Self-Timed Fused Multiply-Add Unit Performance Improvement

I. Sokolov, Y. Stepchenkov, Y. Rogdestvenski, Y. Diachenko, A. Rogdestvenskene, D. Diachenko

Institute of Informatics Problems, Federal Research Center "Computer Science and Control" of the Russian Academy of Sciences

Content

- What are the Self-timed circuits?
- Fused multiply-add (FMA) unit structure
- How can we accelerate self-timed FMA?
- Practical techniques improving self-timed FMA's performance
- Conclusions

Self-Timed Circuits

- They are free of global synchronization and operate based on a request-acknowledge interaction between circuit parts.
- Major principles:
  - ✓ two-phase work discipline,
  - ✓ mandatory detection and acknowledging completion of all switches initiated during the transition to the current phase of work

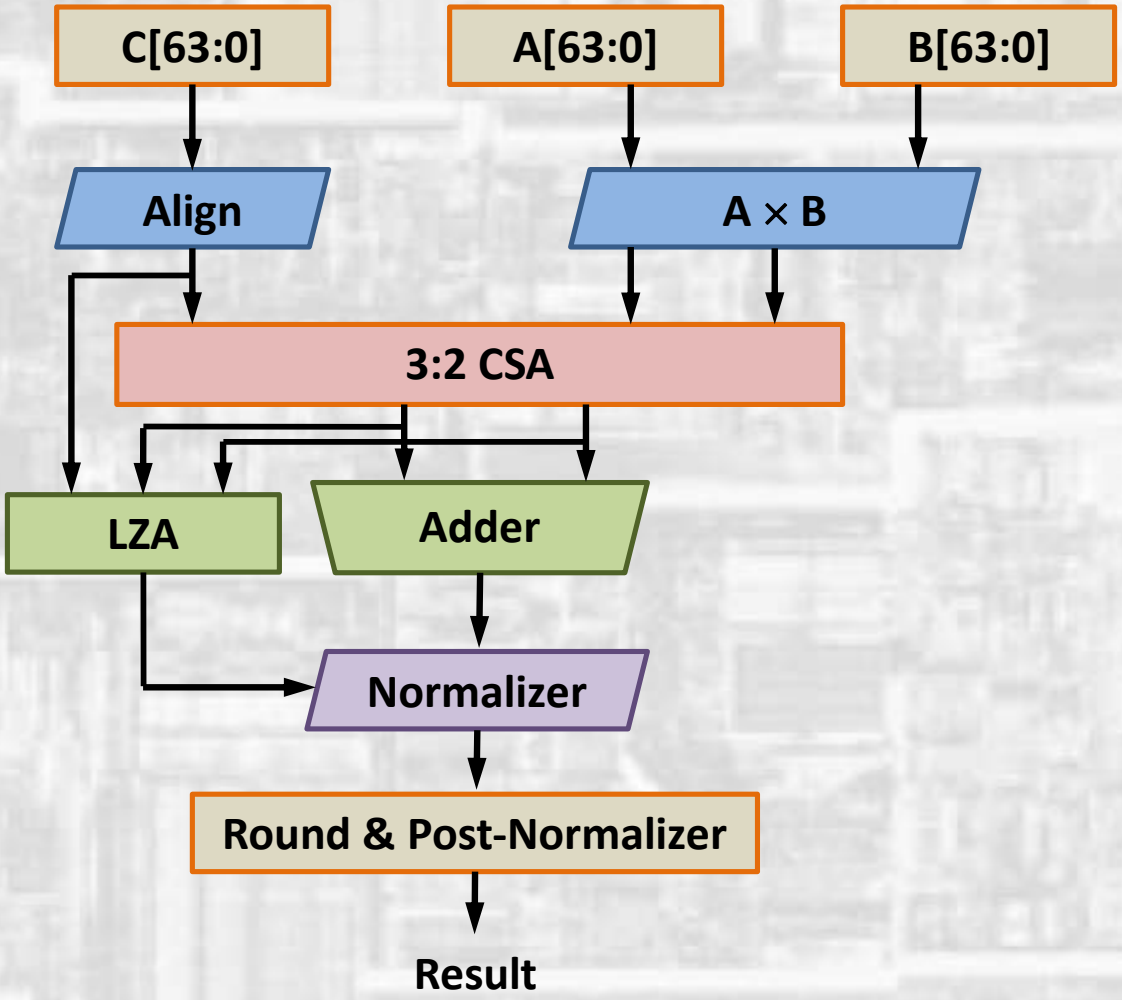
Self-Timed Circuit's Advantages

- Absence of a global clock tree accounting for up to 30% of total power consumption
- Absence of "races" between internal signals
- Correct operation under any operating conditions (supply voltage and ambient temperature)
- Their performance is determined by the actual logic cells' delays in the current operating conditions, not the worst case

Self-Timed Circuit's Drawbacks

- Increased hardware complexity because of redundant data coding and indication subcircuit
- Two-phase operation mode decelerates self-timed circuit's total performance
- Indication subcircuit causes an additional delay increasing in multi-bit units
- Direct conversion of synchronous circuits to self-timed ones can lead to ineffective solutions in performance or an excessive hardware costs

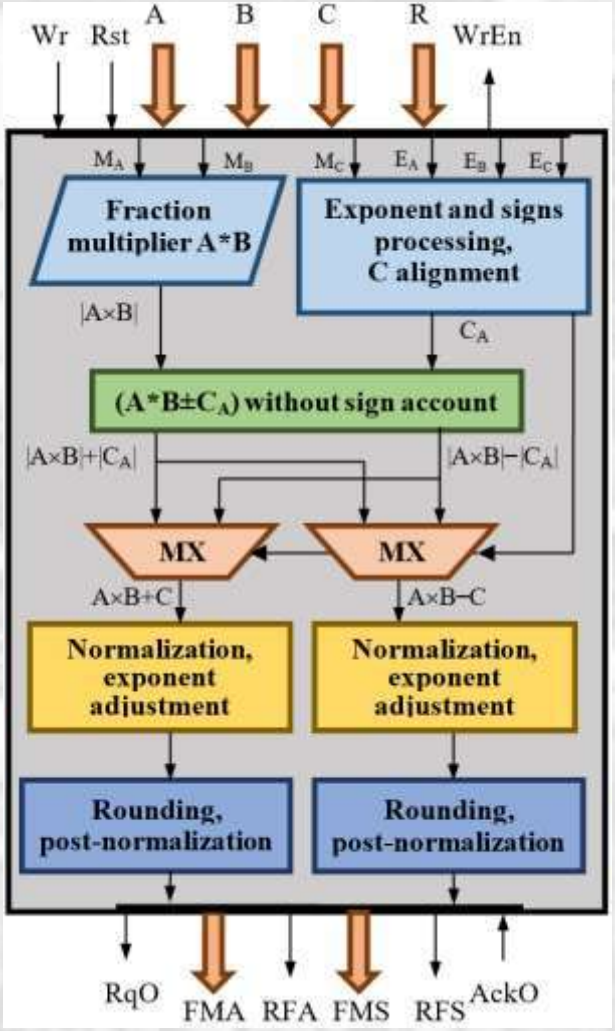
Fused multiply-add (FMA) unit



Cases simplifying calculations

exp(C) – exp(A × B)	Result	Normalization order	Round
[57, ∞)	C	–	–
[2, 56]	A × B ± aligned_C	n	+
[–1, 1]	A × B ± aligned_C	n	+
[–55, –2]	A × B ± C	1	+
(–∞, –56]	A × B	1	+

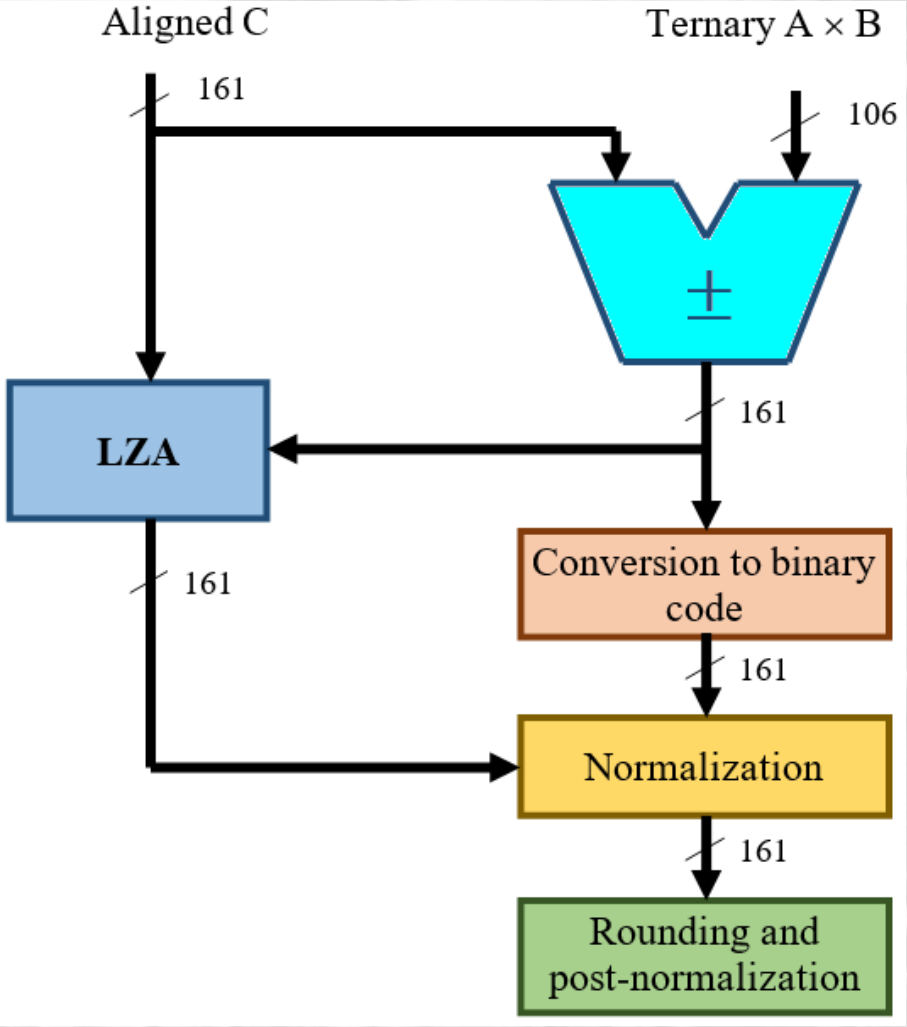
Self-Timed Fused Multiply-Add-Subtract (FMAS) Unit



A, B, C – input operands
R – operation options
Wr – force write input
Rst – asynchronous reset
WrEn – write enable output
RqO – result ready output
FMA – multiply-add result
FMS – multiply-subtract result
RFA – flags of FMA operation
RFS – flags of FMS operation
AckO – read result acknowledging

- IEEE 754 Standard Compliant
- Ternary multiplier and adder-subtractor

Increasing FMAS Performance (1)



Ternary coding simplifies and accelerates adding-subtracting multi-bit operands

Increasing FMAS Performance (2)

An important feature of the self-timed circuits:
their operation time depends on the processed data

***Rule 1*: Do not tie the estimated operating time of individual units and pipeline stages to the maximum time for the worst operating conditions, but use the statistically average operating time.**

***Rule 2*: Use new algorithmic solutions that may be ineffective in synchronous circuitry as they decrease the unit's performance in the worst case.**

Increasing FMAS Performance (3)

The ways to increase self-timed FMAS performance:

- **Using LZA pre-calculation with reduced bit-width**
- **Reducing bit-width of output registers for sum and difference of the product and the third operand due to parallelization of data processing.**
- **Speeding up normalization and post-normalization.**

Optimized Normalization (1)

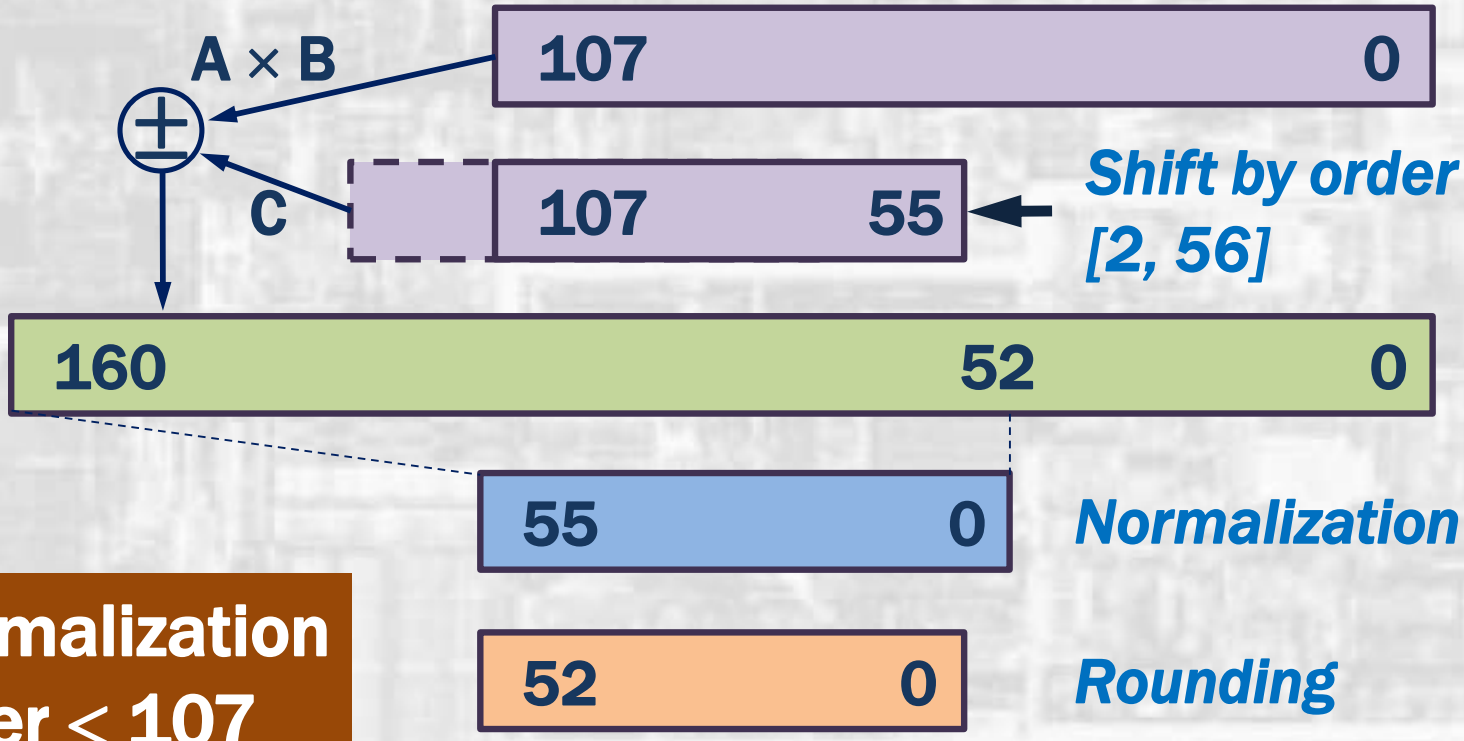
The main idea of the proposed improvement:

- **Align the largest of the third operand or product in the most significant bits of the adder-subtractor,**
- **Shift the rest one by order corresponding to the difference of their exponents.**

Optimized Normalization (2)

Typical algorithm for the case

“exp(C) – exp(A × B)” ∈ [2, 56]

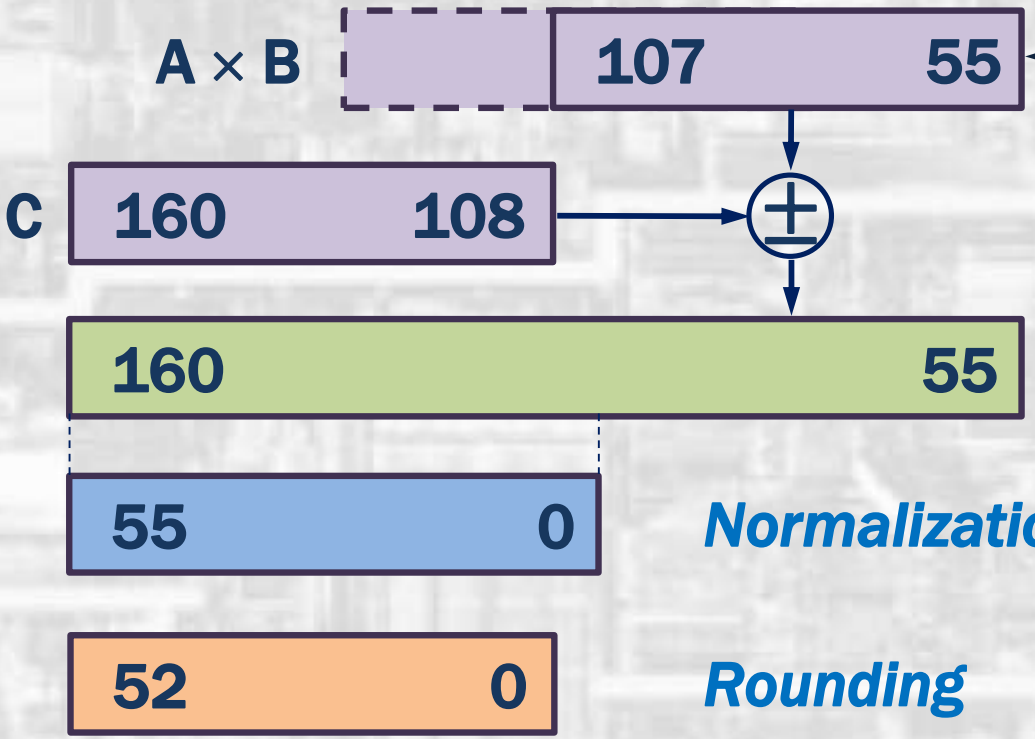


Normalization order < 107

Optimized Normalization (3)

Proposed algorithm for the case

“exp(C) – exp(A × B)” ∈ [2, 56]

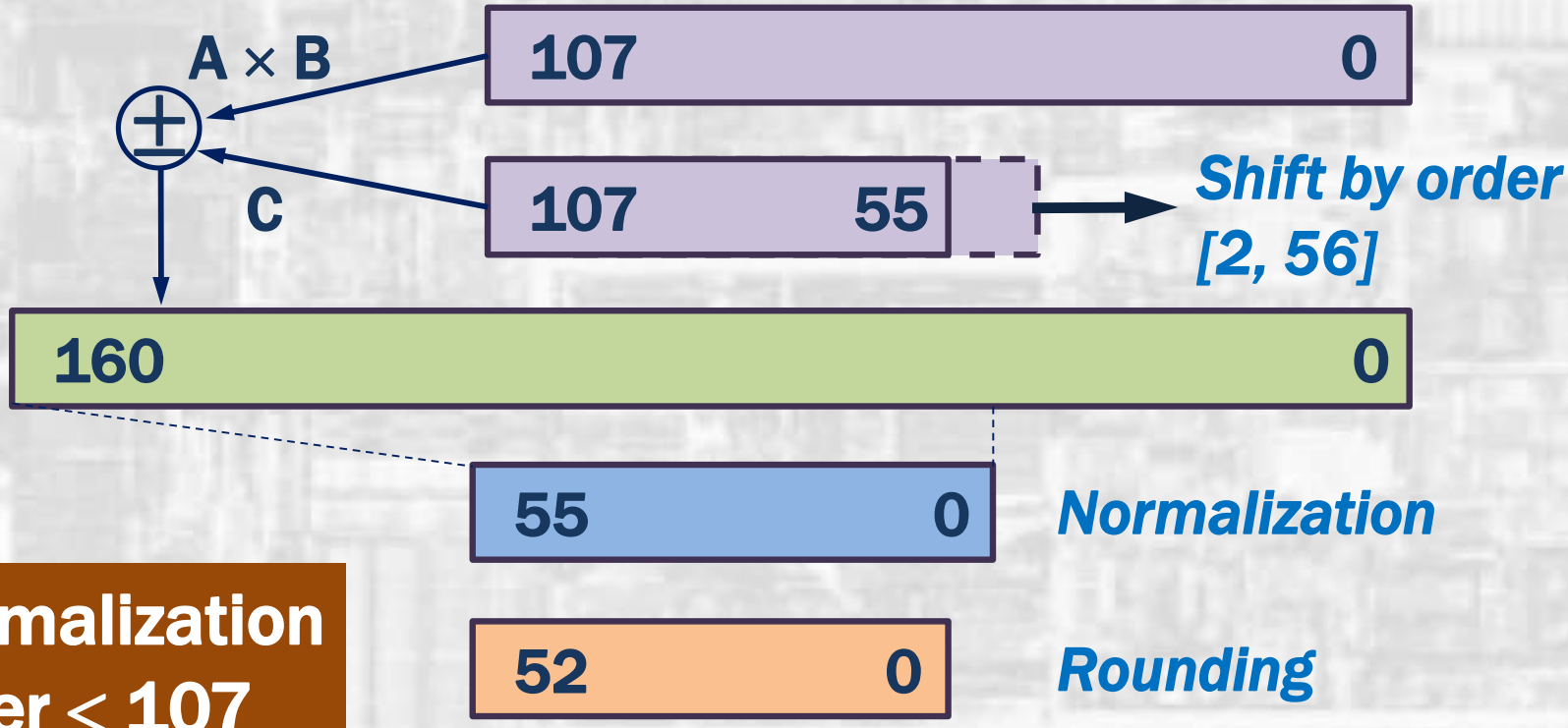


Normalization order < 57 !

Optimized Normalization (4)

Typical algorithm for the case

“exp(C) – exp(A × B)” ∈ [–56, –2]

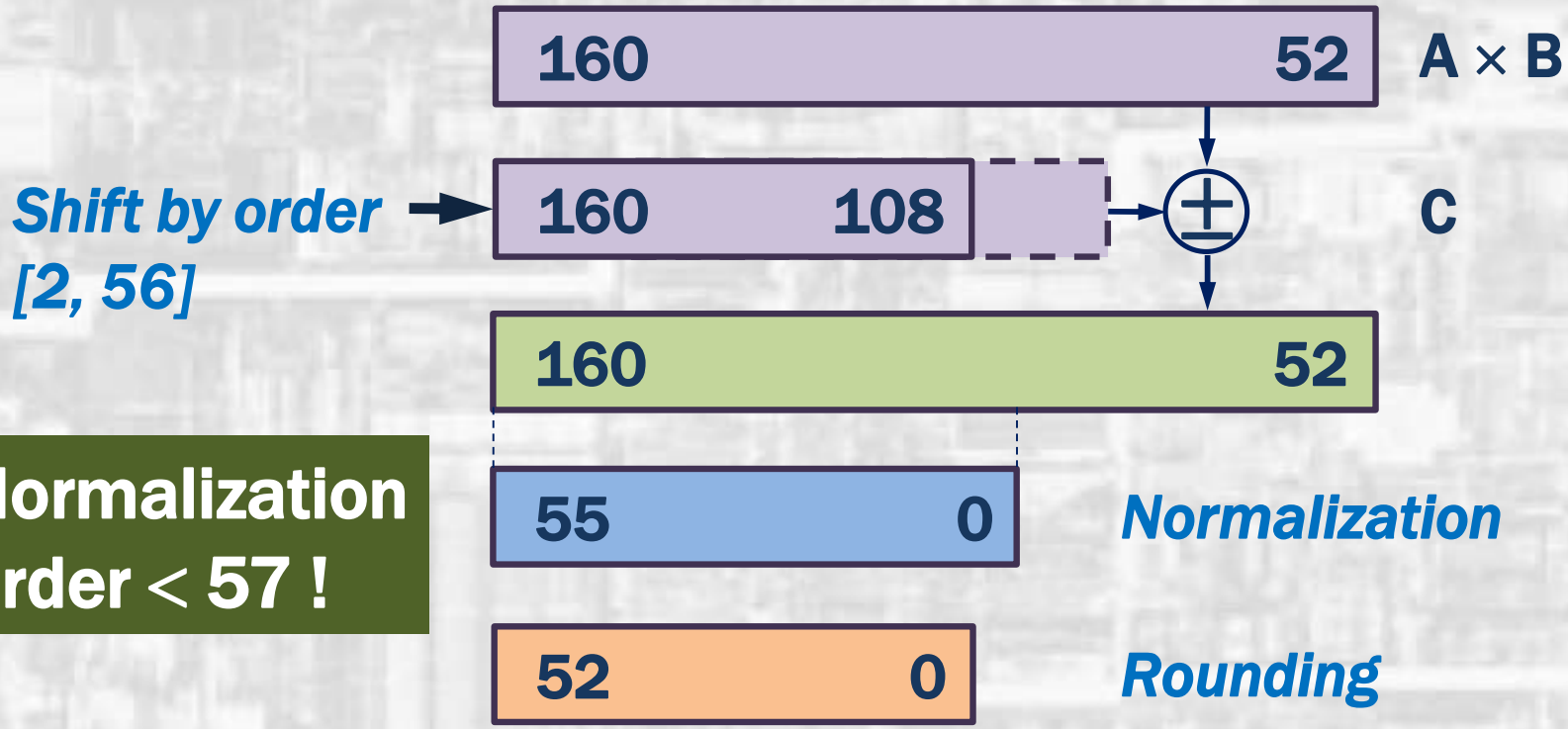


Normalization order < 107

Optimized Normalization (5)

Proposed algorithm for the case

“exp(C) – exp(A × B)” ∈ [–56, –2]



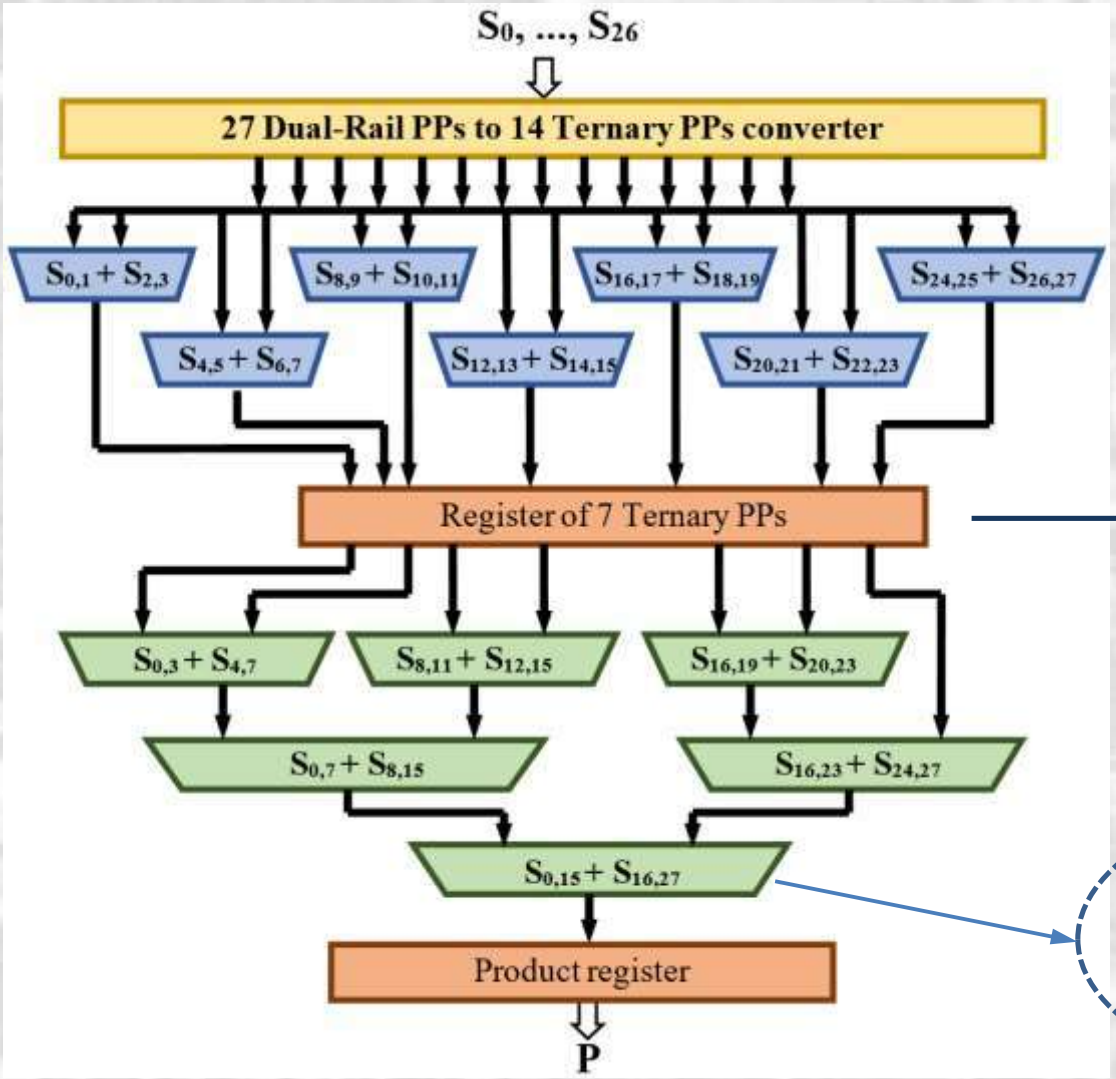
Normalization order < 57 !

Optimized Normalization (6)

Optimization results:

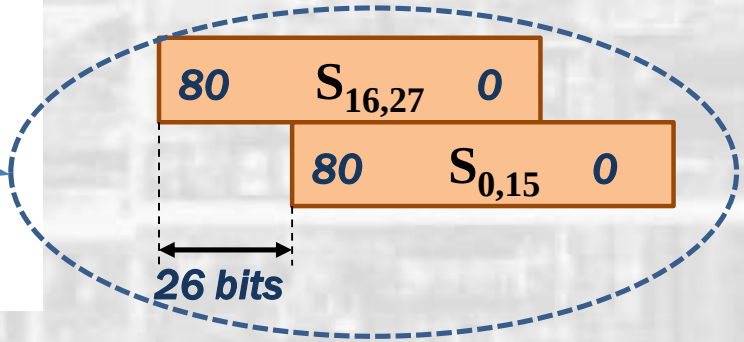
1. **The bit-width of the operand to be normalized is reduced from 161 to 106**
2. **The duration of all basic operations of the critical cycle is decreased by 15 – 20 %**
3. **Hardware complexity of the self-timed fused multiply-add-subtract unit is efficiently reduced**

Ternary Wallace Tree

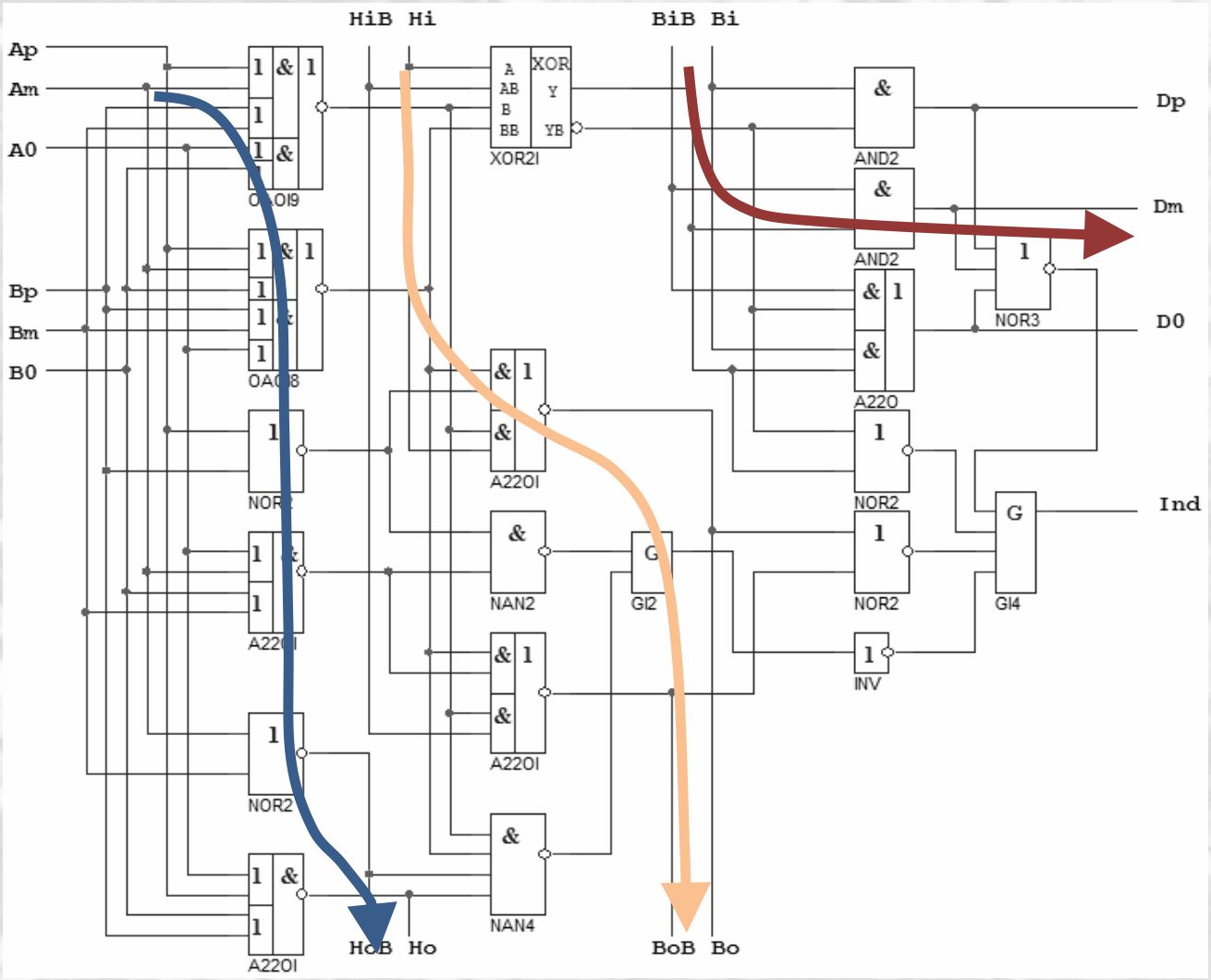


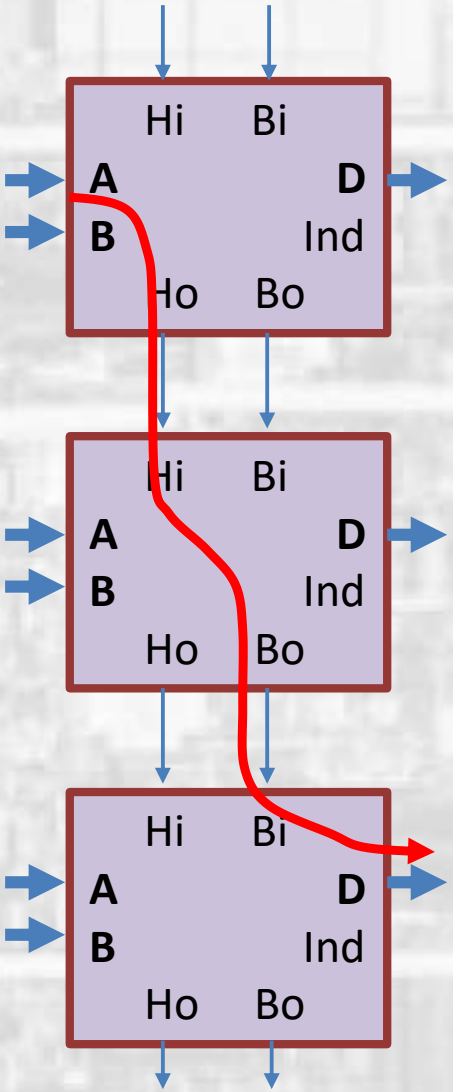
1-st stage

2-nd stage

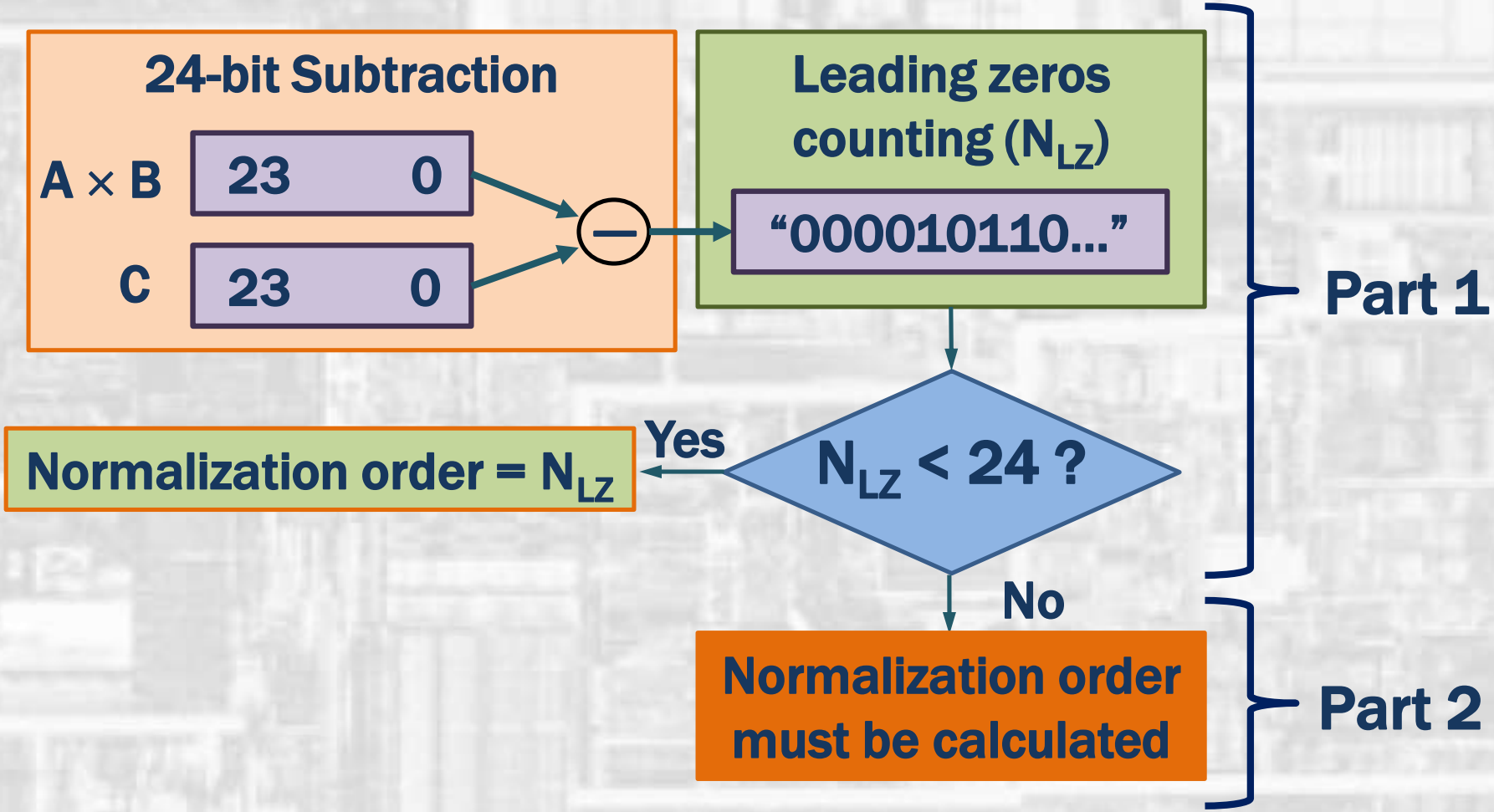


Carry Propagation Paths in the Ternary Adder





LZA Unit


24-bit Subtraction
A × B
23
0
C
23
0
−
Leading zeros counting (N_{LZ})
"000010110..."
N_{LZ} < 24 ?
Yes
Normalization order = N_{LZ}
No
Normalization order must be calculated
Part 1
Part 2

Conclusions

- Self-timed circuit operation discipline allows for wider using speculative result calculation techniques on a base of detailed specifying the possible cases of the input operands ratio
- The proposed normalization technique reduces a bit-width of processed data (by 1.5 times), duration of all primary operations of the critical cycle (in 15-20%), and hardware complexity (in 20-25%)
- Preliminary estimation of the leading zero number of the most significant bits of product and third operand (24 for double precision or 12 for single precision) decreases by 32% an average delay, added by normalization pipeline stage to total Fused Multiply-Add-Subtract unit's delay

Contacts

- **Director:** Academician ***Igor Sokolov***
- **Address:** Institute of Informatics Problems, Federal Research Center “Computer Science and Control" of the Russian Academy of Sciences,
  44 b.2 Vavilova str., Moscow, 119333
- **Phone: +**7 (495) 137 34 94
- **Fax: +**7 (495) 930 45 05
- **E-mail:** isokolov@ipiran.ru

- **Speaker*: Diachenko Yury*** **diaura@mail.ru**