Journal of Circuits, Systems, and Computers
Vol. 16, No. 6 (2007) 1011–1026
© World Scientific Publishing Company

World Scientific
www.worldscientific.com

A SIMPLIFIED 8 × 8 TRANSFORMATION AND QUANTIZATION REAL-TIME IP-BLOCK FOR MPEG-4 H.264/AVC APPLICATIONS: A NEW DESIGN FLOW APPROACH

IHAB AMER*, WAEL BADAWY† and
GRAHAM JULLIEN‡

*Advanced Technology Information Processing Systems (ATIPS),
2500 University Drive, NW, Calgary, AB, Canada T2N 1N4
*amer@atips.ca
†badawy@atips.ca
‡jullien@atips.ca*

MARCO MATTAVELLI

*Signal Processing Laboratory-3, Signal Processing Institute,
Ecole Polytechnique Fèdèrale de Lausanne (EPFL),
CH-1015 Lausanne, Switzerland
marco.mattavelli@epfl.ch*

ROBERT TURNEY

*DSP Systems Engineering, DSP Division, Xilinx Inc.,
115 South 4th Street, Watertown, WI 53094, USA
robert.turney@xilinx.com*

Current multimedia design processes suffer from the excessively large time spent on testing new IP-blocks with references based on large video encoders specifications (usually several thousands lines of code). The appropriate testing of a single IP-block may require the conversion of the overall encoder from software to hardware, which is difficult to complete in the short time required by the competition-driven reduced time-to-market demanded for the adoption of a new video coding standard. This paper presents a new design flow to accelerate the conformance testing of an IP-block using the H.264/AVC software reference model. An example block of the simplified 8 × 8 transformation and quantization, which is adopted in FRExt, is provided as a case study demonstrating the effectiveness of the approach.

Keywords: Design flow; H.264; advanced video coding; DCT; hardware; IP-block; VLSI; FPGA; transform; quantization; SystemC; rapid prototyping; platform; video coding.

1. Introduction

Digital video coding currently has a significant impact on the computer, telecommunications, and imaging industry. Especially with the remarkable progress in the development of products and services offering full-motion digital video transmitted on heterogeneous networks at different resolutions and on different terminals. This explains the reason for the existence and success of industry standards for compressed video representation achieving extremely high coding efficiency and enhanced robustness to different network environments.[1]

Open, interoperable international video coding standards have always been the enabler for the commercial success of digital video appliances. The ITU-T H.264/MPEG-4 (Part 10) advanced video coding (commonly referred as H.264/AVC) is the latest and most advanced member of the family of international video coding standards.[2]

Most of the H.264/AVC standard applications are based on software implementations. Nevertheless, hardware implementations are also desirable for consumer products since they provide consistent advantages in terms of compactness, low power, robustness, low costs, and, most importantly, real-time operation up to HDTV rates. In our previous work,[3–8] hardware implementations of different blocks in the initial H.264 transformation hierarchy model and entropy coding have been presented, while in Refs. 9 and 10, a design flow to accelerate the process of testing the quality of developed IP-blocks with the H.264 software reference model has been specified and described. In this paper, a high-performance hardware implementation of the simplified 8×8 transform and quantization module of the H.264/AVC standard is presented. The proposed design flow is used to assess the quality and the conformance with the standard within a reduced time window. The design flow and the hardware architecture block have been included in the draft of the second edition of the MPEG-4 Part 9 Reference Hardware Description.[11]

The paper is organized as follows. Section 2 provides an overview of the H.264/AVC video coding standard. Section 3 briefly describes the simplified 8×8 transformation and quantization that has been chosen to be implemented in hardware. In Sec. 4, a description of the stages of the proposed design flow is provided, followed by Sec. 5, where more details about the IP-block design features are described. Sections 6 and 7 show the methodology necessary to apply some of the steps in the design flow so that a speedup in the process of designing the IP-block and testing its conformance with the H.264/AVC reference software is achieved. Section 8 presents some simulation results and the achieved performance. Finally, in Sec. 9, considerations on the proposed design flow are discussed; and Sec. 10 concludes the paper.

2. Overview of the H.264/AVC Video Coding Standard

H.264/AVC was developed by the Joint Video Team (JVT), which was formed to represent a formal cooperation between the ITU-T Video Coding Experts Group (VCEG) and the ISO/IEC Moving Picture Experts Group (MPEG), working on the development of a new international standard that is appropriate for conversational and nonconversational audio/video applications.[12–14]

H.264/AVC presents many new video coding tools that make it the most powerful state-of-the-art standard as compared to the currently existing video coding families.[14] Network friendliness and compression performance, which has never been achieved before, at high and low bit rates, are the two most important features that distinguish H.264/AVC from other standards.[15–19]

Unlike other standards, H.264/AVC presents a transformation hierarchy using exactly integer arithmetic-based algorithms. Such specifications eliminate the possible mismatch issues between the encoder and the decoder that have been observed even when the 8×8 DCT/IDCT transform implementation is fully compliant with the IEEE 1180 recommendation providing the specifications for required accuracy for approximations of the floating point implementations.[15,20] In the initial H.264/AVC standard, which was completed in May 2003, the transformation was primarily in the form of 4×4 blocks, which helps reduce blocking and ringing artifacts.

Fidelity range extensions (FRExt, Amendment I), a new amendment that was added to the H.264/AVC standard in July 2004 is currently receiving wide attention in the industry. It actually demonstrates further coding efficiency against current video coding standards, potentially by as much as 3:1 for some key applications. The FRExt project produced a suite of some new profiles collectively called high profiles. In addition to supporting all features of the prior main profile, all the high profiles support an adaptive transform-block size and perceptual quantization scaling matrices.[14] The concept of adaptive transform-block size has proven to be an efficient coding tool within H.264/AVC video coding layer design.[21] This has led to the proposal of a seamless integration of a new 8×8 integer approximation of DCT (and prediction modes) into the specification with the least possible amount of technical and syntactical changes to give significant compression performance at standard definition (SD) and high definition (HD) resolutions.[22–24]

3. H.264 Simplified 8×8 Transform and Quantization

The use of block sizes smaller than 8×8 is limited at SD resolutions and higher. This has led to the proposal of an integer approximation of 8×8 DCT in FRExt to be added to the JVT specification.[24] This transform is applied to each block in the luminance component of the input video stream. It allows for bit-exact implementation for all encoders and decoders. Despite being more complex compared to the 4×4 DCT-like transform that is adopted by the initial H.264 specifications, the 8×8 DCT transform provides excellent compression performance when used for high-resolution video streams requiring a number of operations comparable to the number of operations required for the corresponding four 4×4 blocks using the fast butterfly implementation of the existing 4×4 transform.[22,23]

The 2D forward 8×8 integer transform is computed in a separable way as a 1D horizontal (row) transform followed by a 1D vertical (column) transform as shown in Eq. (1):

$$W = C_f X C_f^T \,. \tag{1}$$

The Matrix C_f is given by expression (2):

$$C_f = \begin{bmatrix} 8 & 8 & 8 & 8 & 8 & 8 & 8 & 8 \\ 12 & 10 & 6 & 3 & -3 & -6 & -10 & -12 \\ 8 & 4 & -4 & -8 & -8 & -4 & 4 & 8 \\ 10 & -3 & -12 & -6 & 6 & 12 & 3 & -10 \\ 8 & -8 & -8 & 8 & 8 & -8 & -8 & 8 \\ 6 & -12 & 3 & 10 & -10 & -3 & 12 & -6 \\ 4 & -8 & 8 & -4 & -4 & 8 & -8 & 4 \\ 3 & -6 & 10 & -12 & 12 & -10 & 6 & -3 \end{bmatrix} \cdot 1/8 \,. \tag{2}$$

Each of the 1D transforms is computed using three-stages fast butterfly operations, as shown in Table 1.[23]

As can be shown from the previous butterfly operations, the 2D transform operation can be implemented using signed additions and right-shifts only, avoiding expensive multiplication implementations. The post-scaling and quantization formulas are provided in Eqs. (3)–(5):

$$qbits = 15 + (QP\,\mathrm{DIV}\,6)\,, \tag{3}$$

$$|Z_{ij}| = SHR(|W_{ij}| \cdot MF + f, qbits + 1)\,, \tag{4}$$

$$\mathrm{Sign}(Z_{ij}) = \mathrm{Sign}(W_{ij})\,, \tag{5}$$

QP is a quantization parameter that determines the level of coarseness of the quantization process. It enables the encoder to accurately and flexibly control the trade-off between bit rate and quality. It takes an integer value that ranges from 0 up to 51 (with low values representing less quantization, hence, better quality of the reconstructed frame). Z_{ij} represents an element in the quantized transform coefficients

Table 1. 1D transform three-stages butterfly operations.

Stage 1	Stage 2	Stage 3
$a[0] = x[0] + x[7]$	$b[0] = a[0] + a[3]$	$w[0] = b[0] + b[1]$
$a[1] = x[1] + x[6]$	$b[1] = a[1] + a[2]$	$w[1] = b[2] + (b[3] \gg 1)$
$a[2] = x[2] + x[5]$	$b[2] = a[0] - a[3]$	$w[2] = b[0] - b[1]$
$a[3] = x[3] + x[4]$	$b[3] = a[1] - a[2]$	$w[3] = (b[2] \gg 1) - b[3]$
$a[4] = x[0] - x[7]$	$b[4] = a[5] + a[6] + ((a[4] \gg 1) + a[4])$	$w[4] = b[4] + (b[7] \gg 2)$
$a[5] = x[1] - x[6]$	$b[5] = a[4] - a[7] - ((a[6] \gg 1) + a[6])$	$w[5] = b[5] + (b[6] \gg 2)$
$a[6] = x[2] - x[5]$	$b[6] = a[4] + a[7] - ((a[5] \gg 1) + a[5])$	$w[6] = b[6] - (b[5] \gg 2)$
$a[7] = x[3] - x[4]$	$b[7] = a[5] - a[6] + ((a[7] \gg 1) + a[7])$	$w[7] = -b[7] + (b[4] \gg 2)$

Table 2. Multiplication factor (MF) for the prototyped architecture.

M	$(i,j) \in G_0$	$(i,j) \in G_1$	$(i,j) \in G_2$	$(i,j) \in G_3$	$(i,j) \in G_4$	$(i,j) \in G_5$
0	13107	11428	20972	12222	16777	15481
1	11916	10826	19174	11058	14980	14290
2	10082	8943	15978	9675	12710	11985
3	9362	8228	14913	8931	11984	11295
4	8192	7346	13159	7740	10486	9777
5	7282	6428	11570	6830	9118	8640

*G_0: $i = [0,4]$, $j = [0,4]$; G_1: $i = [1,3,5,7]$, $j = [1,3,5,7]$; G_2: $i = [2,6]$, $j = [2,6]$; G_3: $(i = [0,4],\ j = [1,3,5,7]) \cap (i = [1,3,5,7],\ j = [0,4])$; G_4: $(i = [0,4],\ j = [2,6]) \cap (i = [2,6],\ j = [0,4])$; G_5: $(i = [2,6],\ j = [1,3,5,7]) \cap (i = [1,3,5,7],\ j = [2,6])$.

matrix. MF is a multiplication factor that depends on $(m = QP \bmod 6)$ and the position (i,j) of the element in the matrix, as shown in Table 2. $SHR(\,)$ is a procedure that right-shifts the result of its first argument a number of bits equal to its second argument. f is defined in the software reference model as $2^{qbits}/3$ for intra-blocks and $2^{qbits}/6$ for inter-blocks.[12,13]

4. Design Flow of an H.264 HW/SW Video Encoder

This section provides a general description of the complete process used to design the proposed IP-block, starting with a HW/SW partitioning of the reference software, passing through functional verification, and ending with physical implementation on a Xilinx Virtex II FPGA. Figure 1 gives the flow chart of the complete design flow.[9]

First, an extensive analysis to validate the choice of converting the integer 8 × 8 DCT transform from software to hardware is performed. Then, a description of the hardware design to be realized, followed by the functional verification using SystemC for the formalism and simulation environment. After the functional verification, simulation at the RTL level of abstraction is performed using the University of Calgary Rapid Prototyping Platform (UCRPP).[25,26] Then, the process ends with system synthesis followed by place and rout, which is a vendor-dependent step, and the physical programming of the FPGA itself.

5. Hardware Prototyping of the IP-Block

Figure 2 shows a block diagram of the developed IP-block architecture. The IP accepts the following inputs: 8 × 8 parallel blocks, QP, a synchronizing clock, and an enabling signal (input valid). It outputs the quantized transform coefficients matrix and the signal (output valid).

The architecture is designed to perform pipelined operations. This drastically reduces the required memory resources and accesses, avoids any stall states, and

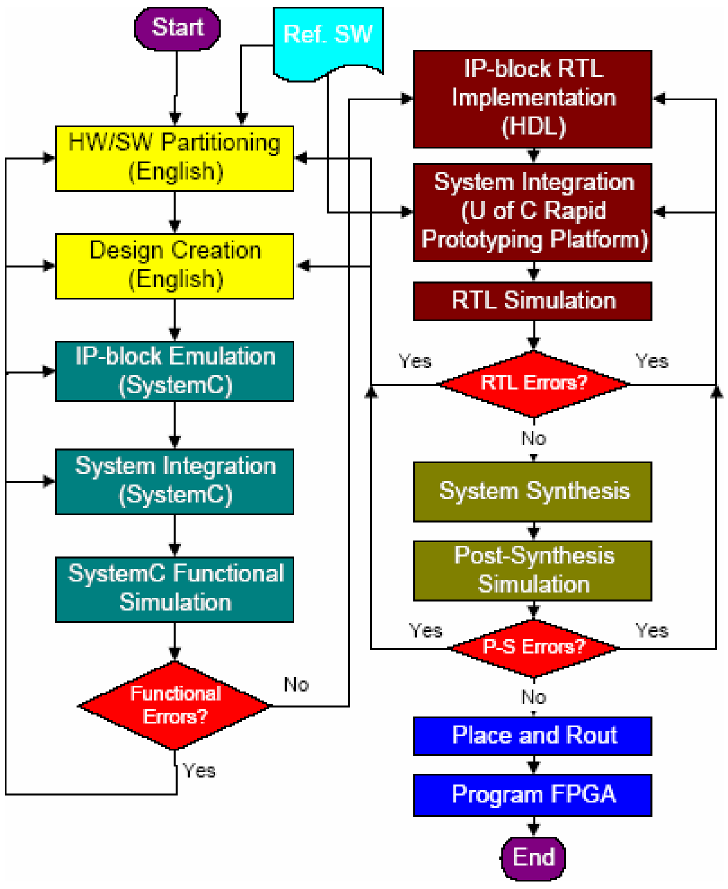

Start
Ref. SW
HW/SW Partitioning (English)
Design Creation (English)
IP-block Emulation (SystemC)
System Integration (SystemC)
SystemC Functional Simulation
Functional Errors?
No
Yes
IP-block RTL Implementation (HDL)
System Integration (U of C Rapid Prototyping Platform)
RTL Simulation
Yes
RTL Errors?
Yes
No
System Synthesis
Post-Synthesis Simulation
Yes
P-S Errors?
Yes
No
Place and Rout
Program FPGA
End

Fig. 1. A flowchart of the complete design flow.

dramatically improves the throughput of the architecture. Figure 3 provides a detailed block diagram of the architecture showing the data flow between the components.

The architecture consists of two main stages. The first one contains two blocks; the “Transform” block, which is composed of three stages of the fast butterfly operations mentioned in Sec. 2, repeated twice (for horizontal and vertical transform), and the “QP-processing” block, which is responsible for calculating the intermediate parameters needed for quantization, such as f, $qbits$, and $(P_0 - P_5)$, which are the values of the multiplication factors at the six different groups of positions in the matrix, as shown in Table 2. Finally, the quantization process takes place in the second main stage of the design. This is done by performing the addition and multiplication operations in the arithmetic block, and finally the shifting operations in the shifter block.

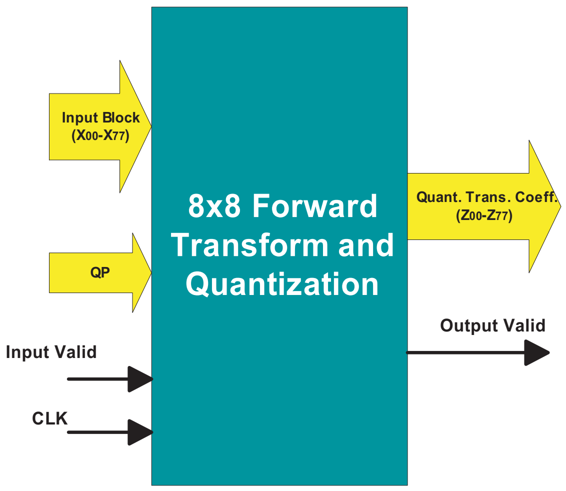


Fig. 2. A block diagram of the proposed hardware architecture.

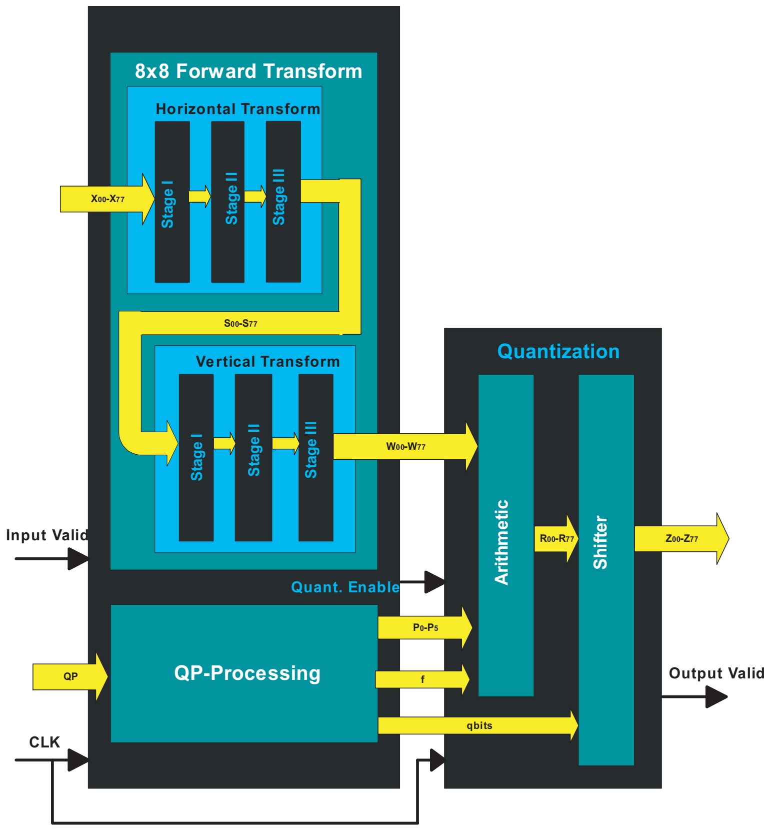


Fig. 3. A detailed block diagram of the proposed hardware architecture.

6. Functional Verification Using SystemC

Design productivity is the key to reduced time-to-market. It is an essential element that should be considered when releasing a new design. Hence, early functional verification is one fundamental step for successful IP providers to avoid prolonged products development phases. This is what gives special importance to the SystemC verification step in the design flow, although a direct transition from abstract design blocks to HDL description blocks is also possible.

SystemC is a hardware design concept that enables the designer to perform early functional verification of developed hardware blocks by facilitating their integration with software in a unified platform. It provides hardware-oriented constructs within the context of C++ as a class library implemented in standard C++. This facilitates the process of integrating SystemC-emulated hardware blocks with any software reference model as they are all originated from the same environment.

In Ref. 10, hardware definition switches enabling concurrent development and testing of different hardware blocks (represented in SystemC) with the JM H.264 software reference model is described. This approach facilitates the development process of hardware blocks during the software stabilization phase. One of the major goals was to reduce as much as possible the modifications required to the code for the inclusion of the corresponding HW described blocks. Figures 4 and 5 provide examples of the modifications that have been introduced to the code in order to embed the described HW block presented in the previous section.[10]

By using this approach, it has been possible to perform behavioral simulations to the DCT block, showing that it is functionally compliant with the reference software.

7. The Rapid Prototyping Platform

A PCMCIA prototyping FPGA card, shown in Fig. 6, has been chosen as the platform to integrate the HDL implementation of the IP-block with the reference software.[25] It connects the FPGA HW with a portable host computer through the PCI bus by plugging it into a standard PCMCIA socket, as shown in Fig. 7.[25]

```
//The "#define" entries for the hardware accelerator
#define DCT_8x8_HW_ACCELERATOR 1

//notice that
//'0' is a value for pure software implementation
//'1' is a value for a SystemC block

//The prototype of the SW function
void sw_dct_8x8(int[ ][MB_BLOCK_SIZE], int[ ][8]);
```

Fig. 4. Examples of the required modifications to the JM file (global.h).

```
//To initiate the hardware module only once
int firstHW_Call = 1;

//Grouping the code in a modular form to facilitate
//HW/SW switching (e.g. 8x8 DCT Function)
void sw_dct(int m[ ][MB_BLOCK_SIZE], int n[ ][8]){
            .
            .
}

//Hardware/Software Switch (e.g. 8x8 DCT Calling)
 if (!lossless_qpprime)
    if(DCT_8x8_HW_ACCELERATOR){
                      sc_dct_8x8(img->m7,   firstHW_Call);
                                     firstHW_Call     =     0;
    }
    else{
       sw_dct(img->m7, m6);
    }
```

Fig. 5. Examples of the required modifications to the JM file (block.c).

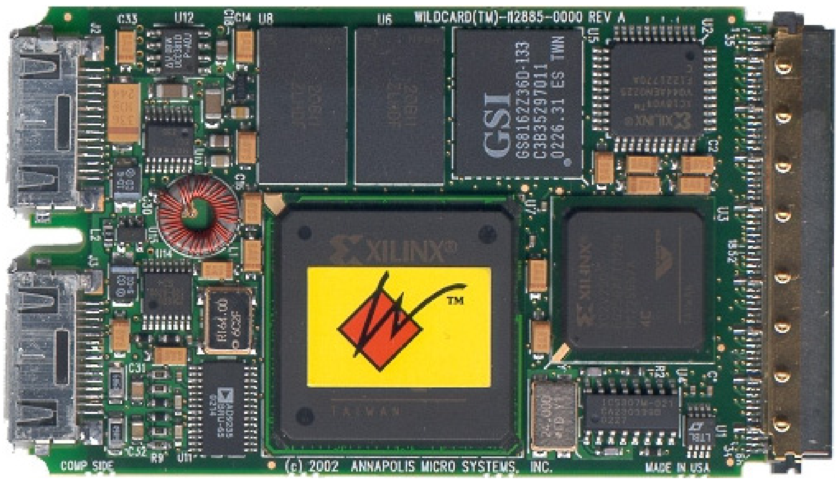

WILDCARD(TM)-112885-0000 REV A
XILINX
(c) 2002 ANNAPOLIS MICRO SYSTEMS, INC.
MADE IN USA
COMP SIDE

Fig. 6. Wildcard II card acting as the prototyping platform.

Some of the components of the prototyping platform are located in the host system, while others are in the pluggable FPGA card.[26] The host side is the main general purpose processor and support chips for a standard PCI bus. Its main storage holds the FPGA device configuration files and the software part of the system (i.e., the H.264 software reference model). Optionally, the host direct memory access (DMA) and interrupt controller can be parts of the design, targeting an increased system performance. The data communication bottlenecks between the hosting computer and the card can be minimized by the use of direct memory access

Fig. 7. Wildcard II connects with a portable host computer through the PCI bus.

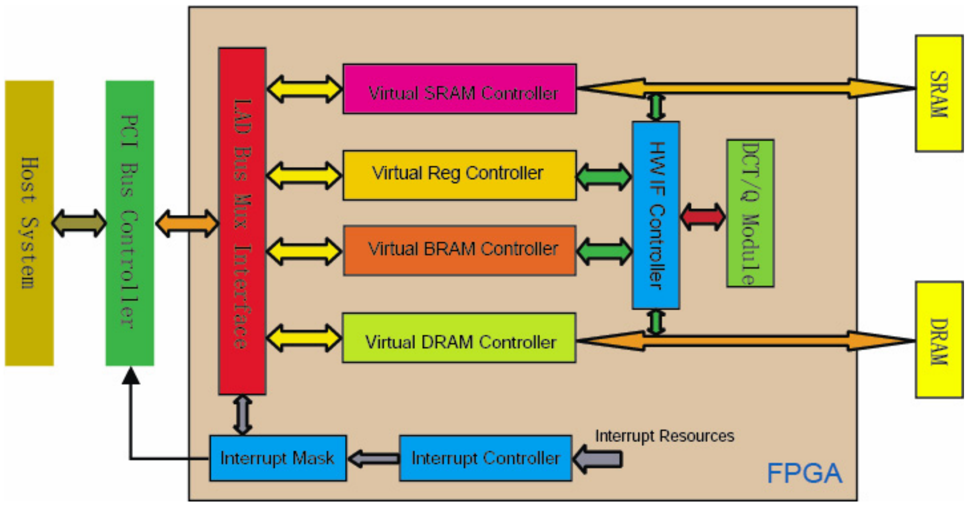


Fig. 8. A block diagram of the DCT/Q module integration using the rapid prototyping platform.

and interrupt-based control. On the other hand, the FPGA card hosts the FPGA chip, programmable clock generator, and a PCI-bus interface controller. A block diagram of the DCT/Q module integration using the rapid prototyping platform is shown in Fig. 8.[27]

The process of integrating the hardware block in the platform is performed through an improved system IP hardware interface controller, which provides an easy way for the IP-block to exchange data between the host system and the memory space on the WildCard II. The controller has a set of interface signals with the memory spaces to deal with the memory operations required by the IP-block. Data are passed from software to hardware through simple virtual socket API function calls.[27]

8. Simulations and Results

In our experiments, the design flow introduced in Sec. 3 was used to accelerate the development process of the IP-block introduced in Sec. 4. We first embedded the SystemC-emulated architecture to the JM FRExt 2.2 project, and compared the output stream (with HW switch set to "1") with the output from the original software (HW switch set to "0"). Figure 9 reports a visual comparison between the output video streams before and after embedding the SystemC blocks.

In addition to the visual comparison, we performed a statistical comparison by comparing the PSNR of the luminance and chrominance components of the reconstructed video sequences with HW switch set to "0", and the corresponding values with HW switch set to "1". We were able to show that the results in both cases are identical, except for the required simulation time, which, as expected, increases when the emulated SystemC block is used due to the overhead of the hardware emulation stage. Figures 10 and 11[10] provide a summary of results reported by JM FRExt 2.2 before and after embedding the 8 × 8 DCT SystemC block.

Then, the UCRPP enables us to test the block starting from the RTL level of abstraction. Simulation was performed using the Mentor Graphics© ModelSim 5.4® simulation tool, and synthesized using Synplify Pro 7.1® from Synplicity©. The target technology is the FPGA device XC2V4000 (BF957 package) from the Virtex-II family of Xilinx©. Table 3 summarizes the performance of the prototyped architecture.[8]

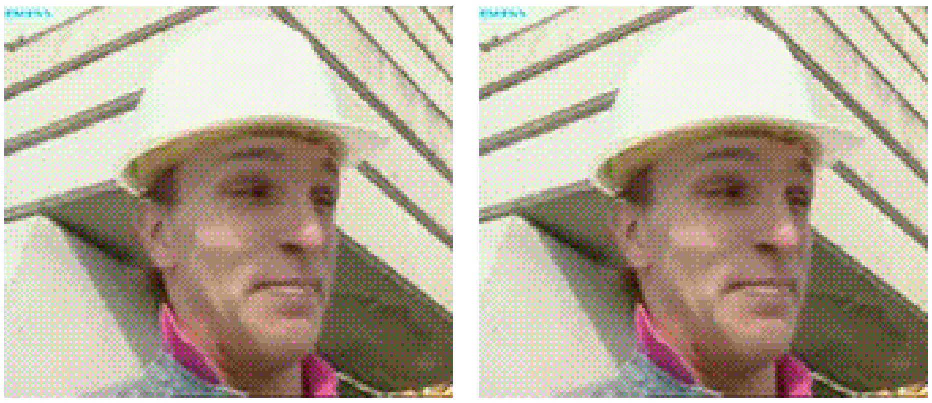
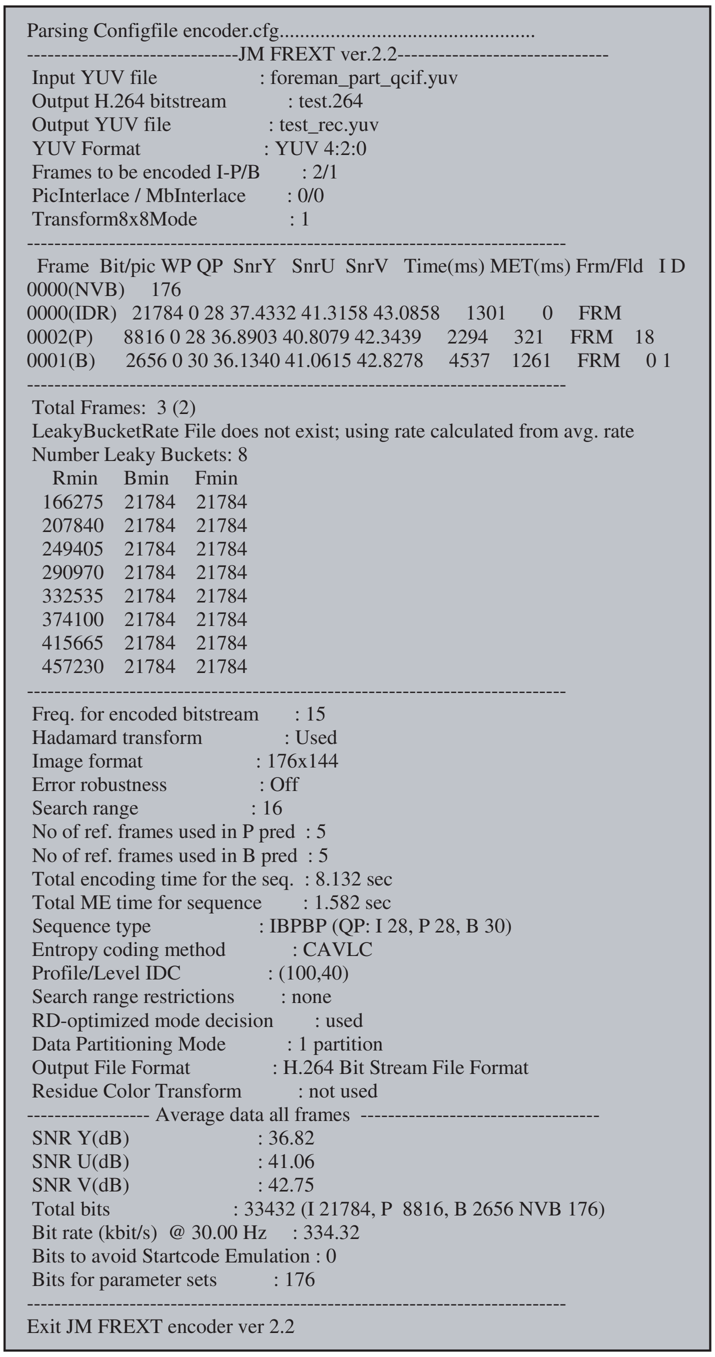
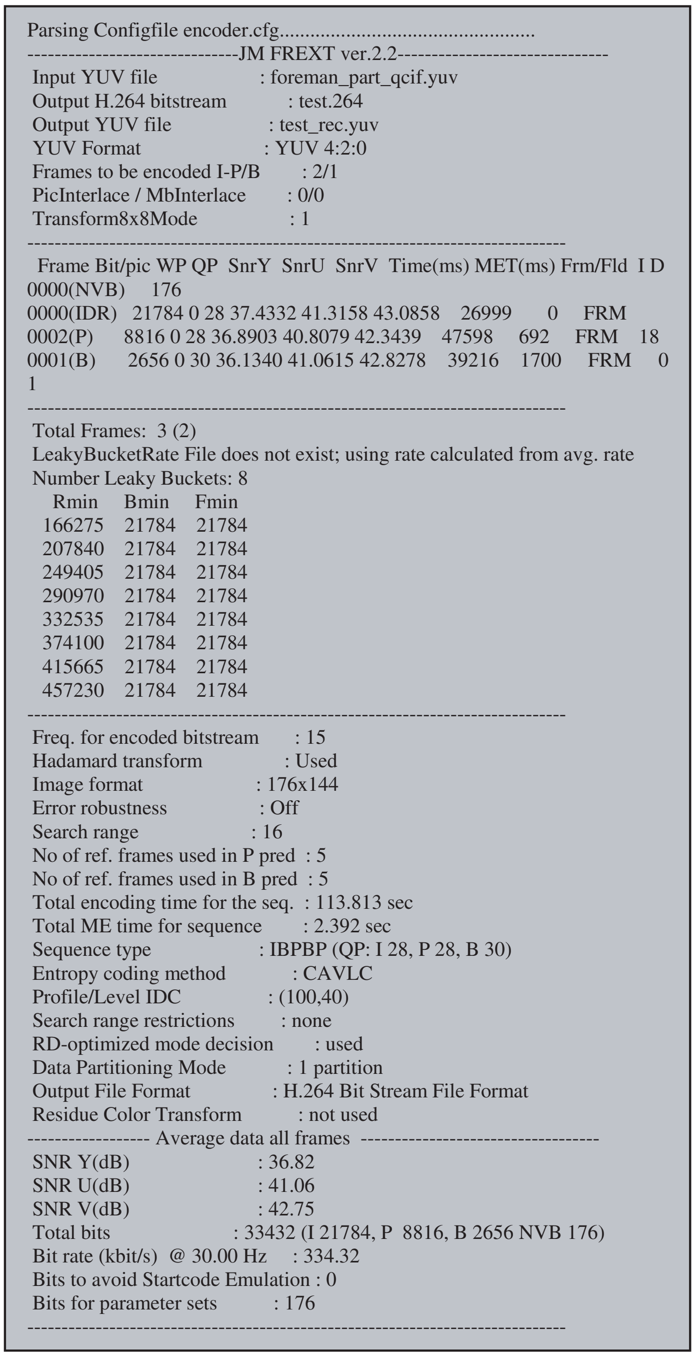

(a) (b)

Fig. 9. Output video stream (a) before and (b) after embedding the SystemC block.



```
Parsing Configfile encoder.cfg..................................................
------------------------------JM FREXT ver.2.2------------------------------
 Input YUV file                  : foreman_part_qcif.yuv
 Output H.264 bitstream           : test.264
 Output YUV file                 : test_rec.yuv
 YUV Format                      : YUV 4:2:0
 Frames to be encoded I-P/B        : 2/1
 PicInterlace / MbInterlace        : 0/0
 Transform8x8Mode                 : 1
-----------------------------------------------------------------------------
  Frame  Bit/pic WP QP  SnrY   SnrU  SnrV   Time(ms) MET(ms) Frm/Fld   I D
0000(NVB)     176
0000(IDR)   21784 0 28 37.4332 41.3158 43.0858     1301       0     FRM
0002(P)      8816 0 28 36.8903 40.8079 42.3439     2294     321     FRM    18
0001(B)      2656 0 30 36.1340 41.0615 42.8278     4537    1261     FRM     0 1
-----------------------------------------------------------------------------
 Total Frames:  3 (2)
 LeakyBucketRate File does not exist; using rate calculated from avg. rate
 Number Leaky Buckets: 8
    Rmin     Bmin     Fmin
   166275    21784    21784
   207840    21784    21784
   249405    21784    21784
   290970    21784    21784
   332535    21784    21784
   374100    21784    21784
   415665    21784    21784
   457230    21784    21784
-----------------------------------------------------------------------------
 Freq. for encoded bitstream       : 15
 Hadamard transform               : Used
 Image format                    : 176x144
 Error robustness                 : Off
 Search range                    : 16
 No of ref. frames used in P pred  : 5
 No of ref. frames used in B pred  : 5
 Total encoding time for the seq.  : 8.132 sec
 Total ME time for sequence        : 1.582 sec
 Sequence type                    : IBPBP (QP: I 28, P 28, B 30)
 Entropy coding method             : CAVLC
 Profile/Level IDC                : (100,40)
 Search range restrictions         : none
 RD-optimized mode decision        : used
 Data Partitioning Mode            : 1 partition
 Output File Format               : H.264 Bit Stream File Format
 Residue Color Transform           : not used
------------------ Average data all frames  ----------------------------------
 SNR Y(dB)                       : 36.82
 SNR U(dB)                       : 41.06
 SNR V(dB)                       : 42.75
 Total bits                    : 33432 (I 21784, P  8816, B 2656 NVB 176)
 Bit rate (kbit/s)  @ 30.00 Hz     : 334.32
 Bits to avoid Startcode Emulation : 0
 Bits for parameter sets           : 176
-----------------------------------------------------------------------------
Exit JM FREXT encoder ver 2.2
```

Fig. 10. Summary of results reported by JM FRExt 2.2 before embedding the 8×8 DCT SystemC block.



```
Parsing Configfile encoder.cfg..................................................
-------------------------------JM FREXT ver.2.2-------------------------------
 Input YUV file                    : foreman_part_qcif.yuv
 Output H.264 bitstream            : test.264
 Output YUV file                   : test_rec.yuv
 YUV Format                        : YUV 4:2:0
 Frames to be encoded I-P/B        : 2/1
 PicInterlace / MbInterlace        : 0/0
 Transform8x8Mode                  : 1
-------------------------------------------------------------------------------
  Frame Bit/pic WP QP  SnrY  SnrU  SnrV  Time(ms) MET(ms) Frm/Fld  I D
0000(NVB)     176
0000(IDR)   21784 0 28 37.4332 41.3158 43.0858    26999       0     FRM
0002(P)      8816 0 28 36.8903 40.8079 42.3439    47598     692     FRM    18
0001(B)      2656 0 30 36.1340 41.0615 42.8278    39216    1700     FRM     0
1
-------------------------------------------------------------------------------
 Total Frames:  3 (2)
 LeakyBucketRate File does not exist; using rate calculated from avg. rate
 Number Leaky Buckets: 8
     Rmin     Bmin     Fmin
   166275    21784    21784
   207840    21784    21784
   249405    21784    21784
   290970    21784    21784
   332535    21784    21784
   374100    21784    21784
   415665    21784    21784
   457230    21784    21784
-------------------------------------------------------------------------------
 Freq. for encoded bitstream       : 15
 Hadamard transform                : Used
 Image format                      : 176x144
 Error robustness                  : Off
 Search range                      : 16
 No of ref. frames used in P pred  : 5
 No of ref. frames used in B pred  : 5
 Total encoding time for the seq.  : 113.813 sec
 Total ME time for sequence        : 2.392 sec
 Sequence type                     : IBPBP (QP: I 28, P 28, B 30)
 Entropy coding method             : CAVLC
 Profile/Level IDC                 : (100,40)
 Search range restrictions         : none
 RD-optimized mode decision        : used
 Data Partitioning Mode            : 1 partition
 Output File Format                : H.264 Bit Stream File Format
 Residue Color Transform           : not used
------------------ Average data all frames  -----------------------------------
 SNR Y(dB)                         : 36.82
 SNR U(dB)                         : 41.06
 SNR V(dB)                         : 42.75
 Total bits                        : 33432 (I 21784, P  8816, B 2656 NVB 176)
 Bit rate (kbit/s)  @ 30.00 Hz     : 334.32
 Bits to avoid Startcode Emulation : 0
 Bits for parameter sets           : 176
-------------------------------------------------------------------------------
```

Fig. 11. Summary of results reported by JM FRExt 2.2 after embedding the 8 × 8 DCT SystemC block.

Table 3. Performance of the prototyped architecture.

Critical path (ns)	CLK freq. (MHz)	# of i/p buffers	# of o/p buffers
14.598	68.5	583	1217
# of I/O reg. bits	# of reg. bits not inc. (I/O)	Total # of LUT	# of clock buffers
1219	16893	29018	1

A 14.598 ns critical path is estimated by the synthesis tool. Since the architecture outputs a complete 8×8 encoded block with each clock pulse at steady state, then the time required to encode a whole SD frame of 704×480 pixels can be calculated as follows[8]:

$$\begin{aligned}
&\text{Time required per CIF frame}\\
&\quad = \text{Time required per block} \times \text{Number of blocks per frame}\\
&\quad = 14.598\,\text{ns} \times (\#\text{pixels per frame})/(\#\text{pixels per block})\\
&\quad = 14.598\,\text{ns} \times (704 \times 480)\text{ppf}/(8 \times 8)\text{ppb}\\
&\quad \approx 77.1\,\mu\text{s}.
\end{aligned}$$

This value is about 216 times faster than the 16.67 ms time required for continuous motion (assuming a refresh rate of 60 frames/s). Similarly, the time required to encode a complete high definition television (HDTV) frame of a 720×1280 pixels resolution, and a 60 frames/s frame rate is 0.21 ms, which is about 79 times faster than the 16.6 ms time required for continuous motion. Hence, the architecture presented in this paper easily satisfies the real-time constraints for SD, HD, and even higher resolution video formats.[8]

9. Comments on the Proposed Design Flow

In this paper, a high-performance IP-block of the simplified 8×8 transformation and quantization, which was recently adopted by the H.264/AVC standard, is developed. The architecture was shown to satisfy the real-time constraints required by different high-resolution digital video applications. A novel design flow that facilitates early functional verification of developed IP-blocks, by assessing the conformance to the reference specification as well as the overall system behavior before physical implementation, was used. The methodology of the design flow accelerated the process of testing the quality of the IP-block by using the H.264/AVC software reference model itself. The presented design flow methodology can obviously be applied to the development of any HW-block, thus constituting an integrated framework for the transformation of pure SW specifications into a real hybrid SW/HW implementations. The design flow tools, the support platform, and a library of IP-blocks are in the process of being included in the ISO/IEC MPEG-4 Part 9 Technical Report called (Reference Hardware Description).

Acknowledgments

The authors would like to thank the Advanced Technology Information Processing Systems (ATIPS) Laboratory, the Alberta Informatics Circle of Research Excellence (iCORE), the Alberta Ingenuity Fund (AIF), the Natural Sciences and Engineering Research Council of Canada (NSERC), CMC Microsystems, Micronet R&D, Canada Foundation for Innovation (CFI), and the Department of Electrical and Computer Engineering in the Schulich School of Engineering at the University of Calgary for supporting this research.

References

1. A. M. Tekalp, *Digital Video Processing* (Prentice-Hall, Inc., New Jersey, USA, 1995).
2. ITU-T Rec. H.264 | ISO/IEC 14496-10 AVC, Draft text of final draft international standard for advanced video coding, http://www.chiariglione.org/mpeg/working_documents.htm, March 2003.
3. I. Amer, W. Badawy and G. Jullien, A proposed hardware reference model for spatial transformation and quantization in H.264, to appear in *J. Visual Commun. Image Rep.* (Special Issue on Emerging H.264/AVC Video Coding Standard).
4. I. Amer, W. Badawy and G. Jullien, Hadamard transform in H.264/MPEG-4 part 10: a hardware prototype, *Proc. Int. Computer Engineering Conf.*, Cairo, Egypt, December 2004.
5. I. Amer, W. Badawy and G. Jullien, Towards MPEG-4 part 10 system on chip: A VLSI prototype for context-based adaptive variable length coding (CAVLC), *Proc. IEEE Workshop on Signal Processing Systems*, Austin, Texas, USA, October 2004, pp. 276–279.
6. I. Amer, W. Badawy and G. Jullien, A VLSI prototype for Hadamard transform with application to MPEG-4 part 10, *Proc. IEEE Int. Conf. Multimedia and Expo*, Vol. 3, Taipei, Taiwan, June 2004, pp. 1523–1526.
7. I. Amer, W. Badawy and G. Jullien, Hardware prototyping for the H.264 4×4 transformation, *Proc. IEEE Int. Conf. Acoustics, Speech, and Signal Processing*, Vol. 5, Montreal, Canada, May 2004, pp. 77–80.
8. I. Amer, W. Badawy and G. Jullien, A high performance hardware implementation of the H.264 simplified 8×8 transformation and quantization, *Proc. IEEE Int. Conf. Acoustics, Speech, and Signal Processing*, Vol. 2, Pennsylvania, Philadelphia, USA, March 2005, pp. 1137–1140.
9. I. Amer, W. Badawy and G. Jullien, A design flow for an H.264 embedded video encoder, *Proc. Int. Conf. Information and Communication Technology*, Cairo, Egypt, December 2005, pp. 505–513.
10. I. Amer, M. Sayed, W. Badawy and G. Jullien, On the way to an H.264 HW/SW reference model: A SystemC modeling strategy to integrate selected IP-blocks with the H.264 software reference model, *Proc. IEEE Workshop on Signal Processing Systems*, Athens, Greece, November 2005.
11. ISO/IEC TR 14496-9, Information technology — Coding of audio-visual objects — part 9 reference hardware description, 2nd edn., July 2005.
12. I. E. G. Richardson, H.264/MPEG-4 Part 10: Transform and quantization, a white paper, http://www.vcodex.com, March 2003.
13. I. E. G. Richardson, *H.264 and MPEG-4 Video Compression: Video Coding for Next-Generation Multimedia* (Wiley, Sussex, England, 2003).

14. G. Sullivan, P. Topiwala and A. Luthra, The H.264 advanced video coding standard: Overview and introduction to the fidelity range extensions, *SPIE Conf. Application of Digital Image Processing XXVII*, Colorado, USA, August 2004.
15. Emerging H.264 standard: Overview and TMS320DM642-based solutions for real-time video applications, a white paper, http://www.ubvideo.com, December 2002.
16. T. Wiegand, G. J. Sullivan, G. Bjontegaard and A. Luthra, Overview of the H.264/AVC video coding standard, *IEEE Trans. Circuits Syst. Video Technol.* **13** (2003) 560–576.
17. K. Denolf, C. Blanch, G. Lafruit and A. Bormans, Initial memory complexity analysis of the AVC codec, *IEEE Workshop on Signal Processing Systems, 2002 (SIPS'02)*, October 2002, pp. 222–227.
18. T. Stockhammer, M. M. Hannuksela and T. Wiegand, H.264/AVC in wireless environments, *IEEE Trans. Circuits and Syst. Video Technol.* **13** (2003) 657–673.
19. M. Horowitz, A. Joch, F. Kossentini and A. Hallapuro, H.264/AVC baseline profile decoder complexity analysis, *IEEE Trans. Circuits Syst. Video Technol.* **13** (2003) 704–716.
20. H. S. Malvar, A. Hallapuro, M. Karczewicz and L. Kerofsky, Low-complexity transform and quantization in H.264/AVC, *IEEE Trans. Circuits Syst. Video Technol.* **13** (2003) 598–603.
21. M. Wien, Clean-up and improved design consistency for ABT, Joint Video Team (JVT) of ISO/IEC MPEG and ITU-T VCEG, doc. JVT–E025.
22. S. Gordon, D. Marpe and T. Wiegand, Simplified use of 8×8 transform — Proposal, Joint Video Team (JVT) of ISO/IEC MPEG and ITU-T VCEG, doc. JVT–J029.
23. S. Gordon, D. Marpe and T. Wiegand, Simplified use of 8×8 transform — Updated proposal and results, Joint Video Team (JVT) of ISO/IEC MPEG and ITU-T VCEG, doc. JVT–K028, Munich, Germany, March 2004.
24. S. Gordon, Simplified use of 8×8 transform, Joint Video Team (JVT) of ISO/IEC MPEG and ITU-T VCEG, doc. JVT–I022, San Diego, USA, September 2003.
25. I. Amer, C. A. Rahman, T. Mohamed, M. Sayed and W. Badawy, A hardware-accelerated framework and IP-blocks for application in MPEG-4, *Proc. IEEE Int. Workshop on System on Chip*, Banff, Alberta, Canada, July 2005, pp. 211–214.
26. T. Mohamed and W. Badawy, Integrated hardware–software platform for image processing applications, *Proc. IEEE Int. Workshop on System on Chip*, Banff, Alberta, Canada, July 2004, pp. 145–148.
27. Y. Qiu and W. Badawy, An integration of the MPEG-4 part 9 reference hardware into the virtual socket co-design platform, ISO/IEC JTC1/SC29/WG11 M12794, Bangkok, Thailand, January 2006.
28. H. Huber, SOC for evolving standards: A new challenge for rapid prototyping using FPGAs?, a white paper, http://www.lsilogic.com.