VIVIDSPARKS

PCU IP Core

Introduction

Floating Point (FP) arithmetic has become more widespread in wireless communications, military applications, aeronautical applications, 3D graphics simulations, aircraft design and signal processing[1], [2], [3], [4], [5]. However, FP number system provide less accurate, less dynamic range and more exceptions resulting in complicated Floating Point Unit (FPU) and power greedy processors. On other hand, POSIT number system offers much better accuracy, better dynamic range and fewer exception at half data width with respect to FP number system. Posits are a new form of number system invented by Gustafson in December 2016. The concept was first publicly shared as a Stanford lecture seminar [6] in February 2017. The first peer-reviewed posit journal paper [7] was published in June 2017. Since then, studies on POSIT correctness [8], [9], [11] accuracy, efficiency are performed by comparing to floats [10].

We have designed world's first Posit Co-processor Unit (PCU) which is equivalent to double precision FPU with POSIT<32,2> configuration. The PCU is the base architecture for SupersoniK, RacEr, FalKon and Tez IP cores. The beauty of our PCU is that circuit is much simpler, silicon efficient and high performance. Our PCU performs addition, subtraction, multiplication, division, square root and other operations about 4X faster with respect to state-of-the-art FPUs . These operations are quite frequent in FP applications.

Provided with core

Design Files	Encrypted RTL
Testbench	Verilog
Constraint Files	Not provided
Simulation Model	Encrypted Verilog model
Supported S/W Driver	N/A

Tested Design Flows

Design Entry	Altera Quartus 16.1
Synthesis	SAltera Synthesis

Supported Features

- Sum of Products
- Addition/Subtraction
- Multiplication
- Multiply Add/Subtract
- Two independent multiplications
- Increment, Decrement
- Two two's complement operations
- Sum of Reciprocals
- Sum of Inverse Square roots
- Sum of Divisions
- Sum of Square roots and
- Many more

PCU Operations

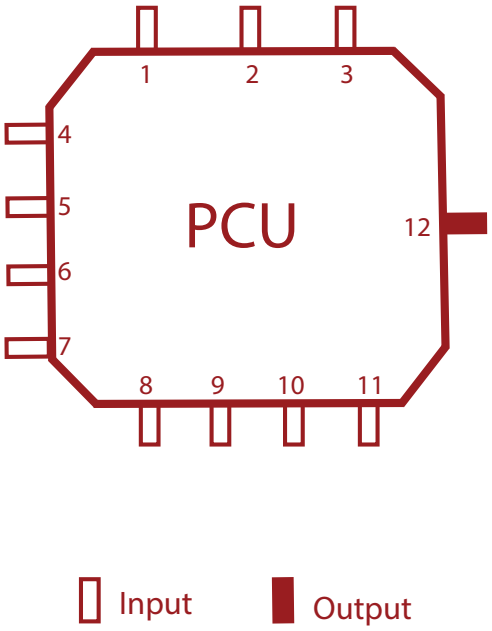
A + D, B + C	A × D + B × D = (A + B) × D
A - D, B - C	A × D - B × D = (A - B) × D
A × D, B × C	1/C, 1/D
A × D - B× C	1/Sqrt(C), 1/Sqrt(D)
A × D + B × C	-A, -B
A + (B × C)	1/D + 1/C
A - (B × C)	1/D - 1/C
A × D + B	1/Sqrt(D) + 1/Sqrt(C)
A × D - B	A++, B++, A--, B--
A, B compare operations	1-A, 1-B
A/D, B/C	Sqrt(D), Sqrt(C)
A/D + B/C	Sqrt(D) + Sqrt(C)
A/D - B/C	Sqrt(D) - Sqrt(C)
Integer to POSIT	POSIT to Integer

Comparisions

PCU	FPU
8 bit Scaling (Unbiased)	11 bit expoent (biased)
No normalization of inputs and ouputs	Normalized inputs and ouputs
Easier exception handling only 0 and ∞	Exception hadling for 0, ∞, -∞, overflow, underflow and NAN
No overflow and underflow	Overflow and underflow exist
Only one rounding mode: round to nearest	Four rounding modes
No sub normal numbers	Sub normal numbers exist
Shorter cycle time	Longer cycle time
Some power can be saved smaller numbers and higher performance for smaller numbers	Power and performance is same for all numbers
32 bit precision	64 bit precision

Port Description

Port numbers from 1 to 4 accepts four operands A, B, C and D having 32 bit data width. 31st bit is sign bit, remaining 31 bits are comprised of regime, exponent bits and fraction bits, these bits keep varying depending on operands, for more information see [6], [7], [8]. Port number 5 is clock signal, the PCU uses a single clock, all input and output interfaces and internal state are subject to this single clock. Port number 6, is reset signal. This signal is active-Low and must be asserted for one clock cycle to ensure correct operation and it is a global synchronous reset which resets all control states in the core; all data in transit through the core is lost when reset is asserted. Port number 7 is a control for Addition/Subtraction operation, active-High is for subtraction operation. By default, this port is set to active-Low. Port number 8 is to enable reciprocal and inverse square root operation, when this port is active-High, port number 9 chooses which operation to perform, port number 9 active-High is for inverse square root and active -Low is for reciprocal. By default Port number 8 is set to active-Low. Port number 10, three bits, are for compare operations, 001 is for <, 010 is for >, 011 is for<=, 100 is for >=, 101 is for == and 110 is for !=, by default this port is set 000 bits. Port number 10, two bits, is for conversion operations, 01 is for Integer to POSIT and 10 is for POSIT to Integer, by default this port is set to 00 bits. The PCU performs operations as mentioned in PCU operations table. Port number 12 is result output having 32 bit data width.



Latency

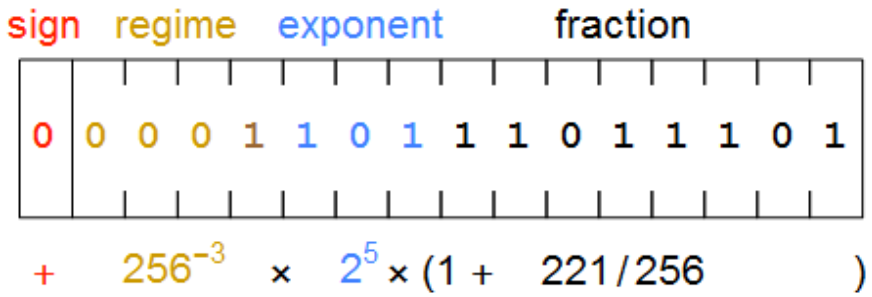
Latency of multiply add, sum of multiply add, sum of reciprocal, sum of inverse square root , division, square root, sum of division and sum of square root operations require 6 cycles where as addition/subtraction ,multiplication, reciprocal, inverse square root operations require 4 cycles, an extra 5th is required to compute division and square root operations. All other operations require 2 cycles. All operations guaranteed to POSIT rounded results.

Design Guidelines with PCU IP Core

A POSIT is made up of four components: sign, regime, exponent, and fraction. A POSIT is specified by its size in bits, *nbits*, and the maximum number of exponents bits, *es*. Suppose we view the bit string for a POSIT as a 2's complement signed integer, ranging from -2^{n-1} to 2^{n-1}. Let *k* be the integer presented by the regime bits, and *e* the unsigned integer represented by the exponent bits, if any. If the set of fraction bits is $\{f_1, f_2, ..., f_{fs}\}$ possibly the empty set, let *f* be the value represented by 1. $f_1 f_2 ... f_{fs}$. Then the value of a POSIT is defined by the following equation:

$$x = \begin{cases} 0, p = 0 \\ \pm\infty, p = -2^{n-1} \\ sign(p) \times useed^k \times 2^e \times f, \text{ all other } p \end{cases}$$

The following figure shows these fields for a 16-bit POSIT with 3 exponent bits, referred to as POSIT<16,3>.



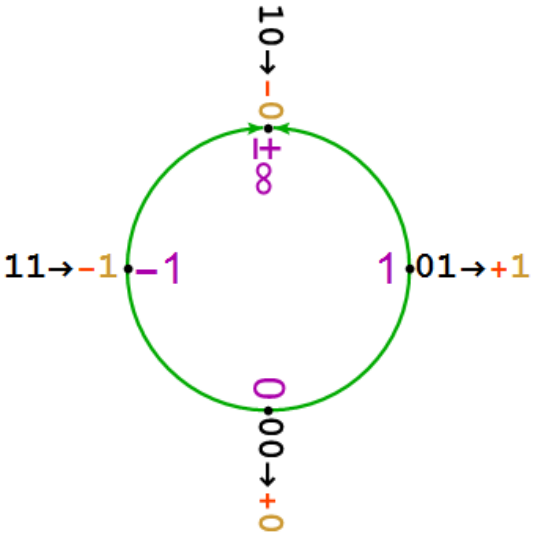
The sign bit 0, shown in red, implies that the value is positive. The regime bits have a run of three 0s terminated by the opposite bit 1, which implies the power of *useed* is -3. *Useed* is defined as $2^{2^{es}}$ and represents the scaling factor of the regime. In this example, the scale factor contributed by the regime is 256^{-3}. The exponent bits 101, shown in blue, represent 5 as an unsigned binary integer, and contribute a scale factor of 2^5. Finally, the fraction bit 11011101, shown in black, represent 221 as an unsigned binary integer, yielding a fraction value of 1.0 + 221/256. The value of this POSIT bit pattern is $477x2^{-27} \sim 3.55393 \times 10^{-6}$.

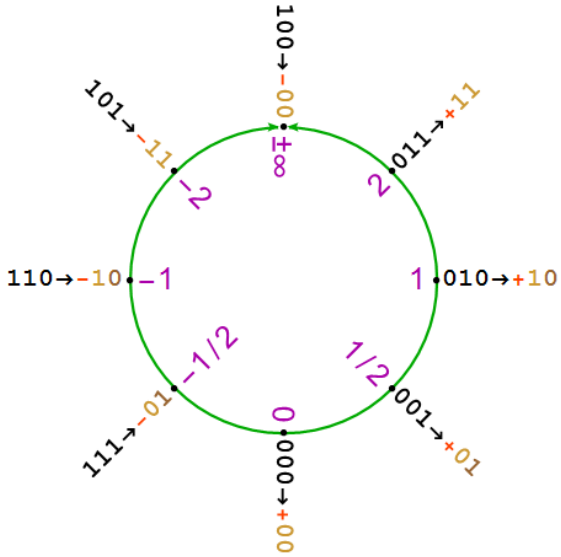
The size of the regime and exponent fields is variable creating a tapered precision real number system, with a dynamic range perfectly symmetric around 1. The minimum and maximum positive number for a POSIT configuration are called *minpos* and *maxpos*. Their values are a function of the scaling factor of the regime and the size of the POSIT:

$$\{minpos, maxpos\} = \{useed^{-nbits+2}, useed^{nbits-2}\}$$

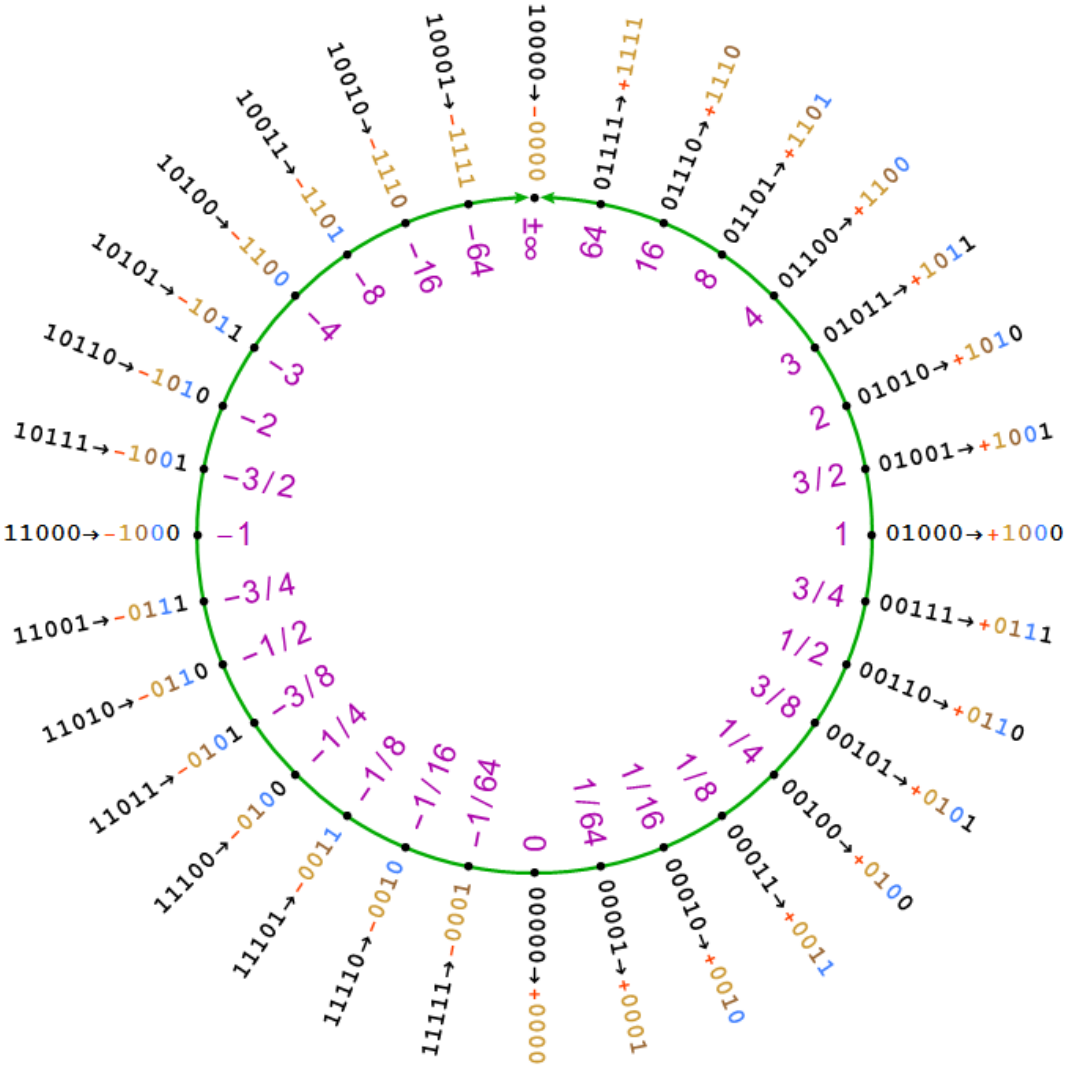
The ratio of *maxpos* to *minpos* is $useed^{2nbits-4}$, which defines the dynamic range of the POSIT. The POSIT format uses regime bits to raise *useed* to the power of any integer from *-nbits+1* to *nbits-1*, otherwise stated, the dynamic range of a POSIT is an exponential of an exponential of an exponential. This allows POSITs to create a larger dynamic range from fewer exponent bits than IEEE floats, leaving more fraction bits available to improve the precision of a value representation.

A wonderful way to visualize the structure of a POSIT configuration is to realize that they derive from Type II unums that mapped binary integers to the projective reals. Projective reals wrap the real number line onto a circle so that negative and positive infinity meet at the top.





The diagram on the left represents a 2 bit POSIT. We move to three bits by inserting a value between 1 and ±∞. It could be any real number greater than 1; it could be *2, 10, π, or googol*. The choice of this number *seeds* how the rest of the ring of unums is populated, to signify its importance this number was given the symbolic name *useed*. As we have seen above, for POSITs this value is set to $2^{2^{es}}$. Further bit expansion follows the rules that negation reflects about the vertical axis, and reciprocation reflects about the horizontal axis. The next figure shows a ring plot of values for a POSIT<5,1>:



Exceptions

In POSIT number system, there are fewer exceptions, as shown in the following table. Not A Real (NAR) exxeption is the numbers not real numbers. One important distinction between NaR and the IEEE 754 "NaN" concept is in comparisons. A NaR tests as equal to itself just like any other bit pattern. It maps to the most negative 2's complement signed integer, and if used it comparison operations, it tests as less than all other POSITs. In this way, there is no need for POSIT comparison operations. All the operations of 2's complement 32-bit integers suffice for comparison. There is no "unordered" concept with POSITs as there is with NaN.

Execptions	Flags
NAR	NAR
±∞	Invalid_OP
Divide by Zero	Invalid_OP

Conversion Wrappers

In order to interface with existing processors or IP cores, VividSparks provide conversion wrappers which converts number from POSIT number system to IEEE-754 Floating point number and IEEE-754 Floating point number to POSIT number system.

Implementation Results

The PCU with POSIT<32,2> configuration is implemented and exhaustively tested in Altera Aria 10 FPGA with device family 10AS027H3F34I2SG, the design is compliant with POSIT standards. Following table shows utilization report and frequency PCU is running. The PCU consumes only 15% of Adaptive Logic Modules (ALMs) and less than 1% of memory bits. In real VLSI design, the PCU consumes even lesser area. VividSparks have highly optimized design for tape-out.

Logic Utilization (in ALMs)	Registers	PINs	Block Memory Units	Frequency
15,626	1,190	234	68, 608	100 MHz

References

[1] Rad-Hard 32 bit SPARC V8 Processor, AT697E.

[2] S. Jacob and K. Padmakumar, "Floating Point Addition using low power CSA", Interantional Journal of VLSI and Embedded systems, vol. 5, Article 09450, Nov. 2014.

[3] K. Diefendorff, P. K. Dubey, R. Hochprung and H. Scales, "Altivec Extension to PowerPC Accelerates Media Processing", IEEE Micro, pp. 85-95, Mar./Apr. 2000.

[4] D. Harris, "A Powering Unit for an OpenGL Lighting Engine", Proc. 35th Asilomar Conf. Signals, Systems, and Computers, pp. 1641-1645, 2001.

[5] N. Ide et al.(Sony Playstation 2), "2. 44-GFLOPS 300-MHz Floating-Point Vector-Processing Unit for High-Performance 3D Graphics Computing", IEEE J. Solid-State Circuits, vol. 35, no. 7, pp. 1025-1033, Jul 2000.

[6] J. Gustafson and I. Yonemoto. (2017, Feb) Beyond floating point: Next generation computer arithmetic. [Online]. Available: https://www.youtube.com/watch?v=aP0Y1uAA-2Y

[7] J. Gustafson and I. Yonemoto, "Beating floating point at its own game: Posit arithmetic," Supercomputing Frontiers and Innovations, vol. 4, no. 2, Jun 2017.

[8] J. Gustafson, "Posit Arithmetic," Mathematica Notebook describing the posit number system, 30 September 2017.

[9] J. Chen, Z. Al-Ars, and H. Hofstee, "A matrix-multiply unit for posits in reconfigurable logic using (open)capi." ACM, 2018, (To Be Published).

[10] P. Lindstrom, S. Lloyd, and J. Hittinger, "Universal coding of the reals: Alternatives to IEEE floating point," in Conference for Next Generation Arithmetic. ACM, 2018, (To Be Published).

[11] https://www.posithub.org/

VividSparks IT Solutions Pvt. Ltd.
CIN: U72200K2014OPC077975
#38 BSK Layout, Hubli-580031, INDIA.
www.vivid-sparks.com
inquiry@vivid-sparks.com