Операции с плавающей точкой

школа синтеза цифровых схем

Занятие №9

19 ноября 2022

ПРИ ПАРТНЕРСТВЕ

YADRO

YADRO · MP

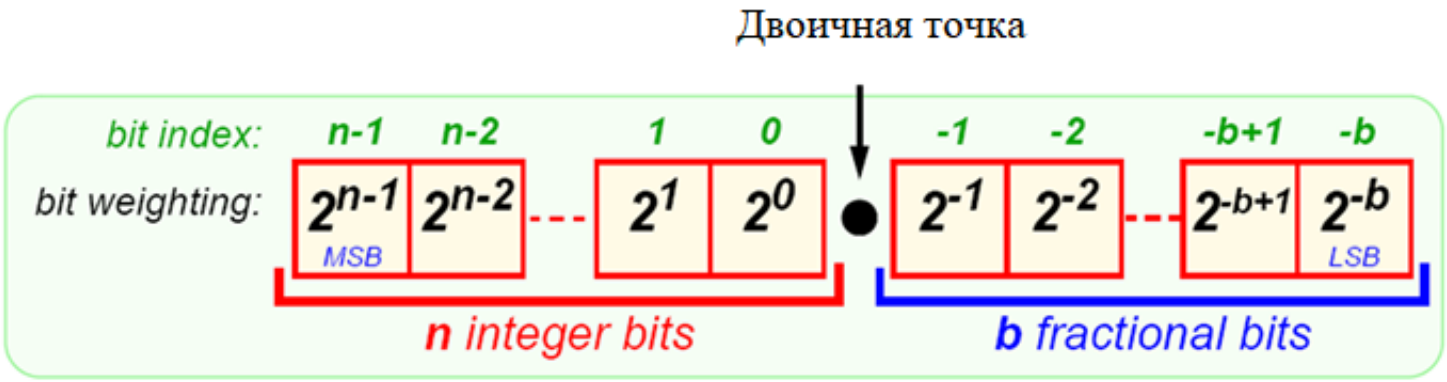
Syntacore™

Илья Кудрявцев

Преподаватель Самарского университета

Дробные числа в формате с фиксированной точкой

Формат с фиксированной точкой предусматривает n целых бит и b дробных бит.



Вес дробных битов:

- 2^{-1} = 0.5
- 2^{-2} = 0.25
- 2^{-3} = 0.125....

Дробные числа в формате с фиксированной точкой

Шестнадцатиричное представление	Двоичное представление	Десятичное представление
7FFF	0111 1111 1111 1111	+0.999969
0001	0000 0000 0000 0001	+0.000031
0000	0000 0000 0000 0000	+0.000000
FFFF	1111 1111 1111 1111	–0.000031
8000	1000 0000 0000 0000	–1.000000

Особенности формата с плавающей точкой

Проблемы формата с фиксированной точкой:

- Ограниченный динамический диапазон;
- Потеря точности (значащих цифр);

Достоинства формата с фиксированной точкой:

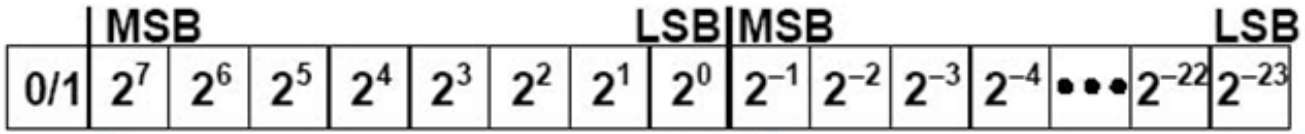
- Простота реализации;
- Использование тех же ресурсов, что и в целочисленных операциях

IEEE-754 : Формат с плавающей точкой (одинарная точность)

$$NUMBER_{10} = (-1)^{S} \times 1.M \times 2^{(E-127)}$$

Смещение

Предполагается



бит знака = S

Показатель степени 8-бит

Мантисса = M 23-бита

0 00000111 1100...00

\+ 7 0.75

Смещение

$+ 1{,}75 \times 2^{(7-127)} = + 1{,}316554 \times 10^{-36}$

Предполагается

1 10000001 0110...00

– 129 0.375

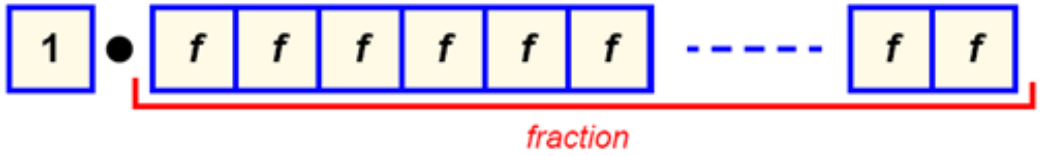
Смещение

$- 1{,}375 \times 2^{(129-127)} = - 5{,}500000$

Предполагается

IEEE-754 : Формат с плавающей точкой

Согласно IEEE 754, мантисса должна быть нормализованной, т.е. представленной в следующей форме (целочисленный бит должен всегда иметь значение 1)



Целая часть мантиссы указывается неявно, т.е. представляет собой «скрытый бит»

Мантисса умножается на знаковый бит, т.е. на +1 или -1

$s = 0$: $(-1)^s = (-1)^0 = 1$ Положительные числа

$s = 1$: $(-1)^s = (-1)^1 = -1$ Отрицательные числа

IEEE-754 : Формат с плавающей точкой

IEEE 754 определяет четыре варианта формата / точности.
Чаще всего используется формат одинарной точности (**Single**) и формат двойной точности (**Double**).

Length (bits)	Format			
	Single	Single Extended	Double	Double Extended
Significand	24	≥ 32	53	≥ 64
Exponent	8	≥ 11	11	≥ 15
Total	32	≥ 43	64	≥ 79

В форматах одинарной и двойной точности используются, соответственно, 23-х разрядная и 52-разрядная мантисса

Для формата с одинарной точностью требуется:

23 бита мантиссы + 8 бит порядка + 1 бит знака = 32 бит.

Для формата с двойной точностью требуется:

52 бита мантиссы + 11 бит порядка + 1 бит знака = 64 бит.

IEEE-754 : Формат с плавающей точкой

- 16-разрядная с фиксированной точкой:
  - 2^{16} = 65536 возможных значений

- 32-разрядная с плавающей точкой:
  - Наибольшее значение: $\pm 6.8 \times 10^{38}$, в стандарте IEEE-754: $\pm 3.4 \times 10^{38}$
  - Наименьшее значение: $\pm 5.9 \times 10^{-39}$, в стандарте IEEE-754: $\pm 1.2 \times 10^{-38}$
- Расширенная точность (40-бит: знак + 8-битная экспонента + 31-битная мантисса)
- Двойная точность: (64-бит.: знак + 11-бит. эксп.+ 52-бит. мантисса)
- 32-разрядная с плавающей точкой:
  - Более высокая точность
  - Больший динамический диапазон
  - Проще в программировании

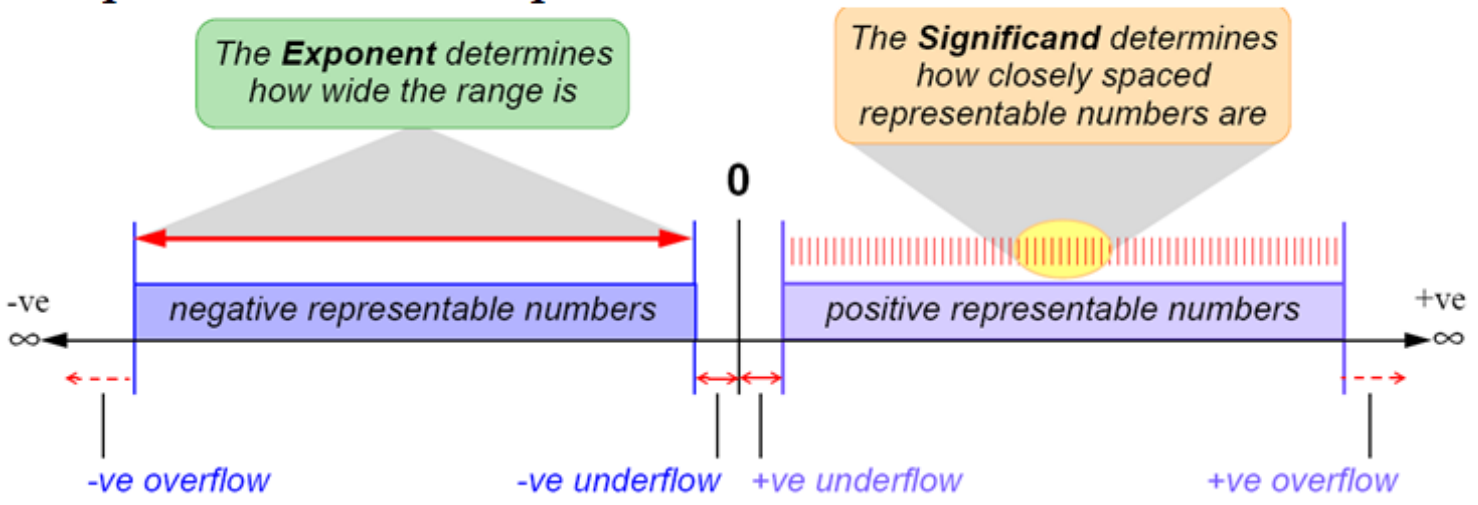
IEEE-754 : Формат с плавающей точкой

Согласно IEEE-754 должны поддерживаться следующие операции с плавающей точкой:

- Сложение, вычитание и умножение
- Деление, вычисление квадратного корня и остатка
- Сравнения
- Операции округления
- Преобразования между форматами
- Преобразования между форматами с плавающей точкой и целочисленными

IEEE-754 : Формат с плавающей точкой

- Мантисса и порядок определяют различные свойства чисел в формате с плавающей точкой
- Мантисса определяет разрешение
- Порядок определяет динамический диапазон числа
- Аналогично арифметике с фиксированной точкой, возможны переполнение и потеря точности



IEEE-754 : Формат с плавающей точкой (32 бит)

Table 1–2. Representation of Special-Case Numbers in IEEE 754-1985 Standard

Meaning	Sign Field	Exponent Field	Mantissa Field
Zero	0/1	All 0's	All 0's
Positive Denormalized	0	All 0's	Non-zero
Negative Denormalized	1	All 0's	Non-zero
Positive Infinity	0	All 1's	All 0's
Negative Infinity	1	All 1's	All 0's
Not a Number (NaN)	0/1	All 1's	Non-zero

NaN	Zero	Infinity	Denormalized
• (+∞) + (-∞) = NaN • NaN + x = NaN	• (+a) + (-a) = +0 • (+0) + (-0) = +0 • (-0) + (-0) = -0	• *overflow*: (a) + (a) = inf • (inf) + x = inf • (inf) + (inf) = inf	• *underflow*: (a) + (-b) = denorm

1.Floating Point Adder/Subtractor (ALTFP_ADD_SUB) Megafunction User Guide

IEEE-754 : Формат с плавающей точкой (32 бит)

Tools & Thoughts

IEEE-754 Floating Point Converter

h-schmidt.net/FloatConverter/IEEE754.html

Translations: de

This page allows you to convert between the decimal representation of numbers (like "1.02") and the binary format used by all modern CPUs (IEEE 754 floating point).

IEEE 754 Converter (JavaScript), V0.22

	Sign	Exponent	Mantissa
Value:	+1	2^{127}	1.9999998807907104
Encoded as:	0	254	8388607
Binary:	☐	☑☑☑☑☑☑☑☐	☑☑☑☑☑☑☑☑☑☑☑☑☑☑☑☑☑☑☑☑☑☑☑☑☑

Decimal representation	3.40282346639e+38
Value actually stored in float:	340282346638528859811704183484516925440
Error due to conversion:	
Binary Representation	01111111011111111111111111111111
Hexadecimal Representation	0x7f7fffff

+1 -1

IEEE 754 Converter (JavaScript), V0.22

	Sign	Exponent	Mantissa
Value:	+1	2^{127}	1.999999761581421
Encoded as:	0	254	8388606
Binary:	☐	☑☑☑☑☑☑☑☐	☑☑☑☑☑☑☑☑☑☑☑☑☑☑☑☑☑☑☑☑☑☑☑☑☐

Decimal representation	3.40282326356e+38
Value actually stored in float:	340282326356119256160033759537265639424
Error due to conversion:	
Binary Representation	01111111011111111111111111111110
Hexadecimal Representation	0x7f7ffffe

+1 -1

IEEE-754 : Формат с плавающей точкой (32 бит)

IEEE 754 Converter (JavaScript), V0.22

	Sign	Exponent	Mantissa
Value:	+1	2^{128}	1.0
Encoded as:	0	255	0
Binary:	☐	☑☑☑☑☑☑☑☑	☐☐☐☐☐☐☐☐☐☐☐☐☐☐☐☐☐☐☐☐☐☐☐☐☐☐☐☐☐☐☐

Decimal representation	inf
Value actually stored in float:	Infinity
Error due to conversion:	
Binary Representation	01111111100000000000000000000000
Hexadecimal Representation	0x7f800000

+1 -1

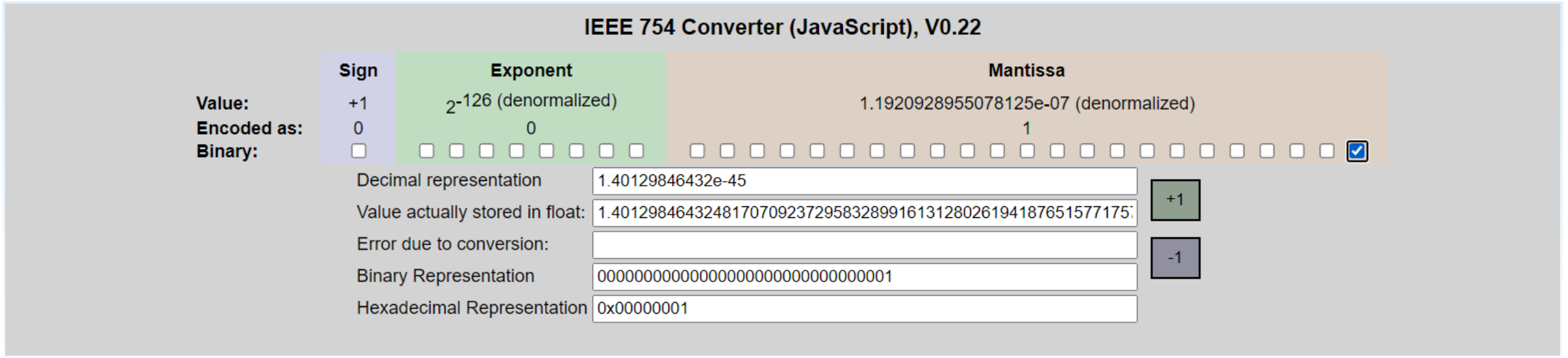
IEEE 754 Converter (JavaScript), V0.22

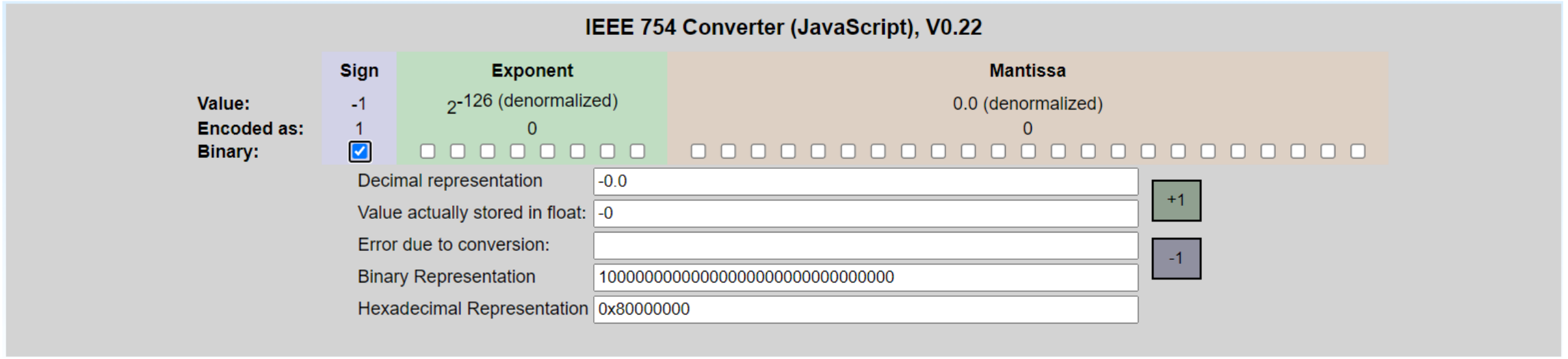
	Sign	Exponent	Mantissa
Value:	+1	2^{128}	1.0000001192092896
Encoded as:	0	255	1
Binary:	☐	☑☑☑☑☑☑☑☑	☐☐☐☐☐☐☐☐☐☐☐☐☐☐☐☐☐☐☐☐☐☐☐☐☐☐☐☐☐☐☑

Decimal representation	nan
Value actually stored in float:	NaN
Error due to conversion:	
Binary Representation	01111111100000000000000000000001
Hexadecimal Representation	0x7f800001

+1 -1

IEEE-754 : Формат с плавающей точкой (32 бит)





IEEE-754 : Формат с плавающей точкой (32 бит)

IEEE 754 Converter (JavaScript), V0.22

	Sign	Exponent	Mantissa
Value:	+1	2^{-4}	1.600000023841858
Encoded as:	0	123	5033165
Binary:	☐	☐☑☑☑☑☐☑☑	☑☐☐☑☑☐☐☑☑☐☐☑☑☐☐☑☑☐☐☑☑☐☑

You entered	0.1
Value actually stored in float:	0.100000001490116119384765625
Error due to conversion:	1.490116119384765625E-9
Binary Representation	00111101110011001100110011001101
Hexadecimal Representation	0x3dcccccd

+1

-1

IEEE 754 Converter (JavaScript), V0.22

	Sign	Exponent	Mantissa
Value:	+1	2^{127}	1.7632415294647217
Encoded as:	0	254	6402534
Binary:	☐	☑☑☑☑☑☑☑☐	☑☑☐☐☐☐☑☑☐☑☑☐☐☐☑☑☑☑☐☐☑☑☐

You entered	3.0E+38
Value actually stored in float:	3000000005497755757778039942811452702 72
Error due to conversion:	5.49775575777803994281145270 3E+29
Binary Representation	01111111011000011011000111100110
Hexadecimal Representation	0x7f61b1e6

+1

-1

IEEE-754 : Исключения в операциях с плавающей точкой

Исключение	Примечания
Переполнение	Результат $\pm\infty$
Потеря точности	Результат равен нулю или денормализован
Деление на нуль	Результат $\pm\infty$
Некорректная операция	Результат NaN (не число)
Неточность	Требуется округление

IEEE-754 : Операции с некорректным результатом

Операция	Примечания
Сложение/вычитание	Операции типа $\infty \pm \infty$
Умножение	Операции типа $0 \times \infty$
Деление	Операции типа 0/0 или ∞/∞
Вычисление остатка	Операции типа x REM 0 или $\infty REMy$
Вычисление квадратного корня	Квадратный корень из отрицательного числа

Денормализованные числа

Побитовое представление: **s_00000000_xxxxxxxxxxxxxxxxxxxxxxx (мантиса != 0)**

Представление числа в десятичном виде: $$F = (-1)^S 2^{-126}(M/2^{23})$$

Образуются при вычитании малых чисел, пример:

a = 0_00000100_00000000000000000000000 ~ 2^{-123}

b = 1_00000011_11010000100111000001111 ~ -1.81488*2^{-124}

1) a = 1.00000000000000000000000 * 2^{-123}

   b = -0.11101000010011100000111|1 * 2^{-123}

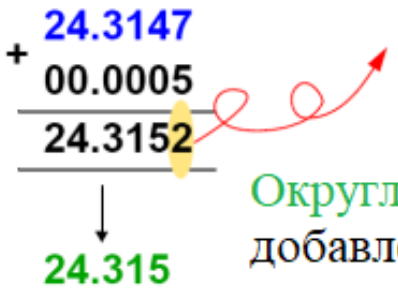
2) sum = 0.00010111101100011111000|1 * 2^{-123} = 0.10111101100011111000100 * 2^{-126}

sum = 0_00000000_10111101100011111000100 ~ 0.74047 * 2^{-126}

Усечение и округление

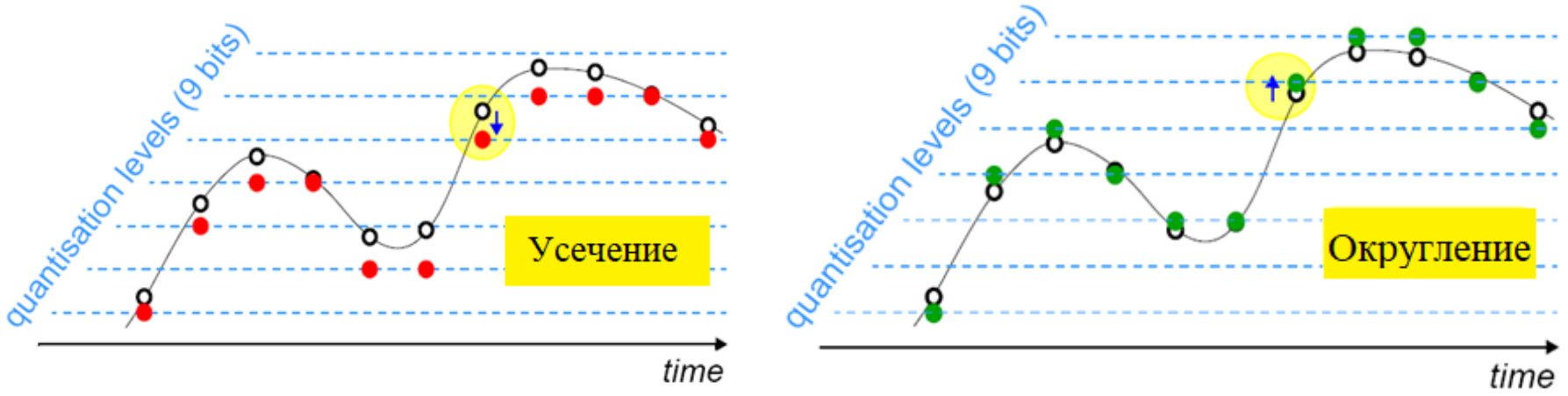
- Усечение и округление представляют собой операции уменьшения точности представления числа для сокращения размера мантиссы
- Рассмотрим пример в десятичной арифметике
- Предположим, что нужно сократить 27.3147 до трех цифр после десятичной точки
- После усечения получим 27.314, а после округления 27.315, так как операции отличаются

Усечение означает отбрасывание младших цифр мантиссы



Округление означает добавление половины младшей значащей цифры перед отбрасыванием

Усечение и округление



Операция округления обеспечивает более высокую точность, чем операция усечения, но требует дополнительной операции сложения.

Округление

Мантисса после сдвига: 11001001100110010101010|101111

L - last bit - последный бит, который помещается в размер мантиссы.
G - guard bit - первый не помещающийся бит
R - round bit - второй не помещающийся бит
S - sticky bit - логическая сумма всех оставшихся битов

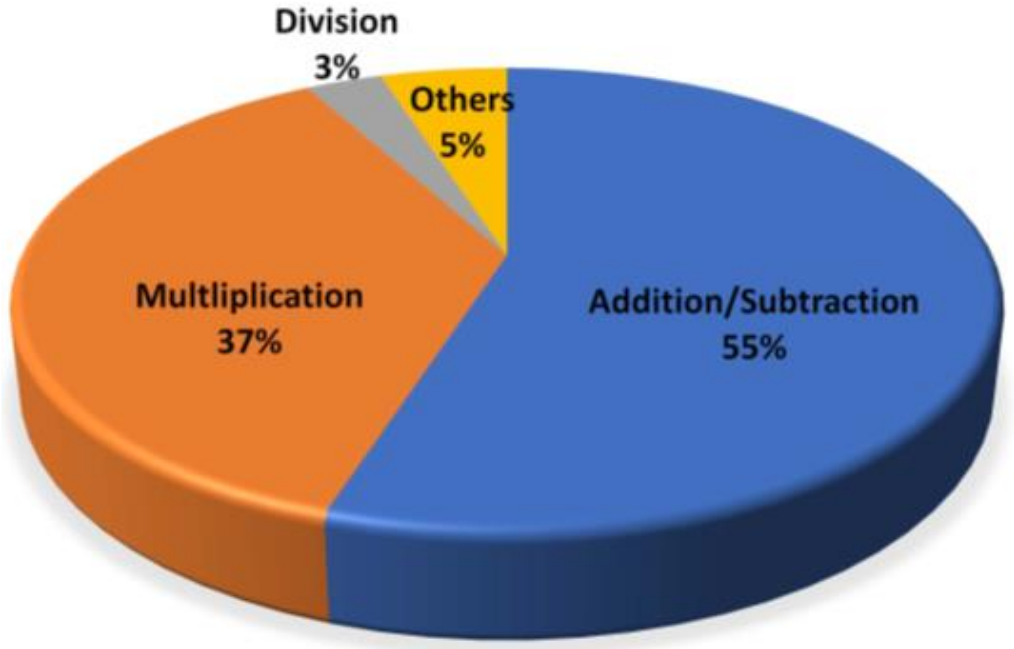
Округление происходит в случае, если условие **G & (L | R | S)** верно.

Таблица 4.1.

L	G	R	St	C_r
X	0	X	X	0
1	1	X	X	1
X	1	1	X	1
X	1	X	1	1

$$C_r = G\,(R + St + L) = 1,$$

Типичные операции IEEE-754



Asynchronous Floating-Point Adders and Communication Protocols: A Survey
Pallavi Srivastava 1, Edwin Chung 1 and Stepan Ozana 2

Алгоритм операции умножения

- Шаг 1 : Умножение мантисс :

$$SIG_R = SIG_A \times SIG_B = 1.375 \times 1.5 = 2.0625$$

- Шаг 2 : Сложение порядков

$$E_R = E_A + E_B = 5 + 7 = 12$$

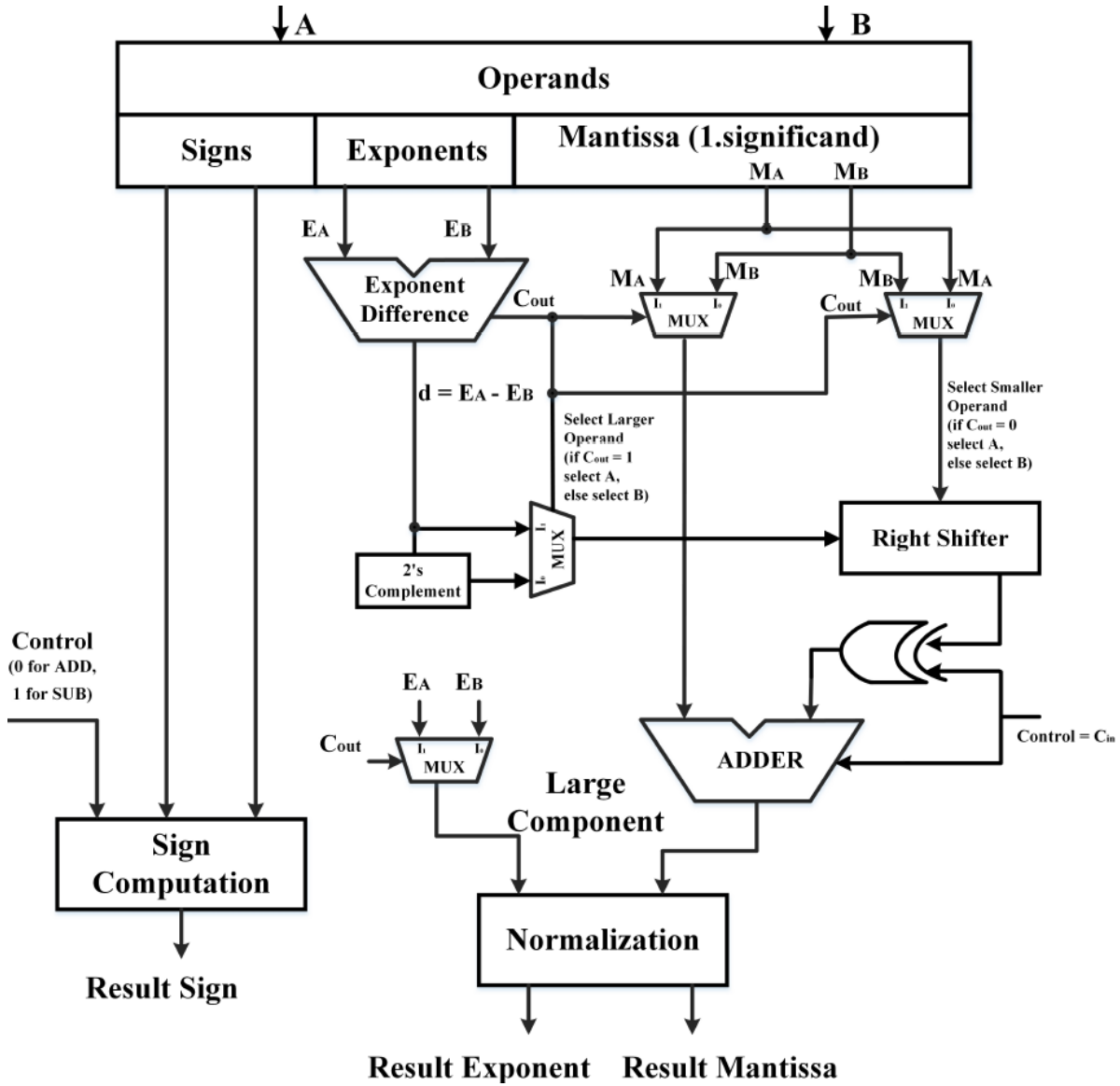
- Шаг 3 : Нормализация результата путем коррекции мантиссы и порядка. (это необходимо, так как мантисса $SIG_R > 2$)

$$SIG_{R_{norm}} = \frac{SIG_R}{2} = \frac{2.0625}{2} = 1.03125$$

$$E_{R_{norm}} = E_R + 1 = 12 + 1 = 13$$

- Шаг 4 : Определение порядка результата $S_R = S_A$ xor $S_B = 1$

Операция сложения

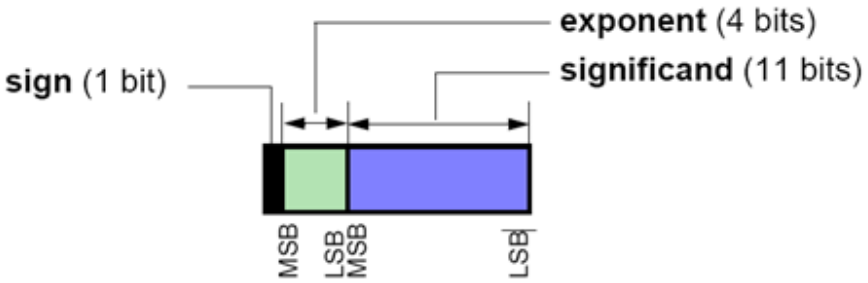


Блочная плавающая точка

- Блочная плавающая точка – это технология работы с блоком чисел с одинаковым порядком (но разными мантиссами);
- Операции осуществляются над элементами блока без необходимости постоянной нормализации;
- Порядок корректируется только в случае сильного увеличения или уменьшения чисел;
- Величина порядка определяется числом максимальной величины;
- За счет упрощения техники применение блочной техники требует значительно меньшего объема ресурсов, чем стандартные операции с плавающей точкой;
- Технология блочной плавающей точки хорошо подходит к алгоритмам типа БПФ и реализуется в сигнальных процессорах.

Нестандартные форматы

- Не существует жесткого требования обязательного соответствия IEEE-754
- Анализируя требования приложения к точности и динамическому диапазону, можно подобрать соответствующие параметры формата с плавающей точкой
- Пример нестандартного формата с 4-х разрядным порядком и 11-разрядной мантиссой (16 разрядов с учетом знака)



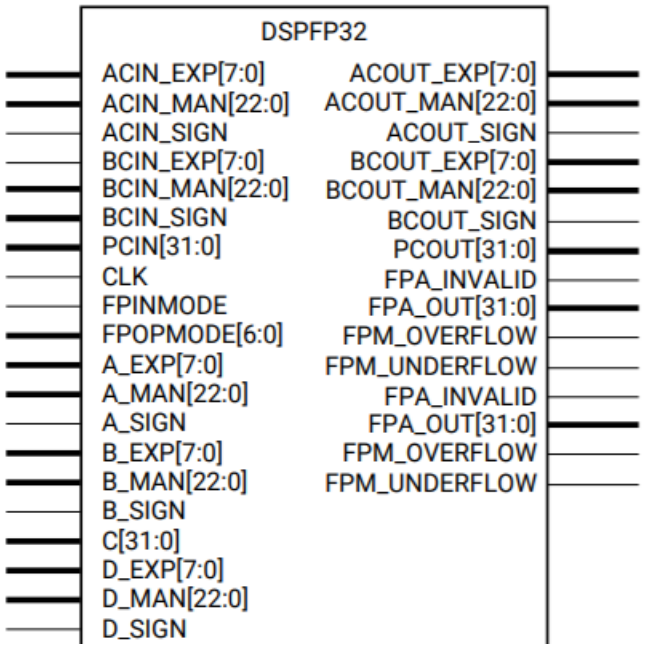
- По сравнению со стандартным форматом обеспечивается более широкий динамический диапазон при незначительном снижении точности

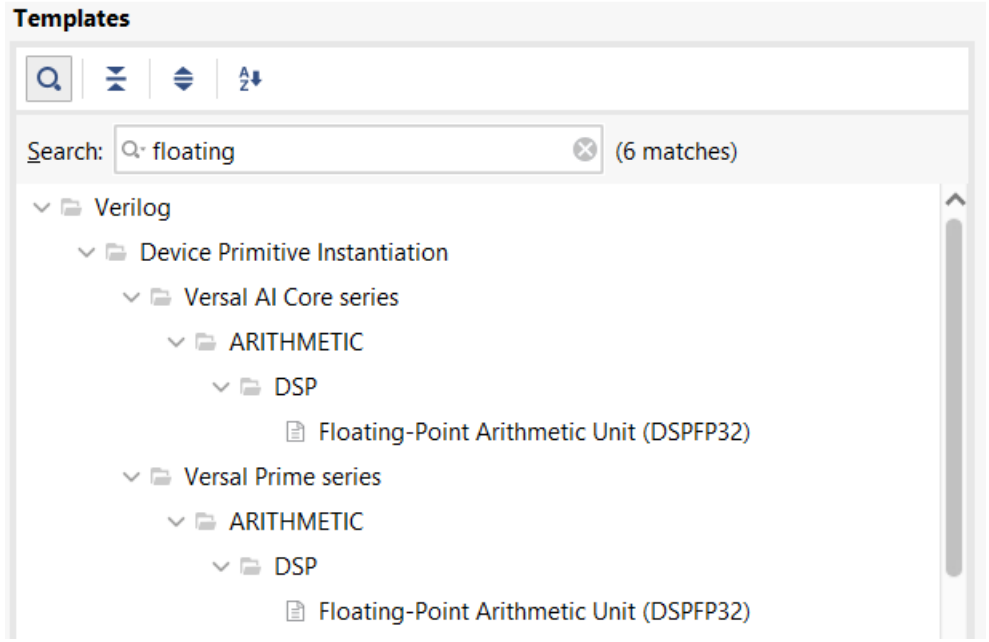
Поддержка операций с плавающей точкой в Versal

DSPFP32

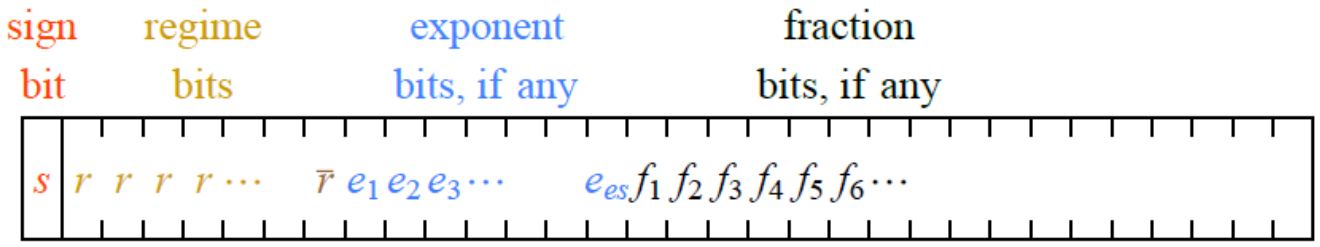
Primitive: The DSPFP32 consists of a floating-point multiplier and a floating-point adder with separate outputs.

PRIMITIVE_GROUP: ARITHMETIC
PRIMITIVE_SUBGROUP: DSP

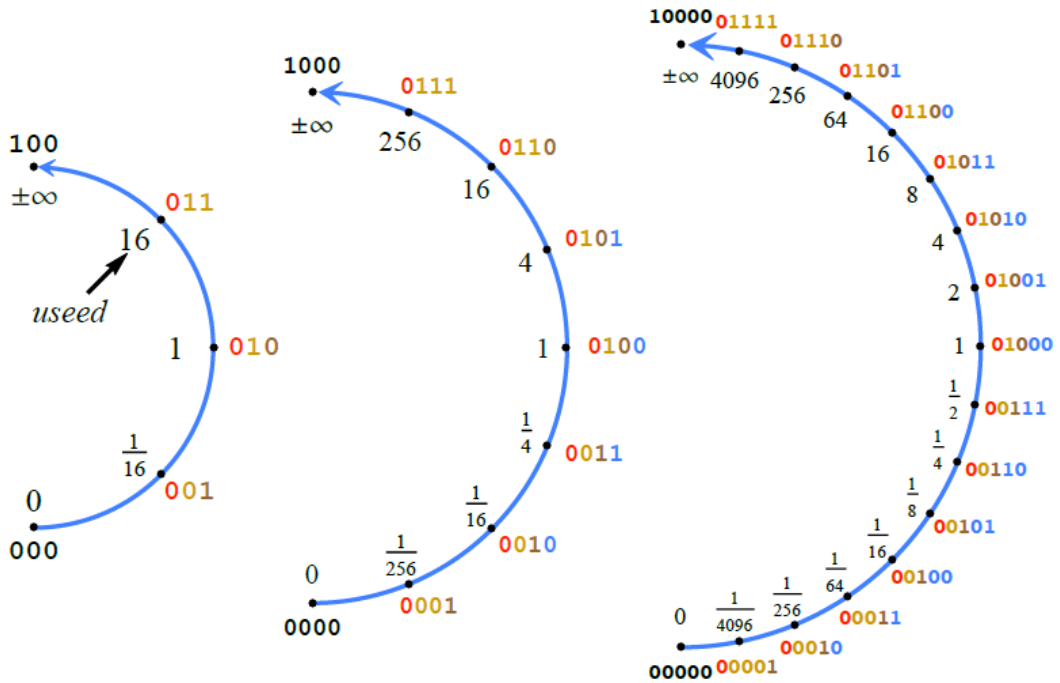




Posits



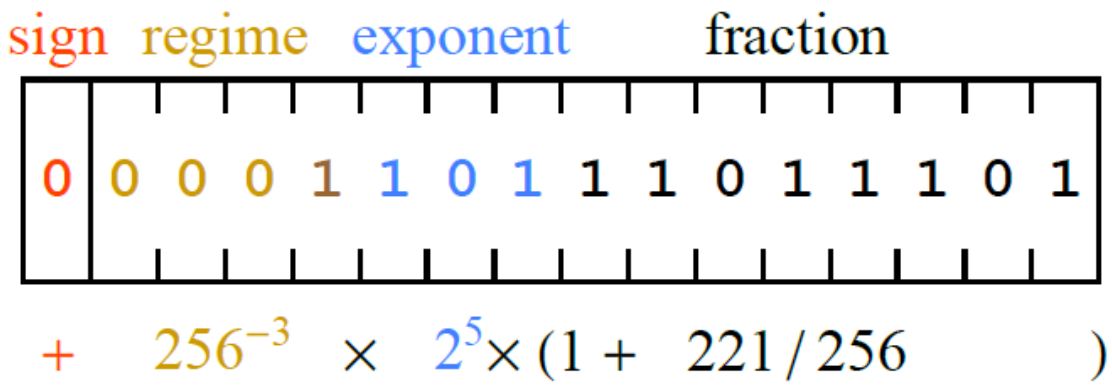
es	0	1	2	3	4
useed	2	$2^2 = 4$	$4^2 = 16$	$16^2 = 256$	$256^2 = 65536$



Beating Floating Point at its Own Game: Posit Arithmetic
John L. Gustafson , Isaac Yonemoto

Posits

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



Beating Floating Point at its Own Game: Posit Arithmetic
John L. Gustafson , Isaac Yonemoto