa en la Figura 2.3.

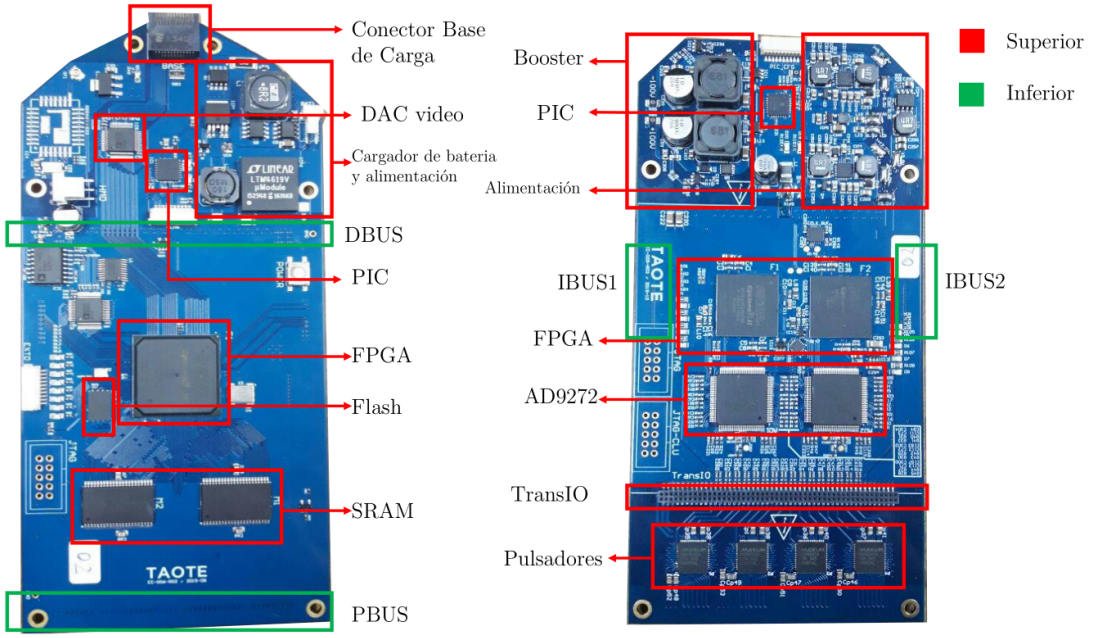


Figura 2.3: Hardware interno de Taote. CLU (izquierda) y FE (derecha). Fuente: [24].

2.2.1. Front-End (FE)

La función principal de esta tarjeta es controlar las señales de RF necesarias para hacer funcionar el transductor y digitalizar los datos recibidos. Un esquema de funcionamiento similar para el FE se puede observar en el trabajo presentado en [1].

De este modo el trabajo del FE consiste en gestionar los disparos de los piezoeléctricos del transductor con los retardos correspondientes, realizar la conversión **ADC**, tratar las señales (filtrado, demodulación y diezmado) y realizar la conformación final de los datos. Un esquema de esto puede verse en la Figura 2.4[1].

Los componentes principales que participan en este proceso son:

- **Cyclone II x2** Chip FPGA principal que controla el funcionamiento del resto de periféricos en el FE.

[1]En el equipo los procesos no ocurren en el mismo orden que los mostrados en el diagrama, especialmente

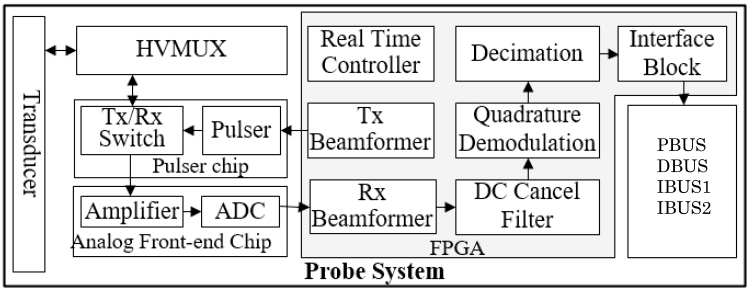


Figura 2.4: Esquema de funcionamiento para un FE.

- **Transductor** Dispositivo convexo de 80 elementos piezoeléctricos con banda de frecuencia centrada en 3.5[MHz].
- **AD9272 x2** Conversor ADC de 8 canales para conversión de datos del transductor.
- **Booster** Conversor DC-DC para elevación de voltaje requerido para el funcionamiento del transductor.

Este sistema genera una ventana de datos (8-bits) en memoria de 80 x 2000 que se almacenan en las memorias **SRAM** de la CLU vistas en la Figura 2.3. Esta ventana es llamada **«Mapa de Muestras»** y es utilizada en etapas posteriores para la generación de imagen en «Modo B». Un esquema simplificado del camino de datos puede verse en la Figura 2.5.

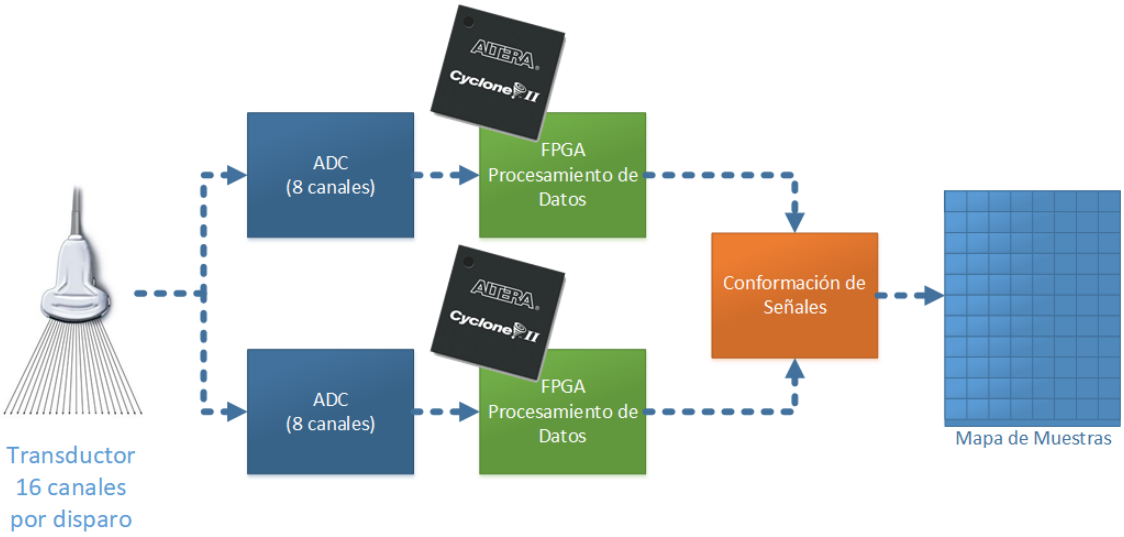


Figura 2.5: Camino para adquisición de datos del transductor.

2.2.2. Central Logic Unit (CLU)

La función principal de esta tarjeta es controlar la interacción con otros procesos propios de un equipo de ecografía como son la proyección de video y la interacción con el usuario. El proceso se inicia con la recepción de datos del «mapa de muestras» entregado por el FE y

su almacenamiento en las memorias SRAM observadas en la Figura 2.3. Un esquema básico del tratamiento que reciben las señales entrantes puede observarse en la Figura 2.6.

En particular la CLU está encargada de la conformación de la «Imagen en modo B», las rutinas de encendido del equipo, inicio y guardado de parámetros, manejo de la interfaz de usuario, proyección de video, e interacción con el sistema de botoneras. Si bien la carga de este chip es considerable en términos de responsabilidades dentro del equipo, el único sistema de alta demanda corresponde a la conformación de «Imagen en Modo B».

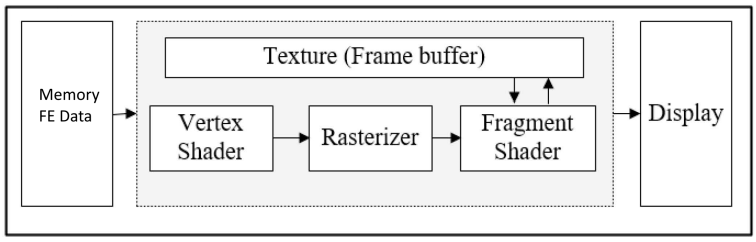


Figura 2.6: Esquema de funcionamiento CLU.

Los componentes principales de este sistema son:

- **Cyclone IV** Chip FPGA principal que controla la interacción con el resto de periféricos de la CLU.
- **ADV7123 x2** Chip para conversión de DAC de video para las salidas VGA y HMD del equipo.
- **Memorias SRAM** Dos chip de memoria de 256KB cada uno para almacenamiento de «Mapa de Muestras» proveniente del FE.
- **Sistema de Botoneras** Mecanismo para navegación por la interfaz del equipo por parte del usuario final.

Dado que la CLU conforma la interfaz final tanto física como virtual del equipo, es el sistema que está sujeto a una mayor cantidad de modificaciones. En este sentido, la CLU es el sistema que mayormente se beneficia de la capacidad de re-configuración de los chips FPGA. En la Figura 2.7 se observa un diagrama simplificado de periféricos que interactúan en el sistema de la CLU.

en lo que concierne a la conformación de la señal. Sin embargo, sigue siendo una buena referencia.

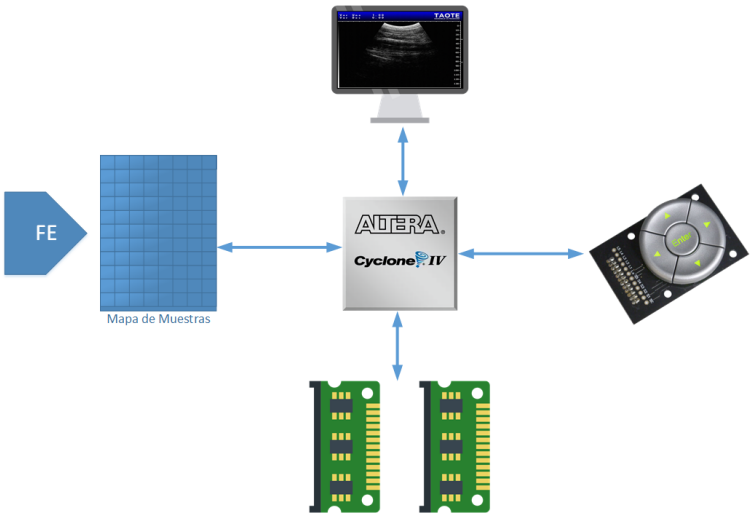


Figura 2.7: Interacción de periféricos en la CLU.

2.3. Integridad de señales en FPGA

Parte importante del diseño en FPGA corresponde a la verificación de restricciones de los circuitos digitales diseñados dada la naturaleza de la propagación de señales requeridas (Ver Figura 2.8). Esto se relaciona al estudio de caminos críticos y restricciones físicas de frecuencia. Fallas de señales entre los módulos diseñados entrega cálculos erróneos o en el peor de los casos una falla general del sistema. Se busca entonces garantizar cierto grado de estabilidad al funcionamiento del sistema en general.

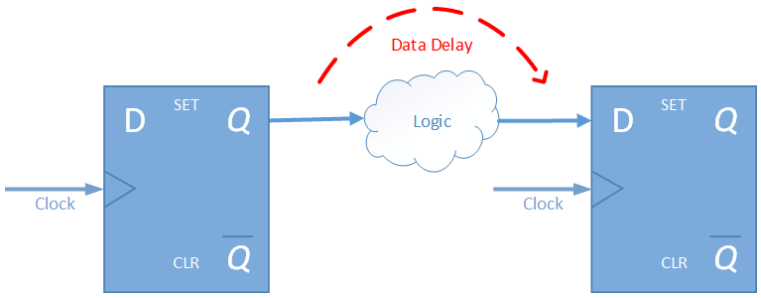


Figura 2.8: Esquema de propagación de señales digitales.

Este proceso se conoce como **verificación temporal** o **timing verification**. Existe documentación extensa dentro de las herramientas de Altera para realizar los análisis necesarios a los chips FPGA que ellos fabrican. En específico resulta útil revisar los Capítulos de **Timing Analysis Overview** y **The Quartus II TimeQuest Timing Analyzer** vistos en [7].

En particular las fallas de estabilidad en los sistemas tienen origen en los fenómenos de *Metaestabilidad* y la falta de *Verificación de Requerimientos Temporales* en los chips FPGA.

2.3.1. Metaestabilidad

La *Metaestabilidad* es un fenómeno que puede causar fallas en dispositivos digitales, cuando una señal pasa entre circuitos con señales asincrónicas o pertenecen a dominios de reloj no relacionados [3].

Altera posee una documentación extensa y otorga el software necesario vía **TimeQuest Timing Analyzer** para analizar el hardware sintetizado a partir de la programación en lenguajes **HDL**. Este software calcula un tiempo medio esperado para los problemas de metaestabilidad detectados llamado **MTBF**, «mean time between failures».

En proyectos grandes, los problemas de metaestabilidad son inevitables. Sin embargo, existen mecanismos de diseño que permiten extender el **MTBF** a valores tan altos como sea necesario para el tipo de aplicación crítica que tenga el producto.

En circuitos digitales secuenciales debe tenerse especial cuidado con mantener los requerimientos de t_{setup} [2] y de t_{hold} [3] para los registros. Cuando una de estas señales viola una de estas restricciones puede entrar en un estado **metaestable**. Puede verse un ejemplo ilustrado de esto en la Figura 2.9, con una bola que debe pasar un montículo. Acá una señal que viola alguno de los requerimientos mencionados puede dejar a un registro en un estado ambiguo respecto a su valor anterior.

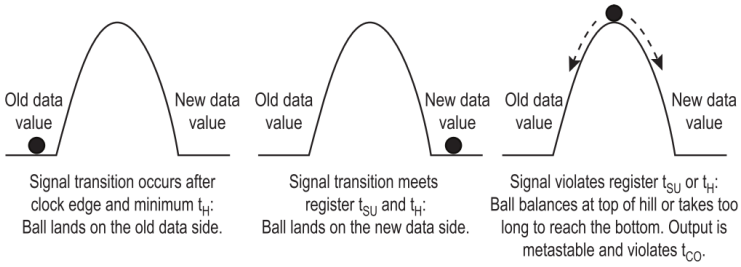


Figura 2.9: Ejemplo de metaestabilidad. Fuente: [3].

De este modo una señal con problemas de metaestabilidad puede resolver a ambos estados, tal como se ve en la Figura 2.10. Si la señal de salida resuelve a un valor válido antes del próximo flanco de reloj, entonces no hay un impacto negativo para el funcionamiento del sistema. En cambio, cuando una señal metaestable no resuelve a un valor dentro de un tiempo determinado queda en estados inconsistentes para el resto del sistema que puede capturar valores distintos para una misma señal o propagar la metaestabilidad a otros sectores del circuito.

Estado Actual de Soluciones

El problema de calcular **MTBF** está cubierto vía software por **TimeQuest Timing Analyzer**. Este realiza un análisis del circuito digital en **HDL** y como se aplica físicamente

[2] Tiempo necesario para una señal ser tomada como válida antes de un flanco de reloj.
[3] Tiempo necesario para una señal ser tomada como válida luego de un flanco de reloj.

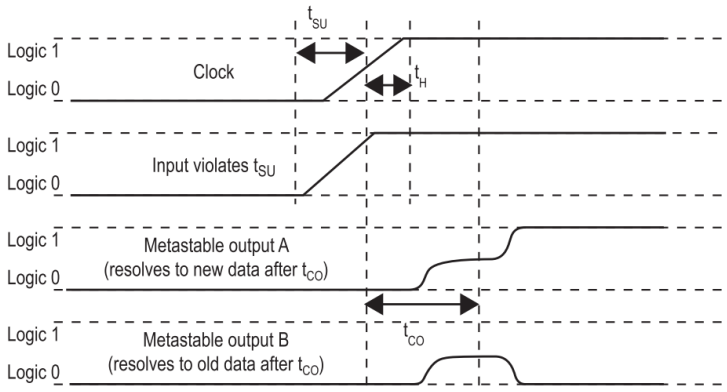


Figura 2.10: Señales dispares por metaestabilidad. Fuente: [3].

en el chip para realizar simulaciones y hacer lo cálculos.

Cuando se detecta un valor de **MTBF** demasiado bajo para los requerimientos del producto se deben analizar los reportes generados por **TimeQuest** para rastrear el origen del problema. Cuando se detecta la señal de falla se implementa una **Cadena de Sincronización** o **Synchronization Chain**. Esto es una secuencia de registros que cumplen las siguientes condiciones:

- Todos los registros en la cadena deben pertenecer al mismo dominio de reloj.
- El primer registro en la cadena tiene como entrada una señal asíncrona o pertenece a un dominio de reloj no relacionado.
- Cada registro en la cadena alimenta solo un registro, excepto el último registro de la cadena que es la señal de salida.

Esta solución puede verse implementada con flip-flops en la Figura 2.11. La **Cadena de Sincronización** puede ser de tantos elementos como sea necesario aumentar el valor de **MTBF**.

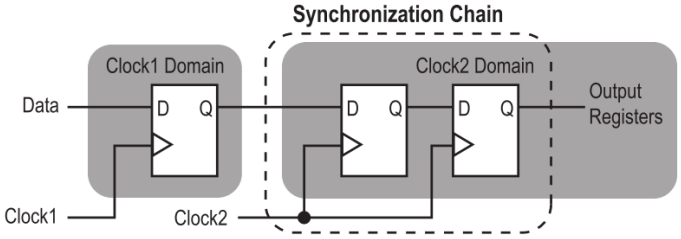


Figura 2.11: Cadena de Sincronización de dos elementos. Fuente: [3].

2.3.2. Restricciones de Frecuencia

En la Sección 2.3.1 se hace referencia a fallas en los circuitos digitales cuando se trabaja con señales asincrónicas o pertenecientes a distintos dominios de reloj. Sin embargo, para

un circuito digital existen **restricciones temporales** o **timing contrains** para un mismo dominio de tiempo dado por los tiempos de propagación de los elementos lógicos ocupados. Estas restricciones definen las frecuencias máximas soportadas por el circuito.

Para medir los tiempos de propagación dentro de la **FPGA**, se trabaja con **TimeQuest Timing Analyzer** (Ver Figura 2.12) que integra un simulador de circuitos de **FPGA** propietario de Altera. Los **modelos temporales** o **Timing Models** que permiten obtener las restricciones del circuito se mencionan con detalle en [4]. Estos modelos deben considerar ciertas condiciones de incertidumbre para el funcionamiento del circuito, entre las que se encuentran principalmente las *condiciones de operación* y *diferencias en el silicio de los chips*.

Las diferencias en el silicio ocurren tanto por los procesos naturales de manufactura como por desgaste. En los modelos utilizados por **TimeQuest** se ocupan rangos de frecuencias que dependen del **speed grade** definido por el fabricante dependiendo del modelo del chip FPGA que cubren estas diferencias de rendimiento. En específico el software realiza análisis de las restricciones para las siguientes condiciones de operación:

- Slow 85C Timing Model.
- Slow 0C Timing Model.
- Fast 0C Timing Model.

Siendo la condición óptima de operación el modelo de **Fast 0C Timing Model** y el peor caso de operación se da en **Slow 85C Timing Model**. Acá «Fast» y «Slow» hacen referencia a la frecuencia esperada de funcionamiento máxima y mínima respectivamente, dentro del rango de frecuencias entregado por su **speed grade**.

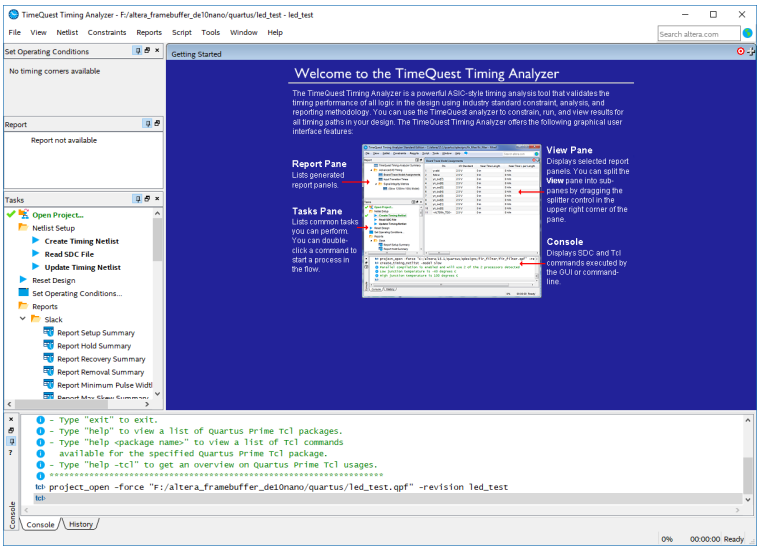


Figura 2.12: Interfaz gráfica de TimeQuest Timing Analyzer.

Reportes Generados

El software de **TimeQuest** genera distintos reportes que permiten analizar el cumplimiento de las restricciones de frecuencia del equipo. Si se obtiene que los **timing constrains** se cumplen, la **FPGA** no tendrá problemas de propagación y estará funcionando en condiciones de operación válidas.

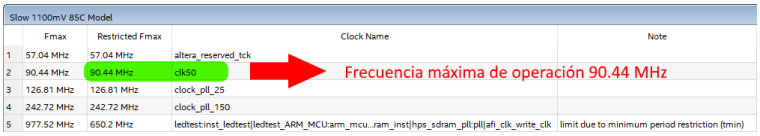


Figura 2.13: Reporte de frecuencias máximas.

En caso contrario, se deben analizar los reportes entregados y rastrear los módulos con problemas. Las soluciones pueden pasar por cambiar la lógica que define el funcionamiento de los módulos para ajustarlos a sus tiempos de propagación o cambiar el ruteo interno de las señales de la **FPGA** ocupando los softwares **Design Space Explorer** [8] o **Chip Planner** [2].

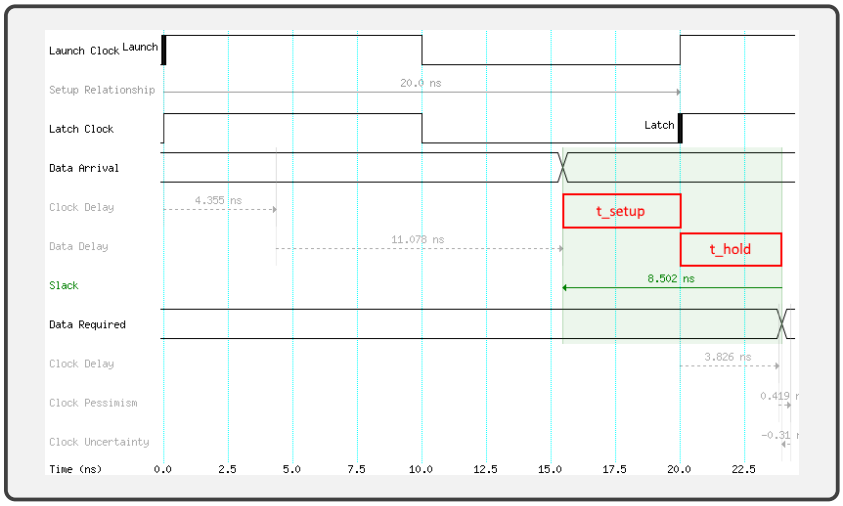


Figura 2.14: Representación gráfica.

En la Figura 2.13 se observa uno de los reporte entregados con la lista de relojes de un proyecto y su frecuencia máxima reportada y en la Figura 2.14 una representación gráfica entregada donde se observa en detalle los tiempos críticos, en particular t_{setup} y t_{hold}. Los cálculos representan el camino crítico más restrictivo, aunque dentro de **TimeQuest** se tiene acceso a una lista completa con la propagación de señales analizadas nodo a nodo.

Capítulo 3

Implementación

El trabajo se divide en pequeñas tareas que permiten avanzar hacía la implementación de un sistema inalámbrico de transmisión de imágenes. Se avanza desde una forma básica de transacción de datos de forma **serial**, pasando por protocolos de transmisión para **Bluetooth** y **WiFi**, para luego realizar una proyección y captura de imágenes a ser transmitidas.

3.1. Contexto y Arquitectura del Sistema

La plataforma de desarollo escogida corresponde a una de las últimas entregas de Terasic, la placa DE10-Nano [1] según se ha visto en el Capítulo 1. Tiene su principal funcionamiento alrededor de un chip híbrido **HPS - FPGA** de nombre Cyclone V SoC, producido por Altera [9], [13]. Un esquema con las conexiones disponibles en la placa se puede observar en la Figura 3.1. Es importante notar que hay periféricos que interactúan de manera directa con el sector de la FPGA y otros con el HPS.

El equipo de **Taote** cuenta con versiones anteriores de esta línea de chips (Cyclone II y IV), por lo que trabajar con los modelos siguientes del mismo fabricante permite construir soluciones coherentes con la electrónica existente en el equipo, al mismo tiempo que proponer actualizaciones al dispositivo.

Una de las principales novedades de este nuevo chip es la integración de un sistema **SoC**, como se ve en la Figura 3.2 Los sistemas **SoC** integran distintos tipos de componentes electrónicos que dan forma a un computador móvil de bajo consumo, de amplio uso en dispositivos móviles, smartphones y sistemas embebidos en general.

El **SoC** está basado en torno a un procesador (HPS) ARM Cortex-A9 capaz de levantar un sistema Linux de bajos recursos. Esto permite explorar el uso de rutinas escritas en software y el acceso a grandes cantidades de memoria DDR3 con el funcionamiento de un sistema basado en **FPGA**.

[1] Terasic DE10-Nano Development Kit. `http://de10-nano.terasic.com`.

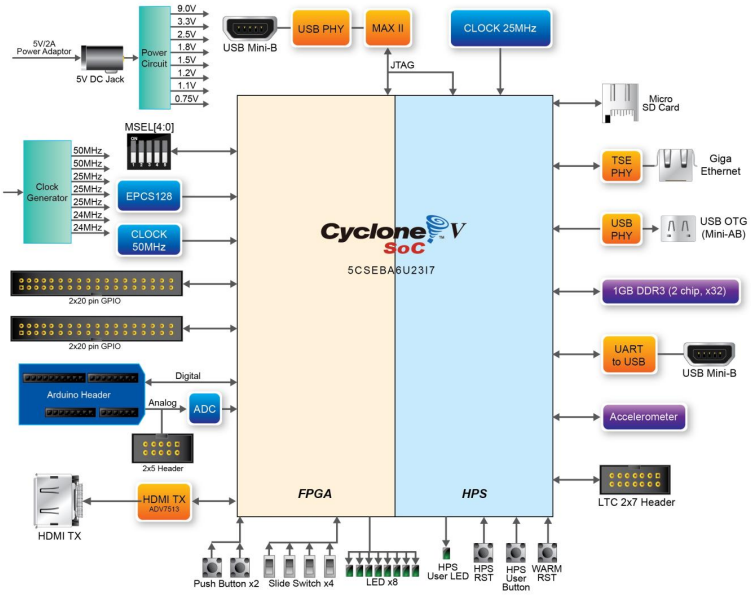


Figura 3.1: Esquema conexiones en la placa de desarollo. Fuente: [13]

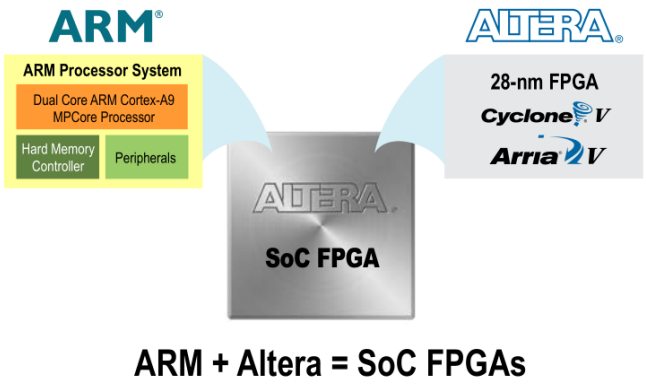


Figura 3.2: Sistema SoC + FPGA. Fuente: [9].

3.1.1. Módulos Bluetooth y WiFi utilizados

Para la comunicación inalámbrica se trabaja con chips de de amplio uso y fácil acceso en el mercado incluidos en la tarjeta de expansión **RFS Daughter Card** . Los modelos corresponden a **HC-05** y **ESP8266EX**, para Bluetooth y Wifi respectivamente. La documentación de funcionamiento de estos módulos es amplia, por cuanto se hicieron populares junto al amplio uso de Arduino en el terreno de prototipos. El empaquetado independiente típico de estos modelos se observa en la Figura 3.3. Un esquema de las interfaces existentes para entablar comunicación con los módulos se observa en la Figura 3.4.

Ambos módulos utilizan comunicación serial como medio principal tanto de configuración interna, como de interfaz para la transacción de datos de emisión o recepción. Esto plantea uno de los primeros focos a abordarse en el desarrollo del trabajo.

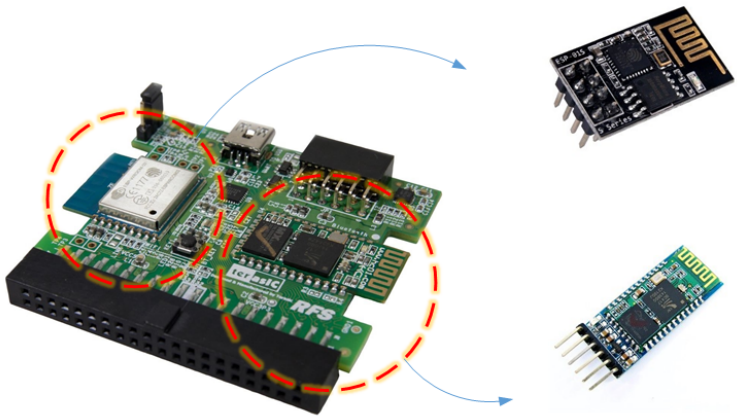


Figura 3.3: Módulos HC-05 y ESP8266EX típicos.

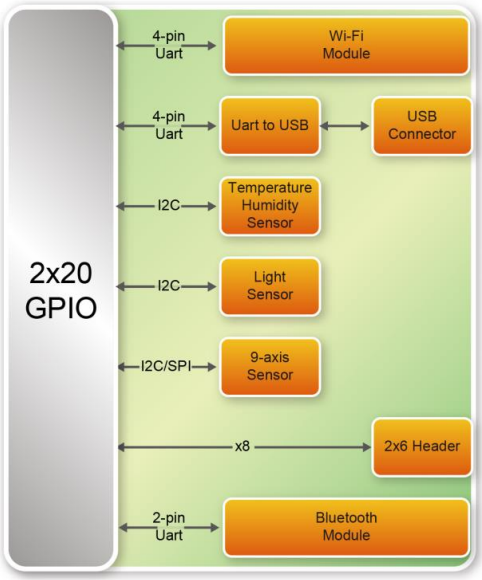


Figura 3.4: Esquema de conexiones RFS Daughter Card. Fuente: [32].

3.2. Comunicación Serial

Se utiliza el protocolo de comunicación serial para la transacción de datos con los módulos vistos en la Sección 3.1.1. Se encuentra como medio de comunicación en diversos dispositivos y periféricos electrónicos, generalmente en paquetes de 8 bits que conforman un carácter [23], [31]. Un esquema de esto se ve en la Figura 3.5.

La transferencia de datos entre dispositivos generalmente consiste de dos canales de datos serializados, uno para transmisión y otro para recepción, según sea el terminal del que se tiene la referencia. Esto se ve en la Figura 3.6.

Según se vio en la Sección 3.1.1, los módulos **Bluetooth** y **WiFi** a utilizar requieren este tipo de protocolo. Dado que la transmisión se realiza con caracteres de 8 bits, se utiliza un

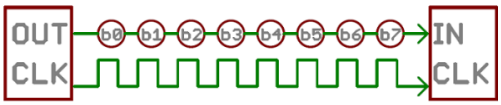


Figura 3.5: Esquema de transmisión serial. Fuente: [31].

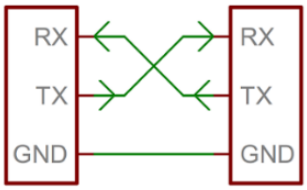


Figura 3.6: Comunicación serial entre dispositivos. Fuente: [31].

módulo UART que convierte datos de 8 bits en paralelo en datos serializados. Esto se ve en la Figura 3.7.

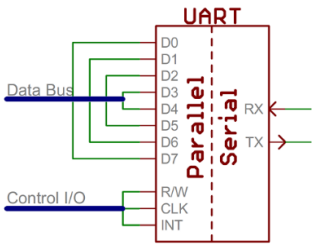


Figura 3.7: Esquema de módulo UART. Fuente: [31].

Para la implementación del módulo UART se manejan distintas opciones, siendo los principales candidatos los **IP Cores** entregados por Altera para sus placas y chips. Entre estos se encuentran los módulos **JTAG UART Core**, **UART Core** y **16550 UART Core**.

3.2.1. UART 16550 IP Core

El módulo escogido para la implementación corresponde al **UART 16550 IP Core** [14]. Este módulo es compatible a nivel de arquitectura con el módulo UART 16550, original de «National Instruments», y que data desde el año 1987 [23]. Al día de hoy este circuito integrado es de uso común y se ha transformado en un estándar en la industria de sistemas embebidos.

Para el caso de estudio se implementa este módulo en la **FPGA** y ocupando los recursos internos de la placa se procede a configurar el circuito digital de forma que el **HPS** tenga acceso al módulo **UART**. Esto se observa en la Figura 3.8.

La recepción y envío de datos desde el punto de vista del **HPS** quedan mapeados a registros de memoria, al igual que la configuración de este módulo. Un ejemplo de esto se puede ver en la Figura 3.9 (aunque para el ejemplo se ha utilizado el módulo **UART IP Core** de implementación más sencilla).

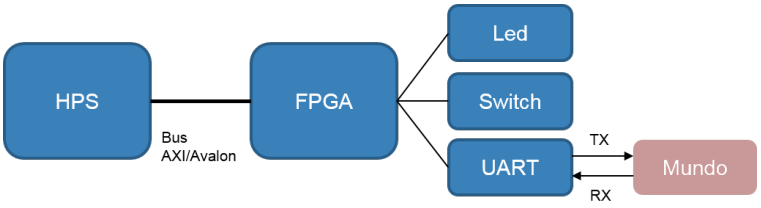


Figura 3.8: Esquema de implementación UART.

Offset	Register Name	R/W	Description/Register Bits													
			15:13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	rxdata	RO	Reserved					(2)	(2)	Receive Data						
1	txdata	WO	Reserved					(2)	(2)	Transmit Data						
2	status (1)	RW	Reserved	eop	cts	dcts		e	rrdy	trdy	tmt	toe	roe	brk	fe	pe
3	control	RW	Reserved	ieop	rts	idcts	trbk	ie	irrdy	itrdy	itmt	itoe	iroe	ibrk	ife	ipe
4	divisor (3)	RW	Baud Rate Divisor													
5	endof-packet (3)	RW	Reserved					(2)	(2)	End-of-Packet Value						

Figura 3.9: Registros mapeados a memoria para un módulo UART. Fuente: [14].

Una de las ventajas principales del módulo **UART 16550 IP** Core es su capacidad para utilizar colas FIFO en la recepción de datos. Esto permite que no se pierdan datos en momentos que el **HPS** se encuentra trabajando en otros procesos del sistema operativo Linux. Gráficamente se puede ver este proceso en la Figura 3.10.

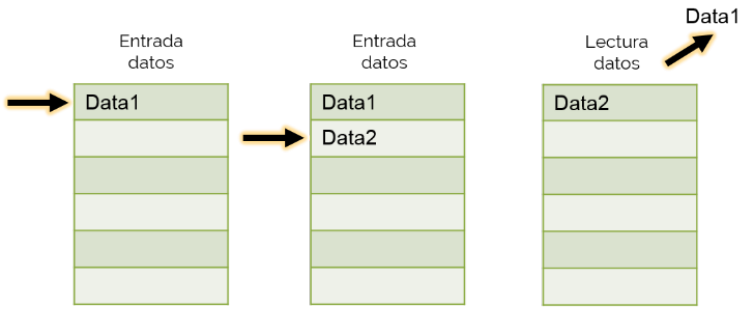


Figura 3.10: Colas FIFO para UART.

La utilización de este módulo se simplifica de forma importante al utilizar la librería «UART Driver API», disponible en la API pública «Altera HWLIB, The Altera HW Manager API Reference Manual». La documentación de esta librería se entrega con cualquier instalación del software **Quartus Prime** [2].

3.3. Comunicación por Cable

El esquema visto en la Sección 3.2 se válida para el envío de imágenes en formato **JPEG** utilizando comunicación cableada. Por un lado se trabaja bajo un ambiente Linux para el

[2]Las nuevas versiones de Quartus Prime instalan un terminal con herramientas de compilación llamado

HPS, y por otro lado se trabaja el envío/recepción como un script de **Matlab**. Esto se observa en la Figura 3.11.

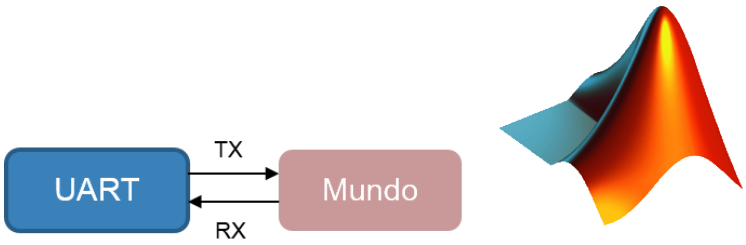


Figura 3.11: Esquema transmisión UART - Matlab por cable.

La recepción de datos se trabaja en primer lugar sin una cola FIFO, lo que genera errores en la transmisión. Como se ha mencionado en la Sección 3.2.1, el error se detecta en cuanto el **HPS** no lee/escribe en el módulo **UART** de forma continua, pues el *kernel* de Linux gestiona distintas operaciones dentro de su sistema y debe atender otros procesos. Un ejemplo de este efecto se ve en la Figura 3.12.

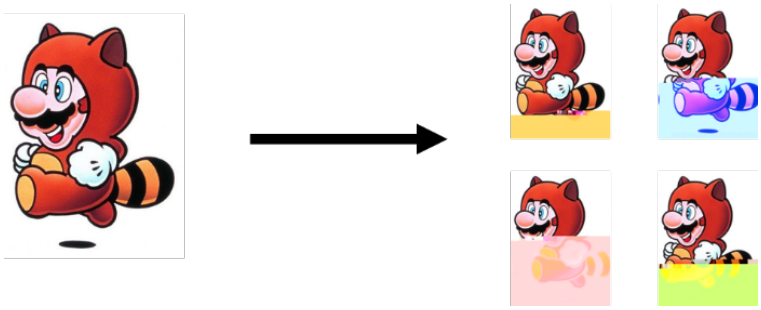


Figura 3.12: Transmisión sin cola FIFO.

3.4. Comunicación Inalámbrica

El esquema visto en la Sección 3.3 utilizado para la conexión cableada puede heredarse para operaciones inalámbricas. Los periféricos **Bluetooth** y **WiFi** manejan ciertos perfiles que permiten crear una capa de abstracción en la comunicación, de modo que los dispositivos manejen una conexión serial del mismo modo que se trabaja de modo cableado. Un ejemplo de este puede verse en la Figura 3.13.

3.4.1. Bluetooth

El módulo **HC-05** utilizado maneja un perfil llamado **Serial Point to Point** (En adelante SPP). Este perfil puede configurarse para que los dispositivos que hacen uso del recurso del

Embedded Command Shell. En mi caso la documentación se encontraba en la siguiente carpeta:

Figura 3.13: Esquema conexión inalámbrica/serial.

canal de **Bluetooth** vean una conexión cableada virtual, de manera transparente. Esto es una propiedad general de los módulos **Bluetooth 2.0** según se ve en [30].

La idea detrás de este puente virtual es reutilizar los mismos programas validados para la transmisión cableada, y heredarlos con cambios mínimos. Un esquema del uso de este sistema puede verse en la Figura 3.14.

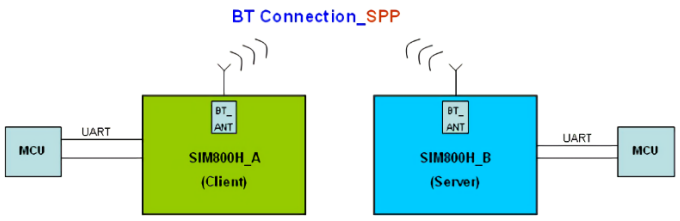


Figura 3.14: Perfil SPP en Bluetooth. Fuente: Documentación de *SIM808*.

Este sistema otorga simpleza en su uso, y la capacidad de transmisión del canal depende principalmente de la calidad del chip utilizado y el ruido electromagnético en la banda de 2.4GHz. Las velocidades para transmisiones correctas de las primeras imágenes de prueba corresponden a 115200bps, limitados principalmente por la calidad del chip.

3.4.2. WiFi

La idea detrás del uso del módulo **ESP8266EX**, es similar a la vista en la Sección 3.4.1. Se establece una conexión cableada virtual haciendo uso de puertos de comunicación TCP + WiFi, bajo un perfil llamado **CIPMODE** [18].

Los pasos a seguir para el envío de información a través de WiFi se resumen a continuación:

- Configuración de un módulo como Access Point (AP), y otro como cliente.
- Abrir un puerto de comunicación TCP o UDP entre ambos dispositivos.
- Entrar en el modo **CIPMODE** para envío de datos de forma transparente.

De este modo, el sistema queda como se ve en la Figura 3.15.

`D:\intelFPGA_lite\17.0\embedded\ip\altera\hps\altera_hps\doc\soc_cv_av`

Figura 3.15: Esquema CIPMODE para WiFi.

3.5. Proyección de Video

Se estudian los mecanismos de proyección de video en sistemas embebidos haciendo uso de acceso a memoria. El espacio de memoria asignado para la proyección se conoce como **Framebuffer** [3].

De momento el equipo **Taote** proyecta las imágenes en tiempo real, en sincronización completa con la proyección a pantalla. Dotarlo de un **framebuffer** permite acceder a recursos de compresión de imágenes y codificación de video aptos para la transmisión inalámbrica.

3.5.1. Recursos de Memoria DDR3

La FPGA puede acceder a los recursos de memoria disponibles para el HPS por una interfaz llamada **FPGA-to-HPS SDRAM Interface** propia del chip. De este modo se tiene un acceso a memoria directo (**DMA** en inglés) desprotegido del sistema Linux que se levanta en el HPS. El sector de memoria al que puede acceder la FPGA por esta interfaz se etiqueta como **SDRAM** en la Figura 3.16. Detalles del funcionamiento interno de la interfaz puede verse en el capítulo **SDRAM Controller Subsystem - SDRAM Controller Subsystem Interfaces** visto en [10] junto con documentación en la web [11].

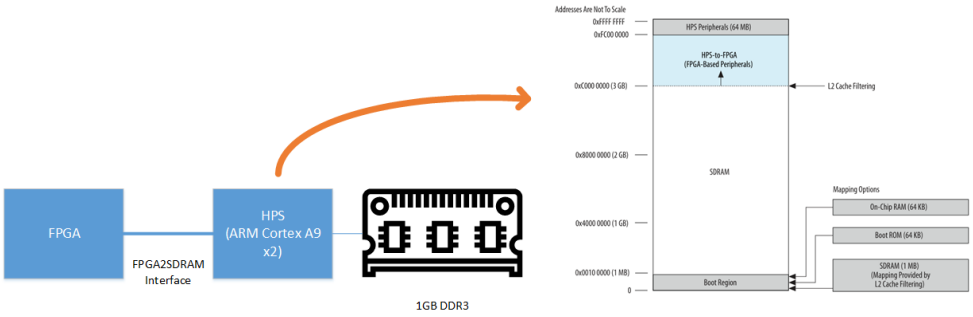


Figura 3.16: Registros principales del HPS.

La FPGA tiene acceso a todo el espacio de memoria por lo que para la integridad de

[3] Una definición más detallada puede verse en:
`https://en.wikipedia.org/wiki/Framebuffer`

funcionamiento del sistema Linux, se modifica en el sector de arranque del sistema espacios delimitados para intercambio de datos con la FPGA y otro para el HPS. De este modo de 1GB disponible se deja 512MB para el sistema operativo, y 512MB para transacción de datos por la FPGA.

3.5.2. Implementación de salida HDMI

Para la implementación de salida de video se ocupan dos elementos principales. En primer lugar se tiene el chip **ADV7513** encargado de recibir señales en formato VGA estándar digital y obtener salida HDMI 1.4. En segundo lugar el módulo **Clocked Video Output II IP Core** [17] se encarga de enviar las señales de video necesarias según el estándar VGA.

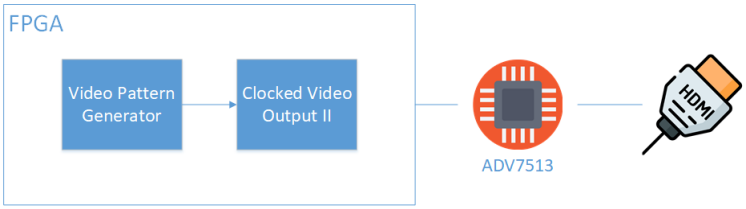


Figura 3.17: Esquema de prueba HDMI.

En esta etapa lo más importante fue validar el funcionamiento de la implementación de salida HDMI independiente del acceso a memoria. Para esto resultó útil apoyarse en el módulo **Test Pattern Generator II IP Core** que genera una secuencia de datos de prueba en tiempo real. Un esquema de esto se ve en la Figura 3.17, y el resultado de las pruebas finales en la Figura 3.18.

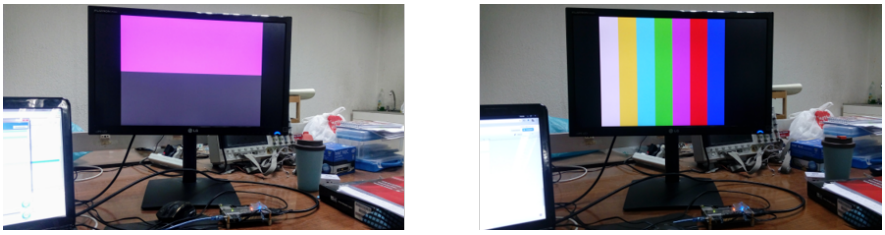


Figura 3.18: Funcionamiento de sistema de video.

3.5.3. Módulo de Framebuffer

Con el sistema de video validado, se reemplaza el generador de patrones con el módulo **Frame Buffer II IP Core** [17]. El esquema de esto se ve en la Figura 3.19.

Esto permite proyectar en pantalla un sector determinado de memoria, donde el tamaño va dado por la cantidad de **frames** (imagen única o fotograma) a almacenar. Para el caso de estudio, solo se almacena 1 **frame** de tamaño 640x480 en formato RGB de 8bits por canal, que ocupa 921,6KB de memoria RAM. El orden en que se almacenan los datos en memoria se observa en la Figura 3.20.

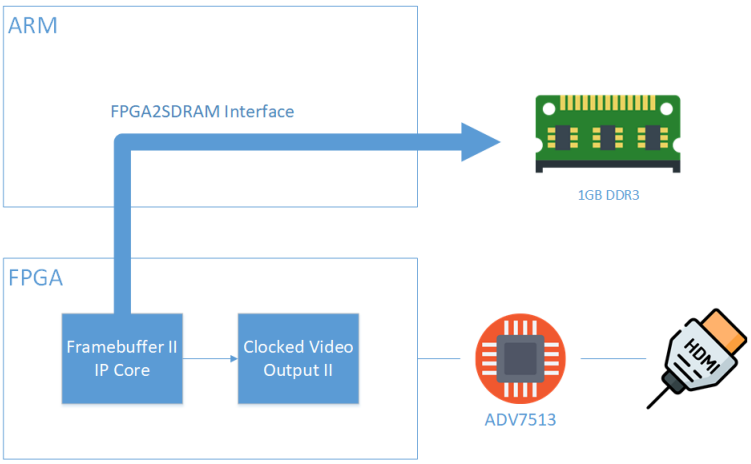


Figura 3.19: Funcionamiento de framebuffer y FPGA2SDRAM.

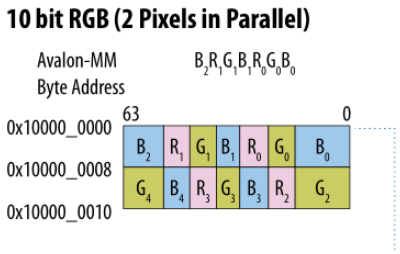


Figura 3.20: Datos de framebuffer en memoria. Fuente: [17].

Las pruebas realizadas a la implementación se pueden observar en la Figura 3.21. La imagen obtenida es esperada pues se accede a un sector de memoria reservado para el **framebuffer**, en donde otros procesos o periféricos no acceden. Por esto se obtienen datos aleatorios en este sector.

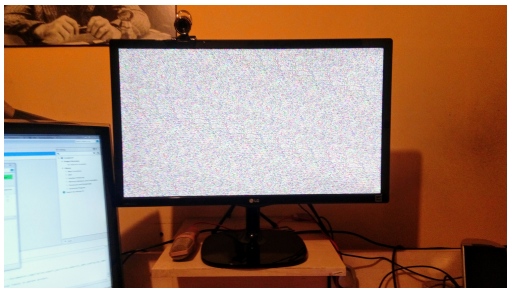


Figura 3.21: Prueba inicial funcionamiento framebuffer.

3.6. Carga y Extracción de Imagen

Los siguientes trabajos corresponden a proyectar imágenes particulares utilizando archivos comprimidos de imagen desde la memoria microSD de la placa de desarrollo. Para esto se

hace uso de compresión de imágenes por software y el mismo sistema de almacenamiento que levanta el sistema Linux.

Dado que la imagen proyectada existe en la memoria RAM, leer las direcciones correspondientes permite extraer una imagen en mapa de bits. Del mismo modo el proceso inverso de escritura, carga una imagen en mapa de bits a la pantalla. De este modo el proceso crucial es el manejo de librerías **JPEG** según lo visto en la Sección 2.1.1.

3.6.1. Librería JPEG

Se utiliza la librería «LIBJPEG» de Independent JPEG Group's JPEG software [4]. Este es un grupo que distribuye librerías de código abierto para el uso de JPEG desde 1991. El lenguaje utilizado es C y no depende de otras librerías.

La principal dificultad al hacer uso de esta librería es realizar un proceso llamado compilación cruzada. Dado que la programación se hace en un PC Windows, y la librería corre sobre una arquitectura ARM distinta, los archivos binarios de la librería no son compatibles entre plataformas de forma directa. Así se realiza el proceso de cross-compiling [5], donde a partir de una plataforma x86 se generan binarios de la librería para ser utilizados en ARM.

Alternativamente, pueden tomarse binarios (y los archivos de enlace o Linker Files) precompilados en la misma plataforma DE10-Nano. Esto simplifica el proceso quedando de la siguiente forma:

- Compilar las librerías JPEG en la plataforma DE10-Nano de forma nativa.
- Tomar los archivos generados y llevarlos al proyecto de software en Windows.
- Realizar la compilación del programa utilizando los binarios generados de las librerías JPEG.

3.6.2. Proyección de Imagen

Se realiza una proyección utilizando una imagen de prueba que se ve en la Figura 3.22. Esta imagen se encuentra pre-cargada en un archivo **.JPG** dentro de la tarjeta microSD que lee el sistema Linux.

Para el uso correcto de la librería **JPEG** se debe tener en cuenta como esta espera ver los datos en memoria. Un esquema de esto se ve en la Figura 23. Esto discrepa con la forma en que están almacenados los datos en el **framebuffer** visto en la Sección 3.5.3 de modo que se debe generar un algoritmo para ajustar los datos.

Los resultados del proceso pueden verse a continuación en la Figura 24. Se observa en las primeras iteraciones como afecta el orden incorrecto de lectura de los datos en la proyección

[4] `http://www.ijg.org/ https://github.com/LuaDist/libjpeg`

[5] `http://www.ridgesolutions.ie/index.php/2013/05/15/cross-compiling-libjpeg-for-linux-on-arm/`

Figura 3.22: Imagen de prueba a utilizar.

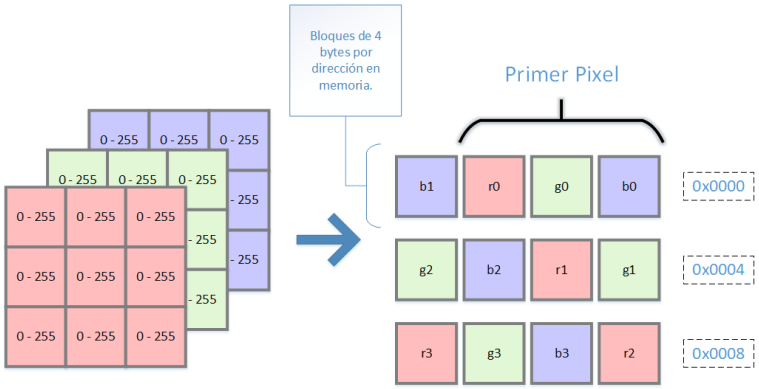


Figura 3.23: Ordenamiento de datos para librería JPEG.

final obtenida. Finalmente, el orden de datos es corregido y se aprecia la imagen correcta.

Figura 3.24: Resultados de carga de imagen.

3.7. Sistema Funcionando en Conjunto

Se tienen dos sistemas principales que se hacen trabajar en conjunto. El sistema de transmisión de datos inalámbrico visto en la Sección 2.1 se hereda para trabajar en paralelo con el sistema de proyección de video y manejo de librerías JPEG para compresión de imágenes vista en la Sección 3.6. Un esquema general de la forma que toma este sistema puede verse en la Figura 3.25, en donde se observa que el proceso de generación de imagen en pantalla funciona de forma independiente al de captura y transmisión de la imagen proyectada.

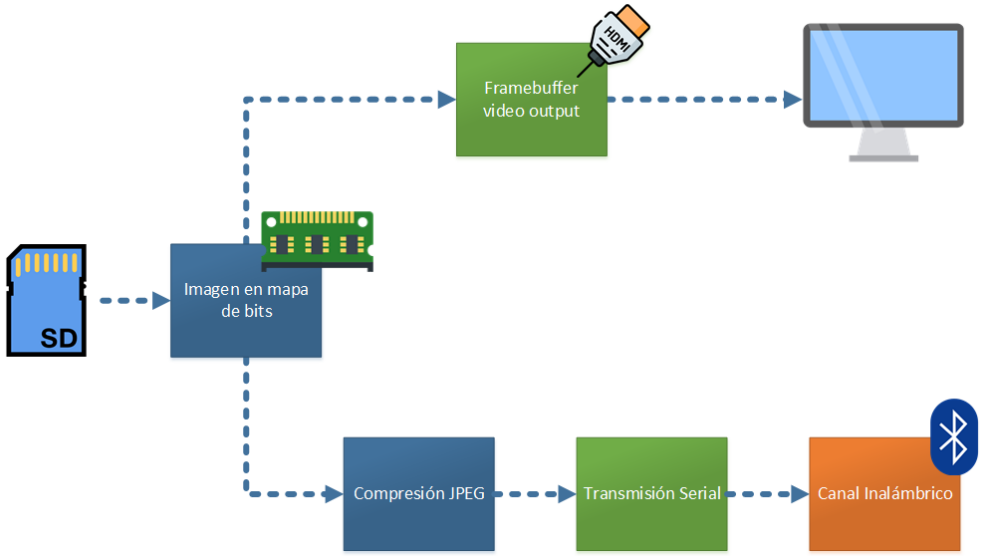


Figura 3.25: Esquema de alto nivel propuesto.

El hardware diseñado en la FPGA a partir de los sistemas vistos debe ser controlado por rutinas de software ejecutadas por el procesador. Para esto los distintos periféricos que tienen interacción física con elementos externos (como lo es el sistema de video o lo módulos de comunicación serial) contienen registros que se ven en el procesador como elementos mapeados a memoria tal como se ha visto en la Sección 2.1 y 3.6. Con esto se construye un programa en C que implementa ambas funcionalidades vistas, ejecutando las siguientes rutinas de software:

- Rutina para levantar el sistema de **Framebuffer**.
- Rutina para cargar una imagen de mapa de bits en pantalla.
- Captura de imagen en pantalla y creación de archivo **JPEG**.
- Envío de bytes que conforman el archivo por comunicación serial.

La tarjeta **RFS Daughter Card** funciona como emisor ocupando el módulo **Bluetooth** en modo **SPP**. Como receptor se ocupa un adaptador **Bluetooth-USB 2.0** genérico visto en la Figura 3.26 que se conecta a un computador corriendo un programa en *Python* para recibir los datos.

Figura 3.26: Adaptador Bluetooth para el receptor.

La activación del módulo **Framebuffer** y la carga de dos imágenes de prueba se reflejan en las capturas de pantalla vistas en la Figura 3.27. Acá se observa en un primer paso la activación del **Framebuffer** y los datos aleatorios almacenados en memoria (junto con un remanente de datos de pruebas anteriores). Se limpia este sector de memoria en el segundo paso y se cargan las imágenes de prueba para el tercer y cuarto paso.

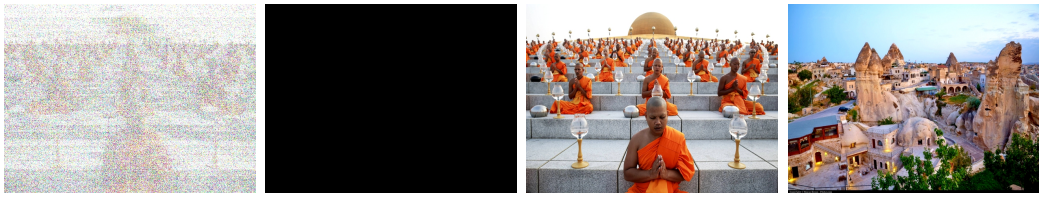


Figura 3.27: Rutina de carga de imágenes a pantalla.

Para cada una de las imágenes de prueba se realiza una extracción de la imagen en pantalla y la generación de un archivo **JPEG**. La compresión de la imagen se realiza con un bajo factor de calidad del 5 % con la finalidad de que sean notorios los efectos de la compresión, sin embargo este es un parámetro ajustable dentro de las rutinas. Las imágenes recibidas para ambos casos se observan en la Figura 3.28.

Figura 3.28: Archivos JPEG recibidos.

Tanto el programa implementado en Linux como el script de *Python* en el receptor entregan datos en el terminal que sirven para conocer el estado de algunos módulos utilizados. La ejecución de los programas se observa en la Figura 3.29. Un dato importante es la diferencia en el tiempo de transacción de datos (0.81 segundos para el emisor y 0.89 segundos para el receptor). En una conexión cableada esto no sucedería, pero dado que el protocolo de **Bluetooth** realiza de mediador, los tiempos extra para lidiar con la interferencia del canal de comunicación generan un mayor tiempo de envío de datos para el receptor.

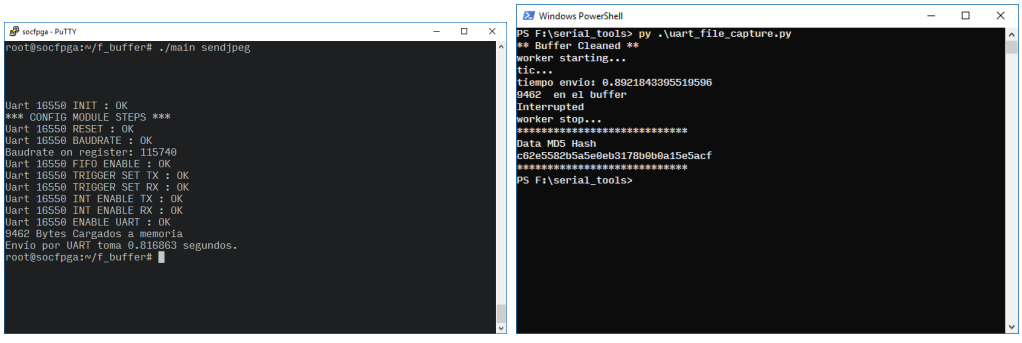


Figura 3.29: Ejecución de programas de emisión y recepción respectivamente.

Capítulo 4

Análisis de Resultados

A partir de las implementaciones realizadas surgen una serie de consideraciones que deben ser abordadas para un sistema de transmisión de imágenes. En particular se abordan las restricciones que limitan la mayor tasa de transferencia de datos y como esto se relaciona con un potencial uso del sistema para transmisión de video.

Además se realizan análisis respecto a los recursos de procesamiento y memoria utilizados para proyectar y procesar imágenes en pantalla. Dado que la proyección de imágenes es un sistema de demanda constante, se debe analizar si el sistema cuenta con las capacidades para realizar a tiempo procesamiento de imágenes.

4.1. Límites en Tasas de Transmisión

Para la transmisión de datos implementada en la Sección 3.4 existen una serie de elementos que imponen un tope a la tasa de transferencia del equipo. Esta cadena de elementos deben ser analizados para determinar una tasa de transferencia adecuada y para considerar si la tecnología es apta para transmisión de video.

Las limitaciones a considerar se listan a continuación:

- Límites de funcionamiento del módulo **UART 16550**.
- Tasa de transferencia máxima aceptada por el módulo HC-05.
- Límites prácticos de la transmisión por Bluetooth 2.0.

De acuerdo a la información entregada por [30] si bien **Bluetooth 2.0** soporta tasas de hasta 3Mbps, en la práctica dado que la banda está afecta a interferencia electromagnética y a otras transacciones propias del protocolo la tasa esperada en ambiente real es de ∼2.1Mbps. Del mismo modo en la *hoja de datos* del fabricante del módulo **HC-05** se menciona que para el perfil SPP la tasa máxima de datos es ∼1.38Mbps, aunque este dato debe ser tomado con cuidado pues en las pruebas de laboratorio estos valores han sido variables dependiendo de la interferencia electromagnética del lugar.

La limitante más clara se observa cuando se analiza la tasa de transferencias máximas alcanzadas por el módulo **UART 16550**. Se realizan pruebas de transmisión cableada ocupando una interfaz USB - UART con un módulo **FT232RL** según se observa en el esquema de la Figura 4.1.

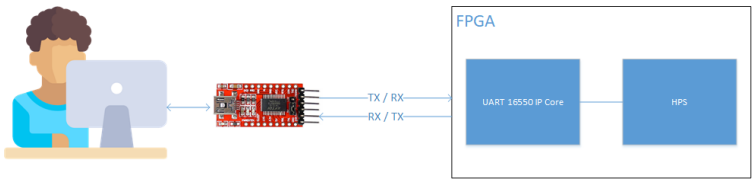


Figura 4.1: Pruebas para transmisión UART.

La metodología de prueba considera la transmisión de una imagen única a la que se le calcula un *hash MD5* para verificar la fiabilidad de los datos recibidos. Esta prueba se itera en varias ocasiones para verificar el límite físico de la transacción de datos. Los resultados se pueden observar en la Tabla 4.1.

Baudrate	**Hash**	**Time [s]**	**Success**
115200	3cf13ba3f9cc41f260368514be022c88	2,332251	TRUE
	3cf13ba3f9cc41f260368514be022c88	2,332248	TRUE
	3cf13ba3f9cc41f260368514be022c88	2,332250	TRUE
512000	3cf13ba3f9cc41f260368514be022c88	0,517945	TRUE
	3cf13ba3f9cc41f260368514be022c88	0,518423	TRUE
	3cf13ba3f9cc41f260368514be022c88	0,518563	TRUE
1024000	3cf13ba3f9cc41f260368514be022c88	0,259274	TRUE
	3cf13ba3f9cc41f260368514be022c88	0,259293	TRUE
	3cf13ba3f9cc41f260368514be022c88	0,259294	TRUE
2048000	daadd9c209450960cfb8c0de42032f12	0,172872	FALSE
	a21a73584106a03b2119aa843409e3c4	0,172790	FALSE
	9e86b948adf308152b061cbd50b33257	0,172875	FALSE
1248000	7ff67dcc0d8625c8808ef2dbba9392e4	0,259276	FALSE
	ec9aa8cd77533a64908ad1a4b93d65ef	0,259246	FALSE
	a03f55fedfeb9c424db3220130fa9e97	0,259094	FALSE

Tabla 4.1: Integridad de datos UART para distintos baudrate.

Se detecta el límite de *baudrate* en ∼1Mbps de datos. Se realizan otras pruebas donde se aumenta la frecuencia del reloj interno utilizado por el módulo **UART 16550 IP Core** pero no se obtienen mejores resultado. De este modo la principal limitante en la tasa de datos viene dado por el protocolo de transmisión UART.

Considerando los trabajos vistos en la Sección 2.1.2, las restricciones permitirían la transferencia de video digital con codificación **HEVC**. En particular de acuerdo al trabajo [27], para una imagen de resolución 560 x 416 y 50 fotogramas por segundo las imágenes de video con tasa de transferencia mayores a 340 kbps son aptas para diagnóstico médico.

4.2. Tiempos de escritura a memorias RAM

Según lo visto en la Sección 3.5 el módulo **Frambuffer** proyecta un sector de memoria RAM a la pantalla. Se analiza si el **HPS** tiene la capacidad para procesar y mover datos a la memoria RAM en un tiempo práctico que permita proyectar imágenes y video de manera fluida.

Es importante destacar que la tasa de refresco de 60Hz en pantalla está en control del módulo **Framebuffer** y se mantiene invariante. Lo que se mide con este análisis es la capacidad de tiempo de escritura del HPS hacia el espacio en memoria visto por el **Framebuffer**.

Para esto se utiliza la siguiente metodología:

- Se inicializa la proyección en pantalla con todos los pixeles en valor 0.
- Se modifica pixel por pixel los valores aumentando de 1 en 1 el valor en memoria respectivo.
- Cuando el frame completo es modificado se vuelve al primer pixel iterando nuevamente el proceso.
- En caso que se llegue al valor máximo de 255 por cada canal RGB del pixel, el se repite el proceso de forma inversa con valores en descenso.

Esta prueba de «stress» genera un efecto de imagen pulsante en pantalla con brillos aumentando y disminuyendo de un color negro a un blanco total. El programa encargado de generar esta rutina entrega los tiempos en que la pantalla pasa entre los valores extremos y con esto puede calcularse el tiempo promedio entre fotogramas. El efecto es sutil pero puede observarse en la Figura 4.2.

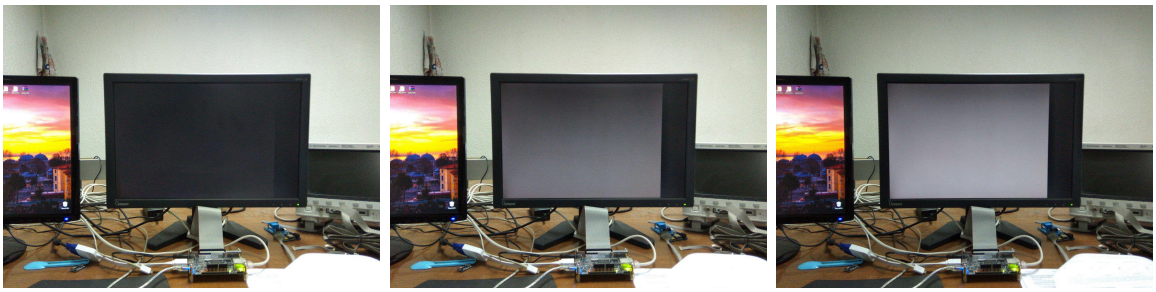


Figura 4.2: Pruebas para escritura a RAM.

Las impresiones que entrega la rutina del programa ejecutado se ve en la Figura 4.3. Acá se observa que el tiempo para escribir 256 imágenes se completa en ~ 0.56 [s]. Esto implica que el intervalo ocupado por una imagen ocurre en promedio en ~ 2.2 [ms]. Una comparativa con los FPS proyectados a pantalla y la capacidad de escritura del HPS en memoria RAM se observa en la Figura 4.4.

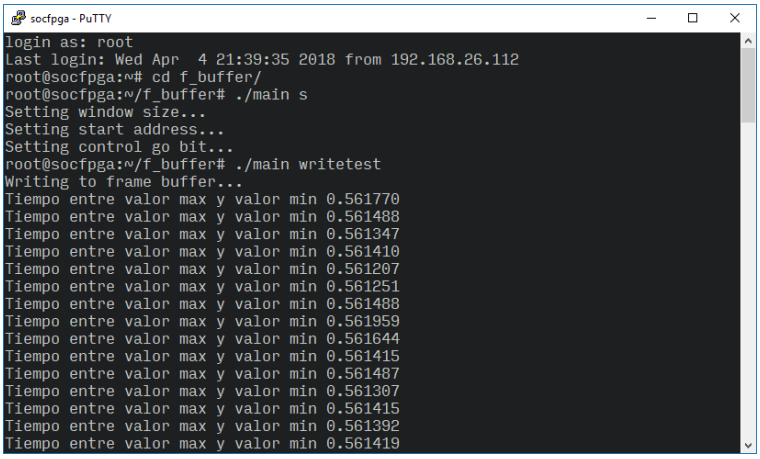


Figura 4.3: Datos entregados por programa de stress.

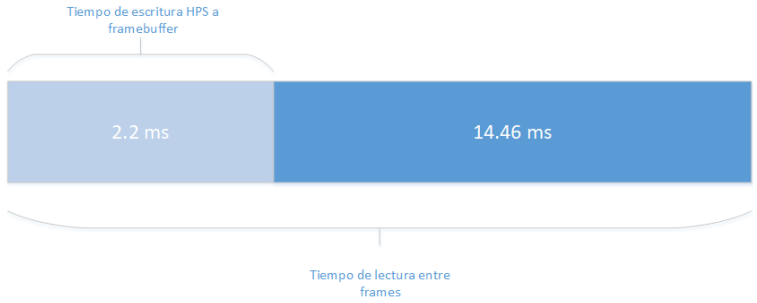


Figura 4.4: Comparación entre tiempos de escritura por el HPS y lectura por el módulo de Framebuffer.

4.3. Diferencias de Funcionamiento en Módulos Bluetooth

Se han detectado diferencias de funcionamiento en módulos Bluetooth basados en el mismo chip HC-05. Estas diferencias hacen referencia a la tasa de transmisión de datos que observa el receptor las cuales se vuelven notorias en condiciones de alta interferencia o baja intensidad de señal.

Según el perfil de funcionamiento SPP visto en la Sección 3.4.1 la tasa de transmisión de datos vista por el módulo emisor depende exclusivamente del *baudrate* entregado por el módulo UART. Sin embargo, en el proceso de recepción existe la interacción de los distintos protocolos que conforman la comunicación Bluetooth [19].

Para plasmar este efecto se realizan pruebas ocupando dos módulos que contienen el mismo modelo pero de distintos proveedores. El módulo HC-05 incluido en la tarjeta **RFS Daughter Card** y un módulo genérico incluido en diversos paquetes de desarrollo para Arduino. Los módulos que internamente reportan un *firmware* distinto se observan en la Figura 4.5.

Las pruebas realizadas consideran el envío de 26240 bytes de datos, bajo el mismo ambiente de interferencia y distancia de 1.5[m] entre módulos Bluetooth emisor y receptor. Los resultados vistos en la Tabla 4.2 consideran el tiempo de envío visto por el receptor junto

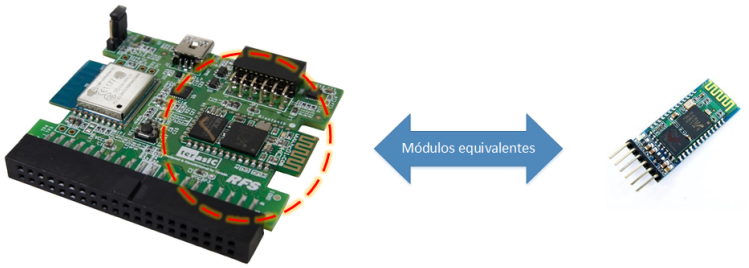


Figura 4.5: HC-05 incluido en RFS Daughter Card y módulo genérico.

con un *hash MD5* para confirmar la correcta recepción de datos. El *firmware* `2.0-20161226` corresponde al módulo genérico y el *firmware* `hc01.comV2.1` al módulo incluido en la tarjeta **RFS Daughter Card**.

Etiqueta	**Firmware**	**Hash**	**Time [s]**	**Success**
A (genérico)	2.0-20161226	59647d4c6e09cd69fc609dd9d58c86a6	2,313689	TRUE
		59647d4c6e09cd69fc609dd9d58c86a6	2,264907	TRUE
		59647d4c6e09cd69fc609dd9d58c86a6	2,269905	TRUE
B (RFS Card)	hc01.comV2.1	59647d4c6e09cd69fc609dd9d58c86a6	2,396240	TRUE
		59647d4c6e09cd69fc609dd9d58c86a6	2,456670	TRUE
		59647d4c6e09cd69fc609dd9d58c86a6	2,380852	TRUE

Tabla 4.2: Diferencias en tiempo de recepción de datos Bluetooth.

El módulo A presenta mejores tasas de transferencia en comparación a B, sin embargo esto no es un indicador directo de que la calidad de funcionamiento sea mejor ocupando el módulo A. En particular en las pruebas de laboratorio se ha notado una mayor resiliencia a las interferencias por parte del módulo B. Esto sugiere que el gasto extra de tiempo está asociado al uso interno de búfferes y mecanismos para lidiar que con la interferencia que están mejor implementados en el módulo B.

4.4. Consideraciones del Sistema Operativo

La rutina de software que controla el funcionamiento de la implementación vista en la Sección 3.7 corre sobre un sistema operativo Linux que funciona de forma independiente a las configuraciones hechas en la FPGA. Esto genera consideraciones sobre el rendimiento de la aplicación de software específica y su tiempo de inicio considerando el proceso de encendido.

4.4.1. Rendimiento de Aplicaciones

La ejecución de rutinas de software de forma directa en el procesador sin las capas de abstracción que genera un sistema operativo es llamado **Programación en Bare-metal**. Una comparativa de rendimiento se puede ver en [6] donde se observa como el uso de sistemas operativos modernos logra equiparar la eficiencia de escribir los programas en **bare-metal**.

Para la comparativa en [6] se construye un programa de transacción y procesamiento de datos entre el HPS y la FPGA. La FPGA genera interrupciones que son capturados por el procesador para la recepción de datos. Con cada entrada de datos se ejecuta una pequeña carga computacional para luego devolver los datos a la FPGA. Esta prueba se ejecuta compilando los programas primero para Linux y luego compilando para correr en bare-metal. Una comparativa de rendimiento se observa en la Figura 4.6.

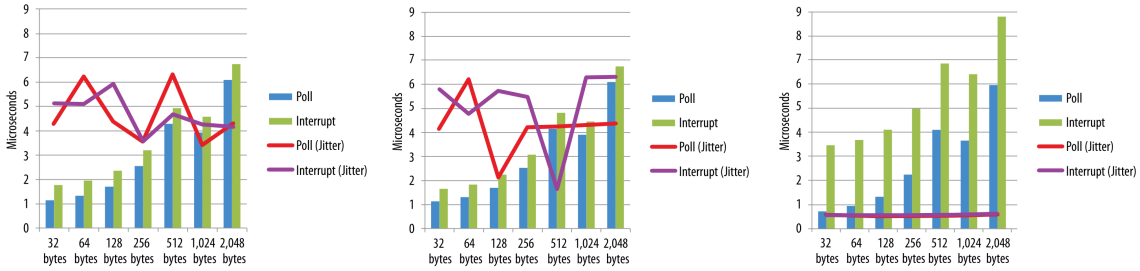


Figura 4.6: Comparativa de rendimiento para software bajo Linux sin carga, con un núcleo con carga y bare-metal respectivamente. Fuente: [6].

Se extiende de este análisis que para el uso de sistemas embebidos el uso de bare-metal no presenta ventajas significativas frente al uso de una capa de abstracción como es el sistema Linux debido a las optimizaciones específicas para cada plataforma que realizan los sistemas operativos modernos. Para el desarrollo de software si se ven ventajas importantes al hacer uso de Linux, pues se obtienen beneficios en la estabilidad del sistema al no tener que hacer ajustes manuales de forma directa a la arquitectura del procesador en específico por cada programa desarrollado y en la portabilidad que se obtiene en el código pues se ocupan las librerías del *kernel* de Linux para interactuar con el hardware.

4.4.2. Inicio de Sistema

El equipo de ecografía Taote inicia sus operaciones en ∼2 segundos. Su uso específico se caracteriza por apagados y encendidos constantes que deben funcionar de forma fluida con bajos tiempos de espera. El uso de un sistema operativo genera un costo extra de tiempo asociado al arranque (boot) del sistema según se observa en el esquema de la Figura 4.7. En específico para la aplicación desarrollada en la Sección 3.7 considerando el encendido de la plataforma de desarrollo el tiempo de inicio es de ∼12 segundos.

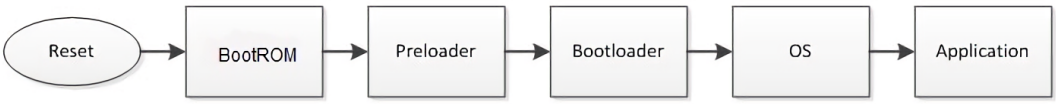


Figura 4.7: Bloques de inicio del sistema. Fuente: [12].

Según se observa en [12] los tiempos de inicio de los primeros bloques se mueven en el orden de los 100 mili-segundos y el tiempo crítico ocurre durante las rutinas de carga del sistema operativo (carga del *kernel* y *scripts* de inicio). Se sugieren las siguientes modificaciones para acelerar el proceso de arranque del sistema:

- Uso de una tarjeta microSD de clase 10.
- Modificar el *kernel* de Linux para disminuir la carga de drivers, funcionalidades y capacidades de depuración.
- Modificar los *scripts* de inicio para remover o retrasar la carga de funcionalidades o servicios.

En caso que esto no satisfaga las necesidades de inicio rápido del equipo pueden considerarse otras alternativas al uso del sistema operativo Linux. Existe la opción de correr programas en forma directa con rutinas en bare-metal según se ha visto en la Sección 4.4.1 y también existen opciones de sistemas operativos reducidos orientados al uso en sistemas embebidos como «microC» [5].

4.5. Resumen de los Análisis Efectuados

Utilizando los análisis efectuados a las distintas partes que componen el sistema propuesto en la Sección 3.7 se construye la Tabla 4.3 que engloba las diferentes aristas consideradas como potenciales problemas para la implementación propuesta.

Problemas Considerados	**Enfoque de Análisis**	**Apreciación Final**
Límites en Tasa de Transmisión	Analizar la cadena de elementos que participan en el proceso de transmisión en búsqueda del elemento más restrictivo para realizar pruebas de transferencia.	**Positiva**. Aunque no muy amplio, existe un margen para incluir la transmisión de video inalámbrico.
Tiempo de Escritura a RAM	Comparar el tiempo que toma dibujar un frame completo en pantalla versus el requerido por la tasa de refresco del sistema de video.	**Positiva**. El tiempo requerido para dibujar un frame ha demostrado ser 6 veces menor a la tasa de refresco requerido para el sistema de video.
Módulos Bluetooth de Diferentes Proveedores	Se realizan pruebas de rendimiento ocupando módulos Bluetooth HC-05 de diferentes proveedores, manteniendo en consideración las tasas de transmisión y la resiliencia al ruido electromagnético.	**Negativa.** Existen diferencias en el comportamiento de los módulos Bluetooth para distintos proveedores. Si bien el impacto parece ser menor, debe tomarse en cuenta para usos críticos del sistema.
Rendimiento de Aplicaciones en Linux versus Bare-metal	Analizar comparativas realizadas por Altera para su propia serie de sistemas FPGA - SoC.	**Positiva.** No existen diferencias de rendimiento considerable al usar una capa de sistema operativo. Se obtienen ventajas considerables para el desarrollo de software al tener una capa pre-configurada para comunicarse con el hardware pre-existente en el sistema.
Tiempo de Inicio de Sistema	Analizar las propuestas entregadas por Altera como opciones de arranque para sus chips Cyclone V SoC y recomendaciones para reducir los tiempos de inicio de sistema.	**Negativa.** Si bien existen métodos para optimizar el tiempo de arranque, es un problema que requiere dedicación especial y que no cuenta con soluciones directas.

Tabla 4.3: Resumen de problemas considerados en los análisis de resultados.

Capítulo 5

Conclusiones y Propuesta

En este trabajo se ha logrado implementar un sistema SoC - FPGA especializado que otorga nuevos mecanismos de conectividad para Taote cumpliendo con los objetivos propuestos para el Trabajo de Título en relación a la transmisión inalámbrica de imágenes. Este sistema permite realizar capturas de forma directa por rutinas de software generando un sistema escalable para la compresión y transmisión de imágenes por distintos medios. Además, la propuesta resulta coherente con el diseño electrónico existente de Taote y permite a futuro elaborar nuevas líneas de trabajo que a partir de la arquitectura SoC - FPGA ocupada, permitiría explotar otras áreas como realizar mejores interfaces de usuario, integración rápida de nuevos periféricos y conectividad en el contexto de «Internet de las Cosas».

Se ha implementado el envío de imagen única haciendo uso de la tecnología Bluetooth y rutinas de software que trabajan con librerías que permiten realizar compresión JPEG. El sistema mantiene modularidad pensando en que una implementación de transmisión digital de video inalámbrico consideraría mayormente solo modificaciones a las rutinas de *software* y no a los diseños de *hardware* hechos en FPGA. Del mismo modo el cambio de protocolo de transacción inalámbrica puede adaptarse de Bluetooth a WiFi con cambios en el *hardware* y con modificaciones mínimas a las rutinas de *software* implementadas.

La propuesta que deriva de este trabajo es que la actualización de los actuales chips **Cyclone II** y **Cyclone IV**, utilizados actualmente en el equipo Taote como núcleo de su arquitectura, a la nueva línea **Cyclone V SoC** junto con la entrega de recursos de memoria RAM permitiría realizar nuevas implementaciones que satisfagan las necesidades de un mercado que demanda mayor conectividad. La capacidad de realizar rutinas de software permite ahorrar tiempo importante de desarrollo al permitir la reutilización de código como fue el caso del uso de la librería pública para la compresión JPEG vista en la Sección 3.6.1 y como podría ser el caso para la implementación de transmisión inalámbrica de video. Los diseños ganan gran ventaja además en términos de confiabilidad y robustez pues las rutinas de software se aplican en un hardware verificado como es el uso del HPS basado en el uso de un procesador ARM Cortex-A9 que otorga garantías en el funcionamiento del sistema.

Finalmente, mencionar que los principales desafíos se mostraron a la hora de generar los conocimientos necesarios para ocupar de forma adecuada la nueva plataforma **Cyclone V**

SoC. El uso conjunto de SoC - FPGA en un mismo chip es relativamente nuevo y marca un cambio importante en la línea de chips anteriores como son **Cyclone II, Cyclone III, Cyclone IV, línea Spartan de Xilinx, etc.** por lo que los recursos en términos de ejemplos y documentación no son de gran abundancia.

5.1. Trabajos Futuros

Pueden derivarse diversos trabajos futuros a partir del diseño estudiado, incluyendo pruebas para la transmisión de video inalámbrico, el estudio de otros sistemas operativos como base para las rutinas de software y la compilación de un sistema Linux personalizado. Cada una de estas temáticas entregaría conocimientos más profundos respecto al funcionamiento y limitaciones que el uso de de los chips **Cyclone V SoC** entregaría en términos de conectividad y personalización para el equipo Taote.

Según se ha visto en la Sección 2.1.2, la utilización del codec HEVC presenta una solución atractiva para la transmisión de imágenes médicas de ultrasonido, tanto en términos de una tasa de transmisión adecuada a las tecnologías Bluetooth y WiFi, como en términos de calidad para el diagnóstico médico. Un trabajo similar al realizado para la entrega de esta memoria aplicado a la transmisión de video entregaría mejores conocimientos respecto al comportamiento del codec en ambientes de alto ruido electromagnético, limitaciones y consideraciones especiales para una implementación de este tipo.

El estudio de otros sistemas operativos se enmarca dentro de la necesidad de cumplir con un tiempo de inicio adecuado para el sistema según se ha visto en la Sección 4.4.2. Es necesario definir cuanto se puede optimizar el inicio del sistema Linux dentro de las sugerencias vistas en [12]. Resultaría útil evaluar la posibilidad de ocupar programación en bare-metal o el uso de otros sistemas operativos como microC. Esto puede acarrear otros problemas que deben ser estudiados como la dificultad de programar rutinas avanzadas sin la abstracción que entrega el sistema operativo Linux.

Resultaría útil además, generar los conocimientos que permitan obtener una versión de Linux personalizada. La versión entregada con la plataforma de desarrollo DE10-Nano entrega un caso de uso específico para los periféricos entregados con es plataforma, pero es altamente probable que una implementación aplicada al equipo Taote necesite una serie de elementos y periféricos personalizados. De este modo se necesitaría una versión de Linux que cargue las configuraciones y drivers para nuestra propia plataforma.

Bibliografía

[1] Ahn, S., Kang, J., Kim, P., Lee, G., Jeong, E., Jung, W., Park, M., and Song, T. K. (2015). Smartphone-based portable ultrasound imaging system: Prototype implementation and evaluation. In *2015 IEEE International Ultrasonics Symposium, IUS 2015*.

[2] Altera Corporation (2007). Engineering Change Management with Chip Planner. *Altera Documentation*, (March).

[3] Altera Corporation (2009). Understanding Metastability in FPGAs. *Altera Documentation*, (July):1–6.

[4] Altera Corporation (2010). Guaranteeing Silicon Performance with FPGA Timing Models. *Altera Documentation*, pages 1–11.

[5] Altera Corporation (2011). Using MicroC/OS-II RTOS with the Nios II Processor. *Altera Documentation*, (May):18.

[6] Altera Corporation (2014a). Bare-Metal, RTOS, or Linux? Optimize Real-Time Performance with Altera SoCs. *Altera Documentation*, (December):12.

[7] Altera Corporation (2014b). Quartus II Handbook , Volume 3 Verification Preliminary Information. *Altera Documentation*, 3(408):1–12.

[8] Altera Corporation (2014c). Quartus II Handbook Volume 2: Design Implementation and Optimization. *Altera Documentation*, 2.

[9] Altera Corporation (2014d). What is an SoC FPGA? *Altera Support Resources*, page 4.

[10] Altera Corporation (2016a). Cyclone V Hard Processor System Technical Reference Manual. *Altera Documentation*.

[11] Altera Corporation (2016b). How and when can I enable the FPGA2SDRAM bridge on Cyclone V SOC and Arria V SOC devices? Disponible en https://www.altera.com/support/support-resources/knowledge-base/embedded/2016/how-and-when-can-i-enable-the-fpga2sdram-bridge-on-cyclone-v-soc.html.

[12] Altera Corporation (2016c). HPS SoC Boot Guide - Cyclone V SoC Development Kit. *Altera Documentation*, page 30.

[13] Altera Corporation (2017a). *DE10-Nano User Manual v1.5.*

[14] Altera Corporation (2017b). Embedded Peripherals IP User Guide. *Altera Documentation*, (June):358.

[15] Altera Corporation (2017c). Intel SoC FPGA Embedded Design Suite User Guide. *Altera Documentation.*

[16] Altera Corporation (2017d). Intel ® Quartus ® Prime Standard Edition Handbook. *Altera Documentation.*

[17] Altera Corporation (2017e). Video and Image Processing Suite User Guide. *Altera Documentation.*

[18] Basu, A. (2016). ESP8266 Wifi With Arduino Uno and Nano. Disponible en https://blogs.msdn.microsoft.com/abhinaba/2016/01/23/esp8266-wifi-with-arduino-uno-and-nano/.

[19] Bhagwat, P. (2001). Bluetooth: Technology for short-range wireless apps. *IEEE Internet Computing*, 5(3):96–103.

[20] Chen, D., Cong, J., and Pan, P. (2006). *FPGA Design Automation: A Survey*, volume 3.

[21] Gorev, M. and Ellervee, P. (2010). FPGA based system for video compression and transmission over bluetooth. *2010 53rd IEEE International Midwest Symposium on Circuits and Systems*, pages 367–370.

[22] Herrera Gajardo, J. I. (2017). *DISEÑO E IMPLEMENTACIÓN DE TÉCNICAS DE PROCESAMIENTO DE IMÁGENES PARA DISPOSITIVO DE ULTRASONIDO PORTÁTIL.* Tesis de pregrado, Universidad de Chile.

[23] National Instruments (2014). Comunicación Serial: Conceptos Generales. Disponible en http://digital.ni.com/public.nsf/allkb/039001258CEF8FB686256E0F005888D1.

[24] Oróstica Navarrete, R. E. (2018). *FACTIBILIDAD DE DOPPLER COLOR EN EQUIPO DE ECOGRAFÍA PORTÁTIL COSTO-EFECTIVO TAOTE.* Tesis de posgrado, Universidad de Chile.

[25] Panayides, A., Loizou, C. P., Pattichis, M. S., Kyriacou, E., Shizas, C. N., Nicolaides, A. N., and Pattichis, C. S. (2013). Ultrasound video despeckle filtering for High Efficiency Video Coding in m-health systems. *IET Seminar Digest*, 2013(1):2–5.

[26] Panayides, A., Pattichis, M. S., and Pattichis, C. S. (2009). Towards Diagnostically Robust Medical Ultrasound Video Streaming using H. 264. *Biomedical.*

[27] Panayides, A. S., Pattichis, M. S., Loizou, C. P., Pantziaris, M., Constantinides, A. G., and Pattichis, C. S. (2015). An effective ultrasound video communication system using despeckle filtering and HEVC. *IEEE Journal of Biomedical and Health Informatics*, 19(2):668–676.

[28] Persons, K. R., Palisson, P. M., Manduca, A., Charboneau, W. J., James, E. M., Charboneau, N. T., Hangiandreou, N. J., and Erickson, B. J. (2000). Ultrasound grayscale image compression with JPEG and wavelet techniques. *Journal of Digital Imaging*, 13(1):25–32.

[29] Piña Lagos, J. C. (2013). *Estudio de transmisión inalámbrica de imágenes en aplicaciones de corto alcance; elección y caracterización entre enlaces analógicos y digitales.* Tesis de pregrado, Universidad de Chile.

[30] Sparkfun Electronics (2014a). Bluetooth Basics - learn.sparkfun.com. Disponible en https://learn.sparkfun.com/tutorials/bluetooth-basics/all.

[31] Sparkfun Electronics (2014b). Serial Communication - learn.sparkfun.com. Disponible en https://learn.sparkfun.com/tutorials/serial-communication/all.

[32] Terasic Inc. (2016). *RFS Daughter Card User Manual.*

[33] Wallace, G. K. (1992). The JPEG still picture compression standard. *IEEE Transactions on Consumer Electronics*, 38(1):xviii–xxxiv.

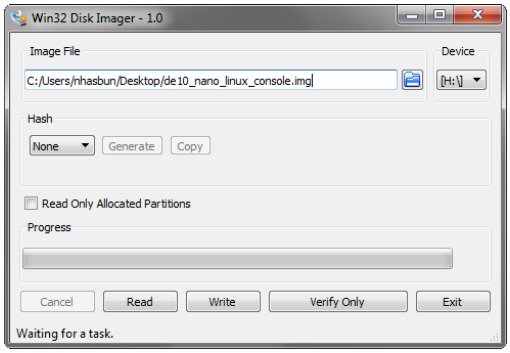
Anexos

Apéndice A

Uso del Sistema Linux

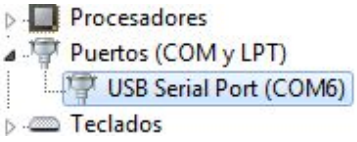
A.1. Carga de imagen a tarjeta microSD

Se carga una de las imágenes entregadas por Terasic y se carga a una tarjeta microSD utilizando el software **Win32 Disk Imager**.



A.2. Comunicación Serial

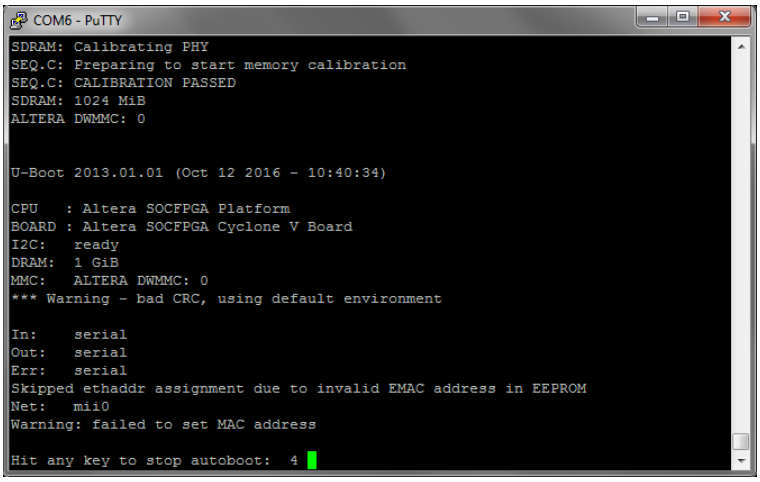
El puerto serial tiene un *baudrate* por defecto de 115200 bps.



A.3. Ajustar parámetros de arranque

La plataforma da 5 segundos para ver si el usuario desea entrar a la configuración del bootloader **U-Boot** de Altera. En esta página (`https://rocketboards.org/foswiki/Documentation/`

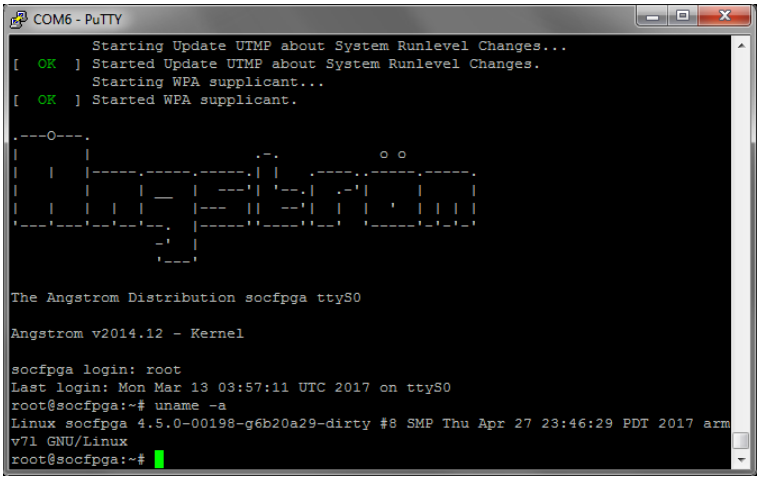
`PreloaderUbootCustomization131`) se encuentra una serie de comandos que pueden ser manejados desde el bootloader.



- **CONFIG_BOOTDELAY**: Delay inicial para comenzar el booteo del sistema.
- **CONFIG_BAUDRATE**: Cambio de baudrate para la conexión UART.
- **env save**: graba cambios a la memoria SD.
- **env default -a**: reset de datos.

A.4. Arranque de sistema Linux

Usuario: «root» (sin password).



La forma de momento más fácil de transferir archivos ha sido utilizando el software **WinSCP**. Se conecta la placa en red, se pide su IP con **ifconfig eth0** y luego se hace transferencia de archivos por **SSH/SFTP**. También se puede conectar por medio de su nombre de red **socfpga**. La placa viene por defecto con un demonio de servidor SSH.

A.5. Instalación de programas por gestor de paquetes

La placa corre una distribución de Linux llamada Ansgtrom Distribution que trae un gestor de paquetes **opkg** similar al gestor **apt** encontrado en las distribuciones Ubuntu. Un buen manual de uso se puede encontrar en la siguiente dirección `https://wiki.openwrt.org/doc/techref/opkg`.

Vale la pena actualizar e instalar al principio de cada imagen de Linux el editor de textos nano, que funciona via terminal.

```
opkg update
opkg install nano
```

Búsqueda de paquetes rápida haciendo,

```
opkg list | grep packagename
```

A.6. Acceso a registros desde terminal

Para revisar registros de forma rápida desde Linux puede ser útil ocupar el programa **memtool**.

- **Read:** `memtool 0xffc25080 4`
- **Write:** `memtool 0xffc25080=0x111`

Apéndice B

Uso de la interfaz FPGA2SDRAM

B.1. Espacios de memoria dedicados para FPGA y Linux

Explicación sobre cómo limitar un espacio para la FPGA y otro para el HPS, de forma que no se solapen las direcciones puede verse en el Capítulo «5.6 Nios II Access HPS DDR3» de [13].

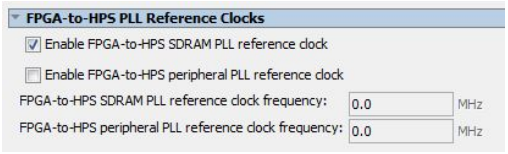
Este es un ejemplo de comandos que van en U-Boot para setear el tamaño de memoria para ser usado por Linux:

```
setenv mmcboot "setenv bootargs console=ttyS0,115200 root=/dev/mmcblk0p2 rw \
rootwait mem=512M;bootz 0x8000 - 0x00000100"
env save
run bootcmd
```

B.2. Activación de interfaz FPGA2SDRAM

A partir del proyecto `DE10_NANO_SoC_GHRD` entregado por Terasic:

1. Desde Qsys activar un reloj para el bridge **fpga2sdram**. Esto es importante y no sale en la documentación.



2. Compilar el proyecto en Quartus.
3. Desde **SoC EDS** ejecutar bsp-editor, y cargar los archivos por defecto de la carpeta `hps_isw_handoff`. Generar archivos.

4. Dentro de la carpeta `software/spl-bsp` ejecutar **make**.
5. Actualizar preloader. Insertar la tarjeta microSD con la imagen de linux-console de Terasic pre-cargada y ejecutar:
   `alt-boot-disk-util -p preloader-mkpimage.bin -a write -d I`
6. Generar archivo `soc_system.rbf` usando al script `sof_to_rbf.bat` entregado por Altera y copiarlo a microSD. Este es uno de lo pasos más importantes y más inconvenientes. El bridge **FPGA2SDRAM** debe ser levantado antes del inicio del sistema Linux, y este es el modo de hacer que el bootloader lo haga antes de iniciar el sistema.

Con esto se puede ocupar cualquier otro proyecto que ocupe la interfaz **fpga2sdram** mientras se mantenga el tipo de interfaz (Avalon-MM Master de 256bits). Es recomendable para debugear y desarrollar mantener el archivo `soc_system.rbf` precargado en la microSD, de este modo el bridge siempre se activa al inicio del sistema y luego uno puede modificar la FPGA con un programa **sof** propio por medio del USB Blaster.

B.3. Debug por JTAG

Se puede corroborar el correcto funcionamiento de la interfaz *FPGA2SDRAM* ocupando el IP Core **JTAG to Avalon Master Bridge** que permite verificar direcciones de memoria por consola. En QSys → System Console:

```
get_service_paths master
set jtag_master [lindex [get_service_paths master] 0]
##Cuidado con elegir el indice para set jtag_master, debe coincidir
open_service master $jtag_master
master_write_32 $jtag_master 0x0000 1
master_read_32 $jtag_master 0x0004 4
```

B.4. Links útiles

Algunos recursos que contienen información más detallada:

- JTAG to Avalon master bridge usage - Altera Forums
  `https://alteraforum.com/forum/showthread.php?t=29188`
- Cyclone V SoC - Shared Memory Controller
  `https://www.alteraforum.com/forum/showthread.php?t=41489&page=3`
- With the Cyclone V SoC, how can I use the FPGA fabric to access the HPS DMA?
  `http://www.alteraforum.com/forum/showthread.php?t=45290`
- How can I enable the FPGA2SDRAM bridge on Cyclone V SOC and Arria V SOC devices? (Intel FPGA support)
  `https://www.altera.com/support/support-resources/knowledge-base/embedded/2016/how-and-when-can-i-enable-the-fpga2sdram-bridge-on-cyclone-v-soc.html`

Apéndice C

Otros Recursos

C.1. Repositorio de proyectos implementados

C.1.1. Github

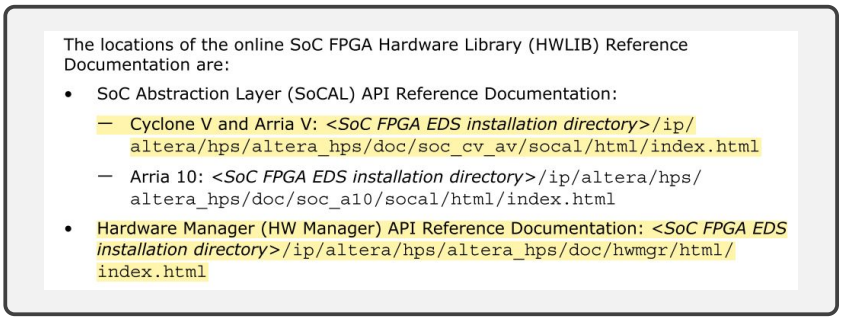
- **uart_core_lib** - Driver - Library for C applications using Altera's UART Core through Avalon Bus on Cyclone V.
  `https://github.com/nhasbun/uart_core_lib`
- **uart_16550_core_lib** - Altera wrappers for C applications using Altera's 16550 UART Core through Avalon Bus on Cyclone V.
  `https://github.com/nhasbun/uart_16550_core_lib`
- **de10nano_ledtest** - Simple test for interfacing FPGA and SoC on a de10 nano board.
  `https://github.com/nhasbun/de10nano_ledtest`
- **vgaHdmi_chip** - Test for video output using the ADV7513 chip on a de10 nano board.
  `https://github.com/nhasbun/de10nano_vgaHdmi_chip`

C.1.2. Bitbucket

- **altera_framebuffer_manager** - Software that works under Linux to take control of Altera Framebuffer IP Core module.
  `https://bitbucket.org/nhasbun/altera_framebuffer_manager`
- **altera_framebuffer_de10nano** - Altera Framebuffer IP Core implementation for a simple video system on a de10nano board. The buffer is stored in a DDR3 memory space using the FPGA-to-HPS SDRAM interface.
  `https://bitbucket.org/nhasbun/altera_framebuffer_de10nano`

C.2. Librerías SoCAL y HWLIB

Las placas SoC FPGA de Altera se entregan con una serie de librerías que pueden ser ocupadas para programas en bare-metal y/o Linux para acceder a configuraciones de bajo nivel. Detalles sobre los alcances de esta documentación se pueden encontrar en el Capítulo **8. Hardware Library** de [15] y la documentación en específico se agrega con las instalaciones de Quartus Prime.



The locations of the online SoC FPGA Hardware Library (HWLIB) Reference Documentation are:

- SoC Abstraction Layer (SoCAL) API Reference Documentation:
  - Cyclone V and Arria V: *<SoC FPGA EDS installation directory>*`/ip/altera/hps/altera_hps/doc/soc_cv_av/socal/html/index.html`
  - Arria 10: *<SoC FPGA EDS installation directory>*`/ip/altera/hps/altera_hps/doc/soc_a10/socal/html/index.html`
- Hardware Manager (HW Manager) API Reference Documentation: *<SoC FPGA EDS installation directory>*`/ip/altera/hps/altera_hps/doc/hwmgr/html/index.html`