# Compresión Run Lenght con FPGA aplicada a imágenes de información geográfica en formatos raster y vector.

Santos Martín López Estrada, René A. Cumplido Parra, Claudia Feregrino Uribe

Instituto Nacional de Astrofísica Óptica y Electrónica. Luis Enrique Erro No 1, Tonantzintla, Puebla, México. e-mail: santosle@ccc.inaoep.mx

Resumen.El presente artículo describe los logros alcanzados para la implementación en un FPGA del algoritmo de compresión Run Lenght aplicado a imágenes de sistemas de información geográfica en formato raster y vector. Se describen los módulos de compresión y descompresión, así como algunos resultados obtenidos en la implementación sobre el FPGA.

Palabras clave: Run Lenght, raster, vector, GIS, FPGA

## Introducción

En múltiples aplicaciones se requiere de información geográfica en formato digital[1], esta información está representada por imágenes vectorizadas o de tipo raster, las cuales son utilizadas por diversos equipos para su procesamiento.

Una imagen de tipo raster[2] es una matriz de datos en donde el color de cada píxel se representa en un byte, generando con ello archivos grandes y repetitivos, como se observa en la figura 1a), mientras que una imagen vectorizada se representa por un conjunto de puntos, los cuales pueden representar coordenadas, polígonos, nodos, arcos y líneas o una combinación de todos, como se observa en la figura 1b).

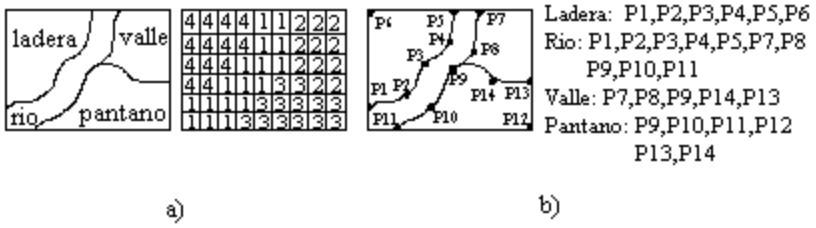


**Fig** 1 a) representación en formato raster; b) representación en formato vectorizado.

Generalmente en aplicaciones donde se requiere procesar este tipo de información se utilizan equipos conectados en red o se tienen dispositivos que la obtienen y envian a otros equipos para ser procesadas, en algunos casos se utilizan medios de comunicación de poco ancho de banda, por lo que para su transmisión es necesario realizar compresión. Un algoritmo que puede ser utilizado en la compresión de imágenes es RLE, Run Lenght Encode (Codificación del tamaño durante la

ejecución)[3], con el que se comprime la imagen codificando la cantidad de veces que se repite un mismo color o una misma coordenada. De esta forma, si en la imagen original se encuentran 20 valores iguales y consecutivos, en el archivo comprimido aparecerá un byte indicando el número de veces que se repite y después el valor en cuestión. Es decir, en lugar de almacenarse en el archivo 20 veces el valor correspondiente, se almacenan sólo 2 bytes: el 1º indica el número de repeticiones y el 2º el valor a repetir.

Los FPGA's y dispositivos programables proporcionan los medios adecuados para este tipo de problemática, ya que son dispositivos en los que se puede desarrollar poder de computo, aprovechando el paralelismo de los algoritmos sin tener que utilizan una computadora personal.

## Estado del arte

En la compresión de imágenes se utilizan diferentes tipos de compresión, con pérdidas o sin pérdidas, en el segundo cado uno de los algoritmos más usados por su sencillez y factores de compresión alcanzados es RunLenght. Apostolopoulous [4] menciona que para la compresión de imágenes y video en tiempo real se requiere un factor de compresión de 70, para lo cual utiliza una codificación de la imagen con la transformada discreta de coseno, en donde sus coeficientes utilizan una codificación de tipo RunLenght, ya que este método de compresión no representa perdidas. Sin embargo Burg [5] señala que la compresión utilizando Run Lenght no siempre es eficiente, ya que depende de las propiedades del flujo de datos, entre más repeticiones se tengan la compresión RLE será más eficiente, por lo que en aplicaciones con imágenes o compresión de video resulta de gran utilidad. En nuestro caso las imágenes de tipo raster.

Jiri Komzak [6] menciona que debido a la naturaleza de los datos en los sistemas GIS, tanto en formato vector como raster es adecuada la técnica de compresión RunLenght y LempelZiv, ya que permiten variaciones de acuerdo al tipo de datos y pueden ser extendidos a múltiples dimensiones.

E. Peña [7] en su articulo menciona que la ITU (Unión Internacional de telecomunicaciones) en la recomendación H.263 señala que para la transmisión de imágenes de calidad y video en tiempo real sobre canales de comunicación de banda angosta, se recomienda utilizar codecs con formato YCbCr, transformada DCT con coeficientes codificados con Run Lenght y compensación de movimiento.

En general en aplicaciones con compresión de imágenes se recomienda utilizar el algoritmo RunLenght o LempelZiv, sin embargo, éste último requiere de un mayor poder de computo, siendo RunLenght el más adecuado para aplicaciones en las que no se justifica la utilización de una computadora, pero sí un dispositivo programable. En este trabajo se toman estas recomendaciones y se implementa en un FPGA el algoritmo RunLenght, de tal manera que se tenga un sistema de compresión descompresión que puede ser utilizado como interface entre equipos que utilizan información de tipo GIS.

## Metodología

En la compresión RLE se deben distinguir los bytes que indiquen que se ha producido la compresión de los que no, esto se puede hacer activando los dos bits de más peso del primer byte (que es el que a la vez hace de contador) y escribiendo a continuación el byte del valor que se repite en la imagen original. Sin embargo se debe generar como comprimido cualquier valor que sea superior a 192 (6 bits menos significativos), para ello se añade un contador de 1. Es decir, un dato de valor mayor a 192 se almacenará como un RunByte de 1 y luego el valor (2 bytes para 1 sólo píxel), como se observa en la figura.

Como se menciono anteriormente los sistemas de información geográfica, manejan archivos de información tipo vector y raster, sin embargo, el formato vectorizado está formado por coordenadas de cada punto geográfico, por lo que al aplicar compresión (Run Lenght) se puede tener un archivo de mayor tamaño que el original, debido a la codificación del Byte Run, por lo que los archivos tipo raster son los más adecuados para comprimir.

Para la implementación del algoritmo en el FPGA se utilizó VHDL y se desarrolló la arquitectura para el módulo compresor mostrada en la figura 2.

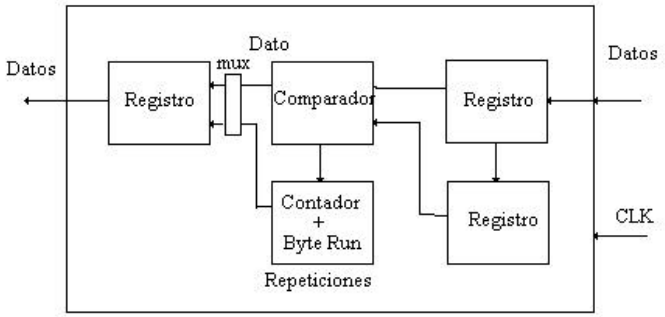


**Fig**. 2. Módulo compresor

Básicamente se tiene el dato de entrada y la señal de reloj, cuando se tengan dos datos similares se incrementará un contador (nivel de compresión), cuando los datos sean diferentes, se enviará a la salida un byte compuesto por los bits del Byte Run (bit 7 y 6) y el valor del contador, seguido del dato que se repite.

El módulo descompresor debe almacenar el dato, detectar si se trata de un dato comprimido o un dato sin comprimir, es decir debe detectar el ByteRun, en caso de ser un dato comprimido, debe enviar a la salida el siguiente dato un número determinado de veces (contador con los 6 bits menos significativos del BytRun). En la figura 3 se muestra un diagrama simplificado de esta parte.

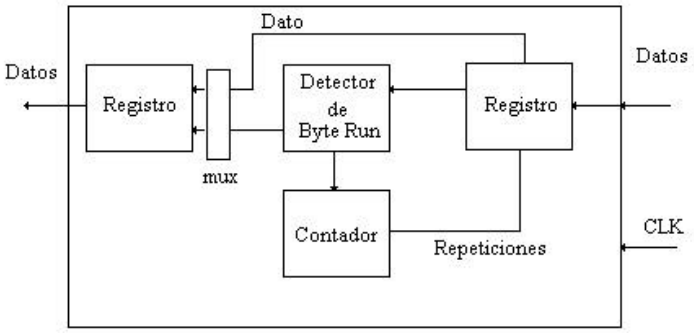


**Fig**. 3. Módulo descompresor

Ambos módulos se desarrollaron en un solo Bloque, controlado por una señal de entrada (Modo) y generando la salida junto con la señal válido para indicar en qué momento la salida es válida, como se muestra en la figura 4.

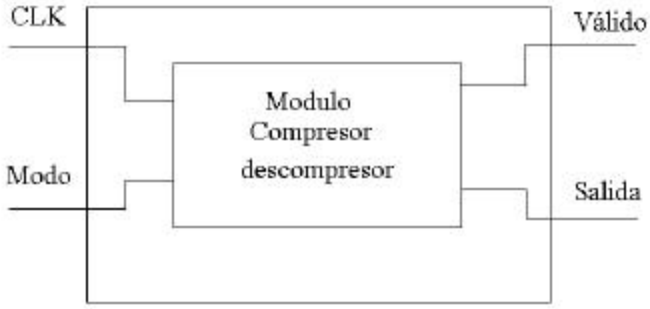


**Fig**. 4. Estructura general del compresor / descompresor.

La arquitectura propuesta se sintetizó en un FPGA XILINX2e, optimizado por velocidad, para realizar compresión y descompresión al vuelo, lo que nos arrojo la utilización de los siguientes recursos:

| | | |
|---|---|---|
| Número de Slices | 1514 de 3072 | 49% |
| Número de Flip-Flop | 952 de 6144 | 15% |
| Número de LUT de 4 entradas | 2930 de 6144 | 47% |
| Número de IOB | 10 de 182 | 5% |
| Número de GCLK | 1 de 4 | 25% |

Al ser optimizado en velocidad el área del FPGA creció, utilizando el 49%, pero se ganó en velocidad, con una frecuencia máxima de operación de 48 MHz determinado por el “critical path”, el cual está definido en el modulo descompresor, teniendo 9 niveles de lógica, sin embargo el 76% del tiempo se consume en rutoe, al utilizar una memoria interna no distribuida, por lo que este tiempo se puede mejorar.

## Resultados preliminares

Se realizaron simulaciones de alto nivel del algoritmo, logrando razones de compresión de 4 a 1 en imágenes de tipo mapa de bits, posteriormente se realizaron algunas pruebas utilizando simulación post-síntesis, ambos experimentos arrojaron los siguientes resultados:

A continuación se muestra un fragmento de los datos de una imagen sin comprimir, los datos están en hexadecimal y separando los valores por dos puntos (:).

42:4d:36:b4:4:**0:0:0:0:0**:36:4:**0:0**:28:**0:0:0**:80:2:**0:0**:**e0**:1:**0:0**:1:0:8:**0:0:0:0:0:0**:b0:4:0:
**c4**:e:**0:0**:**c4**:e:**0:0:0:0:0:0:0:0:0:0:0:0:0:0:0:0**:80:**0:0**:80:**0:0:0**:**80:80**:0:80:**0:0:0**:80:0:8
0:0:**80:80**:**0:0**:**c0:c0:c0**:0:**c0**:**dc**:**c0**:0:**f0**:**ca**:a6:**0:0**:20:40:0:0:20:60:0:0:20:80:0:0:20:a0:
0:0:20:c0:0:0:20:e0:0:0:40:0:0:0:40:20:0:0:40:40:0:0:40:60:0:0:40:80:0:0:40:a0:0:0:40:
c0:0:0:

Fragmento de la misma imagen después del algoritmo de compresión.

42:4d:36:b4:4:**c5:0**:36:4:**c2:0**:28:**c3:0**:80:2:**c2:0**:**c1:e0**:1:**c2:0**:1:0:8:**c6:0**:b0:4:0:**c1:c4**:e:**c2:0**:**c1:**
**c4**:e:**d0:0**:80:**c2:0**:80:**c3:0**:**c2:80**:0:80:**c3:0**:80:0:80:0:**c2:80**:**c2:0**:**c3:c0**:0:**c1:c0**:**c1:dc**:**c1:c0**:0:**c1**
**:f0**:**c1:ca**:a6:**c2:0**:20:40:c2:0:20:60:c2:0:20:80:c2:0:20:a0:c2:0:20:c1:c0:c2:0:20:c1:e0:c2:0:40:c
3:0:40:20:c2:0:c2:40:c2:0:40:60:c2:0:40:80:c2:0:40:a0:c2:0:40:c1:c0:c2:0:

En negritas y subrayado, se encuentran en ambos fragmentos del archivo, indicados los bytes sobre los que se realizó la compresión, en el primer fragmento se observan 5 ceros, los cuales se traducen a **C5** en el Byte Run (192 + 5) seguido del valor que se repite (0), de la misma manera se marcan algunos otros casos como el caso del quinto valor subrayado en el primer fragmento (**e1**), como es un valor mayor a 192, se debe codificar en 2 bytes, el Byte Run (192 + 1) y el valor **e1**. El archivo comprimido se obtuvo con un tamaño de 67KB logrando una compresión de 4:1.

Las pruebas de simulación post-síntesis mostraron resultados similares al comprimir imágenes a una razón de 3 a 1 y de 4 a 1 en imágenes de tipos raster, mientras que en imágenes de tipo vectorizadas se lograron compresiones de 2 a 1, y en algunos casos resulto más grande el archivo comprimido que el original como se esperaba en este tipo de formato, la figura 5 muestra un fragmento de esta simulación.

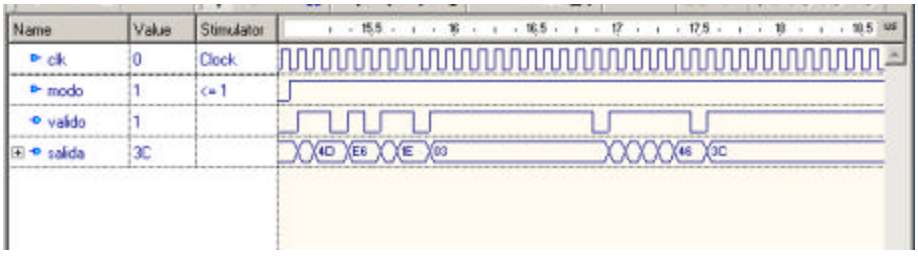


**Fig**. 5. Simulación post-síntesis.

La figura 5 muestra un fragmento de la descompresión de una imagen comprimida almacenada en memoria, es importante notar la señal valido para saber en que momento es válida la salida, ya que se pretende que el compresor descompresor trabaje con imágenes al vuelo.

Las razones de compresión obtenidas, se pueden considerar como aceptables, ya que se trata de un algoritmo de compresión muy sencillo pero con una velocidad de compresión que nos permite trabajar con imágenes al vuelo. Algunos algoritmos similares como LZ son más eficientes, logrando razones de compresión de 6 a 1 pero son más lentos, lo cual nos resta velocidad.

## Trabajo futuro.

De acuerdo a las pruebas obtenidas en la simulación post-síntesis el trabajo a desarrollar consiste en añadir las interfaces adecuadas para transmisión y recepción de imágenes, eliminando parte de la memoria simulada en el FPGA, y reduciendo de esta manera el tiempo del "critical path", permitiendo trabajar con velocidades mayores a 100Mhz.

## Conclusiones

Se presentó una variante del algoritmo de compresión Run Length, el cual fue aplicado con éxito a imágenes de tipo raster, logrando razones de compresión de 4 a 1, sin embargo en el caso de imágenes de tipo raster se lograron razones de compresión de 2 a 1 y en algunos casos se generaron archivos dos veces mayores al original lo cual es un resultado esperado con este tipo de algoritmos.

Al implementar este algoritmo en un FPGA podemos utilizar sistemas de bajo costo y alto desempeño para acoplar a dispositivos que generan este tipo de información y poderla transmitir los archivos por canales de comunicación de banda angosta.

## Referencias


[1] Ronald Briggs "GIS Data Structures" UTDallas. 2002

[2] V. Cubas F. "Formatos Gráficos". Programación Actual. Noviembre 1997.

[3] L. S. Smith "Data Compresión". Universidad de Stirling. 2002

[4] John G. Apostolopolous "Image and Video Compression". Streaming Media Systems Group. Hewlette Packard Laboratories. Abril 2002.

[5] A. Burg. T. Boesch. D. Perels. "Video Compression" Integrated Systems Laboratory ETH Zurich. Verano 2001.

[6] Jiri Komzak, Pavel Slavik "Architecture of System for Configurable Gis Data Compression" Zcech Technical University. 2002.

[7] L.E. Peña, J. E. Preciado "Códec de Video en Tiempo Real Para Comunicación a Baja Velocidad Binaria". Pontificia Universidad Javeriana Colombia. 2000.