Computer Arithmetics:
Floating Point Arithmetic Circuits

Dr. Ahmed Elhossini
Fachgebiet: Architektur eingebetteter Systeme
Institut für Technische Informatik und Mikroelektronik
Fak. IV – Elektrotechnik und Informatik

Technische Universität Berlin

Previous Lecture
Floating Point Numbers

- Floating Point Numbers
  - Finite numbers
  - Floating point number format
  - IEEE floating point standard
    - Single / Double formats
    - Exponent Encoding
    - Special Operands and Subnormal's
- Arithmetic operations
  - Requirements
  - Addition/Subtraction
  - Multiplication/Division
  - Exception

Technische Universität Berlin

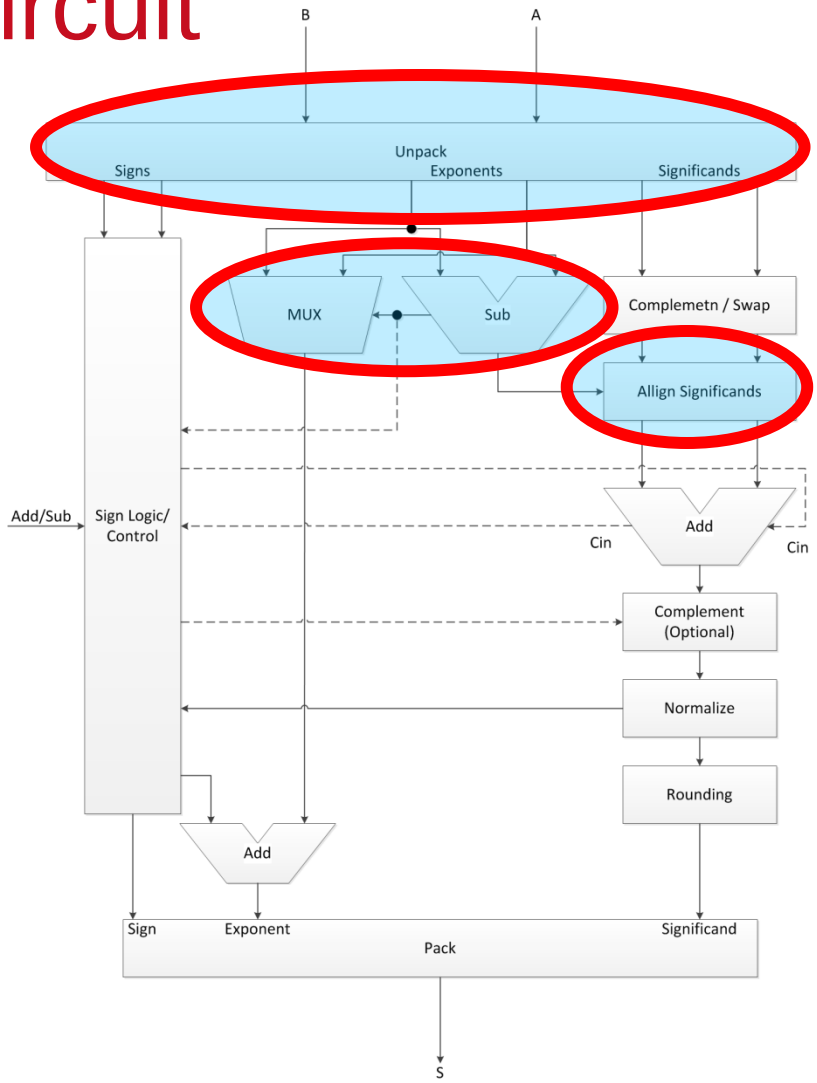
Floating Point Circuits

- Adder Circuit
  - Controlled Shifter – Significant alignment
  - Normalization
  - Rounding
  - Sign Control and Exception Generation
- Multiplier Circuit
- Divider Circuit
- Multiply Accumulate Circuit

Technische Universität Berlin

Floating Point Adder Circuit

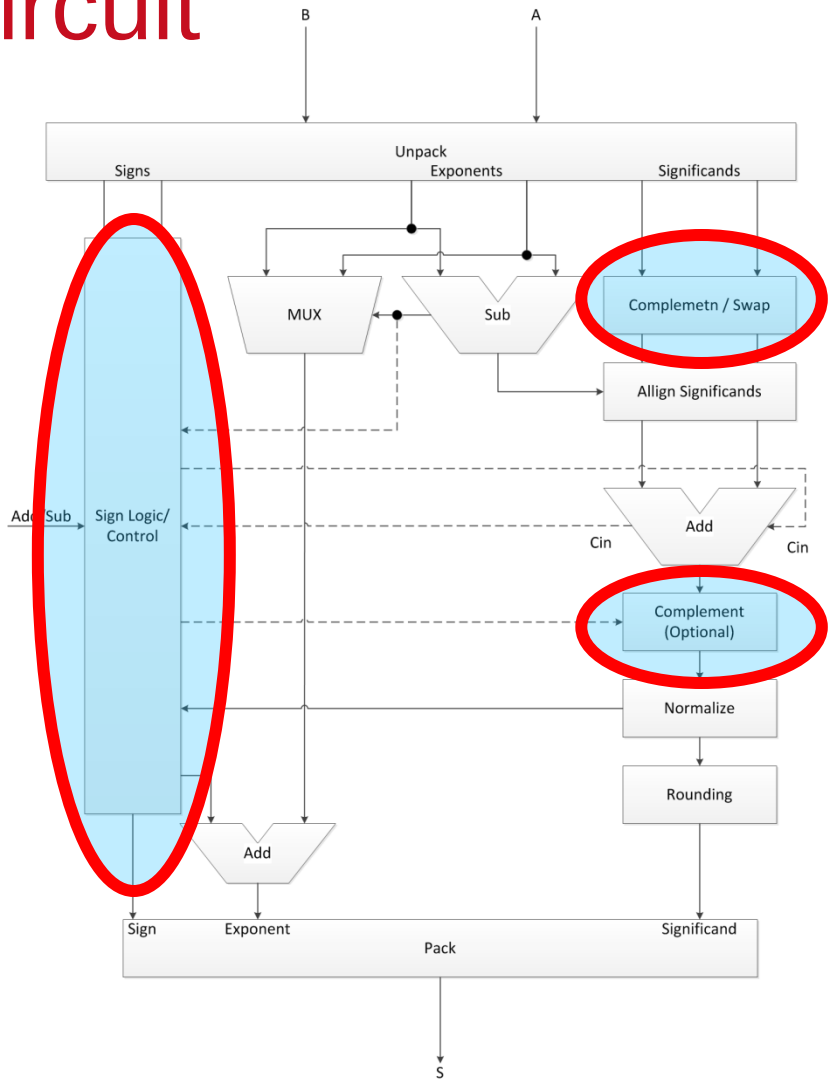
- Separate the exponent and significant components from the inputs
- Subtract Exponents, depending on the result select the maximum value
- Use the difference between the two exponents to align the significant by shifting



Technische Universität Berlin

Floating Point Adder Circuit

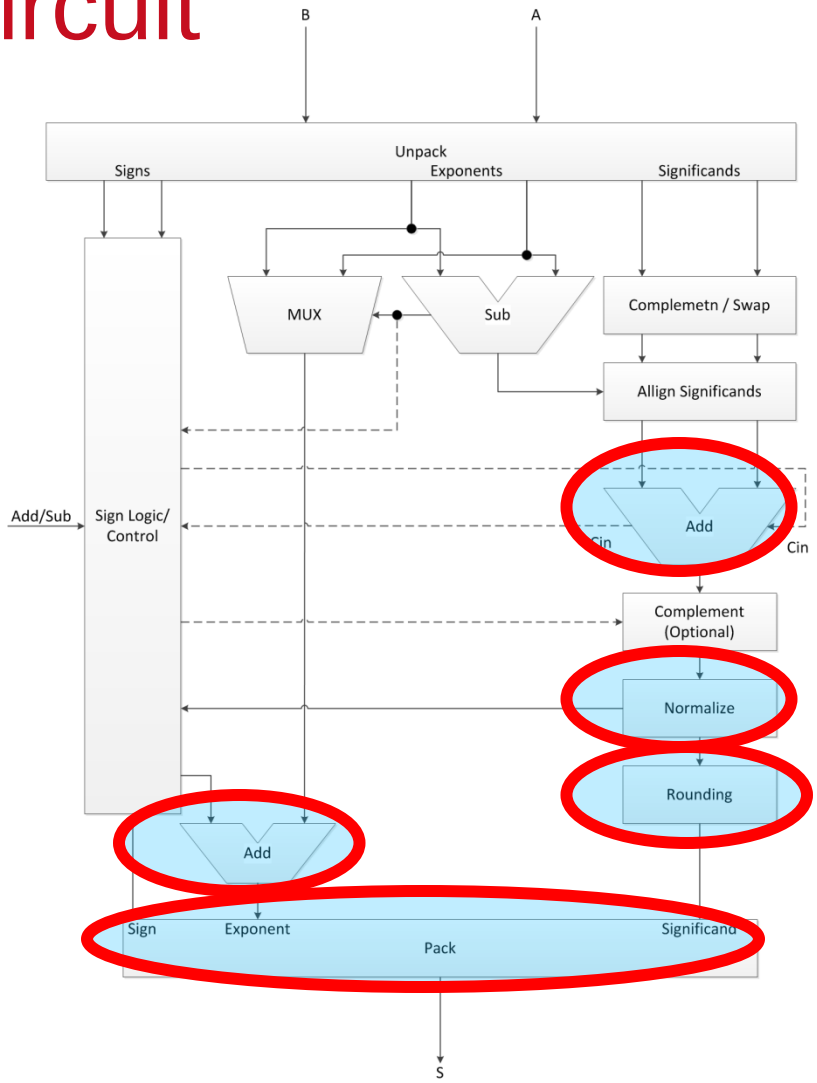
- The sign logic is used to control the adder/subtractor
- Depending on the deference of the exponents, the significant can be swapped or complemented
- Depending on the sign of both operands, complementing the adder output may be required



Technische Universität Berlin

Floating Point Adder Circuit

- Add (subtract significant)
- Normalize the addition result (after complement)
- Round the output to fit the significant
- Adjust the Exponent depending on the normalization result
- Pack the output parts into the output



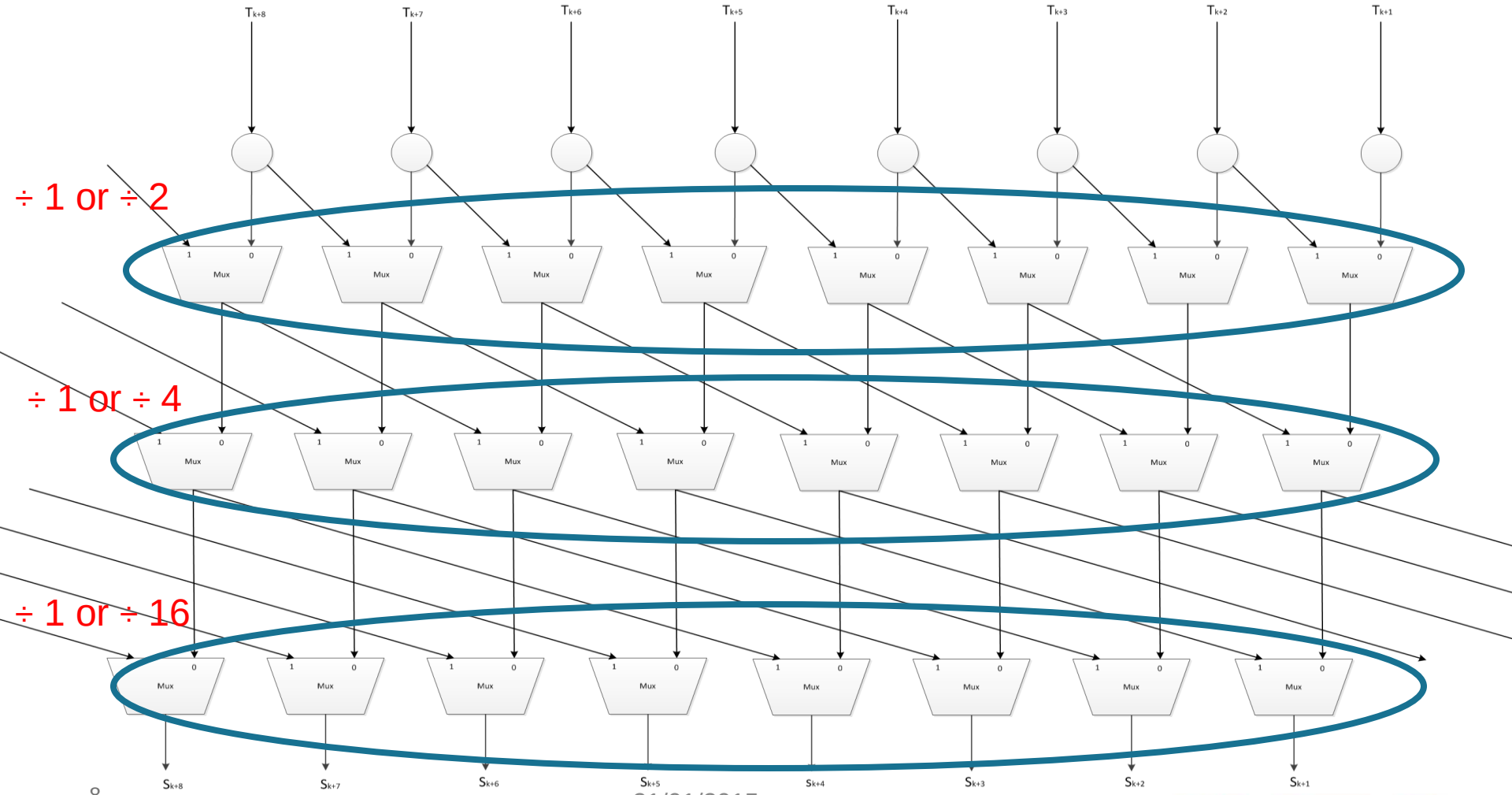
Technische Universität Berlin

Significant Alignment – Barrel Shifter

- Significant Alignment is performed using Barrel Shifter
- Barrel shifter is able to perform controlled shifting (left/right) depending on a specific input
- Multiplexers are usually used to implement such a shifter
- Multiplexer stage is added for each control bit
- Total number of multiplexers needed = number of bits in the input vector * number of bits in the control word

Technische Universität Berlin

Controlled Shifter – Right Barrel Shifter



Technische Universität Berlin

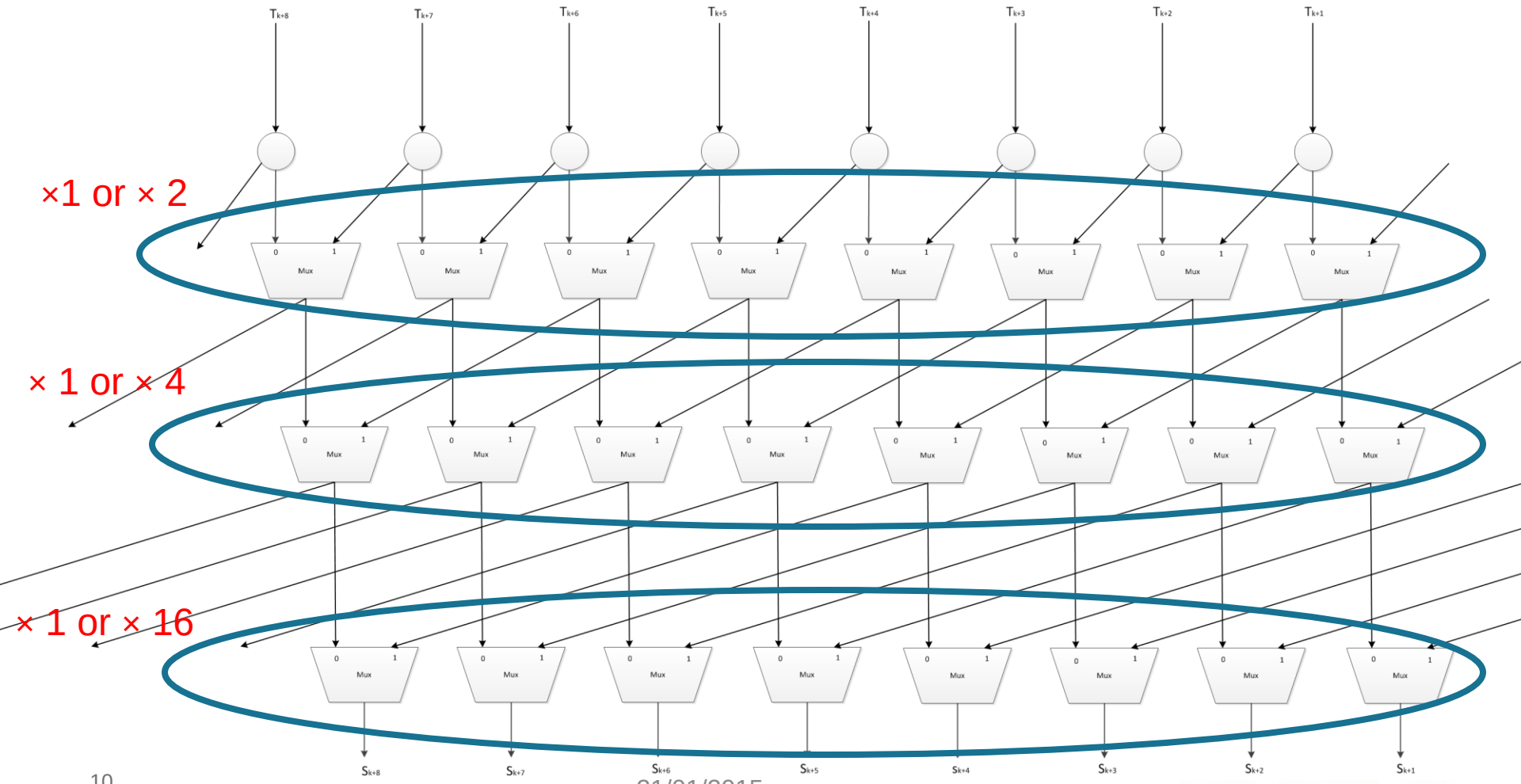
Controller Shifter – Right Barrel Shifter

S_2	S_1	S_0	Operation	Value	Shift Right Value
0	0	0	1 / 1×1×1	1 / 1	0
0	0	1	1 / 1×1×2	1 / 2	1
0	1	0	1 / 1×4×1	1 / 4	2
0	1	1	1 / 1×4×2	1 / 8	3
1	0	0	1 / 16×1×1	1 / 16	4
1	0	1	1 / 16×1×2	1 / 32	5
1	1	0	1 / 16×4×1	1 / 64	6
1	1	1	1 / 16×4×2	1 / 128	7

- Add more stages to allow more shifts
- Can be used in both significant alignment and normalization
- Similar structure can be used to make left shift

Technische Universität Berlin

Controlled Shifter – Left Barrel Shifter



Technische Universität Berlin

Controller Shifter – Left Barrel Shifter

S_2	S_1	S_0	Operation	Value	Shift Left Value
0	0	0	1×1×1	1	0
0	0	1	1×1×2	2	1
0	1	0	1×4×1	4	2
0	1	1	1×4×2	8	3
1	0	0	16×1×1	16	4
1	0	1	16×1×2	32	5
1	1	0	16×4×1	64	6
1	1	1	16×4×2	128	7

- Add more stages to allow more shifts
- Can be used in both significant alignment and normalization

Technische Universität Berlin

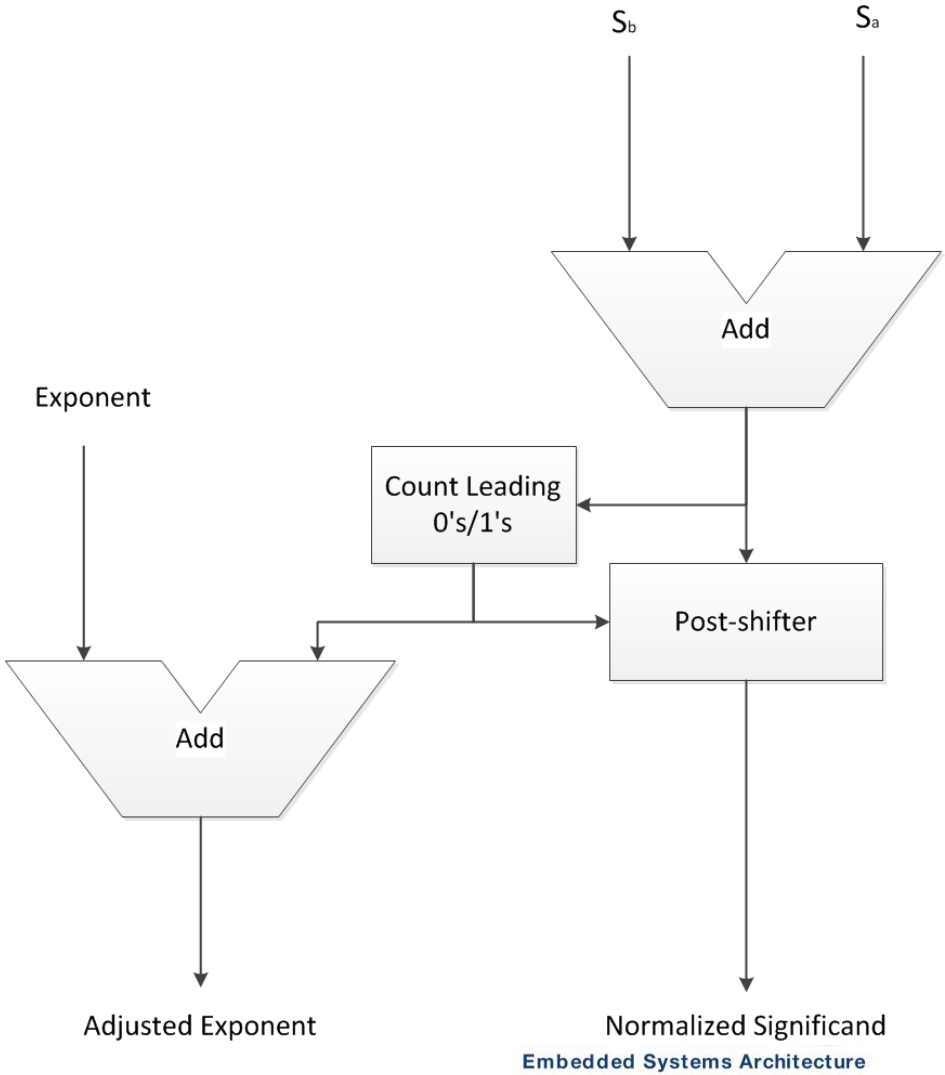
Normalization

- The normalization process is required ensure that significant value is in the range [1,2)
- It is done in two stage
  - Calculate the number of leading zeros/ones
  - Shift the significant left/right
- Leading 0's or 1's is calculated using a combinational logic
  - Leading 1's are found in negative two's complement numbers
- To speed it up, 0's or 1's can be predicted before addition so that Post shifting can be done just after addition

Technische Universität Berlin

Normalization

- Counting leading 0's or 1's in the adder output
- Use this count as input to the barrel shifter
- Post-shifter and Exponent adjustment should wait for the counting process
- Simple implementation using combinational logic approach



Technische Universität Berlin

0's or 1's counting

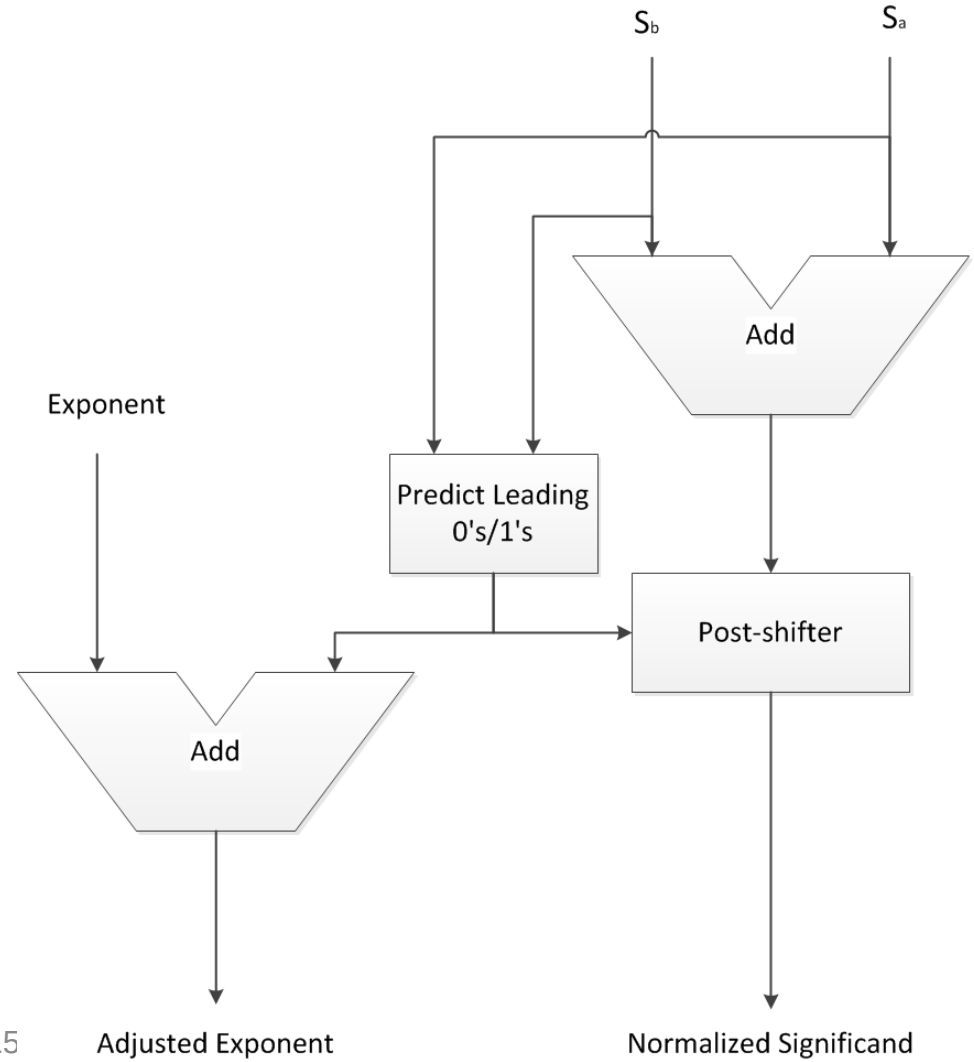
a_0	a_{-1}	a_{-2}	a_{-3}	a_{-4}	a_{-5}	a_{-6}	a_{-7}	s_2	s_1	s_0
1	x	x	x	x	x	x	x	0	0	0
0	1	x	x	x	x	x	x	0	0	1
0	0	1	x	x	x	x	x	0	1	0
0	0	0	1	x	x	x	x	0	1	1
0	0	0	0	1	x	x	x	1	0	0
0	0	0	0	0	1	x	x	1	0	1
0	0	0	0	0	0	1	x	1	1	0
0	0	0	0	0	0	0	1	1	1	1

- 8 x 3 priority encoder for 8 bits string
- For 24 bits (significant length in single word type), we need 24x5 encoder
- Can be implemented using a Look-up table

Technische Universität Berlin

Normalization

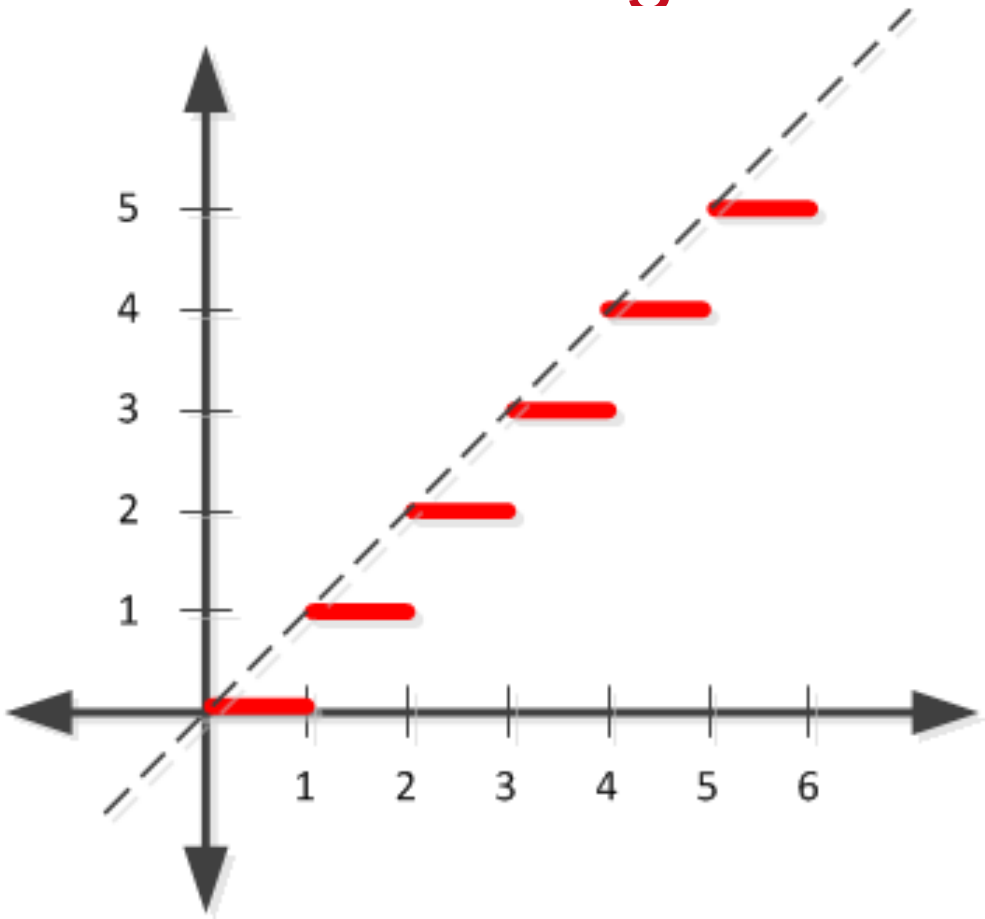
- Predict 0's or 1's before the addition operation
- Post-shifter can start directly after addition



Technische Universität Berlin

Rounding: Truncation or Rounding

- Rounding or Truncation
- Truncation is simple,
  - remove extra bits.

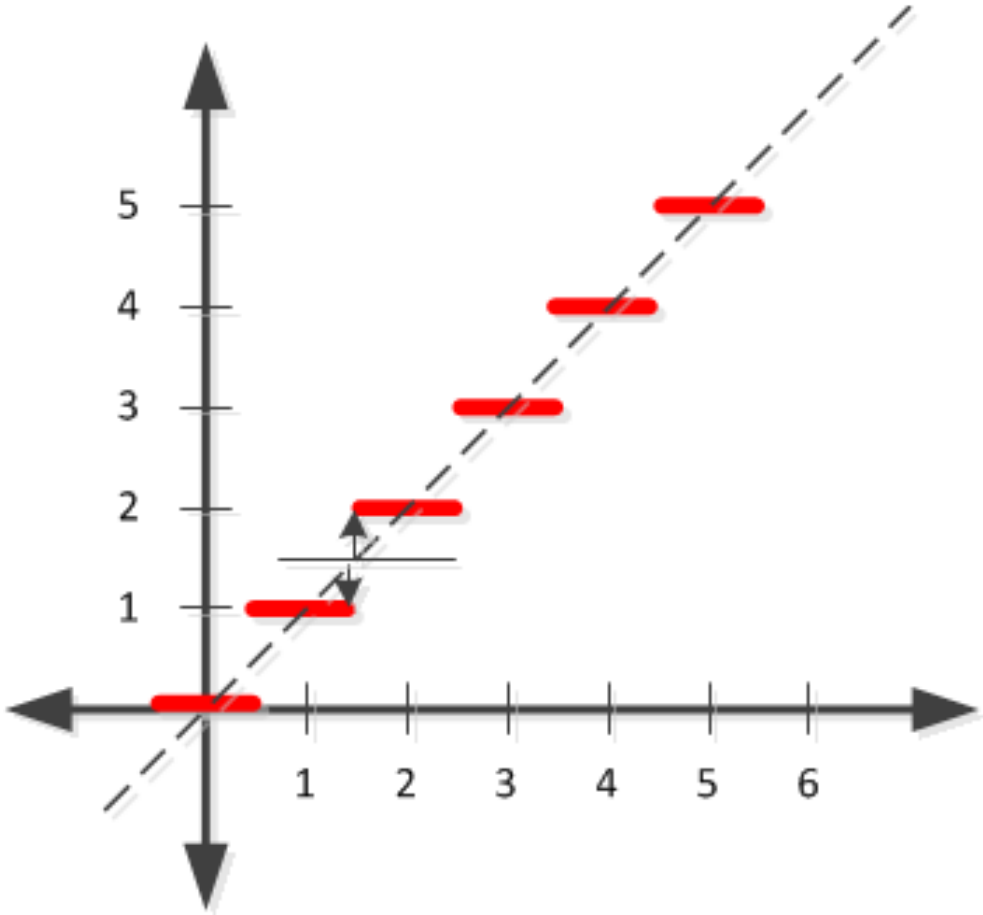


AES Embedded Systems Architecture

Technische Universität Berlin

Rounding

- Rounding can be up, or down to the nearest number



Rounding

- $X = x_2x_1x_0x_{-1}x_{-2}$
- $Y=y_2y_1y_0$
- If two bits to be rounded out:

X_{-2}	X_{-1}	Y	Direction	Error
0	0	Y=int (x)	Down	0
0	1	Y=int(x)	Down	-0.25
1	0	Y=int(x) + 1	Up	0.5
1	1	Y = int (x) +1	Up	0.25

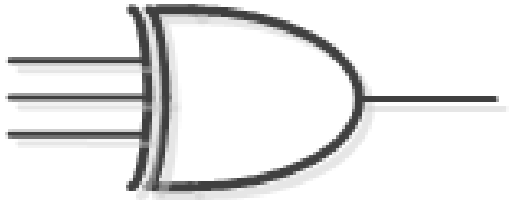
- Combinational logic to encode the extra bits (lookup table can be used)
- Adder circuit is required to adjust the unrounded bits
- If this actual rounding is performed, another stage of normalization may be required

Technische Universität Berlin

Sign Control

- Required to build adder/subtractor circuit
- Compare the sign of the operands to decide the function of the significant adder taking into account the required operation
- From the table it is an XOR function
- Pre-complement and Post complement are controlled by this signal

0 = add 1 = sub	sn_b	sn_a	adder operation
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	1
1	1	0	0
1	0	1	0
1	1	1	1



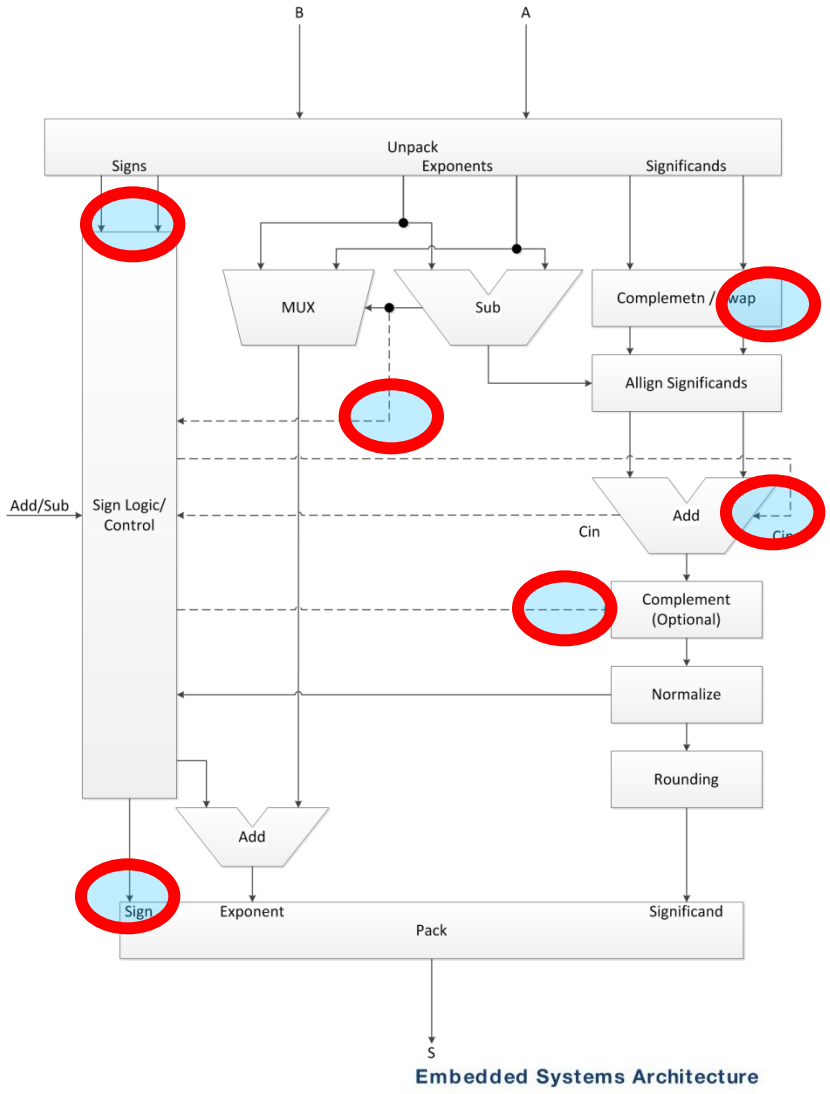
AES Embedded Systems Architecture

Technische Universität Berlin

Sign Control

- The sign of the exponent difference may be used to swap the input
- The output sign depends on the operation being performed, and the sign of the adder result

Operation	Output Sign
Addition of two positive number	Positive
Addition of two negative numbers	Negative
Addition of different sign numbers	The sign of the adder output
Subtraction of two numbers with the same sign	The sign of the adder output
Subtraction of different sign numbers	The sign of the first operand



Technische Universität Berlin

Exceptions in Adder Circuit

- Overflow/Underflow can be detected during exponent adjustment
  - Check the value of the exponent after normalization adjustment (the adder circuit)
- NaN values can be detected in the unpacking and packing stages
  - Check the values of the exponent and significant at the input and output
  - Same circuit as the zero detection circuit we used before
  - Same circuit can also be used to detect the Zero flag
- The Sign flag is the sign of the output

Technische Universität Berlin

Final Notes on the Adder Circuit

- Complex operation
- Several computational stages are required
- Pipelining is important here – 10 stages
  - Several Pipeline stages are required to minimize the critical path delay and increase the clock frequency
  - This is the reason that modern micro-processors are featured with deep pipelining
- Modern micro-processors have more than data path
  - A data path for integer operations which is faster and commonly used
  - A data path for floating point operation which is slower (in number of clock cycles) but less common to use
- Barrel shifting is a common instruction in many micro-processor
  - The existence of such instruction reduce the software size when shifting is required, which is common in many application
  - Resource sharing can be used to reduce the total processor area

Technische Universität Berlin

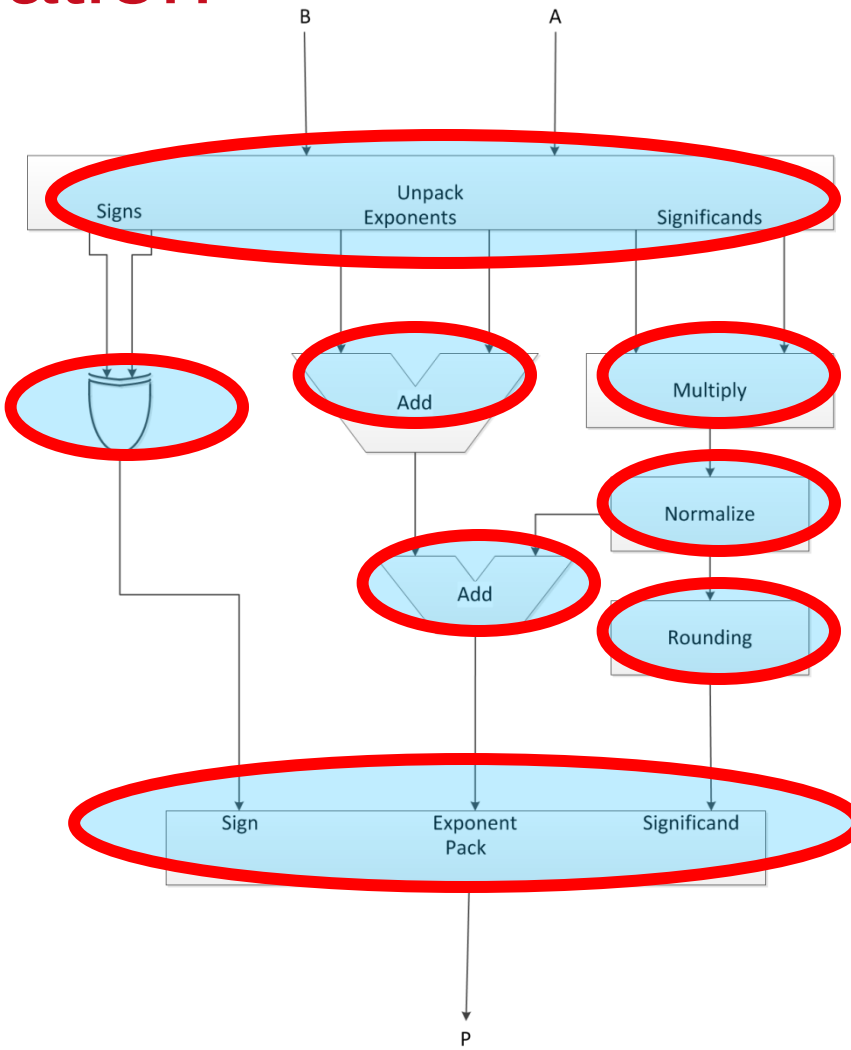
Floating Point Multiplication

- Simple circuit compared to the adder
- Significant component are multiplied directly
- Exponents are added
- Normalization and Rounding are performed to generate the output
  - Barrel shifter is used as shown in the adder circuit
- Exceptions are checked the same way as discussed in the adder circuit

Technische Universität Berlin

Floating Point Multiplication

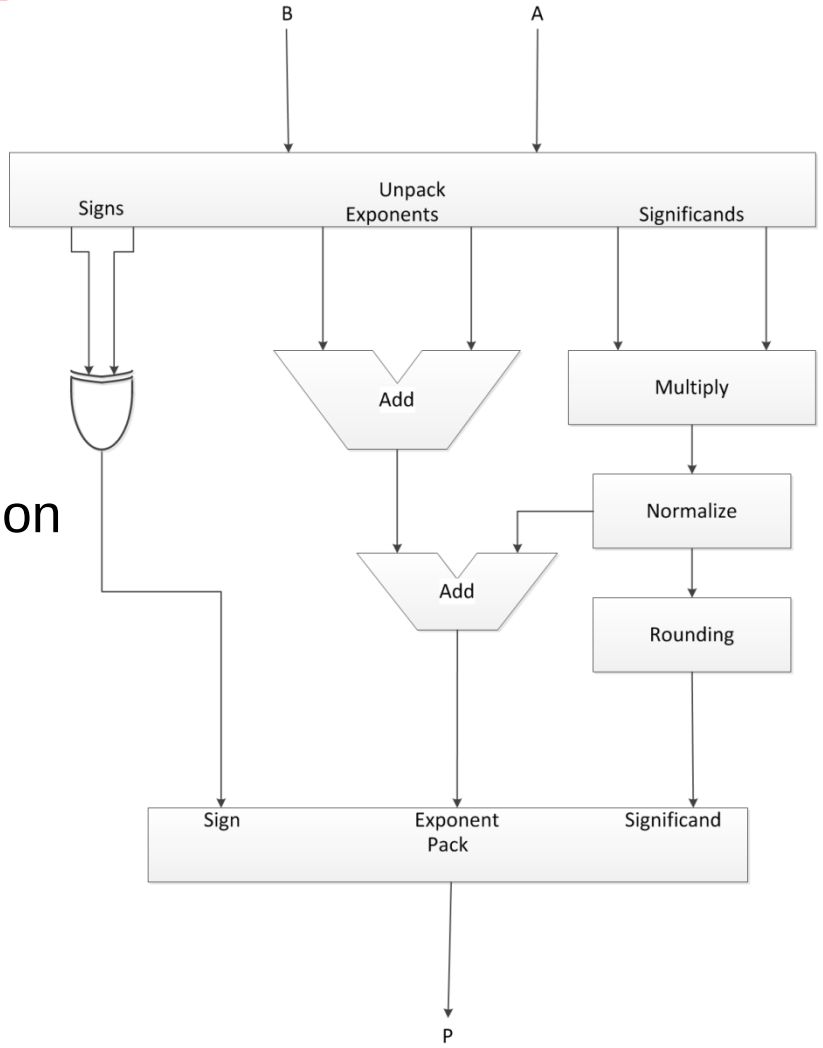
- Unpack operand (check exceptions)
- Multiply Significant(s)
- Add the exponents
- Normalize Significant(s)
- Adjust Exponent
- Round the significant
  - Extra normalization my be needed
- Generate the sign
- Pack the output (check exceptions)



Technische Universität Berlin

Floating Point Multiplier

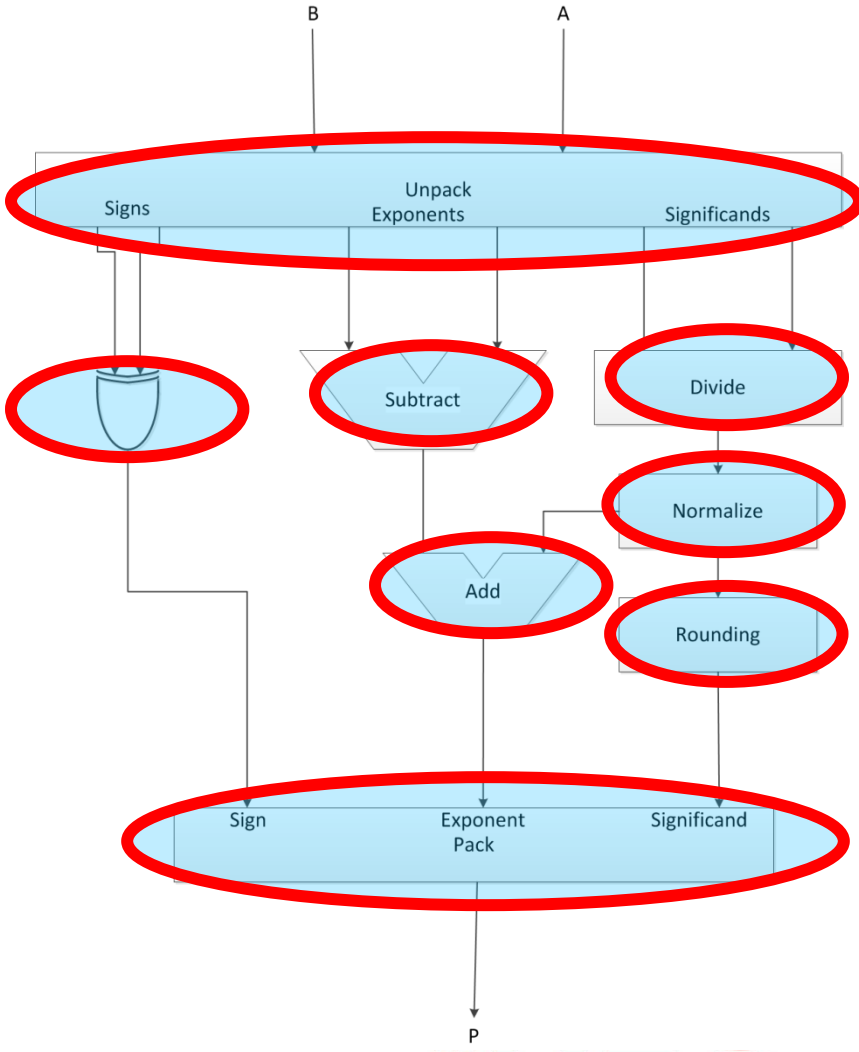
- Pipelining is also important
- 6 stages is required here
- Unsigned multiplier is required
- Barrel shifter is used in the normalization stage (refer to slides 12 – 15)



Technische Universität Berlin

Floating Point Divider

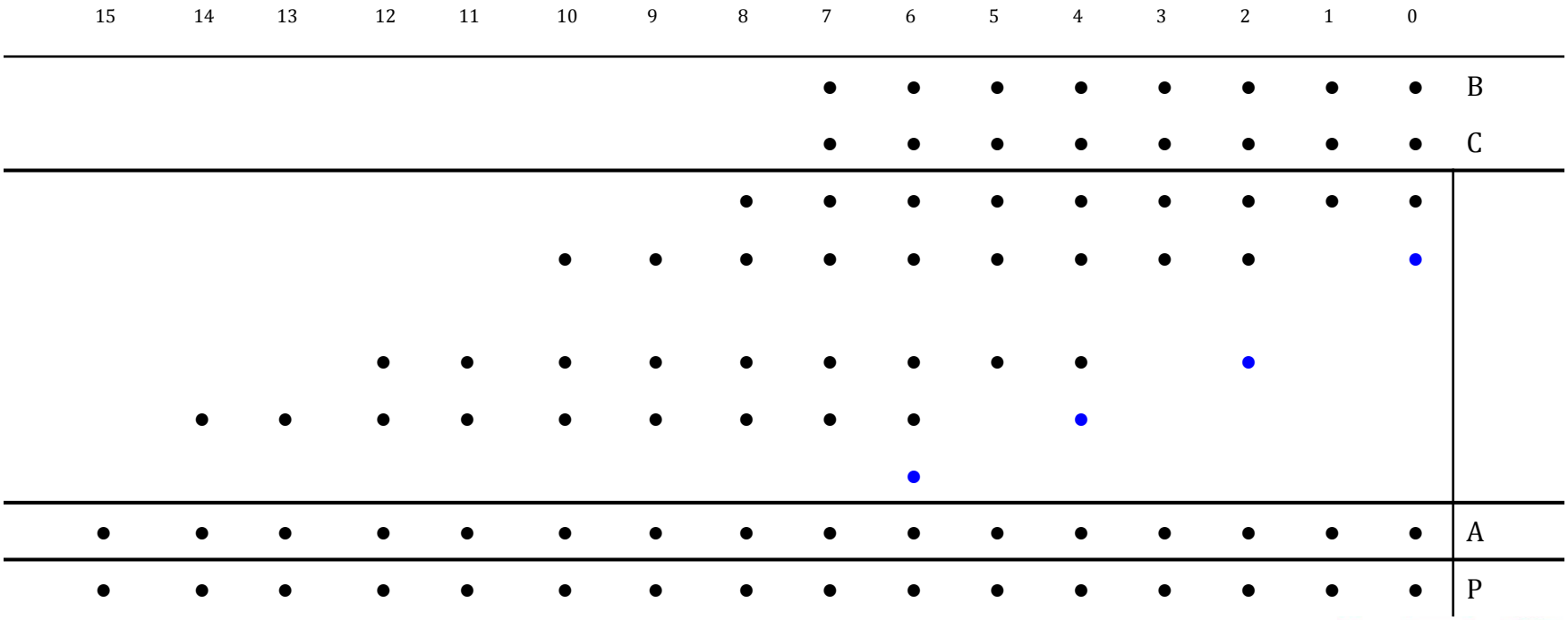
- Multiplication is replaced with division
- Exponents are subtracted not added
- All other components remain the same



Technische Universität Berlin

Multiply Accumulate

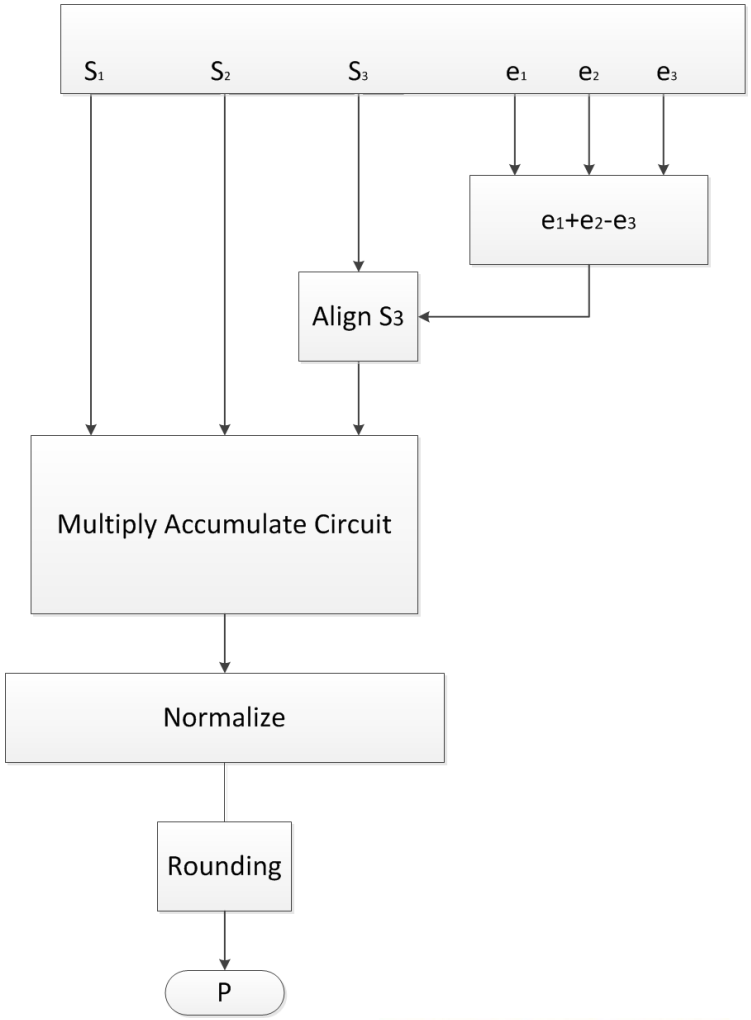
- Multiply accumulate operation is a basic operation in several applications
- Multiplication can be fused with addition in the same circuit



Technische Universität Berlin

Multiply Accumulate Circuit

- Modified version of the adder circuit
- The adder is replace by the fused MAC circuit
- Exponents are calculated from the maximum value of e1+e2 and e3



Technische Universität Berlin

Error Sources in Floating Point Arithmetic Circuits

- There are several sources in floating point arithmetic circuits
  - Representation Errors: Error due to the limited number of bits in the significant
  - Computation Errors: Error due to the rounding stage in all arithmetic circuit
- This error accumulate after several operations
- Increasing the significant part reduce representation error
- Intermediate representation during calculation can reduce final error
- The accumulated error violates the laws of algebra
  - $A + (B + C) \neq (A + B) + C$

Verify ?

Technische Universität Berlin

Summary

- Adder Circuit
  - Controlled Shifter – Significant alignment
  - Normalization
  - Rounding
  - Sign Control and Exception Generation
- Multiplier Circuit
- Divider Circuit
- Multiply Accumulate Circuit
- Error Analysis

Technische Universität Berlin

Questions ?

Questions ?