5500FP

A 24-Trit Balanced Ternary RISC Processor

implemented on FPGA

with an Open Hardware Development Platform

Claudio Lorenzo La Rosa
Independent Researcher
Desio (MB), Italy
claudio.larosa74@tiscali.it

***Abstract*—We present the 5500FP, a balanced ternary RISC processor featuring a native 24-trit word width, implemented on FPGA as a standalone CPU module, with an accompanying open hardware development board. Unlike binary architectures that dominate modern computing, the 5500FP adopts balanced ternary representation ($\{-1, 0, +1\}$, denoted *N*, *Z*, *P*), offering natural sign handling and simplified negation at the arithmetic level. Following RISC principles, the architecture features a regular instruction format, a load/store memory model, and a compact ISA of 120 instructions organized across user and kernel privilege levels. Notably, the ISA includes native atomic instructions (compare-and-swap and fetch-and-add) providing hardware-level synchronization primitives. The processor ISA, internal organization, and key instruction mechanisms are protected by patent and copyright, while the underlying microarchitecture is left open for independent implementation, enabling the research community to develop their own 5500FP-compatible cores freely. The development board is released as open hardware. We describe the architecture, instruction set, instruction formats, development toolchain, and board peripherals, and discuss the design rationale behind key decisions including word width selection and the binary/ternary interface strategy.**

***Index Terms*—ternary computing, balanced ternary, RISC, FPGA, processor architecture, tryte, open hardware, development board, non-binary computing**

I. Introduction

Binary representation has dominated digital computing since the mid-20th century, largely due to the natural mapping of two voltage levels to transistor switching states. Yet the theoretical advantages of ternary and higher-radix number systems have been recognized since the earliest days of computer science [1].

Balanced ternary, in which digits take values in $\{-1, 0, +1\}$, is particularly elegant: it is a symmetric system that requires no separate sign bit, allows trivial negation by inverting all trits, and is the most efficient radix in terms of the radix economy metric [2]. The Soviet *Setun* computer, developed at Moscow State University in 1958–1965, remains the most notable historical implementation of a balanced ternary machine [3].

Despite these theoretical advantages, no balanced ternary processor has achieved widespread adoption, primarily due to the lack of a mature fabrication ecosystem for ternary logic gates. With the availability of modern FPGAs, however, it becomes feasible to implement and distribute working ternary hardware without custom silicon, opening the door to practical research and experimentation.

The 5500FP is designed as a RISC architecture, with a regular instruction format, orthogonal register file, and load/store memory model. The name reflects the processor family designation while the "FP" suffix denotes the FPGA implementation.

This paper makes the following contributions:

- A complete description of the 5500FP RISC architecture and its 24-trit word width rationale;
- A description of the nine instruction formats and the 120-instruction ISA;
- A discussion of native atomic synchronization instructions (CAS, FAA) as first-class ISA citizens;
- A description of the open hardware development board and its peripheral complement;
- A discussion of the binary/ternary interface strategy;
- A macro-assembler toolchain for Windows enabling practical software development (coming soon for Linux and MacOS).

II. Background and Related Work

A. Balanced Ternary Arithmetic

In balanced ternary, a n-trit integer N is expressed as:

$$N = \sum_{i=0}^{n-1} d_i \cdot 3^i, \quad d_i \in \{-1, 0, +1\}$$

Key properties relevant to this design include:

- **Symmetric range**: a 24-trit word represents integers in $\left[-\frac{3^{24}-1}{2}, +\frac{3^{24}-1}{2}\right]$, i.e., approximately ± 141 billion;
- **Trivial negation**: negating a value requires only inverting all trit signs, with no special handling for minimum values as required in two's complement;
- **Natural sign**: the most significant trit's sign directly indicates the sign of the number.

B. Prior Work

The Setun [3] established the feasibility of balanced ternary computing in the 1960s. More recently, several academic groups have proposed ternary logic using emerging technologies such as carbon nanotube FETs, memristors or transistors operating directly in three states. Notably, interest in ternary representation has re-emerged in the context of large language models: BitNet b1.58 [4], developed at Microsoft Research, demonstrates that LLM weights quantized to ternary values $\{-1, 0, +1\}$ can match full-precision model performance while achieving substantial reductions in memory, latency, throughput, and energy consumption. This result provides independent confirmation, from a very different domain, that the three-valued representation $\{-1, 0, +1\}$ — identical to balanced ternary — offers fundamental efficiency advantages over binary. To the author's knowledge, no prior work has released a complete, programmable balanced ternary processor development board targeting the research community.

III. Architecture

A. Word Width: Why 24 Trits?

The choice of 24 trits as the native word width is a central architectural decision that distinguishes the 5500FP from prior ternary processors. Two quantitative arguments support this choice over the power-of-3 alternative (27, 81, 243 trits): the intrinsic advantage of ternary over binary on native silicon, and the unsustainable complexity growth of the power-of-3 scaling series. Both are illustrated in Figures 1–3.

The ternary advantage on native silicon. On native ternary silicon, each physical signal line carries one trit ($\{-1, 0, +1\}$), representing three states rather than two. At the same physical wire count N, a ternary datapath represents 3^N distinct values versus 2^N for binary — a ratio of $(3/2)^N$ that grows exponentially with N. Equivalently, to represent the same numerical range, ternary requires consistently 35–38% fewer signal lines than binary (Figure 3). This translates directly into smaller die area, reduced routing complexity, lower power consumption, and better signal integrity. For example, a range of ± 141 billion requires 39 signal lines in binary but only 24 trits — a saving of 15 lines per datapath. This efficiency is the fundamental motivation for ternary silicon.

Why the ×3 scaling series negates this advantage. Prior ternary architectures scaled word widths as powers of 3: $27 \rightarrow 81 \rightarrow 243 \rightarrow 729$ trits. Each step triples the trit count, imposing additive jumps of +54, +162, and +486 trits per step (see Figure 2). A 243-trit datapath requires 243 physical signal lines — enormous routing complexity that imposes severe constraints on die area, pinout, signal-to-noise ratio, and timing closure. The very advantage of ternary — fewer lines for the same range — is progressively eroded when word width grows by factors of three. The ×3 series is therefore self-defeating as a scaling strategy for ternary silicon.

The ×2 series preserves the advantage. The doubling series ($24 \rightarrow 48 \rightarrow 96 \rightarrow 192$ trits) adds exactly the previous trit count at each step: +24, +48, +96 trits. These increments are moderate, predictable, and physically manageable. Critically, even at the base size of 24 trits, the representable range (± 141 billion) is already *enormously* superior to a standard 32-bit binary processor (± 2 billion) — a 66× wider range with only 50% more signal lines on native silicon (see Figure 1). At 48 trits the ternary range exceeds 3.9×10^{22} — more than 3,000 times wider than a 64-bit binary processor ($\pm 9.2 \times 10^{18}$). Equivalently, representing the same range in binary would require 77 signal lines, versus only 48 trits on native ternary silicon. The ×2 series thus maintains the ternary efficiency advantage at every scaling step while keeping physical complexity under control.

Binary memory compatibility (FPGA prototype only). On the current FPGA implementation, each trit is encoded as 2 binary bits, so a 24-trit word occupies exactly 48 binary bits, mapping directly onto standard 48-bit binary memory interfaces. This is a practical advantage specific to the current prototype; a future native ternary ASIC would interface directly to ternary memory, making binary compatibility a secondary concern.

Field alignment and zero fragmentation. The 5500FP instruction formats use 4-trit fields throughout. A 24-trit word divides exactly into 6 such fields with zero remainder. All sizes in the doubling series share this property. By contrast, 27 trits leave a 3-trit remainder and 81 trits leave a 1-trit remainder, forcing irregular field boundaries in the instruction decoder.

Tryte alignment and string efficiency. A 24-trit word contains exactly 4 trytes ($4 \times 6 = 24$). A single word can hold four 6-trit characters, making string operations naturally word-aligned and efficient.

Register file addressability. With a 4-trit register identifier ($3^4 = 81$), the 5500FP addresses up to 81 general-purpose registers — more than the 32 or 64 of typical binary RISC processors — reducing memory spill/fill frequency.

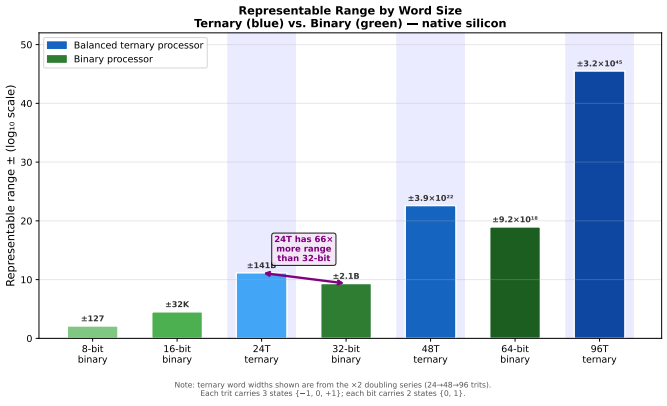

Fig. 1. Representable range ($\log_{10}$ scale) for selected balanced ternary word sizes from the doubling series (blue: 24T, 48T, 96T) compared to standard binary processors (green: 8, 16, 32, 64 bit). Already at 24 trits the representable range ($\pm 1.4 \times 10^{11}$, i.e. ± 141 billion) is 66× wider than a 32-bit binary processor ($\pm 2.1 \times 10^{9}$), despite requiring only 50% more signal lines on native silicon. At 48 trits the ternary range ($\pm 3.9 \times 10^{22}$) far exceeds that of a 64-bit processor ($\pm 9.2 \times 10^{18}$). The ternary advantage grows exponentially with word width.

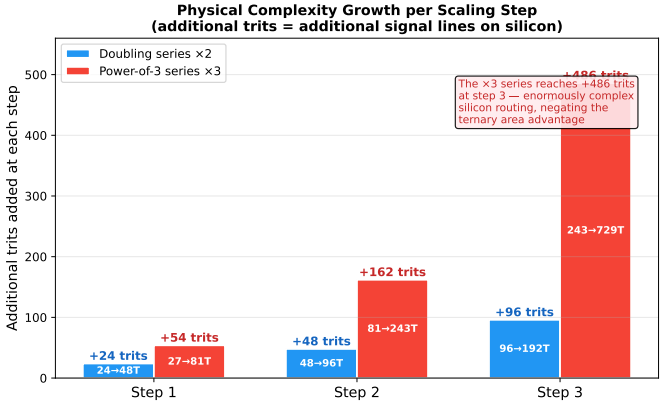

Fig. 2. Additional trits required at each scaling step. Each additional trit corresponds to one additional physical signal line on silicon. The doubling series (blue) grows linearly: +24, +48, +96 trits per step — moderate increments that keep routing complexity manageable. The power-of-3 series (red) grows exponentially: +54, +162, +486 trits per step. At step 3 a single scaling operation adds 486 signal lines, imposing severe constraints on die area, pinout, signal-to-noise ratio, and timing closure, thereby negating the fundamental area advantage of ternary silicon.

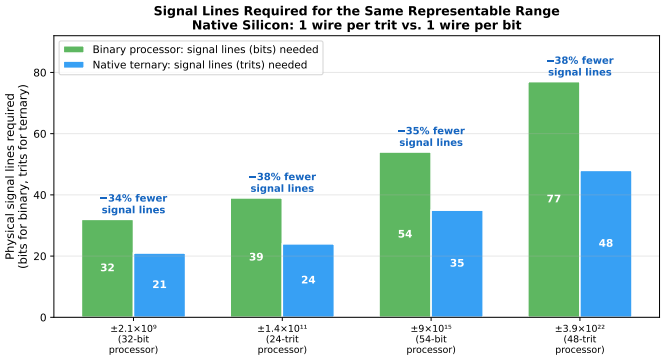

Fig. 3. Physical signal lines required to represent the same numerical range on native silicon: binary processors (one line per bit, green) vs. native ternary (one line per trit, blue). For every target range, ternary requires 34–38% fewer signal lines than binary. This reduction translates directly into smaller die area, simpler routing, and lower power consumption — the primary motivation for native ternary silicon implementations.

B. Trit Encoding on FPGA

Since FPGAs operate natively in binary, each trit is encoded using 2 bits. The 5500FP defines the following trit symbols and their physical voltage levels in the current implementation:

TABLE I
TRIT ENCODING, SYMBOLS, AND PHYSICAL VOLTAGE LEVELS

Ternary Value	Symbol	2-bit Encoding	Voltage
-1	N	`11`	$-3.3\,\mathrm{V}$
0	Z	`01`	$0\,\mathrm{V}$
$+1$	P	`00`	$+3.3\,\mathrm{V}$

The symmetric voltage assignment ($\pm 3.3\,\mathrm{V}$) reflects the symmetric nature of balanced ternary and simplifies analog driver design in future ASIC implementations. The 2-bit encoding is chosen such that the unused combination `10` acts as an invalid/trap state, enabling error detection in logic. Furthermore, the encoding was chosen so that it is possible to represent the individual trits with two binary bits both at the output of the FPGA and at the input, eliminating conversion circuits external or internal to the FPGA. This results in a 48-bit internal binary representation for each 24-trit word.

C. Data Widths: Trit, Tryte, Short, and Word

The 5500FP defines a hierarchy of native data widths based on the *tryte* as the fundamental unit, analogous to the byte in binary systems:

TABLE II
NATIVE DATA WIDTHS

Name	Trits	Ternary Range	ISA suffix
Trit	1	$\{-1, 0, +1\}$	
Tryte	6	$[-364,\ +364]$	`.T`
Short	12	$[-265720,\ +265720]$	`.S`
Word	24	$[\approx -141\mathrm{B},\ \approx +141\mathrm{B}]$	`.W` (default)

The choice of 6 trits per tryte makes the tryte the smallest addressable unit and ensures that the 24-trit word is exactly 4 trytes, providing a clean mapping between address arithmetic and data layout. Load and store instructions (LD, ST) operate on all three widths via the `.T`, `.S`, and `.W` suffixes respectively, with implicit sign extension for narrower loads.

D. Internal Organization

The 5500FP follows a conventional RISC organization — control unit, ALU, register file, and memory interface — in which all internal data paths, registers, and status words reflect a native ternary architecture. Externally, the processor presents a fully balanced ternary interface: all signals on the I/O buses are physical balanced ternary ($\pm 3.3\,\mathrm{V}$), making the 5500FP appear as a purely ternary device to the rest of the system.

Control unit. The control unit manages the fetch-decode-execute cycle. It contains a Program Counter (PC) holding the address of the next instruction, an Instruction Register (IR) holding the current instruction during decode and execution, and a Program Status Word (PSW) storing processor flags and condition codes.

Memory interface. A Memory Address Register (MAR) holds the address of the memory location to access, and a Memory Data Register (MDR) buffers data being read from or written to memory.

Register file. The 5500FP provides up to 81 general-purpose registers, each 24 trits wide, addressed by a 4-trit identifier ($3^4 = 81$). This count exceeds typical binary RISC processors (32–64 registers), reducing memory spill/fill frequency and improving compiler register allocation efficiency.

Dual stack pointers. The processor maintains two physically distinct stack pointer registers: one for user mode and one for system (kernel) mode. A mode register stores the current privilege level. All stack instructions (PUSH, POP) implicitly select the correct stack pointer based on the current mode — no explicit save/restore is required on mode transitions,

providing strong user/kernel stack isolation with zero ISA overhead.

Shift and rotate via signed operand. Shift and rotate instructions (ASH, ROT) encode both direction and magnitude in a single signed balanced ternary operand: a negative value shifts right, a positive value shifts left, and zero is a no-op. This eliminates the need for separate `SHIFT_LEFT`/`SHIFT_RIGHT` opcodes, simplifying the decoder and reducing opcode space usage.

Trit test and branch. Instructions TTT, TTU, and TTF test a single trit from a register and conditionally branch if the tested condition holds. The immediate variants TTTI, TTUI, and TTFI perform the same operation using an immediate operand for the branch offset. This combined test-and-branch mechanism eliminates the need for a separate branch instruction, reducing code size and resolving control hazards in pipelined implementations.

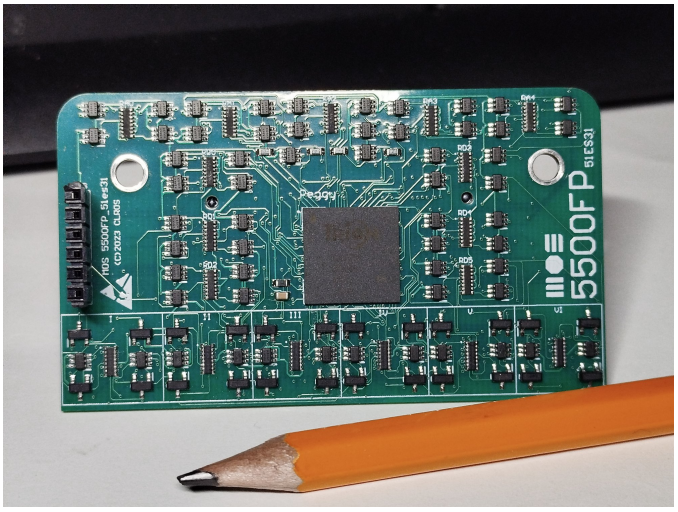

Fig. 4. 5500FP CPU module. A pencil is included for scale, showing the compact form factor of the processor board. The central FPGA package implements the 5500FP core; the edge connector allows the module to be plugged into a development or carrier board.

IV. Instruction Set Architecture

A. Design Philosophy

The 5500FP ISA v1.1 comprises about 120 instructions organized into two privilege levels: *User* mode (accessible to all programs) and *Kernel* mode (restricted to the operating system or supervisor). The architecture follows RISC principles: regular instruction formats, a load/store memory model, and orthogonal register usage. Opcodes are themselves expressed as 4-trit sequences, reflecting the native ternary representation throughout the entire architecture.

B. Instruction Formats

The ISA defines nine instruction formats (A through F, J, J2–J4), each composed of 24-trit fields of 4 trits each. The formats differ in the number of register operands and immediate field size, allowing the ISA to balance code density with expressive power. Table III summarizes all formats.

TABLE III
Instruction Formats (ISA 1.1). Each field is 4 trits wide. Rd = destination register; Rs1–Rs4 = source registers; Imm = immediate field.

Format	Fields (left to right)	Reg	Imm trits
A	OpCode, Rs4, Rs3, Rs2, Rs1, Rd	5	0
B	`[ext]`, OpCode, Rs3, Rs2, Rs1, Rd	4	0
C	`[ext]`×2, OpCode, Rs2, Rs1, Rd	3	0
D	`[ext]`×3, OpCode, Rs1, Rd	2	0
E	`[ext]`×4, OpCode, Rd	1	0
F	`[ext]`×5, OpCode	0	0
J	OpCode, Immediate_{20}	0	20
J2	OpCode, Immediate_{12}, Rs1, Rd	2	12
J3	OpCode, Immediate_{12}, Rs1, Imm_4	1	16
J4	OpCode, Immediate_{16}, Rd	1	16

Formats A–F are register-to-register formats, where the `[ext]` prefix fields act as format discriminators. Formats J–J4 encode immediate values, with J providing the largest 20-trit immediate for unconditional jumps, and J2 being the most common format for memory access and arithmetic immediate operations.

C. Instruction Groups

The ISA is organized into the following functional groups:

- **Math** (ADD, ADDI, ADDS, ADDSI, SUB, SUBI, SUBS, SUBSI, MUL, DIV, INC, INCS, DEC, DECS): standard arithmetic operations with both register and immediate variants. The "S" suffix denotes saturating arithmetic.
- **Ternary Native Functions** (ANY, ANYI, CLD, CLU, CONS, CONSI, DECOT, DECOU, DECOF, ENTI, EPTI, EQUAL, EQUALI, IMPL, MAX, MAXI, MIN, MINI, NTI, PTI, ROD, ROU, SHD, SHU, STI, SUM, SUMI, SWN, SWP, TXOR, TXORI): operations with no direct binary equivalent, exploiting the three-valued nature of the system.
- **Trit Test and Set** (MSKP, STF, STFI, STT, STTI, STU, STUI, TTF, TTFI, TTT, TTTI, TTU, TTUI): instructions to test and conditionally set individual trits or trit fields.
- **Rotate/Shift** (ASH, ASHI, ROT, ROTI): arithmetic shift and rotation operations.
- **Load/Store** (LD, LD.S, LD.T, LD.W, ST, ST.S, ST.T, ST.W, LEA): memory access across all native data widths — Word (W, 24 trits = 4 trytes), Short (S, 12 trits = 2 trytes), and Tryte (T = 6 Trit). The tryte (6 trits) is the fundamental addressable unit, analogous to the byte in binary systems.
- **Stack** (PUSH, PUSH.S, PUSH.T, PUSH.W, POP, POP.S, POP.T, POP.W, PUSHR, POPR, PUSHAG, POPAG, PUSHAGR, POPAGR): stack operations including bulk register save/restore.
- **Jump** (JMP, JSR, JR, RTI): unconditional control flow.
- **Conditional Jump** (JB, JBE, JBEI, JBI, JEQ, JEQI, JNE, JNEI): conditional branches supporting below, equal, and

not-equal conditions with both register and immediate variants.
- **I/O** (IN, IN.S, IN.T, IN.W, OUT, OUT.S, OUT.T, OUT.W): port-mapped I/O with three data widths (Kernel mode).
- **Atomic** (CAS, FAA, FAAI): hardware-level synchronization primitives. *Compare-and-swap* (CAS) atomically compares a memory location with a register value and conditionally updates it, while *fetch-and-add* (FAA/-FAAI) atomically increments a memory location and returns the previous value. These instructions enable lock-free data structures and OS synchronization primitives without software-level spin loops.
- **CPU Management** (CHST, CID, DI, EI, HLT, INT, LDITBR, LDSP, STITBR, STSP): processor control, interrupt management, stack pointer operations and others.

D. Ternary-Specific Instructions

A distinctive feature of the 5500FP ISA is the set of instructions that have no binary equivalent. These exploit the three-valued nature of balanced ternary directly:

MIN/MAX implement element-wise ternary minimum and maximum, equivalent to ternary AND and OR respectively. **CONS** (consensus) and **ANY** implement the standard ternary consensus and any functions. **EQUAL** returns the trit-wise equality. **TXOR** implements ternary exclusive-or. **SUM** computes the ternary sum function. **IMPL** implements ternary implication.

The **DECOT/DECOU/DECOF** instructions decode a trit to its constituent values (T, U, or F respectively), enabling efficient trit-level manipulation without bit masking. **SHD/SHU** shift trit fields down and up within a word. **SWP/SWN** swap positive and negative trit values.

TABLE IV
SELECTED TERNARY-NATIVE INSTRUCTIONS

Mnemonic	Opcode	Description
MIN	—-	Ternary minimum (AND)
MAX	—0	Ternary maximum (OR)
TXOR	—+	Ternary XOR
CONS	–00	Consensus function
ANY	–+-	Any function
EQUAL	–0+	Trit-wise equality
IMPL	-0+0	Ternary implication
SUM	–0-	Ternary sum
DECOT	-0-+	Decode trit T
DECOU	-00-	Decode trit U
SWP	–0-	Swap positive/negative

V. DEVELOPMENT BOARD

A. Overview

The 5500FP development board is released as open hardware under the **CERN Open Hardware Licence Permissive v2 (CERN OHL-P v2)**. This licence allows anyone to use, modify, and redistribute the design, including in commercial and closed-source products, without any obligation to release derivative works. The processor ISA, internal organization, and key instruction mechanisms are protected by patent and copyright; the underlying microarchitecture is left open for independent implementation. The board design, schematics, and bill of materials are freely available.

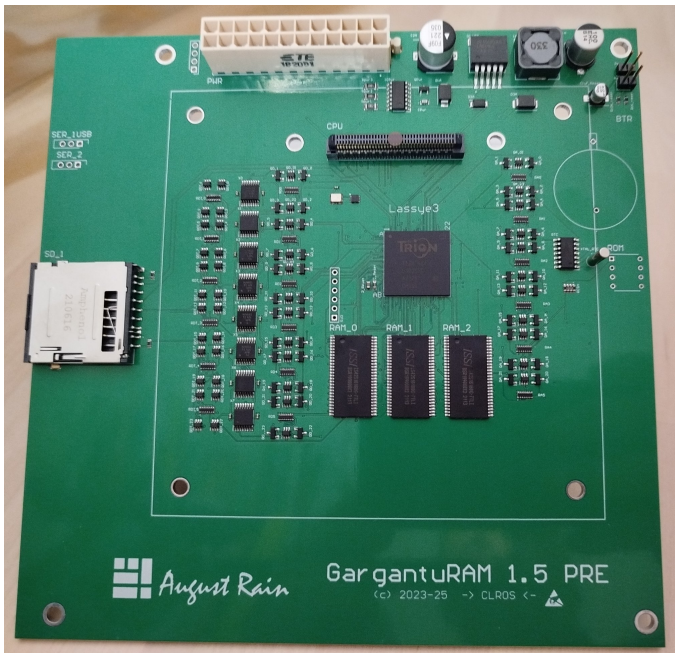

Fig. 5. 5500FP development board (GargantuRAM 1.5 PRE). Visible components include the CPU module connector (top center), the Efinix TRION FPGA, three ISS static RAM modules (RAM_0, RAM_1, RAM_2), SD card slot (SD_1), two serial interfaces (SER_1USB, SER_2), ROM header, and power section (top).

B. FPGA Target

The 5500FP is implemented on an **Efinix Trion T20F256** FPGA, clocked at **20 MHz**. The choice of such a conservative clock frequency is deliberate and driven by the external components responsible for handling the physical ternary signals ($\pm 3.3\,\mathrm{V}$). These components — which translate between the FPGA's binary logic levels and the balanced ternary voltage domain — are not designed to operate at high frequencies: pushing them faster would result in excessive heat dissipation and unreliable signal integrity. The 20 MHz clock is supplied to the CPU by a dedicated oscillator mounted on the GargantuRAM development board, ensuring a clean and stable reference independent of the FPGA's internal PLLs.

It should be noted that this frequency constraint is specific to the current FPGA-based prototype and its interface circuitry. A future ASIC implementation using native ternary logic gates would not be subject to this limitation.

C. Peripherals

The board provides the following interfaces:

- **Serial interfaces (×2, via USB)**: two independent UART channels exposed over USB, used for host communication;

- **SPI ROM**: non-volatile storage for boot code or firmware images;
- **SPI SD card**: removable mass storage for program and data files;
- **Static RAM**: 16M Word (64 MTryte) of static RAM for program and data memory.

D. Binary/Ternary Interface

A fundamental challenge in any ternary system built on binary FPGA fabric is interfacing cleanly with the surrounding binary ecosystem. The 5500FP addresses this at two levels. Internally, each trit is encoded as 2 binary bits (see Table I), allowing the FPGA to process ternary data natively without any conversion circuitry. Externally, dedicated analog interface components convert between the FPGA's binary logic levels and the physical balanced ternary voltage domain ($\pm 3.3\,\mathrm{V}$), so that all signals on the external buses are genuine balanced ternary. For communication with standard binary peripherals, the SPI bus is used: SPI is a simple serial protocol whose signals are directly compatible with the FPGA's binary I/O pins, avoiding the need for ternary-to-binary conversion on the peripheral side.

VI. Toolchain

A. Macro-Assembler

Software development for the 5500FP is supported by a macro-assembler running on Windows. The assembler accepts source files written in 5500FP assembly language, supporting include files, definition of constants, labels, and the full ISA v1.1 instruction set. Output is a binary image transferable to the board via the SD card.

B. Development Workflow

The assembler output provides a color-coded listing that maps each instruction mnemonic to its ternary encoding, with distinct colors highlighting the opcode, register fields, and immediate values within each instruction word. This visual representation aids debugging and helps programmers verify the correct encoding of complex instructions. Figure 6 shows an example listing.

```
jsr     print                   -> 124 (     4G)     00+0000000000000++--+--+
ashi    R15 R15 #4              -> 128 (     4N)     +--+0000000000++-0+--0+-
anyi    r8 r0 #-4               -> 132 (     4V)     +---0000000000-------++
txor    r8 r8 R15               -> 136 (     51)     ++++++-----0+---++--++
anyi    r9 r0 #-3               -> 140 (     55)     +---+0000000000-0----0--
jeq     r8 r9 version_5_1       -> 144 (     59)     ----0000000000++-0---++
jmp     exit_CPU                -> 148 (     5D)     000+000000000000000+--+
```

Fig. 6. 5500FP macro-assembler output. The left column shows assembly mnemonics with operands; the center column shows memory address and byte offset; the right column shows the full 24-trit instruction encoding in balanced ternary, with color-coding distinguishing opcode, register, and immediate fields.

Figure 7 shows an example session executing a test program for the Ternary Native Functions, displaying results in balanced ternary notation using the N/Z/P symbol set ($-/0/+$ respectively).

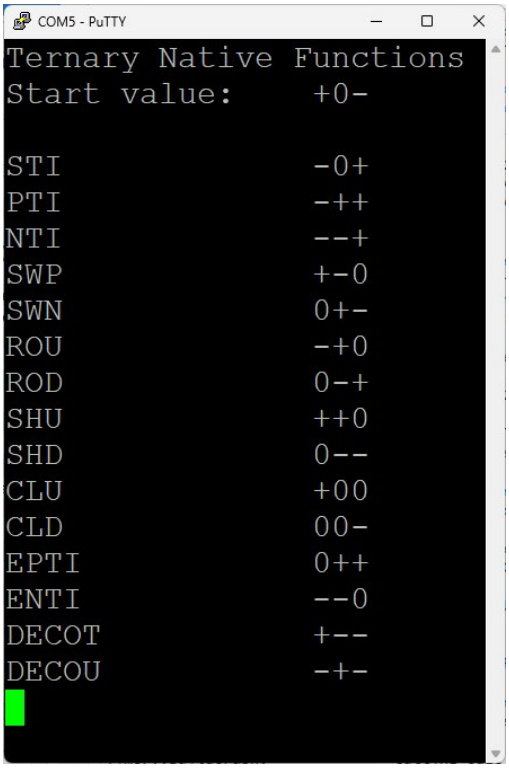

Fig. 7. PuTTY serial terminal showing a Ternary Native Functions test program running on the 5500FP board. Results are displayed in balanced ternary notation using $+/0/-$ for P/Z/N respectively. The green cursor confirms live execution on hardware.

TABLE V
Implementation Results (FPGA)

Parameter	Value
FPGA device	Efinix Trion T20F256
Clock frequency	20 MHz
LE utilization	18137
FF utilization	TBD
BRAM utilization	TBD
Power consumption	TBD

VII. Measurements

These figures reflect the overhead of emulating ternary logic on binary FPGA fabric. A future ASIC implementation is expected to yield substantially different characteristics.

VIII. Discussion

A. Advantages of Balanced Ternary in this Design

The 5500FP design has concretely benefited from balanced ternary in several areas. Negation is trivially implemented by inverting all trit signs, eliminating the asymmetric minimum-value edge case inherent in two's complement arithmetic. The native sign of the most significant trit simplifies comparison and branching logic. The shift and rotate instructions exploit the signed nature of balanced ternary operands to encode both direction and magnitude in a single field, reducing opcode

space and decoder complexity. The dual stack pointer mechanism benefits from the three-valued mode register, providing clean user/kernel isolation with no ISA overhead. Finally, the tryte-based data hierarchy allows four 6-trit characters to be packed into a single 24-trit word, improving string processing efficiency.

B. Limitations and Future Work

Clock frequency. The current implementation operates at 20 MHz, constrained by the external analog components that handle the physical ternary voltage domain (± 3.3 V). While this frequency does not allow the 5500FP to compete with commercial binary processors in raw performance, it has enabled the realization of a fully functional hardware prototype available immediately for real-hardware testing. Software and algorithms written today for the 5500FP ISA will run unmodified on all future implementations of this architecture, including faster ASIC-based ones, establishing a solid and immediately usable software foundation.

Toolchain maturity. The current toolchain consists of a macro-assembler for Windows. No compiler for a high-level language is yet available. An ongoing collaboration with a project contributor is addressing this gap through the design of a new memory-safe language inspired by Rust, targeting the 5500FP ISA natively. The choice of a memory-safe managed language — rather than C — is deliberate: such languages can provide strong process isolation without relying on a hardware MMU, which aligns well with the architectural characteristics of the 5500FP. A longer-term goal is to provide native development tools (assembler and high-level compiler) running directly on the 5500FP itself, eliminating the need for cross-compilation on binary hosts.

Operating system. A minimal operating system kernel is already available, providing basic hardware initialization, interrupt and exception handling, memory read/write, SD card access, and a simple interactive shell capable of loading and executing binaries from the SD card. While not yet a full OS, this kernel demonstrates that the system works correctly not only in its native ternary computational components but also in all foundational subsystems. The kernel is released under a permissive open-source licence and is available at https://github.com/MOS5500/GRam_OS.

Path to silicon. A primary long-term objective is the implementation of the 5500FP architecture on silicon. A native ternary ASIC would eliminate the analog interface overhead of the current FPGA prototype, enabling significantly higher clock frequencies and lower power consumption. A 24-trit ternary processor handles a data range equivalent to approximately 38 binary bits within a single word — substantially wider than a 32-bit binary processor — while potentially occupying less silicon area when implemented with native ternary logic technologies such as carbon nanotube FETs (CNTFETs) or memristor-based gates, which are natural candidates for three-state logic. Two parallel strategies are being pursued: licensing the 5500FP ISA and architecture to third parties wishing to develop their own compatible implementations (following a model similar to ARM's ISA licensing), and exploring partnerships with foundries capable of supporting a direct ternary tape-out.

IX. Conclusion

We have presented the 5500FP, a 24-trit balanced ternary processor implemented on FPGA and distributed as an accessible open hardware development platform. The design demonstrates the practical feasibility of balanced ternary computing on modern reconfigurable hardware and provides a concrete foundation for the research community to explore non-binary architectures without the barrier of custom silicon development.

The processor ISA, internal organization, and key instruction mechanisms are patent and copyright protected; the underlying microarchitecture is open for independent implementation. The development board is open hardware. All board design files and assembler toolchain are available at https://github.com/Ternary-Computer-System/GargantuRAM. This preprint is available on Zenodo at https://doi.org/10.5281/zenodo.18881738.

Acknowledgments

The author wishes to thank Cesare Di Mauro for his invaluable advice during the initial ISA design phase and for his continued support on architectural and design questions throughout the development of this project. Thanks are also due to Peter Reijnders for his enthusiastic support of the project and for his ongoing work on the design of a high-level memory-safe language targeting the 5500FP architecture. Finally, the author is deeply grateful to his wife Tania and his son Biagio for their constant encouragement and unwavering support throughout this work — and simply for being family.

References

[1] D. E. Knuth, *The Art of Computer Programming, Vol. 2: Seminumerical Algorithms*, 2nd ed. Addison-Wesley, 1981.

[2] B. Hayes, "Third base," *American Scientist*, vol. 89, no. 6, pp. 490–494, 2001.

[3] N. P. Brousentsov et al., "Experience in the development of the Setun computer," in *Small Computers and Large Tasks*, Moscow, 1970. [In Russian]

[4] S. Ma, H. Wang, L. Ma, L. Wang, W. Wang, S. Huang, L. Dong, R. Wang, J. Xue, and F. Wei, "The era of 1-bit LLMs: All large language models are in 1.58 bits," *arXiv preprint arXiv:2402.17764*, Feb. 2024.