IOSR Journal of VLSI and Signal Processing (IOSR-JVSP)
Volume 6, Issue 5, Ver. II (Sep. - Oct. 2016), PP 47-53
e-ISSN: 2319 – 4200, p-ISSN No. : 2319 – 4197
www.iosrjournals.org

Efficient design and FPGA implementation of JPEG encoder

Krupali M.Lanjewar[1], R.S.Kawitkar[2]
[1, 2]*(Department of E &Tc S.C.O.E.Vadgaon, Pune, India)*

Abstract: *In this study JPEG standard is used in digital camera which is used to compress the captured image. Hence less space is needed to store the image. Such images are easily shared on the web.*
Methods/Analysis: *Generally, Photos captured from digital camera are big in size, so it takes more time to upload such photos on web. Some time connection speed creates bottle neck on uploading these photos. Best way is to save such photos by .jpg extension and share on the web. Sometimes, while copying the original data, it may not fit in pen drive or memory stick but zip data could easy fit in pen /memory stick. More data one can put in pen drive due to compression. Hence, in this research work, JPEG encoder scheme is introduced to compress the captured image data from sensors so as to reduce the storage requirements. This scheme begins with the input image having width and height multiple of 16 that is, an image 32x32 will produce a strictly compliant JPEG image file. Here the header is the widely employed JFIF. Image resolution is not limited. It takes an RGB input. Hence it is then converted to YCbCr color space. After conversion apply a level shift as specified by the JPEG standard. Then it is applied to DCT block. After the FDCT is computed for a block, each of the 64 resulting DCT coefficients is quantized and then applied for entropy encoding. Finally we get a compressed image.*
Findings: *The proposed JPEG works significantly outperform existing techniques in terms of the parameters like gate counts, memory and clock. It also shows that this JPEG encoder architecture uses 4992 slices, 8273 LUTs, 77I/Os of Xilinx Vertex – 5 xc2v1000-4bg575 FPGA and works at an operating frequency of 37.129 MHz.*
Keywords: *Discrete Cosine Transform* **(***DCT), Field Programmable Gate Array (FPGA), Huffman coding, Joint Photographic Expert Group (JPEG), Quantization.*

I. Introduction

The use of digital technology in day-to-day life and specialized applications are increasing continuously. In particular, the use of digital imagery growing continuously as the technology to create such media becomes cheaper and easily accessible. For example such technological advances in many fields like robotics, medicine and defense. Hence, it increases the need for digital imagery. As more applications demands for higher resolution imagery which depends on it for critical information, storing of data efficiently while maintaining data integrity is main. To illustrate this, consider a standard smart phone having features like 8 megapixel camera. An image is taken in monochrome with 8 bits of precision for each pixel. This requires 8 MB of memory before getting compressed. This requires a total of 240 MB of space per second. Thus, it requires total of 14 GB space per minute. Hence, to store such images in this format is not possible, especially when the proposed system use color images or higher resolution images.

M. Gangadhar et al.[1] formulated a high speed FPGA based scheme of Embedded Block Coding with Optimized Truncation (EBCOT) algorithm used in JPEG 2000. This scheme analyzes the context formation engine used in EBCOT and hence proposes a scheme based on parallel processing of the three coding passes. A three stage pipelined architecture for the Arithmetic Encoder is used to speed up the encoding. When implemented on a XC2V1000 device, the design performs at 50 MHz after place and route. This scheme results that the processing time is reduced by more than 75%.

The scheme [2] involves a complex sub-block discrete cosine transform (DCT), along with other quantization, zigzag and Entropy coding blocks. In this scheme, Verilog design and hardware implementation of pipelined 2-D DCT along with zigzag, quantization and variable length coding is described. 2-D DCT is computed by combining two I-D DCT that connected by a transpose buffer. The scheme uses 4059 slices, 6885 LUT, 58 I/Os of Xilinx Spartan-3 XC3S1500 FPGA and works at an operating frequency of 65.55 MHz. The delay of processing each 8*8 block in an image is also evaluated to be 1.47micro seconds.

The scheme [3] presents FPGA based High speed, low complexity and low memory implementation of JPEG decoder. The pipeline implementation of the system, allow decompressing multiple image blocks simultaneously. The hardware decoder is designed to operate at 100MHz on Altera Cyclon II or Xilinx Spartan 3E FPGA or equivalent. The decoder is capable of decoding Baseline JPEG color and gray images. Decoder is also capable of down-scaling the image by 8. The decoder is designed to meet industrial needs. JFIF, DCF and EXIF standers are implemented in the design.

The scheme [4] presents a linear, highly pipelined direct polynomial fast 2-D DCT algorithm hardware implementation in which the number of multiplication is reduced to 50% of the conventional row-column approach. The coding is simulated using Xilinx 9.1 ISE synthesized using Spartan II. The 1-D and 2-D DCT implementation is done using 18-bit floating point adders, sub tractors and multipliers. The DCT processor saves hardware using module reusability concept and achieves high performance. The chip of 8x8 DCT and Quantization processor are fabricated in a 180 nm CMOS technology. Power analysis is analyzed using CADENCE tool.
The purpose of this research was to reduce the storage requirements and easily data transfer on the web.

II. Proposed Methodology

In this research work proposed a new method for efficient realization of jpeg encoder on Field Programmable Gate Array (FPGA). This scheme consists of taking an input image in RGB format, Conversion of RGB to YCbCr, FDCT followed by level shifting, quantization and then entropy encoding. The complete procedure of proposed JPEG encoder scheme is described in Figure 1. The whole process is modeled into Xilinx which is further integrated at top level entity using structural program. The top level entity “Compressor” is binding these blocks. There are four distinct modes of operation under which the various coding processes are defined:

i) Sequential DCT-based,
ii) Progressive DCT-based,
iii) Lossless, and
iv) Hierarchical

Among these processes, the proposed system has used sequential DCT-based operation.
A summary of the characteristics of the baseline coding processes [5] is given as:

1. DCT-based process
2. Source image : 8-bit samples within each component
3. Sequential process
4. Huffman coding : 2AC and 2DC tables

2.1. RGB to YCbCr Conversion

YCbCr (256 levels) can be computed directly from 8-bit RGB as follows:

$$Y = 0.299\ R + 0.587\ G + 0.114 \qquad (1)$$

$$Cb = -\ 0.1687\ R - 0.3313\ G + 0.5\ B \qquad (2)$$

$$Cr = 0.5\ R - 0.4187\ G - 0.0813\ B \qquad (3)$$

All image file formats does not store image samples in the order R0, G0, B0, ... Rn, Gn, Bn. The proposed system must verify the sample order before converting an RGB file to JFIF.The first process is to convert the input signals (Red, Green and Blue, strobed by Process RGB signal) to the YC_bC_r color space. After conversion apply a level shift as specified by the JPEG standard. CoreGen generated with wrapper file is used here.

2.2 Level shift

Before encoding process computes the forward DCT [5] for a block of source image samples, the samples will be level shifted to a signed representation by subtracting 2P – 1, where P is the precision parameter specified.

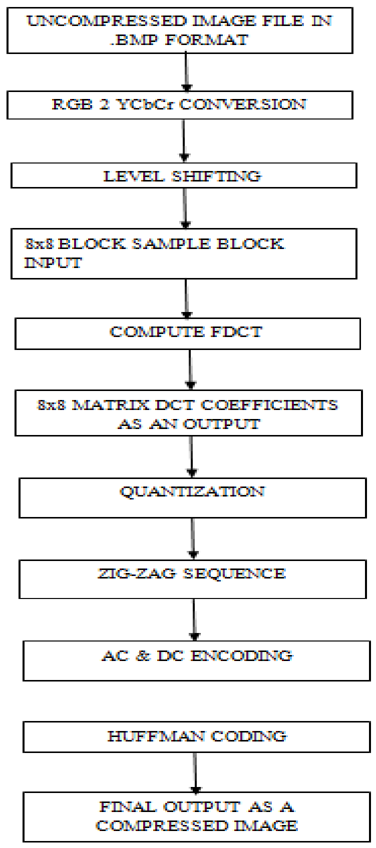

UNCOMPRESSED IMAGE FILE IN .BMP FORMAT
RGB 2 YCbCr CONVERSION
LEVEL SHIFTING
8x8 BLOCK SAMPLE BLOCK INPUT
COMPUTE FDCT
8x8 MATRIX DCT COEFFICIENTS AS AN OUTPUT
QUANTIZATION
ZIG-ZAG SEQUENCE
AC & DC ENCODING
HUFFMAN CODING
FINAL OUTPUT AS A COMPRESSED IMAGE

Figure1. Working procedure of proposed JPEG encoder

2.3. Discrete Cosine Transform (DCT)
There are four distinct modes of operation under which the various coding processes are defined:
1. Sequential DCT-based,
2. Progressive DCT-based,
3. Lossless, and
4. Hierarchical

The simplest DCT-based coding process is referred to as the baseline sequential process. It provides a capability which is sufficient for many applications. There are additional DCT-based processes which extend the baseline sequential process to a broader range of applications.
Among these processes, we have used sequential DCT-based operation. For the sequential DCT-based mode, 8 x 8 sample blocks are typically input block by block from left to right and block-row by block-row from top to bottom.

Procedure for encoding an 8 x 8 block data unit
For DCT-based encoders, the data unit consists of an 8 x 8 block of samples. The procedure [5] for encoding 8 x 8 block data is given as follows.

i. Before computing the FDCT, the input data are level shifted to a signed two's complement representation. For 8-bit input, the level shift is achieved by subtracting 128. Then calculate forward 8 x 8 DCT. Then quantize the resulting coefficients using table specified in frame header. Thus, AC and DC coefficients are obtained.
ii. Using DC table specified in scan header, encode DC coefficients for 8 x 8 block.
iii. Similarly, using AC table specified in scan header, encode AC coefficients for 8 x 8 block

2.4. Quantization
Quantization [6] is the process of selectively discarding visual information without a significant loss in the visual effect. Quantization reduces the number of bits needed to store an integer value by reducing the precision of the integer. Each discrete cosine transform (DCT) component is divided by a separate quantization coefficient, and rounded to the nearest integer. The larger the quantization coefficient (i.e., coefficient weighting), the smaller the resulting answer and associated bits needed to express the DCT component.

Quantization is done to achieve better compression. Quantization reduces the number of bits needed to store information by reducing the size of the integers representing the information. After the FDCT is computed for a block, each of the 64 resulting DCT coefficients is quantized by a uniform quantizer.
The uniform quantizer is defined by the following equation. Rounding is to the nearest integer[5]:

$$Sq_{vu} = round\ \frac{S_{vu}}{Q_{vu}} \qquad (4)$$

Where,
Sq_{vu} is the quantized DCT coefficient, normalized by the quantizer step size.

2.5 Differential DC encoding

After quantization, and in preparation for entropy encoding, the quantized DC coefficient is treated separately from the 63 quantized AC coefficients. The value that shall be encoded is the difference (DIFF) between the quantized DC coefficient of the current block (DC_i) and that of the previous block of the same component (PRED):

$$DIFF = DC_i - PRED \qquad (5)$$

2.6. Entropy encoding

There are two types of the entropy-coding:
1. Huffman encoding
2. arithmetic encoding

The baseline sequential process uses Huffman coding hence, Huffman encoding is used. Huffman coding [7] is used to code values statistically according to their probability of occurrence. Short code words are assigned to highly probable values and long code words to less probable values.

2.7 Zig-zag sequence

After quantization, and in preparation for entropy encoding, the quantized AC coefficients are converted to the zig-zag sequence. The quantized DC coefficient (coefficient zero in the array) is treated separately. The zigzag sequence is specified in Figure 2.

0	1	5	6	14	15	27	28
2	4	7	13	16	26	29	42
3	8	12	17	25	30	41	43
9	11	18	24	31	40	44	53
10	19	23	32	39	45	52	54
20	22	33	38	46	51	55	60
21	34	37	47	50	56	59	61
35	36	48	49	57	58	62	63

Figure 2 Zig-zag sequences of quantized DCT coefficients

2.8 Conversion of Huffman table specifications to tables of codes and code lengths

Conversion of Huffman table specifications [5] to tables of codes and code lengths uses three procedures.
1. generates a table of Huffman code sizes.
2. generates the Huffman codes
3. generates the Huffman codes in symbol value order.

The entries in the tables are ordered according to increasing Huffman code numeric value and length.

2.8.1 Bit ordering within bytes

The root of a Huffman code is placed toward the MSB (most-significant-bit) of the byte, and successive bits are placed in the direction MSB to LSB (least-significant-bit) of the byte. In this way, Huffman encoding starts by looking up the tables and stores the final compressed image. The proposed JPEG encoder is shown in figure 3.

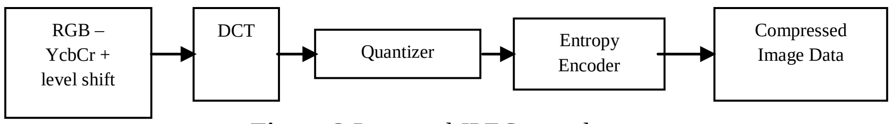
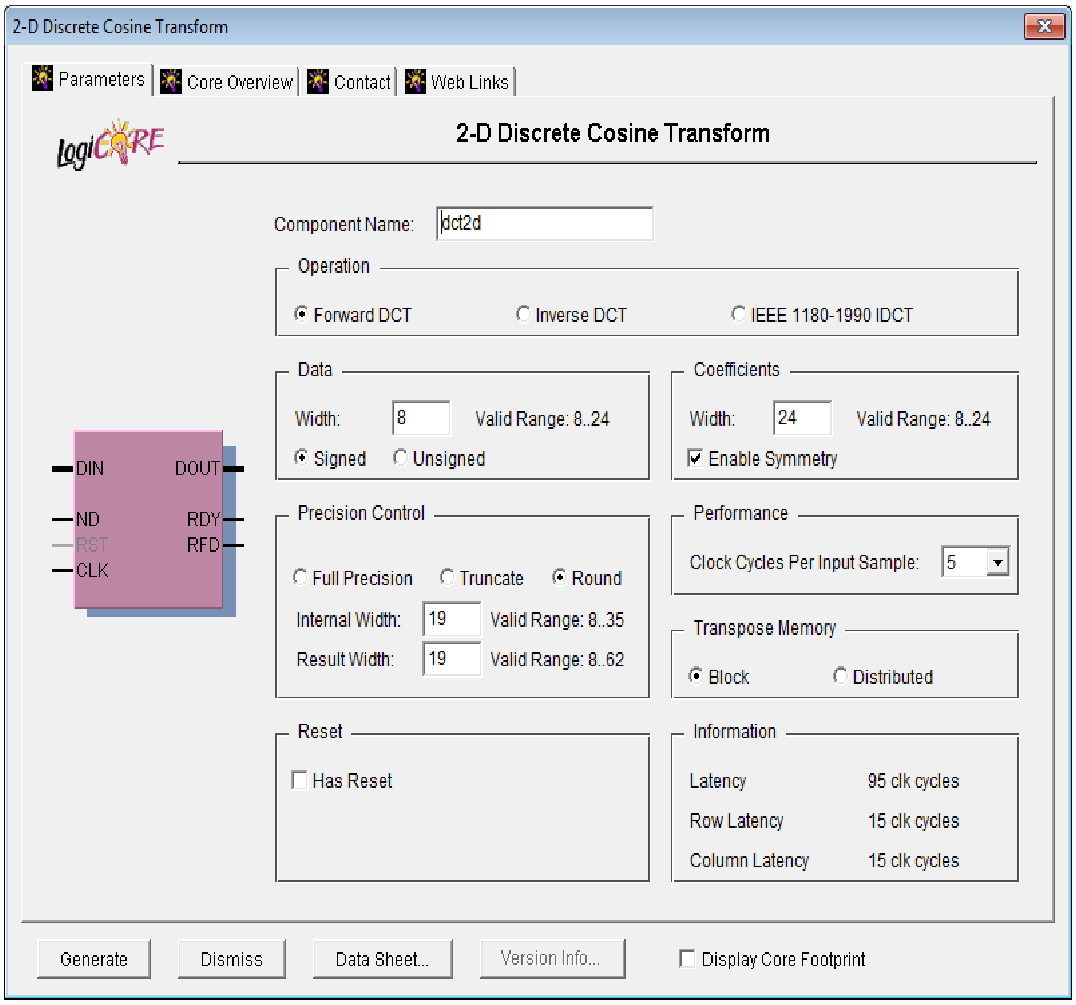


Figure 3 Proposed JPEG encoder

III. Implementation Results

In order to assess the proposed system Xilinx CORE generator is used. The CORE Generator[8] System is a design tool that delivers parameterized cores optimized for Xilinx FPGAs. It provides you with a catalog of ready-made functions ranging in complexity from simple arithmetic operators such as adders, accumulators, and multipliers, to system-level building blocks such as filters, transforms, FIFOs, and memories. A core cannot be remotely accessed from within a project. Cores must reside inside the project directory in order to be accessible to the project. CoreGen is not available for CPLD devices. It is not possible to copy cores from one project to another. You will need to generate new cores if you import a design from another design environment. CoreGen cores are often optimized for a particular device family. To avoid errors in implementation after changing to a new device family, most cores need to be regenerated for the new family.

There are 7 CoreGen generated files used for storing final image. Out of these, DCT block is explained. dct2d.vhd : This is CoreGen generated file with wrapper used for 2-D Discrete Cosine Transform (Forward DCT). Here , Data Width = 8 bits Signed, Coefficients Width = 24 (Enable Symmetry), Precision Control: Round, Internal Width: 19, Result Width: 19, Performance: Clock Cycles per input=9, Transpose Memory = Block, Reset: No. (Results: Latency = 95 cycles, Row Latency = 15 cycles, Column Latency = 15 cycles) are used. It takes as input sixty-four 8 bits signed values and gives output as sixty-four 19 bits signed numbers in the frequency domain (the LSBs are decimals). It is shown in figure 4.



2-D Discrete Cosine Transform

Component Name: dct2d

Operation: [✓ Forward DCT] Inverse DCT IEEE 1180-1990 IDCT

Data: Width: 8 Valid Range: 8..24; [✓ Signed] Unsigned

Coefficients: Width: 24 Valid Range: 8..24; [✓ Enable Symmetry]

Precision Control: Full Precision Truncate [✓ Round]; Internal Width: 19 Valid Range: 8..35; Result Width: 19 Valid Range: 8..62

Performance: Clock Cycles Per Input Sample: 5

Transpose Memory: [✓ Block] Distributed

Reset: Has Reset

Latency	95 clk cycles
Row Latency	15 clk cycles
Column Latency	15 clk cycles

Display Core Footprint

Figure4. CoreGen generated dcd2d GUI

JPEG encoder with 77 Input and outputs are shown in Figure 5.

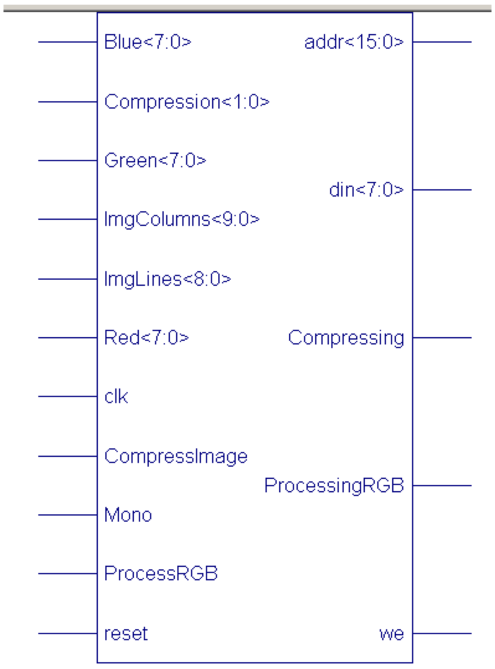


Figure 5: Top level RTL for JPEG Encoder

The device utilization for JPEG Encoder is as reflected in synthesise report and it shown in figure 6 which is part of various levels of RTL.Out of 5120 Slices 4992 are used in this design i.e. 97% utilization. Numbers of Slice Flip Flops are 3949 out of 10240.Number of 4 input LUTs: 8273 out of 10240.

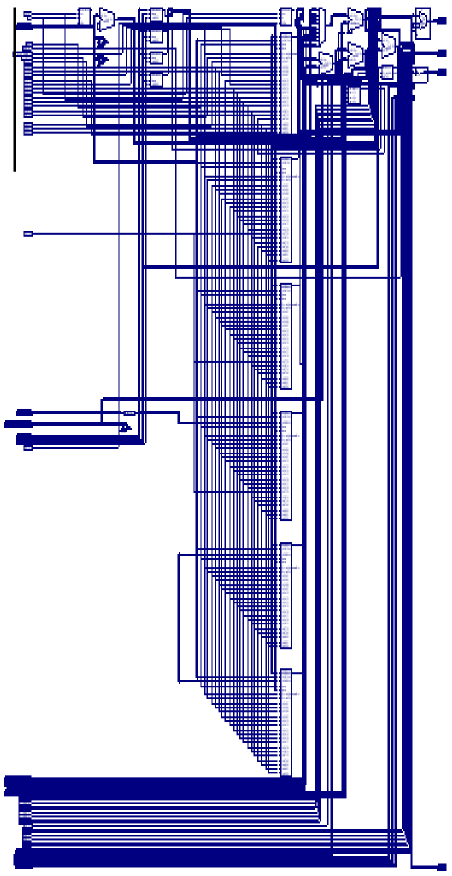


Figure 6: Detailed RTL level JPEG Encoder

The JPEG encoder architecture was described in VHDL. The design was synthesized into a Xilinx Vertex-5 xc2v1000-4bg575 family FPGA. The complete synthesis results to Vertex-5 FPGA are presented in Table 1, whose hardware was fit in an xc2v1000-4bg575 device.

Table 1: Device Utilization Using Xilinx Vertex – 5

LOGIC UNIT	USED	AVAILABLE	UTILIZATION
Number of slice	4992	5120	97 %
Number of slice FF’s	3949	10240	38%
Number of 4-input LUT’s	8273	10240	80%
Number of bonded IOB’s	77	328	23%
Number of BRAMs	13	40	32%
Number of MULT18X18s	9	40	22 %
Number of clocks	1	16	6%

According to synthesis result, constraint yields minimum clock period 26.933ns. Maximum clock frequency can be used is 37.129MHz. Figure 7 shows simulation of DCT block.

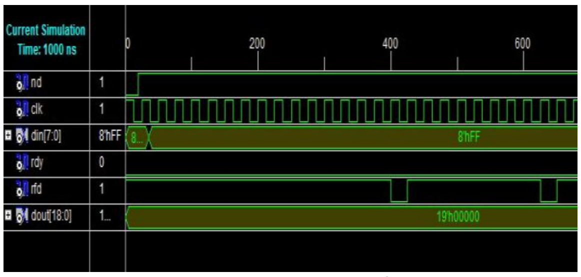


Figure 7: Simulation result of DCT block

IV. Conclusion and future scope

In this work implementation of JPEG encoder architecture for JPEG image compression standard is described. The architectures for the various stages are based on efficient and high performance designs suited for VLSI implementation. The implementation was tested for functional correctness using VHDL with Xilinx tool. Study of different image interchange format, image header format and design implementation flow of CoreGen is carried out. The synthesis is carried out on Vertex 5 platform. The JPEG Encoder implementation on Vertex 5 infers 12002 gate counts. This design works on a frequency 37.129MHz.JPEG standard is used in digital camera which is used to compress the captured image. Hence less space is needed to store the image. Such images are easily shared on the web.

References

[1] M. Gangadhar and D. Bhatia, “FPGA based EBCOT architecture for JPEG 2000,” *Proc. - 2003 IEEE Int. Conf. Field-Programmable Technol. FPT 2003*, vol. 29, pp. 228–233, 2003.
[2] S. Sanjeevannanavar and A. N. Nagamani, “Efficient design and FPGA implementation of JPEG encoder using verilog HDL,” *Proc. Int. Conf. Nanosci. Eng. Technol. ICONSET 2011*, pp. 584–588, 2011.
[3] J. Ahmad, “FPGA based implementation of baseline JPEG decoder,” no. January, 2009.
[4] “VLSI Implementation of 2-D DCT and Quantization processor for JPEG Image Compression,” no. September, 2016.
[5] International Telecommunication Union, “Terminal equipment and protocols for telematic services,” *Study Gr. VIII, ITU, Geneva*, 1988.
[6] V. Series, “Product Obsolete / Under Obsolescence,” vol. 615, pp. 1–10, 2003.
[7] L. Pillai, “XAPP616: Huffman Coding,” vol. 616, pp. 1–16, 2003.
[8] D. Bakalis, K. D. Adaos, D. Lymperopoulos, M. Bellos, H. T. Vergos, G. P. Alexiou, and D. Nikolos, “A core generator for arithmetic cores and testing structures with a network interface,” *J. Syst. Archit.*, vol. 52, no. 1, pp. 1–12, 2006.