Designing an FPGA SoC using a Standardized IP Block Interface

Lesley Shannon, Blair Fort, Samir Parikh, Arun Patel, Manuel Saldana and Paul Chow
Department of Electrical and Computer Engineering
University of Toronto
Toronto, Ontario, Canada, M5S 3G4
{*lesley, fort, parikh, apatel, msaldana, pc*}*@eecg.toronto.edu*

Abstract

Designing Systems on-Chip is becoming increasingly popular as die sizes increase and technology sizes decrease. The complexity of integrating different types of Processing Elements (PEs) that use different communication protocols and interfaces complicates the system-level design methodology. Recent work provided a proof of concept demonstrating how a controller could be used to provide a generic system-level interface that separates the functionality of a PE from its communication protocols and makes the actual physical interconnections between modules a lesser problem. This paper summarizes how the SIMPPL model is able to implement the system-specific requirements of an MPEG-1 video decoder and the overhead this framework incurs.

1. Introduction

Designing Systems on-Chip (SoCs) is becoming increasingly popular as die sizes increase and technology sizes decrease. The complexity of integrating different types of processing elements (PEs) that use different communication protocols and interfaces complicates the system-level design methodology. Recent work presented the SIMPPL model; it uses asynchronous FIFOs to connect different Computing Elements (CEs) to create the system [3]. Figure 1 illustrates the underlying abstraction of a CE. The computational unit, called the Processing Element (PE), uses a controller to act as the physical inter-CE interface and to process the inter-CE communication protocols. The SIMPPL Control Sequencer (SCS) provides control instructions to the PE in addition to those received from other CEs in the system. These instructions are used to direct how each CE is used by the system.

The previous work provided a proof of concept, demonstrating that the controller is a viable CE interface for system design that greatly facilitated system integration for these simple designs [3]. It does not, however, address the effects and changes on the system-level design, integration and verification of more complex systems. This paper summarizes how the SIMPPL model is able to implement system-specific requirements and the overhead this framework incurs. As a sample design for this investigation, an MPEG-1 [2] video decoder is implemented by a four-person design team previously unfamiliar with the SIMPPL framework.

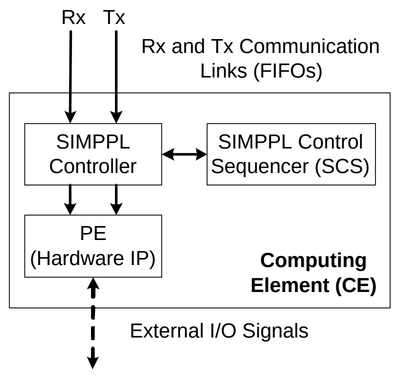

Figure 1. The SIMPPL CE abstraction.

2. Design Methodology

As previously stated, a complex application is required to properly investigate the design benefits and constraints of using the SIMPPL model and controller. FPGAs are often used to implement video and audio applications as they allow designers to parallelize and pipeline data-intensive applications. This suggests that part of the MPEG-2 standard, which is currently used for encoding DVDs, is a good choice for the investigation. The MPEG-2 decoder standard consists of an audio and a video decoder, where the video decoder presents a considerably more complex design problem.

A commercially-available implementation of a parameter-limited MPEG-2 video decoder core for an FPGA uses 7377 slices on a Virtex-E FPGA [1]. Since the Virtex2V2000 FPGA available on the development platform has only 10 752 slices, it is risky to assume that a complete MPEG-2 video decoder design will fit on the FPGA and run at speed. Therefore, the MPEG-1 video decoder is implemented as it closely approximates the complexities of MPEG-2 while still fitting on the available hardware. The main difference between the MPEG-2 and MPEG-1 video decoders is that MPEG-2 supports fully interlaced video whereas MPEG-1 does not. Given that both video decoders use the same functional mod-

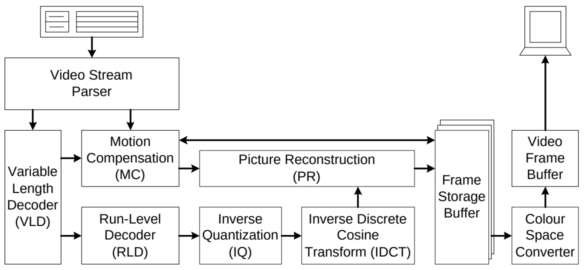

Figure 2. A Block diagram of a standard MPEG-1 video decoder to RGB video display.

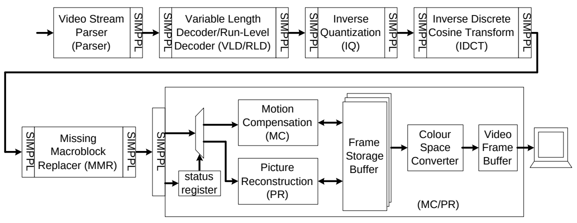
Video Stream Parser (Parser)
SIMPPL
Variable Length Decoder/Run-Level Decoder (VLD/RLD)
Inverse Quantization (IQ)
Inverse Discrete Cosine Transform (IDCT)
Missing Macroblock Replacer (MMR)
status register
Motion Compensation (MC)
Picture Reconstruction (PR)
Frame Storage Buffer
Colour Space Converter
Video Frame Buffer
(MC/PR)

Figure 3. The MPEG-1 video decoder implemented using the SIMPPL model.

ules, where MPEG-2 implements this extra option, the MPEG-1 video decoder is an adequate application for the purpose of investigating the SIMPPL system design methodology.

Figure 2 [1] shows a block diagram implementation of a generic MPEG-1 video decoder outlining the datapath from the compressed data input to the video output display. Recalling that the SIMPPL system-level architecture is fixed as a network of CEs interconnected via Asynchronous FIFOs in the SIMPPL model, this architecture naturally lends itself to a modular design methodology. Initially, the four person design team partitioned the block diagram into PEs that they designed and verified as individual modules. Then the team integrated the SIMPPL controllers, along with their SCSs, with the PEs after which they verified the CE interface.

In contrast to the system-level architecture, the internal structure of the CE is flexible. This is because one specific architecture is not likely to work for all applications, which reduces the usability of the model. Instead, the CE's architectural definition is conceptual: the system-level control and communication must be separated from the PE's computation.

Figure 3 shows a block diagram of the MPEG-1 video decoder they implemented using the SIMPPL model. Obviously, the CE architecture employed in the MPEG-1 video decoder must adapt the CE abstraction, shown in Figure 1, to the system specific requirements. Figure 4 shows a block diagram of the specific CE architecture used by the MPEG-1 video decoder. The PE now has independent input and output SIMPPL controllers called the Consumer and the Producer, respectively, where each controller has its own SIMPPL Control Sequencer (SCS).

If the CE's architecture only had one controller, it could not concurrently consume and transmit data

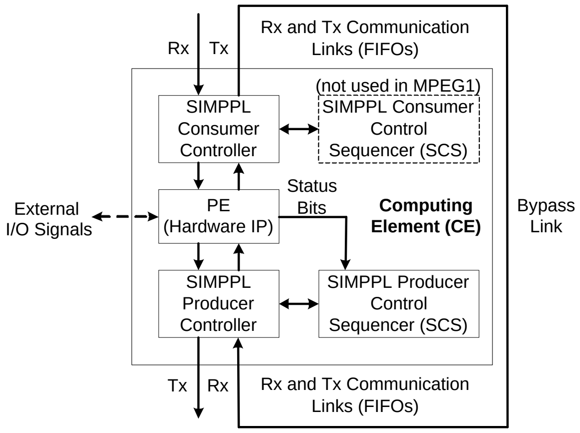
Rx
Tx
Rx and Tx Communication Links (FIFOs)
(not used in MPEG1)
SIMPPL Consumer Controller
SIMPPL Consumer Control Sequencer (SCS)
External I/O Signals
PE (Hardware IP)
Status Bits
Computing Element (CE)
Bypass Link
SIMPPL Producer Controller
SIMPPL Producer Control Sequencer (SCS)
Tx
Rx
Rx and Tx Communication Links (FIFOs)

Figure 4. The MPEG-1 SIMPPL CE implementation.

packets to the system. However, all the PEs in the MPEG-1 application are implemented in a pipelined format as the MPEG encoded data stream is serially encoded. This allows multiple data packets to be processed in flight, which increase the system's throughput. Using independent controllers for receiving and transmitting data allows the Consumer to receive a data packet for processing while the Producer transmits a packet to the adjacent CE.

3. Results and Conclusions

The usage of SIMPPL controllers as the physical and communication protocol interface between CEs greatly facilitated the system level design. The system-level integration took 12 hours, approximately 1% of the total design time of 1607 hours. Furthermore, the PEs only required 3.8% of the total design time to be converted into CEs. The SIMPPL framework attributed approximately 23% more LUTs and 16% more flipflops of overhead to the system design.

Acknowledgments

This research was supported by the Natural Sciences and Engineering Research Council and the O'Brien Foundation. The authors would also like to thank CMC for the SLPS station and Xilinx for the CAD tools.

4. References

[1] CS6651: Amphion MPEG2 Video Decoder for FPGA. Online: http://www.amphion.com/ cs6651.html.

[2] Joint Technical Committee ISO/IEC JTC1. Information technology: Coding of Moving Pictures and Associated Audio for Digital Storage Media at up to about 1.5Mbit/s. *International Standards Organization ISO/IEC 11172-2*, Part 2, 1993.

[3] L. Shannon and P. Chow. Simplifying the Integration of Processing Elements in Computing Systems using a Programmable Controller. In *IEEE Symposium on Field-Programmable Custom Computing Machines*, Apr. 2005.