JSST
Journal of Space Science & Technology

Vol. 7/ No. 1/ Spring 2014
pp. 57-66

FPGA Implementation of JPEG and JPEG2000-Based Dynamic Partial Reconfiguration on SOC for Remote Sensing Satellite On-Board Processing

A. Chekini[1] and H. R. Naji[2*]

1. Department of Electrical and Computer Engineering, Shahid Beheshti University

2. College of Electrical and Computer Engineering, Graduate University of Advanced Technology

[*]Postal Code: 7631133131, Tehran, IRAN

hamidnaji@ieee.org

This paper presents the design procedure and implementation results of a proposed hardware which performs different satellite Image compressions using FPGA Xilinx board. First, the method is described and then VHDL code is written and synthesized by ISE software of Xilinx Company. The results show that it is easy and useful to design, develop and implement the hardware image compressor using new techniques of programmable logic tools for space applications. In this paper the proposed hardware uses the partial reconfiguration technique, and it is put on board of satellite. Appropriate bit streams are produced by synthesis tools; therefore, we have two bit streams which can be configured at any moment of time according to the user request. When users intend the hardware is reconfigured and changed from JPEG to JPEG2000 or vice versa. The Proposed architecture has some advantages other than previous architectures such as high-speed and real-time processing, high flexibility, low cost, high security and low power consumption. This idea can be utilized in modern commercial hardware space board for data compressing due to using partial reconfiguration technique.

Keywords: Compression, Satellite, On-Board Processing, Real-Time Processing, Reconfigurable

Nomenclature

W	wavelet
g	discrete transform
f	discrete transform
S	sum
C	carry
D	distortion
λ	threshold
A , B	adder inputs
i	code-block number
k	discrete transform component
max	maximum
z_i	truncation-point
n	order
JPEG	Joint Picture Expert Group
Gbps	Giga bit per second
DCT	Discrete Cos Transform
DSP	Digital Signal Processing
DWT	Discrete wavelet transform
PCRD	Post Compression Rate-Distortion
ROI	Region of Interest
SPR	Static Partial Reconfiguration
DPR	Dynamic Partial Reconfiguration

Introduction

Image compression is a key technology to realize on-line satellite image transmission economically and quickly. Among various image compression algorithms, JPEG family algorithms are the international standard for still color image compression. Various kinds of satellite images were compressed with JPEG algorithm, and then the relation between compression ratio and image quality were evaluated by Tada in [1]. Moreover, MICRO-SATELLITES are becoming more and more popular for the Earth observation. Their major advantages are

1. M. Sc.
2. Assistant Professor (Corresponding Author)

Recived: 2014.01.15, Accepted: 2014.05.31

their simplicity and cost-effectiveness [2]. They can capture images of the surface of the Earth with a resolution down to 200 m or less than it. As they orbit the Earth in approximately 100 min, they have the capacity to capture many images, and they can hold approximately200 of them in memory. However, they remain in contact with a single tracking station for only a few minutes, and downloading such images presents a bottleneck in the whole process of image acquisition. The obvious solution of this problem is the coding or compression of the images on the board. By regarding to Peixin's viewpoint "The cameras on board microsatellites are usually off the shelf sensors that do not have high fidelity that has to be preserved by all means"[3]. In addition, another bottleneck of downloading the images from such satellites is being time consuming. One of the standard approaches used for compression is the lossy version of joint picture expert group (JPEG) coding. The JPEG image compression standard was developed by JPEG and JBIG committees [4]. Commercial imaging satellites are experiencing rapid growth in resolution, and consequently, are generating data at increasingly high rates. Data generation of several Gbps is not uncommon, and on-board storage capabilities and downlink data rates have not kept pace. On-board image compression can solve this problem, by reducing the size of the data to be stored and down linked and providing flexibility in the downlink content. With regard to this instance, compressing the satellite images before sending from satellite on-board processor to Mission Control Center and Satellite Control Center is necessary. Also Compression allows efficient utilization of channel bandwidth and/or storage size [5].

JPEG2000 has many features which make it ideal for the compression of satellite imagery. These features include high-performance lossless compression, random access to the image content in compressed data, low resolution (thumbnail) versions of the image, and progressive output format. JPEG2000 is also an international standard [4], making the distribution of images to a diverse clientele more straightforward.

However, implementation of high-rate on-board JPEG2000 compression presents special challenges. JPEG2000 is a computationally intensive method, requiring several hundred operations to be performed on each pixel during the compression process. Color image JPEG compression consists of five steps as shown at Figure 1. The JPEG compression of gray scale images can be divided into three main steps that can be seen in Figure 2. In the JPEG compressor, the 2-D DCT is the most critical module to be hardware implemented because of its high complexity algorithm. [6] In the architectural level, this critical point is the responsibility of the sum operations on wide inputs and of the consequent delay generated by the carry propagation in the ripple carry architecture used in the adders. Therefore, our use of the architecture was divided into two 1-D DCT architectures and one transpose buffer. The two 1-D DCT architectures are similar but the bit widths at each pipeline stage are different. [7] This architecture is shown at Figure 3.

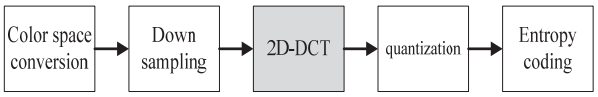

Fig. 1. JPEG compression steps of color images [18]

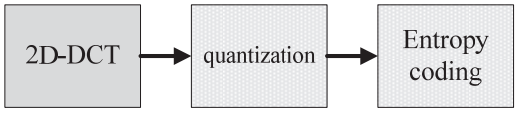

Fig. 2 JPEG Compression flow

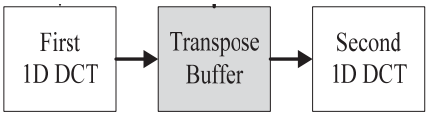

Fig.3 Generic 2-D DCT architecture [7]

Moreover, The new image compression standard was christened JPEG 2000 [8]. At the heart of JPEG 2000, there are two complex DSP algorithms: Discrete Wavelet Transform and Embedded Block Coding with Optimized Truncation [9]. Both of algorithms are compute-intensive and very difficult to optimize in software implementations, so hardware architectures for these units have been built [10].

Nowadays, JPEG2000, in particular, is used for all imaging devices that produce data of high spatial, spectral, radiometric and/or temporal resolution. Typical examples can be found in space borne remote sensing [11]. The actual compression of the proposed approach is based on the PEG2000 standard [12]. this is because it provides a framework for supporting multispectral imagery and uses an efficient wavelet compression scheme, which can utilize lossless and lossy compression seamlessly on an image. It can also divide the image into tiles and compress them independently. Principle of wavelet is based on delay which is calculated by Eq.1.

$$W_{j,k}(t)= 2^{(-j/2)}w(2^{-j}t-k) \quad (1)$$

In this paper we use Eqs.2 to implement discrete transform DWT .

$$g(k)= 2x(2k+1) – x(2k) – x(2k+2) \quad (2)$$
$$f(k) = x(2k) + g(k-1) + g(k)/8$$

A major feature of JPEG2000 is the adoption of the Embedded Block Coding with Optimized Truncation (EBCOT) to produce resolution- or quality-progressive bit streams. EBCOT consists of two

different stages. First, it divides the image into code-blocks of typically 32×32 or 64×64 pixels. In the second step it performs the so-called post compression rate-distortion optimization (PCRD-opt) that minimizes the total distortion of all code-blocks $D = \Sigma D(z_i)$ where z_i indicates the optimal choice of where to truncate the embedded bit stream of code-block i. The optimization is performed with respect to the user-defined compression rate $\Sigma_L(z_i) \leq L_{max}$, which has to be met. Details on EBCOT and its PCRD-opt can be found in [3]. However, it is worthwhile mentioning that PCRD-opt is processing the falling convex hull of the distortion-rate slopes (Figure.3) defined by Eq.3.

$$\frac{\Delta D}{\Delta L} = \frac{D_i^{(n-1)} - D_i^{(n)}}{L_i^{(n)} - L_i^{(n-1)}} = \lambda_i^{(n)} \quad (3)$$

This provides a measure of the distortion reduction ΔD computed by adding ΔL symbols of the code-block i in order to transmit its wavelet coefficients more accurately. For a given L_{max}, the PCRD-opt algorithm determines a minimal threshold λ for all code-blocks *i* to find the truncation-points z_i according to Eq.4.

$$\lambda_i^{Zi-1} < \lambda \leq \lambda_i^{zi} \quad (4)$$

If the distortion-rate slope is steeper, then (Eq.4) selects a later truncation point z_i and allocates more symbols for this code block. Thus the main idea for the implementation of ROIs in JPEG2000 is to boost the slope either by up-scaling of coefficients of ROI-pixels (increase ΔD) [12], or by scaling the slope by a ROI-weight $w_i > 1$ (increase ΔD/ΔL) [13]. However, the standard implementation of ROIs improves the quality of ROIs only relative to non-ROIs for a given compression rate. Furthermore, it is not possible to guarantee a certain quality-level for a particular code-block. This is not appropriate for many applications in remote sensing., but today one specific application of data compression is in remote sensing satellites, which operate in radiation environments that exacerbate soft error rates. Many remote-sensing applications avoid data compression of measurement data because the events of interest are associated with a radiation that can corrupt the final compressed measurement data. This paper will propose an architecture which JPEG and JPEG2000 standards both can handle in hardware on board of remote sensing satellites. This leaves very few options for space-qualified processing elements.

The Virtex series of FPGAs from Xilinx are one of the few devices capable of performing high-rate JPEG2000 compression in space environment. These devices have hundreds of on-board block RAM and multiply elements, millions of programmable gates, and are capable of system clock rates of several hundred MHz, giving them tremendous processing capability. They are also qualified for the space environment, have a total dose of ~200 kRad, and are latch-up immune.

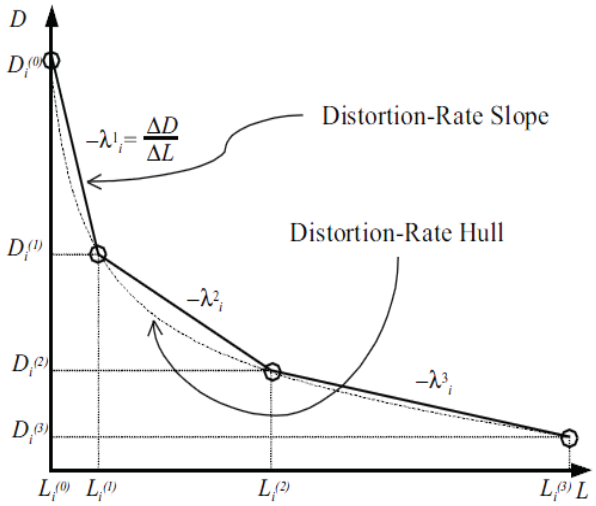

Fig.4 Simplified illustration of the distortion-rate slopes used by EBCOT. [11]

Partial Reconfiguration Technique

One of the biggest disadvantages of using hardware solutions instead of software systems is that hardware is less flexible as compared to software. On the other hand, their other important parameters such as reliability and speed are very attractive. One way for solving this problem is using "Dynamic Partial Reconfiguration". This technique enables some hardware with partial reconfiguration such as FPGA to modify their internal structure while executing, and to convert to a new structure with a different function. Thus it enables us to have hardware flexibility and reconfiguration while processing is going on. With these techniques, we can implement a hardware which was not implemented in the past due to hardware inflexibility [21] .

Concept and Various Methods of Partial Reconfiguration

Partial Reconfiguration is the process of configuring a part of FPGA, while the other parts are running. This feature is shown in the Figure5.

Different methods of partial reconfiguration for a hardware with reconfiguration ability include [14]:

Static Partial Reconfiguration (SPR)

In this type of configuration device is not active during reconfiguration process. The entire system is not running till the configuration on the hardware is implemented. Then the circuit once more starts running with new functionality. SPR configuration is the simplest and the most common way to implement applications on hardware with partial reconfiguration ability.

Dynamic Partial Reconfiguration (DPR)

In this method one part of the hardware is changed, while other hardware parts are running. H ere, an algorithm should be divided into several parts which do not have any timing overlap, and then the sequence and usage time of each part should be managed. In this case, the main problems are: the time required for reconfiguration, the design complexity, increase in the hardware costs, and the increase in design time. Despite these limitations, this method has the following advantages that support its usage: reduction of power consumption, increased utilized area, increased flexibility, and increased hardware reuse. In addition to the benefits, SPR configuration can make a compromise between the time and occupied area. In general, in the designing phase, the design should be divided into two parts. The first part remains without any changes while the program is being executed on the hardware. This part is called Static Region (SR). Static part must include the configuration controller and interface for contact with dynamic part. All inputs and outputs are connected to this section. The static part is connected to dynamic part modules by a fixed interface.

The dynamic part changes according to the control command when the algorithm is executing. Typically modules are loaded on the dynamic part from an external memory as bit stream with the appropriate control command.

Methods of Partial Reconfiguration Technique

Partial reconfiguration technique has some advantages that can help us for dynamic hardware configuration mainly when algorithm or data set changes, for example as we want to change or modify rule set in firewalls at run time. This section separately explains each of the various available methods for dynamic partial reconfiguration.

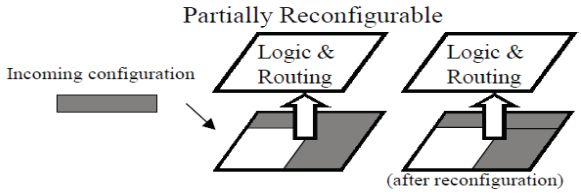

Fig. 5 Partial Reconfiguration Technique [14]

Module -Based Partial Reconfiguration

In this approach each module can be individually designed. We make different bit streams for every module that we want to place in static part and set each string on the external memory to be used in appropriate time. For being sure about having proper communication between static part and dynamic part, we use an interface called BUS MACRO. During design, we should notice this key point that all of our dynamic modules should have the same input and output with the same name in order to prevent troubles in contacting the stationary parts. Figure 6 shows this method. As you can see in this Figureure, we can use several PRM1[22].

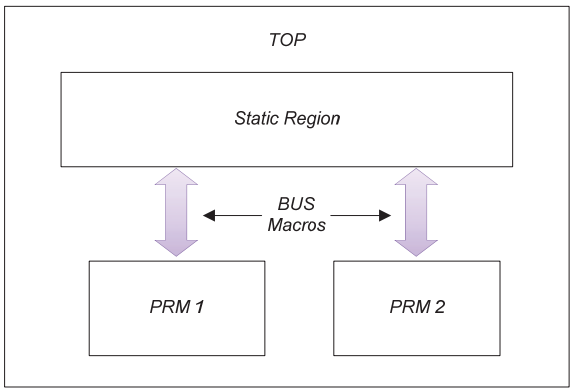

Fig.6 Module-Based Partial Reconfiguration

Systematic design, implementation of large projects and modular design are advantages of this method. Also high storage costs, high reconfiguration delay, and having no tri-state buffer reconfiguration are its disadvantages. In this method of design, we can use several dynamic parts as to be able to configure several dynamic modules in each dynamic region, as shown in Figure 7[14].

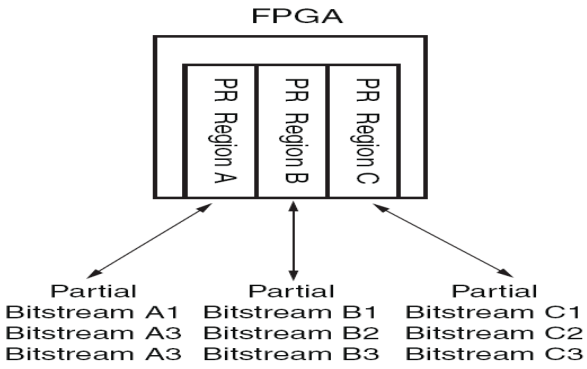

Fig.7 Using several PRM and PRR2 [14]

Difference-Based Partial Reconfiguration

This method is suitable in dynamic region, where a minor change is made on the first module compared with the next module. In this method, bit streams contain only information about the difference between the first and second structure. This method is highly suitable for very large projects that have similar functionality and dynamic parts. Using this method, the configuration time in real-time systems is reduced considerably. It is remarkable that both of the above

1. Partial Reconfiguration Module
2. Partial Reconfiguration Region

methods produce the same bit stream which is used for partial reconfiguration on FPGA [14, 15, 21].

Implementing Partial Reconfiguration Technique

Initially, there were very limited tools for implementing this technique, but recently, "PLAN Ahead" software of Xilinx has offered a graphical tool for implementing partial reconfiguration technique. For further information about the functionality of this software you can refer to [15]. Currently, partial reconfiguration is not supported in all FPGAs, but some FPGAs of Xilinx have this ability, for example: Spartan 3, Virtex II, Virtex II Pro and Virtex 4.

The Proposed Architecture

In this section, we introduce our architecture and then implement it on a real FPGA. Using the knowledge provided in Section 2 and understanding compressors behavior and performance in Section 1, a new architecture can be presented. We consider FPGA in hardware compressor on board of satellite is a real-time processor, this hardware can be used in the common part of the OBC[1] and the main processing board of a satellite (usually in remote sensing) as it is shown in Figure 8. Because our hardware is implemented on FPGA and has a low cost, low power, and low weight, it is very compatible with OBC of more satellites. Therefore, the proposed hardware for space image compressing is so desirable.

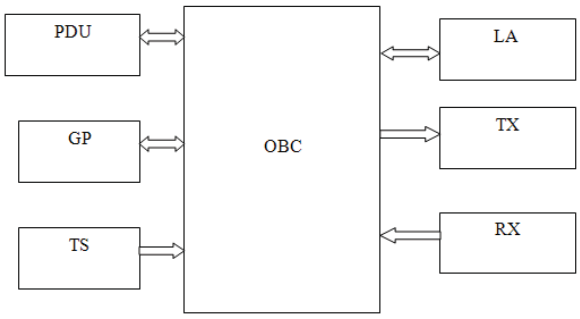

Fig. 8 OBC of satellite board by its relations

Figure 9(a) shows, one FPGA contains a general algorithm for data processing. In fact, this FPGA is the heart of real-time processing hardware system. A Memory Stick is presented with this FPGA. System administrator stands besides this hardware system. Now the question is what the difference between this architecture and the current architecture for OBCs is. In our architecture, using the partial reconfiguration technique, appropriate bit streams are produced by the administrator and placed only on the External Memory Stick. In this architecture, we have classified compressing algorithms in different groups to reduce the processing time. For example, here the system administrator can implement a new compressing algorithm on the Embedded RAMs of FPGA according to the desired target if the admin or user wants the satellite to send information to the earth. Remote controlling of the system is one of the obvious benefits of this architecture. In this hardware the admin (user) from the TT&C[2] Grand station can change the functionality of FPGA. Figure 9 (b) depicts the proposed hardware compressor board on OBC of Remote Sensing Satellites.

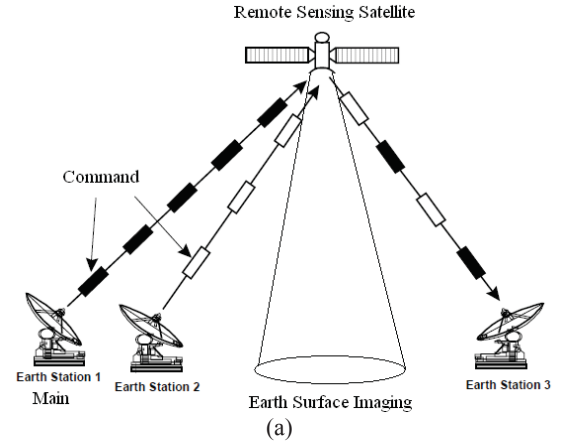

Fig. 9 a) sending command to satellite from Earth (user), b) architecture of proposed hardware on OBC of satellite

Suppose, we can classify compressing algorithms according to the type of user requirement. For using this architecture, the plan should be divided into two static and dynamic parts. Using partial reconfiguration technique, all fixed routines, like common elements in compressing algorithms are placed on the static part,

1. On board computer of satellite

2. Track, Telemetry and Command

but complete algorithms, in this paper JPEG and IPEG2000, are placed in the dynamic part of design. Thus, at any time, the system user can implement the appropriate compressing algorithms in FPGA using partial reconfiguration technique with an appropriate remote command from the Earth.

This architecture, can be configured at any moment of time according to the type of compressing needed. Major advantages of this architecture in comparison to other architectures are: Flexibility to changes of algorithms, low data processing time, and high speed which is because compressing algorithms are executed in hardware.

In this architecture, compressing algorithms are implemented on FPGA through partial reconfiguration technique by system user's command. Therefore, for the following major reasons the on-board data processing speed in this architecture increases significantly in comparison to the other existing architectures: Transferring the compressing algorithms from external Memory Stick (as it is in current architectures) to embedded RAMs of FPGA (in our architecture). Actually, with this idea, the delay time of data transfer from External Memory Stick has been circumvented, and FPGA accesses internal memory with higher speed, consequently, the data processing speed increases. In addition, by eliminating external memory for data processing, and because of the removal of communication wires, the reliability of system increases. This architecture is shown in Figure 10.

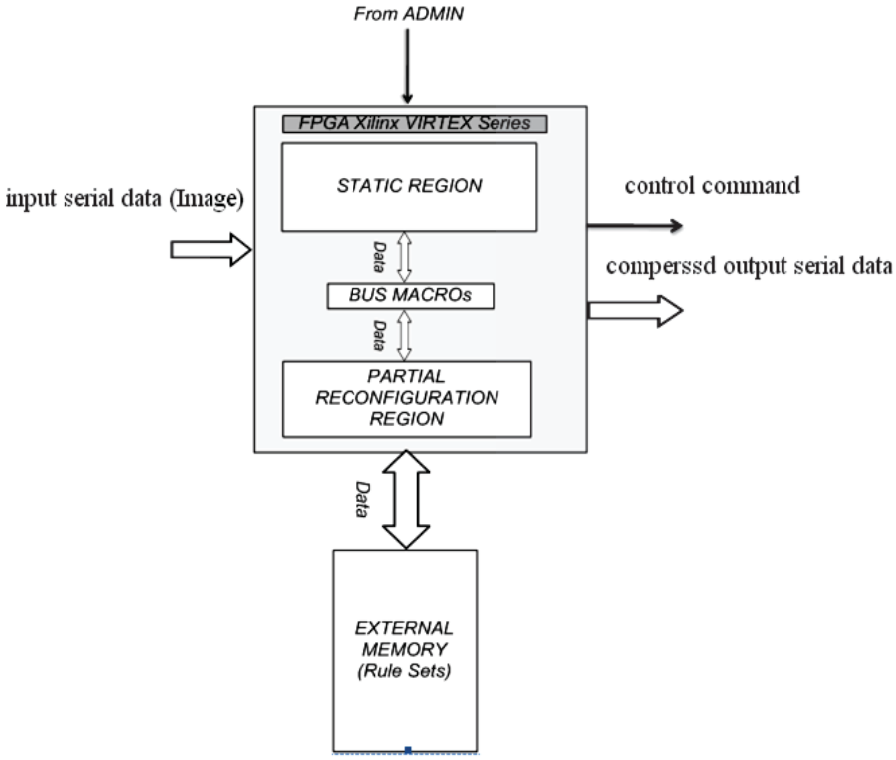

Fig.10 The proposed hardware for on-board Image compressing on satellite OBC

Now on-board compressing hardware for real-time image processing can be designed. The 1-D DCT architecture is proposed in reference [16]. Considering the 1-D DCT algorithm steps, the use of a pipelined architecture between these steps becomes natural. Since the algorithm has six steps, the pipeline will have six stages, where five perform additions/subtractions and one performs multiplications. The five adders in the original DCT architecture are ripple-carry and the multiplier is based on shift-add operations. This architecture uses a single arithmetic unit in each stage to perform all the necessary operations at that stage.

Then the unit inputs are connected by multiplexers to select which value must be used in each clock cycle. The multiplexers controls are generated following the algorithm order. Temporal barriers are obtained to allow the pipeline design with the use of ping-pong registers.

The control block generates the signals to control the pipeline fill-up and emptying through an external signal that indicates if the input values are valid values for the image. We know that the architecture of adders used in the original 1-D DCT is ripple carry [17].

This architecture of adders is quite simple and wide- spread. Its main advantage is the reduced occupied area.

However, there is the disadvantage of the low performance provoked by the slow carry propagation. This slow carry propagation is what defines the critical path of the architecture of 1-D DCT calculation and, as

a consequence, of the 2-D DCT architecture itself. Figure.11 presents the architecture in blocks of a ripple carry adder.

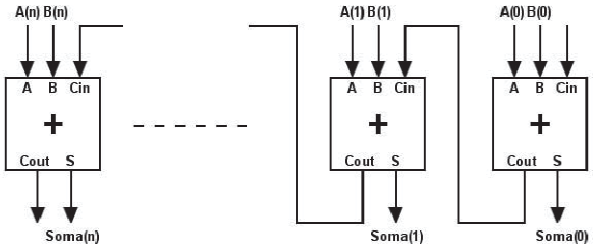

Fig.11 Block diagram of a ripple carry adder

The C_{i-1} being the stage carry in, A_i and B_i the adder inputs, C_i the stage carry out and S_i the resulting sum, we can write the equation that determines the sum as:

$$S_i = A_i \cdot B_i \cdot C_{i-1} + A_i \cdot \overline{B_i} \cdot \overline{C_{i-1}} + \overline{A_i} \cdot \overline{B_i} \cdot C_{i-1} + \overline{A_i} \cdot B_i \cdot \overline{C_{i-1}} \quad (.5)$$

This can be factored in the following way:

$$S_i = C_{i-1} \cdot \left(A_i \cdot B_i + \overline{A_i} \cdot \overline{B_i} \right) + \overline{C_{i-1}} \cdot \left(A_i \cdot \overline{B_i} + \overline{A_i} \cdot B_i \right)$$
$$= A_i \oplus B_i \oplus C_{i-1} \quad (6)$$

The equation that determines the carry can be written as:

$$C_i = A_i \cdot B_i + A_i \cdot C_{i-1} + B_i \cdot C_{i-1} \quad (7)$$

This, also, can be factored in the following way:

$$C_i = A_i \cdot B_i + C_{i-1} \cdot \left(A_i + B_i \right) \quad (8)$$

considering this fact that each stage needs the carry obtained in the previous stage to make the sum, the total time delay is linearly proportional to the n - *i.e.* length of the adder.

In [7] a small logic was developedinvolving the input B, the adder carry in and the operation control signal *AddSub*. With this logic the same arithmetic operator can operate in addition mode or in subtraction mode. These can be seen in the Table 1.

Table 1. Address operation mode control logic

Add Sub	Carry In	Input B	Operation
0	0	B	A+B
1	1	~B	A+(~B+1) or A-B

Besides, it was possible to determine the maxima values generated by each operator and, in the same way, to identify which adders would not generate, in any hypothesis, a significant carry out. This way, it was possible to use the minimum bit width in each stage and, in consequence, the minimum of resources.
Now by presenting the design of adder architecture for use in JPEG compression, and more specifically for use inthe 2-D DCT of the JPEG compressor, our hardware architecture can be implemented. If we compare this architecture with the architecture presented in Section1, we see the FPGA is configured partially, and the rule sets are implemented on the dynamic part of FPGA separately in different classes. We have used XILINX ISE tool for the simulation and implementation. Also EARLY ACCESS and PLANAHEAD tools are used to implement the partial reconfiguration technique. Section 4 provides the results of simulation and synthesis.

Hardware Implementation and Synthesis Results

In this section, we present the results of simulation and synthesis of our proposed architecture on FPGA. We have selected Xilinx virtex4 FPGA family for the simulation and implementation as they support partial reconfiguration technique. To serve this goal we first simulated functionality of JPEG and JPEG2000 in MATLAB/ simulink software. Fantastic high resolution images captured by the recently launched WorldView-1 satellite have been released by DigitalGlobe. This satellite was launched on September 18th , 2007 from Vandenberg Air Force Base. Therefore, we select several different images captured by this satellite being shown in Figure 12. After simulation of our proposed algorithm at MATLAB/ simulink we should write VHDL code for the hardware implementation. Subsequently, VHDL code was written and nether results were obtained.

Fig.12 several different images captured by worldview-1 satellite

Power Consumption

Using XPOWER Analyzer of ISE tools, we have calculated the amount of power consumption for this architecture using different rules. These results are presented in Figure 13.

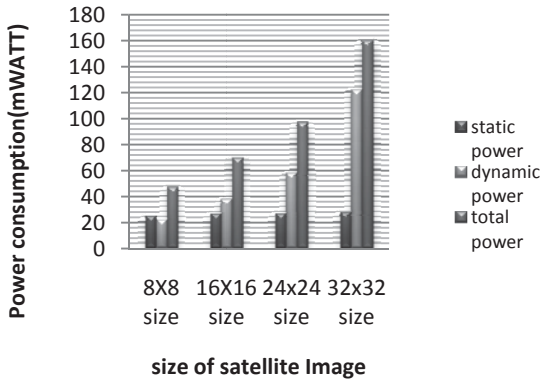

Fig.13 Dynamic and static power consumption

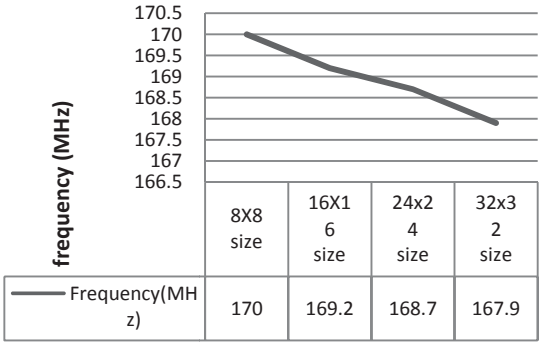

Fig.14 Frequency changes in terms of image block size

As it is noted, power consumption is divided into static and dynamic parts. Static power is almost constant, but using partial reconfiguration technique, the dynamic part is variable. It obviously increases the block size and the processor should compress more input data. This process increases the transmission bit rate in the hardware as well as thedynamic power dissipation rate. Table 2, which is derived from the results of simulation and synthesis of this architecture, shows increasing the image block size, causes increase in the chip temperature. Increasing temperature with block size (boosting number) matches increasing the dynamic power consumption as shown in Table 2.

Table 2. Increasing the block size and its effect on dynamic power consumption

Temperature of chip (FPGA Virtex-4) ^{o}C	block size
55.3	8x8
55. 4	16x16
55.46	24x24
55.52	32x32

Operational Frequeny

Using the results of ISE tool, we can determine the hardware frequency. Figure 14 shows the frequency changes of the hardware in terms of image block size. Figure 14 shows by augmenting the number of block size, circuit working frequency will not change remarkably. Low change of working frequencies is caused by increasing the number of block size.

Utilized Area

FPGA utilized area is one of the most important parameters in designing and using the FPGA as target hardware for design implementation. Table 3 shows the amount of FPGA resources utilization. This table shows device utilization comparison between this architecture and 2D-DCT designed in [18, 19].

Table 3 Resource utilization comparison between our architecture and [19],[20]

Resources Type	Proposed architecture	Work at [19]	Work at [20]
Number of Slices	658	1145	**7260**
Number of Slices FFs	924	1696	**9644**
Number of Slices LUTs	1450	1750	11194
Number of Multipliers	**2**	**11**	**--**

Because of using *addsub* instead of multipliers which was described at section 3, we observe that the device utilization at proposed architecture as compared

with the other implementation is so low. The RTL of proposed architecture is shown in Figure 15.

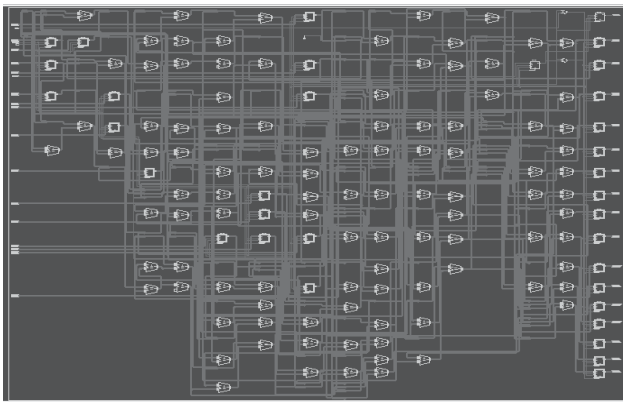

Fig. 15 RTL view of design

Implementation Tools

We have implemented our architecture on ML402 Xilinx embedded kit with virtex-4FPGA that supports partial reconfiguration technique with a 32MB FLASH ROM; therefore, we have a limitation in implementing large rule sets. The size of partial bit stream for reconfiguration of FPGA is almost 4MB. Hence, we can have 8 separate compressing algorithms on external memory. Doing so, we can implement real-time on-board satellite image processing architecture on hardware platform. Our implementation board at laboratory is shown in Figure 16.

Comparison with Other Models

In this section we have compared our proposed architecture with several other architectures. Our architecture using cooperative reconfigurable hardware cores provides a high speed, flexible, low weight, and low power, scalable and modular compression at hardware system which is completely suitable for real time on board of satellites.

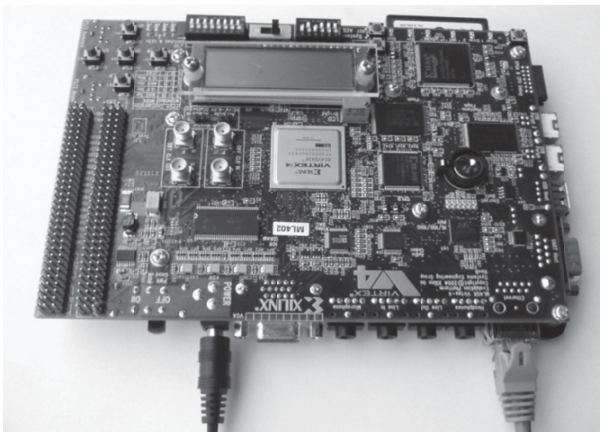

Fig. 16 Hardware platform used to implement our proposed architecture (ML 402 Xilinx)

Most of the current architectures don't have such a high speed processing. As compared to our model, their algorithm is in the memory and should run by a processor, but our model is implemented inside the FPGA and doesn't have the memory-CPU transfer time bottleneck. Moreover, they don't provide the flexibility, scalability and modularity of our model which prevents them from extending or reconfiguring their structure. The main advantages of our proposed architecture are low power consumption, high-speed processing in hardware, modularity, flexibility and high safety because of embedding all design in FPGA. In [20] one DSP for signal processing on board of satellite is proposed, this hardware is not reconfigurable and consume much power, almost 1 watt. But the proposed architecture in this paper is based on FPGA which is reconfigurable and consumes less power relative to the proposed architecture at [20]. In our work all of system is implemented on one chip, so we have system on chip (SOC). For this reason and the elimination of external wires for connecting processors proposed architecture have more reliability. Most proposed architectures are on software.[3, 9] But at our proposed architecture, the whole compression algorithm runs in the hardware. The proposed architectures at [18, 19] have operational frequency of 84MHz, but our proposed architecture operate at 170MHz, as seen previously. In addition, about device utilization, we can say that, because of using *AddSub* instead of multipliers which were described in section 3, we see that the device utilization at proposed architecture in comparison with the other implementation is so low. Therefore, power consumption at our proposed hardware is so less than other implementations [5,18, 20].

Conclusion

This paper proposed a new architecture with high safety, low weight, and high speed, low cost and low power which is compatible with and suitable for being used at on-board image compression on satellite and presented the design of adder architectures for their use in JPEG compression and more specifically in the 2-D DCT of the JPEG compressor. Then presented device utilization by this approach significantly decreases. Also this paper presents the application of reconfigurable hardware as cores in real-time on-board image compression in satellites. The proposed architecture is based on the partial reconfiguration technique. This architecture provides a new and wide horizon at hardware on-board processing design. the presented result shows that high processing speed despite low power consumption and high reliability by transferring full design to one chip, is achieved. We represented this new architecture with partial reconfiguration ability to separate and support several different compressing algorithms based on user command on the Earth and

then put them in an external memory. Then brought the specific part at a specific moment in embedded RAMs of FPGA to increase the processing speed and eliminate long memory access time which leads to reduced degrades system performance, reliability, and creates more bottleneck. By performing image compression on the satellite we can send fewer bits to the Earth Station for each image. Thereupon, power consumption for transmitting data to the Earth is decreased. With this architecture we overcome problems and bottlenecks in most of current architectures. Such hardware implementation has the potential of greatly improving the performance of image compression operations as compared to equivalent software which only carry out the implementations or current software compressions.

Acknowledgment

The authors wish to dedicate sincere thanks to *Dr. Mohammad Eshghi* for his lively discussions during evolution of this study.

References

[1] Tada, T., Cho, K., Shimoda, H. and Sakata, T., "An Evaluation of JPEG Compression for On-Line Satellite Images Transmission," National Space Development Agency of Japan (NASDA) Report, Vol. 1, 2002, pp. 30-39.

[2] Fouquet, M., Sweeting, M. N., "Earth Observation Using Low Cost Micro/Mini Satellites," *Acta Astronautica*, Vol. 39, No. 7, 1996, pp. 823–826.

[3] Hou, P., Petrou, M., Underwood, C.I. and Hojjatoleslami, A., "Improving JPEG Performance in Conjunction with Cloud Editing for Remote Sensing Applications," *IEEE Trans on Geoscience and Remote Sensing*, Vol. 38, No. 1, 2000, pp. 515-524.

[4] Home Site of the JPEG and JBIG Committees, Available, [on line]: http://www.jpeg.org

[5] Mohd Yusof, Z., Aspar, Z., Suleiman, I., "Field Programmable Gate Array (FPGA) Based Baseline JPEG Decoder," *IEEE*, Vol. 6, No. 3, 2000, pp. 218, 220.

[6] Hwang, K., *Computer Arithmetic Principles, Architecture and Design*, *4rd Edition, John Wiley & Sons*, USA, 1979, pp. 98-136.

[7] Agostini, L., "Design of Architectures for JPEG Image Compression," Report of Master Dissertation, Federal University of Rio Grande do Sul, Informatics Institute, Porto Alegre, Brazil, 2007, pp. 143-145.

[8] JPEG 2M0, Part I, Final Committee Draft, Version 1.0, Mar 2000, Available, [on line]: http://www.jpeg.org.

[9] Taubmun, D., "High Performance Scalable Image Compression with EBCOT," *IEEE Trans on Image Processing*, Vol. 9, No. 7, 2000, pp. 1158-1170

[10] Lian, Ch. Jr., Kum- Fu, Ch., Hong Hui, Ch. and Chen, L.G., "Analysis and Architecture Design of Block-Coding Engine for EBCOT in JPEG 2000," *IEEE Trans on Circuits and Systems for Silicon Technology*, Vol. 13, No. 3, 2003, pp.219-230.

[11] Trenschel, T., Bretschneider, M., T.R. and Leedham, G. C, "Region Based Guaranteed Image Quality in JPEG2000," *IEEE Proceedings of the ICICS-PCM, Singapore*, Vol. l, No. 5, 2003, pp.3-4.

[12] Taubman, D. S. and Marcellin, M. W., *JPEG2000 Image Compression Fundamentals, Standards and Practice*, 2nd Edition, kluwer Academic Publishers, Massachusetts, 2002, pp.137-158.

[13] Christopoulos, C., Askelof, J. and Larsson, M., "Efficient Region of Interest Coding Techniques in The Upcoming JPEG 2000 Still Image Coding Standard," *IEEE Proceedings of the International Conference on Image Processing*, Vol. 2, No.4, 2002, pp. 41–44.

[14] *Early Access Partial Reconfiguration User Guide*, Available, [on line]: www.xilinx.com, Version1.2, September, 2008, pp.28-64.

[15] *Plan Ahead UserGuide*, www.xilinx.com, Version 11.3.1, September 16, 2009, pp. 15-33.

[16] Kovac, M. and Ranganathan, N., "JAGAR: A Fully Pipeline VLSI Architecture for JPEG Image Compression Standard," *IEEE*, Vol. 83, No.1, 1995, pp. 247-258.

[17] Weste, N. and Eshraghian, K., *Principles of CMOS VLSI Design Second Edition*, Addison-Wesley, USA, 1995, pp. 125-174.

[18] Dhuhri Kusuma, E. and SriWidodo, T., "FPGA Implementation of Pipelined 2D-DCT and Quantization Architecture for JPEG Image Compression," *IEEE*, Vol. 8, No.3, 2010, pp. 243-248.

[19] Trang, D. and Bihn, N., "A High-Accuracy and High-Speed 2-D 8x8 Discrete Cosine Transform Design," *IEEE*, ICGCRCICT, Vol. 1, No.3, 2010, pp. 135-138.

[20] Nicholson, D. and Kajfasz, Ph., "An Effective Satellite On-Board JPEG2000 Image (De) Coding Implementation Based on PIRANHA Systematic-DSP," *Proceedings of 20th AIAA International Communication Satellite Systems Conference and Exhibit*, Vol. 4, No.7, 2002, pp.46-53.

[21] Yen Lee, T., Hsin Fan, Y., Cheng, Y. M. and Tsai, Ch. Ch., "Hardware-Software Partitioning for Embedded Multiprocessor FPGA Systems," *IEEE Trans, International Journal of Innovative Computing*, Vol. 5, No.10 (A), 2009, pp. 3071–3083.

[22] Bhandari Sheetal, U. and Subbaraman, Sh., "Real Time Video Processing on FPGA Using on the Fly Partial Reconfiguration," *IEEE, International Conference on Signal Processing Systems*,Vol. 6, No. 9, 2009, pp. 379-398.