IMPLEMENTACIÓN HÍBRIDA HARDWARE SOFTWARE DEL ALGORITMO DE DETECCIÓN DE ROSTROS DE VIOLA-JONES SOBRE FPGA

Hernández, Ernesto del Toro [1], Cabrera Sarmiento, Alejandro [2], Sánchez Solano, Santiago [3]

(Recibido enero 2012, Aceptado abril 2012)

[1] Centro de Investigaciones en Microelectrónica, CUJAE
[2] Departamento de Automática Computación, CUJAE
[3] Instituto de Microelectrónica de Sevilla, IMSE-CNM, CSIC
ernesto@electrica.cujae.edu.cu, alex@electrica.cujae.edu.cu, santiago@imse-cnm.csic.es

Resumen: En este trabajo se describe el diseño e implementación de un sistema que se utilizará para realizar la detección de rostros siguiendo el algoritmo propuesto por P. Viola y M. Jones. En el diseño se ha utilizado una placa que contiene un FPGA Virtex-II Pro acoplado con un sensor de visión tipo CMOS. Se han implementado estructuras hardware que permiten la aceleración de la ejecución de este algoritmo en un 37%. Una de las ventajas de este diseño radica en la posibilidad de utilizar imágenes con un tamaño variable para realizar la detección de rostros.

Palabras clave*:* Análisis de manchas/ FPGA/ Procesamiento de Imágenes/ Viola-Jones

HYBRID HARDWARE SOFTWARE IMPLEMENTATION OF VIOLA-JONES FACE DETECTION ALGORITHM USING AN FPGA

Abstract: This work presents a design strategy for face detection using the P. Viola and M. Jones proposal. The system uses a Virtex-II Pro FPGA and a CMOS vision sensor connection in order to fulfill usual requests of image processing algorithms. Hardware implementation of some parts of the algorithm allows a 37% gain in performance over the equivalent software solution. One on the main advantages of this design relies on the possibility of using multiple size images in the face detection process.

Keywords: Algorithm Acceleration/ FPGA/ Image Processing/ Viola-Jones

I. INTRODUCCIÓN

La extracción automática de información en secuencias de imágenes constituye una de las principales características de muchos sistemas de visión artificial dirigidos a diferentes campos de aplicación [1, 2]. En particular, los algoritmos y procedimientos de identificación y localización de rostros en imágenes o vídeos son ampliamente utilizados en sistemas de seguridad y otras aplicaciones orientadas al consumidor, como autofocalización de los rostros en cámaras digitales, juegos de ordenador que requieran tele-presencia o cámaras web que realizan seguimiento de individuos [1].

En los últimos años se han publicado muchas propuestas de algoritmos para detección de rostros basados en diferentes estrategias. Schneiderman y Kanade en [3] proponen un método estadístico para detectar rostros utilizando histogramas para representar una gran variedad de atributos visuales. Otro método de detección de rostros puede ser visto en [4], donde H. Rowley, S. Baluja y T. Kanade entrenan dos sistemas de clasificación basados en redes neuronales: el primero realiza una estimación de la pose de un rostro en una imagen y el segundo se encarga de la detección de rostros en sí. Existen otras propuestas donde se realiza la detección de rostros teniendo en cuenta las texturas y colores de la piel [5, 6].

Una de las técnicas de detección de rostros más ampliamente utilizada es la propuesta por P. Viola y M. Jones en [7]. Este algoritmo, basado en etapas de clasificación de complejidad creciente, es valorado como uno de los mejores en cuanto a tiempo de ejecución y efectividad de detección [1, 2, 8, 9].

Una característica común a la mayoría de los algoritmos de detección de rostros es que requieren una elevada carga computacional. Aunque esta circunstancia no representa un inconveniente importante para los procesadores disponibles en los ordenadores actuales, sí puede suponer un serio inconveniente para su aplicación en sistemas empotrados con velocidades de operación y recursos de

cálculo limitados. Para abordar con éxito el desarrollo de nuevas aplicaciones para sistemas empotrados cuya función será la de sensores de visión distribuidos, surge, por tanto, la necesidad de adaptar este tipo de algoritmos y buscar implementaciones eficientes que combinen componentes hardware y software para acelerar su ejecución.

Por la forma en que se afronta el problema de detección de rostros y por la relativa alta velocidad de ejecución del algoritmo propuesto por Viola y Jones en comparación con otras propuestas, se ha podido encontrar en la bibliografía consultada muchos autores que han abordado la implementación y aceleración del mismo [10-16].

G. Changjian y L. Shih-Lien en [10] utilizaron una tarjeta que contaba con un conector PCIe y un FPGA Virtex 5 LX110T de Xilinx. En conjunto con un software que se ejecuta en un procesador Intel Core 2 Duo a 2.66 GHz con 8 GB de memoria RAM los autores reportan una velocidad de detección de 37 imágenes por segundo. En este trabajo las imágenes deben ser grabadas al FPGA como paso previo a la detección y su tamaño de 256x192 puntos no puede variar.

L. Hung-Chih et al. en [11] utilizan un módulo hardware para implementar una pirámide de imágenes que le permita realizar el barrido en distintas resoluciones. Luego utilizan un buffer para calcular la imagen integral de la ventana que están procesando y a continuación realizar el proceso de cálculo de los clasificadores. Los autores utilizan una placa que contiene un FPGA Virtex-II Pro XC2VP30 e implementan 52 clasificadores en una sola etapa. Las imágenes donde se realiza la detección deben tener un tamaño fijo de 640x480 puntos y el conjunto de los clasificadores no puede ser modificado.

J. Cho, B. Benson, S. Mirzaei y R. Kastner en [12] utilizan un FPGA de Xilinx Virtex-5 LX330 y en él implementan la interfaz para capturar las imágenes directamente de un sensor de visión. Como el cálculo de los clasificadores se realiza utilizando una ventana de tamaño fijo, los autores implementan el hardware necesario para disminuir progresivamente este tamaño de la imagen para poder realizar el barrido de la misma. Al realizar esta implementación, se fijan los factores de escalas por los cuales será modificada la imagen. En imágenes de 320x240 puntos se realiza el barrido en 14 escalas pre-fijadas que no se pueden modificar. En el trabajo se reporta el procesamiento de 61.02 imágenes por segundo y en comparación con una realización software equivalente los autores logran acelerar el proceso de detección. Una de las desventajas de esta implementación es que el tamaño de las imágenes a procesar es fijo y un cambio en el tamaño de las mismas representaría un cambio en todo el sistema.

En otros trabajos consultados se realiza la aceleración del proceso para un tamaño de imagen prefijado que no puede ser modificado luego de implementar el sistema [13-16].

En este trabajo se describe una sistema híbrido hardware software para realizar la implementación y aceleración del algoritmo de detección de rostros propuesto por P. Viola y M. Jones en [7]. El uso de una plataforma de desarrollo que contiene un FPGA resulta fundamental para llevar a cabo las distintas etapas de verificación funcional, ajuste de parámetros, revisión algorítmica y comparación de las distintas soluciones en cuanto a rendimiento, costo y efectividad de la implementación. Para la implementación del sistema de detección de rostros en este trabajo, se ha tenido particular cuidado en la posibilidad de modificar el tamaño de las imágenes luego de su implementación. Con este objetivo se ha diseñado hardware específico que acelere el proceso de detección siempre que no afecte las características de flexibilidad requeridas.

Las principales características del algoritmo de detección de rostro utilizado, así como las modificaciones realizadas para facilitar su implementación, son introducidas en la Sección II. La Sección III describe los detalles de implementación del sistema de detección, basado en una placa de Xilinx que contiene un FPGA Virtex-II Pro [17] a la que se ha conectado un sensor de visión Omnivision 6620 [18]. El sistema utiliza un procesador Microblaze, junto con un periférico para adquisición y visualización de imágenes, que servirá de base para la implementación software del algoritmo. En la Sección IV se proponen dos propuestas de implementación en hardware mediante la utilización de módulos de propiedad intelectual (IP) con el objetivo de acelerar el proceso de detección de rostros. Por último, la sección V contiene las principales conclusiones obtenidas de la realización del trabajo.

II. DESARROLLO

1. Algoritmo de Detección de Rostros Viola-Jones

La detección automática de rostros en na imagen digital es un problema que presenta numerosos inconvenientes, ya que no existe información a priori sobre el tamaño del rostro o su localización dentro de la imagen. Por estos motivos, los algoritmos de detección tienen que estar preparados para manejar rostros con múltiples tamaños y distinta apariencia, que pueden cambiar drásticamente según la posición, iluminación y expresión facial. En imágenes donde aparecen muchos individuos, o donde el vestuario cubre parte del rostro de las personas, es posible que se oculte parcialmente un rostro y se produzcan fallos en los algoritmos.

Descripción del algoritmo

La técnica de detección propuesta por Viola y Jones utiliza una serie de clasificadores, agrupados en etapas sucesivas, que responden a un conjunto de rasgos (features) para detectar la presencia en la imagen de un rostro. Siguiendo la propuesta de [7], en la descripción de este algoritmo se describirán primero las tres características fundamentales del mismo:

A) Imagen integral

El primer paso que se propone para la ejecución del algoritmo de detección de rostros es la elaboración de una imagen integral. Los autores definen una imagen integral

como aquélla en la cual cada punto contiene el resultado de la suma de los valores de todos los puntos situados por encima y a su izquierda en la imagen original, como se muestra en la Figura 1.

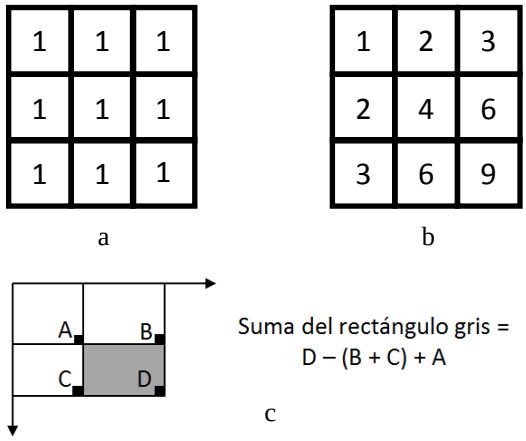

Figura 1. Cálculo de la imagen integral. (a) imagen original, (b) imagen integral, (c) Cálculo de la suma en una imagen integral.

A partir de la imagen integral se puede calcular la suma de todos los puntos contenidos en un rectángulo de la imagen a procesar utilizando solo los cuatro valores que se muestran en la Figura 1-c. Esta característica permite que el cálculo de la suma de los puntos contenidos en un rectángulo de tamaño arbitrario pueda ser realizado en un tiempo constante.

B) Construcción de los clasificadores

La construcción de los clasificadores para realizar la detección de rostros está basado en la selección, utilizando el algoritmo de entrenamiento denominado AdaBoost [19], de un pequeño número de rasgos, correspondientes a estructuras simples formadas por dos, tres o cuatro rectángulos como las que aparecen en la Figura 2. El proceso de aprendizaje excluye una gran cantidad de rasgos no esenciales y centra su atención en un pequeño conjunto de rasgos críticos para la detección de rostros. La evaluación de cada rasgo requiere el cálculo de la diferencia entre las suma de los puntos de las regiones blancas y grises, el cual puede ser realizado de forma muy eficiente al disponer de la imagen integral.

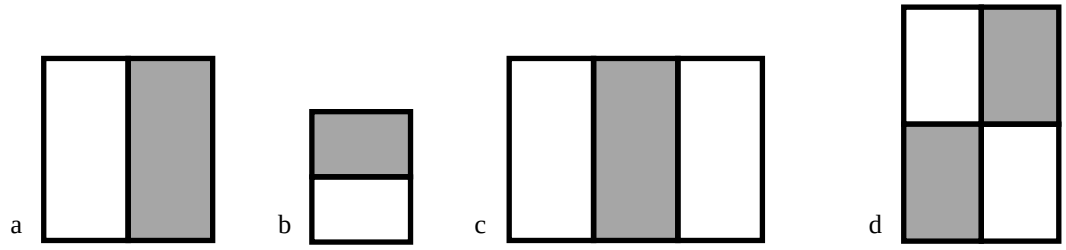

Figura 2. Ejemplo de los rasgos de 2 (a y b), 3 (c) y 4 (d) rectángulos utilizados para realizar la detección de rostros.

C) Creación de la estructura de los clasificadores

La tercera característica de este algoritmo de detección es la combinación en cascada de grupos de clasificadores cada vez más complejos (con mayor número de rasgos). Esta estrategia incrementa de forma considerable la velocidad de ejecución del algoritmo, ya que permite detectar rápidamente las zonas de la imagen que no contienen rostros y dedicar mayor esfuerzo a aquéllas en que hay más probabilidad de existencia de rostros en las primeras etapas.

De forma empírica P. Viola y M. Jones determinaron que la resolución inicial de la ventana para el entrenamiento fuera de 24x24 puntos. A partir de esta elección, los autores del algoritmo utilizaron una variante del algoritmo de entrenamiento AdaBoost, para agrupar los clasificadores en un conjunto de etapas como aparece en la Figura 3.

En cada etapa se evalúan los rasgos de los clasificadores y se comparan los resultados con valores umbrales obtenidos mediante el proceso de entrenamiento. Si el valor de un determinado clasificador es mayor que el umbral el resultado es un valor alpha determinado; si no, el resultado es cero. Los valores obtenidos se van acumulando en la etapa correspondiente y luego se comparan con un valor umbral de la etapa para decidir si la ventana en cuestión no es un rostro. Si se sobrepasa el valor umbral de la etapa, la ventana será transferida hacia otra etapa más compleja que volverá a repetir el proceso hasta que se evalúen todas las etapas.

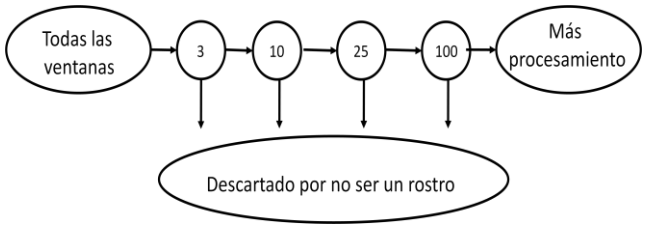

Figura 3. Descripción esquemática de una cascada de clasificadores. Los números representan la cantidad de rasgos que se evalúan en cada etapa.

Para completar el proceso de detección de rostros que encajen en la escala de referencia inicial, la ventana de tamaño 24x24 puntos mencionada con antelación debe desplazarse por toda el área de la imagen. A continuación es necesario aumentar el tamaño de la ventana y volver a realizar la operación de barrido. No obstante, la utilización de la imagen integral hace que el tiempo requerido para calcular los valores de los rasgos permanezca constante a medida que éstos aumentan de tamaño, de acuerdo a la ventana que se esté utilizando. Los valores empleados para escalar la ventana, los valores correspondientes a los pasos de barrido de la imagen completa y el tamaño final de la ventana se definen según la aplicación y el nivel de eficiencia o velocidad que se requiera. Al finalizar la evaluación de todas las etapas, las ventanas que no hayan sido descartadas en el proceso serán consideradas como rostros.

2. Implementación de un sistema para la ejecución del Algoritmo de Detección de Rostros

El desarrollo de algoritmos de procesado de imágenes suele llevar asociado una importante etapa experimental que permita verificar la funcionalidad de las implementaciones alternativas y comparar sus prestaciones. Por este motivo, el uso de una plataforma de desarrollo que facilite el particionado hardware-software de los algoritmos y proporcione mecanismos para adquirir y mostrar las imágenes constituye una importante herramienta para abordar el diseño de este tipo de sistemas.

Para la realización de este trabajo se ha utilizado una placa de Xilinx que incorpora una FPGA Virtex-II Pro xc2vp30 e incluye, entre otras características, un módulo de memoria DDR-RAM, interfaz VGA y conectores de entrada/salida al que se conectará un sensor de visión.

La Figura 4 muestra el diagrama de bloques de la plataforma de desarrollo, implementada mediante el entorno Xilinx Platform Studio y basada en un módulo IP del procesador Microblaze [20] conectado a distintos módulos IP que actúan como periféricos. Estos son: controlador de memoria, temporizador, puertos de entrada/salida paralelos y controlador de interrupciones. Esta interconexión se realiza a través de un bus de conexión de propósito general denominado PLB (Processor Local Bus) [21].

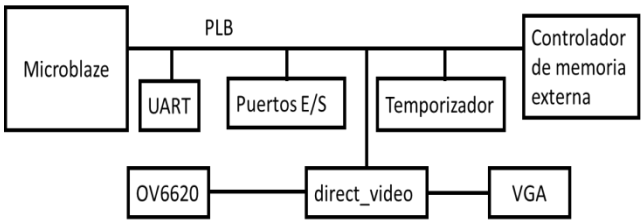

Figura 4. Diagrama de bloques de la plataforma de desarrollo.

Para completar el diseño de la plataforma fue necesario, construir un periférico, denominado **direct_video** en la figura 4, capaz de comunicar el sensor de visión utilizado con el procesador Microblaze y que, además, pudiera utilizarse para mostrar las imágenes a través del puerto VGA dejando libre al procesador para efectuar otras tareas. Las descripciones de los diferentes componentes de la plataforma se detallan en los siguientes apartados.

Interconexión para el sensor de visión y salida VGA

El chip OV6620 es un sensor de visión que incorpora dispositivos ópticos capaces de capturar hasta 60 imágenes por segundo de 352x288 puntos. Puede ser programado mediante una interfaz SCCB (Serial Camera Control Bus) para controlar parámetros como el tiempo de exposición, la ganancia, el balance de blanco y la matriz de colores, entre otros [10]. Dispone de un puerto de salida de 16 bits que se utiliza para transferir la imagen utilizando distintos formatos. El sensor proporciona también las señales de sincronismo clásicas para la transmisión de imágenes (sincronismo vertical, horizontal y reloj de píxel).

Para el desarrollo de los algoritmos de detección de rostros se ha utilizado el formato CIF (Common Intermediate Format) con una codificación YCbCr tipo 4:2:2. Utilizar este tipo de codificación facilita mucho el trabajo con algoritmos que utilizan solo la componente de luminancia de las imágenes. Teniendo en cuenta que la resolución de color del sensor de visión empleado no es muy grande, el uso de esta codificación para algoritmos basados en texturas de la imagen o diferenciación de colores no sería muy eficiente.

El diagrama de bloques del periférico **direct_video** desarrollado aparece en la Figura 5. En su realización se utilizó una memoria de doble puerto implementada con los componentes BRAM de la FPGA. La capacidad de esta memoria permite alojar la imagen transmitida por el sensor de visón (101376 puntos de 16 bits). La escritura en el puerto de datos A es controlada por una máquina de estados denominada *Write Sync FSM*. Esta máquina de estados está sincronizada con la señal de inicio de imagen generada por el sensor, que escribe ordenadamente los datos correspondientes a cada punto (8 bits de luminancia y 8 bits de croma U o V, según corresponda con el formato). El proceso de escritura en el puerto se realiza a la velocidad en que es transmitida la imagen por el sensor (8.86 MHz), mientras que para el proceso de lectura se utiliza el reloj del sistema (100 MHz). Se utilizan en el puerto A 17 bits de direcciones, una señal de habilitación de escritura y el reloj.

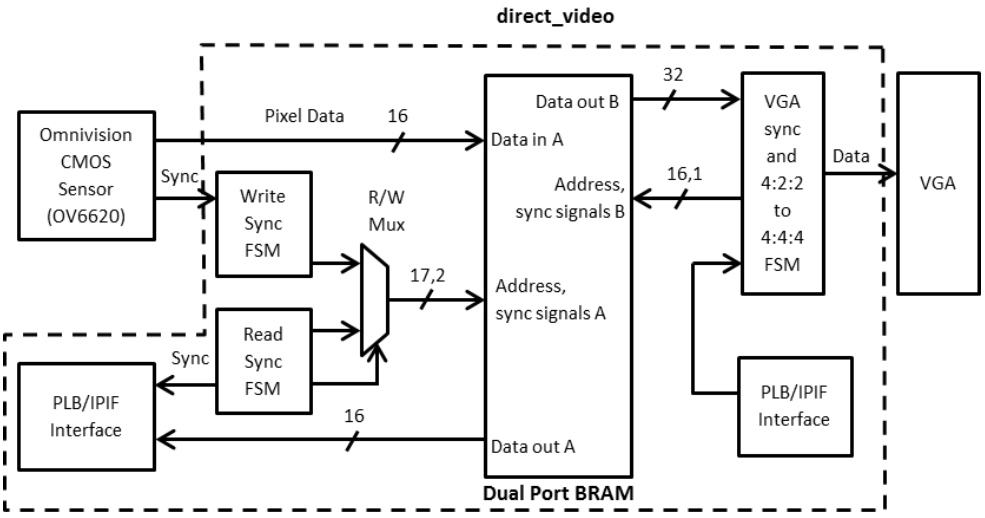

Figura 5. Diagrama del periférico direct-video (Los nombres de los bloques corresponden con los utilizados en el diseño).

Para mostrar la imagen en el puerto VGA, la máquina de estados *VGA Sync* se encarga de efectuar la lectura por el puerto de salida B de la memoria y realizar el cambio de formato de 4:2:2 a 4:4:4 necesario para mostrar correctamente los píxeles. El puerto B de la memoria es de sólo lectura y se utilizan 16 bits de direcciones y la señal de reloj. El cambio en la cantidad de bits de direcciones está dado por el hecho de que se leen 32 bits de datos (o sea el dato de dos puntos consecutivos) para poder realizar el cambio de formato mencionado anteriormente. Esta máquina de estados (VGA Sync) incluye además un conjunto de registros de escritura que permiten añadir un rectángulo a la imagen (desde el software ejecutado en Microblaze) con el objetivo de incluir en tiempo real referencias a los rostros detectados. Tras el cambio de formato, se utiliza un módulo IP proporcionado por Xilinx para convertir el tipo de dato YCrCb a RGB [22]. Todo el proceso de lectura en el puerto B está sincronizado con el reloj VGA que funciona a 50MHz.

Finalmente, el puerto de salida A será utilizado por Microblaze para leer la imagen almacenada en la memoria, a la que se aplicará el algoritmo de detección de rostros. Se ha empleado una estructura sencilla de multiplexado en combinación con otra máquina de estados (Read Sync FSM) que actúa sobre el bus de direcciones y las señales de lectura para no sobrescribir la memoria.

El diseño e implementación del sistema se llevó a cabo utilizando Xilinx Platform Studio versión 10.1.03. Para la síntesis del procesador Microblaze se habilitaron todas las opciones que permiten obtener mejor rendimiento, como uso de Barrel Shifter, multiplicación y división de enteros utilizando hardware específico y empleo de 5 etapas de pipeline. También se incluyeron 8 kB de cache de instrucciones y datos, respectivamente, el módulo IP de depuración de Xilinx (XMD) y un controlador de memoria externa necesario para utilizar los 256 MB de DDR-RAM disponibles en la placa. En la Figura 6 se muestra una fotografía del sistema en funcionamiento.

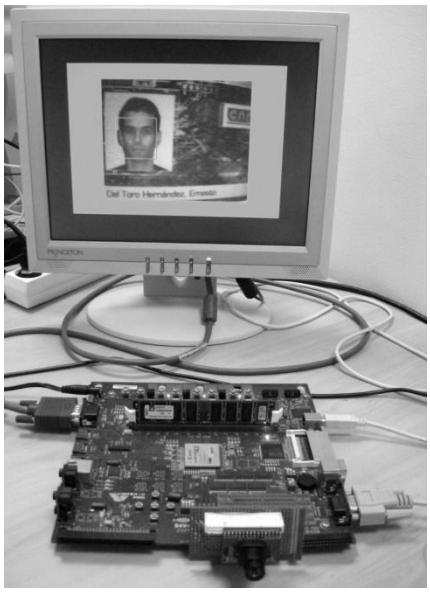

Figura 6. Sistema base para la implementación y prueba de algoritmos de detección de rostros

El sistema básico descrito ocupa el 54% de los Slices de la FPGA Virtex II-Pro (xc2vp30), el 5% de los bloques de multiplicación de 18x18 bits y 132 de los 136 bloques de memoria BRAMB (97%). La elevada utilización de bloques de memoria se debe en gran medida a la implementación de la memoria de doble puerto del periférico **direct_video** (99 en total, incluyendo los utilizados en la interfaz PLB-IPIF).

Análisis de la ejecución del algoritmo de reconocimiento facial

El sistema descrito fue utilizado, en primer lugar, para llevar a cabo una realización software del algoritmo de detección de rostros con el objetivo de detectar las partes críticas en la ejecución del mismo para acelerarlas mediante la transportación de estas funcionalidades hacia hardware implementado en el FPGA. La variante del algoritmo Viola-Jones implementada utiliza los datos de la cascada de clasificadores elaborados para CMUcam [23]. Estos contemplan 244 rasgos de clasificación distribuidos en 5 etapas de 9, 10, 25, 100 y 100 clasificadores, respectivamente. Entre los motivos por los que se decidió utilizar esta variante cabe destacar que está concebida para trabajar con el sensor de visión utilizado, por lo que al hacer la implementación se dispondrá de un modelo funcional con el cual comparar los resultados obtenidos.

Los datos correspondientes a los rasgos de clasificación fueron reescritos y compilados para el procesador Microblaze como constantes. Se utilizaron los mismos tamaños de ventanas que en el código original, predefinidos en: 30x30, 38x38, 48x48 y 60x60 puntos. El escalado de los rasgos para los nuevos tamaños de ventana se calcula con anterioridad para disminuir el tiempo de ejecución y se incluye en el código fuente junto con los valores de umbral y otros datos necesarios para ejecutar el algoritmo.

El cálculo de la imagen integral se realiza coincidiendo con la fase de adquisición de la imagen al iniciarse la ejecución del algoritmo. La aplicación incluye de forma independiente una funcionalidad que permite transmitir la imagen y su componente integral mediante el puerto serie. Esto posibilita la comprobación del algoritmo en más de una plataforma y permite utilizar los mismos datos al comparar diferentes alternativas de implementación.

Para calcular el tiempo de ejecución de este algoritmo, se realizaron una serie de 20 mediciones con distintas imágenes que incluían solo un rostro. El tiempo promedio de ejecución medido fue de 1940 milisegundos. Con objeto de evaluar la distribución del tiempo de ejecución entre las distintas partes del algoritmo e identificar las más adecuadas para una posible implementación hardware con el objetivo de acelerarlas, se utilizó la herramienta de análisis de rendimiento (*profiling*) que incorpora el software de desarrollo de Microblaze. Se trata de una herramienta intrusiva que hace que el compilador añada al código del programa funciones que interrumpen la ejecución del mismo con una frecuencia determinada y guardan en un espacio de memoria reservado el contenido del contador de programa. Una vez terminada la ejecución se realiza un análisis de estos datos y se determina qué función o parte del programa corresponde con cada valor del contador de programa. A partir de este análisis se determina cuánto tiempo estuvo el procesador ejecutando cada una de las funciones, cuántas llamadas a la función se hicieron y otros datos estadísticos. Los resultados se pueden visualizar en un gran número de formatos, como el diagrama de tarta que se muestra en la Figura 7.

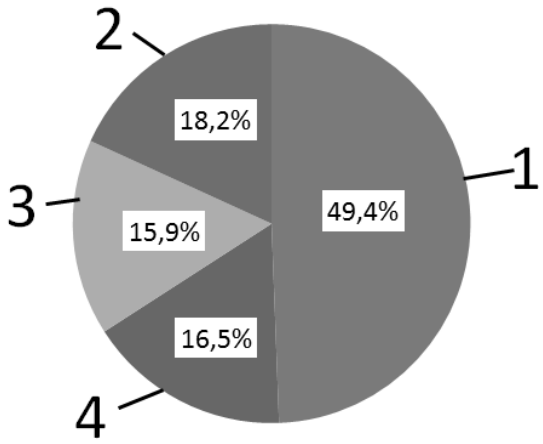

Figura 7. Análisis de rendimiento de la implementación software del algoritmo de detección de rostros.

La Figura 7 muestra el porcentaje de tiempo que está ejecutándose cada función en relación con el tiempo total de ejecución del programa. El número 1 corresponde a la función que realiza el cálculo de los rasgos en cada uno de los clasificadores, que ocupa casi la mitad del tiempo total de ejecución. El número 2 corresponde a la función que realiza el cálculo de la imagen integral. El número 3 al resto del algoritmo de detección de rostros donde se realiza el desplazamiento y cambio de tamaño de la ventana de detección y se guardan las ventanas marcadas como rostros. El número 4 está asociado a otras funciones no directamente relacionadas con la detección de rostros (comunicación con el puerto serie, muestra de los resultados, etc.).

Como se mencionó anteriormente, el diseño hardware inicial de la plataforma de desarrollo ocupa poco más del 50% de los recursos disponibles en la FPGA utilizada, lo cual hace pensar que es viable la implementación de un coprocesador que libere a Microblaze de la ejecución de determinadas partes del algoritmo con el objetivo de acelerar la ejecución del mismo. El análisis de rendimiento realizado permite llegar a la conclusión de que se ganaría en velocidad de ejecución si se acelera el cálculo de los valores de los rasgos de clasificación. A este razonamiento hay que añadir que dicho cálculo tendría que realizarse un mayor número de veces si se decidiera aumentar la base de datos de los clasificadores, con lo que amentaría aún más el tiempo invertido en la ejecución de esta función.

El cálculo de la imagen integral podría ser una segunda función candidata a ser acelerada mediante hardware. Incluso podría pensarse en realizar una modificación a la máquina de estados del periférico **direct_video**, para calcular la imagen integral al mismo tiempo que

Microblaze lee los puntos de la imagen a procesar. Sin embargo, los resultados obtenidos muestran que el cálculo de la imagen integral no demora tanto como la aplicación de todo el conjunto de clasificadores. Además hay que considerar que este cálculo se realiza solo una vez por cada imagen.

3. Aceleración del algoritmo mediante implementación hardware

La implementación software del algoritmo utilizando Microblaze permitió verificar su funcionalidad y ajustar los parámetros necesarios. Posteriormente, y con objeto de acelerar la velocidad de ejecución, se procedió a implementar la etapa correspondiente al cálculo de los clasificadores mediante una estructura hardware diseñado utilizando VHDL (Very-high-speed-integrated-circuit Hardware Description Language) y conectada como un módulo IP a Microblaze. Para llevar a cabo la implementación híbrida hardware software del algoritmo de detección de rostros se consideraron dos alternativas:

1.- Un coprocesador conectado mediante la interfaz FSL (Fast Simplex Link) [24].

2.- Un periférico con capacidad de acceder directamente a la memoria externa.

Coprocesador conectado mediante FSL

Como una primera alternativa para acelerar la ejecución del algoritmo se utilizaron las facilidades de Xilinx para el diseño e implementación de periféricos para construir un coprocesador conectado al procesador Microblaze mediante la interfaz de conexión rápida punto a punto denominada FSL. Este coprocesador se encarga de realizar las siguientes operaciones:

1.- Recibir los valores de los puntos de la imagen integral que corresponden al clasificador a evaluar en cada momento.

2.- Realizar el cálculo correspondiente a este clasificador utilizando los datos obtenidos en la etapa de entrenamiento. Este cálculo puede llegar a requerir 9 multiplicaciones enteras y 18 operaciones de suma.

3.- Realizar la multiplicación del valor anterior por el factor de escala correspondiente.

4.- Enviar al procesador el resultado obtenido tras la realización de las operaciones anteriores.

Los valores de los coeficientes (pesos) para el cálculo de los clasificadores son almacenados en memoria de bloque. La implementación del sistema se basa en una máquina de estados que controla la comunicación con el procesador Microblaze a través de puertos FSL. Este sistema ocupa el 69% de los recursos de la FPGA y 133 bloques de memoria de 136 disponibles. Se utilizan 12 bloques de multiplicación debido a que el coprocesador está concebido para realizar las multiplicaciones en serie con lo cual no hace falta utilizar muchos de estos elementos.

Utilizando el mismo conjunto de imágenes de test que en el caso de la implementación completamente software, el tiempo de promedio de ejecución se reduce a 1600 milisegundos utilizando el coprocesador para realizar las operaciones descritas anteriormente.

Periférico con capacidad de acceder directamente a la memoria externa

El núcleo de procesado de los clasificadores descritos en el apartado anterior es capaz de acelerar el proceso de cálculo debido a que disminuye el número de accesos a la memoria externa que contiene los parámetros del algoritmo. Una vez que se han leído por software los valores de los puntos de las esquinas de un rectángulo determinado, el coprocesador realiza la operación de multiplicación-suma en conjunto con los datos que tiene almacenados en memoria de bloque en el FPGA.

La otra alternativa propuesta en este trabajo para acelerar la ejecución del algoritmo de detección implica reducir los accesos a memoria y, si no es posible, al menos independizarlos del microprocesador.

Para cubrir este objetivo se ha diseñado un periférico con interfaz maestro PLB capaz de conectarse al controlador de memoria externa a través de un bus de conexión PLB independiente. Además, este periférico se conecta mediante una interfaz esclavo PLB al bus del sistema para recibir la orden de ejecución y comunicar el resultado de la operación.

Las operaciones realizadas por el nuevo periférico son equivalentes a las del coprocesador descrito anteriormente, salvo las correspondientes a la etapa de inicialización de datos:

1.- Recibe el número del clasificador, el número de la cascada, la escala utilizada y la dirección base de la ventana con la cual se está trabajando.

2.- A partir de estos datos se generan las direcciones de memoria de los puntos que se evaluarán y se realiza la lectura de los valores correspondientes a dichos puntos utilizando el acceso directo al controlador de memoria.

3.- El proceso continúa con las mismas operaciones que en el caso anterior.

El periférico diseñado se denominó **mem_connection** y su diagrama en bloques se puede apreciar en la figura 8. Para una mejor comprensión se ha dividido el diagrama en bloques en dos procesos:

1. El proceso de generación de las direcciones y su lectura en memoria externa.
2. El proceso del cálculo del rasgo.

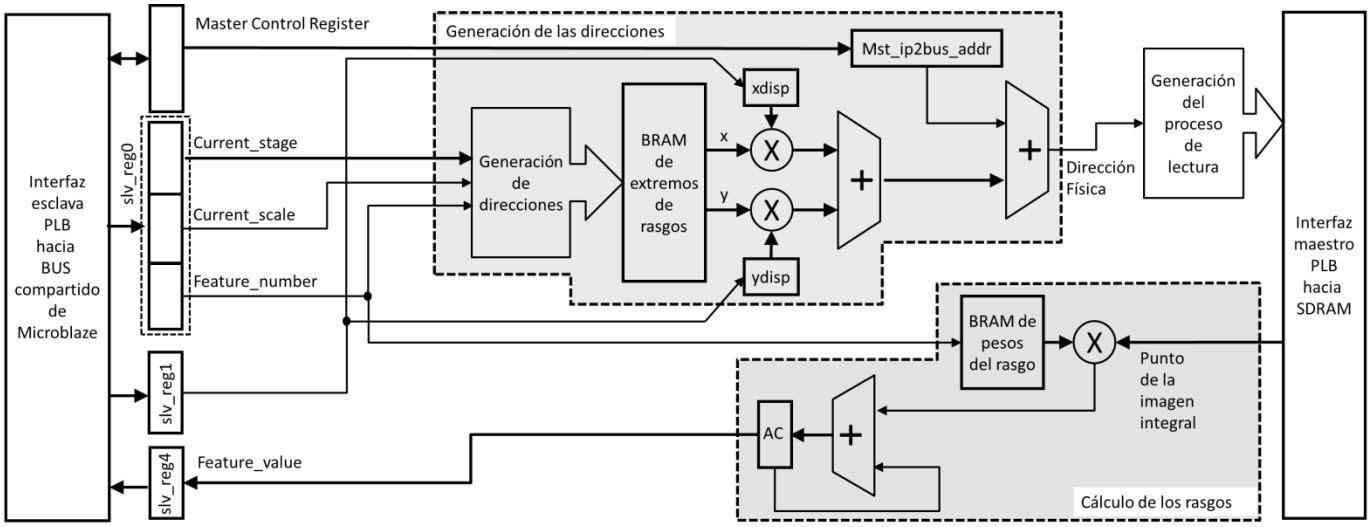

Figura 8. Diagrama de conexión del periférico mem_connection.

Para el proceso de generación de las direcciones, el periférico recibe mediante la escritura en un registro esclavo PLB de 32 bits (**slv_reg0**) el número del clasificador (**feature_number**), la escala a la que se está trabajando (**current_scale**) y el número de la etapa en donde se encuentra el clasificador (**current_stage**). Además el periférico recibe también la dirección base de la ventana deslizante en vez del punto de la imagen integral.

Esta es la principal diferencia con relación al software que se ejecuta en Microblaze. Como el periférico tiene un acceso propio a la memoria externa, desde el software es necesario enviarle la dirección base de la ventana deslizante.

Para realizar la acción correspondiente al paso dos (lectura de los puntos significativos de los rasgos) se utiliza la interfaz de control maestro de PLB y se ejecuta una orden de lectura en la dirección que corresponde a la dirección base de la ventana deslizante. Esta dirección se almacena en el registro **Mst_ip2bus_addr**. La máquina de estados para el control de la interfaz maestro ha sido modificada para ignorar el contenido de la primera lectura y luego efectuar nueve lecturas consecutivas que corresponden a las nueve posibles esquinas del rasgo que se va a calcular en ese momento. Estas lecturas se realizan calculando mediante los datos almacenados en la memoria interna del FPGA (**BRAM de extremos del rasgo**) el desplazamiento que hay que aplicarle a la dirección base para leer cada esquina determinada. En los casos en donde sólo es necesario utilizar seis u ocho puntos (debido al tipo de rasgo que se está evaluando) las restantes lecturas son ignoradas debido a que en los datos se ha grabado el valor de multiplicación cero que no influye en el resultado de la operación de multiplicación suma.

Para generar las nueve direcciones de lectura para la memoria externa a partir de los datos almacenados en la memoria de bloque en el FPGA se utilizan los 16 bits menos significativos del registro **slv_reg1** para modificar los registros de desplazamiento por filas y columnas. En la figura 8 aparecen estos dos registros nombrados **xdisp** y **ydisp**. Los datos que se encuentran almacenados en la memoria de bloque son relativos a la ventana y el desplazamiento por filas y columnas depende del tamaño de la matriz que contiene la imagen integral. Por ejemplo, a partir de las coordenadas relativas (x, y) el cálculo de la dirección absoluta de la memoria se realizaría con la siguiente operación:

$$Addr = base + [(x*xdisp)+ (y*ydisp)] \quad (1)$$

Para una matriz de 120 filas y 176 columnas que corresponde a la imagen el valor de **xdisp** sería 480 y **ydisp** tomaría el valor de 4. Este desplazamiento está directamente relacionado por la forma en que el compilador de C/C++ para Microblaze distribuye una matriz en la memoria. Los datos en las columnas (**ydisp**) son ordenados de forma consecutiva y por este motivo cada cuatro bytes (para el caso de un tipo de datos de 32 bits) se encontrará uno nuevo. Los datos en un desplazamiento por las filas dependen del tamaño de la ventana (específicamente de la longitud de las columnas) y como estos datos son de cuatro bytes es necesario tenerlo en cuenta (120*4).

Los registros **xdisp** y **ydisp** son modificables mediante el software que se ejecuta en Microblaze y permite que se pueda realizar la detección de rostros utilizando un tamaño de imagen variable sin modificar el hardware sintetizado. El registro de desplazamiento por las filas **xdisp** tiene un tamaño de 12 bits lo cual permite acceder imágenes con una altura de hasta 1024 puntos para tipos de datos de 32 bits. El registro de desplazamiento por las columnas **ydisp** tiene un tamaño de 4 bits lo cual permite que el tipo de dato utilizado en la imagen integral pueda ser 8, 16 o 32 bits; las restantes combinaciones en este registro quedarían reservadas para futuras implementaciones que pudieran incluir tipos de datos de 64 y 128 bits. El tamaño mínimo de las imágenes para poder realizar la detección de rostros sería de 30x30

puntos dado por la base de datos de los clasificadores utilizada.

Además de generar las direcciones de los puntos significativos para cada rasgo, la máquina de estados de control maestro se encarga también de generar el conjunto de señales necesarias para efectuar la lectura en memoria externa. Por ejemplo, estas señales son **bus_lock** para ganar acceso exclusivo al bus, **busy** para informar que se está en un estado ocupado, **byte_enable** para marcar los datos válidos entre otras.

Después de que cada punto en la imagen integral es leído, se activa el proceso de cálculo del rasgo correspondiente mediante la utilización de un bloque de memoria que se denomina **BRAM de pesos del rasgo** en donde se encuentra almacenado el valor de los pesos de los rasgos de clasificación. Como se puede apreciar en la Figura 8 este proceso es una operación de multiplicación-suma en donde se va acumulando el resultado de estas nueve operaciones. El valor del rasgo correspondiente queda almacenado en el registro esclavo **slv_reg4** que puede ser leído por el programa que se ejecuta en Microblaze. El resto del algoritmo es ejecutado por el software en Microblaze hasta que sea necesario realizar un nuevo cálculo de un rasgo para otro clasificador.

La interconexión de este periférico con el resto del sistema se muestra en el diagrama de bloques de la Figura 9.

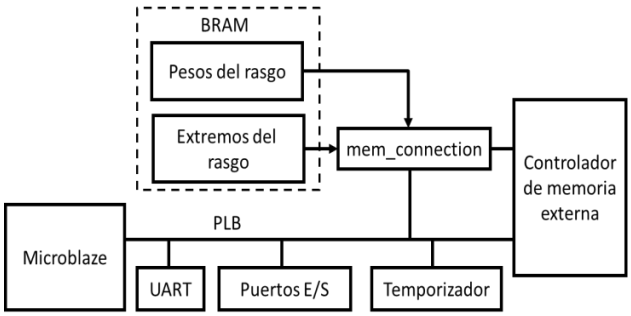

Figura 9. Diagrama del sistema con el nuevo periférico.

El sistema implementado sin el periférico **direct_video** ocupa 5751 Slices del FPGA utilizado (41%) y 37 bloques de memoria RAM (27%).

Para comparar las prestaciones de las diferentes opciones de implementación se volvió a realizar un análisis de rendimiento procesando el mismo conjunto de imágenes mediante un sistema que incluye las tres alternativas de diseño. Las diferentes zonas del diagrama de la Figura 10 muestran los tiempos de ejecución correspondientes al cálculo de los clasificadores cuando se utiliza una implementación totalmente software (1), el coprocesador conectado a través de los puertos FSL (2) y el periférico conectado directamente a memoria (3). La zona marcada con el número 4 representa el porcentaje de tiempo empleado en la elaboración de la imagen integral (8.3%). Las zonas que restan en sentido anti-horario equivalen al resto del algoritmo ejecutado en software (11%), utilizando el coprocesador (11%) y utilizando la conexión directa a memoria (10.4%). La zona que ocupa el 4.4% del tiempo de ejecución corresponde a otras funciones que no están relacionadas con el algoritmo (comunicación por puerto serie, atención a interrupciones entre otras).

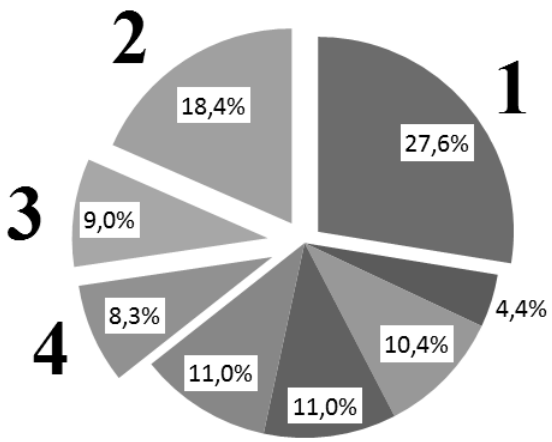

Figura 10. Análisis de rendimiento comparativo de las tres variantes de realización del algoritmo.

El análisis anterior pone de manifiesto la mejora que representa la utilización del periférico conectado directamente a memoria. El tiempo medio de ejecución del algoritmo completo para la detección de rostros utilizando el mismo conjunto de imágenes fue de 1220 milisegundos frente a los 1940 medidos en la ejecución software. Las mediciones realizadas permiten comprobar que se ha aumentado un 37% la velocidad de ejecución del algoritmo frente a la realización software. Una medición fina del proceso de cálculo de los rasgos (sólo la porción implementada en hardware) muestra una aceleración de diez veces con respecto a su contraparte software.

III. CONCLUSIONES

1. La implementación eficiente de algoritmos de procesado de imágenes de elevada carga computacional sobre sistemas empotrados con recursos limitados requiere el empleo de técnicas de diseño híbridas, que permitan acelerar la ejecución de las tareas críticas del sistema mediante su implementación en hardware dedicado.
2. En este sentido, el uso de hardware reconfigurable acelera en gran medida las etapas de elaboración del algoritmo, ajuste de parámetros y verificación funcional, y facilita la comparación entre las distintas opciones de diseño.
3. En este trabajo se ha descrito un sistema basado en FPGA para el desarrollo de un algoritmo de detección de rostros basado en la técnica de Viola-Jones. Se han propuesto dos alternativas para la aceleración del algoritmo que contemplan, respectivamente.

4. El empleo de un coprocesador conectado por FSL y el desarrollo de un periférico con capacidad de acceso directo a memoria. Los resultados obtenidos muestran una reducción del 17% en el tiempo de ejecución para el coprocesador y un 37% para una conexión directa a memoria.
5. Aunque en comparación con la literatura consultada este incremento en velocidad pudiera no apreciarse como significativo, la ventaja de la realización descrita en este trabajo radica en que en ambos casos la aceleración del proceso de detección no está condicionada por un tamaño de imagen prefijado.
6. El cálculo de los rasgos de clasificación se acelera hasta 10 veces de forma tal que no se relaciona con el tamaño de la imagen y se han creado estructuras hardware que permiten el acceso a los puntos significativos en un rango de tamaños de la misma.

IV. REFERENCIAS

1. M.-H. Yang, D. J. Kriegman and N. Ahuja, "Detecting faces in images: a survey " *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 24, no. 1, pp. 34 - 58, 2002.
2. B. Mathew, A. Davis and R. Evans, "A Characterization of Visual Feature Recognition," in *IEEE 6th annual workshop on workload characterization*, 2003, pp. 3-11.
3. H. Schneiderman and T. Kanade, "A statistical method for 3d object detection applied to faces and cars," presented at the IEEE Conference on Computer Vision and Pattern Recognition, Hilton Head Island, SC, 2000.
4. H. Rowley, S. Baluja and T. Kanade, "Neural Network-Based Face Detection," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 20, no. 1, pp. 23 - 38, 1998.
5. C. N. Ravi Kumar and A. Bindu, "An Efficient Skin Illumination Compensation Model for Efficient Face Detection," in *IEEE Industrial Electronics, IECON 2006 - 32nd Annual Conference on*, Paris, 6-10 Nov 2006, pp. 3444 - 3449.
6. S. Krishna and S. Panchanathan, "Combining Skin-Color Detector and Evidence Aggregated Random Field Models towards Validating Face Detection Results," in *Computer Vision, Graphics & Image Processing, 2008. ICVGIP '08. Sixth Indian Conference on*, 16-19 Dec. 2008 2008, pp. 466-473.
7. P. Viola and M. Jones, "Rapid object detection using a boosted cascade of simple features," in *Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition. CVPR*, 2001, vol. 1.
8. A. Bigdeli, C. Sim, M. Biglari-Abhari and B. C. Lovell, "Face Detection on Embedded Systems " in *Proceedings of the 3rd International Conference on Embedded Software and Systems: Lecture Notes in Computer Science*, Daegu, Korea, 14 - 16 May, 2007 2007, vol. 4523, pp. 295 - 308
9. C. Zhang and Z. Zhang, "A Survey of Recent Advances in Face Detection," Microsoft Research 2010.
10. G. Changjian and L. Shih-Lien, "Novel FPGA based Haar classifier face detection algorithm acceleration," in *International Conference on Field Programmable Logic and Applications. FPL 2008.* , 8-10 Sept. 2008, pp. 373-378.
11. L. Hung-Chih, M. Savvides and C. Tsuhan, "Proposed FPGA Hardware Architecture for High Frame Rate Face Detection Using Feature Cascade Classifiers," in *First IEEE International Conference on Biometrics: Theory, Applications, and Systems. BTAS 2007.*, 27-29 Sept. 2007 2007, pp. 1-6.
12. J. Cho, B. Benson, S. Mirzaei and R. Kastner, "Parallelized Architecture of Multiple Classifiers for Face Detection," presented at the IEEE International Conference on Application-specific Systems, Architectures and Processors, 2009.
13. T. Theocharides, N. Vijaykrishnan and M. J. Irwin, "A parallel architecture for hardware face detection," in *IEEE Computer Society Annual Symposium on Emerging VLSI Technologies and Architectures*, 2-3 March 2006 2006, vol. 2.
14. M. Hiromoto, K. Nakahara, H. Sugano, Y. Nakamura and R. Miyamoto, "A Specialized Processor Suitable for AdaBoost-Based Detection with Haar-like Features," in *IEEE Conference on Computer Vision and Pattern Recognition. CVPR '07*, 17-22 June 2007 2007, pp. 1-8.
15. D. Zheng, Z. Feng, W. Tinghui, S. Wei and W. Min-You, "Hecto-Scale Frame Rate Face Detection System for SVGA Source on FPGA Board," in *IEEE 19th Annual International Symposium on Field-Programmable Custom Computing Machines (FCCM)*, 1-3 May 2011 2011, pp. 37-40.
16. R. C. Luo and L. Hsin-Hung, "Design and implementation of efficient hardware solution based sub-window architecture of Haar classifiers for real-time detection of face biometrics," in *International Conference on Mechatronics and Automation (ICMA)*, 4-7 Aug. 2010 2010, pp. 1563-1568.
17. Xilinx Inc. (2008). *Xilinx University Program Virtex-II Pro Development System, Hardware Reference Manual*.
18. Omnivision. (2000, OV6620 single-chip cmos cif color digital camera, datasheet.
19. S. Theodoridis and K. Koutroumbas, "The AdaBoost Algorithm.," in *An Introduction to Pattern Recognition: A Matlab Approach.*, ed: Elsevier, 2010, pp. 63 - 66.

20. Xilinx Inc. (2008). *MicroBlaze Processor Reference Guide*.
21. Xilinx Inc. (2008). *PLB Usage in Xilinx FPGAs*.
22. Xilinx Inc. (2008). *YCrCb to RGB Color-Space Converter v1.0. Product Specification*.
23. A. Rowe, A. Goode, D. Goel, C. Rosenberg and I. Nourbakhsh. (2011). *CMUcam3: Open Source Programmable Embedded Color Vision Platform.* Available: http://www.cmucam.org/
24. Xilinx Inc., "Fast Simplex Link (FSL) Bus (v2.11a), DS449," ed, 2008.