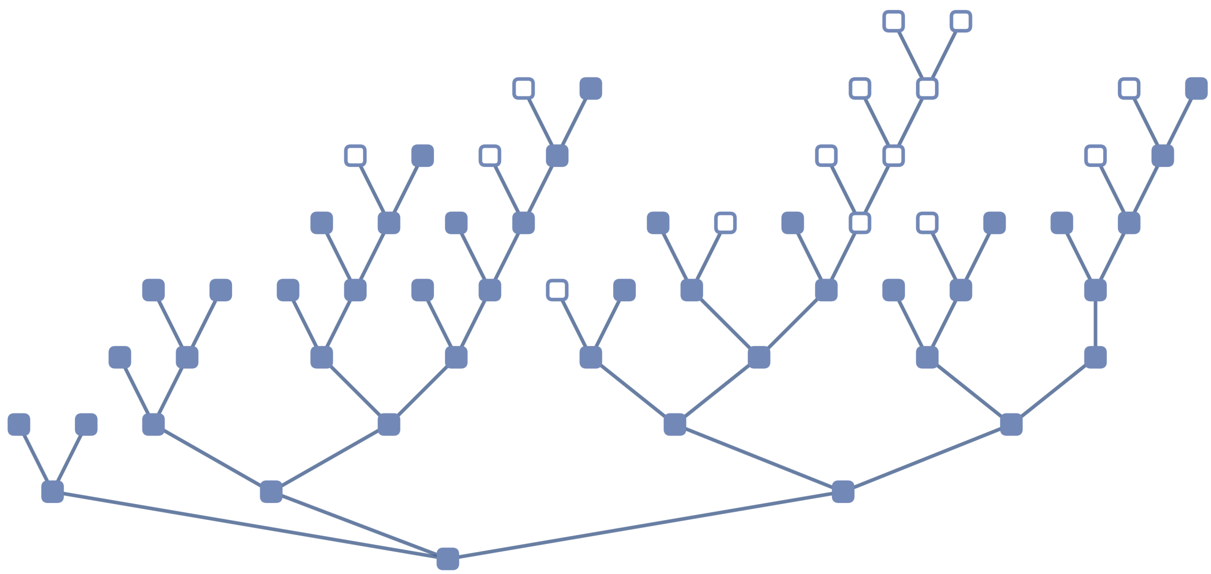
Inventions No. 4
sci–sci.org/2aalu
DOI: 10.5281/zenodo.17526246

SciSci Research
サイサイ・リサーチ

Future Computing



Hensel CPU Arithmetic Logic Units

Circuit Design for Exact Computing with 2-adic Arithmetic

James Douglas Boyd

SciSci Research, Inc. + Future Computing

SciSci Research, Inc.
Boulder, Colorado, United States
www.sci-sci.org

Citation Format:

Boyd, J.D. (2025). Hensel CPU Arithmetic Logic Units: Circuit Design for Exact Computing with 2-adic Arithmetic. SciSci Inventions, 1(4). DOI: 10.5281/zenodo.17526246

Contents

1 2-Adic Arithmetic Logic

The Hensel CPU is being developed by SciSci Research and Future Computing to realize an exact computing capability to replace floating-point computation. Unlike floating point numbers, which are approximations of numbers in $\mathbb{R}$, the Hensel CPU performs exact arithmetic in $\mathbb{Q}_2$, or, moreover, on finite encodings of 2-adic numbers. Arithmetic operations are executed in the Hensel CPU processor by 2-adic arithmetic logic units (2AALUs). At the circuit level, 2AALU operations on FC-3-2025-encoded operands (i.e., finite encodings of coefficients of 2-adic expansions) are performed via combinational logic by Boolean logic gates. Indeed, coefficient values in 2-adic expansions are either 0 or 1, as, in turn, are the entries in FC-3-2025 encodings of operands. Thus, exact computing in $\mathbb{Q}_2$ with the Hensel CPU is compatible with extant MOSFET technology; 2AALUs compute in bits with normal logic gates. Given herein is a circuit-level description of 2AALU arithmetic.

1.1 Setup

We will consider a Hensel CPU model more expansive than that seen in the forthcoming Virtual Hensel demonstration. Here, we will consider a Hensel capable of storing operands up to 30 bits in length, where the respective maximal R_{FC}, L_{FC}, and T_{FC} lengths are 10 bits each. This Hensel can perform arithmetic operations on operands up to 15 bits in length, where the respective maximal R_{FC}, L_{FC}, and T_{FC} lengths are 5 bits each. According to the Hensel architecture, such a CPU would have a cluster of 1024 processor registers for distributed load-store of operands. As previously discussed, a commercial-grade Hensel will have a large bit-width, larger than that considered herein. A key point to underscore in this report is, even for high bit widths, the relative simplicity by which 2-adic operands can be subject to arithmetic.

This report will describe binary additive operations, i.e., addition as performed on input operand pairs. Arithmetic proceeds as follows. We begin with operands $\chi^{\partial}(\mathsf{FC}(q))$ and $\chi^{\partial}(\mathsf{FC}(p))$ and an arithmetic operation $\mu := \Upsilon^{\mathsf{post}}(\Upsilon(-))$. Here, Υ is the main arithmetic step, involving addition on χ^{∂} entries that do not require carrying 1 entries to extra "digit" places. This is the most computationally demanding procedure, i.e., it employs the most gates. In our case, Υ acts on $\chi^{\partial}(\mathsf{FC}(p))$ and $\chi^{\partial}(\mathsf{FC}(q))$, which are each five bits in length, giving an output that is also five bits in length. Next, Υ^{post} is applied to $\Upsilon(-)$, and supplies extra-"digit" 1 values as needed.

The combinational logic for the above procedure is written out, and circuit diagrams are provided. Additionally, it is explained how FC-2-2025 instructions, which have previously been mentioned in the Virtual Hensel and load-store reports, are computed. (It's nothing more than a simple XOR gate computation.)

For a given pair of operands $\mathsf{FC}(q)$ and $\mathsf{FC}(p)$, the above procedures are performed for 1) $\chi^{\partial}_{(*,-,-)}(\mathsf{FC}(p))$ and $\chi^{\partial}_{(*,-,-)}(\mathsf{FC}(q))$, 2) $\chi^{\partial}_{(-,*,-)}(\mathsf{FC}(p))$ and $\chi^{\partial}_{(-,*,-)}(\mathsf{FC}(q))$, and 3) $\chi^{\partial}_{(-,-,*)}(\mathsf{FC}(p))$ and $\chi^{\partial}_{(-,-,*)}(\mathsf{FC}(q))$. All entries are fed from right to left; thus, in the case of $\chi^{\partial}_{(-,-,*)}(\mathsf{FC}(p))$ and $\chi^{\partial}_{(-,-,*)}(\mathsf{FC}(q))$ (i.e., T_{FC}), the entries are reversed, beginning with the right-most pair $({}^{\mathsf{T}}a^p_5, {}^{\mathsf{T}}a^q_5)$ and ending with the left-most pair $({}^{\mathsf{T}}a^p_1, {}^{\mathsf{T}}a^q_1)$. In the case of $\chi^{\partial}_{(*,-,-)}(\mathsf{FC}(p))$ and $\chi^{\partial}_{(*,-,-)}(\mathsf{FC}(q))$ (i.e., R_{FC}), as well as $\chi^{\partial}_{(-,*,-)}(\mathsf{FC}(p))$ and $\chi^{\partial}_{(-,*,-)}(\mathsf{FC}(q))$ (i.e., L_{FC}), addition begins with $({}^{\mathsf{R}}a^p_1, {}^{\mathsf{R}}a^q_1)$ or $({}^{\mathsf{L}}a^p_1, {}^{\mathsf{L}}a^q_1)$. Here, we will simply write (a^p_1, a^q_1) to refer to the first entry for a general pair, with it understood that this is $({}^{\mathsf{T}}a^p_5, {}^{\mathsf{T}}a^q_5)$, $({}^{\mathsf{R}}a^p_1, {}^{\mathsf{R}}a^q_1)$, or $({}^{\mathsf{L}}a^p_1, {}^{\mathsf{L}}a^q_1)$. The same follows accordingly that (a^p_5, a^q_5) can be $({}^{\mathsf{T}}a^p_1, {}^{\mathsf{T}}a^q_1)$, $({}^{\mathsf{R}}a^p_5, {}^{\mathsf{R}}a^q_5)$, or $({}^{\mathsf{L}}a^p_5, {}^{\mathsf{L}}a^q_5)$. The output will be written from left to right, with the (a^p_1, a^q_1) output furthest to the left and the (a^p_5, a^q_5) furthest to the right.

2 Combinational Logic

2.1 Length-5 Addition

We begin with Υ procedures, each of which yields an output ω_i $(1 \leq i \leq 5)$. The first output entry, ω_1, is obtained thusly:

$$\mathrm{Xor}[a_1^p, a_1^q]$$

The Υ procedure for ω_2 is as follows:

$$\begin{array}{r}
\mathrm{Or}[\mathrm{Xor}[\mathrm{And}[\mathrm{And}[a_1^p, a_1^q], \mathrm{Nor}[a_2^p, a_2^q]], \\
\mathrm{And}[\mathrm{Nand}[a_1^p, a_1^q], \mathrm{Xor}[a_2^p, a_2^q]]], \\
\mathrm{And}[\mathrm{And}[a_1^p, a_1^q], \mathrm{And}[a_2^p, a_2^q]]]
\end{array}$$

The Υ procedure for ω_3 is as follows:

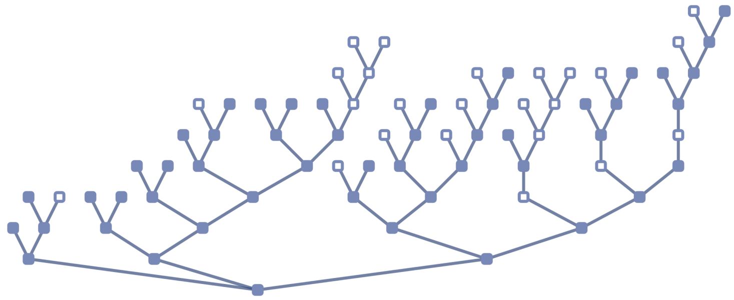
$$\begin{array}{r}
\mathrm{And}[\mathrm{And}[\mathrm{And}[\mathrm{Or}[\mathrm{Or}[\mathrm{Or}[\mathrm{Or}[\\
\mathrm{Or}[\mathrm{Or}[\mathrm{And}[\mathrm{And}[\mathrm{And}[a_1^p, a_1^q], \mathrm{Or}[a_2^p, a_2^q]], \\
\mathrm{Nor}[a_3^p, a_3^q]], \mathrm{And}[\mathrm{And}[a_2^p, a_2^q], \\
\mathrm{Nor}[a_3^p, a_3^q]]], \mathrm{Or}[\mathrm{And}[\mathrm{Nor}[a_2^p, a_2^q], \\
\mathrm{Xor}[a_3^p, a_3^q]], \mathrm{And}[\mathrm{And}[\mathrm{Nand}[a_1^p, a_1^q], \\
\mathrm{Nand}[a_2^p, a_2^q]], \mathrm{Xor}[a_3^p, a_3^q]]]], \\
\mathrm{And}[\mathrm{And}[a_2^p, a_2^q], \mathrm{And}[a_3^p, a_3^q]]], \\
\mathrm{And}[\mathrm{And}[\mathrm{And}[\mathrm{And}[a_1^p, a_1^q], \mathrm{Or}[a_2^p, a_2^q]], \\
\mathrm{And}[a_3^p, a_3^q]], \mathrm{Nor}[a_4^p, a_4^q]]], \\
\mathrm{And}[\mathrm{And}[\mathrm{Xor}[a_4^p, a_4^q], \mathrm{And}[a_3^p, a_3^q]], \\
\mathrm{Nor}[a_5^p, a_5^q]]], \mathrm{And}[\mathrm{And}[\mathrm{And}[\mathrm{And}[a_1^p, a_1^q], \\
\mathrm{Or}[a_2^p, a_2^q]], \mathrm{And}[a_3^p, a_3^q]], \\
\mathrm{And}[a_4^p, a_4^q]], \mathrm{Nor}[a_5^p, a_5^q]]], \\
\mathrm{Not}[\mathrm{And}[\mathrm{And}[\mathrm{Nand}[a_1^p, a_1^q], \\
\mathrm{Xor}[a_2^p, a_2^q]], \mathrm{And}[a_3^p, a_3^q]], \\
\mathrm{Xor}[a_4^p, a_4^q]]]], \mathrm{Not}[\mathrm{And}[\mathrm{Nor}[a_2^p, a_2^q], \\
\mathrm{And}[a_3^p, a_3^q]], \mathrm{Xor}[a_4^p, a_4^q]]]]]
\end{array}$$

The Υ procedure for ω_4 is as follows:

$$\begin{array}{r}
\mathrm{And}[\mathrm{And}[\mathrm{And}[\mathrm{Xor}[\mathrm{Or}[\mathrm{Or}[\\
\mathrm{And}[\mathrm{And}[\mathrm{And}[\mathrm{And}[a_1^p, a_1^q], \mathrm{Or}[a_2^p, a_2^q]], \\
\mathrm{Or}[a_3^p, a_3^q]], \mathrm{Nor}[a_4^p, a_4^q]], \\
\mathrm{And}[\mathrm{And}[\mathrm{And}[a_2^p, a_2^q], \mathrm{Or}[a_3^p, a_3^q]], \\
\mathrm{Nor}[a_4^p, a_4^q]]], \mathrm{And}[\mathrm{And}[a_3^p, a_3^q], \mathrm{Nor}[a_4^p, a_4^q]]], \\
\mathrm{Or}[\mathrm{Or}[\mathrm{Or}[\mathrm{Or}[\mathrm{Or}[\mathrm{And}[\mathrm{And}[\mathrm{And}[\\
\mathrm{Nand}[a_1^p, a_1^q], \mathrm{Nand}[a_2^p, a_2^q]], \\
\mathrm{Nand}[a_3^p, a_3^q]], \mathrm{Xor}[a_4^p, a_4^q]], \\
\mathrm{And}[\mathrm{Nor}[a_3^p, a_3^q], \mathrm{Xor}[a_4^p, a_4^q]]], \\
\mathrm{And}[\mathrm{And}[\mathrm{Nor}[a_2^p, a_2^q], \mathrm{Nand}[a_3^p, a_3^q]], \\
\mathrm{Xor}[a_4^p, a_4^q]]], \mathrm{And}[\mathrm{Nor}[a_3^p, a_3^q], \\
\mathrm{Xor}[a_4^p, a_4^q]]], \mathrm{And}[\mathrm{And}[a_3^p, a_3^q], \\
\mathrm{And}[a_4^p, a_4^q]]], \mathrm{And}[\mathrm{And}[\mathrm{Xor}[a_3^p, a_3^q], \\
\mathrm{And}[a_4^p, a_4^q]], \mathrm{Nor}[a_5^p, a_5^q]]]], \\
\mathrm{Not}[\mathrm{And}[\mathrm{And}[\mathrm{And}[\mathrm{Xor}[a_1^p, a_1^q], \\
\mathrm{Xor}[a_2^p, a_2^q]], \mathrm{Xor}[a_3^p, a_3^q]], \\
\mathrm{And}[a_4^p, a_4^q]]]], \mathrm{Not}[\mathrm{And}[\mathrm{And}[\mathrm{Nor}[a_2^p, a_2^q], \\
\mathrm{Xor}[a_3^p, a_3^q]], \mathrm{And}[a_4^p, a_4^q]]]], \\
\mathrm{Not}[\mathrm{And}[\mathrm{And}[\mathrm{And}[\mathrm{Nor}[a_1^p, a_1^q], \mathrm{Xor}[a_2^p, a_2^q]], \\
\mathrm{Xor}[a_3^p, a_3^q]], \mathrm{And}[a_4^p, a_4^q]]]]
\end{array}$$

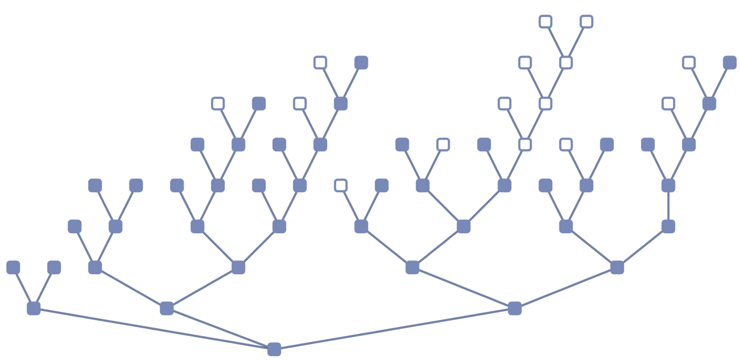
The Υ procedure for ω_5 is as follows:

$$\begin{array}{r}
\mathrm{Or}[\mathrm{Or}[\mathrm{Or}[\mathrm{Or}[\mathrm{Or}[\mathrm{Or}[\\
\mathrm{And}[\mathrm{And}[\mathrm{And}[\mathrm{And}[\mathrm{And}[a_1^p, a_1^q], \\
\mathrm{Or}[a_2^p, a_2^q]], \mathrm{Or}[a_3^p, a_3^q]], \\
\mathrm{Or}[a_4^p, a_4^q]], \mathrm{Nor}[a_5^p, a_5^q]], \\
\mathrm{And}[\mathrm{And}[\mathrm{And}[\mathrm{And}[a_2^p, a_2^q], \mathrm{Or}[a_3^p, a_3^q]], \\
\mathrm{Or}[a_4^p, a_4^q]], \mathrm{Nor}[a_5^p, a_5^q]]], \\
\mathrm{And}[\mathrm{And}[\mathrm{And}[a_3^p, a_3^q], \mathrm{Or}[a_4^p, a_4^q]], \\
\mathrm{Nor}[a_5^p, a_5^q]]], \\
\mathrm{And}[\mathrm{And}[a_4^p, a_4^q], \mathrm{Nor}[a_5^p, a_5^q]]], \\
\mathrm{Or}[\mathrm{Or}[\mathrm{And}[\mathrm{And}[\mathrm{And}[\mathrm{And}[\mathrm{Nand}[a_1^p, a_1^q], \\
\mathrm{Nand}[a_2^p, a_2^q]], \mathrm{Nand}[a_3^p, a_3^q]], \\
\mathrm{Nand}[a_4^p, a_4^q]], \mathrm{Xor}[a_5^p, a_5^q]], \\
\mathrm{And}[\mathrm{Nor}[a_4^p, a_4^q], \mathrm{Xor}[a_5^p, a_5^q]]], \\
\mathrm{And}[\mathrm{And}[a_4^p, a_4^q], \mathrm{And}[a_5^p, a_5^q]]]], \\
\mathrm{And}[\mathrm{And}[\mathrm{Nor}[a_3^p, a_3^q], \mathrm{Nand}[a_4^p, a_4^q]], \mathrm{Xor}[a_5^p, a_5^q]]], \\
\mathrm{And}[\mathrm{And}[\mathrm{And}[\mathrm{Nor}[a_2^p, a_2^q], \mathrm{Nand}[a_3^p, a_3^q]], \\
\mathrm{Nand}[a_4^p, a_4^q]], \mathrm{Xor}[a_5^p, a_5^q]]]
\end{array}$$



And[And[And[Xor[Or[Or[
And[And[And[And[a^p_1, a^q_1], Or[a^p_2, a^q_2]],
Or[a^p_3, a^q_3]], Nor[a^p_4, a^q_4]],
And[And[And[a^p_2, a^q_2], Or[a^p_3, a^q_3]],
Nor[a^p_4, a^q_4]]], And[And[a^p_3, a^q_3], Nor[a^p_4, a^q_4]]],
Or[Or[Or[Or[Or[And[And[And[
Nand[a^p_1, a^q_1], Nand[a^p_2, a^q_2]],
Nand[a^p_3, a^q_3]], Xor[a^p_4, a^q_4]],
And[Nor[a^p_3, a^q_3], Xor[a^p_4, a^q_4]]],
And[And[Nor[a^p_2, a^q_2], Nand[a^p_3, a^q_3]],
Xor[a^p_4, a^q_4]]], And[Nor[a^p_3, a^q_3],
Xor[a^p_4, a^q_4]]], And[And[a^p_3, a^q_3],
And[a^p_4, a^q_4]]], And[And[Xor[a^p_3, a^q_3],
And[a^p_4, a^q_4]], Nor[a^p_5, a^q_5]]]],
Not[And[And[And[Xor[a^p_1, a^q_1],
Xor[a^p_2, a^q_2]], Xor[a^p_3, a^q_3]],
And[a^p_4, a^q_4]]]], Not[And[And[Nor[a^p_2, a^q_2],
Xor[a^p_3, a^q_3]], And[a^p_4, a^q_4]]]],
Not[And[And[And[Nor[a^p_1, a^q_1], Xor[a^p_2, a^q_2]],
Xor[a^p_3, a^q_3]], And[a^p_4, a^q_4]]]]

Figure 1: Illustration of correspondence between the combinational procedure for ω_4 and the graph structure of its circuit. (Inputs are omitted from graph.) In this case, $a^p_1 = \text{True}$, $a^q_1 = \text{False}$, $a^p_2 = \text{True}$, $a^1_2 = \text{False}$, $a^p_3 = \text{False}$, $a^q_3 = \text{True}$, $a^p_4 = \text{False}$, $a^q_4 = \text{False}$, $a^p_5 = \text{True}$, and $a^q_5 = \text{False}$.



Or[Or[Or[Or[Or[Or[
And[And[And[And[And[a^p_1, a^q_1],
Or[a^p_2, a^q_2]], Or[a^p_3, a^q_3]],
Or[a^p_4, a^q_4]], Nor[a^p_5, a^q_5]],
And[And[And[And[a^p_2, a^q_2], Or[a^p_3, a^q_3]],
Or[a^p_4, a^q_4]], Nor[a^p_5, a^q_5]]],
And[And[And[a^p_3, a^q_3], Or[a^p_4, a^q_4]],
Nor[a^p_5, a^q_5]]],
And[And[a^p_4, a^q_4], Nor[a^p_5, a^q_5]]],
Or[Or[And[And[And[And[Nand[a^p_1, a^q_1],
Nand[a^p_2, a^q_2]], Nand[a^p_3, a^q_3]],
Nand[a^p_4, a^q_4]], Xor[a^p_5, a^q_5]],
And[Nor[a^p_4, a^q_4], Xor[a^p_5, a^q_5]]],
And[And[a^p_4, a^q_4], And[a^p_5, a^q_5]]]],
And[And[Nor[a^p_3, a^q_3], Nand[a^p_4, a^q_4]], Xor[a^p_5, a^q_5]]],
And[And[And[Nor[a^p_2, a^q_2], Nand[a^p_3, a^q_3]],
Nand[a^p_4, a^q_4]], Xor[a^p_5, a^q_5]]]

Figure 2: Illustration of correspondence between the combinational procedure for ω_5 and the graph structure of its circuit. (Inputs are omitted from graph.) In this case, $a^p_1 = \text{True}$, $a^q_1 = \text{False}$, $a^p_2 = \text{True}$, $a^1_2 = \text{False}$, $a^p_3 = \text{False}$, $a^q_3 = \text{True}$, $a^p_4 = \text{False}$, $a^q_4 = \text{False}$, $a^p_5 = \text{True}$, and $a^q_5 = \text{True}$.

SciSci Research

サイサイ・リサーチ

Published by SciSci Press