Gigabit Ethernet 1000BASE-T Digital Transmitter Front End

Tofic Esses
UC Berkeley Master of Engineering
EECS
Berkeley, CA, USA
tesses@berkeley.edu

Albert Ho
UC Berkeley Master of Engineering
MSE
Berkeley, CA, USA
albert_ho@berkeley.edu

Abstract*—This paper presents the digital transmit front end for a 1000BASE-T Gigabit Ethernet physical-layer transmitter, which includes a side-stream Linear Feedback Shift Register (LFSR) scrambler, a convolutional encoder, Four Dimensional 5-Level Pulse Amplitude Modulation (4D-PAM5) Mapper, and a two-tap Finite Impulse Response (FIR) interface to thermometer-coded DAC drive signals. Other Ethernet coding designs are insufficient for gigabit operation over twisted-pair copper because the 1000BASE-T channel must tolerate inter-symbol interference, echo, and crosstalk. The key design implements the IEEE Clause 40 PCS transmit flow in SystemVerilog: a MASTER/SLAVE dependent 33-bit LFSR generates scrambling and sign-control bits, a three-state convolutional encoder forms a 9-bit coded word, and a lookup-table-based 4D-PAM5 mapper produces four signed quinary symbols per clock cycle. The RTL design was verified using directed block-level testbenches, an exhaustive mapper test sweeping over mapping mode, encoded word, sign bits, and sign-reversal, and a top-level integration testbench for smoke testing. Synthesis and place-and-route were also completed under an 8 ns clock constraint using a 180μm $\times$ 180μm floorplan. The final implementation achieved post-route setup worst-negative slack (WNS) = 0.072 ns, hold WNS = 0.091 ns, total power = 3.108 mW, standard-cell area = 18,497.407 μm^2, density = 69.958%, and DRC/LVS/DRV clean, demonstrating that the digital TX path meets the 125 MHz symbol-rate timing requirement with a clean physical sign-off design.

I. Introduction

The problem addressed in this project is the design and physical implementation of the digital transmit front end for a 1000BASE-T Gigabit Ethernet physical-layer transmitter. 1000BASE-T is the physical-layer standard that permits 1 Gb/s Ethernet communication over four pairs of Category-5 twisted-pair copper cables. In 1000BASE-T architecture, the copper cable is connected to the Ethernet Media Access Control (MAC) layer through the Physical Layer (PHY) device. The MAC structures raw data into ordered frames for transmission across physical data, while the PHY converts the raw bits of an Ethernet frame into physically realizable electrical or optical signals (voltages/light pulses). The IEEE standard divides the PHY into different sublayers, including the Physical Coding Sublayer (PCS), the Physical Medium Attachment (PMA), and the Medium Dependent Interface (MDI). The PCS performs digital coding/decoding, scrambling, and symbol mapping, whereas the PMA performs analog/mixed signal transmission and reception. The MDI is the physical cable interface.

The focus of this paper is on the Transmit Digital Coding Path of the 1000BASE-T PCS, which lies between the MAC and the PMA circuit. The job of the TX digital path is to accept an 8-bit transmit data word from the Gigabit Media Independent Interface, or GMII, and transform it into a physically realizable 4D-PAM5 vector. The output of the PCS transmit path is a four-symbol vector $(A_n, B_n, C_n, D_n$, with each component is a quinary symbol, which means that it is selected from the voltage set $\{-2, -1, 0, +1, +2\}$.

Basic transmitters that directly map bytes to voltage levels are insufficient for 1000BASE-T operation. The copper channels are bandwidth-limited and noisy, and are affected with issues such as inter-symbol interference. Inter-symbol interference (ISI) refers to when the physical channel spreads a transmitted pulse in time, which causes residual energy from one symbol to overlap and contaminate the next symbol. Furthermore, 1000BASE-T architecture employs full-duplex signaling, which means that each PHY simultaneously transmits and receives data on the same four wire pairs. This means that echo-cancellers, signal-processing blocks that remove self-interference from the received signal, are required in the overall system.

Pulse-amplitude modulation encodes information through the amplitude level of a transmitted pulse. In PAM5, each wire pair carries one of five possible symbol levels. This scheme has a greatly reduced noise margin per level compared to binary signaling $(V_{\text{low}}, V_{\text{high}})$, as decision thresholds are now closer together. 1 Gb/s at a 125 MBd symbol rate requires higher symbol density, but coding and scrambling robustness are now required. The transmitter needs to minimize data-dependent jitter, where transmitted signals are extremely repetitive, crosstalk, or pair correlation, and DC bias, where the long-term average transmitted signal level is nonzero, which can distort the waveform and cause error in receiver operation.

Although the receiver coding path is not implemented in this paper, it still influences the transmit coding choices. In a full 1000BASE-T PHY, the PMA Receive functions usually requires echo/crosstalk cancellation to achieve required error performance.

II. Design Space Exploration

To find a suitable specification for the 1000BASE-T IEEE 802.3, official documentation and specifications were consulted. Consensus was then reached among the different subgroups to implement our designs around this specification. The TX Digital submodule was implemented in Verilog/SystemVerilog with the exact logic from IEEE Std 802.3-2022, IEEE Standard for Ethernet clause 40. In this way, no data corruption or control logic mismatch would occur in the TX/RX datapath of the Gigabit Ethernet standard. Fig. 1 highlights the TX Digital submodule in the Physical Coding Sublayer (PCS) that this report focuses on.

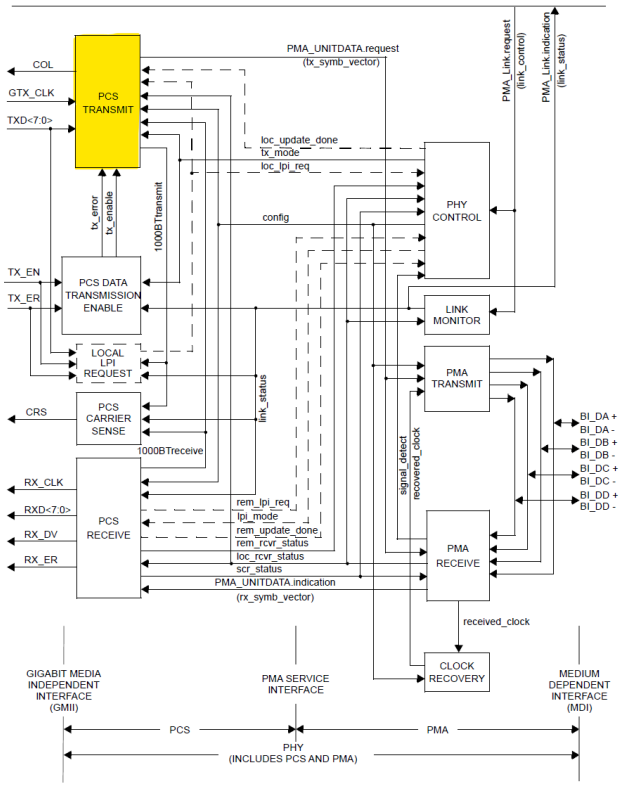

Fig. 1. Physical Coding Sublayer (PCS) with highlighted TX Digital

III. Summary of Specs Derived

The specifications for the digital transmit front end design were derived from the 1000BASE-T physical-layer definition in IEEE 802.3 Clause 40 and as established in the progress report.

For this design, the most important consequence is that the digital transmit datapath must produce one complete four-symbol vector every 8 ns. The implementation-level clock target is:

$$T_{\text{clk}} = 8 \text{ ns}$$
$$f_{\text{clk}} = 125 \text{ MHz} \tag{1}$$

The 8 ns clock period is the primary performance specification for the synthesized and placed-and-routed digital block. The specs for this project are found in the IEEE 802.3ab Standards, and can be summarized as follows [1]:

1) Standard: IEEE 802.3ab (1000BASE-T)
2) Data Rate: 1000 Mbps
3) Symbol Rate: 125 MBd
4) Wiring: 4 pairs of twisted copper cable (Category 5)
5) Input Width: 8 bits
6) Maximum Link Length: 100m

IV. Detailed Design

Please see Fig. 2 for the detailed TX Digital diagram that includes the Scrambler, Encoder, 4D-PAM5 Mapper, and FIR filters.

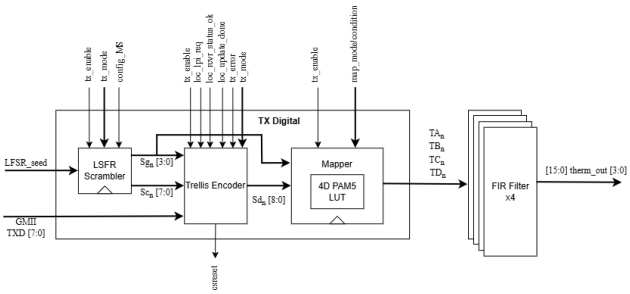

Fig. 2. TX Digital Detailed Diagram

A. Scrambler

The scrambler is the first logical block, and has the goal of ”randomizing” the transmitted symbol stream. Ethernet data can contain long repetitive patterns or long deterministic patterns, which can cause issues such as spectral peaks, or increase bit errors through inter-symbol interference.

The scrambler is implemented as a 33-bit Linear Feedback Shift Register side-stream generator. As a side-stream or additive scrambler, the module does not take in the transmit data byte, but instead generates a side-stream pseudo-randomized vectors Sc_n[7:0] and Sg_n[3:0]. The next logical block, the Trellis Encoder, performs the bitwise XOR operation on Sc_n[7:0] and TXD[7:0], and the 4D-PAM5 Mapper block uses Sg_n[3:0] to randomize the signs of the four PAM5 symbols.

The internal shift register state is stored as Scr_n[32:0]. During reset operation, the RTL loads LFSR_seed. For the corner case of an all-zero seed, LFSR_seed is set to 33’h1 to avoid a lock-up state, where the LFSR output remains 0 indefinitely.

The scrambler RTL module is parameterized by the output width and LFSR depth:

$$\begin{aligned} \text{LFSR\_DATA\_WIDTH} &= 8 \\ \text{LFSR\_DEPTH} &= 33 \end{aligned} \tag{2}$$

The signal config_MS controls whether the PHY operates as either a MASTER PHY or SLAVE PHY. In the full standard implementation, that relationship is determined during Auto-Negotiation, but for this paper, the scrambler is supplied

that role through config_MS. The correct tap polynomial is determined through this signal.

The central state element is the 33-bit register:

$$Scr_n[32:0] \tag{3}$$

where n is the symbol-period time index, marking the time during which one 4D-PAM5 vector is generated. At each clock edge, the LFSR shifts by one position, computing a new bit $Scr_n[0]$. The remaining bits shift as:

$$Scr_n[k] = Scr_{n-1}[k-1], k = 1, 2, ..., 32 \tag{4}$$

The new feedback bit is determined whether the PHY is configured as MASTER or SLAVE. For MASTER mode, the tap polynomial is defined as:

$$g_M = 1 + x^{13} + x^{33} \tag{5}$$

For SLAVE mode, the standard polynomial is:

$$g_S = 1 + x^{20} + x^{33} \tag{6}$$

In the zero-indexed state vector implemented in RTL, the feedback recurrences are defined as:

$$\begin{aligned} Scr_n[0] &= Scr_{n-1}[12] \oplus Scr_{n-1}[32] \text{ (MASTER MODE)} \\ Scr_n[0] &= Scr_{n-1}[19] \oplus Scr_{n-1}[32] \text{ (SLAVE MODE)} \end{aligned} \tag{7}$$

Fig. 3 shows a diagram of the LFSR for the Master and Slave mode.

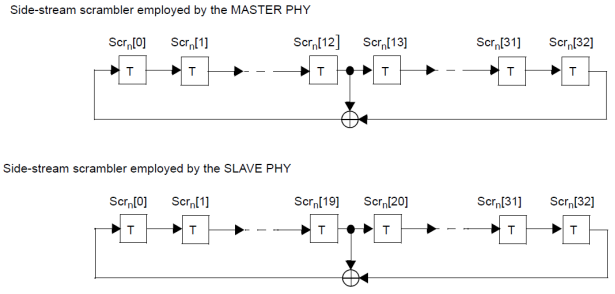

Fig. 3. Side-Stream Scramblers with LFSR (Figure 40-6)

Note that the SystemVerilog state vector is zero-indexed, so the first delay element is represented by bit 0, so the 13th, 20th, and 33rd delayed terms correspond to indexes 12, 19, and 32 respectively.

The register shifts on the rising clock edge. In synthesis, this structure becomes 33 flip-flops, one feedback XOR gate, one small multiplexer for the feedback tap choice, and shift-register wiring.

A critical requirement for LFSR-based sequence generation is to avoid all-zero initialization. If a LFSR enters the all-zero state, every feedback XOR output would be zero, so the LFSR remains stuck at zero, and thus the scrambler would fail to produce a pseudo-random sequence. To prevent this, RTL is implemented to explicitly check the reset seed:

$$Scr_n = \begin{cases} 33'h1, & \text{LFSR\_seed} = 0 \\ \text{LFSR\_seed}, & \text{else} \end{cases} \tag{8}$$

If the top-level system gives an all-zero seed, the scrambler will instead load $33'\text{h}1$.

For reset behavior, the module clears internal registers as follows:

$$\begin{aligned} tx\_en\_reg &= 2'b00 \\ Sy\_n - 1 &= 4'b0000 \\ parity &= 0 \end{aligned} \tag{9}$$

When the LFSR state is available, the scrambler calculates two auxiliary bits from LFSR tap positions:

$$\begin{aligned} X_n &= Scr_n[4] \oplus Scr_n[6] \\ Y_n &= Scr_n[1] \oplus Scr_n[5] \end{aligned} \tag{10}$$

X_n and Y_n are used to create delayed versions of the same maximum-length pseudo-random LFSR sequence using different tap values. As per the IEEE standard, X_n, Y_n, and $Scr_n[0]$ need to be mutually uncorrelated and are thus used along with the auxiliary generating polynomial

$$g(x) = x^3 \oplus x^8 \tag{11}$$

to generate the downstream bit sequence, broken up into 3 parts: $Sy_n[3:0], Sx_n[3:0]$, and $Sg_n[3:0]$.

1) Generation of $Sy_n[3:0]$:

$$\begin{aligned} \text{Sy}_n[0] &= \text{Scr}_n[0], \\ \text{Sy}_n[1] &= \text{Scr}_n[3] \oplus \text{Scr}_n[8], \\ \text{Sy}_n[2] &= \text{Scr}_n[6] \oplus \text{Scr}_n[16], \\ \text{Sy}_n[3] &= \text{Scr}_n[9] \oplus \text{Scr}_n[14] \oplus \text{Scr}_n[19] \oplus \text{Scr}_n[24]. \end{aligned}$$

The RTL also stores the previous value of Sy_n, which is Sy_{n-1}. The standard defines part of the scrambler octet $Sc_n[3:1]$ using the previous value depending on the parity of the current symbol period.

2) Generation of $Sx_n[3:0]$:

$$\begin{aligned} \text{Sx}_n[0] &= \text{Scr}_n[4] \oplus \text{Scr}_n[6], \\ \text{Sx}_n[1] &= \text{Scr}_n[7] \oplus \text{Scr}_n[9] \oplus \text{Scr}_n[12] \oplus \text{Scr}_n[14], \\ \text{Sx}_n[2] &= \text{Scr}_n[10] \oplus \text{Scr}_n[12] \oplus \text{Scr}_n[20] \oplus \text{Scr}_n[22], \\ \text{Sx}_n[3] &= \text{Scr}_n[13] \oplus \text{Scr}_n[15] \oplus \text{Scr}_n[18] \oplus \text{Scr}_n[20] \\ &\quad \oplus \text{Scr}_n[23] \oplus \text{Scr}_n[25] \oplus \text{Scr}_n[28] \oplus \text{Scr}_n[30]. \end{aligned}$$

3) Generation of Scrambler Octet $Sc_n[7:0]$: The scrambler octet $Sc_n[7:0]$ is used downstream by the encoder to scramble the input GMII data. It is formed by combining $Sx_n[3:0]$ and $Sy_n[3:0]$ as follows:

$$Sc_n[7:4] = \begin{cases} Sx_n[3:0], & \text{tx\_enable}_{n-2} = 1 \\ 0000, & \text{tx\_enable}_{n-2} = 0 \end{cases} \tag{12}$$

The lower bits depend on the parity of the current symbol period, which is cleared on reset and toggled every clock cycle.

$$\text{Sc}_n[3:1] = \begin{cases} 000, & \text{tx\_mode} = \text{SEND\_Z} \\ \text{Sy}_n[3:1], & \text{even} \\ \text{Sy}_{n-1}[3:1] \oplus 3'b111, & \text{odd} \end{cases} \tag{13}$$

4) Generation of $Sg_n[3:0]$:

$$\begin{aligned}
\mathrm{Sg}_n[0] &= \mathrm{Scr}_n[1] \oplus \mathrm{Scr}_n[5], \\
\mathrm{Sg}_n[1] &= \mathrm{Scr}_n[4] \oplus \mathrm{Scr}_n[8] \oplus \mathrm{Scr}_n[9] \oplus \mathrm{Scr}_n[13], \\
\mathrm{Sg}_n[2] &= \mathrm{Scr}_n[7] \oplus \mathrm{Scr}_n[11] \oplus \mathrm{Scr}_n[17] \oplus \mathrm{Scr}_n[21], \\
\mathrm{Sg}_n[3] &= \mathrm{Scr}_n[10] \oplus \mathrm{Scr}_n[14] \oplus \mathrm{Scr}_n[15] \oplus \mathrm{Scr}_n[19] \\
&\quad \oplus \mathrm{Scr}_n[20] \oplus \mathrm{Scr}_n[24] \oplus \mathrm{Scr}_n[25] \oplus \mathrm{Scr}_n[29].
\end{aligned}$$

Sg_n is not used by the encoder to form the scrambled data word, unlike Sx_n and Sy_n, but are rather passed to the PAM5 Mapper. The mapper combines Sg_n with a sign-reversal signal to determine the sign of each of the 4 PAM5 symbols. This is required so that each symbol stream has no DC bias.

The final scrambler design implements a side-stream scrambler using a 33-bit LFSR with selectable MASTER and SLAVE feedback polynomials. It generates 3 pseudo-random bit groups $Sx_n[3:0], Sy_n3:0, andSg_n[3:0]$, the first two are used to build the scrambler octet $Sc_n[7:0]$, and the latter $Sg_n[3:0]$ is passed to the mapper for sign-randomization.

B. Trellis Encoder

The second block in the digital transmit coding path is the Trellis Encoder. It receives the scrambler octet $Sc_n[7:0]$ from the upstream scrambler module and the transmit data byte TXD[7:0], to produce the 9-bit encoded word $Sd_n[8:0]$ for the 4D-PAM5 mapper.

The encoder is needed because 1000BASE-T does not directly map 8-bit transmit data into four PAM5 symbols. As described by the IEEE standard, 8 data bits are converted into one transmission of four quinary symbols, and the PCS uses a convolutional encoder during this conversion process.

A convolutional encoder is an encoder whose output depends on both the present input and stored information from previous symbol periods, known as the encoder state. A traditional stateless block encoder's output only requires the current input word. By using a convolutional, or "sliding-window" encoder, the current encoded word depends on current data/control inputs, as well as a 3-bit state variable defined as $cs_n[2:0]$.

A trellis diagram is a graph that represents all possible state transitions. Each column of the trellis corresponds to one symbol period, and each node corresponds to a possible encoder state. Branches connecting nodes represent possible transitions from a previous to next state. In the implemented design, each data-mode symbol period updates the $cs_n[2:0]$ state based on the previous state $cs_{n-1}[2:0]$ and the high two encoded data bits $Sd_n[7:6]$.

The standard defines the first state recurrence as:

$$cs_n[0] = cs_{n-1}[2]$$

and the 9th encoded bit as:

$$Sd_n[8] = cs_n[0]$$

The MSB of the encoder output represents the trellis state, which also carries information from the previous encoder state.

In data transmission mode, the other two state bits are defined through bitwise XOR as:

$$\begin{aligned}
cs_n[1] &= Sd_n[6] \oplus cs_{n-1}[0] \\
cs_n[2] &= Sd_n[7] \oplus cs_{n-1}[1]
\end{aligned}$$

The next state depends on both the previous state and the high two encoded data bits.

The lower six bits $Sd_n[5:0]$ do not affect the convolutional state, but are simply passed to the mapper. Outside of data mode, these bits encode different control/status signals. These are formed as follows:

$$Sd_n[5:0] = Sc_n[5:0] \oplus TXD[5:0] \tag{14}$$

1) Generation of $Sd_n[7:6]$: These two bits interact directly with the convolutional state, and are defined as follows, depending on the current mode, which is determined by a variety of externally supplied control signals.

$$\mathrm{Sd}_n[7:6] = \begin{cases} \mathrm{Sc}_n[7:6] \oplus \mathrm{TXD}[7:6], & \text{Data Transmission} \\ \mathrm{cs}_{n-1}[1:0], & \text{State Reset} \\ \mathrm{Sc}_n[7:6], & \text{Idle/Control} \end{cases}$$

2) Generation of $Sd_n[5:4]$:

$$\mathrm{Sd}_n[5:4] = \begin{cases} \mathrm{Sc}_n[5:4] \oplus \mathrm{TXD}[5:4], & \text{Data Transmission} \\ \mathrm{Sc}_n[5:4], & \text{Idle/Control} \end{cases}$$

3) Generation of $Sd_n[3]$: This bit also determines if the status is currently being encoded.

$$\mathrm{Sd}_n[3] = \begin{cases} \mathrm{Sc}_n[3] \oplus \mathrm{TXD}[3], & \text{Data Transmission} \\ \mathrm{Sc}_n[3] \oplus 1, & \text{Status Encoding} \\ \mathrm{Sc}_n[3], & \text{Idle/Control} \end{cases}$$

4) Generation of $Sd_n[2]$: This bit has the purpose of determining if the local receiver is correctly functioning.

$$\mathrm{Sd}_n[2] = \begin{cases} \mathrm{Sc}_n[2] \oplus \mathrm{TXD}[2], & \text{Data Transmission} \\ \mathrm{Sc}_n[2] \oplus 1'b1, & \text{Local Receiver Status OK} \\ \mathrm{Sc}_n[2], & \text{Idle/Control} \end{cases}$$

5) Generation of $Sd_n[1]$: This bit checks if the local PHY has updated the receiver state as stated by the standard. $\mathrm{cext}_{\mathrm{err}}$ refers to carrier extension, which is a control mechanism used after the end of a frame in certain Ethernet transmission cases.

$$\mathrm{Sd}_n[1] = \begin{cases} \mathrm{Sc}_n[1] \oplus \mathrm{TXD}[1], & \text{Data Transmission} \\ \mathrm{Sc}_n[1] \oplus 1'b1, & \text{Receiver State Updated} \\ \mathrm{Sc}_n[1] \oplus \mathrm{cext}_{\mathrm{err}}, & \text{Other} \end{cases}$$

6) Generation of $Sd_n[0]$: This bit checks for the carrier-extension control bit, in contrast to $Sd_n[1]$ which checked for the carrier-extension-error control bit.

$$\mathrm{Sd}_n[0] = \begin{cases} \mathrm{Sc}_n[0] \oplus \mathrm{TXD}[0], & \text{Data Transmission} \\ \mathrm{Sc}_n[0] \oplus \mathrm{cext}, & \text{Other} \end{cases}$$

The overall encoder design was developed by translating the IEEE standard equations for the 9-bit encoded word $Sd_n[8:0]$ and 3-bit state $cs_n[2:0]$ into SystemVerilog.

First, the standard equations were split into either datapath or control/status equations. The datapath equations described how scrambled data bits are produced during data transmission, through bitwise XOR operation. The control/status equations took in externally driven signals (idle, receiver status, carrier extension, etc.) to modify the lower bits when the encoder was not in data transmission mode.

Sequential state-update logic was also implemented to control the 3-bit convolutional state and the two-cycle transmit-enable delay, another constraint that comes from the overall Ethernet standard. Next-state logic was implemented using the register $cs\_next[2:0]$. The output word $Sd_n[8:0]$ was implemented as combinational logic, with each bit group defined above.

The result of the module is a synthesizable encoder that produces the standard-structured mapper input required by the downstream 4D-PAM5 mapper block.

C. 4D-PAM5 Mapper

PAM stands for pulse-amplitude modulation. In this system, information is encoded in the amplitude level of a transmitted pulse. For a 1000BASE-T PAM5 scheme, there are five levels, so each transmitted symbol is selected from 5 possible amplitude levels, which are:

$$\{-2, -1, 0, +1, +2\}$$

This is a quinary symbol, and these are represented as signed 3-bit values in the implemented mapper before being sent downstream to the FIR/DAC-interface block. 1000BASE-T transmits one PAM5 symbol on each of the four twisted pairs during a single symbol period, so this is a 4D (4-Dimensional) PAM5 scheme. A symbol vector is as follows:

$$(A_n, B_n, C_n, D_n)$$

Each of A, B, C, D is a quinary symbol for the twisted pair of the same name.

A look-up table, such as those in Fig. 4 and Fig. 5, is used because each RTL row can be compared directly against the standard table, and is both synthesizable as combinational logic and can be exhaustively tested over bounded input spaces.

		$Sd_n[6:8]$ = [000]	$Sd_n[6:8]$ = [010]	$Sd_n[6:8]$ = [100]	$Sd_n[6:8]$ = [110]
Condition	**$Sd_n[5:0]$**	**TA_n,TB_n,TC_n,TD_n**	**TA_n,TB_n,TC_n,TD_n**	**TA_n,TB_n,TC_n,TD_n**	**TA_n,TB_n,TC_n,TD_n**
Normal	000000	0, 0, 0, 0	0, 0,+1,+1	0,+1,+1, 0	0,+1, 0,+1
Normal	000001	–2, 0, 0, 0	–2, 0,+1,+1	–2,+1,+1, 0	–2,+1, 0,+1
Normal	000010	0,–2, 0, 0	0,–2,+1,+1	0,–1,+1, 0	0,–1, 0,+1
Normal	000011	–2,–2, 0, 0	–2,–2,+1,+1	–2,–1,+1, 0	–2,–1, 0,+1
Normal	000100	0, 0,–2, 0	0, 0,–1,+1	0,+1,–1, 0	0,+1,–2,+1
Normal	000101	–2, 0,–2, 0	–2, 0,–1,+1	–2,+1,–1, 0	–2,+1,–2,+1
Normal	000110	0,–2,–2, 0	0,–2,–1,+1	0,–1,–1, 0	0,–1,–2,+1
Normal	000111	–2,–2,–2, 0	–2,–2,–1,+1	–2,–1,–1, 0	–2,–1,–2,+1
Normal	001000	0, 0, 0,–2	0, 0,+1,–1	0,+1,+1,–2	0,+1, 0,–1

Fig. 4. Bit-to-symbol mapping even subsets (Table 40-1)

		$Sd_n[6:8]$ = [001]	$Sd_n[6:8]$ = [011]	$Sd_n[6:8]$ = [101]	$Sd_n[6:8]$ = [111]
Condition	**$Sd_n[5:0]$**	**TA_n,TB_n,TC_n,TD_n**	**TA_n,TB_n,TC_n,TD_n**	**TA_n,TB_n,TC_n,TD_n**	**TA_n,TB_n,TC_n,TD_n**
Normal	000000	0, 0, 0,+1	0, 0,+1, 0	0,+1,+1,+1	0,+1, 0, 0
Normal	000001	–2, 0, 0,+1	–2, 0,+1, 0	–2,+1,+1,+1	–2,+1, 0, 0
Normal	000010	0,–2, 0,+1	0,–2,+1, 0	0,–1,+1,+1	0,–1, 0, 0
Normal	000011	–2,–2, 0,+1	–2,–2,+1, 0	–2,–1,+1,+1	–2,–1, 0, 0
Normal	000100	0, 0,–2,+1	0, 0,–1, 0	0,+1,–1,+1	0,+1,–2, 0
Normal	000101	–2, 0,–2,+1	–2, 0,–1, 0	–2,+1,–1,+1	–2,+1,–2, 0
Normal	000110	0,–2,–2,+1	0,–2,–1, 0	0,–1,–1,+1	0,–1,–2, 0
Normal	000111	–2,–2,–2,+1	–2,–2,–1, 0	–2,–1,–1,+1	–2,–1,–2, 0
Normal	001000	0, 0, 0,–1	0, 0,+1,–2	0,+1,+1,–1	0,+1, 0,–2

Fig. 5. Bit-to-symbol mapping odd subsets (Table 40-2)

The 4D-PAM5 Mapper is the final stage in the TX digital path and is responsible for converting the 9-bit Trellis Encoder output word into a four-symbol quinary vector (A_n, B_n, C_n, D_n) where each symbol is driven on a twisted pair. It is implemented to ensure compliance with the IEEE 802.3 requirements for Clause 40 table-based bit-to-symbol mapping and sign-randomization.

There are multiple transmission modes which are selected by the map_mode word. These ensure transmission in the following states, with the required contents, such as: normal, idle, start and end delimiters, carrier extensions, error symbols, and control symbols. The mapper includes odd and even subset Look-up-Tables (LUTs) which enforce proper constellation partitioning, maintain valid state transitions, and maximize Euclidean distance between symbol paths, which directly improves receiver noise immunity and forward error correction performance.

The RTL is implemented using two modules, where mapper.v is the top module, and PAM5_LUT.v is the child module. PAM5_LUT.v contains only the LUTs. It takes in two inputs: map_mode and Sd_n, and produces one output quartet_packed, which is the result after doing the lookup. Mapper wraps PAM5_LUT.v but includes additional inputs, such as Sg_n and tx_enable, and produces the final quinary_symbols_packed which is then passed to the two-tap FIR filters. This wrapper includes additional control logic, and sign inversion of the quartet.

D. Peripherals

In the implemented transmitter, the downstream FIR/DAC interface is after the 4D-PAM5 mapper. The overall transmit datapath generates a four-symbol vector (A_n, B_n, C_n, D_n) once per symbol period.

Each FIR receives one lane of the mapper output, which each produces a separate thermometer-coded output bus:

$$A_n \rightarrow \text{FIR A} \rightarrow \text{therm\_A}[15:0]$$
$$B_n \rightarrow \text{FIR B} \rightarrow \text{therm\_B}[15:0]$$
$$C_n \rightarrow \text{FIR C} \rightarrow \text{therm\_C}[15:0]$$
$$D_n \rightarrow \text{FIR D} \rightarrow \text{therm\_D}[15:0]$$

This connects the digital 4D-PAM5 symbol generator with downstream analog transmitter/DAC blocks.

V. Verification Coverage

Verification was completed using functional simulations of the individual transmit-path blocks, as well as a top-level integration smoke testbench. Behavioral verification from test cases was completed for the design. A summary table for the verification style and the testbenches is provided:

TABLE I
VERIFICATION TESTBENCH SUMMARY

Testbench	DUT	Style	Checks
scrambler_tb.v	scrambler	Directed	7
encoder_tb.v	encoder	Directed	13
mapper_tb.v	mapper	Exhaustive	245,760
tx_digital_tb.v	tx_digital	Integration	11

A. Scrambler

The scrambler testbench verifies the scrambler module using directed tests.

1) Asserts LFSR_seed = 33'd0, then checks that Sc_n[0] == 1'b1 immediately after reset. Because Sc_n[0] = Sy_n[0] = Scr_n[0], this test ensures that zero-seed lock-up state doesn't occur.
2) Drives tx_mode = SEND_Z and then checks that Sc_n[3:0] == 4'b0000.
3) Holds tx_enable = 0, ensures that the delayed signal tx_enable_n-2 = 0, then checks that Sc_n[7:4] == 4'b0000.
4) Drives tx_enable = 1, and then checks that Sc_n[7:4] is still zero after a clock cycle, verifying that there is a two-cycle delay before the upper bits update.
5) Runs the scrambler for 50 clock cycles with the same seed in both MASTER and SLAVE modes, and then checks that the resulting Sc_n values are different.
6) Runs the scrambler under MASTER mode for 100 cycles, and then checks if at least 30 changes occur in Sc_n.
7) The scrambler is run for 50 cycles, and then checks if at least 10 changes occur in Sg_n.

B. Trellis Encoder

The trellis encoder testbench verifies the module using directed tests.

1) Drives Sc_n = 8'b10101010 and status/control inputs as low after reset to check if Sd_n = 9'b010101010.
2) Drives tx_enable high, then drops the current tx_enable low to check that csreset = 1'b1.
3) Asserts loc_lpi_req = 1 and then checks that the carrier-extension error does not flip the value of Sd_n[1].
4) When in idle mode, drives TXD != 8'h0F, tx_error = 1, loc_lpi_req = 0, and then checks that Sd_n[1] is flipped through the cext_err path.
5) When in idle mode, drives TXD = 8'h0F, tx_error = 1, and then checks that Sd_n[0] is flipped through the cext path.
6) When in idle mode, sets tx_mode = SEND_N, drives loc_lpi_req = 1 and then checks that Sd_n[3] and Sc_n[3] are different.
7) When in idle mode, drives loc_rcvr_status_ok = 1 and then checks that Sd_n[2] and Sc_n[2] are different.
8) When in idle mode, drives loc_update_done = 1 and then checks that Sd_n[1] and Sc_n[1] are different.
9) When in data mode, drives TXD = 8'b11001100 and Sc_n = 8'b10101010, and then checks that Sd_n[7:0] = 8'b01100110, making sure it is the same as the output of Sc_n $\oplus$ TXD, ensuring data mode operation.
10) When in data mode, checks that Sd_n[8] == 0 when a high-bit pattern is input.
11) When in data mode, when the same high-bit pattern is input, Sd_n[8] is checked to be == 1.
12) Checks that Sd_n[8] has returned to zero when no longer in data mode.
13) Checks that Sd_n[3] does not change when tx_mode = SEND_Z.

The testbench verifies reset, idle, data-mode behavior, and the behavior of the other status/control signals.

C. 4D-PAM5 Mapper

The 4D-PAM5 mapper was verified through both directed tests of the LUT block and wrapper-level verification of the sign-randomization logic. The PAM5_LUT converts Sd_n[8:0] into the quinary 4-vector.

The LUT golden model covers 13 table-defined mapping modes as derived from the IEEE tables, while the mapper wrapper test checks 15 mapper modes from the namespace, as there are additional carrier-extension and carrier-extension-error modes. The LUT was verified using a Python golden model derived from the IEEE 802.3-2018 Clause 40 Tables 40-1 and 40-2, as shown in Figures 4 and 5. The Python model covers all of the 13 LUT map modes across 512 possible values of Sd_n[8:0], resulting in $13 \times 512 = 6656$ references. A stimulus file is written from these references and checked by the testbench, allowing verification of the RTL PAM5_LUT output with the Python-generated expected output.

An exhaustive wrapper-level RTL testbench was used to verify the 4D-PAM5 mapper, by checking the bounded input space, including sign-randomization logic. Both sign-reversal states are set by holding tx_enable = 0 or 1 long enough to propagate through the internal tx_enable memory. Then, the testbench sweeps through the combinational input space, for a total of:

$$\begin{aligned}&(2 \text{ sign-reversal states}) \times (15 \text{ mapping modes}) \\ &\times (512 \text{ Sd\_n[8:0] values}) \times (16 \text{ Sg\_n[3:0] values}) \\ &= 245,760 \text{ possible combinations}\end{aligned}$$

For each combination, the testbench obtains a base 4-vector from a separate PAM5_LUT instance, and then computes the expected signed output using expected_quinary, and then verifies that this matches the mapper output quinary_symbols_packed.

D. Top-Level Verification

1) During reset, checks that therm_A == 16'd0. Does the same for therm_B, therm_C, therm_D.
2) Checks that therm_A's value is known. Does the same for therm_B, therm_C, therm_D.
3) Sets tx_enable high for 3 clock cycles, and then set low. Checks csreset == 1'b1.
4) Checks that 5 cycles out of 40 have the four thermometer outputs having different values, to verify independent outputs.
5) Checks that there are at least 8 total changes across the four thermometer outputs when tx_enable = 0 for 30 cycles.

A testbench tx_digital_tb.v was implemented as a smoke testbench for the top-level module tx_digital, which instantiates the scrambler, encoder, mapper, and the four FIR/DAC interface lanes. This testbench is checks that the thermometer outputs only contain known values. It also verifies reset connectivity to the FIR outputs.

VI. Synthesis, Placement, and Routing

The physical design flow was run through the Berkeley Hammer/Chipyard VLSI flow. Cadence Genus was used for RTL synthesis and standard-cell mapping, Cadence Innovus was used for floorplanning, placement, clock-tree synthesis, routing, and post-route analysis. Cadence Pegasus was used for physical verification (DRC/LVS).

A. Synthesis

The final RTL implementation was synthesized and physically implemented as a top level module tx_digital, which contains the side-stream scrambler, trellis encoder, 4D-PAM5 mapper, and the four downstream FIR lanes.

IEEE 802.3 Clause 40 states that 1000BASE-T uses 125 MBd signaling on each of the four wire pairs to match the 125 MHz GMII clock, and that the symbol period is 8 ns. Thus, the main performance target for this design is that one symbol-vector must be generated every 8 ns:

$$T_{\text{clk}} = 8 \text{ ns}$$
$$f_{\text{clk}} = 125 \text{ MHz}$$

B. Place & Route

The final place-and-route layout of the tx_digital block covers a $180\mu\text{m} \times 180\mu\text{m}$ die area, and the placement density was finalized at 69.958%. It passes setup and hold timing over worst-case corners, passes max-transition SKY130nm DRVs, and passes Pegasus DRC/LVS. This physical design passes all of the checks performed in the flow, including post-route timing, DRV checks, DRC, and LVS.

VII. PPA Summary

A. Timing

Post place & route timing was evaluated. The setup-critical path, or the longest datapath, was determined to start at scrambler_inst/Scr_n_reg[30], going through the scrambler, encoder, mapper, and FIR-B logic, ending at fir_B_inst/therm_out_reg[10]. The final design meets setup timing constraints with +72 ps slack. The hold-critical path, or the shortest datapath, goes from LFSR_seed[21] through the scrambler register input path, meeting hold time constraints with +91 ps slack. Setup timing was evaluated using the slow-slow corner, and hold timing was evaluated in the fast-fast corner.

TABLE II
Instance Count Summary

Metric	Value
Total placed instances including fillers	3038
Functional instances excluding fillers	1253
Flip-flops	124
Latches	0
Buffer / inverter / delay cells	563
Other combinational logic cells	566
Filler / physical cells	1785

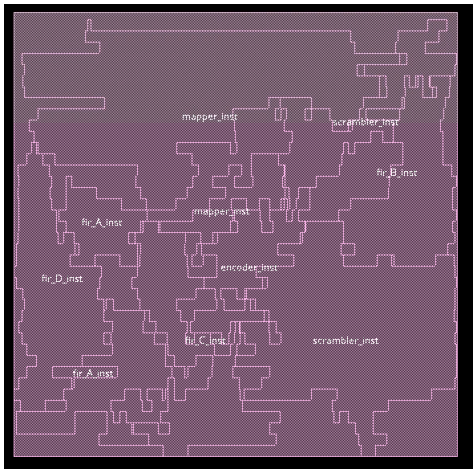

Fig. 6. Placement View

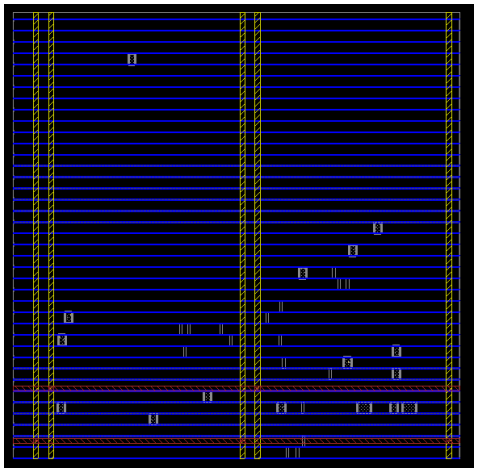

Fig. 7. Floorplan View

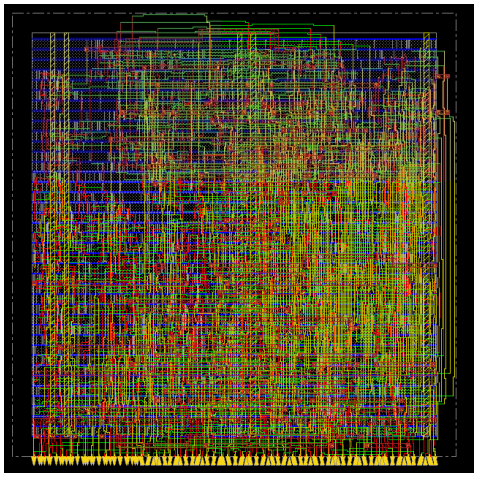

Fig. 8. Physical View

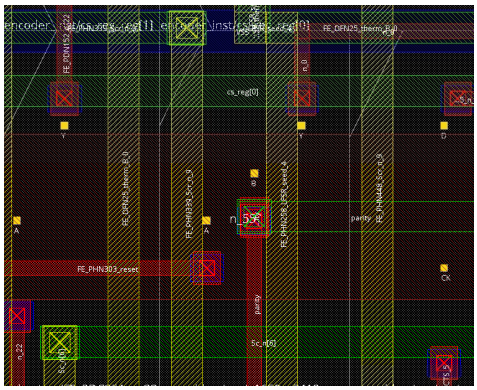

Fig. 9. Zoomed Routing Layout Detail

TABLE III
POST-ROUTE TIMING SUMMARY

Check	Corner	Path Type	Slack	TNS	Result
Setup	`ss_100C_1v60`	Reg-to-reg max	+0.072 ns	0.000 ns	Pass
Hold	`ff_n40C_1v95`	Input-to-reg min	+0.091 ns	0.000 ns	Pass

B. Cell Area Breakdown

Table 3 represents the standard-cell instance areas after P&R, not the full physical die. These areas do not include the routing, power strap, etc.

TABLE IV
PLACE&ROUTE CELL AREA BREAKDOWN

Block	Inst. Count	Cell Area [μm^2]	% Total Area
`scrambler_inst`	320	6,398.784	34.59%
`mapper_inst`	277	2,959.408	16.00%
`mapper_inst/pam5_mapper_inst`	192	1,971.024	10.66%
`fir_B_inst`	124	1,759.666	9.51%
`fir_A_inst`	120	1,740.622	9.41%
`fir_C_inst`	119	1,637.784	8.85%
`fir_D_inst`	109	1,578.748	8.53%
`encoder_inst`	74	1,106.456	5.98%
Total: `tx_digital`	**1258**	**18,497.407**	**100%**

C. Power

Power analysis was calculated in the typical-typical $25°C$, 1.8V corner. All power calculations are vectorless analysis and done under the assumption of default activity, with sequential element activity and primary input activity factors of 0.2.

TABLE V
POWER BREAKDOWN

Power Component	Power [mW]	% of Total
Internal power	1.6569	53.31%
Switching power	1.4502	46.66%
Leakage power	0.00081	0.026%
Total power	**3.1079**	**100%**

Table 5 discusses the breakdown of power amongst sequential, combinational, and clock logic.

TABLE VI
LOGIC-GROUPED POWER BREAKDOWN

Group	Internal [mW]	Switching [mW]	Leakage [mW]	Total [mW]	% Total
Sequential	0.8749	0.0816	0.00030	0.9568	30.79%
Combinational	0.6576	1.1680	0.00050	1.8260	58.76%
Clock comb.	0.1244	0.2005	0.0000095	0.3249	10.45%
Total	**1.657**	**1.450**	**0.00081**	**3.108**	**100%**

VIII. CONCLUSION

This paper discussed the implementation of a 1000BASE-T digital transmit front end, consisting of a 33-bit LFSR side-stream scrambler, a trellis encoder, a 4D-PAM5 mapper, and a four-lane FIR/thermometer code DAC interface. This design follows the 1000BASE-T PCS standard set by IEEE 802.3 Clause 40, and meets the requirement of generating a four-quinary symbol vector every 8 ns with a 125 MHz clock frequency. RTL design and verification was completed for all of the digital transmit modules, and the design was fully physically designed and signed-off, achieving 72 ps setup slack, 91 ps hold slack, 3.108 mW total power, and 18497.407 μm^2 standard-cell area.

REFERENCES

[1] Physical Layer Parameters and Specifications for 1000 Mb/s Operation Over 4-Pair of Category-5 Balanced Copper Cabling, Type 1000BASE-T, IEEE Standard 802.3ab-1999, 1999.

[2] IEEE Std 802.3-2018, Clause 40, “Physical Coding Sublayer (PCS), Physical Medium Attachment (PMA) sublayer and baseband medium, type 1000BASE-T.