Hardware accelerated approach for floating-point multiplication on 32-bit pipelined RISC-V processor

Udit Khanna* Manoj Sharma** Ankita Saxena***

Abstract: Implementing hardware support for all extensions of the RISC-V Instruction Set Architecture inside a processor would lead to avoidable area and power consumption for applications that rarely utilize a particular extension. In this paper, authors have first suggested a modified 3-stage pipeline alternative to the ZSCALE processor (32-bit) by UC Berkeley. Subsequently a hardware-accelerated approach for implementing the single-precision floating-point extension (‘F’) on programmable logic interfaced with the processor is implemented. For clear demonstration of the suggested approach, an IEEE 745-2008 compliant multiplication unit, optimized for performance and area using ancient Vedic Mathematics techniques, has been used. Adducing this, the proposed processor and the accelerator units have been implemented as a system on Xilinx Virtex-6 ‘xc6vlx75t-3ff484’ FPGA platform. A maximum operating-frequency of 335 MHz has been observed for the modified standalone processor.

Keywords: RISC-V Instruction Set, Hardware Acceleration, IEEE 754-2008 Floating point standard, Vedic Mathematics, Pipelined processors.

1. INTRODUCTION

Initiatives to produce open-source industry-competitive processors have been gaining momentum since the OpenRISC platform came into existence. The recent introduction of the open-source RISC-V instruction set by UC Berkeley has further boosted the efforts in this direction. In one such effort, UC Berkeley has released a 32-bit single-issue in-order core by the name of Z-Scale, which is similar in spirit to the ARM Cortex M0 core [1]. Another development is the PULP (Parallel Ultra Low Power) project led by ETH Professors, which is a quad-core RISC-V compatible processor designed for battery-powered IoT applications [2]. A noteworthy mention is deserved by the SHAKTI project at Indian Institute of Technology, Madras, aimed at breaking the barrier between industry and Academia in the field of state-of-the-art processor designs [3].

Since the RISC-V instruction set includes optional extensions for single-precision and double-precision floating point arithmetic, processors based on this ISA can make use of hardware acceleration. In this approach, the support for the optional extension is implemented on specialized hardware such as programmable logic or FPGA that can speed up the computation and can be powered when required, and thus offload the processor pipeline. The second section of the paper detailed the proposed processor architecture. The interfacing and implementation of hardware accelerator for the RISC-V floating-point multiplication instruction is discussed in third section.
2. PROPOSED PROCESSOR ARCHITECTURE

The proposed architecture is a modification of the Z-Scale processor, leading to a simpler implantation. Fig. 1 shows the 3-stage pipeline of Z-Scale processor, consisting of the PC Generation coupled with Instruction Fetch stage, followed by the coupled Instruction Decode and Execute stage, and finally the Data Memory coupled with Write-back stage. Currently, the Z-Scale processor does not include on-chip support for the F-extension of RISC-V ISA which consists of floating-point instructions.

Fig. 1. The Zscale Pipeline

Since the RISC-V is a Load/Store instruction set, a modified architecture is proposed in which the Execute unit and Memory unit form a single stage in parallel because an instruction could either use the ALU or it could use the Data Memory while traversing the pipeline. Thus, the unnecessary passing of arithmetic and logical instructions through the data-memory memory stage is avoided. However, in order to calculate the effective address for Load/Store operations, there arises the need to put an extra 32-bit adder in the decode stage.

Fig. 2. The proposed pipeline.

Further, in order to avoid the area and power used by control-hazard detection logic, it is proposed to couple the Instruction Fetch and Decode into a single pipeline stage. In this manner, a simpler design is created that avoids pipeline stalls related to various Branch and Jump instructions.

Next, in order to be able to extend support for floating-point instructions in an efficient manner, an optional generic accelerator-interface is proposed. In case of a soft-core implementation of the processor on FPGA, the interface can be generated at the time of synthesis if support for floating point instructions is required.

3. HARDWARE ACCELERATED FLOATING POINT MULTIPLIER

3.1. RISC-V Floating Point Multiplication Instruction

The F-extension of RISC-V mandates a stack of 32 floating-point registers, separate from the integer register stack, and a dedicated control & status register ‘fcsr’. For single-precision mode, each of these registers is 32 bits wide. Transfer of floating-point values between memory and registers takes place through dedicated load and store instructions. Fig.3 summarizes the floating-point arithmetic instructions in the RISC-V ISA.

FMUL is the multiplication instruction, which consists of the 7-bit ‘opcode’ for floating point instructions, 5-bit destination register ‘rd’, source registers ‘rs1’ and ‘rs2’, 3-bit rounding-mode ‘rm’, another 5-bit field funct5 to specify the arithmetic operation and a 2-bit field ‘S’ to specify the mode of operation (single- or double-precision).
3.2. Interfacing the processor with the accelerator

The floating-point multiplication logic on FPGA is connected to the RISC-V processor through the accelerator-interface depicted in Fig.4.

When an instruction that requires an application-specific accelerator is detected in the Decode unit of the processor, the Enable signal on the accelerator-interface is asserted and the complete 32-bit instruction-word is passed to the accelerator for further decoding. Along with the instruction-word, the two 32-bit operands are also passed. In the case of floating-point multiplication, the operands are fetched from the floating-point register stack residing inside the processor.

Once the instruction is successfully decoded in the accelerator, it asserts the ‘Busy’ signal that stalls the processor pipeline until the accelerator operation is complete and/or an anomaly is raised in form of a 4-bit ‘exception’ signal that may be interpreted differently for each instruction. The RISC-V ISA supports the following floating-point exceptions:

<table>
<thead>
<tr>
<th>Table 1. RISC-V Floating Point Exceptions</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Exception</strong></td>
</tr>
<tr>
<td>NV</td>
</tr>
<tr>
<td>DZ</td>
</tr>
<tr>
<td>OF</td>
</tr>
<tr>
<td>UP</td>
</tr>
<tr>
<td>NX</td>
</tr>
</tbody>
</table>
On completion of operation, the accelerator outputs the 32-bit result and asserts the ‘Complete’ signal, following which the processor can continue to write the result in the correct destination register. Exceptions in RISC-V ISA are to be written to the ‘fcsr’ register. Figure 5 summarizes the processor pipeline stages including the accelerator operation.

3.3. Multiplication Algorithm

Based on the Urdhwa-Triyagbhyaam sutra, also known as vertical and crosswise method, adopted from ancient Vedic-Mathematics, a 32-bit single precision Floating-Point multiplier has been adopted from the work of Kumar and Lall [5] to implement the arithmetic function of hardware accelerator. The use of this multiplier results in an average pipeline-stall of 6-cycles.

4. EXPERIMENTAL RESULTS

The proposed processor has been designed using Verilog HDL on Questasim HDL Simulator by Mentor Graphics and synthesized on Xilinx Virtex-6 ‘xc6vlx75t-3ff484’ FPGA. Synthesis results yield a clock period of 2.985 ns for the standalone processor, conversely a 335 MHz frequency of operation. The device utilization summary after synthesis of the entire system is tabulated in Table 2 and the RTL Schematic of the pipelined processor is shown in Figure 6.

Table 2. FPGA Device Utilization Summary.

<table>
<thead>
<tr>
<th>Logic Utilization</th>
<th>Used</th>
<th>Available</th>
<th>Utilization</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of Slice Registers</td>
<td>142</td>
<td>93120</td>
<td>~0%</td>
</tr>
<tr>
<td>Number of Slice LUTs</td>
<td>502</td>
<td>46560</td>
<td>1%</td>
</tr>
<tr>
<td>Number of fully used LUT-FF pairs</td>
<td>122</td>
<td>522</td>
<td>23%</td>
</tr>
<tr>
<td>Number of bonded IOBs</td>
<td>185</td>
<td>240</td>
<td>77%</td>
</tr>
<tr>
<td>Number of BUFG/BUFGCTRLs</td>
<td>2</td>
<td>32</td>
<td>6%</td>
</tr>
</tbody>
</table>
5. LIMITATIONS AND FUTURE SCOPE

The proposed system for hardware acceleration is currently based on a pipeline-stall method. Although this method consumes less area than an on-chip execution unit and is also less complex than a diversified pipeline, it negatively impacts the Cycles per Instruction (CPI) metric of performance. A better system may be designed using hardware-interrupts wherein the processor can continue to do other useful work while the accelerator completes its operation. Some improvement in throughput may also be achieved with an out-of-order execution pipeline, similar to the RISC-V Berkeley Out-of-Order Machine (BOOM).

In future, the accelerator may be used for other custom-extensions to the RISC-V ISA, such as dedicated instructions for Digital Signal Processing. For instance, a RISC-V core in a wearable camera based device could interface with an accelerator on configurable logic for image processing applications and for Fast Fourier transform computation. Additionally, the accelerator could be used in security or ciphering applications such as AES encryption and decryption.

6. CONCLUSION

In the current paper authors have presented a novel architecture for a simpler implementation of the Z-Scale pipeline along with a generic accelerator-interface for offloading certain tasks from the processor execution-unit. Based upon this new design-paradigm, various compute-intensive tasks can be performed on programmable logic attached to the processor. This would be particularly useful in emerging IoT applications that would require extension of the RISC-V instruction set.
7. REFERENCES


6. Rui Jia, Colin Yu Lin; Zhenhong Guo; Rui Chen; Fei Wang; Tongqiang Gao; Haigang Yang, “A survey of open source processors for FPGAs”, 2014 24th International Conference on Field Programmable Logic and Applications (FPL), 2-4 Sept. 2014, Page(s): 1 – 6, DOI: 10.1109/FPL.2014.6927482
