VLSI Design, 2002 VOL. 15 (2), pp. 521–528

A Dynamically Reconfigurable Video Compression Scheme Using FPGAs with Coarse-grain Parallelism*

S. RAMACHANDRAN[†] and S. SRINIVASAN[‡]

Department of Electrical Engineering, Indian Institute of Technology, Chennai-600 036, Tamilnadu, India

(Received 12 December 2000; Revised 6 March 2001)

A dynamically reconfigurable scheme for video encoder to switch among many different applications is presented. The scheme is suitable for FPGA implementation and conforms to JPEG, MPEG-1, MPEG-2, and H.263 standards. The scheme has emerged as an efficient and cost-effective solution for video compression as a result of innovative design using well-partitioned algorithms, highly pipelined architecture and coarse-grain parallelism. The reconfiguration time of the video encoder is less than 320 μs while switching from one standard to another. Although the dynamic reconfiguration scheme is presented for a video encoder, the same design methodology may be applied effectively for any other application.

Keywords: FPGA; Coarse-grain parallelism; Video compression scheme; JPEG; MPEG

INTRODUCTION

The advent of SRAM-based Field Programmable Gate Arrays (FPGAs) presented a new capability to the electronics community; dynamically reconfigurable hardware or designs that one can reconfigure on the fly. Hardware designs broadly fit into one of the following configurations. Configurable hardware, a product whose function may be customized once or very few times; reconfigurable hardware, a product whose function may be customized many times; remotely reconfigurable hardware, a product whose function may be customized remotely, by telephone, radio or over the net, while remaining resident in the system and dynamically reconfigurable hardware, a product whose function may be customized on the fly any number of times while remaining resident in the system. Reconfigurable systems may be classified as having either static or dynamic reconfigurability [1,2]. A static reconfiguration refers to having the ability to reconfigure a system only once before execution, but once programmed, its configuration remains on the FPGA for the duration of the application. In contrast to this, the dynamic reconfiguration is defined as the selective updating of a subsection of an FPGA's programmable logic and routing resources while the remainder of the device's programmable resources continue to function without interruption [3]. This means that FPGAs are typically reconfigured many times during the normal operation of an application.

Static RAM based FPGAs can be configured in the field to implement a desired logic function. The architecture of an FPGA has a matrix of configurable logic blocks, programmable interconnects and programmable input/output cells. The designer specifies a logic function and then uses the compiler to map it to a network of configurable logic blocks, which are finally placed and routed. The end product on the FPGAs has a specific configuration that implements the desired logic function. The main limitation with the majority of SRAM-based FPGAs is that it is necessary to load the program code into the device entirely and not partially. It is usually necessary to halt the operation of the entire circuit board while these devices are being reconfigured. Additionally, the contents of any registers in the FPGAs are irretrievably lost during the process. The result is loss of vital data processed, often leading to a complete show down of the system. To address these issues, manufacturers developed a new generation of Dynamically Reconfigurable Programmable Gate Arrays (DPGAs) [2]. In addition to supporting the dynamic reconfiguration of selected portions of the internal logic,

*Portions reprinted, with permission, from ISCAS 2000—IEEE International Symposium on Circuits and Systems, May 28–31, 2000, Geneva, Switzerland.

[†]Corresponding author. E-mail: ram@ee.iitm.ernet.in

[‡]E-mail: srini@ee.iitm.ernet.in

ISSN 1065-514X print/ISSN 1563-5171 online © 2002 Taylor & Francis Ltd
DOI: 10.1080/1065514021000012138

these devices also feature continued operation of any portions of the device that are not undergoing reconfiguration. Further, there are no disruptions either to the device's inputs and outputs or to the system-level clocking. The hardware in such DPGA based systems can be reconfigured to adapt to different computing requirements for different applications. Reconfigurable systems offer higher computational density and higher throughput for many applications compared to conventional fixed hardware systems. Enormous hardware savings are also effected in such systems.

There are two basic approaches to implement dynamically reconfigurable applications: full reconfiguration and partial reconfiguration. Systems designed for full reconfiguration are allocated all FPGA resources in each configuration step, application being partitioned into distinct temporal modules of approximately equal size. In other words, for one application, entire execution code will have to be downloaded at one go. This is generally referred to as a coarse grain configuration. In contrast to this, in the partial reconfiguration only a small amount of code, known as fine grain configuration, needs to be downloaded. Normally, codes equivalent to about fifty logic cells or less is reckoned as belonging to fine grain configuration. However, the fine-grained partitioning is in general a challenging problem to address since it can cause a very high design penalty because of the increased flexibility and complexity of the system [2].

In recent years, dynamic reconfiguration has been applied for diverse applications such as network packet processing [4], image-processing [5–7], computer vision [8] and database searching [9]. Although a lot of research work has progressed in image or video compression using software and hardware, very little work seems to have been carried out in integrated implementation of discrete cosine transform and quantization (DCTQ), variable length coding (VLC), inverse quantization (IQ), inverse discrete cosine transform (IDCT), motion estimation and compensation (ME/MC) on FPGAs/ASICs and reconfiguring them for different applications and features. In reference [10], the system architecture of an adaptive reconfigurable DSP computing engine for numerically intensive front-end audio/video communications is presented. These include finite/infinite impulse response filtering, eight point block DCT, etc., but involve highly interconnected networks and, therefore are difficult to implement on FPGAs. The dynamically reconfigurable video encoder presented in this paper uses moderate-sized coarse grain configuration with very little change in interconnection network resulting in simple design and faster performance and thereby eliminating most of the limitations mentioned earlier. The encoder can be used in the following ways:

1. It can be configured for JPEG, MPEG-1, MPEG-2 or H.263 as per user request.
2. It can switch from one application to another dynamically, based on user request.
3. It can be programmed for color, picture size, channel rate, energy threshold level to achieve the desired image quality.

The paper is organized as follows. In the second section, the architecture of the dynamically reconfigurable video encoder catering to various image and video standards is described. The basic principle involved in partial reconfiguration of VLC is described in the third section. The FPGA implementation detailing module-wise chip size requirement is covered in the fourth section. The fifth section presents the reconfiguration times achieved, results and discussions followed by conclusions in the sixth section.

ARCHITECTURE OF THE DYNAMICALLY RECONFIGURABLE VIDEO ENCODER

An integrated DCT and Quantization block [11], a VLC block [12], a new, fast, one step search (FOSS) block matching algorithm for motion estimation and compensation [13] and an encoder integrating all these modules and featuring a novel automatic quality control algorithm [14,15] have all been implemented by the authors. The present work integrates all these modules offering a complete solution for a dynamically reconfigurable image/video encoder that can switch from one application to another. The VLC block has been modified to suit the present design.

Figure 1 depicts the basic architecture of the dynamically reconfigurable video encoder, capable of processing different video standards such as JPEG, MPEG-1, MPEG-2 and H.263. The design has the flexibility of reconfiguring dynamically from one standard to another. The host computer such as Pentium writes the standard, the picture size to be processed and the bit stream speed required to the encoder. A detailed description of the encoder is given in Ref. [15]. The raw image input is applied block-by-block into DCTQ for processing I frames. The resulting quantized coefficients are applied to the next stage, VLC, where they are assigned variable length codes and buffered by FIFO before they are sent out onto the serial channel as bit streams. VRDY, VSTRT and EOCV are, respectively, the ready, start and end of conversion signals for the VLC. IQ and IDCT designs implemented are exact inverses of DCT and Q processors. IQ and IDCT processes are concurrent to VLC process and reconstruct the image in I frame memory inside the FOSS motion estimator and compensator block. For processing P frames, the host computer inputs the macro block image into the current RAM inside the motion estimator and signals the start of the processing of motion estimation. After completing the same, the motion estimator sends the MOTION VECTOR data to DCTQ processor for further processing. The reconstructed error data are added to the corresponding pixel information in the previous frame RAM inside the

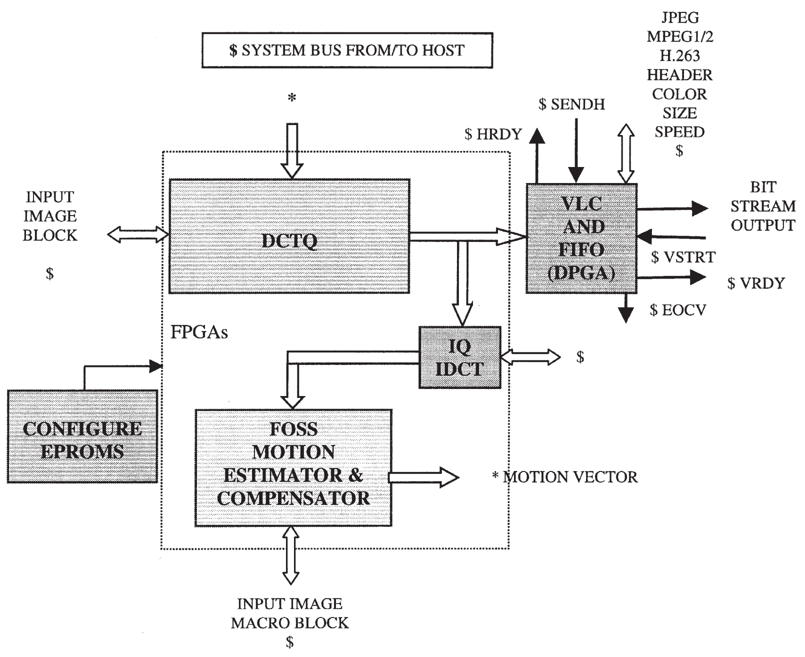

FIGURE 1 The basic architecture of dynamically reconfigurable video encoder.

motion estimator and compensator to construct the P frame. Concurrently, the processed motion vector and variable length encoded error bit streams are transmitted to the serial channel output. Prior to processing each macro block, the header information is processed by the VLC using hand-shake signals, HRDY and SENDH and stored in the First-In-First-Out (FIFO) memory.

DCTQ, IQ, IDCT and ME/MC process codes are located in CONFIGURE EPROMS and are transferred to four FPGAs, each of capacity 100,000 logic gates, once when the system is powered on. Simultaneously, the host processor downloads the entire VLC AND FIFO codes along with I/Os required and suitable for MPEG standard into a DPGA of capacity 30,000 logic gates. Full configuration of DPGA takes only 8 ms or less. As default, the system is configured to process MPEG-2. While the system is processing normally, it may be dynamically reconfigured within 320 μs to cater to another standard if user makes the request through host. This is easily done by downloading the partial configuration of DPGA meant for the new standard using the host processor as explained in "Architecture of the Dynamically Reconfigurable VLC" Section. IQ, IDCT and ME/MC processors are not used while processing JPEG.

DYNAMIC RECONFIGURABILITY SCHEME

Architecture of the Dynamically Reconfigurable VLC

Figure 2 shows the architecture of VLC which uses a DPGA device to effect dynamic reconfigurability. Configuration is the process of loading a design into the DPGA, which is SRAM-based. It can be reconfigured any number of times. The entire device or select portions of a design can be configured. Sections of the device can be configured while others continue to operate undisturbed. Although there are six possible modes of configuration available, the present work uses the fast, efficient parallel port of microprocessor for effecting the configuration data transfer. Setting M0–M2 inputs to MODE 6 does this. The configuration clock, CCLK, is derived from the host processor write signal, WR. CON* is a configuration control signal. Configuration starts on the rising edge of first CCLK when CON* is driven and held low. Configuration continues until CON* is pulled high by the configuration system (host processor). The device moves to the operation state on the first CCLK edge after

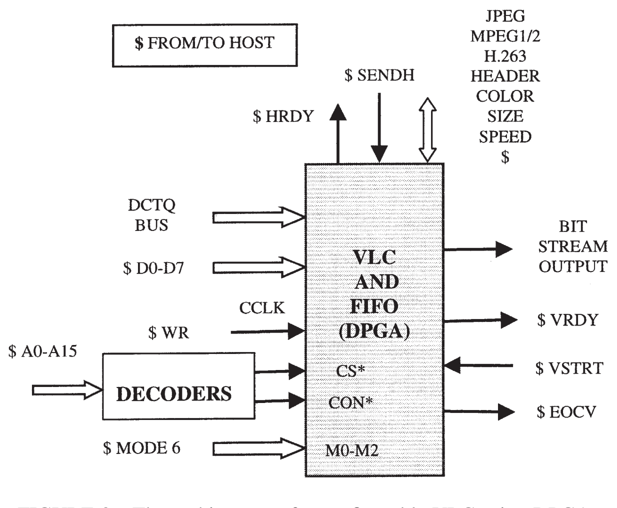

FIGURE 2 The architecture of reconfigurable VLC using DPGA.

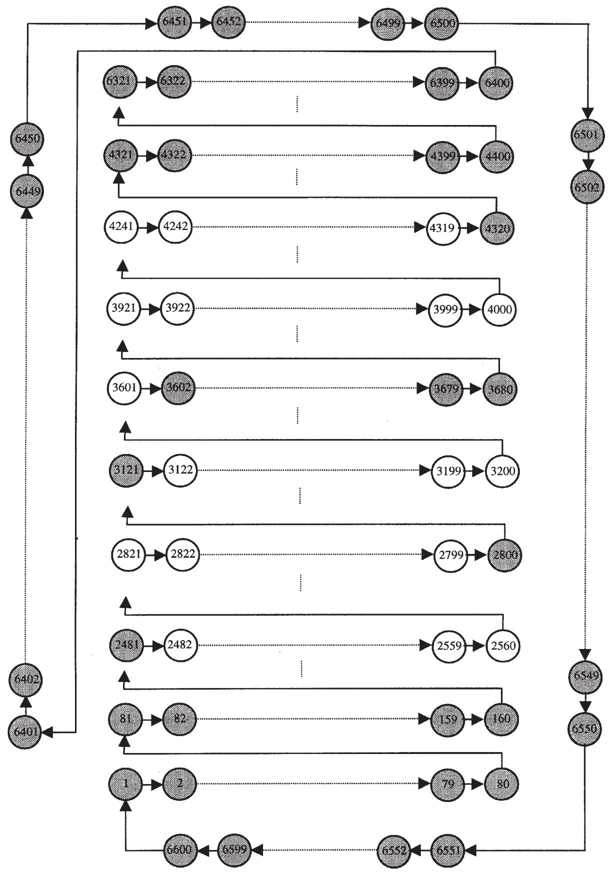

FIGURE 3 Partial configuration cell allocation for MPEG-2 application.

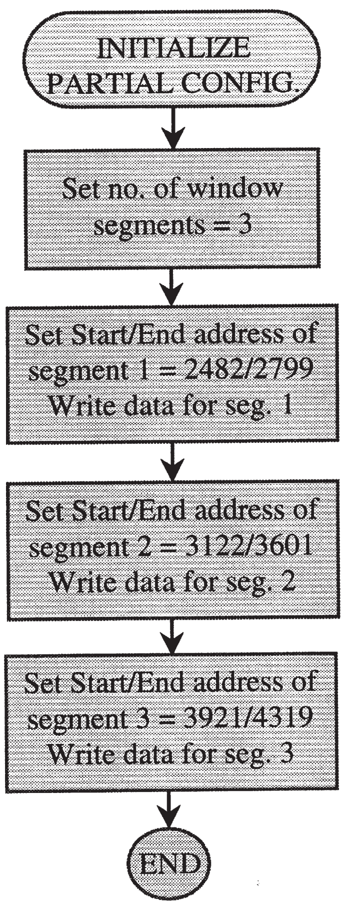
INITIALIZE PARTIAL CONFIG.
Set no. of window segments = 3
Set Start/End address of segment 1 = 2482/2799 Write data for seg. 1
Set Start/End address of segment 2 = 3122/3601 Write data for seg. 2
Set Start/End address of segment 3 = 3921/4319 Write data for seg. 3
END

FIGURE 4 Sequence of partial configuration for example shown in Fig. 3.

CON* is high. CS* is the configuration chip select pin. This pin must be low for configuration to occur. CON* and CS* signals are derived from the decoded address bus of the host processor. D0–D7 are data input pins connected to the host data bus.

Partial Reconfiguration

Figure 3 shows an example of partial reconfiguration cell allocation. The DPGA device [16] contains a total of 6400 logic cells arranged as 80×80 array lined with input/output pin cells. The device can be reconfigured in-system down to cell level and has a maximum bit-stream size of 16,394 bytes. Configuration begins with the bottom left logic cell numbered 1 and ends with the upper right logic cell numbered 6400. Then the I/O cells are configured beginning at 6401 and ending with 6600. By placing windows in the bit stream file, it is possible to configure only a portion of the array. The figure shows the bit stream file scheme used to partially configure three segments for MPEG-2 with cell address ranges: 2482–2799, 3122–3601 and 3921–4319. Cells to be configured partially are shown unshaded. The configured cell address ranges will be different for other standards. Although I/O pin cells are not partially configured in this implementation, they are also capable of being programmed partially.

Figure 4 shows the sequence of partial reconfiguration file that programs the DPGA selectively for the above example. After initialization, the total number of configuration segments that are in the configuration file is identified followed by the start and end addresses of the cells to be programmed in the first segment. This is followed by the actual data, which configures the cells in this segment to the desired circuit. Similarly, cell address range and configuration data for other two segments are also coded in the configuration file. The end of file is also identified finally. It may be noted that the shaded cells that were either configured previously or unused are left untouched—only the cell being programmed on a given clock cycle is changed. The other cells function as in their normal operation mode. This means that a portion of the array can be configured while the rest of the array remains operational. Mode 6 loads data in 8-bit words to decrease configuration time. Each byte in the data stream is setup and held with respect to the rising edge of CCLK described earlier. On power-up, all cells in the DPGA are programmed to be logical zeros. Unused cells in a design remain zeros and do not need to be configured. A configuration compression algorithm in the integrated development system for DPGA uses windowing to

TABLE I Chip size break-up for various functional modules of the reconfigurable video encoder

Functional module	Size in logic gates
DCTQ	1,40,000
IQ + IDCT	1,30,000
FOSS ME/MC	54,000
VLC (JPEG)	28,000
VLC (MPEG-1/MPEG-2)	26,000
VLC (H.263)	24,500

compress the configuration file. The compression algorithm skips unused cells and can reduce the configuration file size by up to 80%. This in turn reduces configuration time and memory storage requirements.

FPGA IMPLEMENTATION OF THE ENCODER

The dynamically reconfigurable video encoder has been implemented using Altera's 10 K series EPLDs with a chip set complexity of about 350,000 logic gates and Atmel's AT6000 series DPGA with complexity of about 28,000 logic gates. The design has been realized by using schematic entries, parameterized modules and VHDL/VERILOG codes. The implemented chip size distribution for various functional modules such as DCTQ, VLC, ME/MC and for various standards, JPEG, MPEG etc. are presented in Tables I and II, respectively. The controller for the encoder is built into the DCTQ module and, therefore, the size is marginally more than its inverse, IQ and IDCT. The memory elements required for Intra (I) and Inter (P) frames are external to the ME/MC module. There is adequate room for accommodating any new/extra feature such as the one-at-a-time step search motion estimation algorithm, for instance, in the FPGAs.

The dynamic reconfigurability for the entire encoder system is confined to only a portion of VLC residing in the DPGA, being a maximum of 1600 logic cells out of a total of 6400 cells available in the device. These are primarily the variable length codes that are to be generated for various standards and a small amount of controlling circuit. The VLC table for JPEG is longer than that for other standards, therefore, additional process codes are required. Other functional circuits that are common to all standards are dual redundant RAM for accepting DCTQ inputs, header processing, FIFO and controller. These occupy a maximum of 4000 cells in the device. These cells are one-time configured when the system is powered on.

TABLE II Total chip size requirement for various standards implemented

Standard implemented	Total size in logic gates
JPEG	1,68,000
MPEG-1/MPEG-2	3,50,000
H.263	3,48,500

RESULTS AND DISCUSSIONS

The reconfigurable video encoder has a highly pipelined and massively parallel architecture in order to achieve the best possible speed performance without sacrificing the quality of the picture. For a color picture, four luminance blocks, Y and two chrominance blocks, Cb and Cr are processed one by one in the order mentioned as per the standard. However, in the case of monochrome pictures, only four blocks are processed. Hence, the processing speed of a monochrome picture will be 150% that for the color picture. The ME/MC, DCTQ, IQ + IDCT and VLC processes are pipelined and work concurrently, taking an average processing time of 2170 ns per image block (8×8 pixels). This works out to processing of color pictures of size 1024×768 pixels at 25 frames per second.

RAM configuration elements inside FPGA allow them to be reconfigured in-circuit by loading new configuration data into the device. Forcing the device into command mode, loading configuration code, reinitializing the device, and resuming normal operation performs real-time configuration. The serial EPROMs used for configuration are of type EPC1 [17] and have been operated at 10 MHz clock. Each bit is downloaded from the EPROM at the clock frequency, i.e. at 100 ns per bit. In order to configure one FPGA of size: 100,000 logic gates, 1,200,000 data bits are required. Since the maximum requirement in this work is 350,000 logic gates for MPEG-1/MPEG-2 standards, the download time works out of 420 ms during the system start-up. Concurrently, the DPGA housing VLC code takes about 8 ms to get fully configured. Subsequently, for a partial reconfiguration only a maximum of 320 μs is required.

Table III presents the reconfiguration time for each of the standards, JPEG, MPEG-1/MPEG-2 and H.263, and the number of cells required to be reconfigured while switching from one standard to another. The DPGA is capable of partial reconfiguration at the rate of 0.2 μs/cell. Therefore, the entire reconfiguration process requires 320 μs or less and can be used to reconfigure

TABLE III Reconfiguration time for various functional modules of the reconfigurable video encoder

Partial functional module requiring reprogramming	Number of cells programmed	Partial reconfiguration time (μs)
VLC (JPEG)	1600	320
VLC (MPEG-1/MPEG-2)	1200	240
VLC (H.263)	900	180

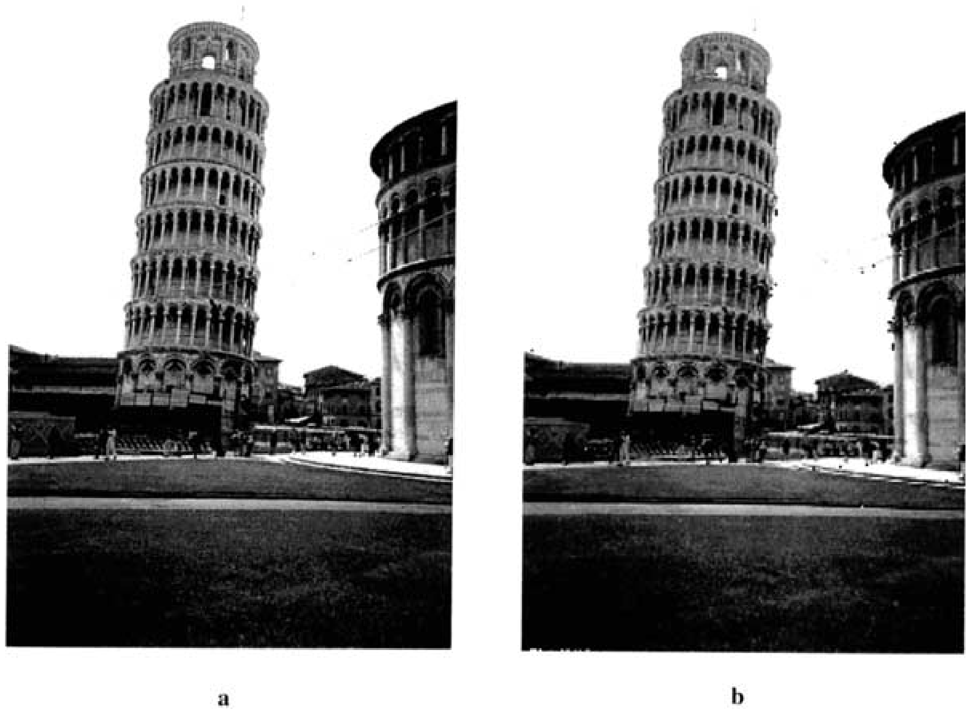

FIGURE 5 Simulation image using JPEG standard. (a) Original Leaning tower of Pisa image (Picture size: 800 × 1224 pixels). (b) Leaning tower of Pisa image reconstructed by the JPEG standard. PSNR: 33.9 dB, Compression: 0.49 bits per pixel, execution time per frame: 50.8 ms.

the required partial VLC dynamically. For instance, let the user who is currently receiving MPEG-2 video stream for surveillance application, request that the present transmission be suspended, and instead transmit a few, past, still color pictures of the same scene with higher resolution, 1600 × 1200 pixels in JPEG format. This example assumes that the same scene is also picturised using still digital camera, which has a buffer storage of size, 64 MB, equivalent to about 20 sequential, raw still frames. This can be accomplished dynamically within 320 μs by loading partial JPEG configuration data, which is of size 1600 cells into the DPGA that houses the VLC without disturbing other working circuit. Since the encoder reconfiguration time is far less than the processing time of 40 ms per frame, it will not miss processing even a single frame. Such a feature finds use in wide variety of applications such as tele-medicine, quality control, video conferencing, etc. Similarly, the reconfiguration time for MPEG-1/MPEG-2 and H.263 are 240 and 180 μs, respectively. It may be noted that only partial and not full VLC code needs to be downloaded in order to effect the reconfiguration.

Maximum chip size required for each of the implemented standards is furnished in Table II as explained in "FPGA Implementation of the Encoder" Section. This means that we need FPGAs totalling 1,216,500 logic gates if we are to implement all the four standards independently. Instead, as explained earlier, we need employ only 350,000 logic gates in the dynamic reconfiguration method, thus effecting considerable savings. Savings will be even more if we are to add more standards such as H.261, MPEG-4, simple profile.

Simulation tests were carried out successfully on the reconfigurable video encoder at 50 MHz clock operation and verified by C and Matlab programs. The encoder can be configured for different standards as explained earlier. Figure 5 shows the reconstructed image obtained by configuring the encoder in JPEG mode. The image quality,

TABLE IV Execution times, Quality of the image (PSNR) and compression (bits per pixel) achieved for some of the images using the reconfigurable video encoder configured in MPEG-2 standard mode

Image	Image size in pixels	Frame numbers	Average execution time/frame (ms)	Average PSNR (dB)	Average compression (bits/pixel)
Susie	256 × 256	0–50	2.0	36.49	0.83
Car	256 × 256	51–100	2.1	35.70	1.18
Bmw	352 × 288	90–110	4.3	35.28	3.09
Titanic	352 × 288	50–83	4.7	35.31	2.74
Rugby	480 × 688	0–9	13.8	36.11	1.54
Table tennis	480 × 720	0–15	12.7	35.12	2.30

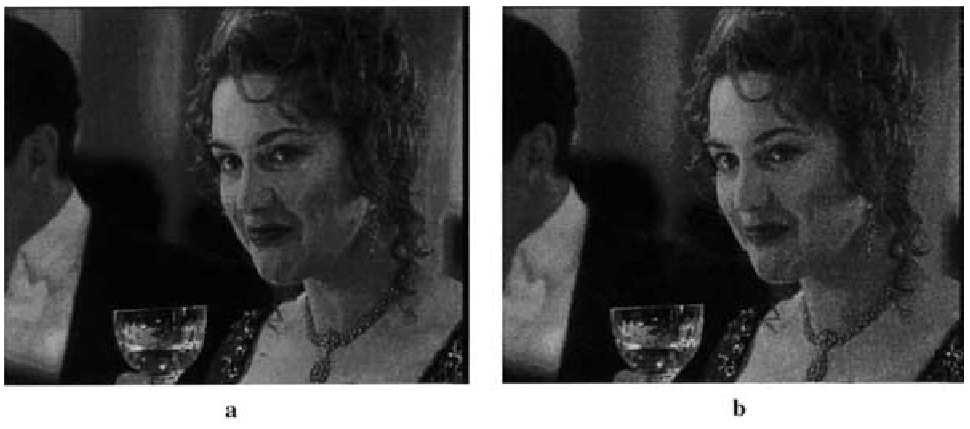

FIGURE 6 Simulation image using one of the configured standards, MPEG-2. (a) Original Titanic image (Frame number: 83, Picture size: 352 × 288 pixels). (b) Same Titanic image reconstructed by the new FOSS motion estimation method.

compression and execution time obtained for the sample image are also indicated in the figure. The standards, MPEG-1, MPEG-2 and H.263 basically use the same functional modules such as DCTQ, VLC, ME/MC etc. and have the same execution sequence and timing, and, therefore, results for MPEG-2 standard only are presented. Table IV presents the results obtained for six different images using the MPEG-2 standard mode and Fig. 6, the reconstructed image for one of the sequences.

CONCLUSIONS

An FPGA implementation of a real time video compression scheme that can be dynamically reconfigured from one standard to another using novel features has been presented. The design is modular and flexible, and therefore, it can be upgraded to accommodate more compatible standards, both present and future, without appreciable increase in hardware. The functional modules such as DCTQ, IQ, IDCT and ME/MC residing in FPGAs presently can be replaced by ASIC resulting in more compact, low power, higher speed and cost-effective system suitable for volume production. In such a system, the VLC module can continue to reside in the DPGA in order to preserve the power of dynamic reconfigurability. A decoder can also be designed on similar lines to build a codec.

References

[1] Sanchez, E., Sipper, M., Haenni, J.O. and Andres, P.U. (1999) “Static and dynamic configurable systems”, *IEEE Transactions on Computers* **48**(6), 556–564.

[2] Zhang, Xuejie and Ng, Kam W. (2000) “A review of high-level synthesis for dynamically reconfigurable FPGAs”, Microprocessors and Microsystems (Elsevier, Amsterdam) **Vol. 24**, pp. 199–211.

[3] Lysaght, P. and Dunlop, J. (1993) “Dynamic reconfiguration of FPGAs”, *Proceedings of the International Workshop on Field-Programmable Logic and Applications*, 82–94.

[4] Lockwood, John W., Naufel, Naji, Turner, Jon S. and Taylor, David E. (2001) “Reprogrammable network packet processing on the field programmable port extender”, *Proceedings of ACM/SIGDA Ninth International Symposium on FPGAs, Monterey, CA*, 87–93.

[5] Estlick, Mike, Leeser, Miriam, Theiler, James and Szymanski, John J. (2001) “Algorithmic transformations in the implementation of K-means clustering on reconfigurable hardware”, *Proceedings of ACM/SIGDA Ninth International Symposium on FPGAs, Monterey, CA*, 103–110.

[6] Ross, O.V. and Turner, M. (1994) “An FPGA-based hardware accelerator for image processing”, *Proceedings of the International Workshop on Field-Programmable Logic and Applications*, 299–306.

[7] Lysaght, P., Stockwood, J., Law, J. and Grima, D. (1994) “Artificial neural network implementation on fine-grained FPGA”, *Proceedings of the fourth International Workshop on Field-Programmable Logic and Applications*, 421–431.

[8] Luk, T.W. and Page, I. (1994) “Hardware–software codesign of multidimensional programs”, *Proceedings of FCCM ’94*, 82–90.

[9] Lemoine, E. and Merceron, D. (1995) “Run time reconfiguration of FPGAs for scanning genetic databases”, *Proceedings of FCCM ’95*, 85–89.

[10] Wu, An-Yeu, Ray Liu, K.J. and Raghupathy, A. (Feb. 1998) “System architecture of an adaptive reconfigurable DSP computing engine”, *IEEE Transactions on Circuits and Systems for Video Technology* **8**(1), 54–73.

[11] Ramachandran, S., Srinivasan, S. and Chen, R. (May–June 1999) “EPLD-based architecture of real time 2D-discrete cosine transform and quantization for image compression”, *IEEE International Symposium on Circuits and Systems*, iii375–378, Orlando, Florida.

[12] Ramachandran, S. and Srinivasan, S. (May 28–31, 2000) “Design and implementation of an EPLD-based variable length coder for real time image compression applications”, *The IEEE International Symposium on Circuits and Systems (ISCAS), Geneva, Switzerland*, I607–610.

[13] Ramachandran, S. and Srinivasan, S. (February 11–13, 2001) “FPGA implementation of a novel, fast motion estimation algorithm for real-time video compression”, *Proceedings of the Ninth ACM International Symposium on Field-Programmable Gate Arrays, Monterey, CA*, 213–219.

[14] Ramachandran, S. and Srinivasan, S. (2002) “A novel, automatic quality control scheme for Real time image transmission”, *VLSI DESIGN Journal*, **14**(4), 329–335.

[15] Ramachandran, S., Srinivasan, S., “Design and implementation of a novel, fast automatic quality control scheme-based MPEG-2 image encoder”, *Proceedings of World Multiconference, SCI 2001 and ISAS 2001*, Orlando, Florida.

[16] Configuration guide for AT6000 series FPGAs, Atmel Corporation, San Jose, CA, pp. 1–20.

[17] Application note, AN116, “Configuring APEX 20 K, FLEX 10 K and FLEX 6000 devices”, ALTERA Corporation, San Jose, CA, pp. 3–15.

S. Ramachandran is a research scholar at the Indian Institute of Technology, Madras. He has a wide industrial experience, having worked in the design of FPGA and Microprocessor based systems. His research interests include parallel algorithms, video processing, DSP, reconfigurable computing using FPGA and ASIC designs. He is the recipient of the Best Design Award at VLSI DESIGN 2000 International Conference. He is a member of IEEE.

S. Srinivasan is a professor in the Electrical Engineering Department at the Indian Institute of Technology, Madras. His teaching and research interests are in the areas of Digital Design, DSP Architectures and Applications and VLSI Design. He has taught as a Visiting Faculty at the University of California, Davis and at the California State University, San Jose. He is also the recipient of German Academic Exchange Fellowship and the Alexander von Humboldt Fellowship. He is currently coordinating the VLSI activity at the Indian Institute of Technology, Madras. Dr Srinivasan is a Senior Member of IEEE, a Fellow of the IETE (India) and a member of the VLSI Society of India.