LiTformer: Efficient Modeling and Analysis of High-Speed Link Transmitters Using Non-Autoregressive Transformer

Songyu Sun[1], Xiao Dong[1], Yanliang Sha[2], Quan Chen[2], Cheng Zhuo[1,3,*]
[1]Zhejiang University, Hangzhou, China; [2]Southern University of Science and Technology, Shenzhen, China
[3]Key Laboratory of Collaborative Sensing and Autonomous Unmanned Systems of Zhejiang Province, Hangzhou, China
*Corresponding email: czhuo@zju.edu.cn

Abstract

High-speed serial links are fundamental to energy-efficient and high-performance computing systems such as artificial intelligence, 5G mobile and automotive, enabling low-latency and high-bandwidth communication. Transmitters (TXs) within these links are key to signal quality, while their modeling presents challenges due to nonlinear behavior and dynamic interactions with links. In this paper, we propose LiTformer: a Transformer-based model for high-speed link TXs, with a non-sequential encoder and a Transformer decoder to incorporate link parameters and capture long-range dependencies of output signals. We employ a non-autoregressive mechanism in model training and inference for parallel prediction of the signal sequence. LiTformer achieves precise TX modeling considering link impacts including crosstalk from multiple links, and provides fast prediction for various long-sequence signals with high data rates. Experimental results show that LiTformer achieves 148-456× speedup for 2-link TXs and 404-944× speedup for 16-link with mean relative errors of 0.68-1.25%, supporting 4-bit signals at Gbps data rates of single-ended and differential TXs, as well as PAM4 TXs.

1 Introduction

The increasing demand for advanced computing capabilities in emerging data-driven applications, such as artificial intelligence (AI), 5G mobile networks, and automotive technologies, emphasizes the need for systems that are energy-efficient, cost-effective, and high-performance [1–4]. In recent years, as the rising costs of silicon manufacturing and constraints in on-chip integration density continue to challenge the industry, chiplet-based high-density heterogeneous integration (HDHI) has emerged and is increasingly prevalent in various applications [5, 6].

High-speed serial links, essential for low-latency communication, rapid data transfer, and effective data processing, form the backbone of such advanced high-speed systems. These links typically include high-speed transmitters (TXs), interconnects (transmission lines), and receivers (RXs), as illustrated in Figure 1 [6, 7]. To accommodate the growing need for high-bandwidth and efficient communication, these links feature extensive density, with hundreds to thousands of signal pathways, and operate at high frequencies and data rates up to gigabits per second (Gbps), which has to address non-trivial signal integrity (SI) issues [6, 8, 9], such as crosstalk, signal attenuation, electromagnetic interference (EMI), *etc*.

In the high-speed links, TXs are one of the most crucial and resource-intensive components, as their performance directly affects the quality of the initial transmitted signals and the overall link integrity [10]. A degraded output signal from the TX can lead to significant distortion at the final RX, resulting in erroneous recovery of transmitted signals. To maintain high-quality and high-speed output signals, which are essential for preserving SI across the transmission

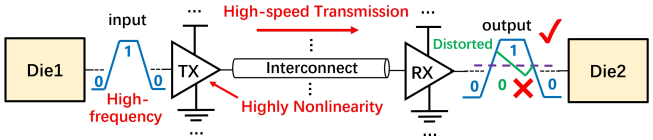

Figure 1: An example of inter-chip high-speed link.

path, TXs must operate at high frequencies and minimize timing errors due to process, voltage, temperature (PVT) variations, varying load conditions, *etc*. They must also adhere to design metrics such as slew rate, signal swing, and equalization [11]. Thus, developing efficient models for these TXs is crucial for accurately predicting signal distortion and inter-symbol interference (ISI), thereby enhancing data transmission quality and reducing error rates. However, the evaluation of TXs in high-speed links often poses several challenges:

- High-speed TXs inherently possess strong nonlinear behaviour, which is further aggravated by parasitics especially when operating at the elevated Gbps data rates, significantly affecting the signal amplitudes or effective spectrum. Accurately capturing such robust nonlinearity at high frequencies is challenging and vital for TX modeling [12].
- The transmission lines and RXs, as loads within the links, will significantly impact TX outputs [13], in additional to complex crosstalk from dense links, which makes TX analysis very complicated and time-consuming. It is insufficient to simplify the problem of modeling TXs merely into modeling input-output signal characteristics of nonlinear circuits. It is hence critical to consider TX dynamics within the links, modeling the impacts of link parameters and crosstalk.
- The nonlinearity and "memory" effect of TXs imply that an individual bit can affect several subsequent bits [14]. Multi-bit effects cannot be simply formulated using single-bit pulse response [15, 16] or edge response [17–19] via linear time-invariant (LTI) principles. Instead, it is necessary to directly predict multi-bit output sequences of TXs.

The most accurate way to analyze TX behavior in many links is using transistor-level models like SPICE models [20]. However, they are very time-consuming due to the complex internal circuit details. Empirical behavioral models such as current source-based models (CSMs) and I/O buffer information specification (IBIS) models exhibit fast speed but usually have limited accuracy or capabilities in modeling complex interdependencies [21–23].

Recently, Artificial Neural Networks (ANNs) have gained interest from both academia and industry, which significantly enhance the speed and efficiency of circuit modeling and simulation, especially for nonlinear components and systems [13, 24–29]. The inefficiency of static neural networks in modeling time-series sequences has further spurred research into time-domain sequence-to-sequence

(seq2seq) neural networks for nonlinear circuit macromodeling [25–29], *e.g.*, Recurrent Neural Networks (RNNs) and Long Short-Term Memory networks (LSTMs) [25, 28–30]. While these models can capture the internal dependency of temporal signal sequences through a recurrent structure, they struggle with vanishing/exploding gradients as well as extensive training time, making them unsuitable for highly nonlinear long-sequence signals with long-range dependencies. Their sequential nature also complicates the handling of non-sequential link parameters, limiting their effectiveness in modeling TX behavior considering various link parameters. In contrast, Y. Zhao *etal*. developed a static Feedforward NN (FNN)-based model that includes link parameters but neglects the internal dependency of the signal sequence and overlooks crosstalk, only supporting weakly nonlinear signals from a single link TX [13, 24].

For the future deployment of TX modeling in high-speed and high-density links, it is critical to:

- Effectively incorporate both input signals and link parameters into the model to capture TX dynamics within links;
- Accurately capture long-range dependencies within the signal for precise long-sequence signal prediction;
- Efficiently account for complex crosstalk from multiple links.

In this paper, to tackle the aforementioned challenges, we propose **LiTformer**, *a non-autoregressive Trans**former**-based model for high-speed **li**nk **T**Xs*, utilizing a non-sequence-to-sequence (nonseq2seq) encoder-decoder architecture. This model employs a non-sequential encoder and a Transformer decoder to achieve precise TX modeling considering link impacts, including crosstalk from multiple links, and provides fast predictions for various long-sequence signals with high data rates. To the best of our knowledge, this is a pioneering effort to apply Transformer-based models for macromodeling nonlinear circuits. Our main contributions are listed below:

- We propose **a novel Transformer-based model with a nonseq2seq encoder-decoder architecture for nonlinear TX modeling**, which not only considers both input signals and link characteristics including crosstalk, but also effectively captures long-range dependencies within the output signal sequences through the attention mechanism.
- We utilize **a non-autoregressive (NAR) approach for model training and inference**. We introduce an innovative one-pass filtered decoding technique that combines a single instance of parallel decoding with a single filtering step, ensuring short inference time and sufficient accuracy for long-sequence signals.
- The proposed *LiTformer* can **efficiently predict arbitrary 4-bit output signals at Gbps data rates for TXs across various links**, supporting single-ended and differential TXs, as well as *PAM4* (4-Level Pulse Amplitude Modulation) TXs.

Experimental results show LiTformer's efficiency in modeling high-speed link TXs for highly nonlinear long-sequence signals, accounting for different link parameters and crosstalk in many links. Our LiTformer demonstrates superior accuracy over previous works in modeling nonlinear TXs [13, 29]. When compared to SPICE [20], the proposed LiTformer achieves up to 148-456× speedup for 2-link TXs and 404-944× speedup for 16-link TXs with minimal deviations between 0.68-1.25%, supporting 4-bit signals with Gbps data rates of single-ended and differential TXs, as well as PAM4 TXs.

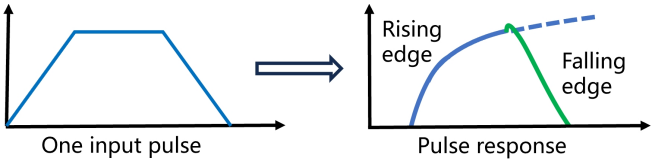

Figure 2: Pulse response of the highly nonlinear TX.

2 Background

2.1 Link Impact on TX Performance

In a set of links, the TX output is significantly impacted by its loading transmission lines and RXs via reflections, absorption, *etc*. These effects also cause near end crosstalk (NEXT) and signal distortions when undesired coupling from adjacent lines interferes with the TX signal, which is more complex in high-speed dense links introducing timing errors and data corruption. In practice, transmission lines and RXs vary greatly to adapt to various situations and ever-evolving demands of communication technologies. As a result, there is a critical need to consider the dynamic interaction between TXs and links, which necessitates TX models to effectively capture the varying characteristics of different links.

2.2 Characterization of Multi-Bit Response

Due to their nonlinear nature, most TXs exhibit "memory" effects where an output bit affects its subsequent bits, causing a ripple effect down the sequence. For a TX with m-bit memory, the response to the current bit $bit(0)$ dependent on the past $m-1$ bits can be expressed as a nonlinear function of time t and m bits $y(bit(-m+1), \ldots, bit(-1), bit(0), t)$. The reactions within the highly nonlinear output signal of high-speed TXs have long-range dependencies. To evaluate overall performance, it is crucial to determine all 2^m responses to m-bit random inputs for TXs [14].

The asymmetry between the rising and falling edges of the pulse response prevents synthesizing the 2^m responses by time-shifting and superposing unit pulses or single bit responses (SBR) via LTI principles [15, 16]. Moreover, with increased nonlinearity at high frequencies, the TX output may drop before rising to stable as shown in Figure 2. Hence, methods that superpose edge responses like double-edge responses (DER) and multiple edge responses (MER) [17–19] fail to completely capture the output. Accurate TX output representation necessitates a direct characterization of the entire multi-bit signal.

2.3 Deep Learning Model for Nonlinear TX

2.3.1 Related Works. ANN-based models have been widely studied for modeling nonlinear circuits in recent years [13, 24–29]. Characterizing the transient input-output behaviour of nonlinear circuits can be treated as a seq2seq task, leading researchers to apply time-domain recurrent neural networks, *e.g.*, dynamic neural networks (DNNs) [26], time-delay neural networks (TDNNs) [27], RNNs and LSTMs [25, 28, 29], to capture sequence dependencies. However, these models could not learn long-range dependencies in highly nonlinear signals due to vanishing/exploding gradients. Their time-sequential nature also renders them non-parallelizable, leading to extensive training and inference times. Moreover, they struggle with handling non-sequential link parameters for accurate modeling of TXs in the links. As a result, these seq2seq models are inadequate for nonlinear TX modeling. Only a few studies, like FNNs in [13, 24], consider link parameters in TX modeling. However, FNNs assume

no temporal relationships within signal sequences and are limited to weakly nonlinear signals at low data rates. Besides, [13, 24] focus solely on a single link without modeling TXs in many links with crosstalk. Thus, it remains an open question to develop an efficient model for high-speed link TXs.

2.3.2 Why Transformer? The Transformer proposed by Vaswani et al. [31] excels in dealing with long-range dependencies and parallel computation with the encoder-decoder structure. To achieve non-sequential inputs of link parameters and sequential output signals, it is natural to pair the Transformer decoder with a non-sequential encoder. Unlike RNNs or LSTMs, the Transformer decoder can manage non-sequential input through its attention mechanism instead of relying on strict sequential information unfolding over time, while effectively modeling highly nonlinear TX signals with long-range dependencies.

3 Basics of Non-Autoregressive Transformer

3.1 Basic Structures of Transformer

3.1.1 Multi-Head Attention Mechanism. The attention mechanism is the key innovation of Transformer, capturing long-range dependencies by allowing each element to attend over other elements regardless of distance, enabling to process the entire sequence simultaneously [31]. With the token embeddings transformed into query (Q), key (K) of dimension d_k, and value vectors (V) by multiplying with three learnable matrices W^Q, W^K and W^V, the attention mechanism calculates scaled dot products of the query with all keys and applies softmax to obtain the weights for the values as:

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V, \tag{1}$$

which allows integrating contextual information across the entire sequence by the summation of values weighted by query-key match.

Multi-head attention runs the above mechanism in parallel across independent attention heads. Each head i has its own parameters, W_i^Q, W_i^K and W_i^V. The outputs from each head are then concatenated and projected with W^O as follows:

$$\begin{aligned} \text{MultiHead}(Q, K, V) &= \text{Concat}\left(\text{head}_1, \ldots, \text{head}_\text{h}\right) W^O, \\ \text{where head}_i &= \text{Attention}\left(QW_i^Q, KW_i^K, VW_i^V\right). \end{aligned} \tag{2}$$

The combination of information from all heads provides a multi-perspective representation of the sequence, enhancing the model's ability to focus on various parts of the sequence concurrently and understand complex relationships.

3.1.2 Positional Encoding. The order of a sequence decisively influences its connotation, with RNNs and LSTMs naturally recognizing this through sequential processing. The attention mechanism processes the token relevance regardless their position, potentially causing randomization and error. Positional encoding (PE) is applied to the token embeddings before the attention mechanism processes the sequence, which assigns each token a position-specific vector to maintain their order information using an embedding matrix [31, 32]:

$$\begin{aligned} PE_{(pos,2i)} &= \sin\left(pos/10000^{2i/d_\text{model}}\right), \\ PE_{(pos,2i+1)} &= \cos\left(pos/10000^{2i/d_\text{model}}\right), \end{aligned} \tag{3}$$

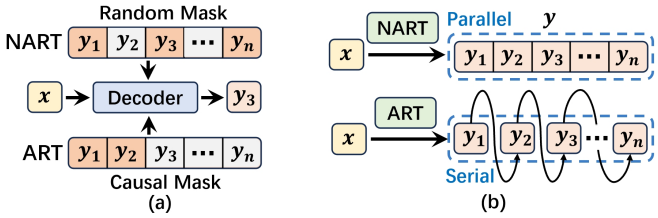

Figure 3: Comparison of NART and ART: (a) training process; (b) inference process.

where d_model is the embedding dimension typically equal to the model dimension, i is the index of the embedding dimension and pos is the position within the sequence. By adding the positional embeddings to the token embeddings, Transformer provides the representation of each token with its position information included.

3.1.3 Position-Wise Feed-Forward Networks. Position-wise Feed-Forward Networks (FFNs) consist of two dense layers mapping from d_model to another space of dimension d_{ff} and then back, which is separately and identically applied to each position in the sequence [31]. The FFN function with a ReLU activation is defined as:

$$\text{FFN}(x) = \max\left(0, xW_1 + b_1\right)W_2 + b_2, \tag{4}$$

where W_1, W_2, b_1, b_2 are learnable parameters.

3.2 Non-Autoregressive Mechanism

For sequence modeling tasks, most models adopt autoregressive (AR) methods, including AR Transformers (ARTs) [33, 34]. Given an input source $\boldsymbol{X}$, ARTs generate a target sequence $\boldsymbol{y} = y_1, y_2, \ldots, y_n$ of length n through a chain of conditional probabilities:

$$p(\boldsymbol{y} \mid \boldsymbol{X}) = \prod_{i=1}^{n} p\left(y_i \mid y_{<i}, \boldsymbol{X}\right), \tag{5}$$

where $y_{<i}$ denotes the target tokens generated from $\boldsymbol{X}$ before the i^{th} token. Conditioning each token on its previous ones, ARTs have to perform n iterations to sequentially decode each y_i, making inference very time-consuming. To enhance the inference speed, NAR Transformers (NARTs) are gaining increasing attention for their efficient parallel decoding capabilities [35–38]. Assuming that output tokens are conditionally independent given $\boldsymbol{X}$, the NAR mechanism cancels out the left-to-right dependency and decomposes the conditional dependency chain of $p(\boldsymbol{y} \mid \boldsymbol{X})$ as:

$$p(\boldsymbol{y} \mid \boldsymbol{X}) = \prod_{i=1}^{n} p\left(y_i \mid \boldsymbol{X}\right), \tag{6}$$

which indicates that y_i relies only on the source $\boldsymbol{X}$, allowing simultaneous sequence decoding. Figure 3 (a) and (b) respectively compares the training and inference process between NARTs and ARTs. ARTs employ a causal mask to ensure predicting y_i based only on prior tokens during training, and sequentially decode each y_i during inference, while NARTs apply random masking to remove sequential dependencies and decodes the entire sequence $\boldsymbol{y}$ in parallel [36].

4 Problem Formulation

4.1 Formulation of TX Dynamics within Links

4.1.1 Overview. Figure 4 depicts the equivalent circuit used for analyzing TXs in an N-link system: (a) for single-ended and (b) for differential TXs. Each TX connects to a load capacitance C_L, one (two)

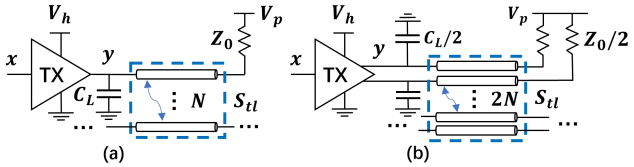

Figure 4: Equivalent circuit model for TXs in an N-link system: (a) single-ended TXs; (b) differential TXs.

non-ideal transmission line(s) characterized by S-parameters $\boldsymbol{S}_{tl}$, and a pull-up impedance Z_0 modeling the RX design with a pull-up level V_p. Reference lines are omitted for simplicity. The input data stream $\boldsymbol{x}$ transmitted by each TX can be formatted as a series of m binary symbols $\boldsymbol{x}_s = x_0, x_1, \ldots, x_{m-1}$ where x_i = "0", "1", $i = 0, \ldots, m-1$. A two-tap finite impulse response (FIR) equalizer is typically located in the TX, which operates through a first-in first-out (FIFO) buffer to linearly adjust the transmitted voltage levels, attenuating low-frequency components and mitigating channel loss. The equalized data sequence $\boldsymbol{x}_s^d = x_0^d, x_1^d, \ldots, x_{m-1}^d$ of $\boldsymbol{x}_s$ is calculated by:

$$\begin{aligned} x_i^d &= H_0 x_i + H_1 x_{i-1}, \qquad i = 1, \ldots, m-1 \\ \text{where } \ & |H_0| + |H_1| = 1, \qquad H_0 > 0, H_1 < 0 \end{aligned} \tag{7}$$

where H_0 and H_1 are the taps of the filter. Due to parasitic coupling of transmission lines, each link potentially interferes with the TX of every other link through NEXT. We aim to predict the interfered TX output signals $\boldsymbol{y}$ in an N-link system given input signal sequences $\boldsymbol{x}$ and link parameters $H_0, V_h, C_L, \boldsymbol{S}_{tl}, Z_0, V_p$.

4.1.2 Parameterization of Input Signal. To transmit $\boldsymbol{x}_s$ through the links, NRZ (Non-Return-to-Zeros) signaling uses trapezoidal wave with each level corresponding to "0" or "1", characterized by its amplitude V_h matching the supply voltage, a signal period t_p and transition time t_{rf} assuming equal rising and falling phase. PAM4 uses equally spaced 4 distinct levels of V_h to carry two bits ("00", "01", "10", "11") per pulse, with a pulse period t_p and transition time t_{rf} between two levels, doubling the data rate. Despite t_{rf}'s drastic variations across signal periods, its proportion remains stable. We use the ratio $r_{rf} = \frac{t_{rf}}{t_p}$ to represent transition time. Therefore, we could fully describe the input signal $\boldsymbol{x}$ by the symbol sequence $\boldsymbol{x}_s$ and its waveform parameters of V_h, t_p and r_{rf}.

4.1.3 Interfered Output Decomposition. Given that the intrinsic output of each TX is independent and is a foundation upon which crosstalk independently contributed by the other $N-1$ links is superimposed, the interfered output $\boldsymbol{y}_i$ of TX$_i$ could be decomposed into its intrinsic output $\boldsymbol{y}_i^{intr}$ absent any crosstalk plus the accumulated crosstalk from other $N-1$ links, expressed as:

$$\boldsymbol{y}_i = \boldsymbol{y}_i^{intr} + \sum_{j=1, j\neq i}^{N} \boldsymbol{C}_{ij}, \tag{8}$$

where $\boldsymbol{C}_{ij}$ is the NEXT from link$_j$ onto link$_i$. For problem regularity, we evaluate the intrinsic output $\boldsymbol{y}_i^{intr}$ in a basic 2-link setup. Analyzing TX$_i$ in an N-link system could then be simplified into analyzing $N-1$ instances of 2-link systems of link$_i$ and link$_j$, focusing on only one input signal each time: (1) determine $\boldsymbol{y}_i^{intr}$ given link$_i$'s input $\boldsymbol{x}_i$ which is conducted once with an arbitrary link$_j$, $j \neq i$, and (2) determine $\boldsymbol{C}_{ij}$ given link$_j$'s input $\boldsymbol{x}_j$, which is repeated $N-1$ times for $j = 1, \ldots, N, j \neq i$.

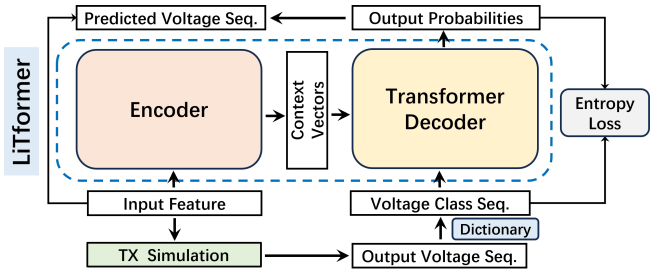

Figure 5: Model Overview of the proposed LiTformer.

4.2 Model Formulation

As per analysis, our objective is summarized as characterizing TX$_i$'s intrinsic output $\boldsymbol{y}_i^{intr}$ and the crosstalk component $\boldsymbol{C}_{ij}$ separately in a 2-link system. We introduce a boolean variable K to differentiate these two cases: K=0 for $\boldsymbol{y}_i^{intr}$ and K=1 for $\boldsymbol{C}_{ij}$. The TX model could then be formulated by the function f as:

$$\boldsymbol{y}_K = f(K, H_0, \boldsymbol{x}_s, V_h, t_p, r_{rf}, C_L, \boldsymbol{S}_{tl}, Z_0, V_p), \tag{9}$$

where $\boldsymbol{S}_{tl}$ represents S-parameters of the transmission lines in a 2-link system, sized 4×4 for single-ended and 8×8 for differential TXs. With the interested TX$_i$'s link$_i$ always treated as the first link in the 2-link system, when $K = 0$ or 1, $\{\boldsymbol{x}_s, V_h, t_p, r_{rf}\}$ is considered the input signal parameters of the first link (victim) or the second link (aggressor), for intrinsic output or crosstalk prediction.

Despite voltage continuity, we reformulate the regression problem of Equation (9) as a classification one to enhance model performance. We categorize the entire voltage range with the step size Δv between each voltage category and construct a dictionary to map a voltage value to its nearest class. Since intrinsic and crosstalk components require different categorization granularity due to their magnitude mismatch, we use two dictionaries with equal length for model structural consistency - $\mathbf{D}_I$ for intrinsic outputs ranging from $v_{\min}^I$ to $v_{\max}^I$ with Δv^I, and $\mathbf{D}_C$ for crosstalk from $v_{\min}^C$ to $v_{\max}^C$ with Δv^C:

$$\mathbf{D}_I = \left\{ v_k : Class_{k+1} \mid v_k = v_{min}^I + k \cdot \Delta v^I,\ k = 0, 1, \ldots, \frac{v_{\max}^I - v_{\min}^I}{\Delta v^I} \right\}, \tag{10}$$

$$\mathbf{D}_C = \left\{ v_k : Class_{k+1} \mid v_k = v_{\min}^C + k \cdot \Delta v^C,\ k = 0, 1, \ldots, \frac{v_{\max}^C - v_{\min}^C}{\Delta v^C} \right\}. \tag{11}$$

$Class_0$ is assigned to a special <*mask*> token in both $\mathbf{D}_I$ and $\mathbf{D}_C$. With sufficiently small Δv, we can accurately reconstruct the original voltage after categorizing it. Eq. 9 is finally converted into:

$$\boldsymbol{y}_c = f_c(K, H_0, \boldsymbol{x}_s, V_h, t_p, r_{rf}, C_L, \boldsymbol{S}_{tl}, Z_0, V_p) = f_c(\boldsymbol{X}), \tag{12}$$

where f_c is the function generating the voltage category sequence $\boldsymbol{y}_c$ from the input features $\boldsymbol{X} = \{K, H_0, \boldsymbol{x}_s, V_h, t_p, r_{rf}, C_L, \boldsymbol{S}_{tl}, Z_0, V_p\}$.

5 Proposed Model Architecture

We implement an encoder-decoder architecture for *LiTformer*, containing a non-sequential encoder and a sequential Transformer decoder as shown in Figure 5. The non-sequential encoder processes unordered non-sequential inputs into context vectors with richer information. The decoder combines these vectors with the class sequence from the simulated TX outputs to generate token-wise class probability distributions, which is used for the cross-entropy (CE)

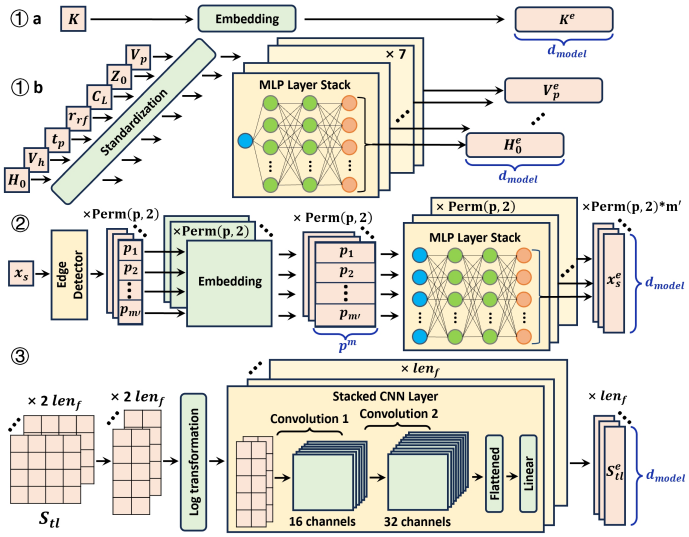

Figure 6: Architecture of the LiTformer encoder: ① for scalar features; ② for the input signal sequence; ③ for S-parameters.

loss calculation against the true class sequence for model optimization. These probability distributions transform into the output signal sequence during inference as detailed in Section 6.2. The proposed nonseq2seq architecture surpasses traditional seq2seq and static models in modeling nonlinear TXs with additional link parameters.

5.1 Encoder Architecture

In this section, we introduce the various components of the proposed *LiTformer* encoder as shown in Figure 6, which independently encodes the input features in Section 4.2 into context vectors with a dimension of d_{model} for subsequent decoder processing.

5.1.1 Scalar Features Encoding. We encode the binary variable K as a d_{model}-dimensional vector K^e through a $2\times d_{\text{model}}$ embedding matrix as depicted in Figure 6 ① a. For scalar features $H_0, V_h, t_p, r_{rf}, C_L, Z_0, V_p$, as shown in ① b, we standardize each by deducting the mean and dividing by the standard deviation to mitigate scale discrepancies and facilitate faster convergence. These standardized features are then converted into d_{model}-dimensional vectors $H_0^e, \ldots, V_p^e$ via stacked Multi-Layer Perceptrons (MLP) layers — each feature fed into an individual single-input MLP with two hidden layers of 16 neurons, ReLU activation in the hidden layers, and linear output activation.

5.1.2 Input Signal Sequence Encoding. To deserialize an m-symbol input sequence $\boldsymbol{x}_s$, we propose to detect its level transition positions for identification. For a signal modulated with p levels, there are $\text{Perm}(p, 2)$ possible directional transitions, where $\text{Perm}(p, 2)$ represents the permutation number of arranging 2 out of p distinct states, in total 2 types of edges for NRZ and 12 for PAM4. The transition from symbol u to v is denoted as $edge^{u\to v}$, where $u, v = 0, 1, \ldots, p-1, u \neq v$. Each x_i is assigned with a position index $i+1$ with valid positions from 1 to m. 0 is reserved to indicate an invalid position.

Traversing the entire sequence of $\boldsymbol{x}_s$, we mark the position of $edge^{x_i\to x_{i+1}}$ at position $i+1$ for rising edges with $x_i < x_{i+1}$ or at i for falling edges with $x_i > x_{i+1}$, where $i = 0, \ldots, m-2$. Since the signal rests at low before and after transmission, there is a rising $edge^{0\to x_0}$ at 1 for nonzero x_0 and a falling $edge^{x_{m-1}\to 0}$ at m for nonzero x_{m-1}. Each edge appears up to m' times, where $m' = m/2$ for even m or

Algorithm 1 Edge Detector for a p-Level m-Symbol Sequence

Input: A p-level m-symbol sequence $\boldsymbol{x}_s = x_0, x_1, \ldots, x_{m-1}$
Output: Arrays of positions for each transition edge $edge^{u\to v}$

Set $m' \leftarrow m/2$ if m is even, or $m' \leftarrow (m+1)/2$ if m is odd
Initialize An array for each $edge^{u,v}$, with $u, v = 0, 1, \ldots, p-1, u \neq v$, of length m' and filled with zeros
Set invalid position marker to 0
Mark $edge^{0\to x_0}$ at position 1 if $x_0 \neq 0$
Mark $edge^{x_{m-1}\to 0}$ at position m if $x_{m-1} \neq 0$
for $i = 1$ to $m-1$ **do**
    if $x_{i-1} < x_i$ **then**
        Mark $edge^{x_{i-1}\to x_i}$ at position $i+1$
    else if $x_{i-1} > x_i$ **then**
        Mark $edge^{x_{i-1}\to x_i}$ at position i
    end if
end for
return Arrays of positions for all $edge^{u\to v}$

$m' = (m+1)/2$ for odd, with zeroes padding for uniformity. We summarize the edge detection algorithm in Algorithm 1, through which we achieve using non-sequential parameters to uniquely determine $\boldsymbol{x}_s$. For an NRZ "1011" sequence, $edge^{0\to 1}$ and $edge^{1\to 0}$ is located at $\{1, 3\}$ and $\{1, 4\}$ respectively. For PAM4 "0131", $edge^{0\to 1}$, $edge^{1\to 3}$, $edge^{3\to 1}$ and $edge^{1\to 0}$ are located at $\{2, 0\}$, $\{3, 0\}$,$\{3, 0\}$ and $\{4, 0\}$, while all other edges are set to $\{0, 0\}$.

As shown in Figure 6 ②, $\boldsymbol{x}_s$ is transformed into m' position indices for $\text{Perm}(p, 2)$ edges, which are embedded into a set of continuous p^m-dimensional vectors through $\text{Perm}(p, 2)$ instances of $(m+1)\times p^m$ embedding matrices unique to each $edge^{u\to v}$. These embeddings individually go through stacked MLPs as in Section 5.1.1 adapted for p^m input, with parameters unique to $edge^{u\to v}$. We finally obtain $\boldsymbol{x}_s^e$ containing $\text{Perm(p, 2)} * m'$ vectors each of dimension d_{model}.

5.1.3 S-parameter Encoding. We keep the frequency response nature of S-parameters, viewing their impact upon the system at each frequency as key factors. $\boldsymbol{S}_{tl}$ with len_f frequency points can be decomposed into a real- and an imaginary-part matrix at each frequency. By exploiting their inherent symmetry to reduce redundancies, we streamline the model by considering only $\frac{n^2+n}{2}$ effective values from an $n \times n$ matrix. We reshape the 4×4 (8×8) $\boldsymbol{S}_{tl}$ with 10 (36) valid entries into 2×5 (6×6) for 2-link single-ended (differential) TXs. Since S-parameters have quite small magnitude with a huge fluctuation in the order, and can be positive or negative, we shift their entries s to positive and apply a logarithmic transformation to linearize and stabilize data distribution:

$$s_{scaled} = log(s + 1.1 * min(S_{tl})). \tag{13}$$

The scaled matrices are processed through two consecutive Convolutional Neural Network (CNN) layers at each frequency point: the first CNN expanding the two input channels corresponding to the real and imaginary matrix into 16 outputs and the second further expanding into 32 outputs, both with a kernel of size 1 and ReLU activation. The resulting feature map is flattened and passed through a linear layer to produce a final $\boldsymbol{S}_{tl}^e$ with a dimension of $(len_f, d_{\text{model}})$. The $\boldsymbol{S}_{tl}$ encoding process is shown in Figure 6 ③.

The encoded features $\{K^e, H_0^e, V_h^e, t_p^e, r_{rf}^e, C_L^e, Z_0^e, V_p^e, \boldsymbol{x}_s^e, \boldsymbol{S}_{tl}^e\}$ are finally combined together into an unordered d_{model}-dimensional

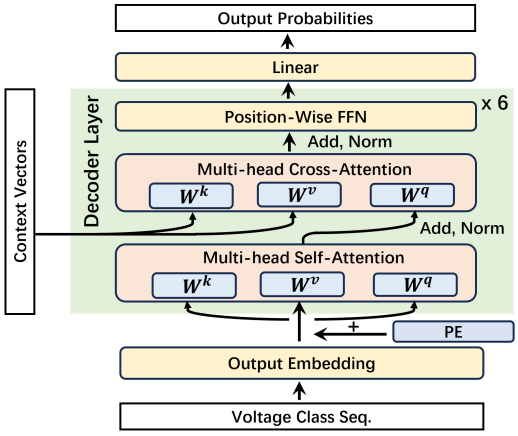

Figure 7: Architecture of the LiTformer decoder.

vector X^e of length $8 + len_f + \text{Perm}(p, 2) * m'$, which is subsequently fed into the decoder for interaction with the output sequence.

5.2 Decoder Architecture

We use a standard Transformer decoder architecture [31] as shown in Figure 7. The voltage signal class sequence is embedded, added to the PE from Eq. 3, and normalized for stability before entering the stacked 6 identical decoder layers. Each layer includes a multi-head self-attention module processing decoder outputs, a multi-head cross-attention module using context vectors for K and V and decoder outputs for Q in Eq. 1, which facilitates interaction between encoded non-sequential input features and the output sequence, followed by a position-wise FNN. The number of attention heads is all set to 8. Any two sub-layers are connected through residual connections and layer normalization. The final output is processed through a linear layer that translates dimensions from d_{model} to dictionary size, determining output probabilities for each token in the output sequence.

6 Training and Inference of NAR LiTformer

6.1 Randomly Masked Training

To develop an NAR model, we modify the standard left-to-right decoder's attention mask to allow context integration from both sides for token prediction. Given the input feature X and a subset of target tokens Z, the decoder calculates probabilities for a predetermined set of target tokens Y_{mask}, treating them as conditionally independent. It predicts $P(y|X, Z)$ for each token y in Y_{mask}. A key advantage in the issue of modeling TX is that the output sequence length, denoted as n, is fixed based on the m-symbol input signal and unaffected by input variability. This eliminates the need for sequence length prediction, reducing the complexity associated with NAR models [35, 36] and boosting performance.

In training, masked tokens are randomly selected as per [36]. We sample the number of masked tokens n_{mask} from a uniform distribution between 1 and n, then randomly choose n_{mask} tokens as Y_{mask}, replacing their values with <*mask*> which is designated $Class_0$ as in Section 4.2. The variety in masking schemes allows the model to adapt to scenarios ranging from easy (fewer masks) to challenging (more masks) when learning missing data [36]. The decoder processes the masked sequence along with context vectors and generates an output sequence of the same length in one pass, instead of sequentially as in RNNs, using the attention mechanism that supports efficient parallel training. The model is trained using stochastic gradient descent to optimize the CE loss between predictions and target tokens across all Y_{mask} tokens:

$$L_{CE} = -\sum_{y_i \in Y_{mask}} \log P\left(y_i \mid \boldsymbol{X}, \boldsymbol{Z}\right). \tag{14}$$

where $P\left(y_i \mid \boldsymbol{X}, \boldsymbol{Z}\right)$ represents the model's probability estimate for the true class of the masked token y_i. Although the decoder predicts the full sequence, the loss is calculated only on Y_{mask}, directing the model to infer missing information.

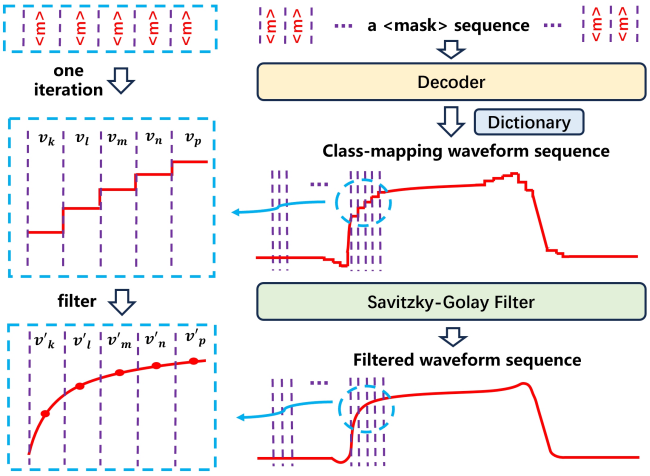

Figure 8: Schematic of the proposed one-pass filtered decoding approach.

6.2 Inference with One-Pass Filtered Decoding

After training, the model can simultaneously predict the masked tokens given X. At the beginning of inference, without any information about the target, we input a fully masked sequence into the decoder and get all tokens predicted in parallel. We choose the class with the highest probability for each token:

$$y_c^* = \arg\max_{y_c \in \mathbf{D}} P(y_c), \tag{15}$$

where $\mathbf{D}$ is the set of classes in $\mathbf{D}_I$ or $\mathbf{D}_C$. The output signal can then be reconstructed from $\mathbf{D}_I$ or $\mathbf{D}_C$. It is observed that the first-time decoded signal is already consistent with groundtruth, with minimal unevenness and deviation stemming from accuracy loss in categorization and model prediction. **One-time decoding is enough to achieve good performance.** We have also observed that **for signal waveform, iterative decoding essentially acts as a filter for decoding convergence,** which smooths signal irregularities without drastically altering its amplitude. Rather than iteratively refining uncertain predictions which risks introducing new errors without an explicit stopping condition [36], we propose a one-pass filtered decoding approach utilizing waveform continuity. As shown in Figure 8, we apply a Savitzky-Golay filter [39] on the first-time decoded and reconstructed signal sequence to smooth it and modify error to obtain the final output. With just a single parallel inference and a filtering process, we achieve efficient inference of the long-sequence signal without decoding latency, in contrast to AR methods which sequentially generate one output at a time.

Table 1: Ranges of input signal parameters, link parameters except for H_0 between 0.8 and 1.0 and the line length.

TX	Input signal parameters			Link parameters			
	V_h (V)	t_p (ps)	r_{rf} (%)	C_L (pF)	Z_0 (Ω)	V_p (V)	l (cm)
TS1	0.8~1.2	80~100	5~20	0.2~1.6	50~70	0.5~1.0	0.1~10
TS2	0.8~1.2	100~150	5~20	0.01~0.5	40~70	0.4~0.8	0.1~10
TD3	1.2~1.8	90~150	5~20	0.01~0.4	50~70	0.4~0.8	0.1~10
TP4	0.8~1.5	60~150	10~20	0.05~0.5	50~70	0.6~1.0	0.5~10

Table 2: Training and test errors of LiTformer and accuracy comparison of intrinsic and crosstalk outputs between the proposed LiTformer and SPICE.

TX	Train error	Test error	Intrinsic output		Crosstalk output	
			CE	Mean AE/RE	CE	Mean AE/RE
TS1	0.335	1.97	2.64	4.75mV/0.61%	1.31	0.36mV/1.90%
TS2	0.362	1.95	2.68	3.47mV/0.48%	1.22	0.29mV/2.19%
TD3	0.350	2.74	3.71	7.36mV/1.26%	1.77	0.49mV/3.32%
TP4	0.328	2.41	3.32	5.53mV/0.95%	1.49	0.39mV/1.93%

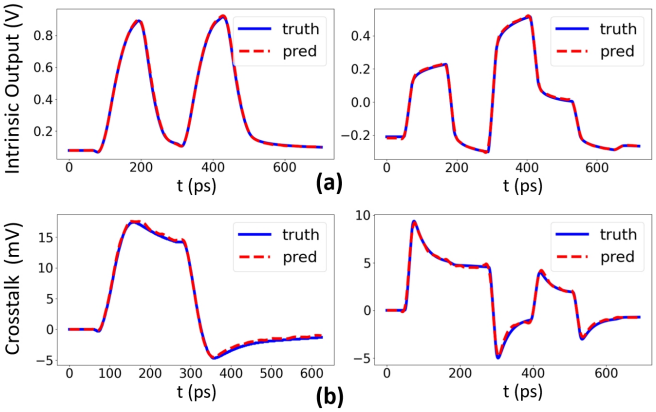

Figure 9: Waveform comparison between the proposed LiTformer and SPICE for: (a) intrinsic output; (b) crosstalk component, on TS2 (left) and TP4 (right).

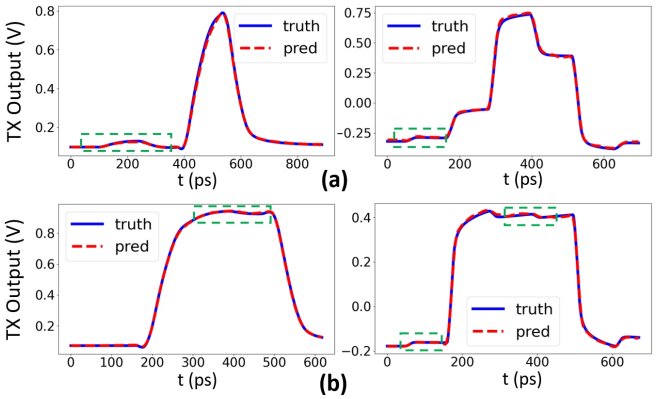

Figure 10: Comparison between LiTformer and SPICE for TX interfered waveforms in: (a) a 2-link system; (b) a 16-link system, on TS2 (left) and TP4 (right).

7 Experimental Results

7.1 Experiment Setup

We evaluated our *LiTformer* on 4 commercial high-speed TX designs: two NRZ single-ended TS1 and TS2, one NRZ differential

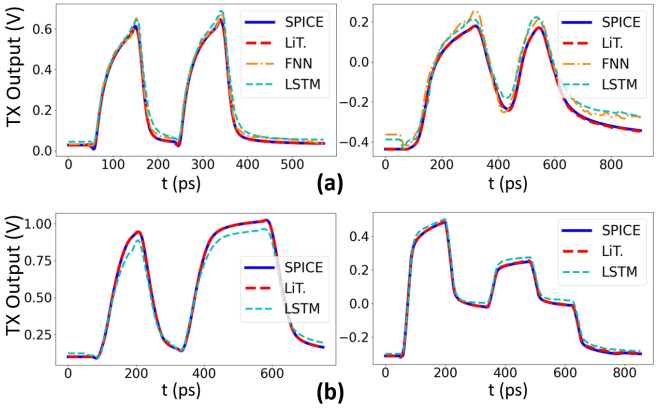

Figure 11: Comparison of 1-link TX output between: (a) LiTformer, FNN [13], LSTM [29] and SPICE with link parameters as model inputs on TS1 (left) and TD3 (right); (b) LiTformer, LSTM [29] and SPICE with only the input signal as model inputs on TS2 (left) and TP4 (right).

TD3 and one PAM4 single-ended TP4. The dataset as groundtruth was derived from HSPICE transient simulation on a server with a 3.00GHz Intel Core i9-9980XE and 128GB RAM. We simulated 2-link TX outputs using random 4-symbol input sequences (4-bit for NRZ, 8-bit for PAM4) with uniformly sampled parameters as detailed in Table 1, with H_0 between 0.8 and 1.0 for all cases. Due to the linear, periodic characteristics and band-limited response of transmission lines, we extracted S_{tl} using 5 frequency points per decade from 10Hz to 100GHz ($len_f = 51$) from different RLGC models of various lengths l. For intrinsic output samples $\boldsymbol{y}_i^{intr}$, we applied the input signal to the first link and kept the second low; for crosstalk $\boldsymbol{C}_{ij}$, we applied the input to the second with the first held high, and deducted $\boldsymbol{y}_i^{intr}$. We generated a 15k dataset with a balanced number of crosstalk and intrinsic samples to avoid bias and improve generalization, which was split into 12k for training, 1k for validation, and 2k for testing. LiTformer was trained in PyTorch on a GeForce RTX 4090 GPU.

We determined $\mathbf{D}_I$ and $\mathbf{D}_C$ by assessing the minimum and maximum values of intrinsic and crosstalk outputs for each TX dataset, to define the categorization range. $\mathbf{D}_I$ spans a 1.6V range for TS1, TS2, TP4, and 1.8V for TD3, each with $\Delta v^I = 1mV$. $\mathbf{D}_C$ ranges from -200 to 200mV with $\Delta v^C = 0.25mV$ for TS1, TS2, TP4, and -200 to 160mV with $\Delta v^C = 0.2mV$ for TD3. Including a <*mask*>, dictionary lengths are 1602 for TS1, TS2, TP4 and 1802 for TD3. We set d_{model} as 512 and trained the model with the Adam optimizer [40] with $\beta_1 = 0.9$, $\beta_2 = 0.98$, $\epsilon = 10^{-9}$, a learning rate of 0.0001, and a batch size of 16. To fully capture the 4-bit response of nonlinear TXs, we added a 1-bit tail for TS1, TS2, TP4, and a 3-bit tail for the more nonlinear TD3. The output sequence for all TXs was set to 501 points.

7.2 Performance Evaluation

LiTformer was trained on the TS1, TS2, TD3, and TP4 datasets for 1280, 1160, 1240, and 1280 epochs respectively, taking about 16 hours. Training and testing CE losses are detailed in Table 2 columns 2-3. LiTformer could accurately predict intrinsic outputs and crosstalk considering various link parameters, with performance assessed by CE losses for class sequences and mean absolute and relative errors (AE/RE) of filtered reconstructed voltage signals shown in Table 2

Table 3: Accuracy and runtime comparison for TX outputs in a 2-link and 16-link system between our LiTformer and SPICE.

TX	2-link system				16-link system			
	Accuracy evaluation	Runtime comparison			Accuracy evaluation	Runtime comparison		
	Mean AE/RE	LiTformer	SPICE	Speedup	Mean AE/RE	LiTformer	SPICE	Speedup
TS1	5.95mV/0.75%	6.51ms	1.28s	197x	7.67mV/0.97%	20.3ms	8.23s	405x
TS2	5.55mV/0.78%	6.21ms	2.41s	388x	4.98mV/0.68%	20.0ms	13.1s	655x
TD3	7.04mV/1.25%	9.21ms	1.36s	148x	6.63mV/1.18%	39.4ms	15.9s	404x
TP4	5.75mV/0.96%	7.23ms	3.30s	456x	6.50mV/1.07%	21.4ms	20.2s	944x

Table 4: Accuracy and runtime comparison of 1-link TX outputs among: our LiTformer, FNN [13], LSTM [29] and SPICE with link parameters as model inputs; our LiTformer, LSTM [29] and SPICE without link parameters as inputs.

TX	with link parameters						without link parameters			
	Mean AE/RE (mV/%)			Runtime Comparison			Mean AE/RE (mV/%)		Runtime Comparison	
	LiTformer	FNN [13]	LSTM [29]	**LiTformer**	FNN [13]	LSTM [29]	**LiTformer**	LSTM [29]	**LiTformer**	LSTM [29]
TS1	4.20/0.53	21.0/2.78	26.0/3.50	6.89ms	0.48ms	9.37ms	1.81/0.26	18.5/2.76	4.42ms	8.65ms
TS2	2.81/ 0.38	30.4/4.11	33.9/4.64	6.00ms	0.38ms	16.5ms	0.86/0.11	38.2/4.98	4.78ms	13.8ms
TD3	5.32/0.94	70.9/12.7	85.9/15.4	5.86ms	0.26ms	21.7ms	1.36/0.25	67.9/12.2	4.64ms	10.5ms
TP4	4.49/0.77	42.2/7.30	57.9/10.4	6.65ms	0.31ms	21.9ms	3.28/0.56	28.0/5.04	4.97ms	9.17ms

columns 4-7. RE is the AE ratio to the output amplitude. LiTformer achieves less than 1.26% mean RE for intrinsic output prediction, establishing a robust foundation for overall TX output estimation. Though it reports crosstalk RE of 1.90-3.32%, the corresponding AE is only around 0.3-0.5mV indicating minor amplitudes. Waveform comparisons with SPICE for TS2 and TP4 shown in Figure 9 reveal close matches for both intrinsic outputs and crosstalk.

To evaluate LiTformer's effectiveness of estimating TX interfered outputs in multiple links, we tested our LiTformer on 1000 simulated samples of 2-link and 16-link TX outputs (4 and 32 lines for TD3). We calculated the interfered output of TX_i by adding the predicted intrinsic output with $K = 0$ and the crosstalk pairing $link_i$ with each of the other links respectively with $K = 1$. Accuracy and runtime comparisons are presented in Table 3. Thanks to the accurate prediction of intrinsic outputs and crosstalk, LiTformer achieves mean REs of 0.75-1.25% for 2-link TX outputs and 0.68-1.18% for 16-link. By calculating intrinsic output and crosstalk components in one batch, LiTformer achieves inference time of 6-10ms with 148-456× speedup over SPICE for 2-link TXs and 20-40ms with 404-944× speedup for 16-link due to its parallel decoding capability, with SPICE's runtime growing drastically with more links. Figure 10 illustrates the 2-link and 16-link TX output waveform comparisons to SPICE, with crosstalk components highlighted, indicating LiTformer's effectiveness in modeling TX dynamics in multiple links.

7.3 Comparison with Related Work

We evaluated our *LiTformer* against FNN [13] and LSTM [29] for single-link TX modeling[1]. For single-link comparison, we removed LiTformer's encoder module of K and input into CNNs the full S-parameters sized 2×2 for single-ended TXs, and S-parameters with effective elements sized 2×5 for TD3. The FNN took in the input signal sequence of a memory length, vector fitting parameters of $\boldsymbol{S}_{tl}$ and H_0, C_L, Z_0, V_p to produce the output signal [13]. To incorporate link parameters, we combined the LSTM output [29] with them through an additional 4-layer FNN yielding the final output voltage. To specifically evaluate LiTformer's strength in modeling sequences with long-range dependencies, we also compared a further simplified LiTformer to LSTM [29] which both focused solely on input-output signal relationships without any link parameters.

Accuracy and runtime comparisons for the two cases are detailed in Table 4. Despite 10-hour training time, our LiTformer demonstrates significantly higher accuracy over FNN [13] and LSTM [29], benefiting from its ability to effectively handle non-sequential parameters and capture long-range dependencies of the output sequence. The parallel decoding of LiTFormer also allows for a speed advantage over the recurrent LSTM [29]. Though both FNN [13] and LSTM [29] employ complex architectures to capture nonlinearities — with the FNN [13] having 37-40 hidden units and LSTM [29] up to 38 blocks and both input memory lengths set to around 500 — their performance on highly nonlinear signals at Gbps is still not optimal, even after extensive training. Figure 11 (a) and (b) visually compare waveforms in the two cases, underscoring our LiTformer's advantage over FNN [13] and LSTM [29] in modeling high-speed link TXs with high nonlinearity.

8 Conclusions

This work introduces LiTformer, an innovative Transformer-based model for high-speed link TXs, with a non-sequential encoder and a Transformer decoder accounting for link effects including crosstalk and long-range dependencies of signals. We employ an NAR mechanism in training and inference for fast prediction. Experiments show that in comparison to SPICE, the proposed LiTformer achieves a prominent speed with 148-456× speedup for 2-link TXs and 404-944× speedup for 16-link TXs while ensuring minimal error margins of 0.68-1.25%, supporting 4-bit signals at Gbps data rates.

Acknowledgments

This work was supported in part by NSFC (Grant No. 62141404, and 62034007) and Major Program of National Natural Science Foundation of Zhejiang Province of China (Grant No. D24F040002).

[1]The FNN in [13] and LSTM model in [29] were not designed for multi-link TX modeling and hence single link is included for comparison.

References

[1] R. Muralidhar *et al.*, "Energy efficient computing systems: Architectures, abstractions and modeling to techniques and standards," *ACM Computing Surveys (CSUR)*, vol. 54, no. 11s, pp. 1–37, 2022.

[2] D. Shi *et al.*, "Toward energy-efficient federated learning over 5g+ mobile devices," *IEEE Wireless Communications*, vol. 29, no. 5, pp. 44–51, 2022.

[3] C. Zhuo *et al.*, "A fast method to estimate through-bump current for power delivery verification," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 42, no. 5, pp. 1643–1647, 2023.

[4] S. Sun *et al.*, "An approximating twiddle factor coefficient based multiplier for fixed-point fft," in *2022 China Semiconductor Technology International Conference (CSTIC)*, 2022, pp. 1–4.

[5] X. Dong *et al.*, "Spiral: Signal-power integrity co-analysis for high-speed interchiplet serial links validation," in *2024 29th Asia and South Pacific Design Automation Conference (ASP-DAC)*, 2024, pp. 625–630.

[6] D. D. Sharma *et al.*, "Universal chiplet interconnect express (ucie): An open industry standard for innovations with chiplets at package level," *IEEE Transactions on Components, Packaging and Manufacturing Technology (TCPMT)*, vol. 12, no. 9, pp. 1423–1431, 2022.

[7] P.-W. Chiu, "Digital intensive transceivers for high-speed serial links," Ph.D. dissertation, University of Minnesota, 2019.

[8] T. Li *et al.*, "Chiplet heterogeneous integration technology—status and challenges," *Electronics*, vol. 9, no. 4, p. 670, 2020.

[9] C.-T. Wang *et al.*, "Signal integrity of submicron info heterogeneous integration for high performance computing applications," in *2019 IEEE 69th Electronic Components and Technology Conference (ECTC)*, 2019, pp. 688–694.

[10] V. Stojanovic *et al.*, "Modeling and analysis of high-speed links," in *Proceedings of the IEEE 2003 Custom Integrated Circuits Conference, 2003.*, 2003, pp. 589–594.

[11] J. Fan *et al.*, "Signal integrity design for high-speed digital circuits: Progress and directions," *IEEE Transactions on Electromagnetic Compatibility*, vol. 52, no. 2, pp. 392–400, 2010.

[12] K. L. Fong *et al.*, "High-frequency nonlinearity analysis of common-emitter and differential-pair transconductance stages," *IEEE Journal Solid-State Circuits*, vol. 33, no. 4, pp. 548–555, 1998.

[13] Y. Zhao *et al.*, "Modular neural network based models of high-speed link transceivers," *IEEE Transactions on Components, Packaging and Manufacturing Technology*, 2023.

[14] D. Jiao *et al.*, "Fast method for an accurate and efficient nonlinear signaling analysis," *IEEE Transactions on Electromagnetic Compatibility*, vol. 59, no. 4, pp. 1312–1319, 2017.

[15] K. Hu *et al.*, "A comparative study of 20-gb/s nrz and duobinary signaling using statistical analysis," *IEEE transactions on very large scale integration (VLSI) systems*, vol. 20, no. 7, pp. 1336–1341, 2011.

[16] H. Kim *et al.*, "Eye-diagram simulation and analysis of a high-speed tsv-based channel," in *2013 IEEE International 3D Systems Integration Conference (3DIC)*. IEEE, 2013, pp. 1–7.

[17] R. Shi *et al.*, "Efficient and accurate eye diagram prediction for high speed signaling," in *2008 IEEE/ACM International Conference on Computer-Aided Design*. IEEE, 2008, pp. 655–661.

[18] J. Park *et al.*, "A novel stochastic model-based eye-diagram estimation method for 8b/10b and tmds-encoded high-speed channels," *IEEE Transactions on Electromagnetic Compatibility*, vol. 60, no. 5, pp. 1510–1519, 2017.

[19] X. Chu *et al.*, "Statistical eye diagram analysis based on double-edge responses for coding buses," *IEEE Transactions on Electromagnetic Compatibility*, vol. 62, no. 3, pp. 902–913, 2019.

[20] "Hspice," https://www.synopsys.com/.

[21] "Ibis version 7.2," https://ibis.org/ver7.2/ver7_2.pdf, 2023.

[22] C. Amin *et al.*, "A multi-port current source model for multiple-input switching effects in cmos library cells," in *ACM/IEEE Design Automation Conference (DAC)*, 2006, pp. 247–252.

[23] C. Kashyap *et al.*, "A nonlinear cell macromodel for digital applications," in *IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, 2007, pp. 678–685.

[24] Y. Zhao *et al.*, "Modeling cascade-able transceiver blocks with neural network for high speed link simulation," in *2022 IEEE Electrical Design of Advanced Packaging and Systems (EDAPS)*. IEEE, 2022, pp. 1–3.

[25] Z. Naghibi *et al.*, "Time-domain modeling of nonlinear circuits using deep recurrent neural network technique," *AEU-International Journal of Electronics and Communications*, vol. 100, pp. 66–74, 2019.

[26] J. Xu *et al.*, "Neural-based dynamic modeling of nonlinear microwave circuits," *IEEE Transactions on Microwave Theory and Techniques*, vol. 50, no. 12, pp. 2769–2780, 2002.

[27] Z. Naghibi *et al.*, "Dynamic behavioral modeling of nonlinear circuits using a novel recurrent neural network technique," *International Journal of Circuit Theory and Applications*, vol. 47, 04 2019.

[28] M. Noohi *et al.*, "Modeling and implementation of nonlinear boost converter using local feedback deep recurrent neural network for voltage balancing in energy harvesting applications," *International Journal of Circuit Theory and Applications*, vol. 49, pp. 4231 – 4247, 2021.

[29] M. Moradi A. *et al.*, "Long short-term memory neural networks for modeling nonlinear electronic components," *IEEE Transactions on Components, Packaging and Manufacturing Technology (TCPMT)*, vol. 11, no. 5, pp. 840–847, 2021.

[30] A. Faraji *et al.*, "Batch-normalized deep recurrent neural network for high-speed nonlinear circuit macromodeling," *IEEE Transactions on Microwave Theory and Techniques*, vol. 70, no. 11, pp. 4857–4868, 2022.

[31] A. Vaswani *et al.*, "Attention is all you need," *Advances in neural information processing systems*, vol. 30, 2017.

[32] J. Gehring *et al.*, "Convolutional sequence to sequence learning," in *International conference on machine learning*. PMLR, 2017, pp. 1243–1252.

[33] L. Li *et al.*, "Long-term prediction for temporal propagation of seasonal influenza using transformer-based model," *Journal of biomedical informatics*, vol. 122, p. 103894, 2021.

[34] G. Zerveas *et al.*, "A transformer-based framework for multivariate time series representation learning," in *Proceedings of the 27th ACM SIGKDD conference on knowledge discovery & data mining*, 2021, pp. 2114–2124.

[35] J. Lee *et al.*, "Deterministic non-autoregressive neural sequence modeling by iterative refinement," in *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, 2018, pp. 1173–1182.

[36] M. Ghazvininejad *et al.*, "Mask-predict: Parallel decoding of conditional masked language models," in *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*. Association for Computational Linguistics, 2019.

[37] N. Chen *et al.*, "Non-autoregressive transformer for speech recognition," *IEEE Signal Processing Letters*, vol. 28, pp. 121–125, 2020.

[38] F. Huang *et al.*, "Directed acyclic transformer for non-autoregressive machine translation," in *International Conference on Machine Learning*. PMLR, 2022, pp. 9410–9428.

[39] A. Savitzky *et al.*, "Smoothing and differentiation of data by simplified least squares procedures." *Analytical chemistry*, vol. 36, no. 8, pp. 1627–1639, 1964.

[40] D. P. Kingma *et al.*, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.