Comparison of pipelined IEEE-754 standard floating point adder with unipipelined adder

Kavita Khare1,*, R P Singh1 and Nilay Khare2

1Department of Electronics and Communication Engineering, MANIT, Bhopal 462 007
2CSE & IT Department, University Institute of Technology, Rajiv Gandhi Prodyougiki Vishvavidyalaya, Bhopal

Received 03 March 2004; revised 25 December 2004; accepted 04 April 2005

Many Digital Signal Processing (DSP) algorithms use floating-point arithmetic, which requires millions of calculations per second to be performed. For such stringent requirements, design of fast, precise and efficient circuits is the goal of every VLSI designer. This paper presents a comparison of pipelined floating-point adder compliant with IEEE 754 format with an unpipeined adder also compliant with IEEE 754 format. It describes the IEEE floating-point standard 754. A pipelined floating point adder based on IEEE 754 format is developed and the design is compared with that of an unpipeined floating point adder and a rigorous analysis is done for speed, area, and power considerations. The functional partitioning of the adder into four distinct stages operates simultaneously for different serial input data stream. It not only increases the speed but also is energy efficient. All these improvements are at the cost of slight increase in the chip area. The basic methodology and approach used for VHDL (Very Large Scale Integration Hardware Descriptive Language) implementation of the floating-point adder are also described. Detailed synthesis report operated upon Xilinx ISE 5.2i software and Modelsim is given. The hardware design is implemented on Spartan IIE FPGA chip.

Keywords: Floating point adder, IEEE floating point standard, Latency, Model-Sim, Pipelining, VHDL, Xilinx ISE 5.2i

Introduction

Floating-point adders are critically important components in modern microprocessors and digital signal processors. The architectures developed so far for floating point adders are based on sequence of significant operations: Swap, shift, add, normalize and round. Due to these operations, the overall process of addition slows down. Floating-point adders must be fast to match with the increasing clock rates demanded by deep sub-micron technologies, also they must be small for being used in parallel processing systems. Since, in the traditional adders, all the stages were performed with single clock cycle, but the frequency of this clock was restricted due to the circuit constraints. Hence, whenever, the addition of a large number of values was performed, the traditional floating-point adders proved to be inefficient. This latency could be overcome if the concept of pipelining in the simple adder is introduced. The floating-point adder had been subdivided into four stages, which were pipelined according to the proper timing sequences. The clock frequency, which could be used for these stages, could be higher as compared to the clock frequency used for traditional floating-point adder. Also, while the two inputs are being processed and passed on to subsequent stages, new inputs enter the initial stage and the cycle continues. This results in overall faster operation. In this paper, the floating-point algorithm is explained and floating point adder implementation using Very Large Scale Integration Hardware Descriptive Language (VHDL) is described. It further deals with the concept of pipelining and the enhanced capability of the floating-point adder. Simple floating-point adder and pipelined adder have been compared in terms of speed of operation and area on chip.

Single Precision Floating Point Representation

The IEEE single precision floating point standard representation requires a 32 bit word, which may be represented from 0 to 31, left to right. The first bit is the sign bit, S, the next eight bits are the exponent bits, 'E', and the final 23 bits are the fraction 'F':

S EEEEEEE EEEEEEEEEEEEEEEEEEEEEEEEEEE FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF

0 1 8 9 31

*Author for correspondence
Tel: 0755-2420777; Fax: 07552670538
E-mail: kavita_khare1@yahoo.co.in
In IEEE-754 format, the significant bit always takes on an implied ‘1’ for the most significant digit assuming that the value represented is normalized.

**Floating Point Algorithm**

The floating-point representation of a number includes three fields: Sign bit ($S_i$), exponent ($E_i$) and the mantissa ($F_i$). The algorithm for floating point addition/subtraction consists of the following steps:

(i) Load the inputs and check for exceptional inputs (NaNs, infinity and zero);
(ii) Align the mantissas (right shift the significant of the smaller operand);
(iii) Add or subtract the mantissas; and
(iv) Normalize the results and generate the exceptions (Fig. 1).

**Pipelining Technique**

Pipelining, a technique for achieving faster clock rates while sacrificing latency, offers an economic way to realize temporal parallelism in digital systems. To achieve pipelining, input process must be subdivided into a sequence of subtasks, each of which can be executed by specialized hardware stage that operates concurrently with other stages in the pipeline. The adder design pipelines the steps, comparing, swapping, shifting, addition, and normalization, to achieve a summation every clock cycle. Each pipeline stage performs operations independent of others. Input data to the adder continuously streams in.

**Implementing Pipelining in Adder through VHDL**

The various VHDL constructs used in addition to the very elementary ones are: Package, procedure, function and port mapping. A floating-point value, $N$, can be represented as

$$N = S \times R^E$$

where $S$ is a fixed value multiplied by a radix, $R$, to some power $E$.

The algorithm to add two numbers with this representation requires that the radix power $E$, be the same in power and sign. If so, the significant components, $S$, can be summed together while keeping the same radix to some power $E$, as the multiplier. If they are not same, adjustment of the significant is one prior to addition (Fig. 2).

The various VHDL modules developed are:

- **Adder_pckg.vhd**: Declares the various data types, functions and procedures used in the design.
- **Adder.vhd**: Consists of the various components, instantiations and their port mapping.
- **Load_check_comp vhd**: First stage of the pipeline. It performs the function of loading the operands, checking for the exceptional inputs, compares the exponents and generates the exponent difference.
Shifting.vhd: Second stage in the pipeline. Its function is to shift the mantissa according to the exponent difference value generated in the previous stage.

Add_sub.vhd: Third stage in the pipeline. It performs the basic addition or subtraction as required.

Normalize.vhd: Fourth stage in the pipeline. Its function is to convert the result into normalized form.

Rreg.vhd, Rreg_bit.vhd, Reg_bitvector.vhd, Reg_Boolean.vhd, Reg_exp.vhd, Reg_int.vhd, Reg_mantissa.vhd, Reg_mnt.vhd, Reg_optype.vhd: Describes the various registers used to interface the various stages.

Fig. 3—Block diagram of pipelined FP adder

Fig. 4—Block diagram of unpipelined FP adder

Fig. 5—RTL schematic of pipelined FP adder

Fig. 6—RTL schematic of pipelined FP adder

Fig. 7—RTL schematic view of unpipelined FP adder
Results

Both unpipelined and pipelined FP adders have been implemented in VHDL (Figs 3-7). On analyzing various summary reports of device utilization and timing (Table 1), the speed of operation of the pipelined adder was found three times more than that of unpipelined adder.

Conclusions

The speed of operation of pipelined adder has been found three times more than that of unpipelined adder. Also, the area required in the pipelined adder is less than that of the pipelined adder.

References