6B-4

Rapid FPGA Prototyping of a Queue Processor Core for Embedded Computing

BEN A. ABDERAZEK ,† TSUTOMU YOSHINAGA † and MASAHIRO SOWA †

In this research work, we present the prototyping of a processor core based on Queue architecture as starting point for application-specific processor design. While previously hardware prototyping required enormous investigation in design effort and special purpose emulators, an emulation approach based on high-density field-programmable-array (FPGA) devices now make hardware prototyping (emulation) practical and cost effective for embedded processor designs. We show how to perform rapid prototyping and optimizations to fully exploit the capabilities of the prototyped Queue processor core, while maintaining a common source base.

1. FPGA Prototyping

In our previous work[1),2)], we have researched about the design of novel queue processor architectures as a starting point for hardware/software design space exploration for general and embedded applications. The choice of this architecture was based on a number of factors, such as suitability for embedded applications[1),3)]. The processor core has been described in Verilog hardware description language (HDL). The design uses a modular design structure with control logic implemented as a set of communicating state machines[2)].

An event-based or cycle level simulation becomes increasingly inadequate to verify a significant execution trace for a given problem, this software-based simulation approaches may not allow all aspects of a design's functionality to be exercised. To solve this problem, we consider prototyping-based emulation. Rapid prototyping substitutes real time hardware emulation for slow simulator-based execution.

By reducing the logic capacity requirements for hardware emulation and using high-density FPGA devices, prototyping of new architectures (toward the development of hardware embedded processors) is becoming accessible. To facilitate such FPGA prototype implementations, we have investigated how to describe the processor architecture to achieve good synthesis results for FPGA implementation. To do so, we have continued our previous research for synthesisable core code (SCoC) with the focus on restricting source code modification for efficient FPGA synthesis to only few modules[2)]. Such optimizations allow the implementation of a medium complexity core in a single high-density FPGA device, significantly reducing the investment in emulation hardware and support tools.

† Graduate School of Information Systems, The University of Electro-Communications, Tokyo, Japan

2. The Pipeline of the PQPpfB Core

In many processors, the control unit is centralized and controls all central processing core functions, introduces pipeline stalls, bubbles, etc. However, especially for pipelined architecture, this control unit is one of the most complex part of the design, even for processors with fixed functionality. When the architecture is designed to be extensible, designing such a state machine is not feasible without some partitioning. In this design, we have decided to break up the unstructured control unit to small, manageable units. Instead of a centralized control unit, which aims to control the complete data path, the control unit is integrated with the pipeline data path. Thus, each pipeline stage is mainly controlled by its own, simple control unit.

3. PQPpfB Core Verification

Verification of the processor is performed at multiple abstraction levels by integrating several approaches. The aim is to validate the functionality as provided by the processor and used by application programs, as well as the correct implementation in hardware. The design verification of the PQPpfB processor core is performed in three phases which are based on the same HDL that describes the core:

Functional simulation: To simplify the verification process, we have developed a front end which displays the pipeline state, the queue-register unit and other information that may be necessary to verify the functionality of the processor model.

Rapid prototyping While simulation allows

very accurate tracing of each individual signal in a design, simulation speeds are too slow to simulate significant programs. Thus, we have decided to developing an FPGA prototype in order to allow real time testing of the PQPpfB core and before actual fabrication.

Gate level simulation To ensure correctness of low-level implementation details, interfaces and timing, we use gate-level simulation for ensuring compliance with design specifications. We specified timing and other constraints using a unified user constraints file (UCF) with the core modules. With gate level simulation, we have validate the actual timing and provide details information about the final prototyped PQPpfB core design (see $ref.2$ for details).

4. PQPpfB Core Synthesis

To better estimate the impact of the description style on FPGA efficiency and to improve key design units, we have explored logic synthesis for FPGAs. The focus of this operation was to optimise critical design parts either for speed or resource utilization. The optimizations of Verilog HDL models for FPGAs are grouped into two distinctive approaches: (1) Adapted coding style and (2) Usage of special-purpose devices on FPGA. In this research work, our prototyping experiments and the results described are based on the Altera Stratix architecture for both area and speed optimizations. We have used the Quartus II design tools for synthesis, simulation, placement and routing and vendor-supplied software for configurations.

5. PQPpfB Core Coding Style

The whole processor is defined by a register transfer level (RTL)-like block diagram. The core is split into small RTL modules. The choice of this coding style was based on the fact that simpler building units are easy to maintain, debug, and modify if desired. Since the building blocks have been developed following the standard IEEE Verilog subset for synthesis, the resulting model is completely synthesisable.

6. FPGA Prototyping Results

The floorplan of the placed and routed PQPpfB processor core is illustrated in $Fig.1$. In $Fig.2$, the achievable processor frequency over qRegister size variation is illustrated. Logic area and power consumption were also investigated; however for space limitation, they are not discussed in this letter.

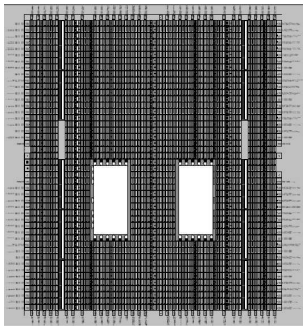

Fig. 1 Floorplan of the placed and routed PQPpfB processor core. The pipeline stages have been placed from the top right to the lower left corner of the target Stratix EP1S25F1020C5 FPGA device.

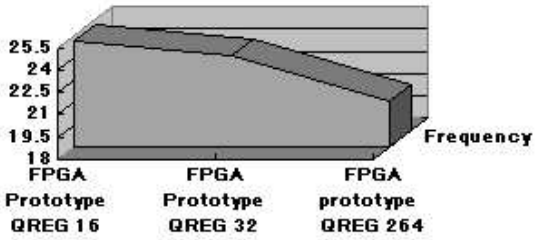

Fig. 2 Achievable processor frequency over qregister size variation.

7. Concluding Remarks

In this work we have introduced a synthesizable queue core. The design of this core is based on communicating modules and serves as the starting point for the development of special purpose applications we are targeting. The processor successfully fits on a single Stratix FPGA device, thereby obviating the need to perform multi-chip partitioning which results in a loss of resource efficiency. Only, few clearly identified such as the Barrier and the qRegister units need to be specially optimized at the HDL source level to achieve efficient resource usage.

References

1) B. A. Abderazek, M. Sowa, and al., " Queue Processor for Novel Queue Computing Paradigm Based on Produced Order Scheme", Proc. of IEEE CS, 0-7695-2138-X/04, pp. 169-177, July 2004.
2) PQPpfB project: http://www.sowa.is.uec.ac.jp
3) M. Sowa, B. A. Abderazek, T. Yoshinaga," Parallel Queue Processor Architecture Based on Produced Order Computation Model", to appear in the Int. HPC Journal of High-Performance Computer Design,(2005).