MAESTRÍA EN CIENCIAS EN SISTEMAS DIGITALES
“PROCESAMIENTO DE VIDEO UTILIZANDO LA
PLATAFORMA DAVINCI”

TESIS
QUE PARA OBTENER EL GRADO DE
MAESTRIA EN CIENCIAS

PRESENTA:
GUSTAVO LUIS VIDAL GONZÁLEZ
BAJO LA DIRECCIÓN DE:
DRA. MIREYA S. GARCÍA VÁZQUEZ
DR. ALEJANDRO A. RAMÍREZ ACOSTA

JUNIO 2011
TIJUANA, B.C., MÉXICO
INSTITUTO POLITÉCNICO NACIONAL
SECRETARÍA DE INVESTIGACIÓN Y POSGRADO

ACTA DE REVISIÓN DE TESIS

En la Ciudad de Tijuana, B.C. siendo las 18:00 horas del día 13 del mes de junio del 2011 se reunieron los miembros de la Comisión Revisora de Tesis, designada por el Colegio de Profesores de Estudios de Posgrado e Investigación de CITEDI para examinar la tesis titulada:

PROCESAMIENTO DE VIDEO UTILIZANDO LA PLATAFORMA DA VINCI.

Presentada por el alumno:

VIDAL GONZÁLEZ GUSTAVO LUIS
Apellido paterno Apellido materno Nombre(s)

Con registro: A 0 9 0 5 5 7

aspirante de:
MAESTRÍA EN CIENCIAS EN SISTEMAS DIGITALES

Después de intercambiar opiniones, los miembros de la Comisión manifestaron APROBAR LA TESIS, en virtud de que satisface los requisitos señalados por las disposiciones reglamentarias vigentes.

LA COMISIÓN REVISORA

Directores de tesis

DRA. MIREYA SARAI GARCÍA VÁZQUEZ
DR. ALEJANDRO ALVARO RAMÍREZ ACOSTA
DR. LLUÍS ÁLVARO AGUILAR BUSTOS
DR. JUAN JOSE TAPIA ARMENDÁRIZ
DR. FRANCISCO JAVIER GARCÍA UGALDE

PRESIDENTE DEL COLEGIO DE PROFESORES

DR. LUIS ARTURO GONZÁLEZ HERNÁNDEZ
En la Ciudad de Tijuana, Baja California, el día 16 del mes Junio del año 2011, el (la) que suscribe Gustavo Luis Vidal González, alumno (a) del Programa de Maestría en Ciencias en Sistemas Digitales con número de registro A090557, adscrito a Centro de Investigación y Desarrollo de Tecnología Digital, manifiesta que es autor (a) intelectual del presente trabajo de Tesis bajo la dirección de Dra. Mireya Sarao García Vázquez y Dr. Alejandro Alvarado Ramírez Acosta, cede los derechos del trabajo intitulado “Procesamiento de Video Utilizando la plataforma DaVinci”, al Instituto Politécnico Nacional para su difusión, con fines académicos y de investigación.

Los usuarios de la información no deben reproducir el contenido textual, gráficas o datos del trabajo sin el permiso expreso del autor y/o director del trabajo. Este puede ser obtenido escribiendo a la siguiente dirección av. del parque no. 1310, mesa de Otay, Tijuana, Baja California, México, cp 22510. Si el permiso se otorga, el usuario deberá dar el agradecimiento correspondiente y citar la fuente del mismo.

Gustavo Luis Vidal González

Nombre y firma
Deseo agradecer especialmente a mi esposa Liz Éthel Muñoz Ramos por su constante confianza, apoyo y paciencia para alcanzar las metas propuestas.

Agradezco a mis padres Elvira González González y Francisco Vidal Bonilla por su apoyo, motivación y por creer en mi potencial. A mis hermanos Walter y Carlos Vidal González por su motivación y gran amistad.

Deseo Agradecer a la familia Muñoz Ramos del cual ahora formo parte por su apoyo, paciencia y motivación durante mi estudio de la Maestría.

Deseo agradecer a todos los maestros que han colaborado en la elaboración de esta tesis, con asesorías, revisiones y comentarios que fueron valiosos para la culminación de esta investigación. Especialmente a mis Directores de Tesis Dra. Mireya Sarai García Vázquez y Dr. Alejandro Álvaro Ramírez Acosta por el tiempo dedicado y seguimiento de mi trabajo.

Agradezco al CONACYT y al Instituto Politécnico Nacional por la oportunidad de continuar con mis estudios y permitirme ser parte de una institución tan prestigiosa e histórica.

Finalmente, mis agradecimientos a todo el cuerpo docente y administrativo de CITEDI.
ÍNDICE
RESUMEN ............................................................................................................................................. iv
ABSTRACT .............................................................................................................................................. v
CAPÍTULO I  INTRODUCCIÓN ........................................................................................................... 6
I.1 Objetivo de la investigación .......................................................................................................... 9
   I.1.1 Objetivos específicos ............................................................................................................... 9
I.2 Organización de la tesis ................................................................................................................. 9
I.3 Aportaciones del trabajo de tesis ................................................................................................. 10
CAPÍTULO II  CODIFICACIÓN DE VIDEO ...................................................................................... 11
   Introducción ........................................................................................................................................... 11
   II.1 La codificación del video ........................................................................................................... 11
   II.2 Estándar MPEG-4 versión 2 ....................................................................................................... 12
   II.3 Alcances y características del estándar MPEG-4 ....................................................................... 12
      II.3.1 Representación codificada de los objetos de media .............................................................. 14
      II.3.2 Descripción y sincronización de datos afluentes para objetos de media .............................. 15
      II.3.3 Entrega de datos afluentes ....................................................................................................... 16
   II.4 Versiones en MPEG-4 ................................................................................................................. 17
   II.5 Funcionalidad Visual en el MPEG-4 ........................................................................................ 19
      II.5.1 Formatos soportados ................................................................................................................ 19
      II.5.2 Eficiencia de Compresión ........................................................................................................ 19
      II.5.3 Robustez en ambientes propensos a errores ............................................................................. 19
   II.6 Perfiles en MPEG-4 .................................................................................................................... 20
      II.6.1 MPEG-4 Perfil Simple (SP) .................................................................................................... 21
      II.6.2 MPEG-4 Perfil Avanzado Simple (ASP) ................................................................................. 25
   II.7 Conclusiones ............................................................................................................................... 27
CAPÍTULO III  PLATAFORMA DE DESARROLLO ........................................................................... 28
   III.1 Tecnología DaVinci ................................................................................................................... 28
   III.2 System-On-Chip ........................................................................................................................ 29
   III.3 Sistemas embebidos para codificador MPEG-4 ...................................................................... 29
   III.4 Módulo de evaluación DVEVM TMS320DM355 ................................................................. 30
      III.4.1 DM355 EVM ...................................................................................................................... 30
      III.4.2 Mapeo de memoria ............................................................................................................ 32
III.4.3 Interruptores de configuración

III.4.4 Componentes en la tarjeta

III.4.4.1 Interfaces EMIF

III.4.4.1.1 Flash, NAND Flash, interface Ethernet

III.4.4.1.2 Interface de memoria DDR2

III.4.4.1.3 Interface tarjeta de memoria

III.4.4.1.4 Interface UART

III.4.4.1.5 Interface USB

III.4.4.2 Interfaces de entrada del puerto de Video/Imagen

III.4.4.2.1 El Chip DAC de salida de video

III.4.4.3 Interface de Ethernet

III.4.4.4 Interface I2C

III.4.4.4.1 MSP430

III.4.4.5 Interfaces para tarjetas secundarias

III.4.5 Instalación del software DVEVM

III.4.5.1 Componentes de software

III.4.5.2 Instalación del software

III.4.5.2.1 Instalación de Linux para la plataforma EVM

III.4.5.2.2 Instalación del software DVSDK

III.4.5.2.3 Instalación de archivos Audio/Video de demostración

III.4.5.2.4 Instalación de LPTB

III.4.5.3 Exportando el sistema de archivos compartido para el acceso con la plataforma

III.5 Conclusiones

IV.1 Arquitectura DaVinci

IV.2 Codificador MPEG-4 SP de Texas Instruments

IV.3 Codificador MPEG-4 ASP de Xvid

IV.3.1 Herramientas de codificación en Xvid

IV.3.2 Plataforma de prueba para xvidcore

IV.3.3 Aplicación “xvid_encraw” controladora del codificador Xvid

IV.4 Conclusiones

V.1 Método de desarrollo

Procesamiento de video utilizando la plataforma DaVinci
RESUMEN

Este trabajo de tesis constituye la implementación de un codificador MPEG-4 ASP (Advanced Simple Profile) en una arquitectura con un procesador ARM bajo la plataforma DaVinci de Texas Instruments. Los métodos y técnicas para la codificación de video bajo el estándar MPEG-4 se analizan. Los componentes que constituyen el hardware de la Plataforma DaVinci se analizan y evalúan para su uso en el desarrollo de este trabajo.

En la implementación se utilizan los componentes de software contenidos en el DVSDK v1.3 (Digital Video Software Development Kit) para plataforma DaVinci. Tal software es destinado a ser ejecutado en el procesador ARM bajo un sistema operativo. Por tal motivo se utiliza una versión de Linux portada al procesador ARM conocido como MontaVista Linux proporcionado en conjunto con el DVSDK v1.3.

Debido al uso de este software, en el trabajo de tesis se analizan y utilizan los componentes de software estandarizados para implementar codificación de video MPEG-4 en la plataforma DaVinci basado en el codificador XVID. Como parte final del trabajo de tesis se evalúa los resultados de la codificación de video en el procesador ARM.

**Palabras Clave:** procesador digital de multimedia, DM355, módulo de evaluación de video digital, DVEVM, codificación de video, MPEG-4, codificador XVID
Using Video Processing on the DaVinci Platform

ABSTRACT

This thesis work constitutes the implementation of an MPEG-4 ASP (Advanced Simple Profile) in an ARM processor architecture within the Texas Instruments DaVinci platform. The methods and techniques for video coding under the MPEG-4 standard are analyzed. The components that constitute the hardware on the DaVinci platform are analyzed and evaluated for their use in the development of this work.

In the implementation, software components are used that are contained in the DVSDK v1.3 (Digital Video Software Development Kit) for the DaVinci platform. Such software is destined to be executed on the ARM processor with an operating system. Thus a version of the Linux operating system is used known as MontaVista Linux, contained alongside the DVSDK v1.3.

Given the nature of the software used alongside the implementation of an XVID based MPEG-4 video coder on the DaVinci Platform, the analysis and utilization of these standardized software components is contained in the thesis work. As a final part of the thesis work the results of video coding on the ARM processor are evaluated.

Keywords: digital media processor, DM355, digital video evaluation module, DVEVM, video coding, MPEG-4, XVID codec
CAPÍTULO I  INTRODUCCIÓN

Hasta hace unos años, la razón principal para la codificación fuente de video era la compresión, ya que el ancho de banda de los canales de transmisión y las capacidades de los medios de almacenamiento fueron los factores limitantes para la velocidad de transferencia de datos. Hoy en día, aunque los medios de almacenamiento y la capacidad del canal han mejorado mucho, la compresión sigue siendo una parte muy importante de la codificación de video. Sin comprimir los datos de video, sistemas como la videoconferencia, televisión digital e Internet no serían capaces de proporcionar una calidad de video razonable [1].


El estándar MPEG-4 incluye herramientas que reducen el ancho de banda en al menos un 50% necesario para distribuir contenido con “calidad DVD” de definición estándar y de alta definición. El estándar MPEG-4 ASP tiene una mejora del 20% en comparación con el estándar MPEG-2 [5], el estándar MPEG-4 AVC tiene mejoras del 50% y más [1].

Estas ganancias significativas en la eficacia de codificación, se deben al desarrollo de numerosas herramientas estándar de codificación que se pueden aplicar según sea necesario.
Introducción

Capítulo I

Procesamiento de video utilizando la plataforma DaVinci para los objetivos fijados por los fabricantes de compresión del codificador. Aunque es costoso implementar las herramientas del estándar MPEG-4 en un entorno de tiempo real, avanzados procesadores de multimedia (PDM) se han integrado recientemente a productos comerciales de codificación. Los procesadores rápidos de propósito general (x86, Intel) también permiten la implementación de las herramientas de codificación del estándar MPEG-4 en tiempo no real, para la edición en formato MPEG-4.

Con el surgimiento de la tecnología de microprocesadores y la aparición de organismos de estandarización en el área de la codificación de video tales como la ISO/IEC [6] y la ITU-T [4], se ha constituido un giro tecnológico importante, que ha permitido a los sistemas embebidos de comunicación audiovisual (multimedia) convertirse en una realidad rápidamente perceptible. Los sistemas multimedia embebidos a base de microprocesadores han sido introducidos en varias áreas de aplicación tales como la televisión digital, la videoconferencia, la videotelefonía, la telemedicina, las terminales de comunicación inalámbricas, los sistemas de video vigilancia, entre otras. En los últimos años, estos sistemas se han convertido en productos para el público. Asimismo, es interesante constatar que actualmente usamos sin darnos cuenta más de una docena de sistemas embebidos en nuestra vida cotidiana.

Un sistema embevido se puede describir como un sistema autónomo, que tiene un conjunto de elementos físicos necesarios para su funcionamiento (procesador, memoria, periféricos de entrada/salida), usando conjuntamente programas (sistema operativo, drivers, etc.) para resolver funciones y tareas específicas.

Las restricciones asociadas a las diferentes aplicaciones de los sistemas embebidos son tan diversas y variables con respecto a su modo de operación. En primer lugar, la mayoría de aplicaciones son implantadas en aparatos compactos y portátiles teniendo restricciones en el grado de integración y consumo de energía. En segundo lugar, el tiempo de salida al mercado se ha convertido en un parámetro crucial en la fase de la concepción del producto. En tercer lugar, estos sistemas tienen restricciones de tiempo real algunas veces rigurosas que necesitan una gran cantidad de cálculos. Los sistemas embebidos por lo general son de naturaleza heterogénea y combinan partes analógicas y digitales, procesamiento de datos a nivel alto, control, partes síncronas y asíncronas, funciones de tiempo real estricto y otras de tiempo real ligero.
A parte de las restricciones citadas anteriormente para los sistemas embebidos, los estándares de comunicación como el GSM, el DECT, el IS-95, el WiFi entre otros, están en constante evolución integrando nuevas aplicaciones de telecomunicación con capacidades multimedia, lo que da por consecuencia que los sistemas multimedia embebidos deben ser flexibles para soportar estas evoluciones. Las nuevas aplicaciones y los estándares que las definen, delegan al desarrollador la complejidad, el cual debe seleccionar la arquitectura y realizar la división de las diferentes partes de la aplicación para esta arquitectura, esta selección sobre la complejidad está basada fuertemente sobre la experiencia del desarrollador.

La arquitectura de sistemas embebidos se ha beneficiado de enormes progresos tecnológicos (una capacidad de integración de más de 200 millones de transistores en 2005), permitiendo integrar sistemas completos en un solo circuito integrado (System on Chip, SoC [7,8]). Con el fin de reducir el esfuerzo de concepción y evitar los costos desorbitantes de la ingeniería no recurrente (Non-Recurring Engineering, NRE) [9], se ha privilegiado cada vez más la reutilización de componentes de una generación a otra. La reutilización puede ser a nivel de sub-bloques del SoC, a esto se le conoce como concepción por ensamblado de IP (Intellectual Property), o a nivel del mismo SoC, a esto se le conoce como concepción basada en plataforma (Platform based design), esto equivale a factorizar el esfuerzo de la concepción para una clase de aplicaciones. Diversas firmas internacionales (Motorola, Zilog, Texas Instruments, Analog Devices, Thomson, Philips, entre otras) han desarrollado estos dos tipos de herramientas para la implementación de sistemas embebidos.

Texas Instruments como líder de la industria, ha generado un ambiente de desarrollo (Platform based design) llamado DaVinci [10], el cual incluye hardware, software y herramientas de desarrollo, estos ambientes de desarrollo DaVinci de TI se les llama módulos de evaluación de video digital (Digital video Evaluation Module, DVEVM) [10]. Los procesadores Media digitales de Texas Instruments usados en estos módulos de evaluación, son procesadores DaVinci multi-núcleo (multi-core) optimizados para los sistemas de video digital, estos son escalables y programables. El portafolio DaVinci incluye procesamiento digital de señales en SoCs, codecs multimedia, proveedores de software autorizados y frameworks, para así ofrecer una plataforma abierta de desarrollo. Todos los procesadores DaVinci están pensados para que su rendimiento y funcionalidad cumplan los requisitos de una amplia gama de equipos digitales de video. DaVinci de TI permite crear un SoC para
Introducción

Capítulo I

Procesamiento de video utilizando la plataforma DaVinci

Aplicaciones de video digital tal como videoteléfonos, IP set-top-box, adaptadores de media digitales, cámaras digitales, información y entretenimiento en el automóvil, cámara de red IP, DVRs, players de media portables, gateways de media, imagen médica, entre otros.

Por lo que se ha mencionado, la complejidad de los sistemas embebidos ha incrementado, haciendo el proceso de diseño cada vez más complejo y requiriendo mucho más tiempo. En este contexto el empleo de herramientas y metodologías estandarizadas ayuda de manera significativa a los diseñadores, desarrolladores, integradores e investigadores a desarrollar sistemas embebidos más complejos en tiempos más cortos. Es en este contexto en el que el proyecto de tesis está enfocado.

I.1 Objetivo de la investigación

El objetivo general de esta tesis es, desarrollar e integrar un codificador MPEG-4 ASP (Advanced Simple Profile) en un entorno embebido utilizando la plataforma DaVinci de Texas Instruments.

I.1.1 Objetivos específicos

Los objetivos específicos del trabajo de tesis son:

- Análisis del estándar MPEG-4 SP y ASP para codificación de video.
- Análisis de la implementación de un codificador MPEG-4 SP en la Plataforma DaVinci DVEVM DM355.
- Implementación de un codificador MPEG-4 ASP en la plataforma DaVinci DVEVM DM355.

I.2 Organización de la tesis

Para el desarrollo del trabajo de tesis se abordaron diferentes temas relacionados a la codificación de video MPEG-4 y al manejo de la plataforma DaVinci DVEVM DM355, por lo que este escrito representa la estructura del trabajo desarrollado. El capítulo 2 presenta los fundamentos de la codificación de video, se explica la necesidad y la utilización de dichos
codificadores en la actualidad. También se presentan las características principales del estándar MPEG-4 versión 2.

En el capítulo 3, se describe la plataforma de desarrollo, la cual es un módulo de desarrollo de Texas Instruments, el DVEVM DM355.

Texas Instruments ha desarrollado su ambiente de desarrollo DaVinci el cual se describe en el capítulo 4, además, en este capítulo se expone un codificador MPEG-4 de código abierto, el cual es considerado como candidato a ser portado a la plataforma de desarrollo.

En base a lo demostrado en el capítulo 4 de la arquitectura DaVinci, en el capítulo 5, se presenta la metodología para implementar de manera modular el código de un codificador MPEG-4 de tal modo que se pueda utilizar con dicha arquitectura. Además, en el capítulo 5 se explica el procedimiento para añadir a la aplicación de procesamiento de video, el manejo de una cámara de captura utilizando el hardware de la tarjeta.

Para medir el desempeño del codificador MPEG-4 ASP Xvid portado a la arquitectura del procesador ARM del DM355 de Texas Instruments, el capítulo 6 se presentan los resultados del desempeño de dicho codificador.

Finalmente, en el capítulo 7 se exponen conclusiones generales del trabajo de tesis.

### I.3 Aportaciones del trabajo de tesis

Las principales aportaciones del trabajo de tesis, se pueden resumir con los siguientes puntos listados a continuación:

- Descripción de la plataforma DaVinci.
- Portabilidad del codificador MPEG-4 ASP Xvid a la arquitectura del procesador ARM del DM355 de Texas Instruments.
- Evaluación del codificador MPEG-4 ASP Xvid en la arquitectura del procesador ARM del DM355 de Texas Instruments.
CAPÍTULO II  CODIFICACIÓN DE VIDEO

Introducción

Los codificadores de video son uno de los componentes claves en las aplicaciones multimedia. Con el fin de evitar la no-interoperabilidad entre un gran número de codificadores, se han propuesto los estándares de codificadores de video tales como el MPEG-4 v2. En este capítulo se describen los fundamentos de la codificación de video y las características principales del estándar MPEG-4 versión 2.

II.1 La codificación del video

Una escena visual (natural o del mundo real) es compuesta por múltiples objetos, cada uno con su forma, profundidad, textura e iluminación. En la adquisición del video digital, una escena es muestreada en un punto del tiempo para producir un cuadro rectangular (una representación de la escena visual completa en ese punto en el tiempo). A su vez, cada cuadro rectangular es compuesto de puntos (o pixeles) que representan una muestra de la intensidad luminosa capturada para una de las localidades del contenido dentro del espacio del cuadro. De tal manera que en la reproducción, el cual consiste del despliegue periódico de las series de cuadros en orden secuencial, se produce la apariencia de una escena en movimiento.

Una vez que el video digital es adquirido, por lo general, su manejo en transmisión y en almacenamiento se vuelve problemático, dado que el tamaño en bytes de dichas secuencias de cuadros es sumamente alto.

La codificación de video, tiene la capacidad de resolver dichos problemas relacionados con almacenamiento o transmisión, al reducir el número de bits por pixel tras procesar el video digital a través de métodos de codificación los cuales explotan la información redundante presente en el video digital.

Entre las características típicas que son relevantes para la compresión de video incluyen características espaciales (variaciones de texturas dentro de una escena, y forma de los objetos, color, etc.) y características temporales (movimiento de objetos, cambios en la iluminación, movimiento de la cámara, punto de vista, etc.).
En el marco de trabajo de tesis se utiliza el estándar MPEG-4 versión 2, debido a que provee de las herramientas con las cuales se pueden crear codificadores (de alta calidad) de audio y de video [11].

II.2 Estándar MPEG-4 versión 2


MPEG-4 se desarrolló basado en tres áreas donde ha tenido éxito:

- Televisión Digital;
- Aplicaciones con gráficos interactivos (contenido sintético);
- Multimedia interactivo (Internet, distribución y acceso a contenido).

El estándar MPEG-4 proporciona los elementos tecnológicos estandarizados que permiten la integración de la producción, de la distribución y del acceso al contenido paradigmas de las tres áreas.

II.3 Alcances y características del estándar MPEG-4

El estándar MPEG-4 proporciona un conjunto de tecnologías para satisfacer las necesidades de los autores, los proveedores de servicios y de los usuarios finales por igual.

- Para los autores, el MPEG-4 permite la producción de contenidos los cuales tienen mayor reutilización y mayor flexibilidad lo que es posible hoy en día con las tecnologías individuales tales como la televisión digital, los gráficos animados, los portales de internet y sus extensiones. Además, ahora es posible administrar el contenido de manera controlada y proteger los derechos de autor.
• Para los proveedores de servicio de red, el MPEG-4 ofrece una información transparente, la cual puede ser interpretada y traducida a los mensajes de señalamiento nativos de cada red con la ayuda de sus estándares apropiados. Lo anterior, sin embargo, excluye las consideraciones de calidad de servicio (QoS, Quality Of Service), por lo que el MPEG-4 provee un descriptor de QoS genérico para los diferentes medias MPEG-4. Debido a que los parámetros de QoS para los diferentes tipos de media, difieren entre los establecidos en el MPEG-4 y las que son especificadas para las redes. La traducción exacta de dichos parámetros se deja a selección de los proveedores del servicio de la red. Entre uno de los beneficios de incluir los descriptores de QoS en media MPEG-4, es que permiten la transmisión optimizada de extremo a extremo en redes heterogéneas.

• Para los usuarios finales, el MPEG-4 proporciona mejores niveles de interacción con el contenido, dentro de los límites establecidos por el autor. También brinda la capacidad de manejar multimedia en las nuevas redes, incluyendo las redes que tienen limitaciones de bajo bitrate, y las redes móviles. Un documento de posibles aplicaciones para MPEG-4 existe en el portal de MPEG [11], el cual describe muchas aplicaciones enfocadas a los usuarios finales.

Para los autores, proveedores de servicios y usuarios finales, el MPEG-4 intenta evitar una multitud de formatos y reproductores propietarios, sin capacidad de interoperabilidad.

MPEG-4 logra estos objetivos al proveer las metodologías estandarizadas para:

1. Representar unidades de contenido auditivo, visual o audiovisual, denominados “objetos de media”. Estos objetos de media pueden ser de origen natural o sintético, esto significa que pueden ser grabados con cámara o con micrófono, o generados por computadora;

2. Describe la composición de estos objetos para crear objetos de media compuesto, los cuales forman escenas audiovisuales;

3. Realizar múltiples tareas y sincronizar los datos asociados con los objetos de media, de tal forma que puedan ser transportados sobre canales de red las cuales proveen una QoS apropiada para la naturaleza de los objetos de media específicos,

4. Interactuar con la escena audiovisual generada en el extremo del receptor.
La escena audiovisual representada en la figura Fig. II.1, ilustra las funcionalidades contenidas en el estándar MPEG-4 anteriormente descritas.

Fig. II.1. Ejemplo de una escena MPEG-4.

II.3.1 Representación codificada de los objetos de media

Las escenas audiovisuales MPEG-4 están compuestas de varios objetos de media, organizados en orden jerárquico. En los extremos más bajos de la jerarquía, se encuentran los objetos de media primitivos, tales como:

- Imágenes Fijas (por ejemplo, un fondo fijo);
- Video objetos (por ejemplo, una persona que habla – sin el fondo);
- Audio objetos (por ejemplo, la voz asociada con esa persona, música de fondo);

El MPEG-4 estandariza un número de tales objetos de media primitivos, capaces de representar ambos tipos de contenido natural y sintético, los cuales pueden ser de 2 o 3
dimensiones. Además de los objetos de media mencionados anteriormente y mostrados en la Fig. II.1, el MPEG-4 define la representación codificada de tales objetos como:

- Texto y gráficos;
- Caras sintéticas que hablan y el texto asociado usado para sintetizar los movimientos y animar la cara, y los cuerpos animados que corresponden a dichas caras;
- Sonido sintético.

Un objeto de media su forma codificada consiste en elementos descriptivos que permiten manejar el objeto en una escena audiovisual al igual que los datos afluentes asociados, si es necesario. Es importante notar que en su forma codificada, cada objeto de media puede ser representado independientemente de su alrededor o fondo.

La representación codificada de objetos de media es tan eficiente como es factible tomando en cuenta las características deseadas. Ejemplos de tales características son la robustez al error, la fácil extracción y edición de un objeto, o tener un objeto a disposición en forma escalable.

**II.3.2 Descripción y sincronización de datos afluentes para objetos de media**

Los objetos de media pueden necesitar datos afluentes, los cuales son transmitidos en uno o más flujos elementales. Un descriptor de objeto identifica todos los flujos asociados con un objeto de media. Esto permite el manejo jerárquico de datos codificados, así como la asociación de meta-información acerca del contenido (la cual se llama “información de contenido de objeto”) y los derechos de propiedad intelectual que se asocian a él.

Cada flujo en sí es caracterizado por una serie de descriptores que contienen la información de la configuración, por ejemplo, para determinar los recursos necesarios para el decodificador y la precisión de los tiempos que se llevan a cabo en la codificación. Además los descriptores pueden acarrear pistas sobre la Calidad de Servicio (QoS) que se requiere para la transmisión (por ejemplo, máxima tasa de bits, tasa de errores de bit, prioridad, etc.).

Sincronización de los flujos elementales se logra a través de un registro de tiempo almacenado en las unidades individuales de acceso los cuales a su vez son contenidos dentro
II.3.3 Entrega de datos afluentes

La entrega sincronizada de datos afluentes desde el origen al destino, en donde se explotan las diferentes QoS que son disponibles en la red, se puede abstraer en términos de una capa de sincronización y una capa de entrega la cual contiene un multiplexor de dos capas, como se muestra en la Figura II.2.

La primera capa multiplexora es manejada de acuerdo a la especificación DMIF, parte 6 del estándar MPEG-4 (Delivery Multimedia Integration Framework). Esta acción de multiplexión puede ser contenida por la herramienta “FlexMux” definida en MPEG, la cual permite el agrupamiento de Flujo Elementales (ES) de bajo consumo computacional. El multiplexor en esta capa puede ser utilizado, por ejemplo, para agrupar ES con requerimientos de QoS similares, reducen el número de conexiones de red o demoras que ocurren en el lapso de un extremo a otro.

La capa “TransMux” (Multiplexación de Transporte) en la Fig. II.2 modela la capa que empareja los servicios de transporte ofrecidos con las QoS solicitadas. Solo la interferencia a esta capa es especificada por el MPEG-4 mientras el mapeo concreto de los paquetes de datos y la señalización de control debe hacerse en colaboración de los entes que tienen derecho sobre el respectivo protocolo de transporte. Cualquier pila existente con protocolo de transporte tal como RTP (Real-time Transport Protocol) / UDP (User Datagram Protocol) / IP (Internet Protocol), AAL5 (ATM Adaptation Layer 5) / ATM (Asynchronous Transfer Mode), o Flujo de Transporte MPEG-2 sobre una capa de enlace adecuada puede convertirse en una instancia específica TransMux. La opción se deja abierta a cada consumidor/proveedor del servicio, y permite que el MPEG-4 se utilice en una variedad de ambientes de operación.
El uso de herramientas multiplexoras FlexMux es opcional y, como se muestra en la Fig. II.2, esta capa puede estar vacía si la instancia TransMux subyacente provee toda la funcionalidad requerida. La capa de sincronización, sin embargo, siempre está presente.

Con respecto a la Fig. II.2, es posible:

- Identificar unidades de acceso, almacenar información de registro de tiempo y del reloj de referencia para el transporte e identificar perdidas de datos.
- Opcionalmente interpolar datos a partir de diferentes flujos elementales hacia flujos FlexMux.
- Transmitir información de control para:
  - Indicar la QoS necesaria para cada flujo elemental y flujo FlexMux,
  - Traducir cada requerimiento QoS a recursos efectivos de red,
  - Asociar los flujos de elementos a objetos de media,
  - Transmitir el mapeo de flujos elementales a canales FlexMux y TransMux.

Partes de las funcionalidades de control son solo disponibles en conjunto con una entidad de control para el transporte como la estructura DMIF.

**II.4 Versiones en MPEG-4**

La versión 1 del MPEG-4 fue aprobada por la organización MPEG en Diciembre de 1998, mientras la versión 2 fue sujeta al mandato de ninguna mejora adicional (parte del
proceso de estandarización) en diciembre de 1999. Después de estas dos versiones principales, más herramientas fueron anexadas bajo enmiendas las cuales se podrían clasificar como versiones, a pesar de que son más difíciles de reconocer como tales. El reconocer las versiones no es tan importante, sin embargo, es aun más importante distinguir entre los perfiles. Las herramientas y perfiles de cualquier versión nunca son remplazadas en versiones subsecuentes, la tecnología siempre se anexa al estándar MPEG-4 en la forma de nuevos perfiles. La Fig. III.3 muestra la relación entre las diferentes versiones. La versión 2 es compatible con la versión anterior, y la versión 3 es compatible con la versión 2, y así sucesivamente. Las versiones de todas las partes principales del estándar MPEG-4 (Sistemas, Audio, Video, DMIF) fueron sincronizadas, después de eso cada parte siguió su propia ruta.

Fig. II.3. Relación entre las versiones de MPEG-4.

Las capas de sistema de versiones posteriores son compatibles con todas las versiones anteriores. En el área de sistemas, audio y visual, las nuevas versiones agregan perfiles, no modifican los existentes. De hecho, es muy importante señalar que los sistemas actuales siempre permanecerán conforme al estándar, porque nunca se modificarán en retrospectiva los perfiles ni los sintaxis de los sistemas, por lo menos no de una manera que los convierta incompatibles con versiones anteriores.
II.5 Funcionalidad Visual en el MPEG-4

El estándar MPEG-4 Visual permite la codificación híbrida de imágenes naturales (basadas en pixeles) y video en conjunto con escenas sintéticas (generadas por computadora). Esto habilita, por ejemplo, la presencia virtual de participantes en video-conferencia. Para este fin, el estándar visual contiene herramientas y algoritmos los cuales soportan la codificación de imágenes fijas naturales (basadas en pixeles) y secuencias de video con sus propias herramientas y soporta la compresión de gráficos sintéticos de parámetros de geometría (por ejemplo, compresión de texto sintético).

II.5.1 Formatos soportados

Los siguientes formatos y tasas de bits son soportados por el estándar MPEG-4 Visual:

- Tasas de bits: típicamente entre 5 kbit/s y más de 1 Gbit/s,
- Formatos de Campo: video progresivo así como video entrelazado,
- Resoluciones: típicamente desde sub-QCIF hasta resoluciones denominados ‘Studio’ (4k x 4k pixeles).

II.5.2 Eficiencia de Compresión

- Para todas las tasas de bits compatibles, los algoritmos dentro de los codificadores/decodificadores son muy eficientes. Esto incluye la codificación compacta de texturas con calidad ajustable entre “aceptable” para tasas de compresión muy altas hasta “casi sin-perdidas”.
- Compresión eficiente de texturas para mapeo de texturas sobre mallas 2-D y 3-D.
- Acceso aleatorio al video para permitir funcionalidades tales como pausa, adelanto y atraso de video almacenado.

II.5.3 Robustez en ambientes propensos a errores

Robustez al error (error resilience) permite acceder a las imágenes y video sobre un amplio rango de medios para la transmisión y almacenamiento. Esto incluye la operación robusta de algoritmos de codificación de imagen y codificación de video en ambientes
propensos a errores a bajos bitrates (es decir, menos de 64 Kbps). Además, hay herramientas que abordan tanto la naturaleza de ancho de banda limitada como los aspectos de robustez al error para el acceso sobre redes inalámbricas.

II.6 Perfiles en MPEG-4

EL MPEG-4 provee un amplio conjunto de herramientas para la codificación de objetos audio-visuales. Dado la necesidad de proporcionar implementaciones efectivas con el uso del estándar, subconjuntos de los sistemas MPEG-4, en forma de conjuntos de herramientas visuales y de audio han sido identificadas, las cuales pueden ser utilizadas para aplicaciones específicas. Estos subconjuntos, llamados “Perfiles”, limitan el conjunto de herramientas que un decodificador necesita implementar. Para cada uno de estos perfiles, uno o más niveles han sido establecidos, los cuales restringen la complejidad computacional. Esta manera propuesta es similar a la del estándar MPEG-2. Un perfil en combinación con un nivel dado permite:

• Que desarrolladores de codificadores implementen solo el subconjunto del estándar que necesiten, mientras se mantiene la compatibilidad con otros dispositivos MPEG-4 desarrolladas bajo la misma combinación, y
• Revisar si los dispositivos MPEG-4 cumplen con el estándar (“pruebas de conformidad”).

Los perfiles existen para varios tipos de contenido (audio, visual y gráficos) y para descriptores de escena. EL MPEG no pre-establece ni asesora la combinación de estos perfiles, pero ha tomado cuidado que existan buenas opciones a escoger entre las diferentes áreas de aplicación.

La parte visual del estándar provee perfiles para la codificación de contenido visual de tipo natural, sintético, e híbrido sintético/natural. Las combinaciones más conocidas de perfiles/niveles son: Simple Visual Profile y Advanced Simple Visual Profile.

El “Simple Visual Profile” provee codificación de objetos rectangulares de video con eficiencia y robustez al error, adecuada para aplicaciones sobre redes móviles, tales como PCS e IMT2000.

El “Advanced Simple Profile” muy similar al Simple dado que solo es destinado a objetos rectangulares de video, pero contiene otras herramientas adicionales, los cuales la hacen más eficiente: Los cuadros Bi-direccionales, compensación de movimiento con
precisión de \(\frac{1}{4}\) de pixel, utilización de tablas de cuantificación adicionales y compensación de movimiento global.

En este trabajo de tesis se trabajó con estos dos perfiles, por lo que en las secciones siguientes se describen con más detalle.

### II.6.1 MPEG-4 Perfil Simple (SP)

El perfil Simple fue creado para aplicaciones que requieren codificación potente con bajo poder computacional, poca latencia en la codificación y transmisión sub-óptima. Destaca en aplicaciones de codificación en tiempo real para equipos móviles y dispositivos inalámbricos, video telefonía, video conferencia y sistemas de video vigilancia.

Dentro de las herramientas de codificación contenidas en el perfil Simple, existen herramientas que dictan la manera de codificar un cuadro dentro de una secuencia de video.

El primero de estas herramientas, establece las diferentes etapas de procesamiento que se utiliza para la generación de un cuadro codificado, tomando en cuenta solo la información contenida dentro del mismo cuadro. Al cuadro codificado que resulta de esta operación se le conoce como cuadro Intra (o también como cuadros “I”). Por lo general son los primeros cuadros a ser codificados, los que se repiten a un intervalo establecido en el codificador o los que se generan cuando se detecta un cambio abrupto en la secuencia de video (ejemplo: cambio de escena).

La herramienta de generación de cuadros Intra se representa de manera general en un diagrama de cinco bloques (figura II.4).

Como punto de partida, se necesita disponer de un cuadro sin codificación el cual es obtenido de la secuencia de video, esto se representa en el diagrama de la figura II.4 con el bloque denominado “Cuadro F\(_n\)”.

Continuando con el bloque denominado “DCT” en la figura II.4, donde se representa la etapa de transformación de la información contenida en el cuadro desde el dominio espacial al dominio frecuencial, mediante el operador de dos-dimensiones conocido como la DCT (Discrete Cosine Transform). Con la finalidad de lograr mejor eficiencia en el proceso de codificación, la DCT se aplica en unidades de bloques de pocos píxeles conocidos como macrobloques para todo el cuadro a codificar.
En el siguiente bloque del diagrama denominado “Cuantificación”, partiendo del macrobloque resultante de la DCT, donde se tienen los coeficientes representativos de la información frecuencial. En este, se atenúan los coeficientes de las altas frecuencias a niveles cercanos a cero, mientras se preservan los coeficientes de las bajas frecuencias. Esto se permite, debido a que la mayoría de la pérdida de información de las altas frecuencias en el cuadro, pasará desapercibido por el observador humano.

Después se realiza la cuantificación, el cual es una operación no-reversible que consiste en emparejar cada coeficiente resultante del macrobloque a un rango de valores correspondiente. Dichos rangos de valores son consecutivos y cada rango a su vez es representado por un número, de tal forma que es posible reducir todos los valores de los coeficientes a un número finito de valores enteros.

En el bloque del diagrama denominado “Codificador Entrópico”, se refiere a un esquema de codificación que implica la asignación de códigos por símbolos en una tabla de longitud variable, donde la tabla ha sido rellenada de una manera específica basándose en la probabilidad estimada de aparición de cada posible valor de dicho símbolo. De tal manera, que los símbolos más comunes a utilizar son los más cortos en la tabla de longitud variable.

La representación final del cuadro codificado es un flujo de bits codificado, el cual contiene la información de los coeficientes del cuadro codificado e información de cabecera.

Fig. II.4: Etapas de Codificación de un cuadro Intra.

En el dominio temporal, existe una gran similitud entre cuadros de video que son adquiridos consecutivamente. Estos cuadros temporalmente adyacentes (cuadros sucesivos en
el orden del tiempo) comparten mucha información, especialmente si la tasa de muestreo temporal (cuadros por segundo) es alto.

Por tal motivo, la herramienta de codificación de cuadros “Inter” (o también conocido como cuadros “P”) contenido en el perfil simple, busca lograr una codificación donde se explota dicha redundancia temporal, tomando en cuenta las similitudes entre cuadros de vecinos. En la codificación de cuadros Inter (cuadro “P”), por lo general se puede lograr la codificación con el 50% de los datos requeridos para almacenar un cuadro codificado. Por esta razón, los cuadros Inter tienden a prevalecer en un esquema de codificación destinado a lograr mayores tasas de codificación para una secuencia de video codificada conforme al perfil Simple.

En esencia la mayoría de los procesos que son utilizados para la codificación en los cuadros Intra (ver Fig. II.4), se vuelven a aplicar en la codificación de cuadros Inter (ver Fig. II.5), con la excepción de nuevos procesos que son introducidos aumentando la complejidad del sistema. El diagrama de los procesos (figura II.5) comienza con la toma de entradas, las cuales consisten en el cuadro sin codificación que se desea codificar (“F_n”, figura II.5) y un cuadro de referencia (“F_{n-1}”, figura II.5) que previamente ha sido codificado de la secuencia de video. El cuadro de referencia que se utiliza puede ser un cuadro Intra (cuadro “I”) u otro cuadro Inter (cuadro “P”).

Debido a que la mayoría de las variaciones entre los cuadros de video son causadas por movimientos de objetos, es deseable conocer en qué medida se presentan dicho movimientos. Por consiguiente, en el proceso representado en el diagrama de la Fig.II.5 por el bloque “Estimación de Movimiento”, se secciona el cuadro previamente codificado en un conjunto de macrobloques que constituyen la totalidad del cuadro. Para cada uno de ellos, se busca (limitada por un rango de búsqueda) la región dentro del cuadro a codificar que comparte mayor similitud con el macrobloque. El resultado de cada búsqueda se almacena en un “vector de movimiento” que representa la posición de la región candidata con respecto al punto de referencia (del macrobloque).

En el siguiente proceso representado por el bloque “Compensación de Movimiento” se busca corregir el movimiento entre cuadros, utilizando desplazamiento en base al los vectores de movimiento registrados para cada macrobloque en el cuadro previamente codificado (o de referencia). Esto con el fin de aumentar la similitud entre los cuadros.
El cuadro previamente codificado (o de referencia) el cual ha sido compensado en movimiento y el cuadro de video a codificado son restados entre ellos para formar un nuevo cuadro la cual contiene poca energía denominado “cuadro residual”. Este cuadro residual se entrega al los resto del sistema que contiene los procesos comunes con la codificación de cuadros Intra (DCT, Cuantificación y Codificador Entrópico, Figura II.5)

La representación final del cuadro codificado es un flujo de bits codificado, el cual contiene la información de los coeficientes del cuadro residual, la información de los vectores de movimiento e información de cabecera [1].

\[\text{Cuadro } F_{r} \rightarrow \text{Cuadro } F_{p} \rightarrow \text{Estimación de Movimiento} \rightarrow \text{Compensación de Movimiento} \]
\[\text{DCT} \rightarrow \text{Cuantificación} \rightarrow \text{Codificador Entrópico} \rightarrow \text{Cuadro } F_{p} \]

**Fig. II.5: Etapas de codificación de un cuadro Inter**

En la Fig. II.6 observamos el orden de codificación del Perfil Simple entre los cuadros Intra (“I”) e Inter (“P”). Esta secuencia se repite mediante parámetros establecidos de refrescamiento de cuadro Intra, o en su caso cuando el codificador detecta un cambio abrupto en la secuencia.

**Fig. II.6: Orden de codificación de cuadros en Perfil Simple.**
II.6.2 MPEG-4 Perfil Avanzado Simple (ASP)

El perfil Advanced Simple (ASP) ofrece desempeño de codificación que puede ser utilizado en ambientes exigentes, donde se requiere codificación a bitrate intermedios y altos. Entre las aplicaciones de codificación de video donde destaca el MPEG-4 ASP, son la difusión para televisión, internet y equipos móviles [13]. Además, proporciona la manera de mejorar la codificación de video en aplicaciones que implementan codificación MPEG-4 perfil Simple (SP).

Dentro de las herramientas de codificación contenidas en el MPEG-4 ASP, existen herramientas que dictan la manera de codificar un cuadro dentro de una secuencia de video. El MPEG-4 ASP cuenta con codificación de cuadros Intra (cuadros “I”) y codificación de cuadros Inter (cuadros “P”), los cuales también son contenidos en SP. Además, en el MPEG-4 ASP se introduce la herramienta de codificación de “cuadros bi-direccionales” (también conocidos como cuadros “B”). En la codificación de un cuadro bi-direccional, por lo general se puede lograr la codificación con el 25% de los datos requeridos para almacenar un cuadro codificado por Intra (cuadro “I”). De manera similar a la codificación de cuadros Inter (cuadros “P”), esta herramienta codifica un cuadro de video, tomando la información adicional contenida en dos cuadros de referencia, los cuales han sido previamente codificados, tales cuadros de referencia pueden ser Intra (cuadros “I”) o Inter (cuadros “P”), sin embargo otro cuadro bi-direccional (cuadro “B”) no puede ser tomado como cuadro de referencia.

Con respecto a los procesos que lleva a la codificación de un cuadro bi-direccional (representado con el diagrama de la Figura II.7), son los mismos procesos que se utilizan para la codificación de cuadros Inter (ver Fig. II.5), pero con ligeras variaciones. La diferencia más notables es la introducción de un cuadro de referencia adicional.

El diagrama de los procesos de la codificación de un cuadro bi-direccional (figura II.5) comienza con la toma de entradas, las cuales consisten en un cuadro que se desea codificar (“F_n”, figura II.5) y dos cuadros de referencia (“F_{n-1}” y “F_{n+1}”, figura II.5) que previamente han sido codificados a partir de la secuencia de video.

En el siguiente proceso de “Estimación de movimiento”, difiere a la realizada en la codificación Inter, la variación en este proceso es que la búsqueda por la mejor pareja de cada macrobloque se realiza en ambos cuadros de referencia (dentro de un rango limitado de búsqueda). En el proceso que continua, se “Compensa el Movimiento”, tomando en cuenta
“vectores de movimiento” para cada macrobloque. Cada uno de ellos puede referirse a una región dentro de solo uno de los cuadros de referencia. Para continuar con los procesos comunes con la codificación Intra e Inter, se entrega el cuadro residual formado a partir de la resta entre los cuadros de referencia compensados en movimiento y el cuadro que se desea codificar.

Al finalizar los procesos que llevan a la codificación un cuadro bi-direccional (cuadro “B” en la figura II.7), se obtiene un flujo de bits codificado, el cual contiene la información de los coeficientes del cuadro residual, la información de los vectores de movimiento e información de cabecera.

Con el objetivo de proporcionar un ejemplo sencillo, en la figura II.8 se presenta una de las posibles secuencias de cuadros codificados bajo el MPEG-4 ASP. Es notable mencionar, que un cuadro bi-direccional no está ligado a tomar referencia de cierta combinación de cuadros futuros o previos con respecto al orden temporal. La decisión se tomará en base a cuál de los cuadros de la secuencia de video presenta la mejor similitud. Lo que sí es pertinente, es que un cuadro “B” debe de hacer referencia solamente a cuadros “I” o “P”, no está permitido tomar referencia de otro cuadro “B”. Esto se aprecia con las flechas de la figura II.8 (“adelanto” y “atraso”).
II.7 Conclusiones

El estándar MPEG-4 contiene herramientas para la codificación, muchas de estas herramientas de codificación se interpretan como procesos computacionalmente exhaustivos al aplicarlos dentro de un procesador. Por esta razón, en ocasiones es necesario limitar el uso de ciertas herramientas conforme a las capacidades que brinda un sistema computacional, esto especialmente para los sistemas que manejan codificación en tiempo real.
CAPÍTULO III  PLATAFORMA DE DESARROLLO

En la actualidad los sistemas electrónicos son el resultado de una convergencia de diferentes tecnologías. Por lo tanto es obligatorio contar con una combinación de experiencia, de técnicas, de herramientas y de una red de contactos para cumplir con las expectativas del mercado.

Las plataformas de desarrollo, ayudan a los desarrolladores a diseñar y construir prototipos para una gran variedad de productos embebidos para telecomunicaciones y redes, tales como dispositivos de seguridad, tarjetas de telecomunicaciones, aplicaciones de control y automatización industrial.

III.1 Tecnología DaVinci

La tecnología DaVinci [10] es una solución basada en el procesamiento de señales, adaptada para aplicaciones de video digital. Esta tecnología provee a los constructores de equipos de video con procesadores integrados, software, herramientas y soporte técnico para simplificar el proceso de diseño y acelerar la innovación.

La línea DaVinci consiste en procesadores escalables, sistemas-en-chip (system-on-chip, SoCs) programables para procesamiento de señales, aceleradores de hardware y periféricos; optimizados para su rentabilidad, desempeño y características en una amplia gama de equipos de video.

La generación DM3x cuenta con un procesador ARM9 [14] con velocidades de hasta 300 MHz, con algunos dispositivos de co-procesamiento de video y con un paquete de codificadores propietarios para acelerar el desarrollo; todo ello bajo un subsistema de procesamiento-video que agrega inteligencia al procesamiento de video.

La generación DM3x, incluyendo los procesadores DM355 [15], son ideales para aplicaciones de cámaras digitales, dispositivos de reproducción de video como cámaras IP, timbres de video, videoconferencia, desplegados digitales, reproductores multimedia portátiles entre muchos más.
III.2 System-On-Chip

El término sistema-en-chip o sistemas-en-chip (System-on-Chip SoC o Systems-on-chip SOC) se refiere a la integración de todos los componentes de una computadora u otro sistema electrónico dentro de un solo circuito integrado (chip) [16].

El desarrollo de sistemas en un solo circuito integrado es uno de los mayores desafíos que los ingenieros han enfrentado. Los SoC son el resultado de una mezcla de microprocesadores, memorias, buses, arquitecturas, estándares de comunicación, protocolos de procesadores, interfaces y otros componentes de propiedad intelectual (Intellectual Property), donde las consideraciones a nivel sistema tales como la optimización de energía, la sincronización, la capacidad de prueba, la conformidad y la verificación son cruciales. Una aplicación típica de los SoC es en el área de sistemas embebidos [16].

El contraste con un micro-controlador es una cuestión de grado. Los micro-controladores típicamente tienen menos de 100 KBytes de memoria RAM (a veces unos pocos KBytes) y en muchos casos son sistemas de un solo chip; mientras que el término SoC se suele utilizar con procesadores más potentes, capaces de ejecutar software como Windows o Linux, los cuales necesitan chips de memoria externa (flash, RAM) y además son utilizados con diferentes periféricos externos.

Un SoC se constituye de hardware y de software, ambos controlan al micro-controlador, al micro-procesador o núcleos DSP, a los periféricos y a las interfaces. El desarrollo del hardware y del software en el proceso de diseño de un SoC se realiza en paralelo [16].

III.3 Sistemas embebidos para codificador MPEG-4


Para aprovechar el poder de procesamiento del procesador DM355, en este trabajo de tesis, se ha implementado el estándar de compresión MPEG-4 ASP (Advanced Simple Profile) [13] para el procesador ARM y ha sido todo un reto. La arquitectura del DM355 es ideal para aplicaciones conformes al estándar MPEG-4, debido a su paralelismo ésta se puede utilizar
III.4 Módulo de evaluación DVEVM TMS320DM355

El módulo de evaluación de video digital (Digital Video Evaluation Module, DVEVM) es una plataforma de evaluación que muestra la capacidad de la arquitectura DaVinci, permitiendo a los desarrolladores evaluar el poder y desempeño de los procesadores DaVinci como núcleo central para el procesamiento de contenido de multimedia.

El módulo de evaluación DVEVMD355 está basado en el procesador de Texas Instruments TMS320DM355 [17]. El módulo de evaluación DM355 permite a los ingenieros y desarrolladores de software evaluar ciertas características del procesador DM355 para determinar si el procesador cumple con los requerimientos de la aplicación a desarrollar. Los desarrolladores pueden crear software para ejecutarse en la tarjeta o expandir el sistema en una variedad de formas. El módulo de evaluación DM355 también se le conoce como DM355 EVM o EVM.

III.4.1 DM355 EVM

El EVM viene con un completo complemento de dispositivos en la tarjeta que se adaptan a una amplia variedad de aplicaciones (ver Fig. III.1). Las características importantes son:

- Procesador DM355 de Texas Instruments con un procesador ARM que opera hasta frecuencias de 216 MHz,
- 1 puerto de entrada de video para video compuesto o súper Video (S-video),
- 1 salida de video compuesto (DAC de 10bits),
- Memoria externa de 128 Mbyte DRAM DDR2,
- Interface de comunicación UART, 2 Interfaces de tarjeta SD/MMC o 1 SD/MMC y 1 interface de Disco CE-ATA,
- Unidad de almacenamiento Memoria NAND Flash con capacidad de 2GBytes,
- 1 códec para audio estéreo AIC33,
- Interface USB2,
- Memoria asignada controlador Ethernet con capacidad de transmisión 10/100Mbs,
- EEPROM SPI,
Capítulo III
Procesamiento de video utilizando la plataforma DaVinci

- Micro-controlador MSP430 programado como interface de Control Remoto IR y Reloj de tiempo real,
- Opciones de arranque configurables para el sistema operativo,
- 8 LEDS para usuario/Interruptor DIP de 4 posiciones para usuario/5 botones de pulso para usuario,
- Fuente de alimentación de voltaje (+5VDC),
- Conectores para tarjetas de extensión (cámara HD),
- Interfaces JTAG de TI (14 Pins)/ JTAG ARM (20 pins).

El DM355 se comunica con los periféricos en el EVM a través del bus de la interface EMIF (External Memory InterFace) de 8/16-bit. La memoria DDR2 se conecta al DM355 a través de un bus dedicado de 16 bits. El bus EMIF se conecta con la unidad de almacenamiento NAND FLASH y el controlador de Ethernet DM9000A (ver Fig. III.1).

El TVP5146 (ver Fig. III.1) en el EVM se utiliza para convertir el video análogo de entrada a una ristra de video digital dirigido al procesador DM355. Posteriormente, el video de salida digital se reconvierte al formato análogo a través de un DAC interno al DM355. Un circuito integrado que convierte el formato de la entrada de video y un DAC interno al DM355 son las
configuraciones estándar en el EVM. Para las funciones de despliegue en pantalla, estas son implementadas en software para el procesador DM355.

El circuito integrado códec AIC33 en la tarjeta permite al DSP transmitir y recibir señales analógicas de audio. El bus I2C es utilizado como la interface de control para el códec, mientras que el McBSP controla la ristra de audio. Las interfaces hacia las señales de audio se hacen a través de conectores de audio de 3.5mm, los cuales corresponden a entradas para micrófono, salidas para auriculares, entrada de línea y salida de línea.

El EVM incluye 8 LEDs para usuario, un interruptor DIP de 4 posiciones, una interface IR, un reloj de tiempo real con 5 botones de pulso para proveer al usuario una interacción con la aplicación. Estas interfaces son implementadas a través de software en el micro-controlador MSP430 y son acezadas para lectura y escritura por los registros I2C.

Una fuente de alimentación de +5Vdc es usada para alimentar a la tarjeta. Reguladores de voltaje en la tarjeta proveen el voltaje de +1.3Vdc al procesador, el voltaje de +3.3Vdc para los periféricos y el voltaje de +1.8Vdc para la memoria DDR2. La tarjeta se mantiene en RESET por el micro-controlador MSP430 hasta que las fuentes de poder están dentro de las especificaciones de operación.

III.4.2 Mapeo de memoria

El procesador DM355 tiene un byte (8 bits) para direccionar el espacio de direcciones. Existen ciertas limitaciones para direccionar el byte, las cuales son determinadas por el periférico de interconexión al DM355. El código del programa y los datos se pueden colocar en cualquier parte del espacio unificado de direcciones. Las direcciones son de varios tamaños en función de la aplicación del hardware.

El mapa de memoria muestra el espacio de direcciones de un procesador DM355 genérico, a la izquierda los detalles específicos de cómo se usa cada región de la derecha (figura III.2). De forma predeterminada, la memoria interna se encuentra al principio del espacio de direcciones. Algunas partes de la memoria pueden ser reasignadas en software utilizando la memoria cache L2 en lugar de RAM fijo. Esta parte incorpora una interface dual EMIF. Una interface EMIF dedicada directamente a la memoria DDR2. La otra EMIF tiene 2 regiones direccionables separadamente llamada chip de espacios habilitados (CE0 &CE1). La NAND
Flash y el controlador de Ethernet son asignados a estos chips de espacios habilitados como se observa en la figura III.3, donde se detallan los espacios de memoria.

III.4.3 Interruptores de configuración

El EVM tiene un interruptor de configuración, que permite a los usuarios controlar el estado de funcionamiento del procesador cuando es reiniciado y determina la fuente del software para el arranque del procesador. El interruptor SW7 configura el modo de arranque que se utiliza cuando el DSP inicia a ejecutarse. Por defecto, los interruptores están
configurados para iniciar su ejecución de la memoria NAND Flash. El interruptor de configuración del EMIF se debe establecer correctamente como se muestra en la Tabla III.1.

<table>
<thead>
<tr>
<th>Pos 2</th>
<th>Pos 1</th>
<th>Boot Pin BTSEL1/BTSEL0</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>ON</td>
<td>ON</td>
<td>0 0</td>
<td>NAND boot CE0 *</td>
</tr>
<tr>
<td>ON</td>
<td>OFF</td>
<td>0 1</td>
<td>Not Supported</td>
</tr>
<tr>
<td>OFF</td>
<td>ON</td>
<td>1 0</td>
<td>Boot from SD/MMC</td>
</tr>
<tr>
<td>OFF</td>
<td>OFF</td>
<td>1 1</td>
<td>Boot from UART</td>
</tr>
</tbody>
</table>

* Configuración por defecto.

III.4.4 Componentes en la tarjeta

Esta sección describe el funcionamiento de los componentes principales de la tarjeta DM355 EVM.

III.4.4.1 Interfaces EMIF

Un EMIF separado de 16 bits con dos chips habilitados, permite dividir el espacio de direcciones y permite el acceso asíncrono en el EVM.

III.4.4.1.1 Flash, NAND Flash, interface Ethernet

El DM355 tiene 2GB de memoria NAND Flash y una interface memoria de la interface Ethernet direccionada en los espacios CE0 y CE1. La memoria NAND Flash es usada principalmente para cargar el inicio del sistema y el sistema de archivos de Linux (sistema operativo usado en este trabajo de tesis). En el DM355, los espacios CE0 y CE1 son configurados en 8 y 16 bits respectivamente.

III.4.4.1.2 Interface de memoria DDR2

El DM355 incorpora un bus dedicado de memoria DDR2 de 16 bits. El EVM utiliza un gigabit de memoria de 16 bits para este bus, para un total de 128 megabytes de memoria para
el programa, datos y almacenamiento de video. El controlador DDR interno utiliza un PLL (Phase Lock Loop) para controlar los tiempos de la memoria DDR. El refresco de la memoria DDR2 es manejado automáticamente por el controlador DDR interno del DM355.

III.4.4.1.3 Interface tarjeta de memoria

El EVM es compatible con dos interfaces de tarjetas multimedia SD/MMC. El segundo conector SD está conectado en paralelo con el conector de la unidad de disco CE-ATA.

III.4.4.1.4 Interface UART


III.4.4.1.5 Interface USB

El DM355 incorpora un chip controlador de USB II. Esta interface se lleva a cabo con un mini conector A/B con su propio regulador de voltaje. Dos puentes se proporcionan para hacer al periférico anfitrión (host) y al USB flexibles en la interface.

III.4.4.2 Interfaces de entrada del puerto de Video/Imagen

El DM355 EVM acepta la captura del video a través de los dispositivos internos a los puertos de video. El circuito integrado TVP5146 de Texas Instruments se utiliza para decodificar la entrada del video compuesto o S-video. El conector J11 se utiliza para las entradas de S-video, mientras que el conector J2 se utiliza para las entradas de video compuesto en el EVM.

El Puerto de entrada también puede ser utilizado con pantallas de LCD conectadas a los conectores J30 y J31. La fuente de captura o el codificador se pueden seleccionar a través del software de control en el bus I2C al acceder a los registros del micro-controlador MSP430.
III.4.4.2.1 El Chip DAC de salida de video

El DM355 incorpora un DAC de salida de video compuesto para conectar a diferentes estándares de salida de video. La salida del DAC se filtra y se le dirige al conector RCA (J4).

III.4.4.3 Interface de Ethernet

El DM355 EVM incorpora un circuito integrado Intel DAVICOM DM9000A controlador de Ethernet, esta interface se conecta al EMIF del DM355. La interfaz con capacidad de 10/100 Mbits es aislada y extraída al conector estándar de Ethernet RJ-45 (puerto P5). La dirección de Ethernet establecida para el EVM se almacena en el circuito integrado DM9000A en su SPI ROM durante la fabricación.

El conector RJ-45 tiene 2 LEDs integrados en el conector. Los LED son de color verde y amarillo, e indican el estado de la conexión Ethernet. El LED verde cuando está encendido, indica que se ha establecido una conexión y cuando parpadea indica actividad en la conexión. El LED amarillo cuando se ilumina, indica que opera en la modalidad dúplex completo.

III.4.4.4 Interface \( \text{I}^2\text{C} \)

El bus \( \text{I}^2\text{C} \) en el DM355 es ideal para interconectar los registros de control de muchos dispositivos. En el DM355 EVM el bus \( \text{I}^2\text{C} \) se utiliza para configurar el decodificador de video, el codificador para formato estéreo, expansions de entrada/salida y comunicarse con el micro-controlador MSP430. La memoria ROM \( \text{I}^2\text{C} \) también contiene su interfaz a través del bus serial. El formato del bus es mostrado en la siguiente figura III.4.

\[
\begin{array}{cccccccc}
\text{Start} & \text{Slave Address} & \text{W} & \text{ACK} & \text{Sub Address} & \text{ACK-S} & \text{Data} & \text{ACK-S} & \text{Stop} \\
\end{array}
\]

**Write Sequence**

\[
\begin{array}{cccc}
\text{Start} & \text{Slave Address} & \text{R} & \text{Data} & \text{STOP} \\
\end{array}
\]

**Read Sequence**

Fig. III.4 Formato del bus \( \text{I}2\text{C} \).

Las direcciones de los periféricos para los dispositivos en la tarjeta se muestran en la tabla III.2.
III.4.4.4.1 MSP430

El DM355 EVM incorpora un control remoto infrarrojo, un reloj de tiempo real y entradas/salidas de bits a través de un micro-controlador MSP430. La interface I2C se utiliza en el procesador DM355 para comunicarse con el MSP430. El MSP430 actúa como un dispositivo esclavo en el bus I2C.

III.4.4.5 Interfaces para tarjetas secundarias

El EVM provee conectores que pueden ser utilizadas para aceptar una tarjeta secundaria para esta plataforma. La tarjeta secundaria permite a los desarrolladores ampliar en su plataforma EVM las funcionalidades y proporcionar las entradas y salidas específicas para la aplicación deseada del cliente. Los conectores para tarjeta secundaria son las interfaces que incluyen McBSP (Multichannel Buffered Serial Port) y expansión serial de entrada/salida. Las señales EMIF que no son utilizadas, pueden ser extraídas como señales GPIO (entradas/salidas de propósito general). La salida del puerto de video se lleva a cabo en la interface de la tarjeta secundaria junto con interfaces de entrada/salida y del dispositivo de captura digital.

III.4.5 Instalación del software DVEVM

Para el desarrollo del trabajo de tesis, se instalo el ambiente de desarrollo DVEVM en la plataforma de trabajo.

El software DaVinci provee códec's de audio y video, interoperables, optimizados en código y preparados para la producción. Además, estos códec’s aprovechan los aceleradores en hardware integrados. Estos códec’s están construidos en marcos de trabajo configurables y
son presentados por medio de API’s publicados en los sistemas operativos más populares (como Linux) para la rápida implementación del software.

Texas Instruments, en acuerdo con Montavista Software Inc; ofrecen la versión de demostración de su sistema operativo para embebidos denominado Linux Professional Edition v5.0 en conjunto con sus herramientas en software para el desarrollo [18].

El kit de base del DVEVM incluye esta versión de demostración. La versión de demostración es un subconjunto del software que Montavista proporciona en la edición profesional completa.

III.4.5.1 Componentes de software

La siguiente figura muestra los componentes de software utilizados para el desarrollo de aplicaciones con el kit DVEVM.

![Diagrama de Componentes de Software](image)

**Figura III.5. Componentes de Software.**

En la figura III.5, todo se ejecuta en el procesador ARM. La aplicación se encarga de las entradas/salidas y el procesamiento de la aplicación. Para procesar video, imágenes, voz y señales de audio, se utilizan el API denominado VISA el cual es dado por el Codec Engine. El Codec Engine a su vez utiliza los codec’s basados en xDM.
III.4.5.2 Instalación del software

Para usar la plataforma de desarrollo, se instaló el DVSDK (DaVinci Software Development Kit) usado por el DVEVM, el cual nos permite tener la plataforma lista para el desarrollo del trabajo de tesis; en las secciones siguientes se describe las partes que se instalaron de manera general.

III.4.5.2.1 Instalación de Linux para la plataforma EVM

La instalación de Linux es una versión de MontaVista Linux Pro v5.0 de evaluación, la cual permite trabajar con la plataforma DaVinci y desarrollar el trabajo de tesis. El sistema operativo Linux que se usa en la plataforma de desarrollo para manejar el DVEVM es el Ubuntu Linux v12.

III.4.5.2.2 Instalación del software DVSDK

La instalación del software DVSDK se hace usando el instalador Linux. El software DVSDK incluye los componentes del Codec Engine, los ficheros de datos con ejemplos, los ficheros cabecera xDAIS y xDM, y un asignador de memoria continua para Linux (CMEM).

III.4.5.2.3 Instalación de archivos Audio/Vídeo de demostración

El software DVSDK contiene algunos ficheros de demostración de audio y video; estos ficheros se instalan siguiendo los pasos del instalador Linux.

III.4.5.2.4 Instalación de LPTB

El software DVSDK contiene el Linux performance test bench (LPTB) para usar en la tarjeta.
III.4.5.3 Exportando el sistema de archivos compartido para el acceso con la plataforma

Aunque la memoria NAND Flash en la tarjeta contiene un sistema de archivos, es más conveniente durante el desarrollo tener el NFS (Network File System) de la tarjeta de desarrollo montado como un sistema de archivos en la estación de trabajo anfitriona (donde se realiza el trabajo en Linux).

Antes de que la tarjeta de desarrollo pueda cargar el sistema de archivos, se debe exportar ese sistema de archivos a la estación de trabajo anfitriona de Linux. El sistema de archivos utiliza un servidor NFS. El sistema de archivos exportado contendrá el sistema de archivos de la tarjeta de desarrollo y sus programas ejecutables.

Un NFS es un daemon (aplicación que se ejecuta de manera invisible a los usuarios) que permite compartir archivos entre máquinas en una red de tal forma: como si los archivos fueran localizados en el disco duro de la máquina del usuario [19].

NFS es útil para compartir directorios de archivos entre múltiples usuarios dentro de la misma red. Utilizando este ambiente, será posible desarrollar los programas en ANSI “C”, además de depurar y compilar, para finalmente lanzar el ejecutable en el DVEVM para la aplicación deseada.

III.5 Conclusiones

La literatura sobre la plataforma de desarrollo EVM, muestran la posibilidad de crear una gran variedad de aplicaciones para el manejo del video digital.

El software proveído en el DVSDK que acompaña el DVEVM permite utilizar un sistema operativo, herramientas de programación como compiladores y depuradores, además de los componentes en software para los dispositivos en hardware específicos para esa plataforma. Esto permite enfocarse más a programar en el ambiente.
CAPÍTULO IV  DAVINCI DE TEXAS INSTRUMENTS

Este capítulo explica la contribución del software contenido en el DVSDK y los estándares de TI que se utilizan para llevar a cabo una aplicación de codificador MPEG-4 SP de TI para el DM355 en la plataforma DaVinci. Con el objetivo de analizar el funcionamiento del codificador de video, el cual ya implementa tanto los estándares y el software relacionado.

En este capítulo se introducirá el funcionamiento general del codificador MPEG-4 ASP del proyecto Xvid, con el objetivo de analizar un codificador de código abierto que es conforme al estándar MPEG-4 ASP. Este codificador será utilizado para verificar su configuración y funcionamiento bajo un ordenador convencional antes de portarlo a la arquitectura del procesador ARM bajo el SO Linux.

IV.1 Arquitectura DaVinci

TI generaliza a dos tipos de desarrolladores que trabajan sobre las plataformas DaVinci [20], el primero de estos desarrolla aplicaciones y su trabajo consiste en integrar todos los diferentes componentes de código del sistema embebido (esto incluye los codificadores disponibles para la plataforma DaVinci) para implementar un dispositivo o producto final. Mientras el segundo tipo desarrollada codificador es conforme a cierto estándar con código optimizado para la plataforma DaVinci para ser utilizado por un desarrollador de aplicaciones. En base a los roles que desempeñan estos dos tipos de desarrolladores, TI diseñó el software para satisfacer las necesidades de un diseño con menor tiempo de desarrollo y enfocado a desarrollo colaborativo. De tal forma que permite a ambos enfocarse en el área del sistema sobre la cual son expertos, mientras que dejan que la arquitectura en software proporcionado por TI se encargue de administrar los detalles específicos de la plataforma que pueden cambiar entre sistemas operativos y entre arquitecturas de los procesadores DaVinci.

En base a lo mencionado, la arquitectura DaVinci de TI consiste en componentes modulares de software que administran recursos (y los detalles específicos del sistema) cuando entran en juego codificadores (conforme al estándar TI) en la aplicación. Los codificadores se llaman en la aplicación mediante APIs estandarizados que proporcionan la
interface con los diferentes módulos de código que contienen las instrucciones de un codificador [21].

El trabajo realizado en este tema de tesis se enfoca en la implementación del codificador MPEG-4 ASP en la arquitectura del procesador ARM bajo el SO Linux. Se explica a continuación en cuál de los tres niveles de abstracción del sistema toma lugar el desarrollo, como muestra la Fig. IV.1.

![Diagrama de relación entre los diferentes niveles de abstracción: espacio de usuario, espacio del kernel (Linux) y el procesador (ARM).](image)

Fig. IV.1: Diagrama de relación entre los diferentes niveles de abstracción: espacio de usuario, espacio del kernel (Linux) y el procesador (ARM).

En el espacio de usuario (“User Space” en la Fig. IV.1) es donde se desarrollan las aplicaciones, aquí el desarrollador tiene acceso al sistema de archivos (“I/O” en la Fig. IV.1). Además, en el espacio del usuario reside el software proporcionado por TI que administra los recursos de los codificadores conforme al estándar de la arquitectura DaVinci que es utilizado en la aplicación.
El espacio del kernel (“kernel space” en la Fig. IV.1), provee la manera de abstraer el control específico del hardware de la arquitectura del sistema mediante un sistema operativo (se maneja Linux en el trabajo de esta tesis). Por lo tanto, para hacer uso de los periféricos o algún componente de hardware en la aplicación se debe utilizar la interface proporcionada al espacio del kernel mediante APIs de Linux (“Linux APIs” en Fig.IV.1) en combinación con controladores para dicho componente de hardware como se observa en la Figura IV.1 (dentro de “Kernel Space”). Adicionalmente, entre los beneficios de contar con un sistema operativo como Linux, se proporciona una estructura de software que permite administrar los procesos que se ejecutan en el procesador en forma de multitarea.

Finalmente el espacio de Hardware del sistema es el nivel más bajo de los niveles de abstracción como lo muestra la figura IV.1. Consiste en el procesador (“ARM system” en Fig.IV.1), coprocesador o algún dispositivo periférico, conectado al sistema en hardware.

**Codec Engine**

El Codec Engine [22], es el software de la arquitectura DaVinci (“Codec Engine” en Fig. IV.1) que se encarga de administrar los recursos y ejecución en etapas de los codificadores de media que son apegados al estándar xDM. Los recursos administrados en Codec Engine son memoria del sistema y canales DMA (“DMAN3 en Fig. IV.1). El Codec Engine obtiene los requerimientos de cada codificador mediante funciones de consulta, y la infraestructura de Codec Engine les cede o reclama dichos recursos. Esta metodología evita los conflictos que usualmente suceden con el uso de múltiples codificadores en un sistema embebido, dado que por lo general un codificador no está contemplado a compartir recursos con otro codificador y estos en dado momento pueden llegar a tomar los mismos recursos del sistema. Finalmente, Codec Engine establece un protocolo de comunicación entre el procesador y co-procesador cuando el codificador utilizado en la aplicación es destinado a ser ejecutado en el co-procesador.

**API VISA**

Este API (“VISA API” en Fig. IV.1) permite manejar un formato común para llamar a los codificadores de media contenidos en VISA (Video, Imaging, Speech and Audio) a través del Codec Engine. Esto ahorra el trabajo al desarrollador que utiliza la arquitectura DaVinci, al no tener que aprender un API para cada codificador que utiliza en la aplicación, ya que el formato general del API se conserva para todos los codificadores.
xDM (xDAIS for Digital Medial) [23] es un estándar de programación ("xDM API" en Fig. IV.1) que se construye encima de un codificador ya existente (por ejemplo "Video Codec" en Fig. IV.1). Para fines de ejemplo en la figura IV.1 esto se representa como una envoltura a un codificador de uno de los tipos de media contenidos en VISA. La envoltura alrededor del codificador representa las interconexiones de código establecidas en el estándar que forman pareja con el Codec Engine para negociar las etapas de ejecución y los recursos requeridas por el codificador.

La ejecución controlada del codificador se logra al contener en diferentes funciones establecidos las diferentes etapas que son requeridas para lograr la operación entera de dicho codificador, desde la consulta de cuantos recursos necesita, la creación de la instancia del codificador, el procesamiento de los datos de media, la consulta de los recursos a ser regresados al sistema y terminando con la eliminación de todos los recursos utilizados.

Es notable mencionar que el estándar xDM solo especifica las funciones para las etapas de procesamiento y opcionalmente para control (control dinámico de parámetros y/o consulta de estado) de los codificadores de VISA. Para el resto de los API’s se apoya en un estándar anterior a xDM conocido como xDAIS el cual define los APIs comunes para algoritmos que se utilizan en los DSP de TI.

Por este motivo en la figura IV.2, se muestra que la relación de las interfaces especificadas en los estándares más recientes como xDM, se apoyan en interfaces que han existido anteriormente. De esta manera, la operación completa del codificador es conformado mediante APIs especificados en xDM y otros elementos contenidos en estándares anteriores.
Fig. IV.2: Diagrama de relación entre las interfaces estandarizadas que llevan al codificador conforme a la arquitectura DaVinci.

xDAIS (eXpress Digital Algorithm Interface Standard) [24] es un estándar precursor a xDM, la cual define los lineamientos de programación, que deben seguir los algoritmos (codificadores, filtros, etc.) de procesamiento en tiempo real destinados a ser utilizados en los DSP y procesadores de TI. Dado que la memoria es un recurso invaluable pero limitado en los sistemas embebidos, el estándar propone una estructuración del código de los algoritmos para el uso eficiente de la memoria del sistema. Además, el estándar establece que los algoritmos deben de ser re-entrantes, y deben permitir que el software que los maneje, almacene y entregue de vuelta su estado previo. Esto permite que toda la instancia del algoritmo en memoria se pueda eliminar dada la necesidad que se necesite disponer de ese espacio en memoria en los intervalos de tiempo cuando el algoritmo en des-uso. Por lo general, esto se permite dado que en los algoritmos de tiempo real, la frecuencia de las iteraciones de procesamiento se dicta en base a la frecuencia de muestreo apropiado para el tipo de media manejado. Por consiguiente, la memoria utilizada para el algoritmo después del procesamiento queda en des-uso hasta que la siguiente muestra sea obtenida, lo cual puede abarcar un gran número de ciclos del procesador. Debido a esto, no es necesario comprometer los recursos de memoria requeridos por el algoritmo en espera de la actualización de las muestras.
Sin embargo, no toda la memoria debe ser eliminada después del procesamiento de resultados, por esta razón el estándar xDAIS distingue dos tipos de memoria que son utilizados en los algoritmos, los cuales no son contemplados en la memoria para los búferes de entrada y salida. XDAIS denomina el primer tipo de memoria como “memoria de trabajo”, el cual consiste en memoria que se necesita momentos antes de realizar las operaciones del procesamiento y deja de ser útil al obtener el resultado. Esta memoria puede abarcar un arreglo, un espacio de memoria o variables que se podrán reciclar después de la obtención del resultado. Posteriormente, la memoria reciclada podrá ser re-utilizada por el sistema operativo, en un proceso u otro algoritmo. Mientras el segundo tipo de memoria identificada en xDAIS es la “memoria persistente”, es la memoria (arreglo, espacio de memoria o variables) que se vuelven a necesitar en la siguiente etapa de procesamiento del algoritmo y debe ser almacenada en memoria no volátil, momentos después del procesamiento del resultado por el algoritmo.

IALG

Mientras xDAIS especifica los lineamientos de programación y modularidad para los algoritmos conformes al estándar, se apoya en una interface establecida (ver “IALG” en figura IV.2) denominada como IALG [25]. Esta interface especifica los API’s que debe implementar el algoritmo según la arquitectura DaVinci para mediar los recursos de memoria requeridas con la capa de software que los administra. Entre lo que destaca en la interface IALG, es el uso de la tabla de funciones conocido como “V Table” el cual especifica las diferentes etapas del algoritmo que requieren el uso de memoria de trabajo y memoria persistente. La tabla mencionada, se utiliza en conjunto con la tabla de memoria “mem table” donde se especifica la cantidad de memoria que es necesaria para utilizar cada una de las dichas funciones (incluyendo el espacio de memoria de las instancias). De tal forma que cada función contenida en IALG, se podrá acceder bajo un mismo objeto con apuntadores a dichas funciones y se podrá consultar la cantidad exacta de memoria que se requiere para el uso de dichas funciones. En la figura IV.3 se indican las funciones (la mayoría especificadas en IALG) que debe implementar un codificador xDM para ser conforme a la norma. El comportamiento de las funciones de la Figura IV.3 se detalla en la siguiente sección mediante la explicación de una aplicación que utiliza un codificador de video (MPEG-4 SP) conforme al estándar xDM.
IV.2 Codificador MPEG-4 SP de Texas Instruments

Con el objetivo de analizar y comprender con más detalle, como una aplicación utiliza un codificador conforme a la arquitectura DaVinci, para el caso específico del codificador MPEG-4 SP desarrollado por TI para el DM355 contenido en el DVSDK. Se analizaron los procedimientos necesarios para llamar al codificador en una aplicación contenido dentro del DVSDK que tiene como objetivo codificar un archivo de video sin codificación, para generar un archivo equivalente codificado según el perfil MPEG-4 SP [26]. Dicho codificador realiza la codificación a partir de unidades de cuadros de video y está contemplado en el diseño para ser utilizado en el coprocesador del DM355.

Se apoyara en el análisis con las figuras Fig.IV.4 y Fig. IV.5 que son los diagramas de flujo generalizados para la aplicación y el lazo de codificación.
La representación del “inicio” de la aplicación se muestra en la figura IV.4, el cual continúa con el bloque “A”. Dado que el codificador MPEG-4 SP de TI necesita un espacio definido en memoria para almacenar momentáneamente los datos del cuadro a codificar (entrada) y los datos del cuadro codificado (salida). Se necesita asignar memoria la cual debe
ser disponible por las diferentes funciones en la aplicación. Por esta razón, se utiliza la función de asignación de espacio de memoria contigua “CMEM_alloc()”, para asignar dichos búferes de memoria, con un tamaño en bytes elegido en base al caso dictado por los límites máximos de tamaño de cuadros que puede manejar el codificador (ancho x alto). Al finalizar la asignación de memoria con “CMEM_alloc()”, la función regresa un apuntador al inicio de los búferes asignados, la cual es tomada en cuenta al momento de procesar el cuadro.

Como paso siguiente en el bloque B, se obtienen todos los parámetros del codificador a partir de un archivo de configuración, estos parámetros son introducidos en una sola estructura conocida como “IMP4VENC1_Params”, el cual es una versión extendida de la estructura genérica que utilizan todos los codificadores de video dentro del API de VISA. Esto con el fin de leer todos los parámetros del codificador, bajo una sola estructura ubicado por un solo apuntador al inicio de la estructura.

Siguiendo con el código de la aplicación relacionado con el bloque C de la figura IV.4. Se invoca el la aplicación al codificador mediante la función “ALG_create()”. Esta función, la cual ha sido declarada en un archivo de cabecera, sirve para simplificar el proceso de llamar a las dos funciones de IALG. Estas funciones son “algAlloc()” (ver Fig. IV.3) donde se obtienen los requerimientos de memoria para las funciones del codificador y la creación de la instancia del algoritmo en memoria con “algInit()” (ver Fig. IV.3).

Opcionalmente en el bloque “D” de la figura IV.4, debido a que el codificador MPEG-4 SP de TI fue diseñado para la utilización de canales DMA, la aplicación debe ceder canales DMA disponibles al codificador MPEG-4 SP de TI para el traslado directo de memoria externa al co-procesador (en donde se llevara a cabo el procesamiento de los cuadros).

En el bloque “E” de la figura IV.4, se representa el lazo de codificación, donde se ciclan la codificación de cuadros de video obtenidos desde un archivo de video sin codificación en el sistema de archivos. A su vez cada cuadro codificado se aloja en el archivo que reflejara la codificación del archivo original según el perfil MPEG-4 SP. En esta aplicación, la condición de escape ocurre cuando la aplicación no tiene más cuadros por codificar del archivo sin codificación o cuando un error es generado desde las funciones de codificación. Con el fin de explicar con más detalle los procesos que ocurren en el lazo de codificación, a continuación se apoya en la explicación con el siguiente diagrama de flujo (figura IV.5).
Las etapas dentro del lazo de codificación relevantes a la codificación y la llamada de las funciones de la interface IALG se representan en la Figura IV.5. Iniciando con el bloque “1”, se calcula (a partir del tamaño en bytes y contador de cuadros) el apuntador de archivo que ubica correctamente el inicio del cuadro próximo a codificar proveniente del archivo de video sin codificación. En base al apuntador obtenido y los parámetros del codificador, se copian los datos del cuadro en el búfer de la cual fue previamente asignada (bloque “A” Fig.IV.4) y ubicado mediante el apuntador relacionado a este espacio de memoria.

En el bloque “2” de la figura IV.5 se asigna la memoria de trabajo que se necesita para realizar la operación de codificación mediante la función algActivate(), la correlación con el flujo de funciones se puede apreciar en la figura IV.3.
Siguiendo con el bloque “3” del diagrama (Fig. IV.5), donde toma lugar la invocación de la función de codificación del cuadro “process()” (al igual que en la Fig. IV.3). Los parámetros de la función de procesamiento son dictadas por el estándar xDM, al igual con los apuntadores a espacios en memoria, entre ellos: a la instancia del codificador, a la estructura de parámetros, a los búferes de entrada/salida del cuadro de video.

Dentro el lazo de codificación, después del procesamiento, en el bloque “4” de la figura IV.5. Se almacenan las variables persistentes (por ejemplo: medidas generadas durante el procesamiento) y se libera la memoria de trabajo mediante la función “algDeactivate()”, ver Figura IV.3.

Finalmente las últimas instrucciones dentro del lazo de codificación se representan en la Figura IV.4 como el bloque “5”. En este bloque, se copia el búfer de salida del codificador, con el fin de añadir el cuadro recién codificado al archivo de salida.

De aquí en adelante, el lazo continuara ciclándose mientras exista un nuevo cuadro para codificar que proviene del archivo de video sin compresión o cuando ocurra error en el procesamiento de codificación.

Regresando de nuevo a la figura IV.4, en el bloque denominado como “F” cuando el último cuadro se ha codificado para el archivo sin codificación y el lazo de codificación ha terminado. Finalmente, la memoria utilizada para la instancia del codificador puede ser liberada. Esto se hace de manera indirecta mediante una función “ALG_delete” declarado como una función en uno de los archivos de cabecera. En esta función, el codificador indica a la aplicación, cuantos recursos de memoria están siendo utilizados por el codificador mediante la función algFree (ver Figura IV.3). Posteriormente, dichos espacios en memoria son liberados por la aplicación mediante instrucciones contenidas en “ALG_delete()”.

Finalmente, en el último bloque denominado “G” del diagrama de flujo (figura IV.4), se liberan los búferes de entrada (asignado para cuadro sin codificación) y de salida (asignado para cuadro codificado). Mediante la función “CMEM_free()” el cual libera los espacios en memoria previamente asignados por “CMEM_alloc()”.

Procesamiento de video utilizando la plataforma DaVinci
IV.3 Codificador MPEG-4 ASP de Xvid

El desarrollo de un codificador conforme al estándar MPEG-4 es un esfuerzo conjunto de desarrolladores y expertos en el área, la cual no solo implica el desarrollo de dicho codificador sino todos el proceso de verificar conformidad al estándar.

Dentro de lo que abarca este trabajo de tesis, el cual consiste en trabajar con un codificador MPEG-4 ASP en la arquitectura del procesador ARM con SO Linux en la plataforma DaVinci. Existe la posibilidad de extrapolar un codificador existente en una plataforma de uso común y portarla a la plataforma embebida.

El codificador de Xvid es una implementación que se apeg a al estándar MPEG-4 v2 de un codificador bajo el perfil ASP, la cual es publicado bajo una licencia de código abierto conocida como GNU GPL (General Public License) [27]. A causa del desarrollo continuo bajo la colaboración de la comunidad del proyecto Xvid y que comparte en código muchos elementos principales con el codificador DivX (Digital Video eXtreme), argumentalmente es uno de los mejores codificadores de código abierto apegados al estándar MPEG-4 ASP disponibles. Lo anterior se debe a que origino bajo el mismo proyecto MAYO hasta que se ramificó cuando el proyecto persiguió establecer un codificador para fines comerciales, mientras el nuevo proyecto Xvid (DivX deletreado al revés) continúo desarrollando un codificador con código abierto.

El codificador Xvid se ha desarrollado en código ANSI C bajo una estructura modular nombrada “xvidcore” [28]. La modularidad del código permite la re-utilización de componentes de software que son propios de ciertos sistemas operativos y arquitecturas de procesadores. Además, dentro del software contenido con Xvid, se encuentran funciones optimizadas en código ensamblador para los algoritmos más exigentes en la codificación como (DCT, IDCT y Estimación de Movimiento). Todo esto con la finalidad de distribuir un solo código eficiente de tipo multiplataforma.

IV.3.1 Herramientas de codificación en Xvid

El codificador de Xvid contiene muchas herramientas abiertas a configuración por el usuario, las cuales permiten elegir un intercambio entre la calidad y la codificación de las secuencias de videos [29]. Es notable resaltar, que muchas de estas herramientas en Xvid no
están contempladas dentro del estándar de un codificador MPEG-4 ASP, por lo que mencionamos que el Xvid se apega al estándar MPEG-4 ASP pero no es conforme a éste.

En la configuración de estas herramientas, es de suma importancia la consideración el tipo del comportamiento deseado para el codificador resultante. Como ejemplo, si el objetivo de la aplicación requiere de un mayor grado de codificación sin importar la carga computacional ni la latencia, usualmente el uso de la mayoría de las herramientas será la configuración a seguir. Sin embargo, el comportamiento deseado para la aplicación de este trabajo requiere del menor uso de carga computacional, con la menor latencia y el cumplimiento con las herramientas contenidas dentro del codificador MPEG-4 ASP establecidas por el estándar. Por esta razón es necesario habilitar solo las herramientas contempladas para un codificador MPEG4-ASP y omitir las herramientas adicionales que son exclusivas del codificador de Xvid dentro de su código.

A continuación se mencionan las herramientas de codificación que cumplen con el perfil MPEG-4 ASP que son configurables dentro del codificador Xvid.

- Codificación de cuadros “I” Intra (ajuste de cuantificador, intervalo de repetición),
- Codificación de cuadros “P” Inter (ajuste de cuantificador),
- Acceso a GOP (permitir acceso a grupo de imágenes posterior a la codificación),
- Codificación de cuadros “B” Bi-direccionales (ajuste de cuantificador, duración de repetición),
- Estimación de movimiento con precisión a ¼ de pixel (QPEL),
- Estimación de movimiento global (GMC),
- Cuantificación alternativa (H.263 o MPEG-4),
- Codificación Entrelazada.

IV.3.2 Plataforma de prueba para xvidcore

Con el objetivo de analizar el desempeño del codificador Xvid, es recomendable trabajan en una plataforma de prueba. Por esta razón, se analizó el código dentro de un ordenador que operaba en un sistema operativo de Linux.

Antes de utilizar el codificador Xvid por primera vez, es necesario configurar el código multi-plataforma de “xvidcore” para la plataforma objetiva. Lo cual se logra ejecutando un
archivo de comandos que consulta la arquitectura y ciertos componentes de software, los cuales son propios de cada plataforma. En base a la información recompilada elige los componentes de “xvidcore” adecuados para la plataforma. Por consiguiente el resultado de esta operación es una nueva carpeta con código personalizado para la plataforma objetiva.

Dentro del código personalizado se proporciona una aplicación denominada “xvid_encraw” que utiliza “xvidcore” para codificar un archivo de video sin codificación. El objeto resultante de esta aplicación es un nuevo archivo codificado con formato MPEG-4 ASP. Al analizar el código y experimentar con la configuración de los parámetros de esta aplicación, se provee la manera de comprobar el desempeño del codificador, bajo un ambiente con capacidades de depuración como ECLIPSE (ver Fig. IV.6).

![](image)

Fig. IV.6: Utilización de ECLIPSE para depurar la aplicación que emplea el codificador MPEG-4 ASP de Xvid.

**IV.3.3 Aplicación “xvid_encraw” controladora del codificador Xvid**

El conocimiento sobre el manejo del codificador Xvid y su comportamiento dentro de una aplicación a desarrollar, es esencial para la integración en la arquitectura del procesador ARM
de la plataforma DaVinci que se utilizara para este trabajo. La aplicación "xvid_encraw" el cual es contenido en el software de Xvid es una aplicación multiplataforma escrita en ANSI C que demuestra el uso del núcleo de funciones de codificación de Xvid. En base al análisis sobre la manipulación de la aplicación “xvid_encraw”, al codificador Xvid. Se obtuvo el conocimiento necesario para poder personalizar el software a los fines enmarcados de este trabajo. La cual es la utilización del codificador MPEG-4 ASP del Xvid incorporándolo a trabajar en una arquitectura de un procesador ARM bajo un SO Linux embebido sobre la plataforma Davinci.

A continuación se generaliza el control sobre el codificador Xvid en la aplicación “xvid_encraw”. La explicación se resume en ocho etapas resumidos a continuación, además la numeración de las etapas se incluyen en el diagrama de flujo (ver figura de apoyo Fig. IV.7).

1) Declaración de los parámetros del codificador Xvid como variables globales con valores por defecto.
2) Dentro de la función "main()", se interpretan los parámetros y los nombres de archivos adicionales que fueron declarados al inicio del la aplicación en la línea de comandos. De esta manera la aplicación permite al usuario actualizar los parámetros que serán utilizados con el codificador Xvid y de manera similar al archivo que es codificado.
3) En base al ancho y altura del video obtenidas como parámetros, se calcula y asigna la cantidad de memoria necesaria para los búferes de entrada y de salida que entraran en juego para la codificación de los cuadros.
4) En base a los parámetros obtenidos, se inicializa la función del codificador como y se le asigna un apuntador para dicha función.
5) Dentro del lazo de codificación de cuadros, se calcula la posición del cuadro dentro del archivo a codificar, posteriormente se copia este cuadro al búfer de entrada la cual fue asignada en 3).
6) Se ejecuta la función principal del codificador de Xvid, la cual depende de apuntadores a las instrucciones que debe ejecutar, a los parámetros de la codificación, a la ubicación en memoria del cuadro a codificar y a la ubicación en memoria donde almacenara el cuadro que codificara.
7) Al terminar la ejecución de la función de codificación, el cuadro calculado almacenado en el búfer de salida se anexarán al archivo codificado.
La condición de salida del lazo de codificación de cuadros será cuando el resultado de la función principal de codificación genere un código de error (por ejemplo, dada la condición en la cual no hay cuadro restante a codificar).

Fig IV.7: Diagrama de flujo de aplicación “xvid_encraw”.

**IV.4 Conclusiones**

La arquitectura Davinci tiene como objetivo evitar problemas que surgen al desarrollar aplicaciones en un sistema embebido sumamente complejo y ofrecer en el mercado soluciones de software para acelerar el tiempo de desarrollo de un sistema basado en una arquitectura Davinci. Al analizar las especificaciones y estándares extensivos que conforman la arquitectura Davinci, se puede apreciar el detalle de la arquitectura respaldada de la experiencia de los desarrolladores de embebidos de TI contenido dentro de la documentación.
Conforme a los codificadores de la arquitectura, es válido remarcar que un codificador conforme a la arquitectura DaVinci, en ningún momento toma recursos del sistema por sí mismo, el codificador siempre debe especificar los recursos que necesita y la capa de abstracción encima de dicho codificador, le debe ceder los recursos antes de continuar. Esto es de suma importancia con el uso de codificadores de video, dado que son los algoritmos que más consumen recursos del sistema en procesamiento y memoria para los tipos de codificadores de media especificados en VISA y generalmente son utilizados en conjunto con múltiples codificadores de diferentes tipos de media, como ejemplo: en una aplicación la cual codifica video en conjunto con un codificador de audio.
CAPÍTULO V PROOTIPO DEL MPEG-4 ASP EN DVEVM DM355

En este capítulo se presenta el método para portar el codificador de video conforme al estándar MPEG-4 de perfil avanzado simple (Advanced Simple Profile, ASP) en la arquitectura del procesador ARM bajo SO Linux embebido de la plataforma DaVinci DM355 de Texas Instruments [17]. Además, se explicaran algunos de los lineamientos establecidos por TI, que el codificador debe seguir para ser utilizado en la Arquitectura DaVinci. También, están contempladas en este capítulo las implementaciones para la codificación de un archivo a partir de uno sin codificación utilizando el codificador MPEG-4 ASP.

V.1 Método de desarrollo

Como se ha comprobado el uso del algoritmo del codificador Xvid bajo una computadora personal y se encontró que este codificador es conforme a las necesidades de un codificador MPEG-4 ASP para uso en la implementación. Se continuó con el trabajo de portar dicho algoritmo al procesador ARM bajo SO Linux embebido del DM355. El método de portar el codificador Xvid se divide en dos partes: En la primera parte se configura el código multiplataforma de Xvid conforme a la arquitectura del procesador ARM9 contenido en el DM355 de Texas Instruments; en la segunda parte se modifica el codificador Xvid para que se apegue a las especificaciones establecidas por el estándar eXpressDSP Digital Media (xDM) de Texas Instruments [23]. Esto con el objetivo de lograr el funcionamiento combinado con los diferentes componentes de software de la plataforma.

V.1.1 Configuración codificador Xvid para arquitectura ARM9

El programa del codificador Xvid tiene el código necesario para configurar el tipo de plataforma en la cual será implementado el codificador Xvid (ver sección IV.3.2). El programa del codificador Xvid genera un reporte de las características de la plataforma en hardware y en software, en base a éste se decide qué componentes son necesarios para la configuración de la implementación del codificador Xvid en la plataforma deseada.

El programa que analiza la configuración de la plataforma de implementación, no tiene la capacidad de analizar una plataforma externa, es decir, este programa solo trabaja en forma...
local. Como el ambiente de desarrollo en el que se trabaja es un ambiente de desarrollo embebido, se optó por analizar la lista de arquitecturas que contempla el programa de configuración y elegir la plataforma que más se apegue a la plataforma del DM355.

Finalmente, los tres parámetros proporcionados en línea de comandos al programa de configuración del codificador Xvid para su implementación en la plataforma DM355 fueron: Plataforma “ARM genérico”, ya que el DM355 tiene el ARM9; tipo de compilador “ARM_V5T_LE-GCC”, de la compañía Montavista Linux; plataforma para la maquina anfitriona “x86_64-pc-linux”.

El programa de configuración del codificador Xvid genera una carpeta para la plataforma DM355 (plataforma objetivo) en la maquina anfitriona que es utilizada en la compilación cruzada.

V.1.2 Portando el codificador Xvid al estándar xDM de TI

La metodología a seguir es inspirada en base a las recomendaciones basadas en el libro de R. Pawate [21], el objetivo de este trabajo es migrar el codificador de una plataforma de escritorio a una plataforma embebida donde espacio (en memoria) y tiempo (MHz) son limitadas [21]. R. Pawate menciona en su trabajo, que el código debe ser modificado para cumplir con ciertos reglamentos, para que pueda ejecutarse en tiempo real, y ser fácilmente integrado con el resto de la plataforma en software. Un código el cual ha sido modificado y finalmente se puede portar a la plataforma lo denomina como código “Golden C”. En dicho trabajo se mencionan 3 pasos generales que se deben seguir tanto para

**Paso 1:** Crear un “ambiente de prueba”.

Después de confirmar que el algoritmo cumple con las necesidades de la aplicación deseada, el primer paso es crear un “ambiente de prueba” con archivos de entrada bien definidos, y los archivos de salida que correspondientes que serán usados después para verificar si la en la portabilidad el algoritmo se apego al los resultados deseados.

**Paso 2:** Convertir el código C al modelo “Golden C”.

En orden para convertir el código en forma modular y satisfacer los requerimientos de procesamiento de embebidos, es mejor que cumpla con ciertos requisitos o reglas las cuales se mencionan a continuación. Hay numerosas reglas a seguir, estas son algunas de las reglas claves:
Regla 1: Organiza el código base. Organiza el código basado en funcionalidad y colócalo en carpetas correspondientes.

Regla 2: El algoritmo no debe desempeñar cualquier tipo de operaciones con archivos de entrada, ni de salida. Todos los datos de entrada y de salida deben ser manejados a través de búferes utilizando apuntadores para aumentar la eficiencia del código. La aplicación o la capa de código más cercana a la aplicación deberá de manejar las operaciones con archivos de entrada/salida, sin embargo el algoritmo a desarrollar nunca deberá manejarlas.

Regla 3: Remover todas las funciones mallocs y callocs.

Mallocs y Callocs son funciones las cuales se utilizan para asignar un espacio de memoria del tamaño establecidos por sus parámetros. Estas funciones por lo general regresan un apuntador que indica el inicio del bloque. En codificadores se utilizan extensivamente para asignar un espacio en memoria para reservar los segmentos a codificar. Esto posiblemente es una de las partes más difíciles del proceso de la conversión, debido que desarrolladores de algoritmos tienden a utilizar mallocs. Sin embargo, para procesamiento con embebidos, se necesita que una parte de la estructura de software se encargue de esta operación. La estructura que maneja estos recursos en el DVSDK es el Codec Engine.

4: Clasifica los tipos de memorias utilizadas entre persistentes, de trabajo y constantes. Memoria persistente es la memoria que necesita ser mantenida de un cuadro a otro, mientras memoria de trabajo es desechable y no necesita ser mantenida de un cuadro a otro. Constantes son tablas o coeficientes las cuales se necesitan en el algoritmo. En aplicaciones cuando se pretende producir en grande escala, constantes pueden ser movidas a ROM, para reducir el costo del sistema total.

Regla 5: Evita el uso de variables globales y estáticas.

Regla 6: Tipos de Datos deben ser aislados a un solo archivo de cabecera (archivo con extensión .h) con explicaciones claras. Esto se volverá útil más tarde cuando se desee comparar código optimizado o kernel proveídos por TI y/o fuentes de código proveniente de terceros. Tipos de datos que emparejan correctamente la longitud de palabra (en bits) del dispositivo y la librería de rutinas son importantes.

Regla 7: El código no debe contener instrucciones específicas a un endian.

Endian es la forma posicional en la cual se representa los sub-componentes que conforman una sola palabra de memoria para una arquitectura dada. Dado que el objetivo es lograr un
algoritmo capaz de funcionar en múltiples plataformas (cada uno con su respectivo endian), es ideal replantear operaciones dentro del algoritmo que dependan del cierto ordenamiento endian.

**Regla 9:** El código debe ser conforme al estándar xDAIS.

Existen herramientas en el mercado que sirven para probar el apego al estándar. De igual manera se puede obtener el estándar del sitio de TI.

### V.2 Codificación de una secuencia de video MPEG-4 ASP

En base a la metodología propuesta en la sección anterior, se ha configurado el codificador Xvid conforme al estándar MPEG-4 ASP para su ejecución con el procesador ARM9 del DM355, y partiendo de esta configuración inicial se ha construido encima del API de VISA para el control del codificador de Xvid de tal manera que sea compatible con los componentes de software de TI para uso en la plataforma DaVinci. Mientras se respeta la estructura necesaria para la utilización del codificador.

Para poner el codificador a prueba en el SO Linux embebido bajo el procesador AMR9 de la plataforma DaVinci, se desarrolló una implementación la cual codifica un archivo de video sin-codificación. La codificación se realiza al utilizar el API de Xvid a través de la interface para los codificadores de video xDM. El resultado de esta operación es un archivo codificado apegado al estándar MPEG-4 ASP.

La aplicación que demuestra en control del API de Xvid “encraw”, permite el uso de estas herramientas al introducir instrucciones adicionales en la línea de comandos en la ejecución. En el caso de que se tenga la necesidad de un esquema de codificación que difiere completamente del defecto, se requiere de la introducción de muchos parámetros adicionales en la línea de comandos y se corre el riesgo de introducir errores de sintaxis.

Una manera de mejorar esto introducir todos los parámetros a través de un archivo de configuración de parámetros, donde está especificado los valores pre-determinados y el rango de los parámetros. Además, con el uso del archivo de configuración de parámetros, se provee la manera de resaltar los parámetros relevantes a las herramientas de un codificador MPEG-4 ASP y limitar las herramientas exclusivas al codificador Xvid. Esto se ejemplifica con la figura V.1., donde se muestra un archivo de configuración denominado “XVIDparams.cfg” en un editor de texto.
Fig. V.1 Archivo configurador de parámetros.

Dado que la aplicación ahora depende de tres archivos: archivo de video sin codificación, archivo codificado (el cual se genera al finalizar la aplicación) y el nuevo archivo de parámetros. También, se puede proveer un archivo para la configuración de archivos de E/S que serán utilizados en la ejecución de la aplicación. Esta configuración se puede presentar en forma de un archivo sencillo de tres renglones que enumera el nombre de cada archivo por posición de renglones (como se muestra en Fig V.2).

Fig. V.2 Declaración de archivos de entrada/salida en un archivo de configuración.

Procesamiento de video utilizando la plataforma DaVinci 62
Por otra parte, se explica a continuación la aplicación de manera resumida, apoyándose con los diagramas de flujo Fig V.3 y Fig V.4, los cuales representan el proceso principal de la aplicación “main()” y el ciclo a nivel de cuadros que se realiza de manera repetida para codificar todos los cuadros de la secuencia de video sin-codificación.

El función main() de en esta aplicación consta en seis partes que se ejecutan de manera secuencial para preparar los parámetros y recursos necesarios para el lazo de codificación. Esta función culmina con el término del lazo y la liberación de recursos utilizados.

Fig V.3. Diagrama de flujo de función principal main() del codificador Xvid en DM355.
A. Primeramente se leen los archivos y los parámetros que serán utilizados en la ejecución de la aplicación a través de sus respectivos archivos de configuración. Esto permitirá abrir dichos archivos y asignarles un apuntador en el sistema de archivos de Linux, que servirá para extraerles su información. Cuando todos los archivos a utilizar en la aplicación han sido leídos e identificados se puede continuar con la siguiente etapa de interpretación.

B. Dentro de la interpretación de sintaxis, el archivo de configuración se analiza y se utiliza para llenar una estructura de parámetros denominado VIDENC1_XVID_PARAMS, la cual es una versión extendida de la estructura IVIDENC1_Params (esta sirve como interface genérica para las opciones de codificación de video según el estándar xDM version1). Además, en esta etapa se revisan que ciertos parámetros estén dentro de los valores establecidos.

C. Es necesario que se asigne un espacio en memoria para los búferes de E/S del codificador. En base a las opciones de altura y ancho del cuadro de video obtenido como parámetros en la estructura VIDENC1_XVID_PARAMS, se puede calcular el espacio necesario para el búfer de cuadro sin codificación (entrada) y el búfer del cuadro codificado (salida). La operaciones de asignación a memoria se realizan por medio de la función Memory_contigAlloc(), la cual regresa una apuntador al espacio asignado.

D. La inicialización del codificador XVID requiere de la introducción de parámetros obtenidos de la estructura VIDENC_XVID_PARAMS. La instancia del codificador se crea utilizando los parámetros estáticos llamados en la función VIDENC1_XVID_Create(). Que este a su vez llama al API de Xvid con las funciones xvid_global(XVID_GBL_INIT) y xvid_encore(XVID_ENC_CREATE).

E. Con esta preparación de parámetros y asignación de memoria es suficiente para empezar con el lazo de codificación de cuadros, el cual realiza en iteraciones por cada cuadro de la secuencia de video proveniente del archivo de video sin-codificación, hasta terminar con el último cuadro del archivo. Esto etapa se divide en cuatro etapas (como se muestra en la figura V.4):

1. Se llena el búfer de entrada al codificador con un cuadro de la secuencia de video sin-codificación.

2. Utilizando la función VIDENC1_XVID_process(), que a su vez llama al API de Xvid con la función xvid_encore(XVID_ENC_ENCODE) que realiza la codificación MPEG-4
ASP de Xvid para al cuadro que se encuentra en el búfer asignada como entrada. Cuando la función de proceso termine su ejecución, el búfer asignada como salida del codificador se actualiza con el cuadro codificado.

3. Finalmente se escribe el cuadro codificado a un archivo que añade a la secuencia codificada con cada iteración.

4. Opcionalmente se pueden, mostrar en la línea de comandos las estadísticas generadas al codificar cada cuadro.

![Diagrama de flujo lazo de codificación](image)

**Fig. V.4. Diagrama de flujo lazo de codificación.**

F. Liberación de recursos
V.3 Implantación en el DVEVM DM355

V.3.1 V4L2 para captura de video en EVM

En esta sección se explica de manera general el método de captura de video para las aplicaciones desarrolladas en el EVM. Dentro del software de desarrollo para el EVM se proveen las herramientas en forma de componentes de software para manejar adecuadamente el hardware que entra en juego en la captura de video. Estos componentes se describen a continuación.

V4L2 (Video For Linux version Two) es un estándar que define el API del mismo nombre la cual es integrado al núcleo de Linux en los sistemas que manejan video [30]. El API estandarizado permite una infraestructura y metodología de acceso común a los controladores de dispositivos que capturan video. De esta manera las variaciones del software entre sistemas que manejan el API se limitan en los controladores específicos de cada sistema.

Por consiguiente, en el EVM el interfaz de hardware para la captura de video con la aplicación a desarrollar, es mediante el uso del API V4L2 y el controlador para el integrado TVP5146. En el EVM, el TVP5146 actúa como fuente indirecta de video digital al convertir las entradas de video que provienen en diferentes formatos, tal como súper video y video compuesto. Esta interacción entre el la aplicación y el hardware de captura se esquematiza en la Fig. V.5
Linux tiene pocas arquitecturas de controladores dominantes, entre las cuales se encuentran los controladores de dispositivos de carácter que se utilizan típicamente en el los controladores de sonido y de video (caso del controlador de captura). Se distinguen en la manera que leen y escriben en ristra con orden FIFO (First In First Out).

Es notable mencionar que una funcionalidad característica de Linux es permitir abstraer los dispositivos de hardware sobre el sistema de archivos. De tal modo que los procesos que interactúan con estos dispositivos lo hacen por medio de operaciones de ficheros. Por lo tanto, un nodo para el controlador de captura de video es creado en el sistema de archivos, la cual se ubica en /dev/video0 (ver Fig. V.5) [31]. Gracias a la abstracción mencionada, con esta ruta se puede leer al dispositivo a través de interfaces bien definidas como operaciones a archivos (por ejemplo: open, read, close) y instrucciones específicas al dispositivo (tal como funciones “ioctl” para consultar/ajustar información sobre el hardware).

Fig. V.5 Niveles de Abstracción para interface con cámara de captura.
V.3.2 Captura y codificación

Para recapitular, en la sección anterior hemos demostrado el uso del codificador XviD utilizando las normas contenidas en la arquitectura de software DaVinci para el control del codificador en la implementación sobre el EVM, la cual se realiza en el procesador ARM del DM355. En la implementación previamente explicada, primeramente se lee un archivo de video sin codificación, luego se codifican mediante el codificador MPEG-4 ASP de Xvid, y finalmente se almacena el archivo de video codificado en el sistema de archivos de Linux.

En esta sección se añade a la implementación, la captura de cuadros de video a partir de un dispositivo de captura la cual está contemplada entre los periféricos del EVM. Sustituyendo de esta manera el archivo de video sin compresión por cuadros de video recién obtenidos directamente del dispositivo de captura, la cual sirve como fuente de video para el codificador Xvid.

El código del lazo consiste en la captura de cuadros de video a través del integrado TVP5146 en modo entrelazado utilizando el controlador de captura V4L2.

Inicialización del Dispositivo para la Captura

A pesar de que se proporciona en el DaVinci SDK el API V4L2 y el controlador del dispositivo de captura, el código para el manejo de este dispositivo es extenso, debe seguir una secuencia exacta y estar plegada de condiciones especificadas dentro del API V4L2. Además, los búferes para los cuadros que se toman del dispositivo de captura se ubican en el mapa de memoria específica al EVM. Por esta razón, aprovechamos el conocimiento de expertos a partir de aplicaciones contempladas como demostraciones. Esto ayudará en la implementación al contar con el procedimiento de captura la cual ha sido previamente comprobada con éxito.

El esta sección vemos la introducción de las funciones IOCTL que se utilizan para inicializar captura.

El dispositivo de captura de video se inicializa con la función initCaptureDevice(), la cual depende de funciones ioctl() y variables claves establecidas en el API V4L2 para establecer el proceso de inicialización:
Como muestra la figura Fig. V.6 la inicialización del dispositivo de captura se consigue de la siguiente manera:

1. Se abre el nodo del dispositivo de captura en el sistema de archivos en Linux ubicado en “/dev/video0” y se le asocia un apuntador de fichero, el cual es utilizado para la mayoría de las operaciones de la iniciación del dispositivo. Después, se consultan las capacidades del dispositivo mediante ioctl(VIDIOC_QUERYCAP). Posteriormente se selecciona la entrada, la cual será establecido para el puerto de video compuesto con la función ioctl(VIDIOC_S_INPUT).

2. La siguiente etapa de la inicialización del dispositivo consiste en seleccionar el formato de datos de la captura. Primero con el estándar, el cual se establece como NTSC utilizando el ioctl(VIDIOC_S_STD). Luego se introduce mediante una estructura definida por los
parámetros de formato de pixel como luma/croma 4:2:2 (V4L2_PIX_FMT_UYVY), tipo de campo como entrelazado (V4L2_FIELD_INTERLACED), la altura y ancho del cuadro como lo conforma la resolución D1. Esta estructura a su vez se utiliza para establecer el formato con la función ioctl(VIDIOC_S_FMT). Es de notarse, que el tanto el formato de pixeles como el tipo de campo establecidos para esta implementación corresponden al formato de video digital soportado por el integrado TVP5146.

3. Es necesario proporcionar búferes de memoria para la captura, por lo tanto siguiente etapa consiste en inicializar los búferes para la captura. Como primer paso de esta etapa se inician los búferes para la captura con el ioctl(VIDIOC_REQBUFS). Luego se verifica el estado de este búfer utilizando ioctl(VIDIOC_QUERYBUF) y por último se mapean al espacio de aplicaciones utilizando la función mmap().

4. La última etapa de la inicialización del dispositivo de captura consiste en pasar el búfer al controlador del dispositivo para incorporarlo a la cola con la función ioctl(VIDIOC_QBUF). Finalmente, se indica al controlador del dispositivo de captura que debe prepararse de tal modo que esté listo para la captura con ioctl(VIDIOC_STREAMON).

La figura V.7 muestra la iteración los procesos involucrados en el procesamiento de un cuadro de video una vez que se comience a ejecutar el lazo de codificación y captura. A su vez, se muestra en la figura el rol de los búferes antes y después del proceso de codificación.
1. Primero, es necesario quitar de la cola un nuevo búfer sin-codificación capturada con el controlador del dispositivo utilizando la función ioctl(VIDIOC_DQBUF), la cual regresa un apuntador designado al búfer de captura. Es notable mencionar, que hasta este momento el búfer de entrada para el codificador no contiene información de la captura.

2. Por esta razón, se realiza una copia de memoria entre búферes utilizando la función memcpy(), donde la fuente proviene del búfer de captura y el destino será el búfer de entrada al codificador. De esta manera al terminar esta operación, se tiene lo necesario para comenzar con el siguiente proceso de codificación: un búfer de entrada para el codificador lleno de una copia de la captura sin-codificación y un búfer de salida para el codificador vacío donde se almacenará momentáneamente la captura codificada.

Fig. V.7 Lazo de codificación de cuadros obtenidos desde cámara de captura.
3. Dado que se cumplen con los requisitos para llamar a procesar la codificación del cuadro utilizando la función process(). Se ejecuta esta operación y como resultado actualiza el búfer de salida del codificador con la copia de captura codificada.

4. Para almacenar los cuadros capturados, se añade cada búfer codificada que resulte de la operación process() al archivo que contendrá la secuencia codificada final, mediante el uso de la función de escritura al sistema de archivos fwrite().

5. Finalmente, para finalizar el ciclo de codificación y captura es necesario de que al búfer de captura sea regresado a la cola con el controlador del dispositivo utilizando la función ioctl(VIDIOC_QBUF).
CAPÍTULO VI RESULTADOS

En este capítulo se presentan la evaluación del codificador MPEG-4 ASP portado en el SO Linux embebido bajo la arquitectura AMR9 del DM355. La evaluación de los codificadores es de suma importancia debido a que permite conocer con detalle las capacidades y las limitaciones de dichos codificadores bajo pruebas controladas. Tanto las pruebas como los resultados de dichas pruebas son expuestas a continuación. Además, con propósito de tener una referencia sobre la cual basar el desempeño se realiza en conjunto la evaluación del codificador MPEG-4 SP el cual es disponible en el DVSDK proporcionado para la plataforma DaVinci.

VI.1 Método de Evaluación de Codificadores MPEG-4 en DM355

En esta sección, se presenta el conjunto de componentes tanto de hardware como de software necesario para realizar las pruebas de evaluación del codificador MPEG-4 ASP y el codificador MPEG-4 SP en el DM355. Aparte de los codificadores es necesario contar con secuencias de video para las pruebas, un decodificador compatible para cada codificador, establecer métricas para medir el desempeño y basar la comparación, entre otras herramientas de procesamiento de video. A continuación, estos elementos se describen con más detalle.

VI.1.1 Secuencias de video utilizadas

Como punto de partida para evaluar el desempeño de los codificadores de video, se necesita contar con secuencias de video de referencia. Las que se utilizaron para las pruebas son las que usualmente se utilizan en las literaturas de codificación de video, estas secuencias comunes sirven como base común para evaluar el desempeño entre diferentes codificadores. Estas secuencias de video se denominan como “bus”, “foreman” y “news”, las cuales presentan movimiento de alta, mediana y baja categoría en las escenas del video respectivamente. Una descripción breve de las secuencias se muestra en la tabla VI.1.

Por lo general, antes de utilizar las secuencias en los codificadores de video es necesario, señalar al codificador información sobre las características de las secuencias de video. Además, es necesario averiguar en las especificaciones del codificador o dentro del código
mismo del codificador cuales formatos son compatibles. La información de las secuencias de video que son relevantes como parámetros en los codificadores se menciona a continuación:

- **Tamaño (ancho x alto):** Las secuencias de prueba obtenidas [32] se encuentran en tamaño CIF (Common Interface Format) el cual es indicativo de cuadros con resolución de 352 x 288.

- **Espacio de colores:** Las secuencias de video utilizadas utilizan el formato “YUV 4:2:0P”. Esta designación indica tres parámetros, la primera “YUV” indica la representación de los colores dentro de un cuadro de video como una composición de tres planos: un plano de luminancia y dos planos de crominancia. De manera relacionada, “4:2:0” indica que la crominancia ha sido sub-muestreada con respecto al plano de luminancia durante la captura de la secuencia de video. Finalmente “P”, es indicativo que las líneas horizontales que constituyen un cuadro de video son desplegadas de manera secuencial y de manera continua [33].

**Tabla VI.1 Descripción del contenido de las secuencias de video de prueba.**

<table>
<thead>
<tr>
<th>Secuencia</th>
<th>Descripción</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Secuencia Bus</strong></td>
<td>Esta secuencia de video, presenta el recorrido de un autobús a través de la calle en la ciudad. Mientras, el autobús avanza, la cámara lo persigue centrándolo en la escena. Con esta secuencia se busca comprobar el funcionamiento del codificador bajo movimiento de categoría alta.</td>
</tr>
<tr>
<td><strong>Secuencia Foreman</strong></td>
<td>Esta secuencia de video, presenta el rostro de un hombre que se expresa al hablar con diferentes gestos faciales. Además, la secuencia se grabo con una agitación en la cámara. Con esta secuencia se busca comprobar el funcionamiento del codificador bajo movimiento de categoría mediana.</td>
</tr>
</tbody>
</table>
VI.1.2 Métricas de desempeño

Para hacer la comparación de desempeño entre codificadores es necesario, contar con una métrica establecida sobre la cual hacer la comparación. Debido a esto, la métrica objetiva utilizada en las pruebas de la tesis es PSNR (Peak Signal to Noise Ratio, Relación señal pico a ruido), la cual comúnmente es utilizada en investigaciones y literatura referente a mediciones cuantitativas de la calidad en video decodificado. Mediante el cálculo de PNSR se puede representar la relación entre la energía máxima de la señal original con respecto al ruido que afecta a la señal reconstruida con respecto a la original. Es notable mencionar que el PNSR generalmente se expresa en escala logarítmica, con unidades de decibeles.

Para evitar el desarrollo de otra herramienta en el trabajo de tesis para el cálculo de PSNR entre la secuencial de video original y la secuencia de video reconstruida (ambos en formato YUV 4:2:0P) en las pruebas, se apoyo con herramientas de código abierto contribuidos por usuarios de la comunidad de Matlab [34].

VI.1.3 Herramientas de código abierto para el procesamiento de video

En el trabajo de tesis realizado, se ha enfocado principalmente al desarrollo sobre los codificadores de video. Dado el caso que el decodificador apropiado sea insuficiente para lograr la decodificación de video bajo cierto esquema de codificación, se apoyara con otras herramientas de código abierto para la decodificación y procesamiento necesario mencionados en las páginas de wiki dirigido a los procesadores embebidos de Texas Instruments. Las paginas muestran herramientas con fines de uso complementario a los codificares de TI, la sintaxis para el uso adecuado de las herramientas y las ligas a los portales...
de internet dedicadas a dichas herramientas [35]. Es notable mencionar que las herramientas utilizadas conforme a su uso en las pruebas constituye solo una fracción de las capacidades de cada diferente software. Por esta razón, a continuación se mencionara sola la contribución de cada herramienta utilizada para la experimentación y el software que lo contiene:

- **FFMPEG** [36], es un conjunto de software de código abierto bajo licencia GPL, la cual se utiliza para convertir video y audio entre formatos. Este software utiliza la librería conocida como “libavcodec”, la cual contiene codificadores y decodificadores de una variedad de formatos. Las herramientas de FFMPEG utilizadas en las pruebas son las siguientes:
  
  o **FFMPEG**, esta herramienta de línea de comandos permite convertir archivos de multimedia entre diferentes formatos.
  
  o **FFPLAY**, esta herramienta permite reproducir videos dentro de los formatos permitidos en FFMPEG bajo una interface sencilla.

- **GPAC** [37], es una estructura de software de código abierto dirigido a desarrolladores e investigadores. El proyecto se enfoca a las tecnologías de presentación, los cuales cubren diferentes aspectos de multimedia (gráficos, animación e interactividad). Este software utiliza la librería relacionada “libgpac” La herramienta utilizada conforme al propósito de las pruebas es la siguiente:
  
  o **MP4Box** [38], un empaquetador con la capacidad de añadir los datos de contenedor conforme a un archivo MPEG4 de multimedia, a un flujo elemental de video.

### VI.1.4 Configuración de codificadores

Por lo general cuando se utiliza un codificador de video, tanto la calidad de imagen como la tasa de compresión del video codificado pueden ser ajustadas mediante parámetros que controlan el esquema de codificación. Por esta razón, es necesario conocer como contribuye cada uno de estos parámetros en la codificación, para obtener el resultado deseado en la codificación. Es notable mencionar, que al establecer estos parámetros, se especifica la medida objetiva que se desea obtener, pero el desempeño final estará sujeta a las limitaciones de lo que el codificador puede entregar bajo dicho esquema de codificación.
Resultados

Aunque la mayoría de los codificadores contienen, una multitud de parámetros sobre las cuales se puede personalizar el esquema de codificación, solo algunos de ellos son contenidos bajo cierto estándar y otros son particulares para cada codificador en cuestión. En las pruebas de evaluación de los codificadores, realizadas en el trabajo de tesis, se ha enfocado a codificar bajo un esquema de codificación conforme al estándar MPEG-4 ASP.

Además de los parámetros que especifican las características del las secuencias de video de prueba (explicados anteriormente), los más relevantes que se utilizaron en las pruebas de evaluación de los codificadores, se mencionan a continuación,

- **Bitrate:** alude al número de bits procesados por unidad de tiempo en el códec. La codificación basándose en el bitrate permite alcanzar el mejor nivel de calidad posible, mientras se mantiene un rango predecible en el tamaño del archivo. Este parámetro puede ser configurado utilizando dos opciones: codificación con velocidad Constante (CBR) y codificación de velocidad variable (VBR). En CBR, el número de bits permanece casi constante durante el transcurso de toda la transmisión. Con respecto a VBR, dependiendo de la complejidad de movimiento presente en el video, el codificador asigna una mayor o menor cantidad de bits de salida. En las pruebas relacionadas se ha elegido verificar el desempeño de ambos codificadores bajo una velocidad de bitrate constante (CBR).

- **Pasos de Codificación:** En muchos de los codificadores, existen dos opciones de codificación: codificación con paso único y paso doble. En la primera opción, el contenido pasa a través del codificador una sola vez y la compresión es aplicada en la medida en que el contenido es encontrado. En la segunda opción, el contenido se analiza durante un primer paso y es codificado una segunda vez, basándose en la información adquirida en el primero paso. Debido al incremento en el tiempo de codificación con respecto a la codificación por paso único, ese tipo de codificación no es buen candidato para codificación de video en tiempo real. Por esto, en las pruebas relacionadas se ha elegido codificación de un solo paso (si es aplicable en el codificador).

- **GOP:** Un GOP (siglas en inglés de Group of Pictures: Agrupación de Cuadros) es la estructura base dentro de la secuencia completa del video codificado, el cual agrupa bajo un cierto orden los diferentes cuadros de video codificados (estos
pueden ser Intra, Inter y Bi-direccionales). Con la información contenido dentro de un GOP, una sub-secuencia de la secuencia completa de video puede ser decodificada, sin necesidad de información adicional. En los codificadores, por lo general se puede establecer el orden e intervalo de los GOP mediante diferentes parámetros. Tales parámetros pueden ser el intervalo de repetición de cuadros Intra (cuadros “I”), la distancia entre cuadros de referencia (cuadros “I” o “P”) y número máximo de cuadros bi-direccionales entre cuadros de referencia (cuadros “B”). El GOP especificado para las pruebas del codificador MPEG-4 ASP se muestra en la figura VI.1, en esta figura se representa un esquema de GOP típico para este tipo de codificador. De manera similar, el GOP especificado para las pruebas del codificador MPEG-4 SP contiene la misma duración de cuadros, sin embargo los cuadros “B” son remplazados por cuadros “P”.

![Figura VI.1 Ordenación de cuadros dentro del GOP especificado para el codificador MPEG-4 ASP.](image)

**VI.1.4 Metodología de la evaluación**

En esta sección se describe los procesos de las pruebas que se realizaron para hacer la evaluación de ambos codificadores, el codificador MPEG-4 ASP de Xvid portado en la plataforma Linux embebido bajo el procesador ARM9 del DM355 y el codificador MPEG-4 SP de TI para el DM355. Para cada codificador, se codificaron las tres secuencias establecidas (“bus”, “foreman” y “news”). Además, para cada secuencia, el esquema de codificación fue sujeta a diferentes niveles de bitrate constante (CBR de: 64kbs, 128Kbps, 384Kbps, 512Kbps y 1024Kbps). Al finalizar de codificar bajo la plataforma DaVinci todas las secuencias bajo diferentes niveles de bitrate en ambos codificadores, se obtuvieron las treinta secuencias de video codificadas esperadas. Posteriormente las secuencias son decodificadas en la
computadora anfitriona. A partir de la secuencia reconstruida (decodificada) y la secuencia original se obtiene la métrica de calidad PSNR. Además, se calcula el porcentaje de compresión a partir de la secuencia codificada y la secuencia original. A continuación se describen con más detalle, las pruebas de evaluación realizadas para ambos codificadores.

En la figura VI.2 se muestra el proceso de evaluación del codificador MPEG-4 ASP en la plataforma Linux embebido bajo el procesador ARM9 del DM355. El proceso al proporcionar una secuencia de video hacia el sistema de archivos de la tarjeta DaVinci (“Video YUV 4:2:0 P” bloque izquierdo de la figura VI.2). Posteriormente, esta secuencia se procesa por el codificador MPEG-4 ASP (Fig. VI.2 “Codificador MPEG-4 ASP Xvid”) bajo el procesador ARM9 del DM355; como resultado se produce una secuencia de video codificado (Fig. VI.2 “Video.M4V”). Este archivo de la secuencia codificada se envía a la computadora anfitriona, la cual comparte conexión bajo una misma red local utilizando el protocolo conocido como TFTP (Trivial File Transfer Protocol).

**Figura VI.2 Esquema de prueba para evaluar PSNR del codificador MPEG-4 ASP.**

Al terminar el envío de la secuencia codificada (Fig. VI.2 “Video.M4V”), esta es decodificada en la computadora anfitriona utilizando el decodificador MPEG-4 ASP. Sin embargo, la secuencia resultante de la decodificación no es idéntica a la original (Fig. VI.2 “Video.YUV’ 4:2:0 P”), debido a que ha sido alterada por el proceso de codificación. Para evaluar estos cambios se apoya en la medida de PNSR del componente de luminancia (Fig. VI.2 “MATLAB PSNR(Y)”), la cual es calculado entre una copia local de la secuencia de
video original y la secuencia decodificada mediante instrucciones en un archivo de comandos de Matlab en la computadora anfitriona.

El proceso de evaluación para el codificador MPEG-4 SP de TI para el DM355 es similar al previamente expuesto para el codificador MPEG-4 ASP (Figura VI.2). Sin embargo, varía con respecto a ciertas modificaciones. Primeramente, debido a que el decodificador del MPEG-4 SP de TI no es compatible con video en formato YUV 4:2:0 P se ha visto a la necesidad de remplazar el decodificador por una herramienta de conversión de formatos conocido como FFMPEG (Fig. VI.3 “Decodificador FFMPEG”), el cual fue previamente expuesto en este capítulo. Además, a pesar que un decodificador se ha encontrado con FFMPEG, existe el problema de compatibilidad entre el decodificador FFMPEG y el formato de video codificado del MPEG-4 SP de TI (Fig. VI.3 “Video.M4V”). Para solucionar esto, otro proceso es introducido mediante el uso de la herramienta MP4Box (Fig. VI.3 “Empaquetador MP4Box”), la cual añade el contenedor conforme a un archivo de multimedia MPEG-4 necesario para ser compatible con el decodificador MPEG-4 contenido en FFMPEG. Con estos procesos es suficiente para completar los procesos y obtener la métrica de PSNR entre la secuencia original y la secuencia decodificada.

**Figura VI.3 Prueba para evaluar PSNR de codificador MPEG-4 ASP de XVID en DM355.**

### VI.2 Resultados de la evaluación de los codificadores

La evaluación realizada sobre cada secuencia a diferentes bitrate, genera dos graficas de desempeño: bitrate/calidad de video (PSNR) y bitrate /porcentaje de compresión (con respecto al tamaño de la secuencia original en bytes). Estas graficas contienen curvas que presentan la
variación de la calidad de video codificado y el tamaño del archivo generado por el mismo, ambos valores están en función al bitrate elegido en el proceso de codificación. A continuación se exponen los resultados mediante una interpretación de dichas curvas.

VI.2.1 Resultados de PSNR

En esta sección se muestran las gráficas de desempeño obtenidas a partir de las pruebas de evaluación de los codificadores: MPEG-4 ASP portado para el ARM9 del DM355 y el MPEG-4 SP desarrollado por TI. Las gráficas (Figs. VI.4, VI.5 y VI.6) muestran la calidad (medido en PSNR) de las secuencias reconstruidas bajo diferentes esquemas de bitrate en la codificación.

En base a los resultados, se puede observar que con respecto a la medida de calidad del video, el codificador MPEG-4 ASP portado al ARM9 del DM355 mantuvo mejor desempeño que el codificador MPEG-4 SP de TI para las tres secuencias “Bus”, “Foreman” y “News” que representan secuencias con diferentes niveles de movimiento.

Con respecto a desempeño total sobre las tres secuencias para ambos codificadores, se observa que se obtiene mejor calidad en la secuencia “News” (Fig. VI.6) que corresponde a una secuencia con movimiento de categoría baja, seguido por “Foreman” (Fig. VI.5) que presenta una secuencia con movimiento de categoría intermedia y finalmente la peor calidad se obtuvo en la secuencia “Bus” (Fig. VI.4), la cual presenta una secuencia con movimiento de categoría alta en escena.

Para la gráfica de la secuencia “Bus” (Fig. VI.4), se puede observar que la calidad por el codificador MPEG-4 ASP portado al ARM9 del DM355 supera en calidad para todos los bitrate al codificador MPEG-4 SP. Especialmente a altos niveles de bitrate, donde el margen de diferencia con respecto al PNSR es mayor con respecto a todas las pruebas realizadas.
Resultados

**Capítulo VI**

Procesamiento de video utilizando la plataforma DaVinci

Figura VI.4 Gráfica de desempeño Bitrate /Calidad de video (PSNR) para la secuencia de “Bus”.

Con respecto a la gráfica de la secuencia “Foreman” (Fig. VI.5), se observa que la calidad entre codificadores es casi similar a bajo bitrate (64Kbps). Sin embargo, a niveles intermedios de bitrate el margen de diferencia en PSNR aumenta, pero a partir de 384Kbps este margen de diferencia se mantiene a niveles altos de bitrate.

Figura VI.5 Gráfica de desempeño Bitrate /Calidad de video (PSNR) para la secuencia de “Foreman”.
Con respecto a la gráfica de la secuencia “News” (Fig. VI.6), se observa que la calidad entre codificadores es casi similar a bajo bitrate (64Kbps). Sin embargo, conforme aumenta el bitrate a niveles bajo-intermedio (128Kbps) el margen de PSNR aumenta. A partir de este bitrate, se mantiene un margen de diferencia casi constante hacia niveles de bitrate altos.

![Gráfica de desempeño Bitrate /Calidad de video (PSNR) para la secuencia de “News”](image)

**Figura VI.6 Gráfica de desempeño Bitrate /Calidad de video (PSNR) para la secuencia de “News”**

**VI.2.2 Resultados de Compresión**

En esta sección se compara el desempeño de los codificadores en base a las gráficas de porcentaje de compresión bajo diferentes esquemas de bitrate (Figs. VI.7, VI.8 y VI.9) obtenidas a partir de las pruebas de evaluación. La medida de compresión utilizada es la “tasa de compresión”, la cual representa la fracción del tamaño de la secuencia de video codificada con respecto al tamaño de la secuencia de video original (codificada/des-codificada). Además, para facilitar la interpretación de la tasa de compresión en las gráficas (Figs. VI.7, VI.8 y VI.9) se presentan en forma de porcentaje. Para la interpretación del desempeño de un codificador con respecto a medida de tasa de compresión, un valor el cual es más cercano a cero representa un mejor desempeño. Esto no debe confundirse con la tasa de ahorro de compresión, dado que en esta medida, un valor obtenido es mejor si es más cercano al valor que representa un ahorro de compresión al 100%.
Con respecto a desempeño total sobre las tres secuencias para ambos codificadores, se puede resumir que se obtiene mejor compresión en la secuencia “News” (Fig. VI.9) (movimiento de categoría baja), seguido por “Foreman” (Fig. VI.8) (movimiento de categoría intermedia) y finalmente la peor compresión se obtiene en la secuencia “Bus” (Fig. VI.7) (movimiento de categoría alta).

![Diagrama de tasa de compresión/bitrate para "Bus" @ CIF](image_url)

**Figura VI.7 Gráfica de desempeño Bitrate /Porcentaje de Compresión para la secuencia de “Bus”.**
Resultados

Capítulo VI

Procesamiento de video utilizando la plataforma DaVinci

Figura VI.8 Gráfica de desempeño Bitrate /Porcentaje de Compresión para la secuencia de “Foreman”.

Figura VI.9 Gráfica de desempeño Bitrate /Porcentaje de Compresión para la secuencia de “News”.

Procesamiento de video utilizando la plataforma DaVinci
VI.3 Conclusiones

Con respecto a las capacidades de configurar el codificador MPEG-4 SP de TI, se observó de primera mano que la personalización de los parámetros es limitado tanto en el codificador como en el decodificador, durante las pruebas de evaluación. Estas limitaciones presentan problemas muy notorias, por ejemplo: al no proveer el manejo de diferentes formatos de espacios de colores para las secuencias de video que usualmente son utilizadas para evaluar a los codificadores.

Además, la secuencia de cuadros codificados en el codificador MPEG-4 SP de TI solo proporciona un elementary stream. Lo cual implica que las secuencias no pueden ser desplegadas en reproductores comunes. Esto no es problema en el codificador MPEG-4 ASP basado en Xvid ya que esta proporciona un stream el cual es conforme al estándar MPEG-4, el cual es reproducible en la mayoría de los reproductores.

Uno de los problemas de configurar el codificador MPEG-4 ASP del Xvid portado al ARM9 del DM355 resulta del hecho que no existe buena documentación sobre el codificador sobre el cual está basado (Xvid).

Utilizar un esquema de codificación a dos pasos presenta la dificultad del uso en sistemas que codifican en tiempo real, dado que el paso adicional de codificación no es recomendable, por los tiempos adicionales que se necesitarán para lograr una mejor codificación para un cuadro de la secuencia.

En base a las pruebas evaluadas, se puede concluir que el codificador MPEG-4 ASP portado al ARM9 del DM355 es un buen candidato como codificador para sustituir el codificador MPEG-4 SP de TI dado que es superior en calidad de video.
CAPÍTULO VII CONCLUSIONES

En este trabajo de tesis se presentó la implementación de un codificador Xvid apegado al estándar MPEG-4 ASP que opera bajo la arquitectura del procesador ARM9 del DM355 y un SO Linux embebido. Es notable mencionar que al trabajar bajo el ambiente de la plataforma de desarrollo DaVinci se necesita un conocimiento interdisciplinario para llevar a cabo una implementación apegándose a diversos estándares, tales como los estándares de programación para utilizar algoritmos en tiempo real dentro de los procesadores de Texas Instruments (xDAIS, xDM, CE), el estándar MPEG-4 para el codificador Advanced Simple Profile y los estándares de los API’s de Linux (por ejemplo V4L2) por nombrar algunos. Este esfuerzo por apegarse a los estándares, se recompensa con la interoperabilidad de la aplicación implementada con los diversos componentes de software y el hardware de la plataforma. A continuación se mencionaran las observaciones que resaltaron al trabajar con los componentes y los estándares en la implementación.

Linux está entre los sistemas operativos viables para trabajar en embebidos. Gracias a que Linux es publicado bajo acuerdos de software libre, las diferentes versiones del kernel se pueden obtener con código abierto, lo cual permite aligerar el kernel y personalizarlo según las necesidades de los sistemas embebidos. Sin embargo, la personalización del kernel para esta plataforma no está contemplada en el marco de este trabajo de tesis, ya que esta labor ha sido contemplada por Montavista con su versión de Linux personalizada para la plataforma DaVinci.

El estándar para el API multiplataforma Códec Engine que opera en conjunto con los algoritmos modulares xDM/xDAIS, especifican las metodologías para operar el API así como las reglas de programación que un algoritmo convencional de audio, video, voz o imágenes debe cumplir para ser utilizado como un algoritmo que opera en tiempo real dentro de los procesadores de Texas Instruments (xDAIS, xDM, CE). Aunque, el estándar es muy extenso, según lo publicado por TI, se recompensa el trabajo con la interoperabilidad del algoritmo en plataformas similares de TI. Esto se debe al compromiso que cumple TI al construir las nuevas plataformas conforme a los estándares o por lo menos construir nuevos estándares encima de los anteriores (tal como xDM se construyo encima de xDAIS). Esto justifica el llegar a dominar los estándares de TI, ya que se puede seguir conservando un buen
codificador o algoritmo y seguirlo portándolo conforme la tecnología de TI va mejorando con procesadores de siguiente generación.

No se realizó el manejo de DMA mediante la estructura de IDMA 3 en la implementación del codificador MPEG-4 ASP de Xvid que se ejecuta en el procesador ARM9 del DM355.

En el trabajo de tesis no trabajo en la programación del co-procesador dedicado del DM355, ya que este es de código cerrado.
REFERENCIAS


Procesamiento de video utilizando la plataforma DaVinci
[38] MP4Box Overview > http://gpac.wp.institut-telecom.fr/mp4box/.