Instruction based validation of Arithmetic Operations on AJIT processor

Asha Devi
Prof. Madhav P Desai
Department of Electrical Engineering
Indian Institute of Technology
Mumbai 400076 India

Abstract

AJIT processor is a SPARC V8 Instruction Set Architecture (ISA) complient processor developed at IIT Bombay.This document describes the set of tests done for instruction based validation of arithmetic operations implemented on the processor.The tests are performed on the following operations:

- Integer Operations
  - Unsigned Integer Operations : addition(add),subtraction(sub),multiplication (umul)and division(udiv)
  - Signed Integer Operations : multplication(smul) and division(sdiv)
- Floating point Operations
  - Single Precision Floating Point operations : addition(fadds),subtraction(fsubs), multiplication(fmuls),division(fdivs) and square root(fsqrts)
  - Double Precision Floating Point operations : addition(faddd),subtraction(fsubd), multiplication(fmuld),division(fdivd) and square root(fsqrtd)
  - Integer to Float Conversion :Integer to single (fitos),Integer to double(fitod)
  - Float to Integer Conversion :Single to Integer (fstoi),Double to Integer (fdtoi)
  - Conversion between floating point formats :Single to Double (fstod) ,Double to single (fdtos)

Validation Scheme

We describe the validation of arithmetic operations of a processor conforming to the SPARC-V8 Instruction Set Architecture (ISA) based on the SPARC Architecture Manual, Version 8 [1]. In order to validate arithmetic operations over the entire range of integers/floats ,the concept of generating random test cases has been used. Linear Feedback Shift Register (LFSR) has been used for generation of pseudo-random numbers [2]. Comparing result of individual random tests with host machine requires huge computational effort as well as occupies enormous memory space. A signature capturing information of each result from all random tests is generated instead. Since, signature captures the history of all test cases, matching of signature with host machine ensures the correctness of individual tests and the validation of the arithmetic operations.
The psuedo code used for testing is described below:

- Initialize a signature say s=0

- then $for(count = 0; count \leq maxcount; count++)$

  {
  - Generate two operands using pseudo random number generator(LFSR)
  - Compute the result for each operation :
    add = a + b ;
    mul = add * b ;
    sub = mul - add ;
    div=mul ÷ sub ;
    sqrt=$\sqrt{}$div;

  - Update the signature s = s xor sqrt
  - Allocate signature to memory

  }

- Compare the final signature with a signature computed on a host machine(Intel)

Note:
1. For all integer operations - a,b are 32 bits except for signed and unsigned division where a is 64 bit.
2. maxcount = 10 million for C ,1 million for FPGA model,10 thousand for Aa model.

LFSR Implementation

Fibonacci Linear Feedback Shift Register (LFSR) has been used for generation of pseudo-random numbers.The 32-bit Fibonacci LFSR equation is given below:
a=((((a≫31)^(a≫6)^(a≫4)^(a≫1)^a)&1)≪31)|(a≫1)

Test Implementation

Tests are located in following directories:

- Unsigned integers(add,sub,umul,udiv) in :

  AjitRepoV2/processor/C/validation/C_tests/arithmetic_operation/unsigned_integer_operation

- Signed multiplication(smul) in :

  AjitRepoV2/processor/C/validation/C_tests/arithmetic_operation/signed_mul

- Signed division(sdiv) in :

  AjitRepoV2/processor/C/validation/C_tests/arithmetic_operation/signed_div

- Floating point single,double precision and conversions(fadds,fsubs,fdivs,fmuls,fsqrts,faddd,fsubd, fmuld,fdivd,fsqrtd,fitos,fitod,fstoi,fdtoi,fstod,fdtos) in :

  AjitRepoV2/processor/C/validation/C_tests/arithmetic_operation/fp_final_all_check

Results

Operation	**FPGA model (1 million)**	**C model (10 million)**	**Aa model (10 thousand)**
unsigned integer operations	pass	pass	pass
signed integer operations	pass	pass	pass
floating point single,double precision	pass	pass	pass

Conclusion

The result generated by the integer and floating point unit is found as expected.

References

[1] The SPARC Architecture Manual Version 8. [Online]. Available: http://www.sparc.org/technical-documents-test-2/specifications/#ARCH.
[2] https://www.xilinx.com/support/documentation/application_notes/xapp052.pdf