A SYSTEM DESIGN METHODOLOGY FOR REDUCING SYSTEM INTEGRATION TIME AND FACILITATING MODULAR DESIGN VERIFICATION

Lesley Shannon, Blair Fort, Samir Parikh, Arun Patel, Manuel Saldaña and Paul Chow

Department of Electrical and Computer Engineering
University of Toronto
Toronto, Ontario, Canada, M5S 3G4
email:{lesley, fort, parikh, apatel, msaldana, pc}@eecg.toronto.edu

ABSTRACT

This paper provides a realistic case study of using the previously introduced SIMPPL system architectural model, which fixes the physical interface and communication protocols between processing elements (PEs) using PE-specific SIMPPL controllers. The implementation of a real-time MPEG-1 video decoder using SIMPPL provides a practical demonstration of how the complexity of system-level design issues are reduced by enabling rapid system-level integration and on-chip verification. The adaptation of the MPEG-1 PEs into the SIMPPL framework combined with the system-level integration was accomplished in 72.5 hours, which is only 4.5% of the overall system design time, instead of the more typical system integration times that can be as much as 30% of the design time.

1. INTRODUCTION

Lengthy design times are an issue for ASIC designs, and more recently for FPGAs, as they can be used to implement complete embedded computing systems on a single chip. One method of reducing design time is to abstract the low-level implementation from the designer by trying to capitalize on previously designed Intellectual Property (IP) modules. However, IP reuse is more complex in hardware design than the reuse of previously designed software functions in the development of new software. Software designers benefit from a fixed implementation platform with a highly abstracted programming interface that allows them to focus on adapting software functionality to new applications. In contrast, the challenges facing hardware designers when trying to reuse IP include non-standardized physical interfaces and communication protocols [1, 2].

Recent work presented the SIMPPL model, *S*ystems *I*ntegrating *M*odules with *P*redefined *P*hysical *L*inks, where Asynchronous FIFOs are used to connect the different Computing Elements (CEs) in a point-to-point manner to create the system [3]. This model is similar at the system-level to Kahn [4] and Dataflow [5] Process Networks, however, SIMPPL also facilitates CE reuse and reprogramming through the underlying abstraction of the CE shown in Figure 1. The computational block, called the Processing Element (PE), uses

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

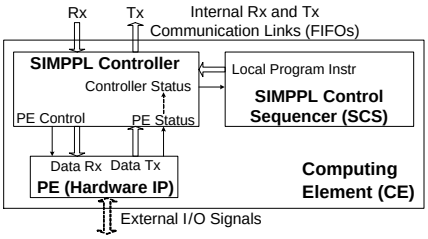

Fig. 1. The SIMPPL CE abstraction.

a controller as the physical inter-CE interface that processes 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. The controller executes these instructions to direct how each PE is used by the system with minimal overhead [3]. Thus, the controller's fixed interface allows designers to decouple the system communication and control from the PE's computation, while the SCS provides the flexibility to reconfigure the control without modifying the physical interface.

The previous work provided a proof of concept for the SIMPPL model. It used a controller to interface small modules in simple video streaming applications to demonstrate that the SIMPPL controller is a viable method of reducing the complexity of design reuse, while adding a small amount of overhead to the CE implementation [3]. This paper extends that work to examine the effects on a non-trivial system-level design and demonstrate the inherent flexibility in SIMPPL's framework that makes it adaptable to different applications.

The focus of this paper is to describe both the benefits and constraints of creating and verifying designs using the SIMPPL model on an FPGA. These include:

- the model's extensibility to new system requirements
- the reduced system-level integration effort
- an on-chip testbed for thorough verification of CE(s)
- a measurement of the resource overhead of the SIMPPL framework
- a demonstration of the minimized system-level redesign effort

Demonstrating these characteristics of the framework requires a complex system as a case study. Figure 2 [6] shows a block diagram implementation of a generic MPEG-1 video decoder [7] outlining the datapath from the compressed data

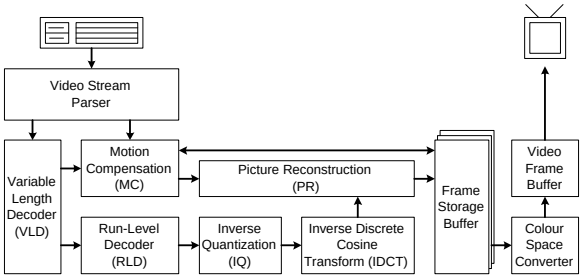

Fig. 2. An MPEG-1 video decoder to RGB display.

input to the video output display. The MPEG-1 video decoder closely approximates the MPEG-2 video decoder, which uses 7377 slices on a Virtex-E FPGA in a commerical implementation [6] and is too large for our prototyping system, yet provides a realistic test system for the system model.

The remainder of this paper begins with Section 2 discussing how adapting a traditional modular system design methodology to use the SIMPPL model can reduce design time. Section 3 describes how the video decoder is implemented within the SIMPPL framework and Section 4 discusses an on-chip testbed that enables thorough verification of each CE. The system implementation results are presented in Section 5 and the conclusions and possible future work are summarized in Section 6.

2. DESIGN METHODOLOGY

When designing an SoC, there are numerous system specific issues to consider. This section describes a generic design methodology for designing within the SIMPPL framework and some of the application-specific issues that must be addressed. The particular design choices made for the MPEG-1 video decoder are then discussed in Section 3.

An initial high-level system specification in languages such as SystemC [8] is becoming increasingly common for complex systems. A designer describes SIMPPL modelled SoCs in these languages by specifying the SIMPPL controller and SCS interfaces for each PE module. Since the SIMPPL system-level architecture is fixed as a network of CEs connected via Asynchronous FIFOs, it lends itself to a modular design methodology. Initially, the PEs are designed and verified as individual modules in simulation and, where possible, on-chip. The SIMPPL controllers and their SCSs are then integrated with the PEs and the resulting CEs are verified in simulation. The remainder of the testing is performed on-chip to verify that data packets are properly received, processed, and retransmitted by the CEs using the on-chip testbed described in Section 4.

While the system-level architecture is fixed, the internal structure of the CE is flexible because a set CE architecture is unlikely 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 as shown in Figure 1. This means that the designer can select: the CE's number of Rx and Tx Communication Links, whether or not the CE interfaces with off-chip I/O, and the number of SIMPPL controllers per CE, as well as their instruction sets, to suit each PE's functionality. Even the granularity of partitioning of the system's dataflow into individual PE's is left unfixed to facilitate the adaptation of different applications to the model.

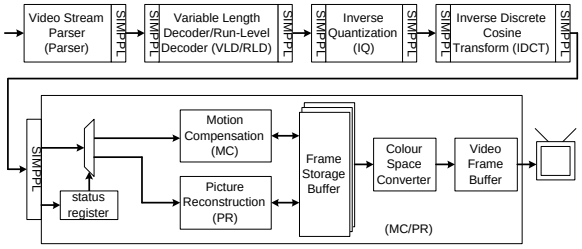

Fig. 3. The SIMPPL model MPEG-1 video decoder.

By analyzing the design's dataflow, a system-level architecture can be chosen that partitions the design into PEs that are reusable in future applications. The designer must also consider the computational latency and throughput of each PE to prevent bottlenecks in the system and to guarantee that the application's latency and throughput requirements are met. Researchers have developed tools that can be used to model and analyze the system [9] or profile it on-chip [10] to meet the system performance requirements.

3. MPEG1 IMPLEMENTATION

The implementation of the MPEG-1 video decoder using the SIMPPL model was done by a four-person design team initially unfamiliar with the model. The changes to the system-level architecture are described first, followed by the application-specific implementation of the CE model from Figure 1 and the updates to the original controller architecture. Finally, the SCS architecture used for this application is discussed. In each case, the benefits and overhead inherent to the updated system model are discussed.

3.1. Adapting MPEG-1 to the SIMPPL model

The design team partitioned the video decoder into PEs before selecting a system-level architecture. While the PE partitions are not made to ensure reusability, they are still readily adaptable to CEs. Figure 3 shows a block diagram of the MPEG-1 video decoder implemented using the SIMPPL model, which has been redesigned as a pipeline due to the serial nature of MPEG-1 encoded data. The system is now divided into five CEs: the Parser, the VLD/RLD, the IQ, the IDCT, and the MC/PR. All the CEs are implemented in hardware except for the Parser. Due to design time limits, the team used software running on a processor, termed a software CE, to read the MPEG encoded data from external memory to generate the data-packets for the VLD/RLD CE.

The direct paths from the Parser and the VLD/RLD CE to the MC/PR CE are removed in the new system model, as shown in Figure 3. While the direct path from the Parser is only used to provide status bits about the transmitted blocks to the MC module, the VLD provides data-packets to both the MC module and the RLD module for processing. Even though the SIMPPL controller supports branching operations, the resulting architecture uses only one datapath. The new architecture must now provide some mechanism for directly passing data between two non-adjacent CEs in the pipeline while ensuring that the order of the transmitted data-packets is maintained. This is addressed by adding new functionality to the SIMPPL controller and is discussed in greater detail in Section 3.2.

SIMPPL controllers act as the interface between PEs and the rest of the system. The predefined interface facilitates

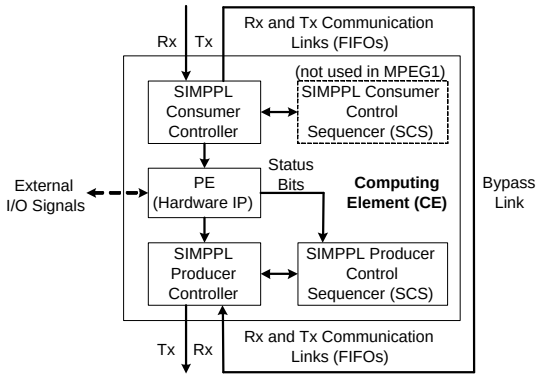

Fig. 4. The MPEG-1 SIMPPL CE implementation.

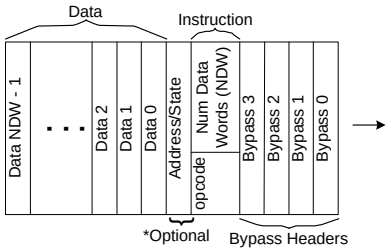

Fig. 5. A Data packet with four bypass instructions.

the physical integration of components but increases the resources required for the interface logic. Each Asynchronous FIFO (Communication Link) between CEs provides buffering for the transmitted data-packets. The Communication Links also allow designers to decouple the clock domains and vary the width of the transmitted data words to increase the bandwidth between CEs. In this system, all the FIFOs are 33 bits wide, where a single bit indicates the beginning of a packet header transmission and the remaining 32 bits pass the state and data values.

The data-packets transferred between modules consists of two or three fields, where the first field is an instruction word consisting of the opcode plus the number of data words in the packet. The second field is optional, providing the state information about this packet as an address word to the PE. For example, the status bits that used to be passed directly to the MC module from the Parser, as shown in Figure 2, are now stored as state bits in the packet. The remaining field in the packet is the data words that are consumed by the PE.

3.2. Adapting the SIMPPL CE for MPEG-1

Figure 4 shows a block diagram of the CE architecture for the MPEG-1 application based on Figure 1's abstraction. All the PEs in the MPEG-1 application are implemented in a pipelined format to allow multiple data packets to be processed in flight. This ensures that the real time throughput requirements for the video display are met. Therefore, each 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). 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.

The primary update to the controller addresses the requirement that the VLD/RLD CE be able to pass data directly to the MC/PR CE, as described in Section 3.1. To resolve this issue, a new bypass instruction is added to the SIMPPL controller instruction set. It enables the controller to receive a packet with a bypass instruction header and then retransmit the packet, less the bypass instruction. Figure 5 illustrates a data packet that has four bypass instructions tagged to the beginning. By tagging *N* bypass instructions in front of a data packet, the packet will bypass *N* controllers before the *N+1* controller processes the actual data packet.

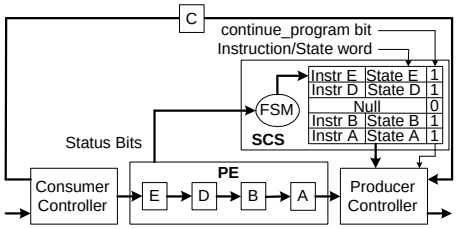

Fig. 6. A CE with multiple packets of data in flight.

3.3. Updating the SCS for the MPEG-1 system

Since the Consumer controller does not create any data packets, it does not run a local program. Therefore, an SCS is not required for this application, although the Consumer supports one as indicated in Figure 4. All data processing performed by the Consumer is based on the data packets received from the preceding CE via the Rx Communication Link. However, the PEs in the MPEG-1 video decoder can consume different types of packets that directly determine the instructions issued by the Producer for each transmitted data-packet. The transmitted instruction's data dependency requires that the the local program in the Producer's SCS be dynamically generated based on the packets received by the Consumer unlike the static programs used in the proof-of-concept study.

Figure 6 illustrates how a CE that received packets *A* through *E* in order, where packet *C* had two bypass instructions, generates the appropriate program for the Producer's SCS. Recalling that the order of packets received by a CE must be maintained when they are transmitted to the subsequent CE, it is imperative that data packets *A* and *B*, which were inflight when packet *C* arrived, are transmitted first. To enable this functionality, the instructions from the Rx Communication Links and those created in the Producer's SCS must have variable processing priority. When the *continue_program* status bit, as shown in Figure 6, is set, the controller continues to fetch available instructions from the SCS, even if there are data packets to be processed on the receive link. Therefore, each Producer's SCS uses a FIFO not only to store the instruction word and the state word for each packet, but the status of the *continue_program* bit as well. The PE enqueues valid instructions into the FIFO for every data packet in flight, setting the *continue_program* bit for each instruction, as indicated in Figure 6.

To ensure that bypassed packets are retransmitted in the proper order, the PE must detect if the Consumer receives a *bypass* instruction. In this situation, the PE will queue a null instruction into the FIFO with the *continue_program* bit set low, as shown in Figure 6. To guarantee that instructions are enqueued in the Producer's FIFO in the correct order, the SCS state machine must push the correct instruction onto the FIFO before the Consumer controller finishes reading the current data packet. The Producer will then dequeue the

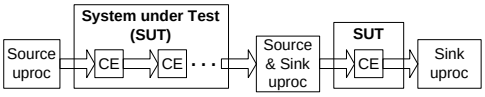

Fig. 7. The on-chip testbed for debugging CEs.

instructions and retransmit the data packets in order. When the "null" instruction is detected with the *continue_program* bit set low, the Rx Communication Link will be given priority. The bypassed packet will then be retransmitted by the Producer to the subsequent CE and the "null" instruction will be dequeued from the FIFO.

Thus, for the example shown in Figure 6, the Producer will transmit packets *A* and *B* from the PE. It will then detect a "null" instruction, with the *continue_program* bit set low, and process packet *C* from the bypass link, while simultaneously dequeuing the "null" instruction. This will be followed by packets *D* and *E* being sent to the next CE.

4. ON-CHIP TESTBED

The standardized physical interface and communication protocols of a CE allow the designer to use a flexible testbed architecture as shown in Figure 7. CEs can be verified individually, as independent processing stages, or in combination with adjacent CEs. Furthermore, since the design is implemented on an FPGA, it is possible to run the testbed on-chip to verify the behaviour of CEs with a large number of data packets to obtain quick and accurate results. Previous work demonstrated that debugging [11, 12] and profiling [10] designs using on-chip resources results in a significant reduction of the time required to obtain information for the designer. Since design verification commonly requires greater than 50% of the overall design time, sometimes as much as 70% [13], it may be possible to reduce the percentage of time spent verifying the design, and thus reduce the overall design time.

The testbed comprises the processors and the software required to generate (Source) and interpret (Sink) data packet streams for the CEs. The MPEG-1 video decoder is designed for a Xilinx Virtex2V2000, so the MicroBlaze^{TM} soft processor is used in this testbed. High-level functions are built to generate each data packet from the instruction and data pointer specified by the user. The user can then quickly alter the number and types of data packets sent by the Source to the System Under Test (SUT) by changing the instructions in the source code and then compiling and downloading the processor executable to the Source Processor. Creating the data stream using software allows a significantly quicker turnaround time for testing the SUT with different data packet streams than is possible with the source data stream coded as a separate hardware module. The Sink Processor runs a program that detects and interprets packets received from the SUT and then allows the user to log them. The Sink processor program can also be combined with the Source processor program to allow designers to log the intermediate state of the design as shown in Figure 7.

The on-chip testbed facilitated the detection of a significant PE error that required the redesign of the MPEG-1 video decoder pipeline. Using the MPEG-1 pipeline from the VLD/RLD CE to the MC/PR CE as the SUT, the design team found that a portion of the design specification for the MC/PR PE had not been implemented. The team created a new CE called the Missing Macroblock Replacer (MMR) CE and inserted it into the decoder pipeline just before the MC/PR CE to correct the error. The modularity and structure of SIMPPL made this change to the pipeline very easy.

Although the on-chip testbed runs orders of magnitude faster than in simulation, it does not likely exhibit the exact runtime behaviour of the final system. A runtime data stream could be irregular with data words sometimes arriving every clock cycle and sometimes delayed for numerous clock cycles, thus the Source and Sink may process data slower or faster than the system at runtime. However, the Consumer and Producer controllers, which interface the CE with its preceding and subsequent CEs, are able to abstract runtime data behaviour from the PE as they separate the communication protocols from the actual data processing. Both are able to properly stall the PE if there is no source data in the Rx Communication Link or no space in the Tx Communication Link so that the PE exhibits correct runtime behaviour independent of the data rate.

5. IMPLEMENTATION RESULTS

This section provides the implementation statistics for the resource usage and the design time of the MPEG-1 video decoder running at 30 frames per second to generate 320 by 200 pixel images on a monitor using the Xilinx Multimedia Board's Virtex 2V2000.

5.1. Resource Usage

Table 1 summarizes the resources used by the hardware CEs in the MPEG-1 video decoder system. Column 1 list the hardware CEs for which the resource usage measurements will be reported in the remaining columns. This excludes the Parser CE as it is a software CE implemented on a MicroBlaze^{TM} and the focus here is on the overhead for hardware PEs. Columns 2 and 3 report the number of LUTs and flipflops used by each PE and SCS respectively. The Virtex 2V2000 provides 56 dedicated Block RAMs (BRAMs) and 56 hard multipliers as extra design resources, along with the homogeneous array of LUTs and flipflops in their Combinational Logic Blocks (CLBs). Column 4 reports the number of BRAMs and multipliers used in the PEs as none are required for the SCSs. Since neither the Producer controller nor the Consumer controller use BRAMs or multipliers, the total logic resource usage for the controllers is reported in terms of LUTs and flipflops in Columns 5 and 6 respectively. Finally, Column 7 reports the percentage of extra LUTs and flipflops required in addition to the PE to create each CE. This is calculated by totaling the number of the LUTs/flipflops used by the two controllers and the SCS and then dividing it by the LUTs/flipflops used by the PEs.

Consumer as well as Producer controllers have relatively consistent resource usage among the CEs due to their similar instruction sets. However, the IQ and IDCT CE have slightly larger controllers because they execute bypass instructions and support variable instruction priority. SCS resource usage for all five CEs is minimal, the maximum being 92 LUTs and 98 flipflops by the VLD/RLD CE's SCS. This is due to the fact that its Producer has to generate both packets for the adjacent IQ CE and bypass packets. The great variance in the CE sizes arises from the different PEs they include, how complex they are algorithmically and how much of their design can be moved into BRAMs and Multipliers.

The MMR PE is the smallest because it is a patch to fix an error. Ideally, its functionality should have been encom-

Table 1. Table of the resource usage of the individual modules and total system.

Module Name	PE (LUTs/ Flipflops)	SCS (LUTs/ Flipflops)	Number of PE BRAMs/ Multipliers	Consumer (LUTs/ Flipflops)	Producer (LUTs/ Flipflops)	% Overhead (LUTs/ Flipflops)
VLD/RLD CE	606/699	92/98	9/0	119/70	217/42	71/30
IQ CE	429/201	86/13	2/2	126/70	302/106	120/94
IDCT CE	1091/1217	67/24	3/16	126/70	302/106	45/16
MMR CE	141/152	47/32	0/0	115/69	217/42	269/94
MC/PR CE	1705/742	0/0	2/5	115/69	0/0	7/9
Total System	7248/4118	292/167	16/23	601/348	1038/296	27/20

passed in the MC/PR CE, but a redesign of the PE required more testing and risked more errors than adding a separate module to implement the extra functionality. Although the percentage overhead of adapting the MMR PE into a CE was 269% in terms of LUTs and 94% in terms of flipflops, the percentage overhead of the LUTs for the MMR CE is greater than 100% because the PE uses less LUTs than the combination of the producer and consumer controllers, however, it minimized the design time required to fix the error as discussed in Section 5.2.

Row 7 summarizes the complete hardware MPEG-1 video decoder system resource usage in terms of LUTs, flipflops, BRAMs and multipliers. Column 2 of Row 7 report the total number of LUTs and flipflops in the complete hardware system, which comprises the resources required for the FIFOs used as Communication Links between CEs, the system reset manager, and the CEs themselves, and the totals for the SCSs. The fifth and sixth columns in the final row report the total overhead of the Consumer and Producer controllers used in the system and the final column is the percent overheard of converting all the PEs to CEs with respect to the total system resource usage. Based on this calculation, the total system overhead of the SIMPPL model for the MPEG-1 design is 27% of the LUTs and 20% of the flipflops used by the system. However, these numbers have been inflated by the unplanned inclusion of the MMR CE to patch the missing part of the design specification. If we assume that the current MC/PR CE includes the MMR PE's functionality, we can remove the MMR CE and reduce SIMPPL's design overhead to 23% of the LUTs and 17% of the flipflops. These numbers can be further reduced to reflect the necessity of system integration and control logic if the SIMPPL framework is not used. Thus, even if the SIMPPL model with its increased flexibility and simplified integration is not employed, a portion of this extra logic is still required to implement dedicated protocols for passing data correctly between the different PEs shown in the original MPEG-1 block diagram in Figure 2. A reasonable first order approximation of the necessary dedicated system control logic is the sum of all the SCSs used in the system as they implement the PE control state machines. Therefore, an approximation of the overhead of the SIMPPL framework is calculated as the resources used by all the controllers divided by the total system resources. The approximated actual overhead of the SIMPPL framework is reduced to 19% of the LUTs and 14% of the flipflops in this MPEG-1 system.

5.2. Design Time Statistics

Table 2 provides a synopsis of the design times required for the different phases of the MPEG-1 video decoder system's hardware design in terms of hours. The first column reports the portion of the design time being measured and Columns 2 through 6 list the different hardware CEs for which these numbers are reported. Column 7 reports the percentage of the total design time that the sum of the hardware CE design times attributes to each phase of the design.

PE design and initial debugging are reported in the second row. This is the design time required to create the initial PE design, debug it in simulation and, where possible, perform some initial on-chip debugging. These values vary greatly depending on the complexity of the PE and how much independent algorithmic development was required. As previously mentioned, the MMR PE is relatively simple and both the IQ PE and IDCT PE are well-defined with readily available example implementations. Both the VLD/RLD PE and the MC/PR PE required a significant portion of time to develop and debug their algorithms.

Before trying to convert their PEs into CEs, the design team members each required about ten hours to learn about the details of the SIMPPL model. This includes understanding the operation of the Consumer and Producer controllers and how the SCS directs the local operations of the PE. These hours are not included in Table 2 as they are a one-time, non-recurring cost for using the SIMPPL framework as opposed to being specifically attributable to the MPEG-1 video decoder design. The PE-Consumer and PE-Producer integration times measure the time required to adapt the PE interface to the controller's requirements. This was most costly in the cases of the MC/PR PE-Consumer integration and the VLD/RLD PE-Producer integration, which required that their PE interfaces be adapted to the controller requirements. The Producer's SCS design time was dependent on the complexity of the dynamic program that could be run by the Producer. Once both the Consumer and Producer had been integrated along with SCS, CE testing was performed to verify that packets were being properly received and retransmitted by the CE.

At this point, the design team was able to perform a second phase of PE verification using the on-chip testbed to thoroughly test the operation of their CE. Once the CEs had been verified, the correct system-level connections and constraint specification files were generated in 12 hours. The total system design time was reported to be 1608 hours of which 1376 hours, or 85.6%, of the time was required for the PE design and initial debugging phase and another 9.9% for the on-chip phase of PE verification. This means that only 4.5% of the time was actually used to generate the CEs from their PEs and then integrate them into the system. For complex designs, system integration can tradition-

Table 2. Table of the CE design and integration times required for the system given in hours.

Measured Design Time	VLD/RLD CE	IQ CE	IDCT CE	MMR CE	MC/PR CE	Percentage of Total Time
PE design and initial debugging	480	80	96	19	700	85.6%
PE-Consumer Integration	3.5	4	1	0.25	15	1.5%
PE-Producer Integration	10	4	1	0.25	0	0.9%
Producer's SCS Design	1.5	3	1	0.5	0	0.4%
CE Testing	2	5	3	0.5	5	1.0%
Second Phase PE Verification	9	8	16	1.5	125	9.9%

Total System Integration Design Time	12
Total System Design Time	1607

ally account for as much as 30% of design time [13], more than 6 times the system integration time for the MPEG-1 video decoder usingthe SIMPPL model. This allowed the designers to focus the majority of their efforts on creating properly functioning PEs, as opposed to system-level control and communication protocols.

An interesting point of comparison is to look at the design times required for transforming the IQ PE into the IQ CE versus the IDCT PE into the IDCT CE. Both CEs were created by the same individual, where the IQ CE was developed first. As can be seen from Columns 3 and 4 in Rows 3 through 6 of Table 2, adapting the IQ PE into a CE required significantly longer than for the IDCT CE. This was due to the designer's learning curve for understanding the SIMPPL CE model. Once the IQ CE was completed, the designer was sufficiently comfortable with the model to successfully implement a comparable interface in six hours as opposed to the 16 hours initially required.

Similarly, by the time the need for the MMR CE was determined, the design team had almost completed the system. The large resource usage overhead of using SIMPPL controllers to adapt the MMR PE into a CE required only 1.5 hours and the increase to the system integration time was nominal due to the designer's familiarity with the model. If an application-specific system model had been used for the video decoder, the time to fix the MC/PR module and reintegrate it into the system would have been significant and the entire verification phase would have to have been repeated. Instead, the complete design of the MMR CE, from the inital PE design to the final on-chip verification required only 22 hours.

6. CONCLUSIONS AND FUTURE WORK

The usage of SIMPPL controllers as the physical and communication protocol interface between CEs greatly facilitated the system-level design. The SIMPPL framework attributed approximately 23% more LUTs and 16% more flipflops of overhead to the system design, where the resource usage of the controllers is independent of the PE's size. The designers required only 4% of the overall design time to adapt the inital PE designs into CEs, which could be thoroughly verified on-chip with large data packet streams to ensure that they exhibited the correct behaviour. The system-level integration time was 12 hours, less than 1% of the total system design time.

This study examined the numerous system-level issues and proposed an effective method of verifying and integrating CEs at a coarse granularity, however, future work will need to address system-level debugging at finer granularity. While the proposed testbed is useful for verifying that data packets flow correctly through the system, it provides no method of detecting internal errors to the CE or PE if the data is processed incorrectly. Since this system is implemented as a pipeline, it facilitates the detection and location of the source of CE-specific design errors. However, the SIMPPL model also supports branching systems where verifying the correct global level operation is more difficult. The complexities of these system architectures compounds the need to obtain CE specific debugging information. Thus, creating a tool that allows the user to obtain CE specific debugging information on-chip is the next phase of research.

7. REFERENCES

[1] H. Chang, L. Cooke, M. Hung, G. Martin, A. J. McNelly, and L. Todd, *Surviving the SOC revolution: A Guide to Platform-Based Design*. Norwell, Massachusetts: Kluwer Academic Publishers, 1999.

[2] M. Keating and P. Bricaud, *Reuse Methodology Manual for System-on-a-Chip Designs*. San Francisco, California: Kluwer Academic Publishers, 1998.

[3] L. Shannon and P. Chow, "Simplifying the Integration of Processing Elements in Computing Systems using a Programmable Controller," in *IEEE FCCM. Symp.*, Apr. 2005, pp. 63–72.

[4] G. Kahn, "The Semantics of a Simple Language for Parallel Programming," in *Proc. of the IPIF Congress 74*, 1974.

[5] E. Lee and T. Parks, "Dataflow Process Networks," *Proceedings of the IEEE*, vol. 83, no. 5, pp. 773–799, May 1995.

[6] "CS6651: Amphion MPEG2 Video Decoder for FPGA," online: http://www.amphion.com/ cs6651.html.

[7] J. Mitchell, W. Pennebaker, C. Fogg, and D. Legall, *MPEG Video Compression Standard*. London, UK: Chapman & Hall Ltd., 1996.

[8] "SystemC Home Page," http://www.systemc.org.

[9] "Ptolemy Home Page," http://ptolemy.eecs.berkeley.edu.

[10] L. Shannon and P. Chow, "Maximizing System Performance: Using Reconfigurability to Monitor System Communications," in *IEEE Int. Conf on FPT*, Dec. 2004, pp. 231–238.

[11] T. Rissa, W. Luk, and P. Cheung, "Automated Combination of Simulation and Hardware Prototyping," in *Int. Conf. on Eng. of Reconfig. Systems and Algs*, June 2004.

[12] K. S. Hemmert, J. L. Tripp, B. Hutchings, and P. A. Jackson, "Source Level Debugger for the Sea Cucumber Synthesizing Compiler," in *IEEE FCCM Symp.*, Apr. 2003, pp. 228–237.

[13] "MEDEA+ EDA Roadmap 2003: Executive Summary Europe," online: http://www.medea.org/webpublic/ publications/publ_relation_eda.htm.