RISC-V Out-Of-Order Data conversion Co-processor

Aneesh Raveendran, Vinayak Patil, Vivian Desalphine, Sobha PM, A David selvakumar
{raneesh, vinayakp, viviand,sobhapm, david} @ cdac.in
Centre for Development of Advanced Computing, Bangalore, INDIA

***Abstract*— Data conversion operations are important and essential part of floating point units in a processor and typical instructions include conversion between various precisions, integer to floating point and vice versa, floating point to fixed point and vice versa etc. Besides few processors have instructions to round and truncate data, sign injections, move data between co-processors registers and general purpose registers and vice versa etc. Conversion operations are part of ARM, MIPS, and RISC-V etc. instruction sets. The conversion functionalities can be part of hardware or may be implemented in software. This paper details an architectural exploration for data conversion co-processor for RISC-V [1] data conversion instructions. The Floating point conversion unit that has been designed with RISC-V data conversion instructions is fully compatible with IEEE 754-2008 standard as well. The floating point conversion co-processor is capable of handling both single and double precision floating point data operands in out-of-order for execution and in-order commit. The front end of the data conversion processor accepts three data operands, rounding mode and associated Op-code fields for decoding. Each conversion operation is tagged with an instruction token for in-order completion and commit. The output of the floating point unit is tagged with either single or double precision results along with floating point exceptions if any, based on the RISC-V instruction set. The co-processor for data conversion integrates with integer pipeline. The proposed architecture for data conversion co-processor with out-of-order execution, in-order commit and completion / retire has been synthesized, tested and verified on Xilinx Virtex 6 xc6vlx550t-2ff1759 FPGA. A performance throughput of 350MFLOPs (data conversion operations) per second have been observed.**

***Index Terms*—IEEE 754-2008 floating point standard, floating point co-processor, RISC processor, RISC-V instruction set, Out-of-order and Microprocessor.**

I. INTRODUCTION

The necessity of integer and floating point data conversion operations in some stages of computation on a digital computer is a major component in practical numerical computation. C99 standard for arithmetic operations specifics several floating point conversion operations as macros to enhance the floating point computation performance. The data conversion functionalities may be on hardware or software. To enable the hardware based data conversion functions, instruction sets do specify special purpose instructions as part of Floating Point Instruction Sets [1]. The combined advantages of higher clock speed, low power, area efficiency and operation specific design possibilities make standalone instruction based data conversion operations more ubiquitous.

We sincerely thank Department of Electronics and Information Technology (DEITY), MCIT, GoI, and Dr. Sarat Chandra Babu, Executive Director, C-DAC, Bangalore for the continuous support, encouragement and the opportunity to work on the topic of this paper.

Floating point conversions follows IEEE-754 2008 standard, which specifies four set of floating point number formats: - single, double, extended and quad precision floating point formats. Fig. 1 shows the bit-wise representation of single and double precision floating point number.

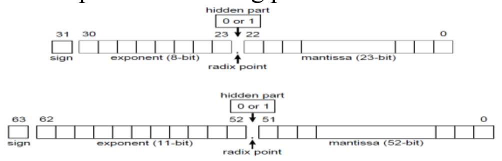

Fig1: Single and Double precision floating point number representation

II. PIPELINE ORGANIZATION OF RISC-V PROCESSOR WITH OUT-OF-ORDER DATA CONVERSION UNIT

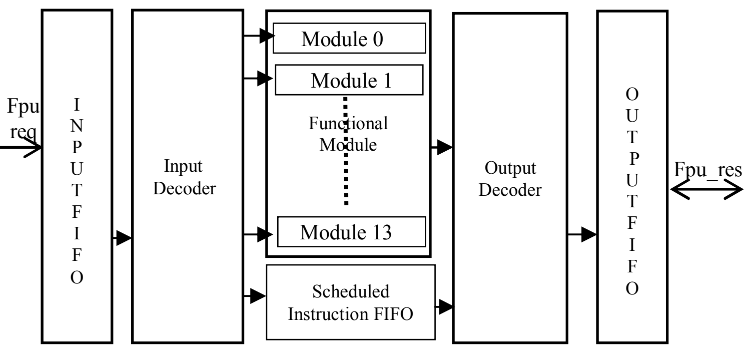

Fig 2: Architecture of Out-of-order conversion Co-processor

Fig. 2 shows top level architecture of out-of-order pipelined data conversion, movement and sign-injection co-processor. Floating point unit accepts FPU requests and generates the responses. To make use of data conversion out-of-order, conversion modules are developed individually, integrated as a single module with specific token for instruction scheduling.

A. FPU request

0……..31……..63	0…….6	0…….6	0……..4	0 1 2	0 1 2
Operand (SP/DP)	Op code	F7	F5	F3	RM

Fig 3: FPU request packet

Input request packet contains operand for conversion with op-code and function fields. Fig 3 shows the FPU request packet format. Each FPU request is stored in input FIFO of depth 2 which is further read by input decoder. Input operand width is fixed to 64-bit which holds, either a 32-bit (single precision or integer) or 64-bit (double precision) floating point or integer data for conversion.

B. Input Decoder

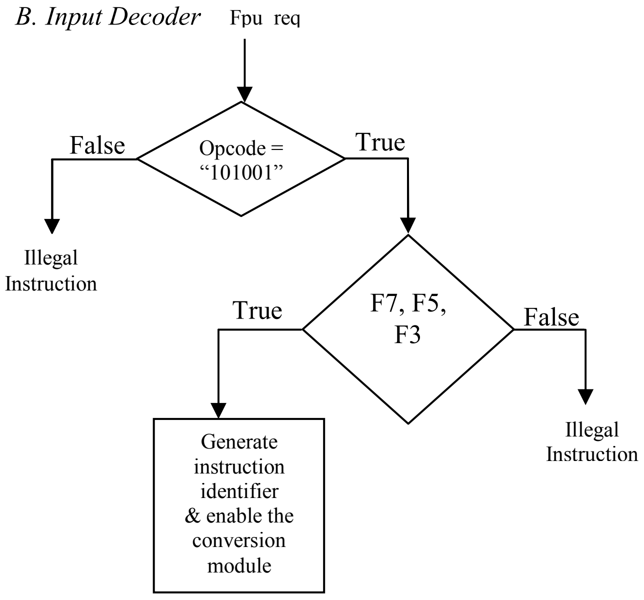

Fig 4: Flow diagram of Input decoder

Fig. 4 depicts the flow diagram of input decoder. Input Decoder in floating point pipeline reads the FPU request packet from the Input FIFO, decodes the floating point operation based on the op-code, F7, F5 and F3 fields. After decoding, corresponding individual hardware module is enabled and instruction token for the specific module is stored in the scheduled instruction FIFO. Illegal floating point operations are tagged with illegal token and stored in the scheduled instruction FIFO.

C. Scheduled Instruction FIFO

Scheduled instruction FIFO stores the instruction tokens, which specifies present operations in the pipeline. Scheduled instruction FIFO has a depth of 6, which is equal to the maximum number of pipeline stages in the co-processor.

D. Functional Module

Functional modules, comprises of 10 individual conversion functions, a move and two sign injection operation module. Operation results and exceptions are raised according to the IEEE 754-2008 standard.

E. Output Decoder

Fig. 5 shows the flow diagram of output decoder. Output decoder commits the oldest instruction from pipeline. To ensure the in-order commit, output Decoder in the pipeline reads the scheduled instruction from the top of the scheduled instruction FIFO and polls for the completion signal from the matched individual module. Once the conversion is completed, the result (32-bit or 64-bit), exceptions (5-bit) from the module are read and stored in output FIFO which has a depth of 2. On completion of conversion, movement or sign injection operation, first token from the top of the instruction scheduler FIFO is removed.

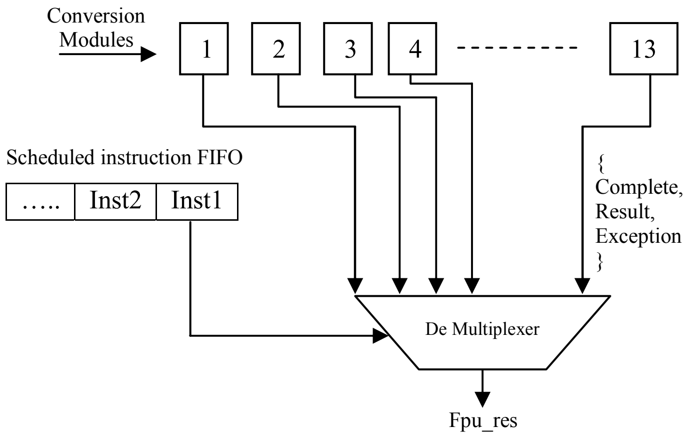

Fig 5: Flow diagram of Output decoder

F. FPU response

Output FPU response packet holds maximum 64-bit result which can be of either 32-bit (single precision floating point or integer) or 64-bit (double precision floating point) result. Exceptions from each individual module are collected and stored in floating point Control and status registers.

III. Verification, Analysis of Out-Of-Order core

The conversion, move and sign injections sub-modules are developed using VHDL, simulated by using ModelSim and synthesized using Xilinx ISE 14.1. The conversion co-processor is primarily implemented and tested on Xilinx Virtex 6 "xc6vlx550t-2ff1759" FPGA. Conversion co-processor is verified by pseudo random verification at Floating Point Test bench level.

For pseudo- random verification at test bench level, the input test vectors are generated using a high level language ("c") and applied to the Device Under Test (DUT) and also to the MATLAB tool. The MATLAB tool generated outputs are considered as the golden output and are compared against the outputs from the DUT. The corner cases are manually applied to the DUT to verify the functionality. The implemented individual conversion modules are operating at 350 MHZ and integrated conversion co-processor achieves a frequency of 350 MHZ on Xilinx Virtex 6 "xc6vlx550t-2ff1759" FPGA.

IV. Conclusion

This research article analyzes the need for the hardware based floating point data conversion operations and proposes a novel architecture for out-of-order data conversion co-processor compatible with RISC-V instruction set. Conversion co-processor is fully compliant with the IEEE754-2008 standard and achieved a throughput of 350MFLOPS on Virtex 6 FPGA.

References

[1] A. Waterman, Y. Lee, D. A. Patterson, and K. Asanovic. The RISC-V Instruction Set Manual,Volume I: Base User-Level ISA. Technical Report UCB/EECS-2011-62, EECS Department, University of California, Berkeley, May 2011.