Abstract—This paper presents a parameterized system-level design framework, which enables rapid and powerful research for hybrid multicore architecture exploration and hardware/software co-design. The framework comprises the component-based hardware design and application compiler, which make it easy for a designer to build stream-oriented applications with FPGA-based hybrid multicore architectures. The high modularity and parameterization of the framework supports fast multicore architecture exploration of different topologies, routing schemes, processor types, customized hardware processing units and memory system organizations. The compiler tool chain is used to map C/C++ based applications onto the soft processing units. Experimental results targeting the JPEG encoding application demonstrate the feasibility and performance improvement of this framework.

I. INTRODUCTION

Multicore system has emerged as the standard for embedded system design which can provide high performance required by today’s multimedia and communication applications. Although the FPGA based multicore system design provides customization for topologies, processors, custom hardware processing units, memory size, interconnection structures and other parameters, these flexibilities enlarge the design space and create the difficulties in identifying the specific architecture and a suitable mapping method. Thus the design of these systems requires tuning of a large number of parameters in order to find the most suitable multicore architecture for a target application domain, in terms of performance, resource utilization, and energy consumption. This increases the complexity of multicore system design. This paper is arranged as follows. Section II reviews related work. Section III gives overall design and provides experimental results. Finally, Section IV presents conclusion and future work.

II. RELATED WORK

Jason Wu et al. [1] presented a system level design flow for design space exploration. Both their design flow and our framework could provide designers information on the implementation of alternative architectures whereby the tradeoffs among resource utilization, design time, performance and so on. They only consider the number of the processing units. However, the throughput of a stream-oriented application is significantly impacted by not only the number of the processing units, but also the buffer size allocated for the communications between the processing units.

An effort related to ours is Heracles [2], which is comprised of soft hardware (HDL) modules, an application compiler tool chain, and a graphical user interface. Heracles is a component-based framework that gives researchers the ability to create complete, realistic, synthesizable, multicore architectures for fast, high-accuracy design space exploration. But it only presents homogeneous multicore architectures without any user custom hardware processing units.

For different topologies and interconnection structures, we refer to [3]. The author demonstrates a design flow of an FPGA-based multicore system for the JPEG encoding application and explores different interconnection structures for multicore systems. However, the author does not provide any experimental results.

III. OVERALL SYSTEM AND EXPERIMENTAL RESULTS

A. Overall System Framework

Fig. 1 shows an overview of our system level design framework which consists of three main parts: the software environment, FPGA-based hardware design and evaluation environment. The monolithic applications written in C/C++ language are directly compiled using GCC cross compiler and profiled by GNU profiler with the component library, after that the hybrid multicore design space exploration merges design constrains, component library and outputs of the compiler and profiler in the software environment flow. In general, the
optimizations of computations, communications and memory interfaces were addressed separately in the previous work [1][2]. The FPGA-based hardware design flow combines tradeoffs of resource utilization, design time, and performance together. At last, designers can implement the alternative design with the information provided by the previous flow and measure the performance and cost in order to verify the correctness of the framework.

B. Experimental Results and Analysis

The JPEG encoding application aims to demonstrate the framework onto an FPGA shown in Fig. 2. The algorithm mainly divides into four tasks: Color Conversion (CC), two-dimension Discrete Cosine Transformation (2D-DCT), Zig-Zag Scan Quantization (ZZQ) and Variable Length Coding (VLC). The 2D-DCT is identified as the critical task according to the profiling result which is about 62%. Thus the 2D-DCT was designed in hardware and as a co-processor. The above description is mainly about the computation, and we also add two processors P0 and P1 to optimize the communication of the whole structure. P0 scatters the source data for different links and P1 gathers the processed data in sequence.

The structure shown in Fig. 2 is implemented on Xilinx XC6VLX240T. The processing unit is MicroBlaze which can be customized by tuning parameters, and the communication interface among them is Xilinx FSL link. The stream in/out interfaces are directly interconnected to the memory controller for high throughput. The customized MicroBlazes and 2D-DCT co-processors execute at 100MHz, and the throughput of one link is 0.78 frame (1920*1080) per second. The timing sequence of one link is shown in Fig. 3. We can see that processor P0 reads the source data from RS0 port and scatters the data to W0. The black block RS0 and gray block W0 represent the number cycles of RS0 and W0 in Fig. 3. It is the same to other processors.

IV. CONCLUSION AND FUTURE WORK

This paper introduces a parameterized system-level design framework, which enables rapid and powerful research for hybrid multicore architecture exploration and hardware/software co-design. Experimental results targeting the JPEG encoding application demonstrate the feasibility and performance of this framework at 0.78 frame (1920*1080) per second per link. In the future, more applications will be explored by our framework.

REFERENCES


Fig. 2. The Overall System Structure

Fig. 3. The Timing Sequence of Link0