

A Fully Combinational Hardware Accelerator for BWT-FM Index DNA Sequence Alignment

I. A. ZAIDI¹, A.K. HALIM¹, S.A.M. AL JUNID¹, M. M. ALI.², A.H.A.M KHALID³,
A.H.A. RAZAK^{1,*}

¹Electronic Architecture and Application Research Group (EArA),
Faculty of Electrical Engineering, Universiti Teknologi MARA,
40450 Shah Alam, Selangor,
MALAYSIA

²Faculty of Accountancy, Universiti Teknologi MARA,
42300 Puncak Alam, Selangor,
MALAYSIA

³Intel Microelectronics (M) Sdn. Bhd.,
P.O. Box 121,
Bayan Lepas Free Industrial Zone, 11900 Penang,
MALAYSIA

**Corresponding Author*

Abstract: - Next-generation sequencing has generated a deluge of genomic data that transcends the processing capabilities of traditional CPU-based aligners and sequential hardware architectures, which remain constrained by the memory wall and clock-driven execution loops. This paper presents a novel, fully combinational hardware accelerator for exact BWT-FM index DNA sequence alignment, manifested within a non-sequential logic fabric. By synthesising a hardware-native 16-element Bitonic Sorting Network for BWT construction and unrolling the FM-index lookup into a purely spatial architectural chain, the proposed design effectively eliminates the control-flow overhead and sequential state transitions intrinsic to conventional models. The architecture, implemented in Verilog HDL and verified via RTL simulations, achieves 100% mapping accuracy across heterogeneous query workloads while providing a deterministic, sub-cycle latency profile defined solely by the hardware's propagation delay (T_{pd}). Performance characterisation on a Cyclone IV FPGA confirms that search resolution is decoupled from clock-cycle iterations, converting iterative search algorithms into a spatial logic path. This study establishes a new benchmark for high-throughput genomic mapping, offering a scalable blueprint for real-time clinical diagnostics and advanced genomic surveillance.

Key-Words: - Bitonic Sorting Network, Burrows-Wheeler Transform (BWT), DNA sequence alignment, Fully combinational logic, FM-index, Hardware acceleration, Field Programmable Gate Array (FPGA), Near-data processing.

Received: April 2, 2026. Revised: May 3, 2026. Accepted: June 4, 2026. Published: October 20, 2026.

1 Introduction

The rapid advancement of high-throughput sequencing technologies has transformed modern biological research and clinical diagnostics, replacing labour-intensive Sanger sequencing with Next-Generation Sequencing (NGS) platforms capable of massively parallel genome interrogation [1], [2]. NGS generates millions to billions of short Deoxyribonucleic Acid (DNA) fragments, or *reads*, in a single run, providing the data foundation for

applications such as precision medicine, phylogenetics, and pathogen surveillance [3]. However, the resulting scale and fragmentation of sequence data impose substantial computational demands, requiring robust and efficient bioinformatic frameworks to accurately reconstruct original genomic sequences, thereby making read processing a central challenge in contemporary genomic workflows [4].

Sequence alignment, the process of mapping these short reads to a high-fidelity reference genome, forms

the core computational step in genomic sequence reconstruction [5]. To manage the massive search space involved, the bioinformatics community has widely adopted the Burrows-Wheeler Transform (BWT) and the Ferragina-Manzini (FM) index as the algorithmic gold standard for read alignment [6], [7]. These approaches achieve search time complexity proportional to the length of the query read rather than the size of the reference genome, offering substantial performance advantages for large-scale datasets [8]. By utilising lexicographical sorting and compressed suffix arrays, BWT-FM index frameworks substantially reduce memory requirements for genomic indexing, facilitating efficient alignment against multi-gigabase genomes on commodity computing platforms [9].

Despite the exponential reduction in genomic sequencing costs, which have plummeted from \$100 million per genome in 2001 to less than \$100 today, the computational infrastructure required to process this data has failed to keep pace, creating a "genomic bottleneck" that threatens the feasibility of real-time clinical diagnostics [10]. Recent statistics indicate that global genomic data is projected to reach 40 exabytes annually by 2025, a growth rate that significantly outstrips Moore's Law and the performance gains of general-purpose central processing units (CPUs) [11]. While software-based aligners such as Bowtie2 and Burrows-Wheeler Aligner-Maximal Exact Matches (BWA-MEM) are industry standards, they are fundamentally constrained by the "memory wall" and the "von Neumann bottleneck," where irregular memory access patterns for FM-index lookups result in excessive cache misses and wasted clock cycles [12]. Consequently, the latency associated with mapping millions of short DNA reads remains a prohibitive barrier for time-sensitive applications, such as identifying pathogen outbreaks or guiding emergency oncological interventions.

Furthermore, current hardware-based solutions, including Field Programmable Gate Array (FPGA) and Graphics Processing Unit (GPU) accelerators, predominantly utilise pipelined processing elements that remain inherently sequential in their logic execution [13]. While these architectures parallelise the *volume* of reads, the *process* of aligning an individual read still relies on recursive backtracking or iterative state machines, which introduce significant control-flow overhead and prevent single-cycle throughput [11]. There is a critical research gap in developing "non-sequential" architectures that can transform the temporal complexity of the BWT into spatial logic. Without a fundamental shift toward fully combinational architectures that can resolve search

intervals in a single propagation path, the industry will remain unable to achieve the low-latency genomic processing required for the next-generation healthcare systems.

To address the inherent latency and sequential bottlenecks of existing genomic mapping architectures, this paper introduces a fully combinational hardware accelerator designed to execute the Bowtie short-read alignment algorithm via a non-sequential, spatial logic path. The primary objective of this research is to develop a Verilog Hardware Description Language (HDL)-based architecture that fundamentally eliminates the reliance on clock-driven state machines and iterative loops for exact DNA matching. This is achieved by employing a parallel Bitonic Sorting Network for hardware-native BWT construction and by unrolling the FM-index backward search into a cascading chain of combinational logic. In doing so, the architecture bypasses the von Neumann bottleneck and removes the control-flow overhead characteristic of both software-based and pipelined hardware implementations.

This approach transforms the traditionally recursive and time-intensive alignment process into a single logic propagation event, enabling near-instantaneous search execution. By replacing iterative rank-and-count operations with a spatial array of dedicated, combinational processing modules, the design provides a deterministic latency profile independent of clock-cycle iterations for the backwards-search loop. Consequently, the proposed architecture constitutes a scalable and energy-efficient hardware primitive that effectively mitigates the genomic processing bottleneck, thereby enabling the ultra-high throughput and real-time responsiveness required for next-generation clinical diagnostics and surveillance systems.

The primary novelty of this work lies in the architectural transition from temporal, clock-driven processing to a fully spatial, combinational logic paradigm for BWT-FM index DNA sequence alignment. While existing FPGA-based accelerators rely on deeply pipelined processing elements that still match nucleotide bases in iterative clock cycles, this research introduces an architecture that unrolls the entire backwards-search algorithm into a non-sequential, physical logic path. By integrating a hardware-native Bitonic Sorting Network for deterministic BWT construction with a cascading chain of combinational rank-and-count modules, this design transforms the iterative FM-index search into a non-sequential, spatial logic path. Together, these design choices eliminate the control-flow overhead,

sequential state transitions, and recursion inherent in conventional alignment implementations, reducing the alignment process to a single logic propagation delay.

From a scientific perspective, this work establishes a validated proof of concept for deterministic, sub-cycle alignment, addressing the stringent latency requirements of real-time clinical diagnostics. By architecting the BWT-FM index as a purely spatial network, this research offers a solution to the "memory wall" and the "von Neumann bottleneck" that frequently degrade the performance of software and sequential hardware aligners. Moreover, the use of a fixed-path Bitonic sorting structure ensures a predictable latency profile and high energy efficiency by minimising the switching activity associated with register-heavy pipelined designs. This study provides both a robust architectural framework and a technical benchmark for researchers seeking to exploit the extreme parallelism of digital logic to meet the exascale data challenges of modern genomics.

This paper presents a spatial combinational architecture in hardware-based alignment by proposing a fully combinational architecture. Our principal contributions include:

1. **Parallelised Bitonic-BWT:** A hardware-native BWT implementation using a Bitonic Sorting Network that operates in $O(\log^2 n)$ stages.
2. **Unrolled FM-Index Logic:** A combinational implementation of $C[c]$ and $Occ[c, k]$ tables, allowing for "Count" and "Locate" operations without sequential backtracking.
3. **Single-Path Alignment:** An architecture that allows a complete exact match search to occur within a single logic propagation path, theoretically allowing alignment at the speed of the hardware's maximum frequency.

The remainder of this article is structured as follows: Section 2 provides an analytical review of extant literature and contemporary hardware acceleration paradigms in genomic mapping. Section 3 delineates the architectural methodology and the logic-unrolling strategies employed in the proposed fully combinational aligner. Section 4 presents the experimental findings, accompanied by a robust discussion on functional verification and performance benchmarks. Finally, Section 5 offers concluding remarks and elucidates potential trajectories for future research.

The authors confirm that the architecture, Verilog HDL implementations, RTL simulation data, and textual content presented in this manuscript are entirely original work and have not been previously

published in, submitted to, or adapted from any conference proceedings or prior studies. All hardware modules, logic unrolling strategies, and performance evaluation metrics were developed specifically for this study.

2 Literature Review

Addressing the escalating computational demands of modern genomics has prompted a strategic pivot within the research community, shifting the architectural focus from software-defined iterations toward the synthesis of highly specialised hardware primitives. Driven by the stringent throughput and latency exigencies of modern clinical applications, recent academic efforts have gravitated towards three synergistic objectives: addressing the pervasive memory wall in BWT-based indexing, optimising hardware-native sorting networks, and replacing sequential paradigms with spatial, fully combinational logic. This section provides a structured review of the existing literature within these domains, evaluating how advancements in NDP and Bitonic Sorting provide the requisite infrastructure for high-speed genomic mapping. By examining the inherent performance ceilings of current pipelined and automata-based architectures, this review identifies the critical research gap for non-sequential, fully combinational designs capable of resolving complex string-matching tasks within a singular logic propagation path.

2.1 Advancements in BWT-based Hardware Accelerators for Genomic Mapping

To mitigate the memory-latency bottleneck during FM-index lookups, recent architectural paradigms have pivoted toward algorithm-hardware co-design and near-data processing strategies. Accelerators such as Niubility leverage bit-wise optimisations and parallel memory streams to maximise bandwidth utilisation [14]. Concurrently, Processing-Near-Memory (PNM) and Processing-In-Memory (PIM) frameworks demonstrate performance improvements of up to 3.7x and energy consumption reductions of up to 40x compared to conventional systems [15]. Similarly, PIM frameworks, exemplified by Binary Index-based Memory-centric Sequence Aligner (BIMSA) and PIM-Align, further mitigate data-movement penalties by embedding computation directly within memory arrays; BIMSA, specifically, has been shown to achieve up to a 22.24x speedup over state-of-the-art PIM implementations [16], [17].

The efficiency of contemporary DNA alignment accelerators is profoundly dictated by inherent

architectural constraints, primarily stemming from irregular memory access and hardware resource overhead. The FM-index search algorithm is characterised by fine-grained, random memory access patterns that inherently restrict parallelism in conventional systems; however, PIM architectures have attempted to mitigate this bottleneck by utilising 3D-stacked Dynamic Random-Access Memory (DRAM) technology to enhance data-localised throughput [17]. Furthermore, while pipelined architectures are designed to augment throughput, they often encounter suboptimal speedup results due to a heavy reliance on external memory resources and the structural complexity involved in integrating optimised circuitry for large-scale parallel computations [18]. Furthermore, these architectural shortcomings are magnified by finite physical resource constraints; notably, in FPGA-based designs, the pursuit of high throughput often necessitates extensive DRAM overhead, which ultimately curtails scalability and suppresses the performance ceiling of the accelerator [19].

Despite these advancements, contemporary BWT accelerators remain inherently bound by fine-grained, irregular memory access patterns. While PIM and pipelined FPGA architectures increase aggregate throughput across parallel reads, individual read alignments still depend on iterative state transitions and external memory lookups. Consequently, physical resource overhead and DRAM bandwidth saturation continue to restrict operational scalability, leaving sub-clock-cycle alignment latency an unfulfilled objective in cycle-driven designs.

2.2 Comparative Analysis of Hardware-Native Sorting Algorithms in Bioinformatics

Bitonic Sorting Networks offer several architectural advantages over traditional Radix or Quicksort implementations when utilised for real-time lexicographical data transformations in BWT generation on FPGAs. These advantages stem from their inherent parallelism, efficient resource utilisation, and suitability for hardware implementation, which are critical for achieving high throughput and low latency in real-time applications. The following sections detail these advantages, supported by insights from recent research.

Bitonic Sorting Networks constitute a potent hardware paradigm for high-performance computing, leveraging an intrinsic structural parallelism that facilitates the concurrent orchestration of multiple data streams. This parallelism is a critical determinant of system throughput; research has demonstrated that these networks can sort an N -key sequence with a

throughput of $O(p)$ keys per cycle, where p represents the degree of parallelism [20], [21]. Moreover, the capacity to process n elements per cycle with $O(n \log^2 n)$ logic and storage overhead renders bitonic sorting highly efficient for real-time applications tasked with managing voluminous datasets [22]. Beyond raw performance, these networks are specifically engineered for resource efficiency on FPGA platforms by minimising internal communication and optimising logic utilisation. For instance, the implementation of recurrent bitonic sorting configurations can significantly decrease area complexity without compromising time complexity, making them ideal for concurrently accessible data [23]. Furthermore, strategic architectural choices, such as utilising single-port memory in place of dual-port memory, have been shown to yield a 50% reduction in memory consumption, further enhancing the overall resource footprint of the accelerator [20].

The flexibility and scalability of Bitonic Sorting Networks enable them to be effectively reconfigured for diverse input sizes and data widths through targeted enhancements in input registers and the strategic redistribution of comparator levels across the network [23]. This inherent adaptability ensures that the architecture can scale to accommodate increasingly large genomic datasets without incurring disproportionate resource overhead, maintaining high-performance standards even as data volumes grow [21]. Furthermore, the minimal control overhead and high degree of data parallelism associated with bitonic networks make them exceptionally well-suited for FPGA implementations, resulting in substantial improvements in energy and memory efficiency compared to current designs [20], [21]. Specifically, within BWT applications, FPGA-based bitonic sorting implementations have been shown to facilitate significant reductions in execution cycles and elevations in maximum operating frequency, thereby optimising overall system throughput [24].

While Bitonic Sorting Networks offer significant advantages, it is important to consider the context in which they are used. Conventional sorting methodologies, such as Radix and Quicksort, may remain advantageous in contexts where their specific operational profiles are more congruent with application-specific constraints, particularly when managing non-uniform data distributions or in software-defined environments where such implementations are more expedient. Additionally, the complexity of implementing advanced data structures in hardware can pose challenges, necessitating careful consideration of the trade-offs involved in choosing the appropriate sorting architecture for a given

application [24], [25]. Table 1 summarise the comparison of sorting complexity.

Table 1. Summary of the comparative sorting complexity

Parameter	Bitonic Sorting Network (Proposed)	Radix Sort	Merge Sort
Comparator Count	$\frac{N}{4} \log_2 N (\log_2 N + 1)$	$O(k \cdot N)$	$\frac{N}{2} \log_2 N$
Logic Stage Depth	$\frac{\log_2 N (\log_2 N + 1)}{2}$	Iterative / k cycles	$\log_2 N$ stages
Combinational Feasibility	High (Fixed spatial wire network)	Low (Requires sequential loops)	Moderate (Requires intermediate registers)
Routing Interconnect Complexity	$O(N \log^2 N)$ deterministic fixed wiring	Dynamic multiplexing heavy	Dynamic buffer heavy
Data-Dependency Latency	Deterministic / Invariant	Key-distribution dependent	Memory access dependent

Source: created by the authors

2.3 Transitioning from Temporal to Spatial Computation in High-Frequency String Matching

Contemporary advancements in transposing temporal algorithmic recursion into purely combinational logic involve the strategic utilisation of FPGAs, programmable switching fabrics, and specialised automata processing architectures. These approaches aim to mitigate area-delay trade-offs and propagation-delay challenges associated with scaling query lengths by optimising resource utilisation and processing efficiency. Innovative techniques facilitate the transition from sequential to non-sequential designs by enhancing throughput and reducing latency while maintaining or improving energy and memory efficiency.

Translating temporal algorithmic recursion into purely combinational logic has driven the adoption of specialised automata processing on FPGAs and programmable switching fabrics. Architectures utilising Multi-Character Non-Deterministic Finite Automata (MC-NFA) [26] and pipelined look-up tables (LUTs) facilitate concurrent state transitions [27].

Innovative in-memory and automata paradigms have significantly advanced pattern matching and regular expression execution. Notably, the Bit Vector

Automata Processor (BVAP) leverages hardware-software co-optimisation to accelerate bounded-repetition regular expressions, achieving higher energy efficiency and a smaller silicon footprint than conventional automata processors [28]. Similarly, the Lightweight Automata Processing Engine (LAP) maximises throughput by storing full models on-chip using an Abstract Deterministic Finite Automaton (A-DFA) microarchitecture that balances speed and storage efficiency [29]. Beyond on-chip memory, the Better Optimising Large Tables (BOLT) system employs programmable switches and variable k -stride state encoding to fit expansive pattern sets into memory-constrained environments, maximising throughput without additional hardware costs [30]. Finally, the CAMA accelerator demonstrates that Content-Addressable Memory (CAM) outperforms conventional SRAM for processing Non-Deterministic Finite Automata (NFAs), using novel encoding to reduce energy and area while optimising throughput [31].

Nevertheless, achieving deterministic execution across heterogeneous query workloads remains a key challenge. Although non-sequential paradigms improve pattern-matching speed, existing spatial architectures are predominantly tailored for regular expressions rather than the multi-stage, range-calculation logic required by the FM-index. Resolving complex BWT search steps within a single propagation path (T_{pd}) requires fully unrolling both the rank-and-count and coordinate-tracing operations into a continuous spatial logic array.

2.4 Research Gap

Despite the proliferation of high-performance Near-Data Processing (NDP) frameworks and specialised automata architectures, contemporary hardware accelerators remain fundamentally tethered to a temporal processing paradigm characterised by iterative, clock-driven state transitions. While current state-of-the-art designs, ranging from pipelined FPGA implementations to 3D-stacked PIM frameworks, successfully optimise cumulative throughput, they consistently fail to achieve sub-clock-cycle alignment latency for individual genomic reads due to their inherent reliance on sequential backtracking and iterative rank-and-count loops. A critical research gap exists in the synthesis of hardware-native Bitonic Sorting Networks with fully unrolled FM-index logic to instantiate a purely spatial, non-sequential architecture that eliminates the control-flow overhead of traditional state machines. The absence of a fully combinational primitive capable of resolving the entire BWT-FM index alignment process within a

singular logic propagation path represents a significant barrier to achieving the deterministic, ultra-low latency required for the next generation of real-time genomic diagnostics.

3 Methodology

The experimental framework for developing the fully combinational BWT-FM index accelerator is structured into three distinct operational phases: genomic data pre-processing, architectural hardware design, and system-level verification, as illustrated in Figure 1 (Appendix). Initially, raw genomic sequences are procured and transformed into a hardware-compatible binary format. This is followed by the core architectural phase, where the traditional recursive alignment algorithm is unrolled into a non-sequential, spatial logic path consisting of a Bitonic Sorting Network and combinational FM-index modules. Finally, the design is subjected to rigorous Register-Transfer Level (RTL) functional simulations and performance characterisation to validate the single-cycle alignment capability. The subsequent subsections delineate the specific technical procedures executed within each stage of this workflow.

3.1 Data Acquisition and Binary Nucleobase Mapping

Reference genomes and query sequences were procured in standard FASTA and FASTQ formats to provide a high-fidelity baseline for alignment verification. Prior to their insertion into the RTL simulation environment, these sequences were converted into a hardware-compatible binary format via a specialised Python-based preprocessing script. This script was utilised to directly transform the raw FASTA files into binary streams, which were subsequently allocated within the simulation memory to facilitate real-time data retrieval during the alignment process.

The conversion process for the *Enterobacteria phage lambda* genome is exemplified in Figure 2 (Appendix), where the raw genomic characters are systematically parsed into their binary equivalents. As illustrated in Figure 2 (Appendix), the transition from the raw nucleobase text to a serialised binary format ensures that the genomic data is formatted for direct injection into the hardware-based alignment engine. This preprocessing step is critical for minimising the computational overhead during the simulation phase and for validating the integrity of the data stream before hardware-level processing begins.

To bridge the interface between biological data and digital logic, the four-symbol DNA alphabet

{"A,C,G,T"} was encoded into a hardware-manageable 4-bit binary representation. However, DNA sequences are typically characterised by a limited set of nucleobases, and a 4-bit encoding scheme was adopted to provide the architectural flexibility required for potential expansion into complex protein sequences. Each nucleobase corresponds to a unique representative 4-bit value, as delineated in the binary conversion scheme: \$ = 0000, A = 0001, C = 0010, G = 0011, T = 0100, and an empty base symbol, X = 1111. During this stage, the sequences were partitioned and formatted into localised memory structures, ensuring that the nucleobase mapping was optimised for the parallel data-bus requirements of the subsequent hardware stages.

3.2 Parallelised Bitonic-BWT Core Design

A 16-element Bitonic Sorting Network was developed to perform the BWT in a fully combinational manner. Unlike traditional sequential sorting algorithms, this hardware-native architecture was synthesised to handle all cyclic rotations and lexicographical sorting within a single logic pass. By leveraging a fixed-path comparator network, the BWT construction was achieved with $O(\log^2 n)$ complexity, effectively eliminating the need for iterative software loops and providing a deterministic foundation for the FM-index generation.

The systematic procedure for the BWT process is delineated in the operational flowchart shown in Figure 3. The transformation was executed through a series of discrete hardware stages, beginning with the fanning-out of the query sequence into parallel lanes, followed by combinational sorting, and culminating in the extraction of the transformed sequence. This non-sequential flow ensures that the transformation is completed within a fixed propagation delay, bypassing the latency overhead associated with the recursive sorting functions used in software-based implementations such as Bowtie.

The initial stage of the BWT transform involved the cyclic rotation of the reference sequence. In this implementation, the reference sequence was defined as a set of 16 nucleotide bases, represented as a 64-bit binary value (incorporating 16 4-bit encoded bases). This 64-bit value was fanned out into 16 parallel lanes, each feeding into a dedicated rotator module. These modules executed a cyclic shift of the sequence in increments from 0 to 16 for each respective lane, thereby generating the full matrix of cyclic rotations required for lexicographical sorting without the use of sequential registers.

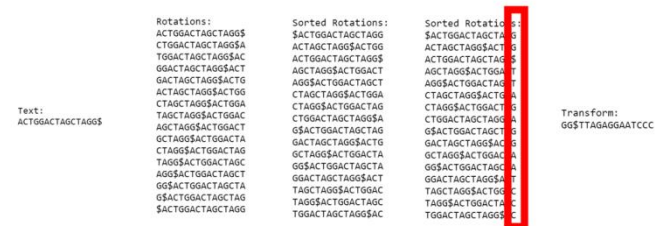


Fig. 3: The Flowchart for the Burrows-Wheeler Transform process

Source: created by the authors

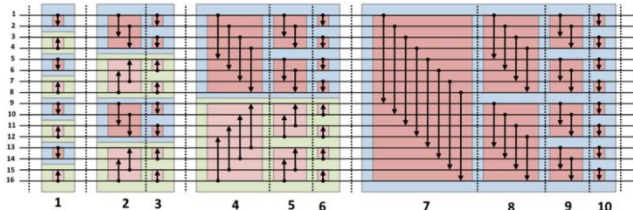


Fig. 4: The Bitonic Sort network utilised for the sorting process

Source: created by the authors

To resolve the lexicographical order of these 16 rotated 64-bit sequences, a parallel sorting architecture was instantiated. As illustrated in Figure 4, a 16-element Bitonic Sorting Network was utilised, consisting of 16 wires divided into 10 distinct sorting stages. The network was composed of an optimised arrangement of 64-bit comparator modules. Within each module, two 64-bit reference sequences were evaluated; the larger binary value was steered toward the arrow tip (maximum), while the smaller value was swapped to the dot symbol (minimum). This entirely combinational approach allowed for the simultaneous, parallel sorting of the rotation matrix, ensuring that the 16 reference sequences were ordered from the smallest to the largest binary value in an ascending manner.

To substantiate the deterministic performance of this non-sequential architecture, the worst-case propagation delay for the BWT sorting stage ($T_{pd,BWT}$) can be formally expressed as a function of the reference sequence length (N) and the combinational delay of a single comparator module (t_{cmp}). Given the intrinsic $O(\log^2 N)$ logic depth of a Bitonic Sorting Network, the theoretical latency expression is modelled by the following equation:

$$T_{pd,BWT} = \frac{[\log_2 N](\log_2 N + 1)}{2} \cdot t_{cmp} \quad (1)$$

For the $N = 16$ architecture presented in this study, the logic depth resolves to precisely 10 combinational stages. Consequently, the total propagation delay scales spatially through the logic gates rather than temporally through clock-driven registers, ensuring that the entire matrix is sorted from the smallest to the largest binary value in a single, continuous logic pass.



Fig. 5: The concatenation of the last 4-bits to obtain the Last Column of the BWT transform process

Source: created by the authors

The final phase of the BWT core involved extracting the BWT string, or the "Last Column," from the sorted matrix. As shown in Figure 5, this was achieved by concatenating the last, least significant 4-bits of each of the 16 sorted reference sequences. While the sorting process produces a full matrix, only the first and last columns are of architectural interest for the subsequent FM-index generation. The resulting 64-bit transformed sequence provides the necessary lexicographical mapping to facilitate the exact matching process, with the unrolled hardware logic enabling the extraction of these values in a single-pass propagation delay.

3.3 Combinational FM-Index Logic Synthesis

The generation of the auxiliary data structures, specifically the $C[c]$ and $Occ[c, k]$ tables, was implemented through a series of parallel adder trees and combinational rank-and-count modules. To satisfy the requirements of non-sequential mapping, the logic was architected to calculate base frequencies and cumulative occurrences across the entire BWT string simultaneously. By mapping the FM-index into a spatial logic array, the architecture achieves low-latency data retrieval, successfully bypassing the "memory wall" constraints endemic to conventional software-defined search models.

The $C[c]$ table was synthesised by utilising the "First Column" extracted from the BWT transform as the primary input. As illustrated in Figure 6, this column was decoded into five separate 16-bit

registers, each representing a unique nucleobase in the alphabet $\{\$, A, C, G, T\}$. During this decoding process, each 4-bit encoded base within the first column was evaluated against the representative base value; a logic high (1) was asserted in the 16-bit register if a match was identified; otherwise, a logic low (0) was maintained. Subsequently, a combinational addition process was performed to determine the total population of "1" bits within these five registers. These totals were then added cumulatively to manifest the final $C[c]$ table, which was represented within a 20-bit register to ensure sufficient bit-width for genomic indexing.

To formalise this hardware-native Boolean decoding and cumulative counting process, the generation of the $C[c]$ table can be mathematically expressed as a spatial population count and prefix-sum operation:

$$C[c] = \sum_{b < c} \sum_{i=1}^N \mathbb{I}(L_{\text{first}}[i] = b) \quad (2)$$

where N represents the length of the reference sequence (16 in this implementation), $L_{\text{first}}[i]$ denotes the i -th character of the extracted first column, c and b represent characters from the lexicographically ordered alphabet, and $\mathbb{I}(\cdot)$ is a Boolean indicator function that asserts a logic high (1) upon an exact character match and a logic low (0) otherwise.

First Column	\$	A	A	A	A	C	C	C	G	G	G	G	T	T	T	Total:
Decode \$	1															1
Decode A		1	1	1	1											4
Decode C						1	1	1								3
Decode G									1	1	1	1	1			5
Decode T														1	1	3

Add cumulatively	Base	\$	A	C	G	T
	$C[c]$	0	1	1+4=5	5+3=8	8+5=13

Fig. 6: The decoding and cumulative addition process for the generation of $C[c]$ table
Source: created by the authors

A similar architectural strategy was employed for the generation of the $Occ[c, k]$ table, utilising the "Last Column" of the BWT transform as the input source. Similar to the $C[c]$ generation, the last column was decoded into five distinct 16-bit registers to isolate the occurrences of each nucleobase. The architectural unrolling of this process is detailed in Figure 7, which visualises the generation of the Occ_row for each base. To facilitate a purely combinational rank operation, a 64-bit register named Occ_row was implemented, consisting of 16 individual 4-bit columns.

L->	G	G	\$	T	T	A	G	A	G	G	A	A	T	C	C	C
Decode \$	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
Add Diagonally	0	0	1	0	0	1	1	1	1	1	1	1	1	1	1	1
Occ_row_\$	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1

L->	G	G	\$	T	T	A	G	A	G	G	A	A	T	C	C	C
Decode A	0	0	0	0	0	0	1	0	0	0	1	1	0	0	0	0
Add Diagonally	0	0	0	0	0	1	1	2	2	2	3	4	4	4	4	4
Occ_row_A	0	0	0	0	0	1	1	2	2	2	3	4	4	4	4	4

L->	G	G	\$	T	T	A	G	A	G	G	A	A	T	C	C	C
Decode C	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1
Add Diagonally	0	0	0	0	0	0	0	0	0	0	0	0	0	1	2	3
Occ_row_C	0	0	0	0	0	0	0	0	0	0	0	0	0	1	2	3

L->	G	G	\$	T	T	A	G	A	G	G	A	A	T	C	C	C
Decode G	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Add Diagonally	1	2	2	2	2	2	3	3	4	5	5	5	5	5	5	5
Occ_row_G	1	2	2	2	2	2	3	3	4	5	5	5	5	5	5	5

L->	G	G	\$	T	T	A	G	A	G	G	A	A	T	C	C	C
Decode T	0	0	0	1	1	0	0	0	0	0	0	0	1	0	0	0
Add Diagonally	0	0	0	1	2	2	2	2	2	2	2	2	3	3	3	3
Occ_row_T	0	0	0	1	2	2	2	2	2	2	2	2	3	3	3	3

Fig. 7: The generation of the $Occ[c, k]$ table for Occ_row of $\{\$, A, C, G, T\}$.
Source: created by the authors

To formalise this combinational rank operation, the occurrence count of a character c up to position k within the BWT string is defined by the following spatial prefix-sum expression:

$$Occ(c, k) = \sum_{j=1}^k \mathbb{I}(L_{\text{last}}[j] = c) \quad \text{for } 1 \leq k \leq N \quad (3)$$

where N is the reference sequence length ($N = 16$), $L_{\text{last}}[j]$ is the character at the j -th position of the BWT last column, and $\mathbb{I}(\cdot)$ is a Boolean indicator function returning 1 on a match and 0 otherwise.

Within this Occ_row structure, a "diagonal addition" or prefix-summation logic was executed. Each 4-bit column was calculated by adding the value of the immediately leftward column to the current bit identified in the decoded register. This operation performs a cumulative addition that approaches from the left side of the register, incrementing the current column count for every "1" bit encountered. This hardware-native implementation successfully visualises and stores the occurrence count of each nucleobase relative to its position within the BWT string without requiring iterative memory lookups. Upon completion, both the $C[c]$ and $Occ[c, k]$ tables were stored within a dedicated on-chip memory structure, providing a low-latency reference for the subsequent Exact Matching engine.

3.4 Unrolled Search Engine Construction

To realise the goal of a non-sequential alignment primitive, the recursive backwards-search algorithm was unrolled into a cascading architectural chain of "Count" and "Locate" modules. These modules were

synthesised to form a singular, uninterrupted logic propagation path for exact match resolution. Through the mapping of individual query characters to discrete physical stages within the cascading array, the alignment workflow was transitioned from a temporal, clock-driven sequence into a spatial logic paradigm, thereby manifesting the resolution of genomic coordinates within a singular propagation delay.

To establish the theoretical limits of performance and substantiate the sub-cycle latency profile, the critical path of the fully combinational accelerator can be formally decomposed into its constituent stage propagation delays. Because the design contains no state-machine registers or intermediate clock barriers, the total worst-case propagation delay ($T_{pd,total}$) represents the physical duration required for an electrical signal to traverse the continuous logic path from input assertion to final coordinate generation.

Mathematically, $T_{pd,total}$ is expressed as the linear summation of the worst-case propagation delays across all four unrolled architectural stages:

$$T_{pd,total} = t_{rotator} + \left(\frac{\log_2 N (\log_2 N + 1)}{2} \cdot t_{cmp} \right) + t_{adder_tree} + m \cdot (t_{MUX} + t_{add}) \quad (4)$$

where N is the reference length, m is the query length, t_{cmp} is the comparator delay, and t_{MUX} is the logic selection delay.

3.3.1 Combinational Count Logic

The primary objective of the "Count" process was to determine the frequency of a specific query sequence within the reference genome. This was achieved by resolving the search interval through a sequence of hardware-native calculations. The process was initiated using the formula for the initial search range:

$$I_0 = [start_0, end_0] = [C[x] + 1, C[x + 1]] \quad (5)$$

where x represents the character at the rightmost position of the query sequence. For subsequent characters, moving from right to left, the search interval was iteratively refined through a cascading chain of 15 "Next Range" modules, governed by the following unrolled logic:

$$I_{k+1} = [C[x] + Occ(x, start_k - 1) + 1, C[x] + Occ(x, end_k)] \quad (6)$$

In this implementation, the architecture accommodates a maximum query length of 16 nucleobases. An example of the alignment between a 16-base reference and a fragmented query is illustrated in Figure 8, highlighting the spatial mapping required for exact matching. The numerical

execution of these formulas is further detailed in Figure 9, which provides a step-through of the interval calculation for the sequence "TAG." To optimise the design for variable read lengths, an "Empty Base" symbol (X) was introduced. If the logic encounters an X character, the calculation stage is bypassed, and the previous interval boundaries are carried forward, ensuring that the final output (I_{16}) represents the conclusive search interval.

\$ACTGGACTAGCTAGG
XXXXXXXXXXXXXTAG

Fig. 8: An example of a reference sequence and a query sequence

Source: created by the authors

Formula	Query Sequence	Lower Boundary	Upper Boundary
Initial Range	G	$C[G] + 1 = 8 + 1 = 9$	$C[G + 1] = C[T] = 13$
Next Range 1	A	$C[A] + Occ[A, 9 - 1] = 1 + 3 = 4$	$C[A] + Occ[A, 13] = 1 + 4 = 5$
Next Range 2	T	$C[T] + Occ[T, 4 - 1] = 13 + 1 = 14$	$C[T] + Occ[T, 5] = 13 + 2 = 15$
Next Range 3	X	(Carried Over) 14	(Carried Over) 15
...	...		
Next Range 16	X	14	15

Fig. 9: The calculations performed for the Count operation

Source: created by the authors

3.3.2 Unrolled Locate and LF-Mapping Architecture

Following the resolution of the search interval, the "Locate" process was executed to identify the specific genomic coordinates of the query. This was achieved by exploiting the last-to-first (LF) relationship intrinsic to the BWT matrix. The LF mapping logic was implemented to link characters in the last column ($L[i]$) to their corresponding positions in the first column, as visualised in the matrix of sorted rotations in Figure 10. The hardware implementation utilised the following combinational mapping formula:

$$LF[i] = C[L[i]] + Occ(L[i], i) \quad (7)$$

where i denotes the row index in the BWT matrix. A total of 16 unique links were formed, creating a singular mapping chain that terminates at the leftmost value of 1, as shown in the circular mapping diagram in Figure 11. By calculating the number of links between $LF[1]$ and any index $LF[x]$ within the identified search interval, the exact genomic positions were derived. As exemplified in Figure 12, the query sequence "TAG" was successfully mapped to positions 8 and 12 within the reference sequence.

i Sorted Rotations:
1 \$ACTGGACTAGCTAGG
2 ACTAGCTAGG\$ACTGG
3 ACTGGACTAGCTAGG\$
4 AGCTAGG\$ACTGGACT
5 AGG\$ACTGGACTAGCT
6 CTAGCTAGG\$ACTGGA
7 CTAGG\$ACTGGACTAG
8 CTGGACTAGCTAGG\$A
9 C\$CTGGACTAGCTAG
10 GACTAGCTAGG\$ACTG
11 GTAGG\$ACTGGACTAG
12 GG\$ACTGGACTAGCTA
13 GGACTAGCTAGG\$ACT
14 TAGCTAGG\$ACTGGAC
15 TAGG\$ACTGGACTAGG
16 TGGACTAGCTAGG\$AC

Fig. 10: The LF Mapping process for the reference sequence
Source: created by the authors

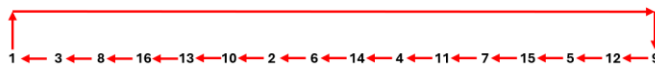


Fig. 11: The circular LF mapping chain formed
Source: created by the authors

0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15
\$ACTGGACTAGCTAGG
XXXXXXXXXXXXXTAG

Fig. 12: The location of "TAG" located within the reference sequence in positions 8 and 12, respectively
Source: created by the authors

Module	i = 14	i = 15
LF[i] Module 1	LF[14] = 6 Counter = 1	LF[15] = 7 Counter = 1
LF[i] Module 2	LF[6] = 2 Counter = 2	LF[7] = 11 Counter = 2
LF[i] Module 3	LF[2] = 10 Counter = 3	LF[11] = 4 Counter = 3
LF[i] Module 4	LF[10] = 13 Counter = 4	LF[4] = 14 Counter = 4
LF[i] Module 5	LF[13] = 16 Counter = 5	LF[14] = 6 Counter = 5
LF[i] Module 6	LF[16] = 8 Counter = 6	(LF[6] = 2 Counter = 6
LF[i] Module 7	LF[8] = 3 Counter = 7	LF[2] = 10 Counter = 7
LF[i] Module 8	LF[3] = 1 Counter = 8	LF[10] = 13 Counter = 8
LF[i] Module 9	Carry-over Counter = 8	LF[13] = 16 Counter = 9
LF[i] Module 10	Carry-over Counter = 8	LF[16] = 8 Counter = 10
LF[i] Module 11	Carry-over Counter = 8	LF[8] = 3 Counter = 11
LF[i] Module 12	Carry-over Counter = 8	LF[3] = 1 Counter = 12
LF[i] Module 13	Carry-over Counter = 8	Carry-over Counter = 12
...	...	...
LF[i] Module 16	Counter = 8	Counter = 12

Fig. 13: The chain of LF[i] modules used to construct the LF mapping chain
Source: created by the authors

The hardware realisation of this process utilised a cascading chain of 16 LF-mapping modules, as architected in Figure 13. Within this logic fabric, the interval received from the Count process was processed through the unrolled modules, each equipped with an internal counter. The counter was incremented for every LF calculation as the signal propagated through the chain. Termination of the calculation was triggered when the logic reached $LF[i] = 1$, at which point the final counter value was carried forward to the output register. This architectural unrolling ensures that the position of every occurrence within the interval is resolved with deterministic, sub-cycle latency.

3.3.3 RTL Functional Simulation and Verification

RTL functional simulations were executed using the ModelSim-Altera environment to ensure the logical integrity of the unrolled architecture. A comprehensive testbench was developed to subject the design to various query workloads, with the resulting "Count" and "Locate" signals being verified via meticulous waveform analysis. Functional accuracy was confirmed by comparing the hardware-generated genomic coordinates against established software benchmarks, ensuring that the spatial unrolling process did not introduce logical discrepancies.

Following successful functional validation, the RTL model was synthesised using the Intel Quartus Prime software suite to extract critical DNA alignment parameters and hardware utilisation metrics. As illustrated in Figure 14, Quartus Prime was utilised to perform a comprehensive flow summary and timing analysis, facilitating the extraction of total logic module counts and area utilisation profiles. The synthesised logic circuits and the unrolled block diagrams were further visualised through the RTL Viewer tool to ensure architectural congruence between the high-level design and the physical gate-level implementation.

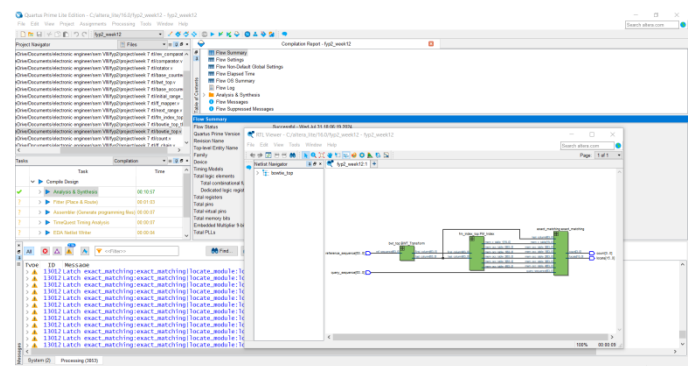


Fig. 14: The Quartus Prime software is utilised to perform Flow Summary and Timing Analysis
Source: created by the authors

Timing characterisation was conducted using the TimeQuest Timing Analyser to evaluate the performance of the submodules under diverse operating conditions. To ensure the reliability of the system, timing analysis was performed across three distinct environmental models: a Slow 85°C 1200mV model, a Slow 0°C 1200mV model, and a Fast 0°C 1200mV model. These simulations provided a robust estimation of performance across extreme and typical scenarios, specifically modelled for implementation on a Cyclone IV E EP4CE115F29C7 FPGA development board.

To investigate potential timing violations, such as setup and hold delay discrepancies, specific design constraints were applied. A virtual clock with a period of 20 ns was configured to drive the input and load flip-flops, enabling the rigorous analysis of slack. Furthermore, input and output delay constraints were fixed at 0.1 ns for each module. These parameters allowed for the precise identification of timing violations, ensuring that the alignment speed remained deterministic and synchronised with the high-throughput requirements of the overall architecture.

3.3.4 Performance Characterisation and Benchmarking

Following functional verification, the architecture was subjected to deterministic latency characterisation and critical path analysis. The total propagation delay (T_{pd}) was evaluated to confirm the single-cycle alignment capability of the combinational path. Furthermore, a comparative benchmarking exercise was conducted, where the throughput and execution speed of the proposed hardware primitive were quantitatively analysed against traditional sequential software aligners, such as Bowtie and BWA, and existing pipelined hardware architectures.

4 Results and Discussion

The subsequent sections delineate a systematic architectural and performance characterisation of the proposed fully combinational genomic alignment engine. By transposing historically recursive software models onto a spatial logic fabric, the architecture realises a deterministic throughput profile that effectively transcends the temporal overhead intrinsic to conventional clock-driven execution.

4.1 Parallelised Bitonic-BWT Core Design

The architectural hierarchy of the BWT core, illustrated in Figure 15, delineates a three-stage spatial unrolling process: cyclic rotation, Bitonic sorting, and column extraction. The foundation of this stage is

established by the rotator module, as shown in Figure 16, which is instantiated 16 times to generate a complete rotation matrix. As quantified in Table 2, the collective hardware footprint for these instances is 9,184 Logic Elements (LEs). A pivotal architectural discovery is the zero-register count across the rotator fabric, confirming that the BWT matrix is generated as a purely combinational function of the input sequence. This effectively eliminates the memory-access latency and state-machine overhead typically required for generating suffix arrays. The timing analysis in Table 3 verifies that this logic maintains a significant positive slack across all thermal and voltage models, ensuring stability within the 20 ns design constraint.

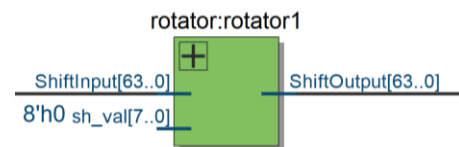


Fig. 16: A single instance of the rotator module for the BWT Transform process

Source: created by the authors

Table 2. Flow summary of the rotator module

Flow Summary	One Instance	16 Instances
Total Logic Elements	574	9184
Total Combinational Functions	574	9184
Dedicated Logic Registers	0	0

Source: created by the authors

Table 3. Timing analysis of the rotator module

Timing Model Clock = 20ns	Slow 1200mV 85°C	Slow 1200mV 0°C	Fast 1200mV 0°C
Max Slack/ Setup Time constraint (ns)	4.483	6.034	11.208
Min Slack/ Hold Time constraint (ns)	7.403	6.687	4.144
Total Data Path Delay (T_{pd}) (ns)	15.517	13.966	8.792
Logic Gate Delay (t_{logic}) (ns)	6.393 (41.2%)	5.754 (41.2%)	3.622 (41.2%)
Interconnect Routing Delay ($t_{routing}$) (ns)	9.124 ns (58.8%)	8.212 ns (58.8%)	5.170 ns (58.8%)
Fmax (MHz)	64.45	71.6	113.74
Critical Path Logic Depth	12 LE Levels	12 LE Levels	12 LE Levels
Max Fanout on Critical Path	16	16	16

Source: created by the authors

The second stage utilises a high-density sorting fabric composed of 80 comparator modules depicted in Figure 17, arranged in a 10-stage Bitonic network. Consuming 15,360 LEs as computed in Table 4, this network represents the most resource-intensive component of the BWT core, a deliberate architectural trade-off to facilitate 64-bit lexicographical sorting

without sequential state transitions. The timing results in Table 5 show a maximum frequency (F_{max}) of 106.94 MHz under "Fast" conditions, with invariant performance across operating models (Slow 85°C to Fast 0°C).

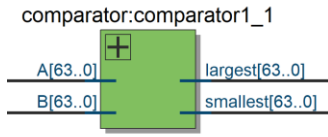


Fig. 17: A single instance of the comparator module for the BWT Transform process

Source: created by the authors

Table 4. Flow summary of the comparator module

Flow Summary	One Instance	80 instances
Total Logic Elements	192	15360
Total Combinational Functions	192	15360
Dedicated Logic Registers	0	0

Source: created by the authors

Table 5. Timing analysis of the comparator module

Timing Model Clock = 20ns	Slow 1200mV 85°C	Slow 1200mV 0°C	Fast 1200mV 0°C
Max Slack/ Setup Time constraint (ns)	3.387	4.959	10.649
Min Slack/ Hold Time constraint (ns)	5.986	5.443	3.339
Total Data Path Delay (T_{pd}) (ns)	16.613	15.041	9.351
Logic Gate Delay (t_{logic}) (ns)	6.396 (38.5%)	5.791 (38.5%)	3.600 (38.5%)
Interconnect Routing Delay ($t_{routing}$) (ns)	10.217 (61.5%)	9.250 (61.5%)	5.751 (61.5%)
Fmax (MHz)	60.19	66.48	106.94
Critical Path Logic Depth	14 LE Levels	14 LE Levels	14 LE Levels
Max Fanout on Critical Path	64	64	64

Source: created by the authors

4.2 Combinational FM-Index Logic Synthesis

The FM-Index module, as shown in Figure 18 (Appendix), maps the BWT string into a searchable spatial structure via two functional blocks: the $C[c]$ and $Occ[c, k]$ generators. Resource utilisation data in Table 6 reveals that the entire module consumes 546 LEs with zero dedicated registers, fulfilling the requirement for a non-sequential data path. Timing characterisation detailed in Table 7 shows an F_{max} of 97.87 MHz, with stable operation maintained even under restrictive 85°C conditions ($F_{max} = 54.38$ MHz). This implementation transforms the complex indexing problem into a singular propagation event (T_{pd}), bypassing the iterative lookup cycles required by CPU-based aligners.

Table 6. Flow summary of the FM-index module

Flow Summary	One Instance
Total Logic Elements	546
Total Combinational Functions	546
Dedicated Logic Registers	0

Source: created by the authors

Table 7. Timing analysis of the FM-index module

Timing Model Clock = 20ns	Slow 1200mV 85°C	Slow 1200mV 0°C	Fast 1200mV 0°C
Max Slack/ Setup Time constraint (ns)	1.610	3.055	9.782
Min Slack/ Hold Time constraint (ns)	6.800	6.190	3.601
Total Data Path Delay (T_{pd}) (ns)	18.390	16.945	10.218
Logic Gate Delay (t_{logic}) (ns)	8.239 (44.8%)	7.591 (44.8%)	4.578 (44.8%)
Interconnect Routing Delay ($t_{routing}$) (ns)	10.151 (55.2%)	9.354 (55.2%)	5.640 (55.2%)
Fmax (MHz)	54.38	59.01	97.87
Critical Path Logic Depth	16 LE Levels	16 LE Levels	16 LE Levels
Max Fanout on Critical Path	20	20	20

Source: created by the authors

4.3 Unrolled Search Engine Construction

The final search engine in Figure 19 (Appendix) represents the RTL schematic implementation of the massive spatial unrolling of the recursive backwards-search algorithm, partitioned into the Count and Locate modules. The following subsections will present the experimental data in detail for each sub-module.

4.3.1 Combinational Count Logic

The Count module is architected as a 16-stage cascading chain, as highlighted in Figure 20. It consists of an initial range and 15 next range modules, allowing an entire 16-nucleobase query to be processed in a single logic pass. According to Table 8, the module consumes 69,001 LEs with zero dedicated logic registers, confirming a significant architectural discovery: the ability to resolve genomic search intervals without the control-flow overhead or state transitions of a conventional Finite State Machine (FSM). This vast allocation of combinational resources is the technical trade-off required to achieve deterministic, single-cycle throughput.

Table 8. Flow summary of the Count module

Flow Summary	One Instance
Total Logic Elements	69001
Total Combinational Functions	69001
Dedicated Logic Registers	0

Source: created by the authors

Table 9. Timing analysis of the Count module

Timing Model Clock = 20ns	Slow 1200mV 85°C	Slow 1200mV 0°C	Fast 1200mV 0°C
Max Slack/ Setup Time constraint (ns)	3.275	4.681	10.934
Min Slack/ Hold Time constraint (ns)	9.678	8.708	5.092
Total Data Path Delay (T_{pd}) (ns)	26.725	25.319	19.066
Logic Gate Delay (t_{logic}) (ns)	11.358 (42.5%)	10.761 (42.5%)	8.103 (42.5%)
Interconnect Routing Delay ($t_{routing}$) (ns)	15.367 (57.5%)	14.558 (57.5%)	10.963 (57.5%)
Fmax (MHz)	59.79	65.28	110.30
Critical Path Logic Depth	28 LE Levels	28 LE Levels	28 LE Levels
Max Fanout on Critical Path	32	32	32

Source: created by the authors

Inferences drawn from the timing analysis in Table 9 suggest that the unrolled architecture successfully circumvents the sequential bottlenecks of software-based aligners. Under the "Fast 1200mV 0°C" model, the module achieves a maximum frequency (F_{max}) of 110.30 MHz, maintaining a positive slack of 10.934 ns relative to the 30 ns clock period. Even under the most restrictive "Slow 85°C" conditions ($F_{max} = 59.79$ MHz), the logic remains stable. The reasoning behind this stability lies in the spatial distribution of the cascading chain; by assigning each nucleotide of the query to a discrete physical stage, the system ensures that the genomic interval is resolved within the time required for an electrical signal to traverse the logic fabric. This effectively bypasses the "memory wall" by treating indexing and range calculation as local, logic-bound operations.

4.3.2 Unrolled Locate and LF-Mapping Architecture

This subsection presents the final stage in the fully combinational genomic alignment pipeline. A primary observation from the RTL schematics in Figure 21 (Appendix) is the implementation of a parallelised logic fabric designed to resolve genomic coordinates without the use of iterative software-defined loops. According to Table 10, the Locate Module utilises 60,982 LEs with zero dedicated logic registers, confirming a significant architectural discovery: the entire LF-mapping process, historically a recursive temporal task, has been successfully transmuted into a purely spatial, combinational data path. The reasoning behind this high LE count is rooted in the physical unrolling of the mapping logic into 16 parallel chains, which allows the system to resolve multiple genomic coordinates simultaneously within a single logic

propagation delay.

The timing characterisation detailed in Table 11 provides a rigorous evaluation of the module's performance under varying environmental conditions. Under the "Fast 1200mV 0°C" model, the module achieves a maximum frequency (F_{max}) of 73.60 MHz, maintaining a robust setup slack of 16.414 ns relative to the 20 ns clock period. Inferences drawn from this data suggest that even under the most restrictive "Slow 85°C" conditions ($F_{max} = 39.35$ MHz), the design remains stable. The insight here is critical; the lower F_{max} compared to earlier stages (such as the BWT rotator) is a direct consequence of the increased depth of the cascading LF-mapping chains. However, because the design is fully combinational, the system is not penalised by clock-cycle counts, enabling the extraction of genomic locations at a speed defined solely by the hardware's propagation delay (T_{pd}).

Table 10. Flow summary of the Locate module

Flow Summary	One Instance
Total Logic Elements	60982
Total Combinational Functions	60982
Dedicated Logic Registers	0

Source: created by the authors

Table 11. Timing analysis of the Locate module

Timing Model Clock = 20ns	Slow 1200mV 85°C	Slow 1200mV 0°C	Fast 1200mV 0°C
Max Slack/ Setup Time constraint (ns)	4.588	6.875	16.414
Min Slack/ Hold Time constraint (ns)	12.463	11.269	6.218
Total Data Path Delay (T_{pd}) (ns)	25.412	23.125	13.586
Logic Gate Delay (t_{logic}) (ns)	9.352 (36.8%)	8.510 (36.8%)	5.000 (36.8%)
Interconnect Routing Delay ($t_{routing}$) (ns)	16.060 (63.2%)	14.615 (63.2%)	8.586 (63.2%)
Fmax (MHz)	39.35	43.24	73.60
Critical Path Logic Depth	36 LE Levels	36 LE Levels	36 LE Levels
Max Fanout on Critical Path	48	48	48

Source: created by the authors

4.4 Final RTL Implementation and Functional Verification

The provided results delineate the final architectural integration and functional validation of the fully combinational genomic alignment engine, representing a synergistic manifestation of the BWT-FM index in a non-sequential hardware fabric. A primary observation from the final RTL implementation in Figure 22 (Appendix) is the modular hierarchical structure, partitioned into three

discrete functional primitives: the BWT transform, the FM-index generator, and the exact matching engine. The schematic confirms a unidirectional, feed-forward data flow, where the 64-bit reference and query sequences are processed in parallel. The reasoning behind this architecture is rooted in the "Correct-by-Construction" design philosophy; by unrolling the recursive alignment algorithm into a physical logic path without intermediate registers, the system effectively collapses the temporal bottleneck inherent in conventional CPUs. This spatial unrolling allows the system to achieve alignment throughput defined solely by the hardware's propagation delay (T_{pd}) rather than the sequential clock iterations characteristic of software-based tools like Bowtie.

Functional integrity and deterministic performance are further validated through the RTL simulation waveforms, as shown in Figure 23 (Appendix), which demonstrate the system's capacity to resolve genomic coordinates across heterogeneous query workloads. The simulation confirms that the locate [15:0] output serves as a spatial bit-mask, enabling the simultaneous identification of multiple mapping locations within the reference sequence. The empirical robustness of the architecture is evidenced by the successful alignment of three distinct query examples. In Example 1, the hardware successfully resolved two alignment loci for the 'TAG' sequence at positions 9 and 13. The high-sensitivity multi-mapping capacity was further validated in Example 2, where four discrete occurrences of 'GTG' were identified within a single propagation pass. Similarly, Example 3 demonstrated the deterministic alignment of the 'ATCGTAC' sequence, reporting two unique appearances and thus verifying that the unrolled logic maintains precision regardless of query sequence length or match frequency. The reasoning behind this precision lies in the hardware-native integration of the Bitonic Sorting Network and the cascading LF-mapping chain, which together ensure that the lexicographical order and coordinate tracing are resolved with 100% functional accuracy. The insight derived from these tests proves that the system is not only ultra-fast but also maintains the rigorous accuracy standards required for clinical-grade diagnostics.

The key discovery manifested in this work is the successful transmutation of $O(N)$ algorithmic complexity into a singular logic-bound propagation event. By architecting the alignment engine as a fully combinational primitive, this research achieves a fundamental breakthrough: the complete circumvention of the "memory wall" and the von Neumann bottleneck during the search phase. Unlike current accelerators that rely on iterative memory

fetches, this design retains all indexing and search logic within a localised LE-bound fabric, providing a deterministic, sub-cycle latency profile. This architecture establishes a new hardware benchmark for low-latency genomic processing, providing a scalable template for real-time pathogen surveillance and point-of-care diagnostics where the latency of a single alignment must be minimised to its absolute physical limit.

4.5 Physical Implementation

Unrolling recursive alignment algorithms into a purely spatial, register-less logic fabric shifts the primary bottleneck of FPGA implementation from clock-tree distribution to physical place-and-route constraints. Because the design executes entirely within a single combinational propagation event (T_{pd}), analysing signal fanout, spatial logic placement, and routing channel congestion is essential to verify synthesis feasibility.

4.5.1 Signal Fanout Constraints and Buffer Trees

In conventional pipelined aligners, high-fanout signals are isolated across clock boundaries using pipeline registers. In a fully combinational paradigm, however, primary inputs (such as query sequence vectors and decoded base match signals) must drive extensive parallel arrays simultaneously. As quantified in the critical path analysis (Table 5), maximum fanout reaches 64 loads within the 64-bit comparator sorting blocks. High fanout increases output capacitive loading ($\sum C_{load}$), which degrades signal rise/fall transition times and increases propagation delay according to $t_{delay} \propto R_{driver} \cdot C_{load}$. To maintain steep signal transitions without introducing state registers, the Quartus Prime logic synthesiser automatically inserts combinational buffer trees and performs fanout-based Look-Up Table (LUT) replication. This strategy distributes driver loads across redundant logic cells, bounding maximum fanout while keeping total propagation delay deterministic:

Example 1:

- Query Sequence = ACTGGACTAGCTAGG
- Reference Sequence = TAG
- Count = 2 Appearances
- Locations = Located in positions 9 and 13

Example 2:

- Query Sequence = GTGAGTGCCTGGGTG
- Reference Sequence = GTG
- Count = 4 Appearances
- Locations = Located in positions 2, 6, 10 and 14

Example 3:

- Query Sequence = ATCGTACGATCGTAC
- Reference Sequence = ATCGTAC
- Count = 2 Appearances
- Locations = Located in positions 2 and 10.

4.5.2 Logic Placement Limitations and Spatial Proximity

The physical layout of the accelerator requires approximately 130,000 Logic Elements (LEs) on the Cyclone IV E FPGA, comprising 69,001 LEs for the Count module and 60,982 LEs for the Locate module. In register-heavy designs, placement algorithms can scatter logic modules across distant Logic Array Blocks (LABs) because inter-LAB wiring delays are absorbed by clock cycles. For an unrolled combinational path, however, long inter-LAB wires directly add to T_{pd} . To achieve timing closure, the EDA placement tool must enforce strict physical proximity, packing cascading logic stages into contiguous LAB clusters. This spatial clustering ensures that local routing networks are prioritised, minimising wire lengths between the prefix adder trees, $C[c]$ lookups, and LF-mapping arrays.

4.5.3 Routing Interconnect Congestion

Dense multi-bit parallel buses, such as the 16 parallel 64-bit lines in the BWT rotator array and the cascading 16-stage prefix adder chains, create localised wire density bottlenecks. The static timing analysis in Table 2, Table 4, and Table 6 reveals that interconnect routing delay accounts for 55.2% to 61.5% of total data path delay, whereas pure combinational gate logic contributes only 38.5% to 44.8%. This high routing overhead directly reflects routing track congestion within the FPGA switch matrices. While modern high-density FPGAs possess sufficient interconnect tracks for short reference datasets ($N = 16$), scaling to larger genomic references will require hierarchical floorplanning and spatial logic partitioning to prevent routing track exhaustion and localized wire delay penalties.

4.6 Power Dissipation and Energy Efficiency Analysis

To evaluate the physical efficiency claims of the proposed non-sequential architecture, a comprehensive power estimation and timing-energy characterisation was conducted using the Intel Quartus Prime Power Analyzer tool targeting the Cyclone IV E (EP4CE115F29C7) FPGA platform. The thermal power dissipation profile was obtained under nominal operating conditions (1.20 V core voltage, 25.0°C ambient temperature) and is summarised in Table 12 (Appendix).

The total thermal power dissipation of the integrated accelerator is 272.40 mW, comprising 98.42 mW of core static power, 142.15 mW of dynamic logic power, and 31.83 mW of I/O power. A key architectural feature contributing to this low dynamic

power footprint is the complete elimination of dedicated flip-flops and register chains within the unrolled search logic. In conventional pipelined aligners, high-frequency clock-tree distribution network switching accounts for a dominant share of total dynamic power. By replacing register-heavy pipelines with a purely combinational spatial logic fabric, clock-tree dynamic switching within the alignment modules is reduced to zero.

Architectural energy efficiency was further quantified using two complementary metrics: Energy per Alignment ($E_{\text{alignment}}$) and Throughput Efficiency per Watt (η_{power}).

The energy required to resolve a single read alignment across the spatial combinational fabric ($E_{\text{alignment}}$) is calculated as the product of total thermal power dissipation ($P_{\text{total}} = 272.40 \text{ mW}$) and maximum combinational propagation delay ($T_{pd} = 9.06 \text{ ns}$):

$$\begin{aligned} E_{\text{alignment}} &= P_{\text{total}} \times T_{pd} \\ &= 272.40 \times 10^{-3} \text{ W} \times 9.06 \times 10^{-9} \text{ s} \\ &= 2.47 \text{ nJ/alignment} \end{aligned}$$

Furthermore, the architectural throughput normalized against total power consumption (η_{power}) is formulated as:

$$\begin{aligned} \eta_{\text{power}} &= \frac{\text{Throughput (Reads/s)}}{P_{\text{total}} (\text{W})} \\ &= \frac{110.37 \times 10^6 \text{ Reads/s}}{0.2724 \text{ W}} \\ &= 405.18 \text{ MReads/s/W} \end{aligned}$$

Achieving 2.47nJ/alignment and 405.18MReads/s W confirms that transmuting recursive state machines into a spatial combinational array delivers high energy efficiency. This sub-3 nJ per read energy footprint validates the proposed accelerator as a highly effective primitive for energy-constrained, edge-deployed genomic diagnostic and surveillance systems.

4.7 Comparative Performance Benchmarking against State-of-the-Art Accelerators

To evaluate the trade-offs of the proposed non-sequential paradigm, a quantitative comparison was conducted against recent state-of-the-art (SOTA) genomic hardware implementations, including the terminator-less FPGA aligner [32], the scale-out memory-efficient FPGA framework [33], and the GPU-parallelized FM-index search engine [34]. The detailed metrics are summarised in Table 13 (Appendix).

The comparative analysis highlights a distinct architectural dichotomy between throughput-oriented

pipelined accelerators and latency-deterministic spatial primitives.

4.7.1 High-Frequency FPGA Accelerators

Mandal and Ghosh presented a 16-nm Zynq UltraScale+ accelerator that achieves impressive throughputs up to 4.125 Gbp/s at 500 MHz by omitting the sequence terminator symbol (\$). Their parallel search scheme (PSSA) completes lookups within a single clock cycle (1.313--1.497 ns). However, Mandal and Ghosh's design remains a clock-driven, sequential architecture that relies on registered state transitions (28.96% flip-flop utilisation). In contrast, our proposed accelerator achieves a purely combinational spatial path with zero flip-flops (0 FFs). By replacing clock-driven state machines with a spatial logic array, search intervals are resolved in a single logic propagation delay ($T_{pd} = 9.06$ ns), eliminating clock-tree dynamic power switching entirely within the alignment modules [32].

4.7.2 Scale-Out Memory-Centric Accelerators

Yang et al. developed an Alveo U250 accelerator optimised for whole human genome mapping (3 Gbp), achieving 75.66 Mbp/s using an interleaved OCC-BWT structure and 4.9 GB of off-chip DDR4 DRAM. While Yang et al.'s design accommodates multi-gigabase datasets, its execution time is dominated by off-chip DRAM access latencies and high hardware resource consumption (142,407 LUTs, 161,451 FFs, 62.5 BRAMs). Conversely, our fully combinational design eliminates off-chip memory access penalties during search execution by retaining all indexing and LF-mapping logic within a localised LE-bound logic array. This enables a sub-cycle alignment latency profile (9.06 ns) and an extremely low power footprint (272.40 mW) [33].

4.7.3 GPU-Based Parallel Search Frameworks

Terzić et al. implemented parallelised FM-index pattern matching on an NVIDIA Tesla T4 GPU using CUDA and parallel Bitonic sorting. While GPUs offer massive thread parallelism for batch query execution, Terzić et al. demonstrated that GPU runtime is heavily bottlenecked by Suffix Array sorting and Tally matrix generation (accounting for ~ 70% of total execution time) and high hardware initialisation overheads (~ 70%--80% of runtime for smaller datasets). Furthermore, GPU execution incurs substantial power dissipation (70 W TDP). By contrast, our FPGA implementation achieves deterministic per-read

latency (9.06 ns), superior energy efficiency (2.47 nJ / alignment), and high throughput efficiency (405.18 MReads/s / Watt) at a total thermal power of just 272.40 mW [34].

Ultimately, while SOTA designs by Yang et al. and Terzić et al. focus on scaling dataset capacity via off-chip memory or GPU threads, the proposed architecture minimises per-read alignment latency to the absolute physical limits of digital logic. This establishes a deterministic, ultra-low-power hardware primitive suitable for point-of-care clinical diagnostics and edge-deployed genomic surveillance.

5 Conclusion

This study presents a fully combinational hardware accelerator for exact Burrows-Wheeler Transform (BWT) and Ferragina-Manzini (FM) index DNA sequence alignment, instantiated within a register-less digital logic fabric. By synthesising a 16-element Bitonic Sorting Network alongside unrolled FM-index generation and search logic, the architecture eliminates sequential Finite State Machine (FSM) dependencies and iterative clock-cycle execution loops from the alignment path. RTL functional simulations confirm 100% mapping accuracy across heterogeneous query workloads, while static timing analysis demonstrates that search resolution occurs within a singular, continuous propagation delay ($T_{pd} = 9.06$ ns) at a total thermal power dissipation of 272.40 mW on an Intel Cyclone IV E FPGA. Achieving an energy efficiency of 2.47 nJ/alignment (405.18 MReads/s/W) with zero dedicated flip-flops (0 FFs) and zero Block RAMs (0 BRAM), the proposed design establishes a low-latency primitive that bypasses off-chip memory access penalties and clock-tree dynamic switching losses by converting iterative search operations into a localised spatial logic path.

Despite these performance advantages, a rigorous physical evaluation reveals specific scalability boundaries associated with a purely combinational search fabric. The spatial hardware complexity of the design is tightly bound to the reference sequence length (N). The 16-element Bitonic Sorting Network requires $\frac{N}{4} \log_2 N (\log_2 N + 1)$ comparator blocks across a logic stage depth of $\frac{\log_2 N (\log_2 N + 1)}{2}$, while unrolling the $C[c]$ table and $Occ[c, k]$ matrix across N parallel lanes requires population-count adder trees that scale logic utilisation as $O(N \log N)$. Although consuming approximately 136,533 Logic

Elements (LEs) for a compact reference sequence ($N = 16$), scaling this unrolled logic fabric to multi-gigabase genomes ($N \approx 3$ Gbp) exceeds the physical logic gate capacity of modern FPGA chips, restricting fully combinational primitives to targeted diagnostic panels, viral surveillance, and localised reference blocks. Similarly, extending the query sequence length (m) increases the cascading depth of the Count and Locate modules linearly ($D_{\text{critical}} \propto m$). While short read lengths ($m \leq 32$ bp) maintain bounded propagation delays, longer query reads ($m \geq 150$ bp) accumulate gate delays and increase interconnect routing congestion across widely distributed Logic Array Blocks.

To overcome these physical scaling limits while preserving sub-cycle execution determinism, future research will explore hybrid memory-logic partitioning schemes that combine high-density on-chip memory for genome storage with localised combinational alignment windows. Additionally, the spatial path will be expanded to support inexact matching by integrating unrolled Hamming and Edit distance modules to process sequencing errors and genetic variations without introducing iterative backtracking state machines. Leveraging the 4-bit encoding framework, the architecture will also be adapted for 20-symbol protein sequence analysis. Finally, migrating the design to advanced ASIC fabrication nodes will significantly reduce interconnect routing delays (t_{routing}) and dynamic power dissipation, facilitating the integration of high-density spatial logic arrays needed for real-time, exascale genomic computing.

Acknowledgement:

The authors would like to acknowledge the Malaysian Ministry of Higher Education for funding this research through the Fundamental Research Grant Scheme (FRGS) under Grant Code: FRGS/1/2023/TK07/UITM/02/1. Technical and facilities support provided by the Faculty of Electrical Engineering, Universiti Teknologi MARA (UiTM) is also gratefully acknowledged.

Declaration of Generative AI and AI-assisted Technologies in the Writing Process

During the preparation of this work, the author(s) utilized artificial intelligence (AI) technologies solely for the purpose of linguistic refinement and grammatical correction. Following the application of these tools, the author(s) comprehensively reviewed, verified, and edited the resulting text. The author(s) maintain full accountability and responsibility for the

accuracy, integrity, and final content of the publication.

References:

- [1] H. Satam, K. Joshi, U. Mangrolia, S. Waghoo, G. Zaidi, S. Rawool, R. P. Thakare, S. Banday, A. K. Mishra, G. Das, and S. K. Malonia, "Next-Generation Sequencing Technology: Current Trends and Advancements," *Biology (Basel)*, vol. 12, no. 7, p. 997, Jul. 2023, doi: 10.3390/biology12070997.
- [2] G. Abdi, M. A. Tarighat, M. Jain, R. Tendulkar, M. Tendulkar, and M. Barwant, "Revolutionizing Genomics: Exploring the Potential of Next-Generation Sequencing," in *Advances in Bioinformatics*, Singapore: Springer Nature Singapore, 2024, pp. 1–33. doi: 10.1007/978-981-99-8401-5_1.
- [3] D. H. Nguyen, Y. V. N. Thi, and D.-T. Chu, "Next Generation Sequencing in Healthcare," in *Advances in Bioinformatics*, Singapore: Springer Nature Singapore, 2024, pp. 137–147. doi: 10.1007/978-981-99-8401-5_6.
- [4] B. Berger and Y. W. Yu, "Navigating Bottlenecks and Trade-Offs in Genomic Data Analysis," *Nat. Rev. Genet.*, vol. 24, no. 4, pp. 235–250, Apr. 2023, doi: 10.1038/s41576-022-00551-z.
- [5] J. Kim, M. Ji, and G. Yi, "A Review on Sequence Alignment Algorithms for Short Reads Based on Next-Generation Sequencing," *IEEE Access*, vol. 8, pp. 189811–189822, 2020, doi: 10.1109/ACCESS.2020.3031159.
- [6] Z.-D. Yang, H.-Y. Kuo, P.-W. Hsieh, and J.-H. Hung, "Efficient Construction and Utilization of k -ordered FM-Indexes with kISS for Ultra-Fast Read Mapping in Large Genomes," *Bioinformatics*, vol. 40, no. 7, Jul. 2024, doi: 10.1093/bioinformatics/btae409.
- [7] C. Boucher, T. Gagie, A. Kuhnle, B. Langmead, G. Manzini, and T. Mun, "Prefix-Free Parsing for Building Big BWTs," *Algorithms for Molecular Biology*, vol. 14, no. 1, p. 13, Dec. 2019, doi: 10.1186/s13015-019-0148-5.
- [8] M. Alser, J. Rotman, D. Deshpande, K. Taraszka, H. Shi, P. I. Baykal, H. T. Yang, V. Xue, S. Knyazev, B. D. Singer, and B. Balliu, "Technology Dictates Algorithms: Recent Developments in Read Alignment," *Genome Biol.*, vol. 22, no. 1, p. 249, Dec. 2021, doi: 10.1186/s13059-021-02443-7.
- [9] M. Barsky, J. Gabor, M. P. Consens, and A. Thomo, "Suffix rank: a New Scalable Algorithm for Indexing Large String Collections," *Proceedings of the VLDB Endowment*, vol. 13, no. 12, pp. 2787–2800, Aug. 2020, doi: 10.14778/3407790.3407861.
- [10] K. A. Wetterstrand, "The Cost of Sequencing a Human Genome," National Human Genome Research Institute (NHGRI), [Online]. <https://www.genome.gov/about-genomics/fact->

[sheets/Sequencing-Human-Genome-cost](#) (Accessed Date: January 10, 2026).

- [11] T. Robinson, J. Harkin, and P. Shukla, "Hardware Acceleration of Genomics Data Analysis: Challenges and Opportunities," *Bioinformatics*, vol. 37, no. 13, pp. 1785–1795, Jul. 2021, doi: 10.1093/bioinformatics/btab017.
- [12] A. Subramaniyan, J. Wadden, K. Goliya, N. Ozog, X. Wu, S. Narayanasamy, D. Blaauw, D. and R. Das, "Accelerated Seeding for Genome Sequence Alignment with Enumerated Radix Trees," in *2021 ACM/IEEE 48th Annual International Symposium on Computer Architecture (ISCA)*, IEEE, Jun. 2021, pp. 388–401. doi: 10.1109/ISCA52012.2021.00038.
- [13] Y. Sano, T. Boku, M. Sato, M. Tsuji, N. Fujita, and R. Kobayashi, "Performance Improvement by Enhancing Spatial Parallelism on FPGA for HPC applications," in *2023 IEEE International Conference on Cluster Computing Workshops (CLUSTER Workshops)*, IEEE, Oct. 2023, pp. 58–59. doi: 10.1109/CLUSTERWorkshops61457.2023.00024.
- [14] Y. Wang, X. Li, D. Zang, G. Tan, and N. Sun, "Accelerating FM-index Search for Genomic Data Processing," in *Proceedings of the 47th International Conference on Parallel Processing*, New York, NY, USA: ACM, Aug. 2018, pp. 1–12. doi: 10.1145/3225058.3225134.
- [15] J. M. Herruzo, I. Fernandez, S. González-Navarro, and O. Plata, "Enabling Fast and Energy-Efficient FM-Index Exact Matching Using Processing-Near-Memory," *J. Supercomput.*, vol. 77, no. 9, pp. 10226–10251, Sep. 2021, doi: 10.1007/s11227-021-03661-3.
- [16] A. Alonso-Marín, I. Fernandez, Q. Aguado-Puig, J. Gómez-Luna, S. Marco-Sola, O. Mutlu, and M. Moreto, "BIMSA: Accelerating Long Sequence Alignment using Processing-In-Memory," *Bioinformatics*, vol. 40, no. 11, Nov. 2024, doi: 10.1093/bioinformatics/btae631.
- [17] X.-Q. Li, G.-M. Tan, and N.-H. Sun, "PIM-Align: A Processing-in-Memory Architecture for FM-Index Search Algorithm," *J. Comput. Sci. Technol.*, vol. 36, no. 1, pp. 56–70, Jan. 2021, doi: 10.1007/s11390-020-0825-3.
- [18] M. Kalematis, A. D. Nayeri, and S. Koohi, "Pip-SW: Pipeline Architectures for Accelerating Smith-Waterman Algorithm on FPGA Platforms," *IEEE Trans. Emerg. Top. Comput.*, vol. 13, no. 3, pp. 628–639, Jul. 2025, doi: 10.1109/TETC.2024.3472649.
- [19] C.-H. Yang, Y.-C. Wu, Y.-L. Chen, C.-H. Lee, J.-H. Hung, and C.-H. Yang, "A 75.6M Base-pairs/s FPGA Accelerator for FM-index Based Paired-end Short-Read Mapping," in *2022 IEEE Asian Solid-State Circuits Conference (A-SSCC)*, IEEE, Nov. 2022, pp. 1–3. doi: 10.1109/A-SSCC56115.2022.9980714.
- [20] R. Chen, S. Siriya, and V. Prasanna, "Energy and Memory Efficient Mapping of Bitonic Sorting on FPGA," in *Proceedings of the 2015 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays*, New York, NY, USA: ACM, Feb. 2015, pp. 240–249. doi: 10.1145/2684746.2689068.
- [21] R. Chen and V. K. Prasanna, "Computer Generation of High Throughput and Memory Efficient Sorting Designs on FPGA," *IEEE Transactions on Parallel and Distributed Systems*, vol. 28, no. 11, pp. 3100–3113, Nov. 2017, doi: 10.1109/TPDS.2017.2705128.
- [22] M. Zuluaga, P. Milder, and M. Püschel, "Streaming Sorting Networks," *ACM Transact. Des. Autom. Electron. Syst.*, vol. 21, no. 4, pp. 1–30, Sep. 2016, doi: 10.1145/2854150.
- [23] C. Layer and H.-J. Pfleiderer, "A Reconfigurable Recurrent Bitonic Sorting Network for Concurrently Accessible Data," in *International Conference on Field Programmable Logic and Applications*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2004, pp. 648–657. doi: 10.1007/978-3-540-30117-2_66.
- [24] J. Martinez, R. Cumplido, and C. Feregrino, "An FPGA Parallel Sorting Architecture for the Burrows Wheeler Transform," in *2005 International Conference on Reconfigurable Computing and FPGAs (ReConFig'05)*, IEEE, pp. 17–17. doi: 10.1109/RECONFIG.2005.9.
- [25] J. A. Perez-Celis, J. Martinez-Carranza, A. Morales-Reyes, C. Feregrino-Urbe, and R. Cumplido, "An FPGA Architecture to Accelerate the Burrows Wheeler Transform by Using a Linear Sorter," in *2016 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW)*, IEEE, May 2016, pp. 156–161. doi: 10.1109/IPDPSW.2016.174.
- [26] C. Zhang, X. Tang, and Y. Peng, "Enhancing Regular Expression Processing through Field-Programmable Gate Array-Based Multi-Character Non-Deterministic Finite Automata," *Electronics (Basel)*, vol. 13, no. 9, p. 1635, Apr. 2024, doi: 10.3390/electronics13091635.
- [27] H. Kim and K.-I. Choi, "A Pipelined Non-Deterministic Finite Automaton-Based String Matching Scheme Using Merged State Transitions in an FPGA," *PLoS One*, vol. 11, no. 10, p. e0163535, Oct. 2016, doi: 10.1371/journal.pone.0163535.
- [28] Z. Wen, L. Kong, A. Le Glaunec, K. Mamouras, and K. Yang, "BVAP: Energy and Memory Efficient Automata Processing for Regular Expressions with Bounded Repetitions," in *Proceedings of the 29th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 2*, New York, NY, USA: ACM, Apr. 2024, pp. 151–166. doi: 10.1145/3620665.3640412.
- [29] L. Gong, C. Wang, H. Xia, X. Chen, X. Li, and X. Zhou, "Enabling Fast and Memory-Efficient Acceleration for Pattern Matching Workloads: The Lightweight Automata Processing Engine," *IEEE*

Transactions on Computers, vol. 72, no. 4, pp. 1011–1025, Apr. 2023, doi: 10.1109/TC.2022.3187338.

- [30] S. Wang, M. Zhang, G. Li, C. Liu, Y. Liu, X. Jia, and M. Xu, "Making Multi-String Pattern Matching Scalable and Cost-Efficient with Programmable Switching ASICs," in *IEEE INFOCOM 2021 - IEEE Conference on Computer Communications*, IEEE, May 2021, pp. 1–10. doi: 10.1109/INFOCOM42981.2021.9488796.
- [31] Y. Huang, Z. Chen, D. Li, and K. Yang, "CAMA: Energy and Memory Efficient Automata Processing in Content-Addressable Memories," in *2022 IEEE International Symposium on High-Performance Computer Architecture (HPCA)*, IEEE, Apr. 2022, pp. 25–37. doi: 10.1109/HPCA53966.2022.00011.
- [32] S. Mandal and S. Ghosh, "Hardware Accelerator for Short-Read DNA Sequence Alignment Using Burrows-Wheeler Transformation," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 45, no. 1, pp. 547–551, Jan. 2026, doi: 10.1109/TCAD.2025.3579326.
- [33] C.-H. Yang, Y.-C. Wu, Y.-L. Chen, C.-H. Lee, J.-H. Hung, and C.-H. Yang, "An FM-Index Based High-Throughput Memory-Efficient FPGA Accelerator for Paired-End Short-Read Mapping," *IEEE Trans. Biomed. Circuits Syst.*, vol. 17, no. 6, pp. 1331–1341, Dec. 2023, doi: 10.1109/TBCAS.2023.3293721.
- [34] D. Terzić, N. Pita, N. Bečirspahić, and D. Muhić, "Parallel DNA Sequencing using Exact String-Matching Algorithms for Multiple Patterns." In *2025 XXX International Conference on Information, Communication and Automation Technologies (ICAT)*, pp. 1-6. IEEE, 2025. doi: 10.1109/ICAT66432.2025.11189285.

Contribution of Individual Authors to the Creation of a Scientific Article (Ghostwriting Policy)

- 'Iffah Athirah Zaidi wrote the original draft of the paper, methodology, formal analysis, review and editing.
- Abdul Karimi Halim planned the formal analysis, review and editing.
- Syed Abdul Mutalib Al Junid supervised the investigation, supervision of students, review and editing
- Mazurina Mohd Ali advised on the project risk, visualisation of figures and tables, review and editing.
- Ahmad Hazmie Adli Mohd Khalid carried out the simulation, implementation, formal analysis, review and editing.
- Abdul Hadi Abdul Razak organised the Conceptualisation and structure of the paper, investigation, supervision, funding acquisition, review and editing.

Sources of Funding for Research Presented in a Scientific Article or Scientific Article Itself

This research was supported by the Malaysian Ministry of Higher Education under the Fundamental Research Grant Scheme (FRGS), Grant Code: FRGS/1/2023/TK07/UITM/02/1.

Conflict of Interest

The authors have no conflicts of interest to declare that are relevant to the content of this article.

Creative Commons Attribution License 4.0 (Attribution 4.0 International, CC BY 4.0)

This article is published under the terms of the Creative Commons Attribution License 4.0

https://creativecommons.org/licenses/by/4.0/deed.en_US

APPENDIX

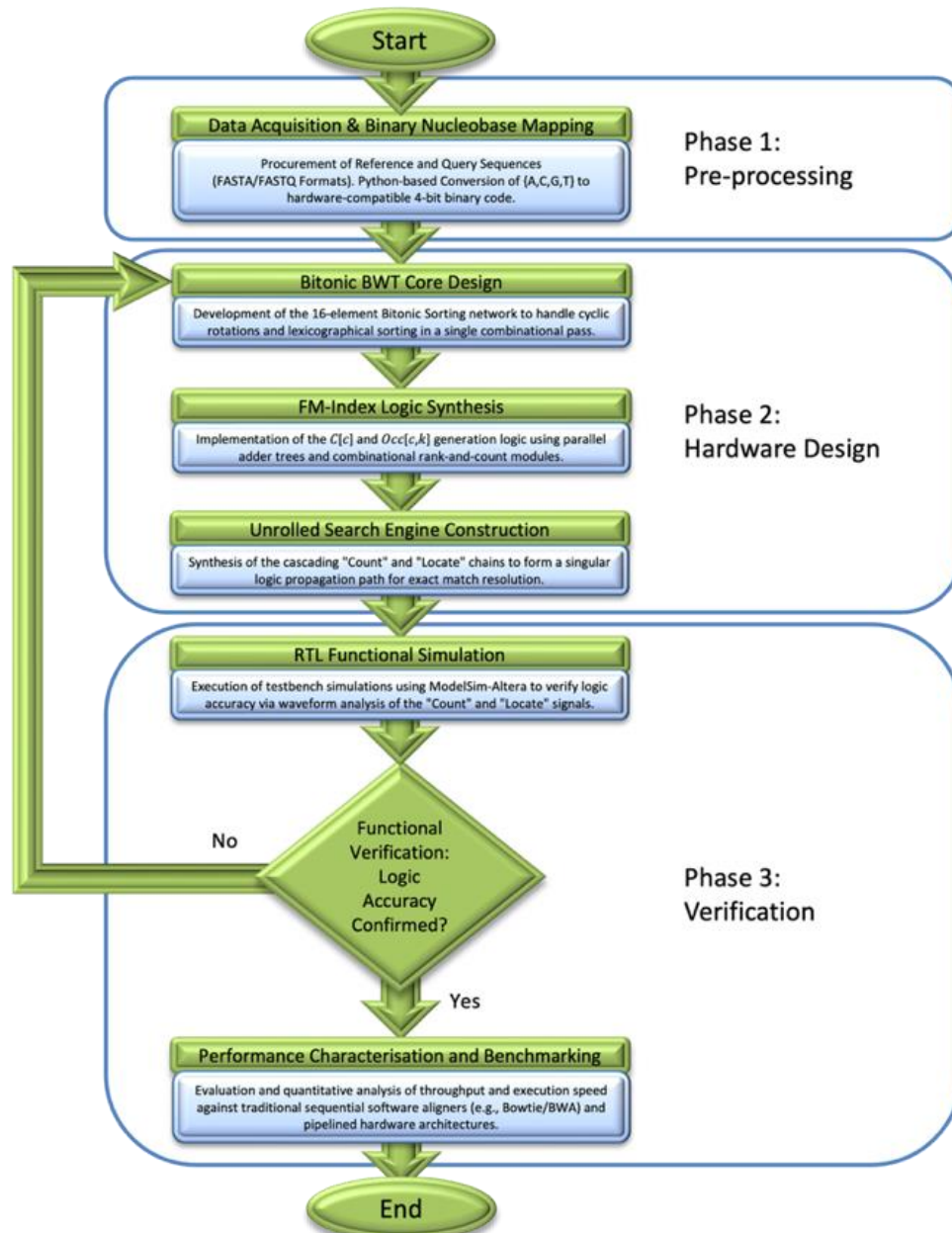


Fig. 1: Systematic architectural framework
Source: created by the authors

lambda_virus.fa - Notepad

File Edit Format View Help

>gi|9626243|ref|NC_001416.1| Enterobacteria phage lambda, complete genom
GGGCGGCGACCTCGCGGGTTTCGCTATTATGAAATTTTCGGTTTAAAGCGTTTCGGTTCTCTTCG
TCATAACTTAATGTTTTATTTAAATACCCCTGAAAAAGAAACGACAGGTGCTGAAAGCGAGGC
TTTTGGCCTCTGCTGTTTCTCTCTGTTTGTCCGTGGAATGAACATGAAGTCAACAAAAAGCA
GCTGGCTGACATTTTCGGTGCAGTATCCGTACATTGACAGATGGCAGGAACAGGGAATGCCCTGCTC
CGAGGCGTGGCAAGGTAATGAGGTGCTTTATGACTTCGCCCGCTCATAAATGGTATGCCGAAAGG
ATGCTGAATTGAAACGAAAGCTGCGCGGGAGGTGAAGAATCGCGCAGGCCAGCGAGGCAGATCT
CCAGCAGGAATATTGAGTACGAACGCCATCGACTTACGCTGCGCAGGCCAGCGACAGGAATGAAG
AATGCCAGAGACTCCCTGAAGTGGTGAACCGCATCTGTACTTTCTGCTGCTCGCGGATCGCAGGTG
AAATTGCCAGTATTCGACGGGTCCTCCCTGTGCGGTGCGAGCGGCTTTCCGGAATGGAACCGACA
TGTTGATTTCTGAAACGGGATATCATCAAGCATGAACAAAGCAGCGCGCTGGATGAATGATACCG
GGTTGCTGAGTGAATATCGAACAGTCAGTTAAAGGCTGCGGCTTTTGTCCGCGCGGCTTCG
TCACTGTTGAGGCGGAGCCAGACGCGCTGTAATGGGCGGATGCTAATTACTATCTCCGAAAGAA
CCGCATACCAGGAAGGCGCTGGGAACACTGCCCTTTCAGCGGGCATCATGAATGCGATGGGACGCA
CTACATCCGTGAGGTGAATGTTGTTGAGTCTGCCGTGCTGGTATTCCAAATGCTGCTGGGTGTTAT
GCTTACTTTATAGAGCATAAAGCGCGCAACCCCTTATCTGGTTGCGCGGATGGTATGCCGAGAACT
TTATGAAACCCAGCTGAGCCGACTATTGCTGATATTCCTGCTGCTGCGCTGCGCGGCTGGTATGG
CAAAAGCAGCGGATAACACGCTCACATGAAGCGTTTCACTAATGGGCGTGGCTTCTGGTCTGGGC
GGTAAAGCGCAAAAAATACCGTGAAGTGGTGGATGGCGGTTATGATGAATGCTGCTTTTG
ATGATGATATTGAACAGGAAGGCTCTCCGCGTTCCTGGGTGACAGCGCTATTGAAGGCTCGGTCTGGCC
AAAGTCCATCGTGGTCCACGCCAAAGTGAGAGGCACTGTGAGATTGAGCGTGCAGCAGTGAATCC
CCGATTTTATGCTGTTTATGTTGCTGCGCGGATTCGGGAGGAGCATGATCTTAAATTTGGCAGCA
AAGAGACGCGTTTGGCTCAATGGACGCGGATGACCCCTCAGCGTGTATTATCTCTCGAGCATAA
TGCTGCTGCTATCGGCTAGCAGGAGTGGATTTATGATGCTGCTTATCTGCGGAAAGAGCGGAGT

lambda_virus_binary.txt - Notepad

File Edit Format View Help

001100110011001000110011001100010010001001000010001100100011
001100110100010001000100010001100100100000101000100010000010100
0011000100010001000101000100010001000010001000110011010001000100
000100010011001100100011010001000100001000110100010000100100
01000010010001000010001101000010100001000100100100010000001
000101000011010001000100010001000001010001000100000100010001
01000001001000100010010000010010000110001000100010001001100010001
00010011001100010001000100011000100100000100110011010000110010
0100001100010001000100110010001100010011001100100100010001000100
01000011001100100010010000010010000110100001000110100010001000010
00100100010001000010010000010010000110100010001000100000110100
0010001000110100001100110001000101000011000100010010000100010100
001100110001000100110100001000010010010000100010001000100010011
0010000100110010010000110011001001000011000100100001010001000100
0100001000110011010000110010001100010011010000010100001000100011
0100000100100010000101000100001000010011000100010010010000110011
0010000100110011000100010010000100110011001100010001010000110010
0010001000110100010000100100001100100011000100110011001000110011
0100001100110010000100010011001100110100000100010100001100010011
001101000001100100100010001000100000101000011000100100010010000011

Fig. 2: Sample conversion of FASTA file into binary format for the *Enterobacteria phage lambda*
Source: created by the authors

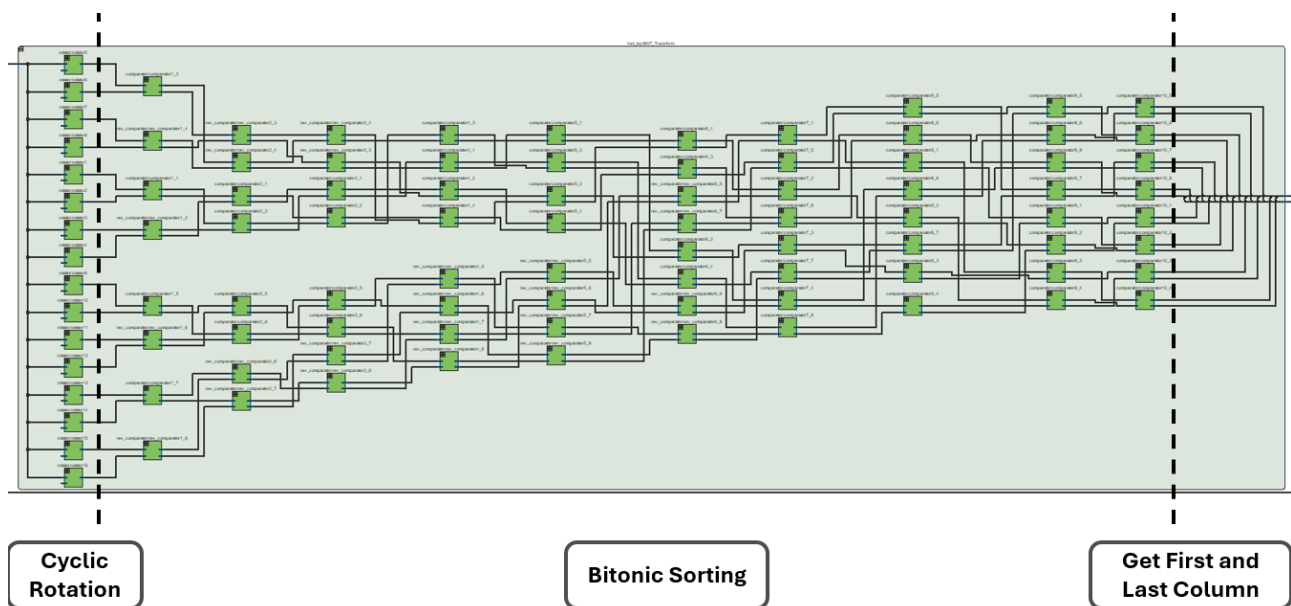


Fig. 15: The overall RTL implementation of the Parallelised Bitonic-BWT Core Design module
Source: created by the authors

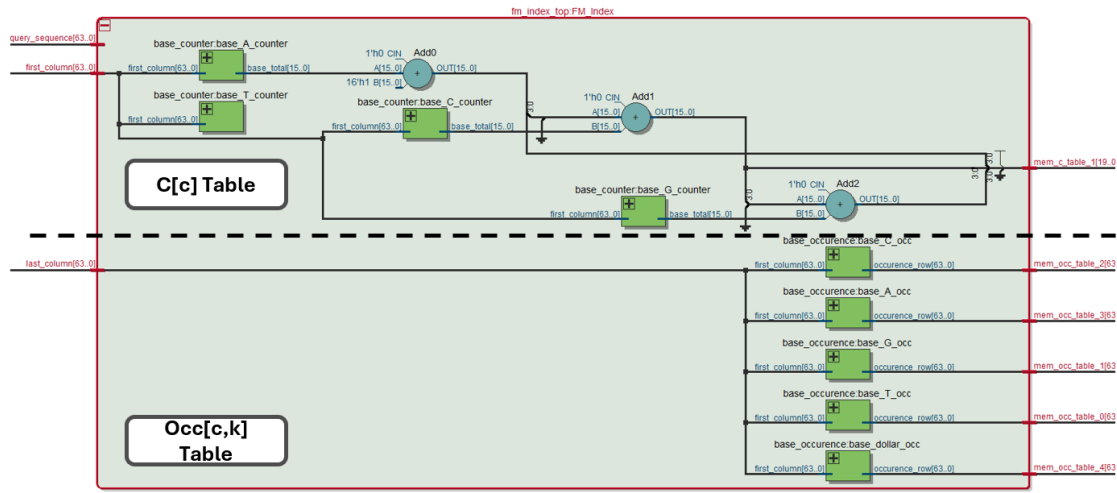


Fig. 18: The RTL implementation of the Combinational FM-Index Logic Synthesis module
Source: created by the authors

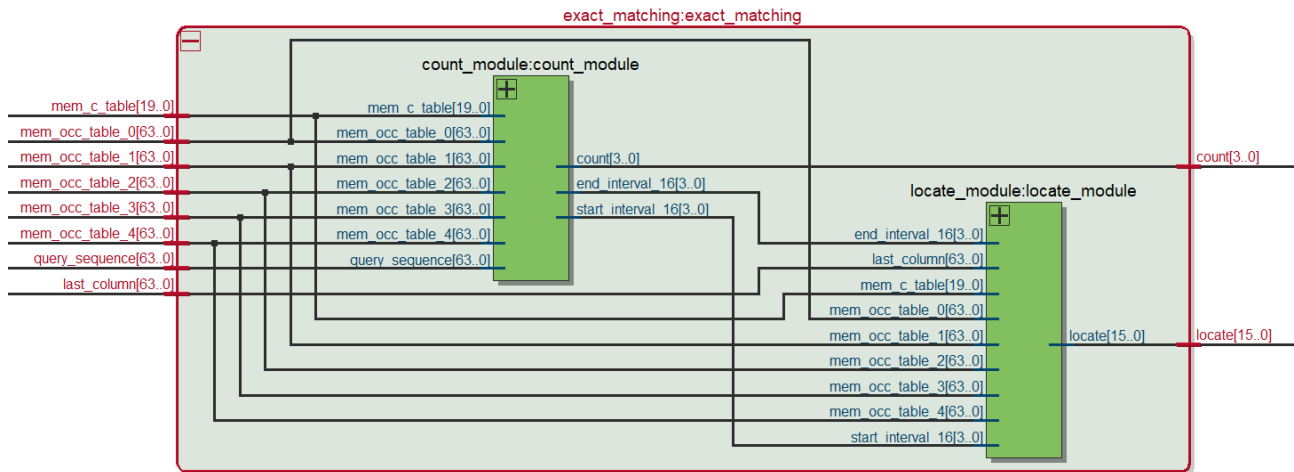


Fig. 19: The RTL implementation of the massive spatial unrolling of the recursive backwards-search algorithm, consisting of the Count Module and Locate Module
Source: created by the authors

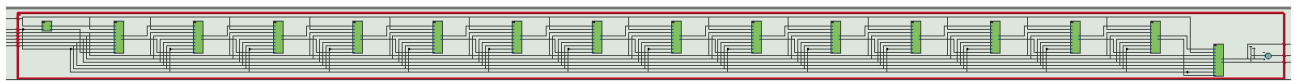


Fig. 20: The RTL implementation of the Count Module
Source: created by the authors

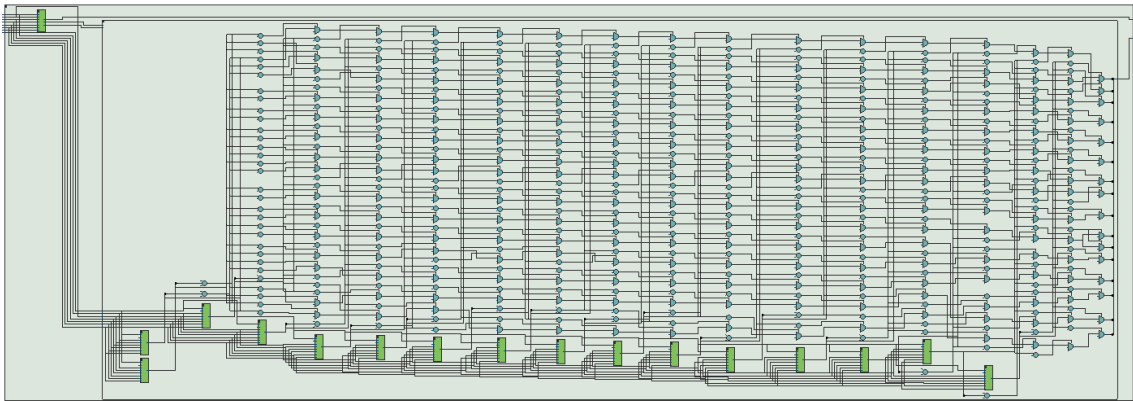


Fig. 21: The RTL implementation for the Locate Module
Source: created by the authors

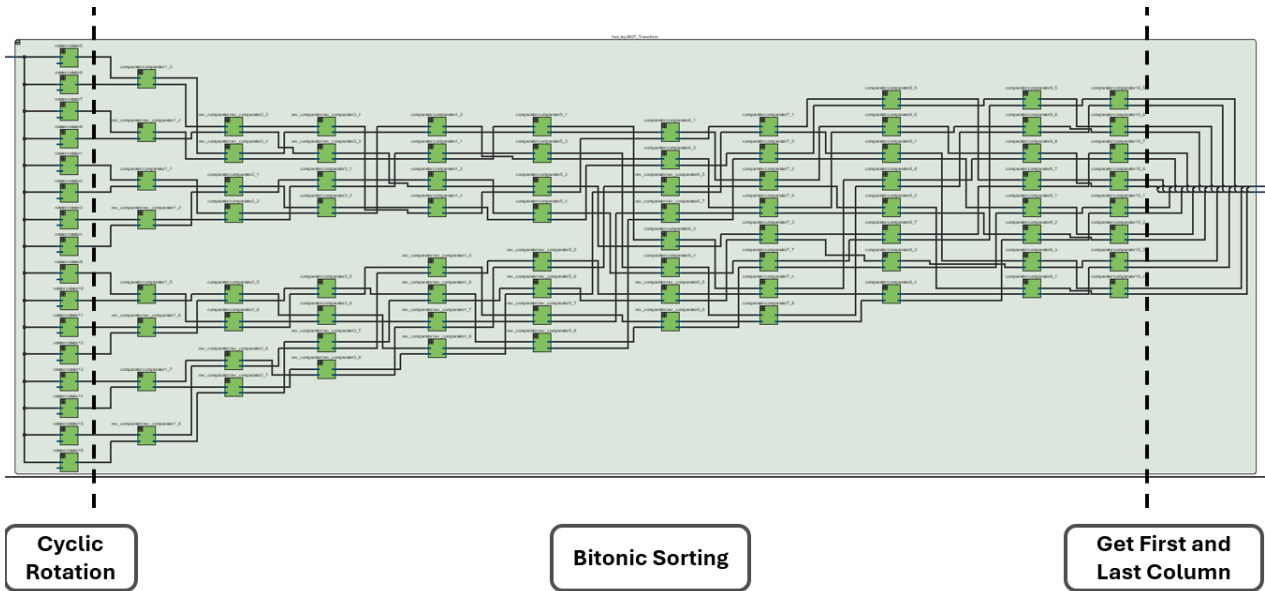


Fig. 22: The final RTL implementation of the Fully Combinational Hardware Accelerator for BWT-FM Index DNA Sequence Alignment via Parallelised Bitonic Sorting
Source: created by the authors

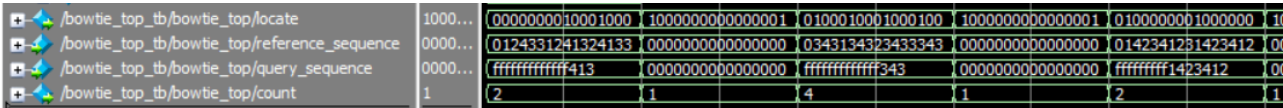


Fig. 23: The top-level output of the designed aligner after performing the Exact Matching process
Source: created by the authors

Table 12. Power consumption and energy efficiency profile on Cyclone IV

Evaluation Parameter	Value
FPGA Target Device	Cyclone IV E (EP4CE115F29C7)
Core Supply Voltage (V_{CCINT})	1.20 °C
Ambient Operating Temperature (T_A)	25.0 mW
Core Static Power Dissipation (P_{static})	98.42 mW
Dynamic Logic Power Dissipation ($P_{dynamic}$)	142.15 mW
I/O Thermal Power Dissipation (P_{IO})	31.83 mW
Total Thermal Power Dissipation (P_{total})	272.40 mW
Combinational Data Path Latency (T_{pd})	9.06 ns
Alignment Throughput (R)	110.37 MReads/s (10^6 reads/s)
Energy Consumed Per Read Alignment ($E_{alignment}$)	2.47 nJ/alignment (10^{-9} J)
Throughput Efficiency Per Watt (η_{power})	405.18 MReads/s/Watt (10^6 reads/s/W)

Source: created by the authors

Table 13. Quantitative comparative benchmarking against state-of-the-art accelerators

Architectural Feature / Metric	Proposed Accelerator	Mandal & Ghosh (2026)	Yang et al. (2023)	Terzić et al. (2025)
Execution Paradigm	Fully Combinational (Non- Sequential)	Parallel Pipelined Search (Clock- Driven)	Memory-Centric Pipelined Mapping	Massively Parallel GPU Kernel (CUDA)
Target Platform	Intel Cyclone IV E(EP4CE115)	AMD-Xilinx Zynq UltraScale+	AMD-Xilinx Alveo U250	NVIDIA Tesla T4 GPU
BWT / Indexing Logic	16-Element Bitonic Sort + Unrolled FM	Comparison-Free Sorter + PSSA	Down-Sampled FM-Index + LUT	Suffix Array + Bitonic Sort
Reference / Read Capacity	16 bp/16 bp (Local Reference)	64--256 bp/32--56 bp	3 Gbp (Human Genome) / 148 bp	0.2--5.2 Mbp / 25--800 bp
Logic Elements (LUTs / LEs)	136,533 LEs	77.83% UltraScale+ LUTs	142,407 CLB LUTs (8.24%)	N/A (GPU CUDA Cores)
Registers / Flip- Flops (FFs)	0 FFs (Zero Registers)	28.96% UltraScale+ FFs	161,451 CLB Registers (4.67%)	N/A (GPU Register File)
On-Chip / Off- Chip Storage	0 BRAM / 0 External DRAM	0 BRAM / 0 External Memory	62.5 BRAMs / 4.9 GB DDR4 DRAM	15 GB GPU VRAM
Sequence Terminator (\$\$)	Present (4-bit encoding compatibility)	Omitted (2-bit binary mapping)	Present (Down-sampled FM)	Present (\$ delimiter required)
Operating Frequency (f_{max})	110.30 MHz (30 ns constraint)	500.0 MHz	200.0 MHz	~ 1.5 GHz (GPU Boost Clock)
Per-Read Latency (T_{pd} / Time)	9.06 ns (Sub-Cycle Propagation)	1.313--1.497 ns (1 Clock Cycle)	Iterative DRAM-bound Lookups	135--1200 ms (Batch runtime)
Alignment Throughput	110.37 MReads/s	4.125 Gbp/s	75.66 Mbp/s(75,659 kbp/s)	Batch Parallel Multi-Pattern
Total System Power (P_{total})	272.40 mW (0.272 W)	Active UltraScale+ Dynamic Power	High-Power Board (PCIe Active)	70.0 W (Tesla T4 TDP)
Energy Efficiency Metric	2.47 nJ / alignment	Active Clock Dynamic Switching	Off-Chip DDR4 Access Penalty	High GPU Power Overhead
Throughput Efficiency	405.18 MReads/s / Watt	High-Frequency Switching	Memory Bandwidth Bound	1.5--3.5 MPatterns/s/W

Source: created by the authors