Abstract—For most integrated circuit (IC) design houses, building a dedicated field-programmable gate array (FPGA) verification board before the chip is tapped out is usually a requirement. The time and money spent on such boards are usually high. A versatile verification/fast prototyping platform consisting of an FPGA board and the associated System Software is presented. The FPGA board is connected to the host computer through a peripheral component interconnect interface. The System Software running on a Microsoft Windows environment is developed so that all the real-time data that were generated during the verification process can be downloaded and displayed on the host computer. Hence, the system acts like a logic analyzer with very large storage memory. Furthermore, the software provides interfaces for personal computer peripherals, such as a Universal Serial Bus camera, a computer monitor, and a soundcard, if the test data for multimedia applications are necessary. The results can be also demonstrated in real time through these devices. Because multimedia data are usually complicated, and the size is huge, the design and thorough verification are usually difficult and take lots of time. The proposed system is both versatile and convenient to use, and verification of designs becomes much easier. An MPEG-4 Simple Profile Video Encoder and a 3-D surround sound processor were realized with the proposed system. Both cases can run in real time. In this paper, the highly complicated H.264/Advanced Video Coding I-Frame encoder is also used to demonstrate the proposed system.

Index Terms—Field-programmable gate array (FPGA), H.264/Advanced Video Coding (AVC), intellectual properties (IPs), logic analyzer, multimedia application-specified integrated circuit (ASIC), prototyping, system-on-a-chip (SoC), verification.

I. INTRODUCTION

SYSTEM-ON-A-CHIP (SoC) design has become an integrated circuit (IC) design trend. With its growing complexity, problems such as hardware/software code design and system-level verification become more and more complicated. One of the methods to reduce the design difficulty is the block-based design with reusable intellectual properties (IPs). Verifications of every IP and its integrated results are very important and time consuming when the system is complicated. A common experience shows that the time spent on the verification process is usually much longer than the time spent on the design phase. Thus, how to shorten the verification period becomes an important issue. Dedicated field-programmable gate array (FPGA) based verification boards are usually used in such verification processes; however, building such boards is also time consuming and costs lots of money. Furthermore, many designs involving one or more embedded processors require software–hardware coverification, and simple FPGA boards are not up to this challenging task.

Recently, multimedia applications have become more popular. Design and verification of a multimedia application-specified integrated circuit such as codec and media processors are even more difficult today because, to achieve better performance, the algorithms are more complicated, multimedia data are usually complex, and the size is huge. For example, the data of the digital video signal to be processed are usually over 10–100 MB/s. It is very difficult to test over a wide range of data to assure that the design is good, and therefore, standard test sequences are mostly used. Once there is an error, it is also hard to trace the problems. Except for a logic analyzer, most designs include testing circuits or provide boundary scan chains to help debugging; however, limitations do exist in such approaches. For example, if an error happens 30 min after the system starts, it is extremely time consuming to trace such errors if traditional methods are used.

In addition, verification on a dedicated FPGA-based system is usually required before the design is tapped out. The effort to build such a system is no less than that of designing the chip. Verification of the system usually relies on tools such as logic analyzers, which have quite a few limitations and are rather inconvenient to use. The life cycles of new products become shorter; therefore, fast prototyping is crucial. Based on the above reasons, more sophisticated verification/prototyping tools are necessary.

Most FPGA chip suppliers such as Xilinx and Altera and their third-party providers offer peripheral component interconnect (PCI) based evaluation boards, so that one can port their designs on the FPGA chip and use the host computer to provide and receive data for the evaluation board [1], [2]. However, there are no convenient interfaces for users to observe the results. For example, a visualization tool, such as SignalScan from Cadence, is very useful for very large scale integration (VLSI) designers. To use an FPGA board as a verification platform, some tools are necessary to make it convenient and powerful. This proposed system is divided into four parts: 1) the PCI/FPGA-based verification board; 2) the Windows-based System Software; 3) the multimedia interface; and 4) the Tunnel Application Programming Interfaces (APIs). The verification board uses Altera Cyclone FPGA chips. The entire
software part, which runs under the MS Windows environment, includes a PCI driver, a logic analyzer tool with waveform display/editor, a software tool for data communication between the host computer and the board, an application tool for interfacing to the PC monitor and soundcard, and user APIs for communication between the proposed System Software and the user’s programs.

We implemented an MPEG-4 Video Encoder and other designs on this system [3]. Currently, we use it to verify our design of an H.264/Advanced Video Coding (AVC) I-Frame encoder. H.264/AVC is a very sophisticated coding standard and is known as the next-generation high-performance video codec [4]. It was reported that the H.264/AVC I-Frame encoder has better performance than most current still-image compression standards such as JPEG and JPEG2000 [7], [8]. One approach to implement it is to combine a microprocessor and some dedicated hardware circuits; therefore, we shall use the host computer to play the role of the microprocessor and port the circuits on the FPGA board.

In Section II, architectures of the PCI/FPGA-based verification board and the Windows-based System Software are presented. Some schemes of using Tunnel APIs are described in Section III. In Section IV, the H.264/AVC I-Frame encoder architecture is briefly reviewed. The system demonstration is in Section V. Conclusion and future works are described in Section VI.

II. SYSTEM ARCHITECTURE OF THE VERIFICATION/FAST PROTOTYPING PLATFORM

The proposed system can be divided into four parts: 1) the PCI/FPGA-based verification board; 2) the Windows-based System Software; 3) the host computer/FPGA board interfacing protocol; and 4) the Tunnel APIs. They are described in the following subsections.

A. PCI/FPGA-Based Verification Board

The verification board mainly consists of a System Control chip, an Altera Cyclone user FPGA, an FPGA download circuit, and expansion card slots. A PCI target module and some control modules are contained in the System Control chip. The PCI target module to cope with the PCI protocol is designed to take care of the communication between the circuit in the FPGA and the host computer. All control signals and data between both sides have to be transmitted through this module. It separates the input signals/data to the user’s design from PCI signals and encapsulates the feedback data to the host computer so that they can be transferred through the standard PCI interface. To make our system more reliable and efficient, PCI burst mode transmission is employed. Thus, all the signals, including control signals to the user’s circuit in the FPGA to and from the host computer, have to be latched and processed before they are sent. Because the clock signal is provided from the PCI interface to the user FPGA, this creates problems if the circuit has to be held during the data transfer cycles. Therefore, a clock control module is designed to control the input clock signal to the FPGA to prevent errors. Furthermore, it is possible to test the user’s circuit step by step, like a software debugger that is used in the development of a microprocessor-based system. This feature has to be initiated and controlled by the System Software in the host computer. An FPGA and its download circuit are provided for the user to download and execute his/her designs. The user’s designs are downloaded into the FPGA by Joint Test Action Group (JTAG) [6]. To make the platform more flexible, the user can develop his/her expansion card and plug into the slot. Modules such as analog-to-digital converters, digital-to-analog converters, and synchronous dynamic random access memory (SDRAM) are a few good examples. The basic architecture is shown in Fig. 1. Fig. 2 is the snapshot of the verification board.

B. Windows-Based System Software

The design of the basic System Software is to provide control to the verification board and the interface software. It also acts as the data center when it is necessary to transfer data among different parts of the system. However, real-time data transfer may not be possible due to the following facts. First,
MS Windows is not a real-time operating system. Second, the circuit might be too complicated to run in real time on the FPGA. Third, the amount of data might be too huge. There are also some useful software tools that are similar to the ones seen in QuartusII and SignalScan. Fig. 3 depicts the architecture of the System Software. Data for verification process can be divided into two categories—data to be used as the circuit input and data to be compared with the circuit output. This is similar to the common postsimulation running on workstation tools, except that ours is much faster. The ASCII hexadecimal format is the only format supported in this system. The data can be also created by using the WaveForm Editor, which is included in the software. Users have to configure the pin connection between their circuit and system control module by using the configuration module, which will help the administration module analyze data and help the WaveForm player to display and the WaveForm Editor to edit input signals. The WaveForm player and the configuration module are shown in Figs. 4 and 5, respectively. Other system parameters such as the video frame rate, the video size, the audio sample rate, and the audio channel numbers, can be also set up with the configuration module. The administration module is responsible for analyzing input data, transmitting data to the board, and receiving and analyzing data that are sent back from the board. The data generated by the user’s circuit can be displayed. The functionality of the WaveForm player is the same as that of a logic analyzer. Because the WaveForm player takes the advantage of the huge capacity of a computer hard disk, it can store and show a much larger amount of data compared to traditional logic analyzers.

We also provide two multimedia modules—a video player and an audio player. These two functions allow video and audio data that are produced from the FPGA board to be displayed or played back in real time under the Microsoft Windows environment. Fig. 6 is the video module. When dealing with the video module, we have to designate a dedicated bus that contains the video data. The data format should follow the one mentioned in the following section. Moreover, one dedicated pin has to be used to indicate whether the video data in the bus are ready for display. Other parameters that are related to playing a video sequence, like the frame size and the frame rate, have to be initialized. Fig. 7 is the screenshot of playing stored data that are generated by the circuit. Functions of the audio module are similar to those of the video module, except that the audio parameters are different. Another advanced feature is the fast Fourier transform (FFT) module. To use the FFT module, the FFT size parameter is required, and prewindowing (such as Hanning or Hamming windows) is usually necessary. Fig. 8 is an example of using the FFT module.

C. Protocol and Multimedia Interfaces

There are currently three interfaces in the software—the video player, the audio player, and the communication protocol between the host computer and the board. We take advantage of the PCI burst data transmission mode such that the transfer can be more efficient. There are 1024 double words to be transferred at a time for each data exchange session. The I/O pins of the FPGA are divided into four 32-pin groups. That is, the maximum number of I/O pins to be connected to the System Control chip is 128. Each pin can be either an input pin to receive data from the host or an output pin to provide data to the host. One can use only one, two, or three pin groups instead of using all four groups. Therefore, there are up to 128 usable pins to be simultaneously monitored by the host. We can also use 64 pins to receive data from the host and let the host monitor the other 64 pins. Other combinations are also allowed. The arrangement of pin connections between the user’s circuit (preloaded in the FPGA) and the System Control chip on the board can be done in the configuration module within the System Software, as seen in Fig. 5. It is noted that the clock signal to the user’s circuit will be gated during the data communication period to prevent errors. This is achieved with the clock control module.

The video player supports YUV420, YUV422, and RGB. The data of each 32-bit pin group from the board are limited to 1 pixel for the RGB format. That is, the output pins of the user’s circuit can only contain 24 pins for the RGB data (eight bits for each color component), and the other eight pins are for user-defined control signals. Fig. 9(a) illustrates the protocol. Because there are four pin groups available, the other three groups can be used for other types of signals instead of video data. For the YUV420 format, the protocol is defined as shown in Fig. 9(b). The YUV422 format is similarly defined. Unsigned data are used in all three formats. Video resolutions can be quarter common intermediate format (QCIF), common intermediate format (CIF), quarter video graphics array (QVGA), and video graphics array (VGA). The default video frame rate is 30 frames/s and can be altered to meet the user’s requirement. The video player also supports still-image display.

The audio player adopts mono/stereo 8/16/32 bits pulse code modulation data format. The sampling rates can be either 22.05 or 44.1 kHz. Similar to the video cases, only the first pin group contains one sample of audio data. If the 8-bit mono format is used, the least significant byte is for audio data. If the 8-bit stereo format is used, the first byte is for the left channel, and the
Fig. 4. WaveForm player.

second byte is for the right channel. The other unused pins in the first group can be used as user-defined control signals. The protocols for the 16-bit mono and 32-bit stereo cases are shown in Fig. 10. Other formats are similarly defined. If 8-bit formats are used, the audio data should be unsigned. Otherwise, the data should be 2’s complement signed numbers. In addition, because home theaters have become more and more popular, audio players of 5.1 audio channels are in demand. The audio player supporting more channels will be added with appropriate peripherals. Note that only 64 pins out of 128 pins (i.e., one of the four pin groups) can be used to transfer video and audio data. The related setup can be also done in the configuration module.

D. Tunnel APIs for the User’s Programs

One other important feature is an interface between the System Software and the user’s application programs. Our system provides user APIs for the user’s programs, called the Tunnel APIs, to interact with his/her own circuit. The user’s programs will be able to transmit data to the design and receive processed data from the circuit through the interface. It requires the user’s program to be compiled with our library. The library encapsulates all the details of communications between the user’s programs and the System Software. Our system provides such a mechanism by using interprocess call. Before the user’s
program starts to transmit data to the circuit on the FPGA, the connections between the user's program and the System Software will be built. The system will check if the pin assignment of the user's program is consistent with the setting in the System Software. When transmitting data, the user's program has to send the data on a clock-by-clock base. The data will be received, transformed, and sent to the user's circuit as the input by the System Software and verification board. The output of the user's circuit will be sent back to the user's program in a similar way. Our system builds a virtual tunnel between the user's program and his/her hardware circuit with fewer redundancies. This feature enables the user to verify not only his/her hardware design but the software part in every design stage as well, particularly in the early stage. This functionality makes our system more flexible, and it is expected to help the user to develop and test almost any joint software–hardware designs. All data sent to the circuit and received from the circuit will be recorded by the System Software, and the user can observe the waveform on a screen. This is particularly useful when multimedia applications are concerned, because it is sometimes not easy to verify the circuit simply with the waveforms obtained from bus signals. To make sure that the design is fully functional, it is necessary to observe longer sequences of data.

III. HOW TO USING TUNNEL APIs

A. Usage of Tunnel APIs

There are three ways of using Tunnel APIs. First, if the user’s program and hardware are tightly coupled, which means that the actions of the program are affected by the signals at each clock cycle and vice versa, the user’s program has to send data and ask the System Software to immediately transmit to the verification board. In addition, the output of the circuit will
be also immediately sent back to the program. This is more suitable to some kinds of software–hardware cowork systems, or the software has to play the role of the hardware controller in the early development stage. Second, if the user’s program and hardware are not tightly coupled, the user’s program can send data in the batch mode. When each test vector is sent to the user’s circuit, and the corresponding circuit output is captured, all results will be sent back to the user’s program. Last, our system provides an interrupt mode. The user’s program can also generate data using the batch mode. A signal is used to trigger the interrupt mode when there are data to transmit. The user’s software can also set up the interrupt mode in the System Software. If an interrupt occurs, the user’s program can also check all the outputs from the user’s circuits or generate next test vectors for the user’s circuit.

B. Incremental Development

When developing a new hardware like a video codec, verification of each module has to be done before integration. Without other modules, it is usually not convenient and/or possible to fully individually verify a module. With our platform, the user can convert the design from a software module running on the host computer into a hardware module running on the FPGA one by one. Those software modules running on PC are called “virtual modules” or “virtual components.” An example is depicted in Fig. 11. Once a hardware module is verified, it can be combined with other modules. One by one, the overall design is integrated. In each integration stage, the integrated part can still verify with other software modules. We call this the “incremental development.”

C. Distributed Verification

Due to the flexibility of the Tunnel APIs, our system can even extend the verification process crossing computers. The user’s programs are not necessary to be executed in the same computer where our verification system is located. For example, computer A, in New York, runs a program to generate all test vectors. Target testing circuit and our platform are in computer B, in Los Angeles. The test vector generated by computer A will be sent to computer B as input vectors of the target circuit. The outcome generated by the target circuit will be fed back to the program running on computer A. Computers A and B are connected via a network. This function can be also used in other complex scenarios.

IV. MEDIA PROCESSING APPLICATIONS

The proposed system has been used in designing a few multimedia applications reported in [3]. In this paper, the H.264/AVC I-frame encoder is used to demonstrate the system for its higher implementation complexity [4]. H.264/AVC, which is also known as MPEG-4 Part 10 video codec, is the next-generation video codec standard. Compared to the currently popular MPEG-4 video codec, its computational complexity is at least four times higher. Its encoding procedure is also much more complicated. Therefore, hardware–software
codesign is necessary. The H.264/AVC I-frame is for still-image compression. It is reported that it has higher compression efficiency compared to JPEG and JPEG2000 [7], [8]. Therefore, it is expected to be the new codec for digital still cameras. This makes it a very good example to demonstrate the proposed system.

When encoding, an image is partitioned into slices first, and each slice consists of macroblocks. To encode every macroblock, pixels that are adjacent to the macroblock are used to generate prediction values of each pixel within the current macroblock. There are two types of predictions for H.264/AVC luminance pixels—Intra4×4 and Intra16×16. When the Intra4×4 mode is utilized, each macroblock is divided into 16 4×4 blocks. The adjacent pixels of each block are used to generate prediction values corresponding to each pixel in the blocks. There are nine prediction directions. Pixels with labels A to M in Fig. 12 are the adjacent pixels of the current image block. The arrows indicate the prediction direction. When the Intra16×16 mode is used, there are only four prediction directions. Therefore, there are 13 prediction types. After finishing the Intra4×4 and Intra16×16 predictions, the costs generated from all prediction types will be compared, and the one with the lowest cost is chosen as the prediction mode for this macroblock. Predictions for chrominance pixels are similar to Intra16×16, except that 8×8 blocks are used. The transformation and the quantization are then performed on the differences between image pixel values and predicted values. An entropy-coding process follows to produce the bitstream using the quantized data. Those who are interested in the H.264/AVC codec can refer to [4] and [5].

The H.264/AVC I-frame encoder in our project is divided into a software part and a hardware part. Because intra
predictions, transformation, and quantization in the I-frame encoding dominate most of the computation power; these are done in the hardware part. The software part is responsible for system initialization, providing video data, entropy coding, and bitstream generation. Fig. 13 is the flowchart of the overall encoder. It is noted that entropy coding is done by the host computer using pure software programs. The proposed design flow is to verify the hardware design without spending too much effort on the software side, which will be replaced by a reduced instruction set computer (RISC) processor and a digital signal processor (DSP) in the future.

V. SYSTEM DEMONSTRATION

Fig. 14 depicts the hardware part of our encoder. There are, in total, five submodules in our design. The I4MB, I6MB, and ChromaPred modules are responsible for each prediction type, as described in the previous section. The Coeff module performs transformation and quantization over the residual data. The Core module is for hardware flow control and management of shared resources such as memory and registers. When developing this system, we used the method mentioned in Section III-B. Presimulation using Cadence simulation tools is usually necessary. Then, we modified the program to send test vectors (i.e., image data) to the I4MB module running on the FPGA. The flowchart is depicted in Fig. 15. Each module and integrated modules are verified in the same way. Fig. 16 shows the verification result of the integrated module combining the I16MB and Coeff modules. Whenever the pin “level_wen” is high, one quantized coefficient is generated and appears on the bus named “level_to_ram.” These data are generated in real time when the encoder runs on the FPGA board. The software part is executed in the host computer, whereas the hardware part is downloaded into the FPGA. After all the modules are integrated, the encoding process is performed, as shown in Fig. 13. Fig. 17 shows the system operating architecture. A virtual tunnel is built. Some standard testing sequences are used to verify this encoder. The final bitstream is output to a file and can be decoded by using the JM software decoder [9]. Most of the encoded bitstreams are correct; however, several bugs caused by register or bus overflow are revealed in certain frames. These errors are corrected soon after they were found. The overall gate count is 28,057 using the TSMC .35 technology library [10], and the required embedded memory [static random access
memory (SRAM)] is 5760 bits. The hardware design of this encoder is also verified on Sun Workstation Blade1000 with 600-MHz UltraSPARCIII CPU and 1-GB system memory. Verilog-XL is adopted as our simulator. Presimulation, which verifies the functionality, only takes about 4337.1 s when 50 64×64 video frames are encoded. Postsimulation, which can verify timing and functionality of a design, takes 14 415.4 s (more than 4 h). With our platform, it takes only 52.3 s. Based on our H.264 I-Frame encoder design, the actual execution time on the FPGA side for the encoding process plus the bidirectional data transfer is merely 13.1 s. Although the standard 32-bit PCI interface runs at 33 MHz, the peak data transfer rate (tested only on one-way transfer from PC to FPGA) is only 33.6 MB, and the overall speed is decreased. There are two major reasons for this. First, because of the limited buffer size of the control chip, the burst length is only 4 kB. Therefore, host computer interrupts so frequently appear that the transfer is slowed down. Second, to make the data transfer as general as possible for all
kinds of user designs, the System Software needs to align the input and output data of the user design into the transfer data format based on the user pin assignment (e.g., Fig. 5) using bit operation microinstructions. This also slows down the data transfer rate. The verification time can be shortened if these limitations are lifted. This is very efficient for such a big project.

VI. CONCLUSION AND FUTURE WORKS

A versatile PC/FPGA-based verification/fast prototyping platform is presented. It was applied to the implementation of an H.264/AVC I-frame encoder and many other designs. The system provides very convenient Windows-based software and hardware interfaces for users to perform verification, testing data storage/reverification, I/O of signals, and so on. With the abundant resources of a PC, the system greatly reduces the time spent on design and debugging process. It is very suitable for multimedia SoC designs since one does not have to consider all I/O peripherals provided by the system in the early design stage.

Comparison among our platform, presimulation, and post-simulation may be not fair enough since the proposed platform takes the advantage of the FPGA-based hardware emulator. The presimulation and post-simulation take lots of time when the amount of testing vectors is as huge, as most multimedia applications; therefore, this is simply not acceptable for any time-to-market products. Although it is possible to use commercially available FPGA boards, they usually lack those useful tools provided by the proposed system. By combining the resources of a PC, the proposed system is more powerful and versatile.

There are some desired features that will be integrated to improve system performance. First, we plan to use other FPGA chips with an embedded RISC such as the Altera Stratix series chips so that it can be more versatile. Second, PCI-based interface cards are sometimes inconvenient to install, although PCI interfaces generally have higher bandwidth. A system with a Universal Serial Bus (USB) connection to the host computer may be a good alternative. Third, a step-by-step debugging tool should be provided when the design includes an embedded processor. Last, the H.264 encoder, as a demonstration for the proposed system, will be integrated with our own DSP and RISC processor as a full hardware encoder.

REFERENCES