Scalable High-Level Synthesis Method for Parallel Data Reading in Cold Data Storage Optical Discs

Authors: Luo Ke, Jian Yugen, Hongyu Gao, Lu Ping, Chen Jincai, Chen Jincai

Date: 2025-12-09T00:00:00+00:00

Abstract

High-density optical disc storage technology, as a crucial component of cold data storage, demonstrates significant advantages within the context of new quality productive forces. Its green energy-saving, ultra-long-term preservation, and low-cost characteristics offer efficient solutions for massive cold data management. However, with the explosive growth of data volume and the rapid advancement of emerging storage technologies, conventional optical disc readback signal processing methods face severe challenges regarding speed, noise suppression, and resource efficiency. This paper addresses the requirements of new quality productive forces for high-performance, intelligent, and green data storage, and proposes two innovative parallel optical disc readback signal processing methods based on high-level synthesis (HLS) technology. Firstly, a parallel partial response maximum likelihood (PRML) signal processing method based on HLS is designed, which enhances signal processing speed by approximately 2.99 times through pipeline optimization, dataflow, and parallel computing strategies, thereby significantly improving data access efficiency. Secondly, a parallel neural network maximum likelihood (NNML) signal processing method based on HLS is proposed, which effectively suppresses nonlinear noise interference through a neural network equalizer, reducing the average bit error rate by approximately 30% and enhancing signal processing quality. Experimental results indicate that the parallel PRML method achieves a processing speed of 144.50 Mb/s, while the parallel NNML method delivers superior bit error rate performance, with each approach offering distinct advantages in resource consumption and performance. This study not only provides novel insights for the high-performance and intelligent evolution of optical disc storage technology, but also serves as an important reference for the innovation and development of green data storage technologies under the new quality productive forces framework.

Full Text

Preamble

A Scalable High-Level Synthesis-Based Parallel Data Readout Method for Optical Discs in Cold Data Storage

Ke Luo[13], Yugen Jian[13], Hongyu Gao[13], Ping Lu[23], Jincai Chen[13]
[1]Wuhan National Laboratory for Optoelectronics, Huazhong University of Science and Technology, Wuhan 430074, China
[2]School of Computer Science and Technology, Huazhong University of Science and Technology, Wuhan 430074, China
[3]Key Laboratory of Information Storage System, Ministry of Education of China, Huazhong University of Science and Technology, Wuhan 430074, China

Abstract

High-density optical disc storage technology, as a critical component of cold data storage, offers significant advantages in the context of new quality productive forces, including energy efficiency, ultra-long-term preservation, and low cost, making it an ideal solution for managing massive cold data. However, with the exponential growth of data volume and the rapid development of emerging storage technologies, traditional optical disc readback signal processing methods face significant challenges in speed, noise suppression, and resource utilization. Addressing the demands of new quality productive forces for high-performance, intelligent, and green data storage, this paper proposes two innovative parallel optical disc readback signal processing methods based on high-level synthesis (HLS). First, a parallel partial response maximum likelihood (PRML) signal processing method is designed, leveraging pipeline optimization, data streaming, and parallel computing strategies to achieve a 2.99× speedup, significantly enhancing data access efficiency. Second, a parallel neural network maximum likelihood (NNML) signal processing method is introduced, which employs a neural network equalizer to effectively suppress nonlinear noise interference, reducing the average bit error rate by approximately 30% and improving signal processing quality. Experimental results demonstrate that the parallel PRML method achieves a processing speed of 144.50 Mb/s, while the parallel NNML method exhibits superior bit error rate performance, each excelling in resource consumption and performance metrics. This research not only provides novel insights into the high-performance and intelligent evolution of optical disc storage technology but also contributes to the innovation and development of green data storage technologies in the era of new quality productive forces.

Keywords: Optical disc storage; high-level synthesis; data readout; signal detection; neural networks; parallel processing

1. Introduction

1.1 High-Level Synthesis Technology

Since 2011, FPGA high-level synthesis technology has made significant progress. Xilinx, a leading global FPGA vendor, acquired the startup AutoESL Design Technologies Inc., whose HLS tool AutoPilot was the first HLS tool based on the emerging low-level virtual machine (LLVM) compiler infrastructure and has been widely applied in FPGA engineering practice. Cong et al. proposed advanced HLS solutions for multiple application domains based on the AutoPilot tool and Xilinx' s domain-specific system-level experimental platform, marking the beginning of HLS technology adoption in practical FPGA implementations. The HLS synthesis process employs finite state machine principles to perform logical analysis of algorithms implemented in high-level languages, followed by hardware resource allocation to generate hardware circuits that implement the algorithm.

The scheduling method plays a central role in the synthesis process. Cong et al. proposed a scheduling approach that transforms scheduling constraints into a system of difference constraints, enabling various optimization strategies. Experimental results demonstrated that this method effectively supports constraints on resources, frequency, latency, and relative timing while optimizing the longest path delay, expected total delay, and slack distribution. Zhang et al. further advanced a scheduling framework based on differential constraint systems that minimizes resource consumption in streaming synthesis, thereby improving streaming synthesis speed.

FPGAs exhibit significant advantages in streaming data processing. HLS provides convenient streaming processing directives, enabling developers to implement FPGA hardware circuits without concerning themselves with RTL-level streaming details, and has been widely applied across various data processing domains. In machine learning, Sun et al. proposed a general framework called FILM-QNN for quantizing and accelerating multiple deep neural network models on different embedded FPGA devices. This framework employs an intra-layer mixed-precision quantization algorithm combined with various FPGA architecture optimization techniques such as DSP packing, weight reordering, and data packing to enhance overall throughput. Additionally, Hao et al. proposed an automatic co-design methodology that utilizes HLS tools to generate synthesizable C code for deep neural network FPGA accelerator design. Beyond neural network acceleration, researchers have investigated other aspects of HLS tools. Wong et al. proposed DONGLE, an HLS storage interface that enables HLS programs to adapt to multiple storage devices, thereby improving code library clarity and development efficiency. Xu et al. proposed a scalable linear programming-based strategy that eliminates false stalls during HLS synthesis of dataflow circuits, saving resource overhead during synthesis. Guo et al. presented a parallelization compilation framework using C/C++ HLS dataflow programs capable of generating complete compiled module layouts. Chi et al. pro-

posed an efficient HLS-based method for implementing single-source shortest paths on large-scale power-law graphs. Fibich et al. evaluated and discussed the final hardware implementation performance of two freely available high-level synthesis tools through four case studies.

In recent years, HLS technology has continued to evolve across multiple domains. Brignone et al. proposed the SILVIA framework, which utilizes HLS-specific LLVM transformation passes to automatically identify and exploit superword-level parallelism, packing multiple operations into a single DSP and significantly reducing resource utilization. He et al. developed the HLSTransform method, achieving energy-efficient inference of the Llama 2 model on FPGA using HLS, with speedups of 2.46× and 0.53× compared to CPU and GPU respectively, while significantly reducing energy consumption. Zhao et al. proposed the GNNHLS framework for evaluating graph neural network inference performance on FPGA, achieving maximum speedups of 50.8× and 5.16× compared to CPU and GPU baselines respectively. Lau et al. introduced the RapidStream IR infrastructure, supporting pipelining of real designs at arbitrary levels and integrating HLS modules, vendor IPs, and handcrafted RTL designs to improve design frequency and portability. Szafarczyk et al. studied dynamic loop fusion techniques, proposing methods for implementing dynamic loop fusion in HLS that effectively improve resource utilization and performance. While numerous studies have accelerated neural network models using FPGAs, the core contribution of this paper is not neural network model acceleration per se, but rather proposing a system implementation scheme that differs from existing PRML sequence detection techniques by incorporating neural networks. This work represents the first systematic application of HLS to optical disc PRML signal processing, proposing a modular, parallel, and scalable hardware architecture that complements rather than duplicates existing work.

Currently, various novel optical storage technologies remain in the research and exploration phase, lacking consideration of hardware implementation complexity and cost. Existing commercial optical discs employ chip-packaged read/write control chips, with key technologies lacking transparency, which hinders the development of more compatible read/write technologies. Based on the mature PRML detection algorithm from magnetic storage and optical disc data readout processes, this paper employs flexible HLS tools for hardware implementation of PRML detection algorithms while exploring the feasibility of applying neural networks in edge devices combined with PRML methods, evaluating their detection effectiveness and hardware overhead. This research provides a scalable high-level synthesis-based parallel data readout solution for various optical storage systems, offering significant reference value.

1.2 Optical Disc Readback Signal Processing Technology

In optical disc storage systems, existing PRML detection flow primarily includes two key modules: an equalizer and a detector. The former suppresses and eliminates inter-symbol interference in readback signals, while the latter detects

and recovers written data from the equalized signals. To improve readback signal quality and reading reliability, researchers have proposed various crosstalk cancellation methods and detection algorithms.

For instance, Kim et al. proposed an improved decision feedback equalizer to reduce crosstalk between information bits and validated the method' s advantages in processing speed and bit error rate through simulation analysis. Saito et al. proposed an equalization method using multi-beam readout to eliminate adjacent track crosstalk, compensating signals through multiple optical pickup heads to improve signal quality. Koonkarnkhai et al. proposed a method combining two Turbo decoding iterations to compensate for crosstalk noise in readback signals, reducing noise impact. Nakthanom et al. proposed a multilayer perceptron and minimum mean square error-based equalizer that utilizes multilayer perceptrons to process nonlinear signal interference, effectively reducing inter-symbol crosstalk in readback signals. Miyashita et al. early proposed an adaptive equalizer implemented using the least mean square (LMS) algorithm and applied it to the PRML flow, significantly reducing system bit error rate. Reddy et al. proposed a high-speed feedforward Viterbi detector employing traceback architecture and utilizing FPGA hardware resources for acceleration. Experimental results showed this decoder effectively reduces decoding latency. Mozaffari et al. proposed a method incorporating resource overhead clipping in Viterbi algorithm hardware implementation to further reduce latency.

In recent years, optical disc data readout technology has made significant progress in high-resolution data writing and signal acquisition. The Shanghai Advanced Research Institute of the Chinese Academy of Sciences proposed a sub-diffraction-limit optical storage readout method based on polarization modulation. This method achieves sub-resolution data readout with 500 nm spacing by controlling the polarization characteristics of reflected signals, with readout speeds approximately 15× faster than traditional fluorescence methods. However, this method still faces challenges, such as dependence on material polarization modulation characteristics that may limit its applicability across different material systems.

An important advancement in optical disc data writing is super-resolution three-dimensional optical storage technology. The Shanghai Institute of Optics and Fine Mechanics of the Chinese Academy of Sciences, in collaboration with Shanghai University of Science and Technology, developed dual-beam regulated aggregation-induced emission storage technology, achieving super-resolution data storage with a spot size of 54 nm and track pitch of 70 nm, and completed 100-layer multilayer recording with a single-disc equivalent capacity reaching 1.6 Pb, breaking the optical diffraction limit. Despite breakthroughs in storage density, this technology still faces practical application issues, such as potential signal interference and increased bit error rates during readout due to multilayer recording, and the long-term stability and durability of aggregation-induced emission materials require further verification. Additionally, the glass-ceramic storage technology developed by German startup Cerabyte demonstrates

exceptional durability in extreme environments. This technology employs laser etching of nanoscale ceramic layers on ultra-thin glass chips, resisting harsh conditions such as high temperatures and saltwater corrosion while maintaining data integrity, making it suitable for long-term cold data preservation. However, this technology remains in early stages, with relatively low storage density and limited write speeds, and faces challenges in large-scale production and cost control.

Microsoft' s Project Silica utilizes femtosecond lasers to inscribe three-dimensional nanostructures in quartz glass, achieving high-density storage of approximately 7 TB per glass plate. This technology permanently stores data by altering the internal structure of glass, offering extremely high durability and stability. However, Project Silica still faces challenges. Since data is stored in digital form without applicable equalization and detection processes, error correction remains necessary to prevent file corruption from bit errors. Additionally, complex readout technology must be preserved long-term along with the glass medium, including decoding, optical readers, and machine learning algorithms. These dependencies may create difficulties in future data access. Shanghai Jiao Tong University proposed a deep learning-based error-free long-lifespan optical storage scheme that uses femtosecond lasers to write nanostructures in fused silica and deep neural networks for high-precision readout. However, its high technical complexity prevents comparison with commercial optical disc technology, and it has not resolved mass production and cost issues, making it less practical than current optical disc detection technologies.

In summary, optical disc data readout technology continues to evolve in signal acquisition and detection, from traditional algorithm optimization to novel technology introduction, aiming to improve storage density and signal quality, reduce bit error rates, and control hardware resource consumption. Future integration with HLS technology and new methods promises to further enhance optical disc storage system performance and efficiency. Despite extensive research on PRML signal processing flows, neural network equalizer design, and HLS tool optimization, several major challenges remain. First, traditional PRML signal processing methods primarily address linear channel noise, with limited equalization effectiveness and bit error rate performance when facing complex interference from high-density optical disc recording, such as inter-symbol interference and nonlinear distortion. Second, while neural networks have been introduced to improve equalization capability, issues such as high hardware implementation complexity, large resource consumption, and poor deployment flexibility remain prominent. Particularly in FPGA resource-constrained scenarios, deep network structures are difficult to effectively map into high-performance, low-latency circuit architectures. Furthermore, existing research primarily focuses on algorithm-level performance improvements, lacking systematic discussion on modular design complexity, cross-module data synchronization bottlenecks, and the trade-off between fixed-point precision and bit error rate when applying HLS to large-scale data stream signal processing. Therefore, there is an urgent need

to propose a scalable optical disc readback signal processing framework that balances processing quality and resource efficiency, capable of adapting to practical application scenarios such as multi-module parallelism and high-density readout under cold data storage requirements.

To address these challenges, this paper proposes a scalable high-level synthesis-based parallel data readout method for cold data storage optical discs, utilizing HLS technology for efficient processing of optical disc readback signals. This method improves signal processing speed and reduces hardware resource consumption through streaming processing and parallelization design, offering good portability and scalability. Additionally, this paper combines neural network equalizers with Viterbi detectors to further improve readback signal quality and reduce bit error rates. Compared with existing technologies, the proposed method demonstrates clear advantages in system complexity, compatibility, and manufacturing cost, providing a new solution for efficient and reliable optical disc readback signal processing.

2 HLS-Based Parallel PRML Processing Method

Existing optical disc PRML signal detection systems primarily consist of an equalizer module and a detector module. The equalizer module denoises optical disc readback signals to significantly improve signal quality, while the detector module recovers signals from the equalized readback signals. The PRML signal processing flow for BDXL Blu-ray discs is illustrated in Figure 1 [Figure 1: see original paper]. The readback signal from the optical pickup head undergoes adaptive equalization for signal shaping, making the readback signal quality approach the ideal channel signal. The equalized signal then passes through a Viterbi detector to obtain the recovered write sequence. The recovered write sequence is convolved with the partial response target of BDXL discs to generate the ideal channel signal. The difference between the ideal channel signal and the readback signal yields the error e, which is then fed back to the adaptive equalizer for coefficient updates.

2.1 HLS-Based Adaptive Equalizer Implementation

The implementation of an optical disc adaptive equalizer can be divided into two parts: signal equalization and equalizer tap coefficient updates. For signal equalization, the optical disc equalizer belongs to the class of finite impulse response filters, operating on the principle of convolving a signal with a set of coefficients to make the signal conform to the ideal channel signal. For equalizer tap coefficient updates, adaptive updating requires an ideal signal as the target.

The HLS-based adaptive equalizer module decomposition is shown in Figure 2 [Figure 2: see original paper]. The equalizer module performs convolution on readback signals using a set of tap coefficients, while the equalized signal amplitude mapping module quantizes the amplitude of equalized signals to match the amplitude level of Viterbi detector state transitions. The ideal signal con-

volution module receives the write sequence recovered by the Viterbi detector and convolves it with the partial response target to obtain the ideal signal. The ideal signal amplitude mapping module functions similarly to the equalized signal amplitude mapping module, scaling the ideal signal amplitude to the same level as the readback signal. After receiving the readback signal and mapped ideal signal, the equalizer tap coefficient update module updates the equalizer coefficients and passes the updated coefficients back to the equalizer module.

After implementing each module's functionality through HLS, they are deployed into FPGA via IP core generation. Each module receives a clock signal, and every clock cycle, all modules execute one data computation in parallel. Therefore, in HLS implementation, decomposing an overall module into sub-modules where each sub-module performs only one type of data computation can significantly improve the overall module' s computational parallelism, thereby enhancing system operating speed.

During the convolution computation in the equalization module, a total of 13 multiplications and additions are performed. Since these multiply-accumulate operations belong to the same operation type, they can be executed in parallel simultaneously. Block Random Access Memory (BRAM) in FPGAs can store data, but BRAM typically has at most two read ports, limiting readout rate if an array is stored in a single BRAM. The HLS ARRAY_{PARTITION} directive maps array data to different BRAMs, enabling simultaneous data reading from the same array.

As shown in Figure 3 [Figure 3: see original paper], using BDXL discs as an example, the 13 tap coefficient arrays of the equalizer are mapped into different BRAMs. The array temporarily storing readback signals also has a length of 13 and is mapped to different BRAMs. Since both arrays are mapped to different BRAMs, the 13 multiplications can execute in parallel. After multiplication completes, the results are placed into array t, which is also mapped to different BRAMs. During addition operations, values can be fetched in parallel and then summed sequentially to finally obtain the equalized signal.

From Figure 3, the array storing readback signals has the same length as the tap coefficient array. However, the entire readback signal sequence is far longer than the tap coefficient array length. To avoid storing all readback signals, a shift register approach is proposed for readback signal storage. As shown in Figure 4 [Figure 4: see original paper], the temporary array is initially zero-valued. When the equalizer module begins computation, it fetches the first readback signal value r_1 from the FIFO queue, performs parallel multiply-accumulate operations to obtain the first equalized signal y_1, then shifts all values in the temporary array rightward, and fetches the readback signal r_2. The same computation yields y_2. When retrieving readback signal r_{13}, it overwrites r_1, and finally computes y_{13}. Since the temporary array is mapped to different BRAMs, the rightward shift and overwrite process of all values in the temporary array also executes in parallel. This shift register approach eliminates the need to store all readback signals, and because the readback signal overwrite process is

parallel, the hardware resource consumption and latency of the equalizer module implemented this way are significantly reduced.

After computing the equalized signal sequence $\vec{y}$, it must pass through the equalized signal amplitude mapping module to obtain the mapped signal sequence $\vec{y}'$, which scales the equalized signal amplitude to the same level as Viterbi detector state transition amplitudes. Using BDXL as an example, when the partial response target is $[1, 2, 2, 2, 1]$, the state transition amplitude reaches its maximum during the transition from state $b'1111$ to $b'1111$, receiving a bit value of 1. Therefore, the state transition amplitude is the convolution of sequence $[1, 1, 1, 1, 1]$ with $[1, 2, 2, 2, 1]$, yielding a maximum state transition amplitude of 8. Similarly, the minimum state transition amplitude is 0, corresponding to the transition from state $b'0000$ to $b'0000$. The equalized signal amplitude mapping module must employ normalized computation to scale the $\vec{y}$ sequence amplitude between 0 and 8. The computation from $\vec{y}$ sequence to $\vec{y}'$ sequence is shown in Equation (1):

$$\vec{y}' = \frac{\vec{y} - \min(\vec{y})}{\max(\vec{y}) - \min(\vec{y})} \times 8$$

From Equation (1), computing $\vec{y}'$ sequence requires first obtaining the maximum and minimum values of the $\vec{y}$ sequence. Therefore, the equalized signal amplitude mapping module needs to store a certain number of equalized signals.

The equalizer tap coefficient update module requires the error between the ideal signal and readback signal to update coefficients. The readback signal can be stored in BRAM within the equalizer tap coefficient module, while the ideal signal must be calculated through the ideal signal convolution module and ideal signal amplitude mapping module. The ideal signal convolution module implementation mirrors the equalizer module, with the i-th ideal signal computed as shown in Equation (2):

$$g_i = \sum_{j=1}^{13} c_j \times PR_j$$

where PR_j represents the j-th partial response target value. The ideal signal amplitude mapping module implementation follows the same principle as the equalized signal amplitude mapping module but scales within the range of -1 to 1, as shown in Equation (3):

$$\vec{g}'' = \frac{\vec{g} - \min(\vec{g})}{\max(\vec{g}) - \min(\vec{g})} \times 2 - 1$$

After computing the ideal signal, the error e is calculated as shown in Equation (4):

$$e = y - g$$

The equalizer tap coefficient update module updates coefficients using the error between the ideal signal and readback signal. The coefficient update formula is shown in Equation (5):

$$c_{i+1,j} = c_{i,j} - \mu \times \text{sign}(e_i) \times r_{i-j+7}$$

where $c_{i+1,j}$ represents the j-th tap coefficient during the $(i+1)$-th update, and sign is the sign function.

2.2 HLS-Based Viterbi Detector Implementation

As shown in Figure 5 [Figure 5: see original paper], the Viterbi detector is decomposed into three modules: forward iteration computation module, traceback module, and detection sequence reversal module, connected via FIFO queues. This FIFO connection enables data streaming between modules, reducing overall Viterbi detector latency. To better represent the Viterbi detector module decomposition, Figure 5 includes the equalization module (comprising the equalizer module and equalized signal amplitude mapping module) and the equalization coefficient update module (comprising the ideal signal convolution module, ideal signal amplitude mapping module, and equalizer tap coefficient update module).

The HLS-based forward iteration computation module is shown in Figure 6 [Figure 6: see original paper]. In the HLS implementation, each state' s transition direction and transition amplitude are placed in separate arrays, with the ARRAY_{PARTITION} directive mapping both arrays'values to different BRAMs. To facilitate traceback algorithm operation, the previous moment' s state must be recorded during forward computation. To better implement this in HLS, the state transition computation method in Viterbi forward calculation is modified. As shown in Figure 6, the yellow dashed boxes represent Viterbi algorithm computation unit storage patterns, with upper boxes indicating current states and lower boxes indicating transition directions. Using state $b'0000$ in the green box as an example, the current moment state $b'0000$ can transition from the previous moment state $b'0000$ receiving a bit 0, or from the previous moment state $b'1000$ receiving a bit 0. During computation, the previous moment state $b'0000$ state metric computation unit receives the equalized signal and calculates the state transition amplitude $SM(0,0)$ from state $b'0000$ to $b'0000$. The previous moment state $b'1000$ state metric computation unit receives the equalized signal and calculates the state transition amplitude $SM(0,4)$ from state $b'1000$ to $b'0000$. The corresponding path metrics PM_0 and PM_8 are fetched from path metric storage BRAM and added to obtain two source direction transition amplitudes for the current moment state $b'0000$, denoted as a and b. If a is less than b, the current moment state $b'0000$ transitioned from the previous moment

state $b'0000$, and its transition direction is recorded as 0. Other states follow the same forward computation method as state $b'0000$.

The traceback module receives traceback information from the forward computation module. If the forward computation module receives k equalized signals, to recover the corresponding write sequence, the traceback module must first store k traceback information outputs from the forward computation module, then select any valid state as the starting state, execute the traceback algorithm based on its traceback information, and finally obtain k write sequences. As shown in Figure 7 [Figure 7: see original paper], when each state' s forward computation completes, the forward computation unit places the previous moment' s transition direction information into the corresponding bit position. For example, after completing forward computation for state $b'1111$, it is placed in the bit_{15} position (the most significant bit of traceback information), while state $b'0000$' s previous moment transition direction information is placed in the bit_0 position (the least significant bit). The traceback information placement is accomplished through shifting: bit_{15} is shifted left by 15 bits to the most significant bit position, while bit_0 requires no shifting to occupy the least significant bit position. To facilitate this shift-based traceback information composition, the actual HLS implementation uses 16-bit traceback information instead of 10 bits, enabling state direction transition information to be shifted directly by the state' s decimal value, eliminating additional computation during traceback information composition. After all state transition direction information is placed, the traceback information is assembled and output to the traceback module by the forward computation module.

After traceback information enters the traceback module, since it is a 16-bit integer data composed of each state' s transition direction, the traceback module must employ indexing to execute the traceback algorithm. As shown in Figure 8 [Figure 8: see original paper], the index table has two layers: the upper layer is the index table for transition direction 0 with initial index 0, and the lower layer is the index table for transition direction 1 with initial index 0. To conveniently illustrate the traceback algorithm execution, an example is provided. When traceback information is parsed as 19 (binary representation 10011) and the traceback starting state is $b'0000$, the received transition direction is 1. The previous moment state of $b'0000$ must be found in the index table for transition direction 1. In the index table design, the index value corresponds exactly to the state value. Therefore, $b'0000$' s previous moment state is the value at index 0 in the transition direction 1 index table, which is 8 (binary $b'1000$). Similarly, when state $b'1000$ searches for its previous moment state, it obtains transition direction 1 from the traceback information and must search in the transition direction 1 index table. With an index value of 8, its previous moment state is 12 (binary $b'1100$). After traceback completes, the recovered write sequence consists of the least significant bits from each traceback state' s binary representation. For example, when traceback information is 19, the recovered write sequence is 001111.

After traceback module execution, since the recovered write sequence is obtained through traceback algorithm, the sequence is reversed. Therefore, the traceback module outputs the sequence to the detection sequence reversal module for data flipping. The detection sequence reversal module functions simply: it receives the write sequence and outputs it in reverse order. The design philosophy of the entire HLS-based Viterbi detector is dataflow execution between modules and pipelined execution within modules. This design determines that the more modules are decomposed, the lower the overall system latency, hence the detection sequence reversal module is also implemented as a separate module.

2.3 HLS-Based Parallel PRML Signal Processing Method Implementation

The adaptive equalizer module and Viterbi detector module combine to form the optical disc PRML signal processing system. In the HLS-based adaptive equalizer module and traceback Viterbi detector module design, to achieve low latency and low hardware resource consumption for the entire PRML system, a decomposition approach is implemented: large modules are broken into multiple small modules with dataflow between small modules and pipelined computation within each small module. Each small module performs only partial computation or logic judgment, with at most three loop statements in program implementation (specific data provided in subsequent sections). After decomposing large modules into small modules, since each small module can receive a clock signal, every clock cycle each small module executes one computation, and FIFO queues connect each pair of small modules, making their execution independent. When the PRML system begins execution, small modules can operate in parallel, thereby reducing overall system latency. Beyond module decomposition, computation within small modules employs pipelining, where some computation steps execute in parallel between two data computations, reducing latency of individual small modules and consequently the entire system.

Through decomposition into multiple small modules combined with internal pipelining, optical disc PRML system latency is significantly reduced, achieving one recovered write bit per clock cycle, which essentially reaches the computational limit of a single PRML module. To further improve optical disc readback signal processing speed, a parallel PRML processing method is proposed, employing multiple PRML modules executing in parallel. Optical disc readback signal processing lacks continuity; after meeting the minimum Viterbi forward computation depth, readback signals can be recovered in blocks with correctness essentially equivalent to full recovery. Therefore, while maintaining bit error rate, parallel execution of multiple PRML modules can further improve overall optical disc readback signal processing speed.

As shown in Figure 9 [Figure 9: see original paper], using three parallel PRML modules as an example, a readback signal distribution module is introduced before the parallel PRML computation modules. Signal distribution is achieved by outputting a specified number of readback signals: after outputting a des-

ignated number of readback signals to one PRML module, it switches to the next PRML module for readback signal output. The number of output readback signals must satisfy the minimum Viterbi algorithm forward computation depth, which varies with disc density—higher density requires larger values, lower density requires smaller values. After the Viterbi detector recovers the equalized signal output from the equalization module into a 0/1 sequence, the recovered write sequence merging module receives all recovered write sequences in a quantity-based manner. If the readback signal distribution module distributes k readback signals to each PRML module, the merging module must receive k recovered write sequences from each PRML module in the order of the distribution module' s output. Assuming the readback signal distribution module first outputs k readback signals to PRML module #1, then to modules #2 and #3, the merging module must receive k recovered write sequences from module #1, then from modules #2 and #3, ensuring the recovered write sequences remain ordered. This readback signal distribution and recovered sequence merging approach increases latency of the overall parallel PRML computation module, but the parallel PRML signal processing method makes this increased latency negligible, thereby improving overall signal processing speed.

The readback signal distribution module and recovered write sequence merging module are implemented through FIFO queue connections. As shown in Figure 10 [Figure 10: see original paper], using FIFO queues eliminates the need for control signal connections between the readback signal distribution module and each equalizer module. Direct connection would raise questions about when the distribution module should output signals to the equalizer module. Since FIFO queues provide a native data-empty signal interface, the equalizer module can determine via the FIFO queue interface whether data exists in the current FIFO queue. If no data, it waits for input; if data exists, it receives the data and begins equalization. Simultaneously, the distribution module need not determine whether the equalizer module has completed execution—it simply outputs signals to FIFO queues connected to each equalizer module in sequence, requiring only appropriate FIFO depth configuration based on actual system runtime needs to prevent blocking. In addition to the distribution module-equalizer module connection via FIFO queues, each Viterbi detector module and the recovered write sequence merging module also connect through FIFO queues. The implementation of equalizer modules, Viterbi detector modules, and equalization coefficient update modules follows the previously described methods.

3 HLS-Based Parallel NNML Processing Method

3.1 Neural Network-Based Equalization Method

In PRML signal processing, the equalizer aims to make readback signals approach the ideal signal without ISI, enabling subsequent detectors to recover the signal into a 0/1 sequence. As optical disc density and readback signal acquisition rates increase, adaptive equalization methods cannot effectively eliminate nonlinear noise interference in optical disc readback signals, degrading

signal processing quality. To improve equalizer processing quality, this section proposes a neural network equalizer constructed based on readback signal sample blocks and system characteristics affecting readback signals, implemented in hardware using HLS and accelerated through parallel computation methods. The neural network equalizer replaces the adaptive equalizer; therefore, aside from equalizer replacement, other processing steps remain unchanged.

The neural network equalizer training method is shown in Figure 11 [Figure 11: see original paper]. To collect training datasets, a random sequence generation function produces original 0/1 bit sequences through modulo operations. The original bit sequences undergo 17PP modulation encoding and NRZI transformation to obtain the original write sequence (recorded bits). To reduce training time, readback response coefficient sequences replace impulse response functions in readback signal generation, convolving recorded bits with readback response coefficient sequences to generate readback signals, then adding noise to better approximate real physical signals. After the entire recorded bit sequence is convolved with the readback response coefficient sequence and noise is added to obtain the readback signal sequence, the readback signal sequence is fed into the neural network for training, with the output layer producing predicted results. To obtain error feedback during training, true labels (ground truth) are required. In adaptive equalizers, the target is the ideal signal obtained by convolving the partial response target with the write sequence. Since the ideal signal represents the optimal channel signal, in neural network equalizer training, the ideal signal obtained by convolving the partial response target with recorded bits serves as the ground truth label. The difference between the label and prediction yields the required training error, which then trains the neural network equalizer through backpropagation.

To meet neural network equalizer training requirements, readback signal sequences require preprocessing. As shown in Figure 12 [Figure 12: see original paper], assuming the readback signal sequence length is K, C readback signals are extracted sequentially from left to right to form a sample sequence, with the sampling interval length being C. The sampling interval slides across the readback signal sequence from left to right with a step size of 1, a process called equalization sampling, with the interval length termed sampling length. Each sample sequence serves as a training sample input to the neural network equalizer, with amplitude range scaled to between 0 and 1 through mapping. Different grayscale values in Figure 12 represent different readback signal values, all floating-point data. Through step-size-1 sampling, the final training sample matrix size is $C \times S$, where S is calculated as shown in Equation (6):

$$S = K - C + 1$$

The neural network equalizer training outputs S predictions, each requiring a corresponding true label. Therefore, the number of true labels is also S, obtained by convolving the partial response target across the recorded bit se-

quence with a step size of 1 from left to right. The label corresponding to a prediction from a sample sequence is obtained by convolving the bit sequence at the corresponding position with the partial response target, maintaining a one-to-one relationship. For unified computation, label values are also scaled to floating-point data between 0 and 1. Since label computation relates to the partial response target, target selection must correlate with actual optical disc channels for optimal training effectiveness. Different disc types have different partial response targets; for example, BDXL discs use $[1, 2, 2, 2, 1]$, while BD-ROM discs use $[1, 4, 6, 4, 1]$.

For optimal performance, a separate neural network equalizer is trained for each specific signal-to-noise ratio (SNR) using the Levenberg-Marquardt backpropagation algorithm. The network adopts a typical shallow feedforward structure: one hidden layer, one input layer, and one output layer. The input layer contains 9 neurons corresponding to 9 readback sampling points, with a single output neuron calculating the equalized signal for the center bit. Full connectivity is employed between layers. The hidden layer uses hyperbolic tangent sigmoid activation (tansig) to simulate the bipolar characteristics of magnetic recording. The output layer activation function performs linear scaling, mapping equalized output to the -1 to +1 range. The loss variation curves during training for BD-ROM and BDXL equalizer networks are shown in Figure 14 [Figure 14: see original paper], demonstrating good generalization capability. The loss function variation curves during training show that the model' s mean squared error (MSE) on validation and test sets decreases synchronously with the training set, converging to similar low levels. This indicates the model does not overfit training data but learns universal features of nonlinear crosstalk in optical disc channels. Furthermore, the model maintains error levels on unseen test data (approximately 10^{-2} magnitude) similar to the validation set, further proving reliable generalization capability. This robust performance ensures the neural network equalizer can effectively handle nonlinear distortion in optical disc read channels during actual deployment.

Based on the above methods for constructing training datasets and corresponding labels, this section proposes a basic three-layer neural network equalizer structure, as shown in Figure 13 [Figure 13: see original paper]. $\vec{r} = \{r_i | i = 1, 2, \ldots, K\}$ is the input sample sequence, where each sample signal in the input layer connects to hidden layer neurons via full connectivity. The number of hidden layer neurons equals the number of input samples. The hidden layer includes a normalization unit, linear computation unit, and activation unit. To reduce computation latency across multiple neuron layers, the neural network equalizer adopts a three-layer structure containing only one hidden layer to learn nonlinear crosstalk in the readback process. The three-layer perceptron structure ensures nonlinear fitting capability while avoiding latency and resource overhead from overly deep networks, making it suitable for edge deployment. The readout architecture adopted in this study conforms to ISO international standards for optical disc channel characteristics, simply replacing the existing finite impulse response (FIR) equalizer with a neural network equalizer. Neural

network equalizer training used 819,200 samples, with 70% for training, 15% for validation, and 15% for testing. Each sample contains 9 sampling points from the readback window and one corresponding center recorded bit.

3.2 HLS-Based Parallel NNML Signal Processing Method Implementation

In neural network equalizer signal processing, combining the neural network equalizer with a Viterbi detector and leveraging the block-processable nature of readback signals enables parallel NNML signal processing, as shown in Figure 15 [Figure 15: see original paper]. The Viterbi detector module implementation mirrors that described in Section 3. The readback signal distribution module still distributes readback signals through counting, and the recovered write sequence merging module still merges sequences through counting.

The neural network equalizer design is shown in Figure 16 [Figure 16: see original paper]. After receiving the readback signal sequence $\vec{r}$, the hidden computation layer module performs linear computation with the tansig activation layer module as shown in Equation (7), outputting sequence $\vec{y}_1$:

$$\vec{y}_1 = \text{tansig}(\mathbf{W}_1 \times \vec{r} + \vec{b}_1)$$

In the hidden computation layer module, all weight values in matrix $\mathbf{W}_1$ and all bias values in $\vec{b}_1$ are placed in separate arrays, with the HLS ARRAY_{PARTITION} directive mapping array data to different BRAMs to prevent data read blocking in subsequent computations. After receiving intermediate computation value $\vec{t}$ sequence, the activation layer applies the tansig function to each data element, then outputs the activated $\vec{y}_1$ sequence. The tansig activation layer module employs a lookup table approach to approximate the tansig function, reducing activation layer computation latency. After receiving the hidden layer output $\vec{y}_1$ sequence, the linear computation layer module performs linear computation as shown in Equation (8):

$$y_2 = \vec{w}_2 \times \vec{y}_1 + b_2$$

Similar to the hidden computation layer module implementation, the linear computation layer module stores weight vector $\vec{w}_2$ and bias b_2 in different BRAMs. All weight and bias values are obtained from neural network equalizer training —after training completes, model parameters are saved, exported as text, and finally input into each neural network equalizer computation layer in the HLS implementation. Model parameters exported after training are floating-point data. To reduce computation latency, these parameters are represented using fixed-point data formats in HLS implementation. To scale the neural network equalizer output back to the original data range, an inverse mapping operation follows linear layer computation. In the HLS neural network equalizer implementation, data mapping module 1 receives intermediate computation value

$\vec{t}'$ sequence and performs inverse mapping computation to restore data to the initial range, as shown in Equation (9):

$$\vec{y}' = \frac{\vec{t}' + 1}{2} \times (\max(\vec{y}) - \min(\vec{y})) + \min(\vec{y})$$

To reduce neural network equalizer computation latency, the hidden computation layer module design method is shown in Figure 17 [Figure 17: see original paper]. The two-dimensional weight matrix $\mathbf{W}_1$ is decomposed row-wise. Assuming matrix $\mathbf{W}_1$ has size $n \times n$, each weight vector has length n, totaling n weight vectors. Each computation unit contains one weight vector' s data, stored in different BRAMs. When a computation unit receives readback signals, it executes multiply-accumulate operations. This decomposes the original n multiply-accumulate computations in $\mathbf{W}_1 \times \vec{r}$ into n computation units executing one multiply-accumulate in parallel. Since each computation unit must perform multiply-accumulate on the same readback signal sequence, a readback signal replication module is designed with multiple streaming output interfaces. Whenever it reads one original readback signal, it outputs identical amplitude readback signals through multiple streaming interfaces to different computation units, which execute computations simultaneously. Computation units employ shift registers for readback signal reception, combined with FIFO connections to enable dataflow of input and output data. In this weight matrix $\mathbf{W}_1$ decomposition design across different computation units, since each computation unit operates in parallel, computation latency does not increase even when the number of hidden layer neurons grows.

Each computation unit' s output is processed by the tansig activation function, followed by linear superposition. The tansig computation in this paper involves 4 exponential operations. To reduce tansig function computation latency, the tansig activation layer module design adopts a lookup table approach. The tansig expression is shown in Equation (10):

$$\text{tansig}(x) = \frac{e^{2x} - e^{-2x}}{e^{2x} + e^{-2x}}$$

When the independent variable is in $[3, \infty]$ or $[-\infty, -3]$, the dependent variable range does not change significantly, with values approaching 1 or -1. Therefore, when intermediate computation value t exceeds 3 or falls below -3, tansig function output can be approximated as 1 or -1, with negligible impact on subsequent computations. After handling tansig output for independent variable ranges $[3, \infty]$ and $[-\infty, -3]$, a lookup table approach processes tansig output for independent variable range $[-3, 3]$. Sampling occurs at interval q within $[-3, 3]$; assuming $q = 0.1$, the independent variable sequence becomes $[-3.0, -2.9, -2.8, \ldots, 2.8, 2.9, 3.0]$, denoted as independent variable sequence $\vec{x}$. Each value in sequence $\vec{x}$ is individually input into the tansig function to compute the dependent variable sequence

$[-0.9951, -0.9940, -0.9926, \ldots, 0.9951, 0.9940, 0.9926]$, with sequence length denoted as L. When $q = 0.1$, $L = 60$. In the HLS tansig activation unit implementation, the dependent variable sequence is stored in an array of length L, which serves as the tansig function lookup table in the activation unit. After obtaining the lookup table, tansig output for $[-3, 3]$ can be obtained through index computation. Denoting the index as p, its computation is shown in Equation (11):

$$p = \frac{t - (-3.0)}{q}$$

Thus, index p is computed by subtracting the lookup table's left boundary value from the intermediate computation value and dividing by step size q. After computing the index value, the tansig activation unit can directly obtain tansig function output from the lookup table array via indexing. Since this lookup table is used only for index lookup without participating in any computation, the ARRAY_{PARTITION} directive is not required to map the lookup table array to different BRAMs.

The linear computation layer module stores linear layer weight vector $\vec{w}_2$ and bias b_2, mapping the weight vector $\vec{w}_2$ array to different BRAMs. The linear computation layer module then receives each tansig activation unit' s output, multiplies each tansig activation unit output by the corresponding weight coefficient in $\vec{w}_2$, and accumulates all multiplier results to obtain intermediate computation value t', outputting it to data mapping module 1. Data mapping module 1 implements inverse mapping computation, finally obtaining the entire neural network equalizer output value. While linear computation layer module and data mapping module 1 computations are simple, merging the two modules cannot achieve per-data-computation latency less than one clock cycle even with pipelining optimizations. Therefore, for overall neural network equalizer computation efficiency, the neural network equalizer' s linear layer is ultimately decomposed into two computation modules.

4 Experimental Evaluation

4.1 HLS Co-Simulation Method

HLS synthesizes C/C++ algorithm code into RTL implementation. Optimization directives can improve synthesized RTL quality. In HLS, synthesis report analysis provides RTL hardware resource consumption and computation latency, enabling synthesis quality assessment. In HLS, the core function implementing a module algorithm is called the top-level function. During synthesis flow, top-level function implementation correctness and post-synthesis RTL correctness must be verified, requiring test data and test functions for the top-level function. Simulation compiles and runs the top-level function through test functions, which must use the top-level function and include expected values with conditional statements comparing top-level function outputs against expected

values to verify correctness. After simulation, top-level function synthesis generates RTL implementation, and HLS employs an RTL adaptation mechanism to simulate test function functionality, including expected values and conditional statements. Test data is then input for co-simulation with the top-level function' s RTL implementation. If the top-level function' s RTL output matches expected values, the synthesized implementation is correct. The FPGA model used in this paper' s experiments is xcvu440-flga2892-1-c. According to Vivado HLS.

4.2 Parallel PRML and Parallel NNML BER Performance Analysis

Optical disc readback signal detection accuracy is influenced by multiple factors, particularly target recording bits and adjacent recording bits, with other factors such as optical pickup head position during operation and electronic noise also significantly impacting detection results. For comparison convenience, this section compares adaptive equalizers and neural network equalizers using only readback signals. The primary difference between parallel PRML and NNML signal processing methods lies in the equalizer used.

To compare equalization quality between adaptive and neural network equalizers and HLS implementation performance, two comparison experiments are designed. The adaptive equalizer uses 9 equalization coefficients with partial response target $[1, 2, 2, 2, 1]$. After sufficient iterative training, equalizer coefficients converge, with equalized output detected by a Viterbi detector. This experimental scheme is denoted as BDXL+FIR. The neural network equalizer uses input readback signal sample vector size 1×9 with partial response target $[1, 2, 2, 2, 1]$. After training with sufficient data, better neural network equalizer coefficients are obtained for each SNR, with equalized output detected by a Viterbi detector. This experimental scheme is denoted as BDXL+NNE. Additionally, another neural network equalizer uses input readback signal sample vector size 1×5 with partial response target $[1, 4, 6, 4, 1]$, denoted as BDROM+NNE. HLS implementations using 22-bit and 32-bit fixed-point numbers are denoted as BDROM+NNE+HLS22 and BDROM+NNE+HLS32 respectively. Both comparison experiments generate readback signals using optical disc channel models.

Figure 18 [Figure 18: see original paper] compares the bit error rates of the two equalization experiments under system electronic noise of 6-20 dB. From BDROM comparison results, appropriate data representation precision in HLS implementations of adaptive and neural network equalizers enables readback signal processing accuracy close to simulation accuracy. Using low-bit-width fixed-point numbers for neural network equalizer HLS implementation significantly degrades readback signal processing accuracy. For example, using 22-bit fixed-point numbers for neural network equalizer data computation increases system bit error rate to 10^{-1} magnitude at 20 dB SNR, while 32-bit fixed-point numbers maintain system bit error rate within acceptable range, demonstrating that 32-bit fixed-point numbers can adequately represent neural network equalizer data and computations. Meanwhile, BDXL experimental results show

that under 6-20 dB electronic noise interference, the neural network equalizer-based readback signal processing method produces fewer recording bit errors than the adaptive equalizer-based method, i.e., lower bit error rates. Under low SNR electronic noise conditions, neural network equalizers demonstrate better equalization performance than adaptive equalizers.

4.3 Parallel PRML and Parallel NNML Resource Consumption and Latency Analysis

In the parallel PRML equalization coefficient update component, within readback signals of identical quality, equalizer coefficients do not change significantly after sufficient iterations. As shown in Figure 19 [Figure 19: see original paper], equalizer coefficients stabilize after 200 iterations. Leveraging this characteristic, when iteration count is set to 500, the optimized equalizer coefficient update component latency reaches 44.49 µs.

In optical disc signal processing, data representation and computation involve decimals. To reduce hardware resource overhead from floating-point data and lower module computation latency, this paper does not use floating-point data to represent decimals. Optimized PRML and NNML modules use appropriate fixed-point data for representation and computation, thereby reducing hardware resource consumption and computation latency. Experiments show that using 1 sign bit and 10 integer bits (11 bits total) for integer representation meets PRML equalization detection integer computation requirements, while 11 bits for fractional representation meets PRML equalization detection fractional computation requirements. For NNML modules, as shown in Figure 20 [Figure 20: see original paper], when forward computation module depth is 1000 and fixed-point bit width is 32, adjusting integer bits to 14 maintains NNML bit error rate magnitude at 10^{-4}.

Typically, increased fixed-point bit width leads to increased computation resources. This paper experimentally employs 32-bit fixed-point numbers while ensuring reliable detection results. Error correction coding design could relax bit error rate requirements to reduce precision requirements, but this would affect error correction code efficiency and cost, also impacting resource consumption. Considering the large design space for fixed-point bit width, integer bits, and fractional bits, this paper does not extensively explore the precision-resource consumption trade-off. As shown in Figure 21 [Figure 21: see original paper], with 32-bit fixed-point numbers, forward computation module depth of 300 maintains bit error rate magnitude at 10^{-4}, while depth of 600 further reduces bit error rate at the cost of greater resource consumption. A computation module depth of 500 meets optical disc readback signal NNML detection requirements.

Since the HLS-based parallel PRML and parallel NNML processing methods decompose large modules into multiple small modules, breaking complex computations into multiple simple computations, and employ different optimiza-

tion pragmas in each small module based on computation principles, per-data-computation latency can be maintained within one clock cycle even when reducing each small module' s clock frequency. As shown in Figure 22 [Figure 22: see original paper], with original write sequence size of 64 KB, three PRML modules process in 3.46 ms, achieving signal processing speed of 144.50 Mb/s. Three NNML modules process in 3.76 ms, achieving signal processing speed of 132.97 Mb/s. Parallel PRML signal processing speed exceeds parallel NNML. Parallel NNML resource consumption exceeds parallel PRML, particularly in BRAM resource consumption, where parallel NNML consumes more than 7× that of parallel PRML.

5 Conclusion

Optical disc readback signal processing technology is a core technology in optical disc storage systems, determining data read/write reliability and ensuring long-term data preservation requirements, which is of great significance for massive cold data storage. Addressing the efficient readout requirements for cold data storage optical discs, this paper proposes a scalable high-level synthesis-based parallel data readout method that utilizes HLS technology for efficient processing of optical disc readback signals. Through theoretical analysis, methodology design, and experimental validation, this research achieves the following important results:

1. **High-performance parallel processing framework**: The proposed parallel PRML method achieves a processing speed of 144.50 Mb/s through pipeline optimization, data streaming, and parallel computing strategies, representing a 2.99× improvement over existing PRML. The parallel NNML method reduces average bit error rate by approximately 30% through neural network equalizers, significantly improving signal quality. These two methods each excel in speed and precision, providing flexible options for different application scenarios.

2. **HLS optimization innovations**: Through HLS techniques such as module decomposition, streaming processing, and fixed-point optimization, the challenges of complex hardware deployment and high resource consumption for neural network equalizers are addressed. Experiments demonstrate that 32-bit fixed-point implementation can maintain bit error rate at 10^{-4} magnitude while significantly reducing resource overhead, validating HLS' s tremendous potential in optical disc data readout signal processing.

3. **System-level contributions**: This research not only provides new insights for the high-performance and intelligent evolution of optical disc storage technology, but its modular design and parallel architecture can also be extended to other storage media, offering important references for green data storage technology development.

Compared with existing research, this work demonstrates significant advantages:

Unlike novel storage schemes relying on specific materials (such as polarization modulation technology [28]) or facing high latency issues (such as glass-ceramic storage [30]), this method exhibits superior performance in generality, real-time capability, and cost-effectiveness through algorithm-hardware co-design. Compared with existing FPGA acceleration schemes [9,17], its block-parallel strategy and resource optimization methods are better suited for large-scale edge cold data storage device processing scenarios.

Future research can expand in three directions: (1) Exploring lightweight adaptive online training neural network architectures to further reduce resource consumption; (2) Developing lightweight edge neural network maximum likelihood detection methods adapted for high-density storage media (such as super-resolution three-dimensional optical storage [29] and glass storage [31]); (3) Investigating energy consumption optimization strategies under heterogeneous computing architectures. These directions will drive cold data storage technology toward greater efficiency and intelligence.

References

[1] Wright A. Worldwide IDC Global DataSphere Forecast, 2024–2028: AI Everywhere, But Upsurge in Data Will Take Time[R]. International Data Corporation, 2024.

[2] Pinciroli R, Yang Lishan, Alter J, et al. Lifespan and Failures of SSDs and HDDs: Similarities, Differences, and Prediction Models[J/OL]. IEEE Transactions on Dependable and Secure Computing, 2023, 20(1): 256-272. DOI:10.1109/TDSC.2021.3131571.

[3] IEEE International Roadmap for Devices and Systems. Mass Digital Storage[A/OL]. Institute of Electrical and Electronics Engineers, 2023[2025-05-28]. https://irds.ieee.org/images/files/pdf/2023/2023IRDS_{MDS}.pdf. DOI:10.60627/4HS9-2098.

[4] Cong J, Liu Bin, Neuendorffer S, et al. High-Level Synthesis for FPGAs: From Prototyping to Deployment[J/OL]. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 2011, 30(4): 473-491. DOI:10.1109/TCAD.2011.2110592.

[5] Cong J, Zhang Zhiru. An efficient and versatile scheduling algorithm based on SDC formulation[C/OL]//Proceedings of the 43rd annual conference on Design automation - DAC ' 06. San Francisco, CA, USA: ACM Press, 2006: 433[2025-05-27]. http://portal.acm.org/citation.cfm?doid=1146909.1147025. DOI:10.1145/1146909.1147025.

[6] Zhang Zhiru, Liu Bin. SDC-based modulo scheduling for pipeline synthesis[C/OL]//2013 IEEE/ACM International Conference on Computer-Aided Design (ICCAD). San Jose, CA: IEEE, 2013: 211-218[2025-05-27]. http://ieeexplore.ieee.org/document/6691121/. DOI:10.1109/ICCAD.2013.6691121.

[7] Owaida M, Alonso G, Fogliarini L, et al. Lowering the latency of data processing pipelines through FPGA based hardware acceleration[J/OL]. Proceedings of the VLDB Endowment, 2019, 13(1): 71-85. DOI:10.14778/3357377.3357383.

[8] Turan F, Roy S S, Verbauwhede I. HEAWS: An accelerator for homomorphic encryption on the amazon AWS FPGA[J/OL]. IEEE Transactions on Computers, 2020, 69(8): 1185-1196. DOI:10.1109/TC.2020.2981868.

[9] Sun M, Li Z, Lu A, et al. FILM-QNN: Efficient FPGA acceleration of deep neural networks with intra-layer, mixed-precision quantization[C/OL]//Proceedings of the 2022 ACM/SIGDA international symposium on field-programmable gate arrays. Virtual Event, USA and New York, NY, USA: Association for Computing Machinery, 2022: 134-145. https://doi.org/10.1145/3490422.3502364. DOI:10.1145/3490422.3502364.

[10] Hao Cong, Zhang Xiaofan, Li Yuhong, et al. FPGA/DNN Co-Design: An Efficient Design Methodology for Intelligence on Edge[C/OL]//Proceedings of the 56th Annual Design Automation Conference 2019. Las Vegas NV USA: ACM, 2019: 1-6[2025-05-27]. https://dl.acm.org/doi/10.1145/3316781.3317829. DOI:10.1145/3316781.3317829.

[11] Wong L Y, Zhang Jialiang, Li J (Jane). DONGLE: Direct FPGA-Orchestrated NVMe Storage for HLS[C/OL]//Proceedings of the 2023 ACM/SIGDA International Symposium on Field Programmable Gate Arrays. Monterey CA USA: ACM, 2023: 3-13[2025-05-27]. https://dl.acm.org/doi/10.1145/3543622.3573185. DOI:10.1145/3543622.3573185.

[12] Xu Jiahui, Josipović L. Suppressing Spurious Dynamism of Dataflow Circuits via Latency and Occupancy Balancing[C/OL]//Proceedings of the 2024 ACM/SIGDA International Symposium on Field Programmable Gate Arrays. Monterey CA USA: ACM, 2024: 188-198[2025-05-27]. https://dl.acm.org/doi/10.1145/3626202.3637570. DOI:10.1145/3626202.3637570.

[13] Guo Licheng, Maidee P, Zhou Yun, et al. RapidStream: Parallel Physical Implementation of FPGA HLS Designs[C/OL]//Proceedings of the 2022 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays. Virtual Event USA: ACM, 2022: 1-12[2025-05-27]. https://dl.acm.org/doi/10.1145/3490422.3502361. DOI:10.1145/3490422.3502361.

[14] Chi Yuze, Guo Licheng, Cong J. Accelerating SSSP for Power-Law Graphs[C/OL]//Proceedings of the 2022 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays. Virtual Event USA: ACM, 2022: 190-200[2025-05-27]. https://dl.acm.org/doi/10.1145/3490422.3502358. DOI:10.1145/3490422.3502358.

[15] Fibich C, Tauner S, Rossler P, et al. Preliminary Evaluation of High-level Synthesis Tools - Xilinx Vivado and PandA Bambu[C/OL]//2018 IEEE 13th International Symposium on Industrial Embedded Systems (SIES). Graz: IEEE, 2018: 1-4[2025-05-27]. https://ieeexplore.ieee.org/document/8442100/. DOI:10.1109/SIES.2018.8442100.

[16] Brignone G, Bosio R, Ottati F, et al. SILVIA: Automated superword-level parallelism exploitation via HLS-specific LLVM passes for compute-intensive accelerators[J/OL]. ACM Transactions on Reconfigurable Technology and Systems, 2025, 18(2): 1-16. DOI:10.1145/3705324.

[17] He A, Key D, Bulling M, et al. HLSTransform: Energy-efficient llama 2 inference on FPGAs via high level synthesis[A/OL]. arXiv, 2024. https://arxiv.org/abs/2405.00738.

[18] Zhao Chenfeng, Dong Zehao, Chen Yixin, et al. GNNHLS: Evaluating Graph Neural Network Inference via High-Level Synthesis[A/OL]. arXiv, 2023[2025-05-28]. https://arxiv.org/abs/2309.16022. DOI:10.48550/arxiv.2309.16022.

[19] Lau J, Xiao Yuanlong, Xie Yutong, et al. RapidStream IR: Infrastructure for FPGA High-Level Physical Synthesis[C/OL]//Proceedings of the 43rd IEEE/ACM International Conference on Computer-Aided Design. Newark Liberty International Airport Marriott New York NY USA: ACM, 2024: 1-11[2025-05-28]. https://dl.acm.org/doi/10.1145/3676536.3676649. DOI:10.1145/3676536.3676649.

[20] Szafarczyk R, Nabi S W, Vanderbauwhede W. Dynamic Loop Fusion in High-Level Synthesis[C/OL]//Proceedings of the 2025 ACM/SIGDA International Symposium on Field Programmable Gate Arrays. Monterey CA USA: ACM, 2025: 211-222[2025-05-28]. https://dl.acm.org/doi/10.1145/3706628.3708871. DOI:10.1145/3706628.3708871.

[21] Kim K, Kim S H, Koo G, et al. Decision feedback equalizer for holographic data storage.[J]. Applied optics, 2018, 57(15): 4056-4066.

[22] Saito K. Multibeam crosstalk cancellation method with binarized data for optical disc readout[C]//2022 Conference on Lasers and Electro-Optics Pacific Rim (CLEO-PR). Sapporo, Japan: IEEE, 2022: 1-2.

[23] Koonkarnkhai S, Warisarn C, Kovintavewat P. A novel ITI suppression technique for coded dual-track dual-head bit-patterned magnetic recording systems[J]. IEEE Access, 2020, 8: 153077-153086.

[24] Nakthanom S, Koonkarnkhai S, Kovintavewat P. An MLP-based equalization for bit-patterned magnetic recording systems[C/OL]//2024 International Technical Conference on Circuits/Systems, Computers, and Communications (ITC-CSCC). Okinawa, Japan: IEEE, 2024: 1-5. DOI:10.1109/ITC-CSCC62988.2024.10628313.

[25] Miyashita H, Nakamura A, Nakajima T, et al. New prml for asymmetrical signals in high-density optical disks[J]. Japanese Journal of Applied Physics, 2002, 41: 1787.

[26] Reddy B R, Varalakshmi B. FPGA based efficient implementation of viterbi decoder[J]. International Journal of Computer Engineering in Research Trends, 2015, 2(12): 1076-1082.

[27] Mozaffari Kermani M, Singh V, Azarderakhsh R. Reliable low-latency viterbi algorithm architectures benchmarked on ASIC and FPGA[J/OL]. IEEE Transactions on Circuits and Systems I: Regular Papers, 2017, 64(1): 208-216. DOI:10.1109/TCSI.2016.2610187.

[28] Zhang Li, Li Wenwen, Wang Zhongyang. Sub-Diffraction Readout Method of High-Capacity Optical Data Storage Based on Polarization Modulation[J/OL]. Nanomaterials, 2024, 14(4): 364. DOI:10.3390/nano14040364.

[29] Zhao Miao, Wen Jing, Hu Qiao, et al. A 3D nanoscale optical disk memory with petabit capacity[J/OL]. Nature, 2024, 626(8000): 772-778. DOI:10.1038/s41586-023-06980-y.

[30] Lunt B M, Riviera F, Santos J, et al. Writing for the Future. What are the Options?[J/OL]. Journal of Imaging Science and Technology, 2025, 69(2): 1-7. DOI:10.2352/J.ImagingSci.Technol.2025.69.2.020402.

[31] Anderson P, Aranas E, Assaf Y, et al. Project Silica: Towards Sustainable Cloud Archival Storage in Glass[J/OL]. ACM Transactions on Storage, 2025, 21(1): 1-31. DOI:10.1145/3708996.

[32] Wang Chu-Han, Ma Jie, Feng Yu-Du, et al. Error-Free Long-Lifespan Optical Storage Enhanced by Deep Learning[J/OL]. Laser & Photonics Reviews, 2024, 18(6): 2301042. DOI:10.1002/lpor.202301042.

Author Contributions

Ke Luo proposed optimization strategies for the parallel PRML signal processing method and designed the HLS implementation schemes for the equalizer and Viterbi detector. Yugen Jian was responsible for algorithm simulation and optimized the dataflow scheduling strategy. Hongyu Gao designed the parallel NNML signal processing architecture and optimized hardware deployment of the neural network equalizer. Ping Lu participated in project resource coordination, providing important support for experimental platform construction and hardware deployment. Jincai Chen was responsible for overall project guidance and resource coordination, and led project management and research efforts.

Note: Figure translations are in progress. See original paper for figures.