Conversión de RGB a YCbCr en System Generator y HLS

Romina Molina and Veronica Gil-Costa

Universidad Nacional de San Luis
San Luis, Argentina
{rmolina,gvcosta}@unsl.edu.ar

Abstract. Los sistema de recuperación de imágenes basados en contenidos (CBIR) típicamente ejecutan dos tareas de alto costo computacional: (1) generación de la base de datos e indexación y (2) recuperación. Este trabajo se enfoca en la primera tarea la cual es inherentemente paralela, debido a que las imágenes son generalmente divididas en varias partes y cada parte se procesa por separado y de manera similar. Para ello se presenta un conversor de espacio de color de RGB a YCbCr para plataformas de FPGAs basadas en SoCs. Esta conversión de color se utiliza como parte del proceso de extracción del descriptor de la distribución de color, el cual forma parte del estándar MPEG-7 y es utilizado en sistemas CBIR. Se presenta la implementación del conversor haciendo uso de System Generator y de Vivado HLS y se efectúa una comparación de los resultados sobre la plataforma ZYNQ - ZC7020 Evaluation Kit.

Keywords: MPEG-7, system generator, descriptores, HDL, multimedia.

1 Introducción

La recuperación de imágenes basada en contenido (CBIR) es un área cuyo objetivo es desarrollar herramientas para recuperar información visual y tiene aplicaciones importantes en muchas áreas incluyendo medicina, educación, diseño de arquitecturas, agricultura. Para representar una imagen, se utilizan los contenidos visuales como el color, la forma, la textura, etc..

El estándar de representación MPEG-7 (también conocido como Interfaz de Descripción de Contenido Multimedial) ha sido amplianente utilizado en los últimos años. Surge debido la necesidad de estandarizar la repesentación del contenido de la información multimedia. Trabajos como [1] y [2] exponen mejoras de MPEG-7 respecto a MPEG-4. Es un estándar ISO/IEC desarrollado por el grupo MPEG [3],[4],[5]. Dentro de sus especificaciones se definen la semántica y la sintáxis para lograr una adecuada descripción del contenido.

MPEG-7 se encuentra estructurado en 7 partes: sistemas, lenguaje de descripción y definición, visual, audio, entidades genéricas y esquemas de descripción multimedia(MDS), software de referencia y conformance testing. En

la parte visual de MPEG-7, los descriptores se clasifican en: descriptores normativos (describen colores, formas, texturas y movimiento de datos visuales), descriptores básicos y descriptores para localización. Dentro de los campos de aplicación de este estándar se encuentran: bibliotecas digitales, aplicaciones biomédicas [6], búsqueda de personas, aplicaciones educativas, control de tráfico, sistemas de vigilancia [7], edición multimedia, entre otras.

La contrucción de los descriptores MPEG-7 que representan las imágenes, es un proceso computacionalmente intensivo lo cual lo hace un buen candidato para ser paralelizado. Por lo tanto, en este trabajo se propone utilizar plataformas paralelas que permitan acelerar la construcción de estos descriptores. En particular, se propone utilizar las FPGAs basadas en SoC (System on Chip) que han mostrado ser una solución eficiente debido a su paralelismo inherente. Las FPGAs son de alta eficiencia energética y proporcionan un alto poder de cómputo debido a la posibilidad de adaptar los diseños basados en FPGA a una arquitectura específica. Estas características las hacen ideales para las operaciones de procesamiento de imagen o visión por computador. Los SoCs hacen referencia a las nuevas tecnologías que integran un microcontrolador, procesador, DSPs, módulos de memoria, osciladores, contadores, temporizadores, interfaces externas, AD/DA, entre otros componentes. Debido a la complejidad de los chips, esta tecnología puede ser programada, no solo con VHDL o Verilog, sino con HDL de más alto nivel como SystemVerilog, SystemC, C/C++. Trabajos como [8], [9] reflejan el uso de los SOC en el procesamiento de imágenes.

Recientemente se ha incrementado el uso de System Generator para realizar diseños hardware para FPGA. Áreas como control [12], electrónica de potencia [13], procesamiento de señales [14],[15], entre otras. En lo que respecta al procesamiento de imágenes, ha sido utilizado para implementar las operaciones básicas como binarización, conversión a escala de grises y mejora del contraste [16], [17], [18].

Este paper se enfoca específicamente en el desarrollo del bloque correspondiente a la etapa del cambio de espacio de color de RGB a YCbCr, implementado mediante System Generator y Vivado HLS, y se compara la utilización de los recursos que reporta cada una de esas herramientas. Este trabajo se organiza de la siguiente manera. La Sección 2 describe el proceso de construcción de descriptores para imágenes utilizando el Color Layout Descriptor de MPEG-7. En la Sección 3 de detalla la implementación de la etapa que involucra la conversión del espacio de color RGB a YCbCr utilizando System Generator y Vivado HLS. La Sección 4 presenta compara el número de recursos requeridos por System Generator y Vivado HLS. Finalmente, se concluye en la Sección 5.

2 Generación de descriptores de color en MPEG-7

Un descriptor es una representación de una característica definida sintáctica y semánticamente. Una imagen puede quedar caracterizado por más de un descriptor. Uno de los descriptores más utilizados es el Descriptor de la distribución de color (CLD - Color Layout Descriptor), el cual permite obtener la distribución

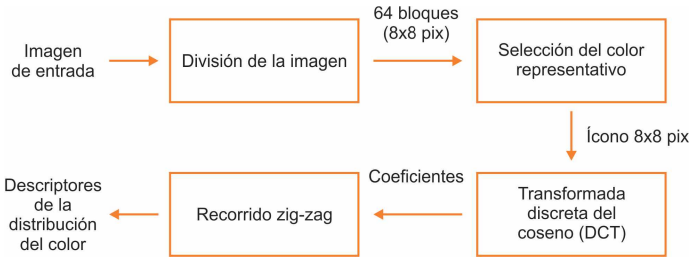

Fig. 1. Proceso de extracción de la distribución de color.

espacial del color de una imagen y es uno de los descriptores más rápidos y precisos del estándar MPEG-7.

El CLD está compuesto por 3 vectores, cada uno correspondiente a los canales Y (luminancia), Cb (crominancia azul) y Cr (crominancia roja) de la imagen. Los pasos para la obtención del mismo son los siguientes:

- Generación de imagen de 8x8 bloques: La imagen a procesar es dividida en 64 bloques de igual tamaño, para obtener una matriz de 8x8 bloques.
- Detección del color más representativo: De cada bloque se calcula el promedio con los píxeles pertenecientes a éste. Con estos valores se arma una nueva imagen de 8x8 píxeles.
- Cambio de espacio de color a YCbCr: Se realiza una transformación del espacio de color de RGB a YCbCr.
- Transformada Discreta del Coseno (DCT): La luminancia Y, la crominancia roja Cr y la crominancia azul Cb son transformadas en una matriz de 8x8 mediante la transformada Discreta del Coseno (DCT).
- Cuantización de coeficientes: Se realiza un reordenamiento y cuantización de los coeficientes de las matrices resultantes del paso anterior.

Una de las etapas de la extracción del descriptor de la distribución del color es la transformación de color de RGB a YCbCr. Este último espacio de color mencionado es el recomendado por el estándar MPEG-7 y hace referencia a la luminancia y crominancia (diferencia de azul y rojo respectivamente) de una imagen. Trabajos como [10], [11] efectúan una implementación de esta conversión de espacios en VHDL.

La transformación de RGB a YCbCr está dada por las siguientes relaciones:

$$Y = 0.299 * R + 0.587 * G + 0.114 * B \tag{1}$$

$$Cb = -0.169 * R – 0.331 * G + 0.500 * B \tag{2}$$

$$Cr = 0.500 * R – 0.419 * G – 0.081 * B \tag{3}$$

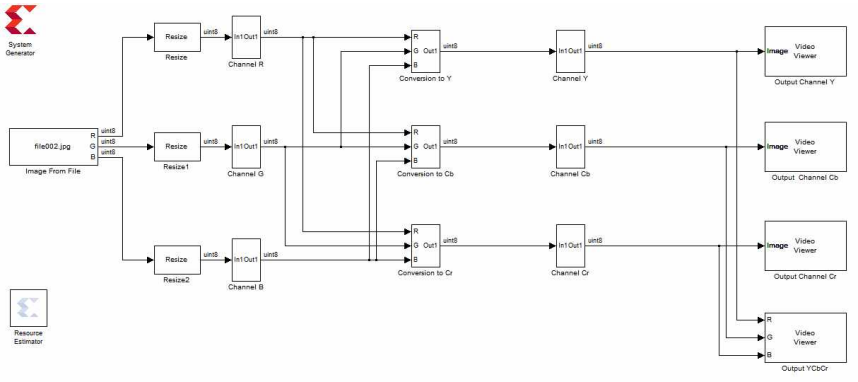

Fig. 2. Esquema en Simulink del bloque de conversión de espacio de color RGB a YCbCr.

3 Uso de plataformas paralelas para la conversión del espacio de colores

En esta sección se presentan los diseño paralelos de la etapa de conversión del espacio RGB a YCbCr utilizando System Generator y Vivado HLS. Para disminuir los recursos de hardware en la implementación, las ecuaciones presentadas en la seccion anterior se redefinen en punto fijo de la siguiente manera:

$$Y = 77 * R + 150 * G + 29 * B \tag{4}$$

$$Cb = -43 * R–85 * G + 128 * B + 32767 \tag{5}$$

$$Cr = 128 * R–107 * G–21 * B + 32767 \tag{6}$$

3.1 Implementación en System Generator

Es esquema general desarrollado en Simulink puede verse en la Fig. 2. Está compuesto por un bloque encargado de la lectura de la imagen, de una etapa de pre-procesamiento, de las etapas encargadas de la conversión, una etapa de recuperación de la imagen y, finalmente de la muestra en pantalla de los resultados obtenidos (tanto en canales separados como unidos).

El pre-procesamiento de la imagen en Simulink se efectúa mediante el esquema de la Fig. 3. Esta etapa está compuesta por los siguientes bloques: (a)Bloque Convert 2-D to 1-D: Convierte la imagen en un arreglo de 1D; y (b)Bloques Frame conversion and unbuffer: Especifica que el modo de muestreo de la señal de salida del bloque sea *frame-based*[1] y convierte la matriz de entrada en un arreglo de escalares.

La imagen de entrada es cargada al sistema mediante el bloque Image from file. La misma tiene un tamaño de 8x8 píxeles y se encuentra en RGB. Este

[1] http://www.mathworks.com/help/dsp/ref/frameconversion.html

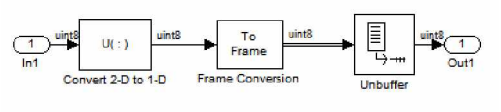

Fig. 3. Bloques que efectúan el pre-procesamiento de la imagen.

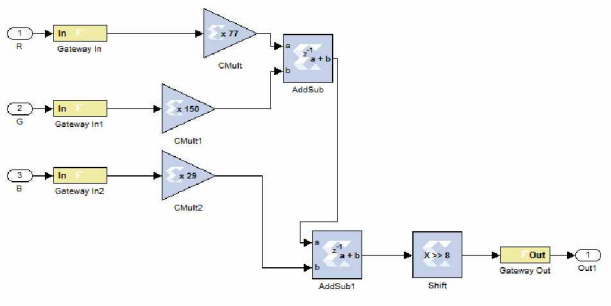

Fig. 4. Esquema en Simulink de la generación del canal Y.

tamaño de la imagen se debe a que en esta etapa de la extracción del descriptor, la misma fue reducida en los pasos previos a la conversión de espacio de color. El funcionamiento del sistema presentado en este trabajo, también fue verificado para una imagen de entrada de 512x512 píxeles en RGB.

Para obtener el valor de luminancia Y, la Fig. 4 muestra como la ecuación (4) es traducida a los bloques de System Generator. Se hizo uso de:

- Gateway In: Este bloque es la entrada a la parte del diseño a ser implementada en System Generator del diseño total. Para este caso, se emplean 3 Gateway In, los cuales se corresponden a los 3 canales de la imagen (R, G, B).
- Cmult: Efectúa las multiplicaciones entre la señal de entrada y el factor correspondiente. En la configuración de este bloque se carga el valor del multiplicador en punto fijo. Este bloque implementa un operador de ganancia, donde la salida es el resultado de la multiplicación entre el valor constante y la señal de entrada.
- AddSub: Realiza la suma o resta entre 2 señales de entrada.
- Shift: Realiza el corrimiento de 8 bits hacia la derecha.
- Gateway Out: Este bloque es la salida de la parte del diseño en system Generator hacia el resto del sistema realizado en Simulink. Se utiliza un solo bloque debido a que solamente se genera la salida correspondiente al canal Y. Como se puede observar que, para la generación de los valores para la luminancia, están implicados los 3 canales de la señal de entrada.

La misma estructura utilizada para la generación del canal de luminancia se emplea para generar los canales de crominancia, con la diferencia que se le

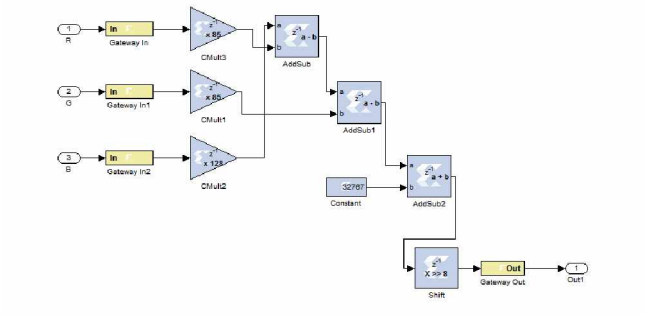

Fig. 5. Esquema en Simulink de la generación del canal Cb.

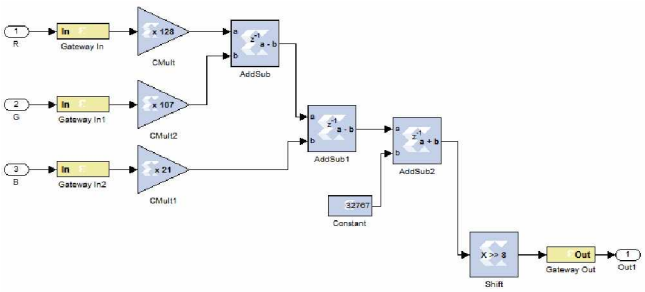

Fig. 6. Esquema en Simulink de la generación del canal Cr.

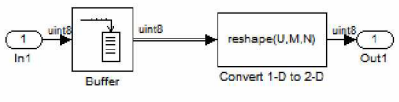

Fig. 7. Bloques que efectúan la recuperación de la imagen.

incorpora un bloque Constant, debido a que, como se puede observar en las ecuaciones (5) y (6) se suma un valor de 32767.

Los bloques de Simulink encargados de efectuar la recuperación de la imagen a través del arreglo obtenido del procesamiento realizado por los bloques de System Generator son los siguientes:

- Bloque Data type conversion: convierte al formato entero sin signo.
- Bloque Buffer : Convierte las muestras escalares para enmarcar la salida a menor velocidad de muestreo.
- Bloque Convert 1D to 2D: Convierte el arreglo de 1D a la matriz de la imagen correspondiente.

La Fig. 7 muestra la conexión de los bloques mencionados anteriormente. La visualiación de las imágenes obtenidas para cada canal, así como la imagen

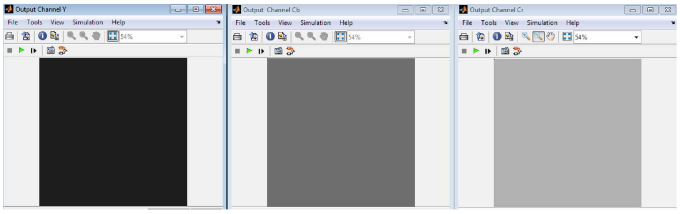

Fig. 8. Salidas: Canal Y, canal Cb y canal Cr respectivamente.

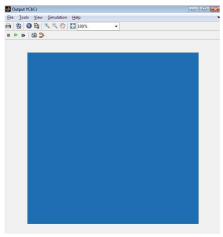

Fig. 9. Salidas: Canal Y, canal Cb y canal Cr respectivamente.

compuesta por los 3 canales se efectúa mediante el bloque Video Viewer. El resultado de la conversión de espacio de color obtenido mediante System Generator se puede observar en la Fig. 8. En la Fig. 9 se presenta la imagen resultante de combinar los 3 canales (Y, Cb y Cr).

3.2 Implementación en Vivado HLS

La implementación en Vivado HLS de la conversión de color se efectúa mediante la codificación en lenguaje C de las ecuaciones en punto fijo (4), (5), (6) mencionadas anteriormente.

La función que efectúa la transformación está compuesta por un bucle, el cual va leyendo un arreglo y efectuando las operaciones necesarias para la conversión. Al estar en punto fijo antes de salir del bucle, se efectúa un shift hacia la derecha sobre el resultado de cada operación, para obtener el resultado definitivo.

Como HLS permite definir directivas para mejorar la performance del diseño, se utilizó la directiva de pipeline para el bucle. En lo que respecta a las directivas relacionados a los recursos a utilizar en la FPGA, se optó por hacer uso de aquellas que permiten seleccionar que las operaciones aritméticas se efectúen en los DSPs disponibles. Sin estas últimas directivas, el hardware generado no hace uso de los DSP e implementa las funciones con lógica combinacional. En la Fig. 10 se muestra el flujo de la función que implementa la conversión del descriptor.

4 Resultados

La plataforma elegida para la implementación fue ZYNQ-7 ZC702 Evaluation Board (xc7z020clg484-1). La misma dispone de 280 BRAM 18K, 220 DSP,

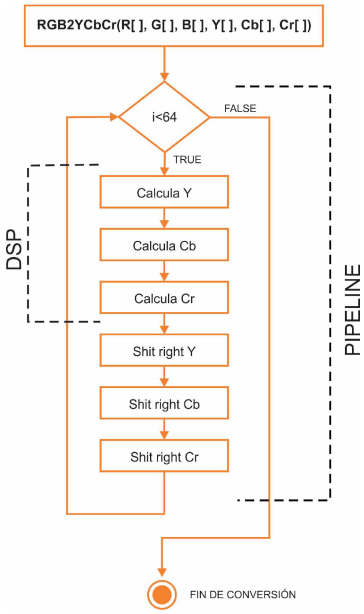

Fig. 10. Salidas: Canal Y, canal Cb y canal Cr respectivamente.

106400 DSP y 53200 LUT. En la tabla 1 se muestra la comparación de recursos entre ambos diseños. La columna Sys Gen muestra los recursos utilizados por System Generator. HLS_nd y HLS presentan los recursos requeridos al realizar la implementación de la conversión del espacio de colores mediante Vivado HLS. El primer caso corresponde a los recursos utilizados sin directivas y el segundo usando las directivas de pipeline y DSP.

Recurso	Sys Gen	HLS_nd	HLS
SLICE	51	109	55
LUT	118	275	105
FF	6	157	126
DSP	8	0	8
BRAM	0	0	0

Table 1. Estimación de recursos.

Considerando las columas Sys_Gen y HLS se puede observar, a diferencia de la cantidad de flip-flops, que existe poca variabilidad entre el resto de los recursos. Sin embargo, al no ustilizar directivas en HLS, la distribución de los recursos cambia notablemente. El tiempo que le insume a la FPGA obtener el resultado, cuando se procesa una imagen de 8x8 píxeles RGB, cuyo diseño se

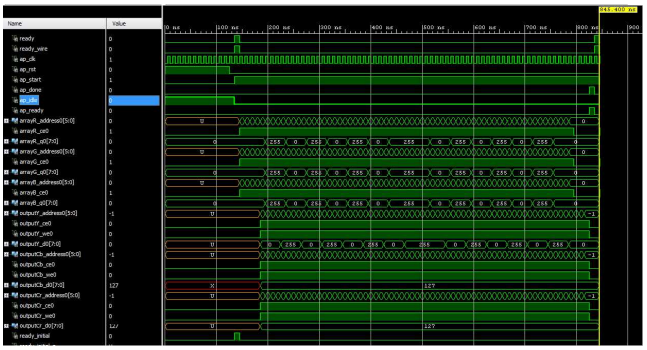

Fig. 11. Resultado de simulación para el diseño con HLS.

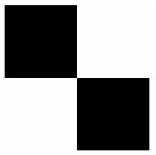

Fig. 12. Imagen de entrada para la medición de tiempo de procesamiento del bloque desarrollado en HLS.

efectuó mediante HLS es de 845 ns, mientras que al utilizar System Generator el tiempo es de 640 ns. El mismo se puede observar en la Fig 11, donde el marcador amarillo indica el final de procesamiento. Para esta medición de tiempo, se utilizó la imagen de entrada que se muestra en la Fig. 12.

5 Conclusiones

MPEG-7 es un estándar que permite la representación de contenido multimedia mediante la generación de descriptores. Adicionalmente, su papel en el procesamiento digital de señales en 2D resulta fundamental incluso en aplicaciones específicas de bio-medicina, control de tráfico, telecomunicaciones, entre otras.

En este trabajo, se presentó el diseño e implementación del descriptor denominado Color Layout de MPEG-7 utilizando plataformas paralelas FPGA basadas en SoCs. Tanto las FPGA como los SoC se perfilan como las alternativas de buen rendimiento para la extracción de los descriptores de MPEG-7 en tiempo real, debido a que las diversas extracciones pueden realizarse en forma paralela con el adicional de que cada operación de extracción es altamente paralelizable.

Se utilizaron y compararon en términos de utilización de recursos las herramientas System Generator y Vivado HLS, el cual hace uso de C/C++. Ambas se perfilan como buenas herramientas al momento de diseñar hardware a alto nivel. En el caso particular de cambio de espacio de color, la diferencia de los recursos utilizados no difieren en grandes valores, quedando a una elección subjetiva el uso de una herramienta u otra.

References

1. Yesun Joung, Kyuheon Kim, Kyeongok Kang, *The development of MPEG-7 interface over MPEG-4*, Consumer Electronics, pp.276-277 , 2003.
2. Myungseok Ki, Kyuheon Kim, *MPEG-7 over MPEG-4 systems decoder for using metadata*, Consumer Electronics. ICCE, pp. 245 - 246 , 2006.
3. I. Sezan, P. van Beek, *MPEG-7 standard and its expected role in development of new information appliances*, Consumer Electronics. ICCE. pp. 274 - 275, 2000.
4. Yong Rui, Huang, T.S., Shih-Fu Chang, *Digital image/video library and MPEG-7: standardization and research issues*, Acoustics, Speech and Signal Processing, Vol. 6, pp. 3785 - 3788, 1998.
5. V. V. Vinod, A. Lindsay , *MPEG-7: its impact on research, industry, and the consumer*, Multimedia Computing and Systems, pp. 7-11, 1999.
6. N.H. Eltonsy, G.D. Tourassi, A. Fadeev, A.S.Elmaghraby, *Significance of MPEG-7 Textural Features for Improved Mass Detection in Mammography*, Engineering in Medicine and Biology Society, pp.4779-4782, 2001.
7. N.H. Eltonsy, G.D. Tourassi, A. Fadeev, A.S.Elmaghraby, *Chaotic Synchronization of MPEG-7 Descriptors for Interpretation in Surveillance Video*, Circuits and Systems for Video Technology, Vol. 11, pp.688 - 695, 2001.
8. Zhenni Li, Jingjiao Li, Liang Li, Yue Zhao, Chaoqun Rong, *A SoC design and implementation of dynamic image edge detection based on the LEON3 open source processor*, Natural Computation, pp. 1263 - 1267, 2013
9. M. Borda,B. Belean, R. Terebes, R. Malutan, *FPGA based SoC for automated cDNA microarray image processing*, E-Health and Bioengineering Conference, pp. 1 - 4, 2011
10. Ahirwal, B., Khadtare, M., Mehta, R., *FPGA based system for color space transformation RGB to YIQ and YCbCr*, pp. 1345 - 1349, 2007.
11. Lamjed Touil, Abdessalem Ben Abdelali, Mtibaa Abdellatif, Elbey Bourennane, *R'G'B' to Y'CbCr color space conversion Using FPGA*, Wireless, Mobile and Multimedia Networks, pp. 255 - 258, 2008.
12. Behnam, B., Mansouryar, M., *Modeling and simulation of a DC motor control system with digital PID controller and encoder in FPGA using Xilinx system generator*, Instrumentation Control and Automation, pp. 104-108, 2011.
13. Thangavelu, A., Varghese, M.V., Vaidyan, M.V, *Novel FPGA based controller design platform for DC-DC buck converter using HDL Co-simulator and Xilinx System Generator*, Industrial Electronics and Applications, pp. 270 - 274, 2012.
14. Vidal, M, Cruces, R. ; Zurita, G., *Digital FIR filter design for diagnosing problems in gears and bearings using Xilinx's system generator*, 2014.
15. Vidal, M, Cruces, R. ; Zurita, G., *Implementation and simulation of IIR digital filters in FPGA using MatLab system generator*, Colombian Workshop on Circuits and Systems (CWCAS), pp. 1-5, 2014.
16. Alareqi Mohammed, Elgouri Rachid, Hlou Laamari, *High Level FPGA Modeling for Image Processing Algorithms Using Xilinx System Generator*, Journal of Computer Science and Telecommunications, Vol. 5, Issue 6, pp. 1-8, 2014.
17. Neha. P. Raut, Prof.A.V.Gokhale, *FPGA Implementation for Image Processing Algorithms Using Xilinx System Generator*, IOSR Journal of VLSI and Signal Processing, Vol. 2, Issue 4, pp. 26-36, 2013.
18. Aniket A. Ingle, Vrushali G. Raut, *Hardware software co-simulation of edge detection for image processing system using delay block in XSG*, Journal of Research in Engineering and Technology, Vol. 3, Issue 5, pp. 549-553, 2014.