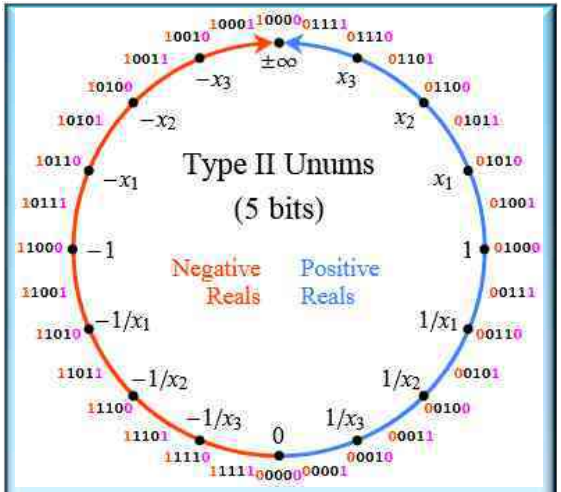
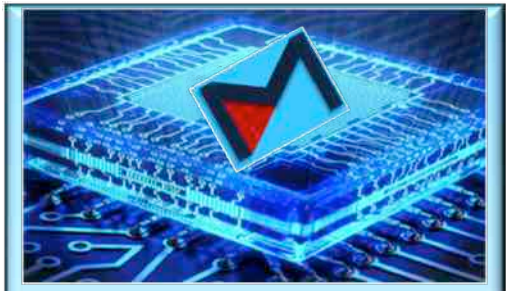
Sensonics Devices

A POSIT Compute Engine for HPC and ML

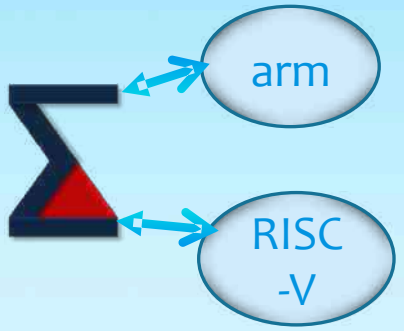
Introduction



POSIT (N, es) is a real number representation system that has the potential to outperform IEEE 754



We have developed a POSIT Compute Engine (PCE) and established that our algorithms achieve full accuracy targets



Seamless integration

The POSIT Compute Engine

The PCE can be a game changer both for ML and HPC applications

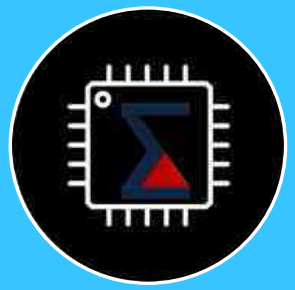
PCE can also incorporate a USB3 interface and used as an external "accelerator" for C programs executing in a Laptop or Intel PCs

PCE can be optimized for low power and used in battery powered IoT devices for ML in consumer form factors

POSIT COMPUTE ENGINE – WIP

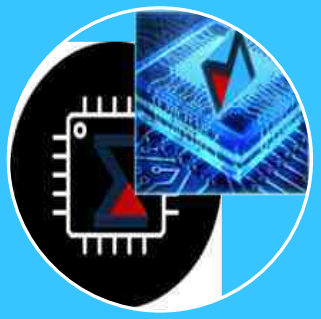
Modify GCC/LLVM so that existing C codes can directly offload to PCE

Build a USB based accelerator on FPGA



Enhance PCE to accommodate a Vector Processing Engine (VPE)

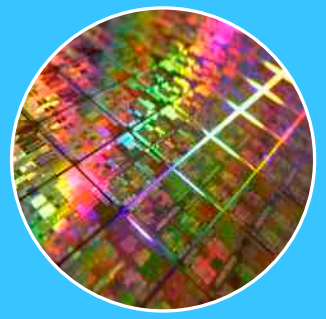
POSIT COMPUTE ENGINE – CUSTOMISATION



Define and create a family of Engines with varying levels of pipeline depth and / or scalar parallelism for ML or HPC

Integrate into the RISC V Instruction Execute Engine; add Exception Handler; Cache and Memory Control

Update GCC/LLVM to accommodate all POSIT and VPE Instructions



Silicon Prototype

POSIT INSTRUCTIONS FOR RISC V

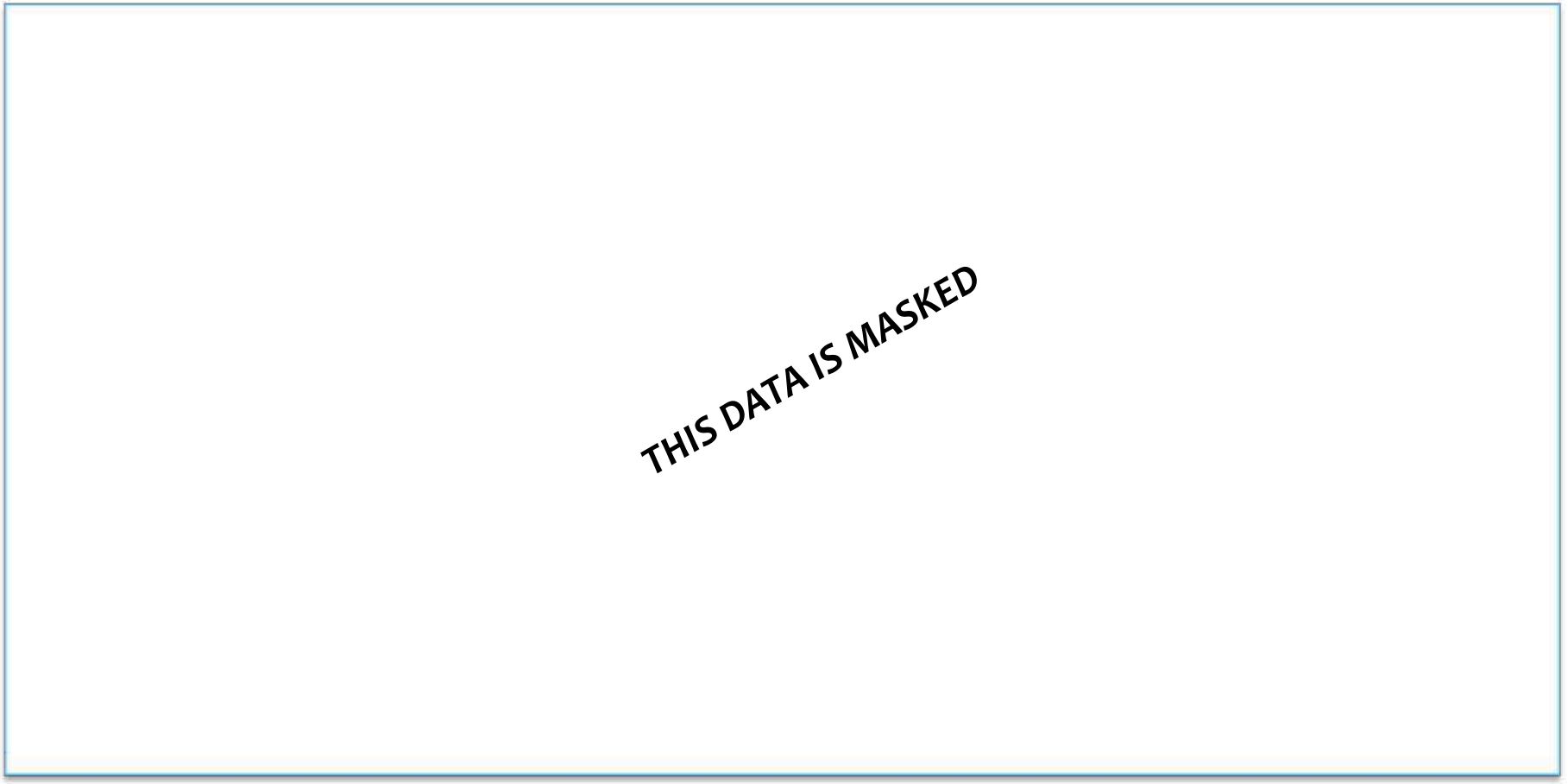
POSIT INSTRUCTIONS -1

S #	INSTRUCTION MNEMONIC	DESCRIPTION
1	PLW rd, rs1, offset	loads a POSIT (32,2) from memory location pointed by (integer) rs1 + a signed immediate offset into POSIT register rd
2	QULW rd, rs1, offset	loads a QuireUnit from memory pointed by (integer) rs1 + a signed immediate offset into Quire (POSIT register rd (PRF<31:16>))
3	PSW rs2, rs1, offset	stores a POSIT (32,2) from POSIT register rs2 to memory pointed by (integer) rs1 + signed immediate offset
4	QUSW rs2, rs1, offset	stores QuireUnit from POSIT register rs2 (PRF<31:16>) to memory pointed by (integer) rs1 + signed immediate offset
5	PADD.S rd,rs1,rs2	perform a POSIT (32,2) addition between rs1 and rs2 writing the result to rd
6	PSUB.s rd,rs1,rs2	perform a POSIT (32,2) subtraction between rs1 and rs2 writing the result to rd
7	PMUL.S rd,rs1,rs2	perform a POSIT (32,2) multiplication between rs1 and rs2 writing the result to rd
8	PDIV.S rd,rs1,rs2	perform a POSIT (32,2) division between rs1 and rs2 (rs2/rs1) writing the result to rd
9	PINV.S rd,rs2,1	Pseudo Instruction: perform a POSIT (32,2) division between rs1 and rs2 =1 (1/rs1) writing the result to rd
10	PMIN.S rd,rs1,rs2	perform compare of rs1 and rs2, write the smaller to rd
11	PMAX.S rd,rs1,rs2	perform compare of rs1 and rs2, write the larger to rd
12	PPHASE.S rd,rs1,rs2	computes phase of vector with X, Y components stored in rs1, rs2 (Arctan (rs2/rs1) and writes the result to rd
13	PATAN.S rd,rs1,rs2	Pseudo Instruction: computes arctan(rs2/1) stored in rs1, rs2 and writes the result to rd
14	PSQRT.S rd ,rs1	computes the square root of the contents of rs1 (PRF) and writes the result to rd. ($\sqrt{N}$)
15	PSQRTINV.S rd ,rs1	computes the inverse of the square root of the contents of rs1 (PRF) and writes the result to rd. ($1/(\sqrt{N})$)
16	PMAG.S rd,rs1,rs2	computes magnitude of vector with X, Y components stored in rs1, rs2 and writes the result to rd ($\sqrt{(X^2+Y^2)}$)
17	PMAGINV.S rd,rs1,rs2	computes the inverse of the magnitude of vector with X, Y components stored in rs1, rs2 and writes the result to rd ($1/\sqrt{(X^2+Y^2)}$)
18	PSINH.S rd ,rs1	computes the hyperbolic SINE of the contents of rs1 (PRF) and writes the result to rd.
19	PCOSH.S rd, rs1	computes the hyperbolic COSINE of the contents of rs1 (PRF) and writes the result to rd.
20	PTANH.S rd ,rs1	computes the hyperbolic TANGENT of the contents of rs1 (PRF) and writes the result to rd.
21	PEXP.S rd, rs1	computes the EXPONENT of the contents of rs1 (PRF) and writes the result to rd.
22	PXPWRY.S rd, rs1, rs2	computes $rs1^{rs2}$ (PRF) and writes the result to rd.
23	PXPWRYINV.S rd, rs1,rs2	computes $rs1^{(1/rs2)}$ (PRF) and writes the result to rd.
24	PASINH.S rd ,rs1	computes the hyperbolic INVERSE SINE of the contents of rs1 (PRF) and writes the result to rd.
25	PACOSH.S rd, rs1	computes the hyperbolic INVERSE COSINE of the contents of rs1 (PRF) and writes the result to rd.
26	PHYPMAG.S rd,rs1,rs2	computes magnitude of hyperbolic vector with X, Y components stored in rs1, rs2 and writes the result to rd ($\sqrt{(X^2-Y^2)}$)
27	PHYPMAGINV.S rd,rs1,rs2	computes the inverse of the magnitude of hyperbolic vector with X, Y components in rs1, rs2 and writes the result to rd ($1/\sqrt{(X^2-Y^2)}$)
28	PHYPHASE.S rd ,rs1, rs2	computes the phase of hyperbolic vector with X, Y components stored in rs1, rs2 (Arctanh (rs2/rs1) and writes the result to rd
29	PATANH.S rd ,rs1, rs2	Pseudo Instruction: computes the ArctanhY (Arctanh (rs2/1) and writes the result to rd

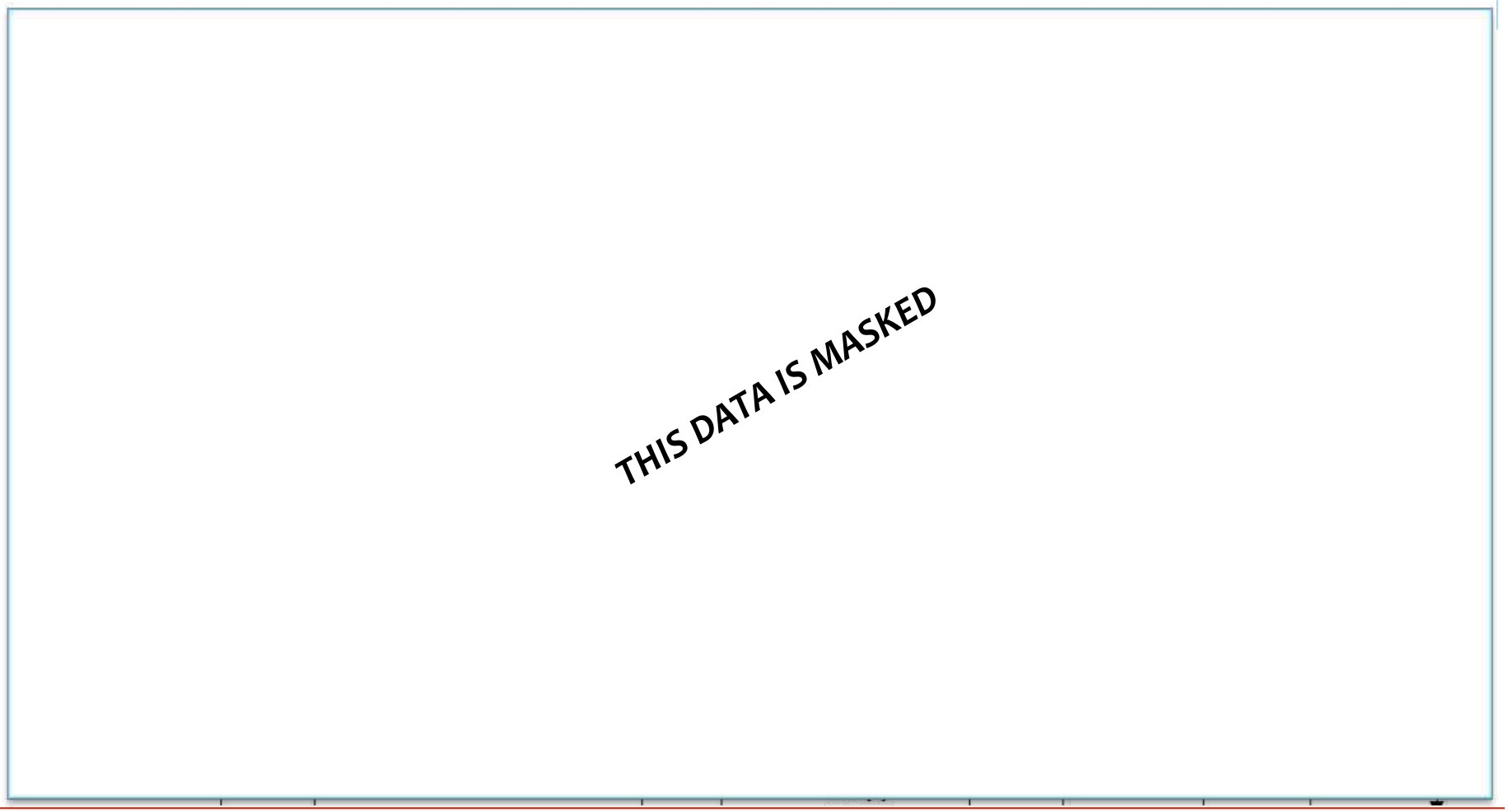
POSIT INSTRUCTIONS - 2

S #	INSTRUCTION MNEMONIC	DESCRIPTION
30	PLOGN.S rd, rs1	computes the Natural Logarithm of the contents of rs1 (PRF) and writes the result to rd.
31	PSIN.S rd ,rs1	computes the SINE of the contents of rs1 (PRF) and writes the result to rd.
32	PCOS.S rd, rs1	computes the COSINE of the contents of rs1 (PRF) and writes the result to rd.
33	PTAN.S rd ,rs1	computes the TANGENT of the contents of rs1 (PRF) and writes the result to rd.
34	PASIN.S rd ,rs1	computes the INVERSE SINE of the contents of rs1 (PRF) and writes the result to rd.
35	PACOS.S rd, rs1	computes the INVERSE COSINE of the contents of rs1 (PRF) and writes the result to rd.
36	PSIGMOID.S rd, rs1	Computes the approximate sigmoid of rs1 and writes the result to rd.
37	QCLR.S	Clears the Quire: q <--0
38	QNEG.S	Negates the Quire: q <-- - q
39	QROUND.S	Rounds the Quire and writes to a destination POSIT Register: rd <-- q
40	QMADD.S q, rs1, rs2	Computes: rs1 × rs2 + q --> q
41	QMSUB.S q, rs1, rs2	Computes: - rs1 × rs2 + q --> q
42	PSGNJ.S rd,rs1,rs2	POSIT sign-injection instruction: the result comprises all bits from rs1, (except the sign bit which is taken from rs2).
43	PSGNJN.S rd,rs1,rs2	POSIT sign-injection instruction: the result comprises all bits from rs1 (except the sign bit whch I sthe opposite that of rs2).
44	PSGNJX.S rd,rs1,rs2	POSIT sign-injection instruction: the result comprises all bits from rs1 (except the sign bit which is XOR of rs1, rs2 sign bits).
45	PMV.X.W rd,rs1,rs2	moves the contents of POSIT register rs1 to the integer register rd
46	PMV.W.X rd,rs1,rs2	moves the contents of integer register rs1 to POSIT register rd
47	PLT.S rd,rs1,rs2	perform rs1 < rs2 comparison and store result in rd
48	PLE.S rd,rs1,rs2	perform rs1 <= rs2 comparison and store result in rd
49	PEQ.S rd,rs1,rs2	perform rs1= rs2 comparison and store result in rd
50	PCVT.W.S	converts a POSIT number in POSIT register rs1 to a signed 32-bit in integer register rd.
51	PCVT.S.W rd,rs1,rs2	converts a signed 32-bit integer in register rs1 into a POSIT in POSIT register rd
52	PCVT.WU.S rd,rs1,rs2	converts a POSIT number in POSIT register rs1 to an unsigned 32-bit in integer register rd.
53	PCVT.S.WU rd,rs1,rs2	converts an unsigned 32-bit integer in register rs1 into a POSIT number in POSIT register rd

POSIT Datapath - 1



POSIT Datapath - 2



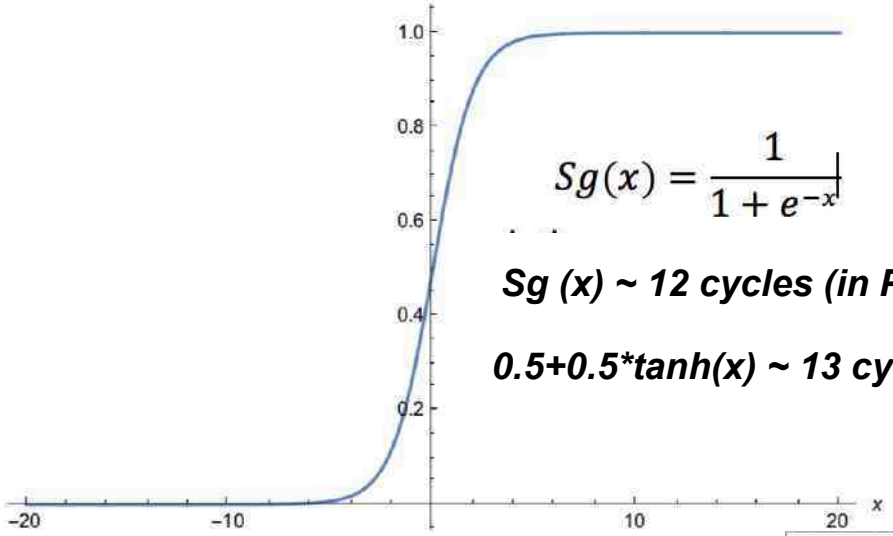
POSIT ERROR CHARACTERIZATION SUMMARY

POSIT Function Summary

Functions and their Cycle latency

Function	Cycle latency	Input Range	+Error	-ve error
1/x	7	$y=1, 1 \le x \le 2, \Delta=2^{-12}$	3.00E-10	-2.30E-10
y/x	8	$y=1, 1 \le x \le 2, \Delta=2^{-12}$	4.50E-10	0
Cos(x)	10	$0 \le x \le \pi/2, \Delta=2^{-12}$	1.60E-10	0
Sin(x)	10	$0 \le x \le \pi/2, \Delta=2^{-12}$	1.65E-10	0
Sqrt(x^2+y^2)	9	$y=1, 0 \le x \le 1, \Delta=2^{-12}$	1.40E-9	0
1/Sqrt(x^2+y^2)	13	$y=1, 0 \le x \le 1, \Delta=2^{-12}$	1.0E-09	-1.20E09
Atan(y/x)	9	$y=1, 0 \le x \le 1, \Delta=2^{-12}$	4.50E-10	-4.50E-10
Sinh(x)	10	$0 \le x \le 1, \Delta=2^{-12}$	2.50E-09	-1.50E-09
Cosh(x)	10	$0 \le x \le 1, \Delta=2^{-12}$	4.50E-09	-2.50E-09
Sqrt(x^2-y^2)	19	$x=1, 0 \le y \le 1, \Delta=2^{-12}$	0	-2.00E-05
Atanh(y/x)	19	$x=1, 0 \le y \le 1, \Delta=2^{-12}$	0.029	-0.015
1/Sqrt(x^2-y^2)	20	$x=1, 0 \le y \le 1, \Delta=2^{-12}$	0	-0.089
Ln(x)	19	$1 \le x \le 2, \Delta=2^{-12}$	1.7E-09	-1.67E-09
Sqrt(x)	20	$1 \le x \le 2, \Delta=2^{-12}$	1.4E-09	-2.10E-09
1/Sqrt(x)	21	$1 \le x \le 2, \Delta=2^{-12}$	2.40E-09	0
Acosh(x)	37	$1 \le x \le 2, \Delta=2^{-12}$	2.30E-09	-6.20E-08
Asinh(x)	37	$0 \le x \le 1, \Delta=2^{-12}$	0.0015	-0.001
Acos	25	$0 \le x \le 1, \Delta=2^{-12}$	6.10E-08	-6.10E-08
Asin	25	$0 \le x \le 1, \Delta=2^{-12}$	6.10E-08	-6.10E-08
Tan(x)	15	$-1 \le x \le 1, \Delta=2^{-12}$	1.57E-09	-5.00E-10
Tanh(x)	17	$1 \le x \le 2, \Delta=2^{-12}$	2.00E-09	-1.50E-09
Exp(x)	11	$0 \le x \le 1, \Delta=2^{-12}$	5.70E-09	-1.00E-10
x^y	30	$y=1, 1 \le x \le 3, \Delta=2^{-12}$	6.50E-09	-2.00E-09
$x^{(1/y)}$	30	$y=1, 1 \le x \le 2, \Delta=2^{-12}$	6.50E-09	-2.00E-09

POSIT SIGMOID FUNCTION



$$Sg(x) = \frac{1}{1 + e^{-x}}$$

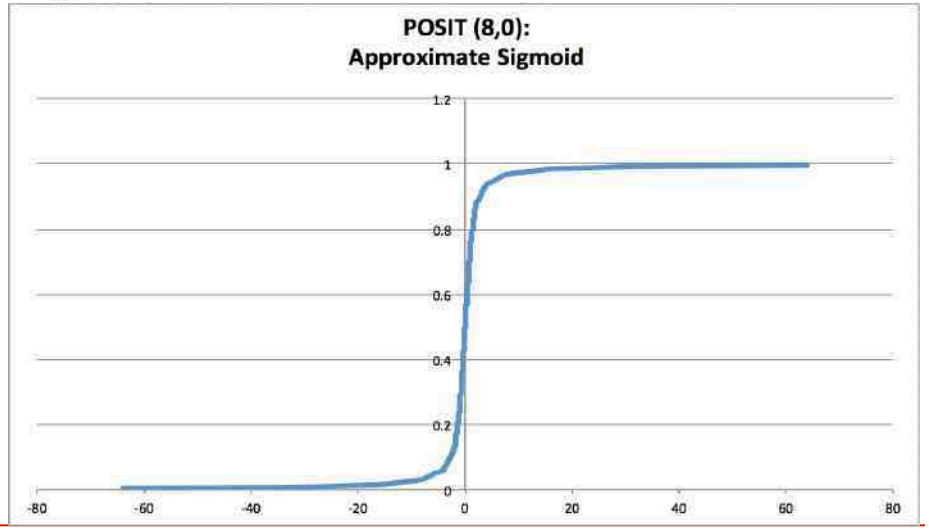
Sg (x) ~ 12 cycles (in PCU)

0.5+0.5*tanh(x) ~ 13 cycles (in PCU)

The approximate sigmoid computes in 1 cycle; POSIT (16,1) or (8,0) is the preferred usage

It is possible (not yet implemented) to do 2 (or 4) such computes per cycle, or embed inside a Branch Instruction without additional overhead

It can significantly improve ML algorithm speeds



POSIT FOR ML ACCELERATION

POSIT(32,2) can consistently deliver **end** accuracies > 8 - 9 digits. This is equivalent to **end** accuracies achieved by double precision float

- Thus simply migrating to POSIT will halve the memory BW and computational complexity of Real Number computation ***without sacrificing end accuracy***

The approximate sigmoid computes in 1 cycle and if used can additionally further reduce computational load

Thus POSIT can enable migration of many ML Algorithms from Cloud to Edge

Interested? Questions?
Suggestions?
Please contact us at:
enquiry@sensonics.in