Second Prize

H.264 VBS-BMA-Based Hardware Infrastructure Implementation on an FPGA

Institution: **Ching Yun University/ Department of Electronic Engineering**

Participants: **Wenxian Qian, Songzhi Gu**

Instructor: **Ou Qianmin**

Design Introduction

The block matching method, which is used for motion estimation, plays a key role in motion picture coding systems. Replacing the fixed-block-size block matching algorithm (FBS-BMA) with the H.264 variable-block-size block matching algorithm (VBS-BMA) addresses the issue that video object changes cannot be processed effectively, further improving video compression efficiency. Our design reduces the complex H.264 VBS-BMA calculation and features low latency, low power, and high throughput, delivering better coding performance.

H.264 VBS-BMA supports 4 x 4, 4 x 8, 8 x 4, 8 x 8, 8 x 16, 16 x 8, and 16 x 16 blocks. When serial frame data is transmitted via a network, the user can choose the most appropriate block for matching according to the current bandwidth, thereby obtaining the best transmission speed and frame quality. However, the H.264 VBS-BMA calculation is more complex. Therefore, we designed an efficient very-large-scale integration (VLSI) hardware structure that features high computing throughput to cut the complex calculation time required by H.264 video coding, reduce the calculation frequency, and improve coding performance. Our design uses the following elements:

- *Hardware*—Because the design has numerous complicated calculations for compression, we needed an effective system to process video frames in real time. To process serial frame data of a specific size and frequency effectively, we used Altera's Nios® II development kit. The Nios II processor is a soft-core processor based on the RISC architecture. It synthesizes a processor circuit (but not a hard core) on FPGA fabric, permitting scalable development. The Nios II processor can serve as the hierarchy setting for memory, and add processor instructions

independently to perform special calculations. While using Altera's development board and FPGA to develop hardware, we used the Quartus® II software and SOPC Builder, which is integrated in the Quartus II software, to develop the test platform. Besides adding our block matching circuit to the test platform via SOPC Builder, we also used Ethernet chip-embedded control hardware and components, and then transmitted them to the development board for rapid prototyping.

- *Software*—We used the Nios II Integrated Development Environment (IDE) to write and compile test software, implement network transmission between PCs, and migrate µC/OS-II into the hardware structure of the Nios II processor system.

- *PC*—We designed a graphical interface program that can be executed in the Microsoft Windows operating system (OS). The PC is connected to the test platform through the network, transmitting motion frames and vectors, and accomplishing calculation, estimation, compensation, and decoding. It displays real-time estimation frames on the PC screen to facilitate users' viewing.

Function Description

Users load serial frame data through the graphical user interface (GUI) as shown in Figure 1. Next, the test platform is connected as the the connection status shows (see Figure 2). After the frame is sent to the test platform via the network, the system performs the motion vector calculation, motion estimation, and motion compensation. Then, the estimation frame is sent back to the PC via the network and displayed on the monitor (see Figure 3).

Figure 1. User Loads Serial Frame Data on PC

Figure 2. PC Connects to the Test Platform

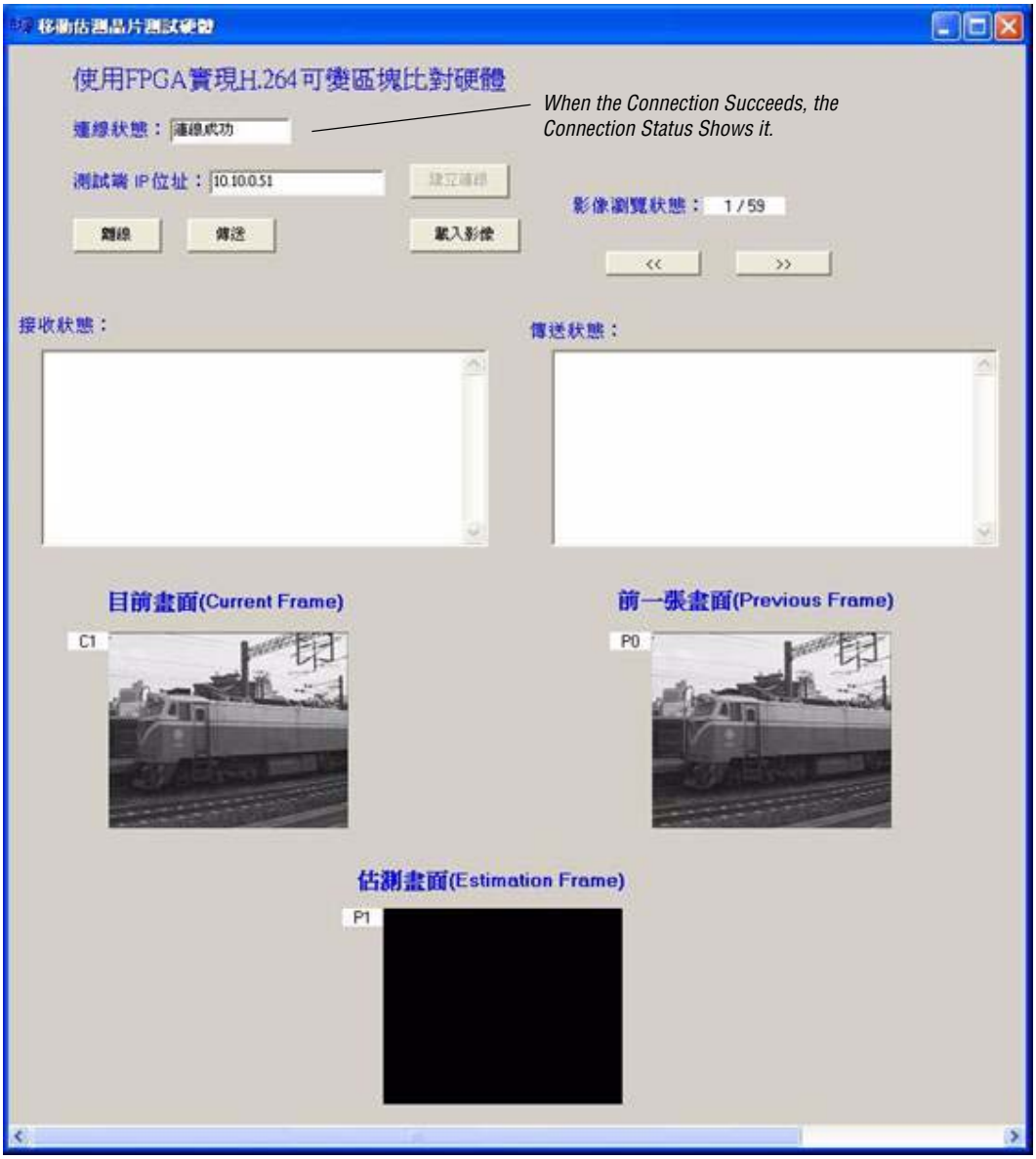


Figure 3. Test Platform Returns Estimation Frames after Processing

Performance Parameters

Our design has the following performance parameters.

- The design targets the Cyclone® EP1C20F400C7 device, which has 20,060 logic elements (LEs), 294,912 memory bits, and 301 pins. The design uses 8,881 (44%) LEs, 110,352 (34%) memory bits, and 176 (58%) pins.

- The system adds some necessary interface modules in the Nios II processor to improve system integration. Because most of the peripherals have related intellectual property (IP) cores that are well designed and tested, they can be used to accelerate hardware validation and software

development. We used many Altera IP cores, including SDRAM, Ethernet PHY chip, UART, SRAM, and flash memory. Additionally, the Nios II processor's 32-bit processing significantly enhanced the image processing efficiency and networking, and improved serviceability.

- The Nios II processor serves the design's control and algorithm cores. Because many IP cores and user IP cores can be added as peripherals, the system is more flexible.
- We enabled the motion estimation test function. After testing, we could estimate the frame result in real time.
- Currently, the device implements 4 x 4 block matching. In the future we can add other blocks using the same principles.

Design Architecture

Figure 4 shows the design concept of the H.264 VBS-BMA FPGA hardware structure. Users can load serial frame data through the GUI, transfer it to the test platform using Ethernet, test the H.264 block matching circuit, send the data back via the network, and (after test platform computation) display the frames on the PC interface for real-time viewing.

Figure 4. H.264 VBS-BMA System Design Concept

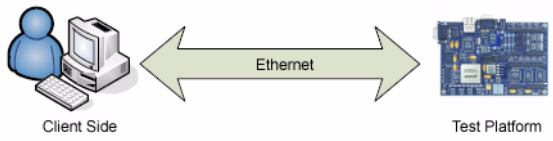


Figure 5 shows the PC operation. The current frame of the loaded serial frame data is transmitted to the Ethernet with a TCP socket. The PC receives motion vectors that are sent back from the test platform and decodes them to obtain the new estimation frame.

Figure 5. PC Operation

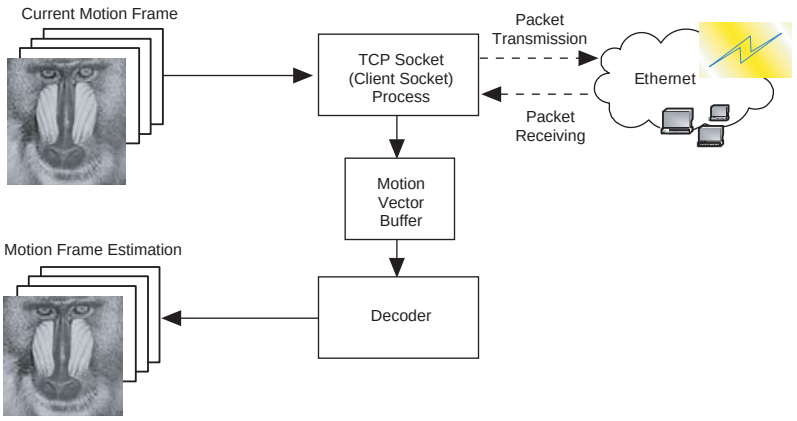
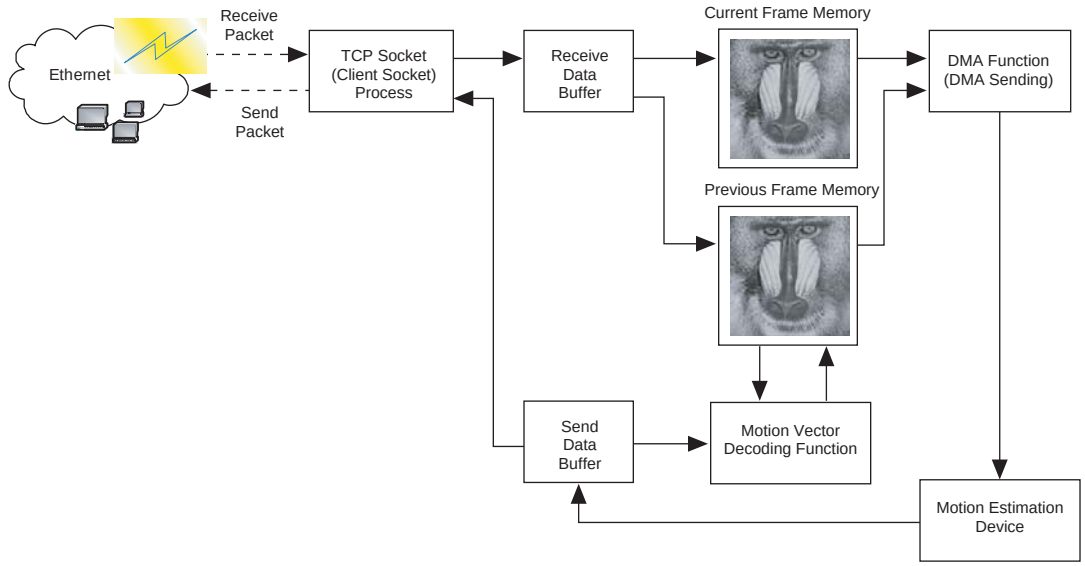


Figure 6 shows the test platform operation. First, the TCP socket receives the frame data package that the PC transmits from the Ethernet and caches it in the Receive Data Buffer. The first frame received is stored in the Previous Frame Memory as the previous frame, and the last frame is stored in the Current Frame Memory as the current frame. After the current frame is received, the current and previous frames are transferred to the block matching circuit for motion estimation and motion vector calculation. The DMA device accelerates data transmission from the memory to the block matching

circuit. Motion vectors obtained from the calculation are stored in the Send Data Buffer and are sent back to the PC via a TCP socket. During transmission to the PC, motion vectors are decoded to rebuild the estimation frame using the original previous frame. The new estimation frame is then put into the Previous Frame Memory to replace the original previous frame and become the previous frame for the next calculation.

Figure 6. Test Platform Operation



As shown in Figure 7, after the test platform is connected to the PC, it receives frame data (Cx) over the network from the PC. The first frame (C0) is transmitted directly to the Previous Frame Buffer for initialization; all other frames are transmitted to the Current Frame Buffer. After the current frame is received, the current and previous frames are transmitted to the motion estimation device for motion vector calculation. After the test platform conducts the motion compensation on the motion vector and previous frame, the generated estimation frame (Px) is transmitted to the Previous Frame Buffer to serve as data for the next motion estimation. Simultaneously, motion vectors are sent back to the PC via the network. Px and Px-1 (the previous frame) are stored in the PC. The system then determines whether data transmission is finished; if it is not, it continues to transmit Cx and repeats the whole process until it finishes.

Figure 7. System Flow Chart

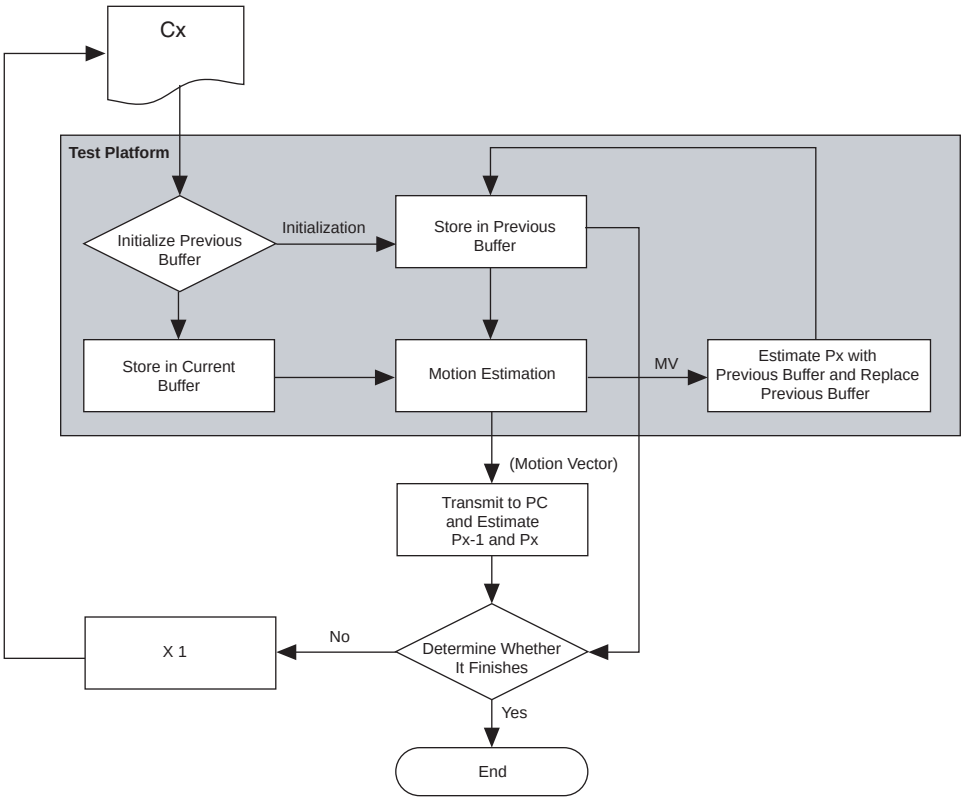


Figure 8 shows the test platform's hardware structure. The Nios II processor creates the hardware structure of the matching circuit. We used the Avalon® bus to connect peripherals like traditional peripheral models; for example, an LCD shows the test platform's IP address, an Ethernet MAC/PHY transmits and receives data packets, and SDRAM stores the previous and current frames. We used a direct memory access (DMA) model for effective data transmission between the peripherals and memory.

Figure 8. Test Platform Hardware Structure

Adapted from the Nios II Processor Reference Handbook

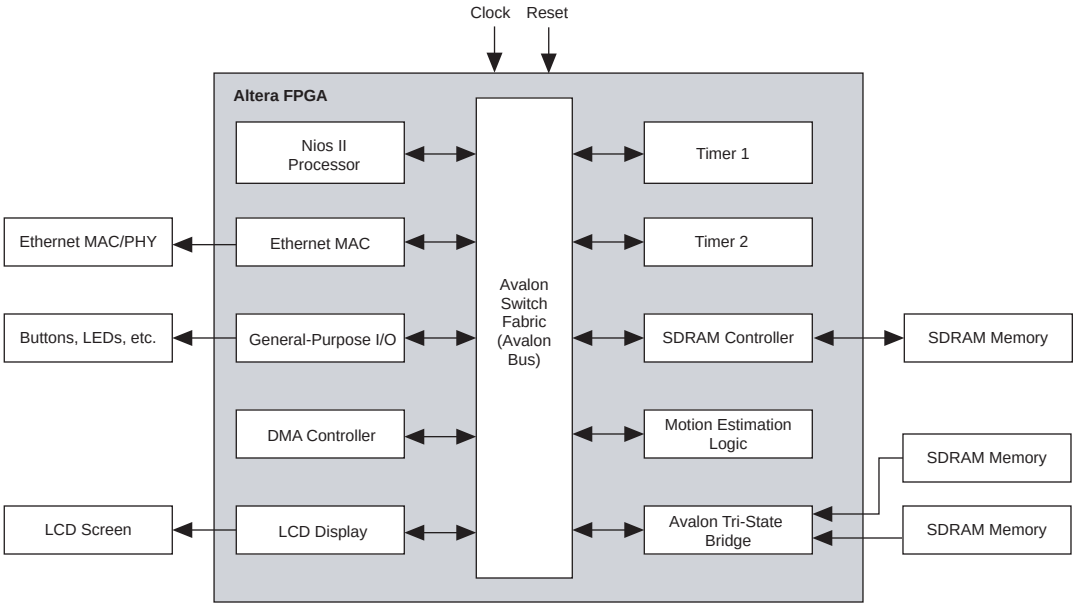


Figure 9 shows the test platform's software structure. It provides support from the inside to the outside. The first internal layer is the matching circuit's hardware structure, which supports the whole software structure implementation. The second layer is the hardware driver, which is generated automatically during the creation of the hardware structure and allows components in the outside layer to use hardware in the first layer. The third layer is Altera's hardware abstraction layer (HAL) library that enables components in the outside layer to use hardware. The other two layers are for the embedded operating system (OS) and protocol stack. The final layer is our application.

Figure 9. Test Platform Software Structure

Adapted from Using Lightweight IP with the Nios II Processor Tutorial

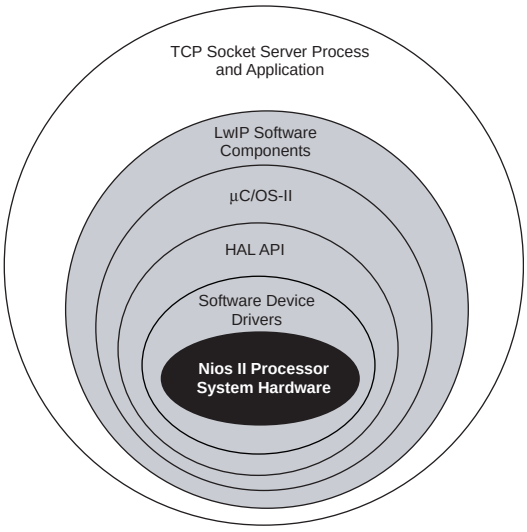
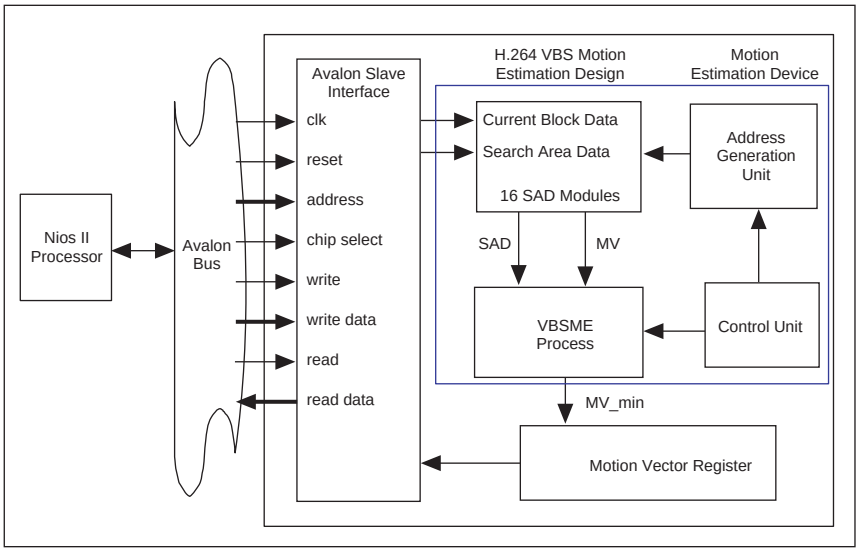


Figure 10 shows the system structure of the motion estimation device. The Avalon bus sends frames to the matching circuit. After the calculation, the Avalon bus reads motion vectors from the circuit's motion vector register and sends them to the PC.

Figure 10. Motion Estimation Device System Structure



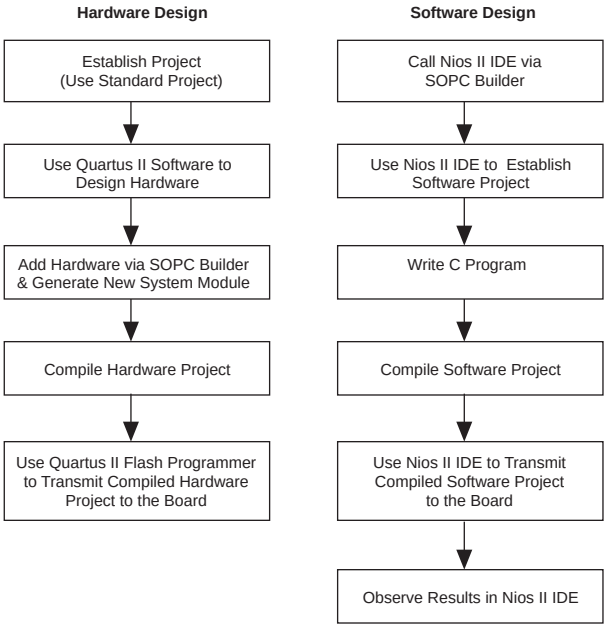
Our motion estimation architecture contains 16 sum of absolute differences (SAD) modules, a VBSME processor, an address generation unit (AGU), and a control unit. These modules operate as described below:

- Each of the 16 SAD modules handles a different SAD calculation based on the sub-block and its search area.
- The VBSME processor sums the SAD calculation from 16 basic blocks generated by the 16 SAD modules, while composing a SAD of different-sized sub-blocks and primary blocks and searching for the best motion vector.
- The AGU controls memory data reading and writing.
- The control unit controls coordination between the other modules.

Design Methodology

Figure 11 shows the hardware and software design flow charts.

Figure 11. Hardware and Software Design Flow Charts



Hardware Design

As shown in Figure 11, we used a standard project based on Nios II examples and wrote the hardware program using the concept of layered modules. After compilation, we opened SOPC Builder, integrated custom user IP, and added a DMA device to enhance the system performance. After we built the hardware structure, SOPC Builder automatically organized the hardware devices and generated the related drivers and files needed for software development. We transmitted the compiled test platform to the development board using the Quartus II software, using the FPGA combined with other hardware required by the test platform to perform rapid prototyping.

Next, we used the Nios II IDE to test the software code and compile the test software in the test platform. The Nios II IDE contains µC/OS-II (a real-time OS) and a protocol stack (light-weight IP), and a small TCP/IP protocol used in the embedded system, which helped us implement network communication between the TCP server socket and the PC. µC/OS-II provides quick responses, we migrated it to the hardware structure of the Nios II processor.

Software Design

As part of the software design, we modified the Altera-provided simple socket server IP. We set the IP address, subnet mask and port name, modified the read, receive, write, and transmit program, set the system library's send and receive buffer sizes, and downloaded the compiled project to the test platform.

To allow users to test the platform, we wrote a graphical interface program executed in Microsoft Windows using the Borland C++ Builder software version 6.0. Because we used TCP/IP, we used the client TCP socket to connect the PC with the test platform. Then, we were able to transmit motion frames and vectors, load serial frame data in the PC to test the block matching circuit, and send the estimation images generated by the test platform back to the PC for the user to view in real time.

Design Features

Our design has the following features:

- The H.264 block matching circuit uses Altera's DMA device to manage the transmission between the block matching circuit and SDRAM, i.e., to provide effective transmission without using the Nios II processor.
- We used SOPC Builder to add Ethernet chip control hardware and automatically generate the related drivers required for the system's network functions.
- The user IP is customized and integrated in SOPC Builder to perform complex computation. Additionally, the layered modular hardware circuit design reduces the complexity and allows users to customize the design.

Conclusion

In an era of software, designers seem to pay more and more attention to the study and development of software. However, our design shows that the H.264 VBS-BMA algorithm used for software can also be implemented in hardware to provide better performance. Additionally, Altera's simple socket server IP allows users to observe the estimation frame in real time without processing complex pins with a logic analyzer and signal generator and writing a complex testbench.

The Nios II processor allows users to create various devices—such as SDRAM, an Ethernet PHY chip, UART, SRAM, and flash memory—and embed them in the processor. The user can develop these modules directly with SOPC Builder, reducing the peripheral hardware design's complexity and development time.

In addition to gaining a deeper understanding of the Nios II processor during the design contest, we learned how to solve problems and achieve peace of mind, which is the most important asset that cannot be obtained from classroom teaching.