Journal of
Engineering Sciences
Faculty of Engineering
Assiut University
JES
مجلة
العلوم الهندسية
كلية الهندسة
جامعة أسيوط
journal homepage: http://jesaun.journals.ekb.eg

Comparative Analysis of On-Chip FPGA Memory Architectures for Viterbi Decoder Implementation in DVB Systems

Received 16 December 2024; Revised 22 May 2025; Accepted 22 May 2025

Asmaa Mosbeh [1]
Aly A. I. Younes[2]
Hassan Moustafa[3]
Khalil Yousef [4]

Keywords
BRAM, URAM, LUTRAM. DVB, FPGA, DVB System, Viterbi Decoder

Abstract: High-definition Digital Video Broadcasting (DVB) systems demand high data rates, resulting in increased hardware complexity and power consumption, with the Viterbi decoder (VD) being a key contributor. The substantial memory resources required for these high data rates drive this research, which investigates the impact of diverse static random-access memory (SRAM) architectures on Zynq FPGA and their embedded memory resources, aiming to design power-efficient and less complex Viterbi decoders. Besides, the effect of different memories architectures on the transceiver (Tx-Rx) system has been studied. Viterbi decoder is implemented with different memory architectures on xczu7ev-2ffvc1156: flip flops, distributed RAM, block ram (BRAM), and UltraRAM^{TM} (URAM). BRAM IP from AMD has significantly improved the dynamic power of Viterbi decoder by 97% compared to other available memories. Effectiveness of the employed BRAM is ensured by saving about 50% of the total power of the baseband transceiver system. That Tx-Rx operates at a frequency of 125MHz with a throughput of 62.3 Mbps, a code rate: ½, and 16APSK modulation scheme. Viterbi decoder has achieved a reduction in power compared to sleepy keeper and space time trellis code (STTC) with about 44% and 61% respectively. Functionality of the proposed VD architecture for signal to noise ratio (SNR) of 0 dB at additive white Gaussian noise (AWGN) channel and vector length of 3,264 bits is verified. Hardware validation on ZCU104 based on DVB standard is also done and reported.

1. Introduction

Field-programmable gate Arrays (FPGAs) from the AMD/Xilinx are renowned for their versatile architecture, offering a diverse array of on-chip memory and logic resources vital

[1] Electrical Engineering Dept., Faculty of Engineering, Assiut University, Assiut, Egypt. asmaa.mosbeh@eng.sohag.edu.eg
[2] Electrical Engineering Dept., Faculty of Engineering, Assiut University, Assiut, Egypt. ali.ibrahim1@eng.au.edu.eg
[3] Electrical Engineering Dept., Faculty of Engineering, Cairo University, Cairo, Egypt. hmostafa@uwaterloo.ca
[4] Electrical Engineering Dept., Faculty of Engineering, Assiut University, Assiut, Egypt. kyousef@aun.edu.eg

https://doi.org/10.21608/jesaun.2025.344883.1393

for complex digital system design. AMD/Xilinx FPGAs provide dedicated Block RAM (BRAM) and UltraRAM™ (URAM) memory blocks, in addition to distributed RAM. These pre-engineered memory modules offer distinct advantages in terms of capacity and performance, and their availability as intellectual property (IPs) streamlines the development process for memory-intensive applications. The thoughtful selection and efficient implementation of these components directly determine the performance, resource utilization, and power consumption of FPGA designs. Utilizing alternative BRAM configurations, a power saving of up to 82% on BRAMs and 50% on total dynamic power is observed under a constant frame size [1].

Video applications within communication systems, such as Digital Video Broadcasting (DVB), necessitate the processing of substantial data volumes, leading to significant memory resource demands and power consumption. This necessitates focused research and development efforts to optimize these applications. Notably, discussions of DVB complexity and power consumption are incomplete without the Viterbi decoder (VD), a core transceiver block that critically impacts both resource utilization and power dissipation [2]. VD is one of the widely used algorithms with convolutional encoder in different digital communication schemes such as: WiMAX, UWB, 802.11a/g, DVB, CDMA, and LTE. Viterbi decoder with convolutional encoder is used in channel coding as forward error correction (FEC) scheme to detect corrupted messages from the channel and correct them, it's also used in data storage systems [3-4]

However, Viterbi decoder is employed in various applications: communication systems, natural language processing, bioinformatics, and speech recognition, its consumed power can exceed 35% of the entire power of the system. In addition, it represents about 76% of system complexity in one second of instruction execution in Tx-Rx such as IEEE 802.11 a/g WLAN standards under certain conditions: 128 states and 54 Mbps. Thus, low power VD design and implementation acquires the interest of researchers in disciplinary research direction. Minimization of VD dissipated power would have a direct impact on Tx-Rx implementation of various applications [5-6]. This paper presents a comprehensive comparative study of configurable logic blocks (CLB) flip-flops, distributed RAM, BRAM, and UltraRAM™ within the UltraScale+™ architecture-based FPGAs providing a hierarchy of on-chip memory resources with a greater number of memory resources compared to UltraScale™ FPGAs. By dissecting their architectural nuances, performance attributes, and typical application contexts, we aim to empower designers with the insight necessary for optimal resource allocation. Specifically, we will scrutinize key parameters such as storage density, access speed, power dissipation, and routing complexity, elucidating their influence on overall system efficacy.

This paper is organized as follows: Section 2 introduces the background and related work of memory resources on FPGA and Viterbi decoder in DVB. Section 3 discusses the convolutional encoder and decoder in DVB scheme. In section 4, memory architectures mapping for VD is presented. Section 5 reports and discusses the different implementation

results of VD utilizing the available memory architectures, functional verification, and hardware validation. Conclusion is drawn in Section 6.

2. Background and Related Work

UltraScale+™ FPGAs provide a hierarchy of on-chip memory resources, enabling designers to address a wide array of application requirements for data storage, thus reducing the reliance on external, high-latency dynamic random-access memory. This research investigates the memory architecture of the logic fabric of the Zynq UltraScale+™ ZCU104 FPGA device [7-9], to optimize the implementation of a Viterbi decoder for use in digital video broadcasting applications, which require high memory usage.

Table 1: Memory for xczu7ev-2ffvc1156 FPGA device [7]

Memory	Capacity	Ports	Read latency
LUTRAM	6.2Mb	--	0 cycle
BRAM	11.501Mb	2 per BRAM	1-2 cycles
URAM	27Mb	2 per block	3-N cycles

2.1. FPGA Memory Architecture

AMD/Xilinx UltraScale+™ architecture devices utilize CLBs as the primary resource for implementing general-purpose combinational and sequential circuits. LUTRAM or distributed RAM can be implemented using CLBs. CLBs have two types of slices, as shown in Fig. 1 SLICEL for logic contains look-up tables (LUT) and storage elements such as flip-flops and latches. SLICEM for memory slice can utilize its LUTs as static RAM (SRAM) by incorporating a separate write address (WA), write enable (WE), and clock signal. LUT can function as a 64-bit (1x64) single-port RAM, where each input combination addresses a unique 1-bit location. Read and write operations to these distributed RAM elements are random access, meaning any address can be accessed in a single clock cycle, not sequentially. To enable simultaneous read and write operations, a single LUT can often be configured as dual-port RAM (e.g., 2x32-bit), allowing concurrent access to different memory locations. For larger capacities or more complex multi-port requirements, multiple LUTRAMs can be combined (cascaded or arranged in parallel). Each slice provides 8 LUTs. A single SLICEM can provide up to 256 bits of dual-port RAM or up to 512 bits of single-port RAM. Generally, LUTRAM is recommended for memory sized between 64 and 128 bits, unless the target device has limited SLICEM or logic resources. It typically exhibits lower clock-to-output latency and fewer placement constraints. In contrast to dedicated memory resources such as BRAM and URAM, distributed RAM's asynchronous read capability improves latency [7-9].

BRAM, is an embedded memory blocks resource in AMD/Xilinx UltraScale™ architecture-based devices, offers low-latency access. Each BRAM block comprises 36k bits of memory, which can be configured into various configurations, including 32K x 1, 16K x 2, 8K x 4, 4K x 9, 2K x 18, or two independent 18 kbits.

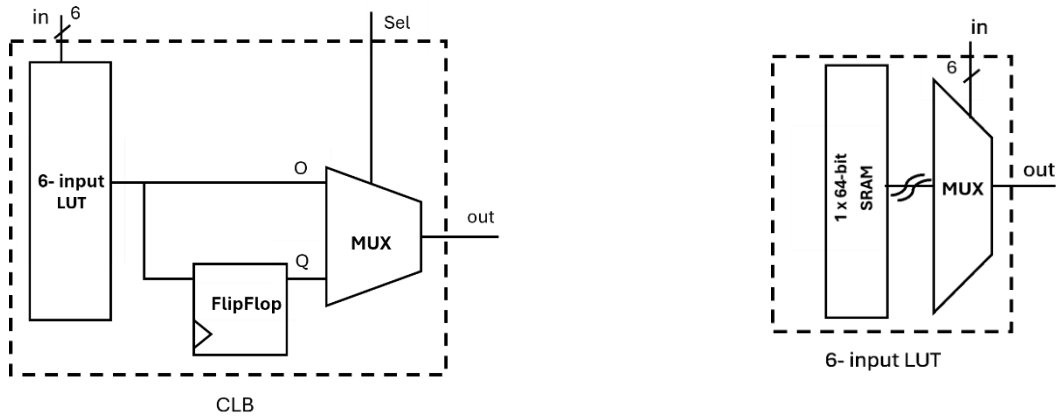

Fig. 1: CLB based on UltraScale™ architecture [7]

Larger BRAMs can be formed by cascading the data outputs of adjacent BRAM. The ZCU104 provides 312 BRAMs, supporting single-port, simple, and true dual-port modes, depending on the width of read and write. True dual-port offers more flexibility. BRAM can have separate clocks for reading and writing operations. Application-specific requirements for power, latency, and timing determine whether BRAM operates in read-first, write-first, or no-change mode. Read-first mode gives priority to the read operation till WE are asserted. Write-first or transparent mode is simultaneously written, and the output is the current written data till WE are deactivated. Read-first and write-first modes are not suitable for power-efficient designs, as the output is simultaneously switching. The optimal power efficiency mode in BRAM is the no-change mode. No-change mode as name implies, the output is not changed during write operation; it remains on the last read data. BRAMs have a one clock cycle for read operation. Read and write operations on BRAMs require a clock edge. Memory contents are initialized or cleared through the configuration bitstream [8].

The ZCU104 also has 96 URAM on-chip memory. Organized as 72-bit x 4K entries, UltraRAM™ blocks provide a storage capacity of 288 Kb offers eight times the storage capacity of BRAM. These single-clock, synchronous memory elements are arranged in a columnar fashion within the device architecture. Each UltraRAM™ block supports dual-port operation, with each portable to execute a read or write operation per clock cycle. UltraRAM™ is characterized by fixed read behavior, without the user-defined flexibility of read-first, write-first, and no-change modes. Compared to BRAM, UltraRAM™ offers rapid on-chip access, albeit with increased latency. URAM, like BRAM, is implemented using SRAM technology and fabricated using a 16 nm process [8][10].

For memory depths exceeding 64 bits but not exceeding 128 bits, the optimal memory resource selection depends on several factors: first, the availability of BRAM or URAM; if neither is available, distributed RAM should be used. Second, if asynchronous read capability is required, distributed RAM is mandatory due to its inherent support. Third, data widths exceeding 16 bits are generally better suited to BRAM or URAM. Fourth, design flexibility, supporting different modes of operations, and varying width and depth configuration are better served by BRAM than URAM. For applications prioritizing fast access, URAM is preferred, whereas LUTRAM is optimal when latency is a primary concern [7-9].

2.2. Viterbi Decoder in DVB

The DVB is a consortium comprising various industry stakeholders, including broadcasters, manufacturers, and network operators, focused on developing global standards for digital television and data services. DVB standards address all facets of digital television, from transmission to interactivity, and are designed to meet the evolving needs of the broadcast industry and consumers. Over the years, the ongoing evolution of DVB standards underscores their continued importance in accommodating the increasing demand for greater bandwidth and interactive service provision, driven by the need to support higher volumes of data. To ensure robust transmission, the DVB standard incorporates FEC through a convolutional code characterized by a code rate of ½ and a constraint length of 7. The resulting encoded symbols, each of length 1632, are then input to the Variable Decoder (VD) for further processing [2].

Viterbi decoder receives the corrupted message from the channel by sequentially passing it through branch metric unit (BMU), add compare-select unit (ACSU), and survivor memory unit (SMU) as illustrated in Fig. 2 BMU compares the income bits with the expected one generated in the encoder and counts the number of flipped bits introduced by the channel. ACSU adds the previous path metric (PM) stored in PM memory (PMM) with current branch metric (BM) and selects the path with the minimum cumulative metric, representing the most likely output sequence from the encoder. The VD updates the PM for the following calculation and stores the selected survivor path in SMU. Finally, SMU decides the decoded bits based on maximum likelihood (ML) principle. The ACSU and SMU draw most of the decoder's power usage out of Viterbi's modules [11-12].

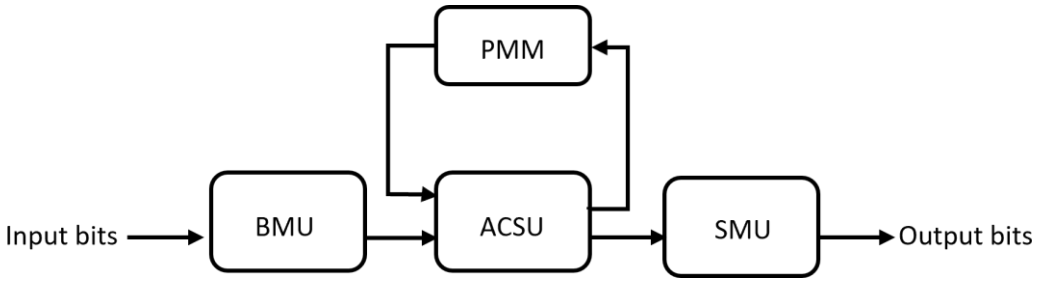

Fig. 2: Block diagram of Viterbi decoder.

2.3. Related work

Many challenges in very-large-scale integration (VLSI) design have arisen because of the growing need for increasingly sophisticated and potent digital systems. Trade-off between speed, area, and power is a challenge since gains in one aspect sometimes come at the expense of the others. Power consumption has significantly grown due to the unrelenting search for handheld devices. Concurrently, the need for more compact designs has arisen from the desire for smaller form factors, which may result in higher power densities and lower error margins [13].

As SMU and ACSU are the main contributors to power consumed in VD, various work has been introduced to enhance power consumption through these blocks. ACSU consists of

adders and comparators. Different adders' utilization in ACS would improve dissipated power in the decoder [14]. However, this approach is not effective in large input stream as this requires large memory size at which adders are not the main contributor. In this work, length of input stream comes to VD input is 1632 symbol with a code rate of ½. Thus, memory size here surges a large dynamic power. Enhancement of memories in SMU and PMM is essential for low-power VD implementation.

Various research efforts have explored the utilization of SRAM IP at 180 nm technology node [15]. Prasad et al. [16] have proposed a ZMesh topology in implementing a reconfigurable VD for network-on-chip applications. This design reduces memory requirements in the BMU and eliminates the need for SMU, resulting in decreased power consumption. Register Bank of memories in SMU has been replaced by RAM implemented in CMOS 180 nm process for lower power VD in [5][17]. Parallelization is a common technique utilized to accelerate Viterbi decoding, particularly for encoders with large constraint lengths. Designers replicate the hardware to parallelize the operations and boost the decoder speed [18]. This parallelization should be considered in very low power mode as repeating the memory blocks increases the occupied area and multiplies consumed amount of power. Gate diffusion input (GDI) logic, with its reduced transistor count, offers a potential solution to the area overhead associated with parallelization in VD designs while maintaining high performance [19].

While all the mentioned techniques focus on hardware development, architectural or algorithmic suggestions can also reduce the VD power. Traceback algorithm (TB) consumes power less than register exchange (RE) algorithm [20]. Lower power dissipation can be accomplished in PMM by trimming the search paths in the decoder. This reduces the area and the power. Instead of using ML approach, adaptive decoders can be designed, and the number of search paths is updated depending on the channel state at this time [21-22].

Previous research has demonstrated that memory resources within a VD significantly impact power consumption, with the SMU and PMM modules being key areas of interest. While the use of SRAM has been suggested to mitigate power consumption, a comprehensive investigation into the influence of different SRAM architectures on VD, and specifically on the SMU and PMM, remains unexplored. This work addresses this gap by examining the effect of employing various memory architectures available on the ZCU104, including CLB flip-flops, LUTRAM, BRAM, and URAM, on the power consumption of the VD. The study aims to determine the optimal memory architectures for the SMU and PMM, respectively, providing insights for minimizing power consumption in VD implementations.

Ref. [23] presented a BRAM-based implementation of ternary content-addressable memory (TCAM) for software-defined networks on a AMD/Xilinx Virtex-6 FPGA, reporting a 40% improvement in power efficiency and a one-cycle latency over earlier designs. A BRAM-based Stripe Memory Engine (SME) is presented in [24] for object detection in vision and scene analysis, demonstrating low power consumption and accurate results. Prior investigations into FPGA-embedded RAM mapping have focused on preserving functional

integrity while minimizing dynamic power. These efforts resulted in a 21-26% reduction in RAM power and a 7% reduction in core power [25-26].

In this paper, a low-power VD is implemented on 16 nm xczu7ev-2ffvc1156 FPGA using the memory resources: CLB flip-flops, LUTRAM, BRAM, and URAM [9]. Dynamic power in all implementations is considered. The impact of various implementations on the overall Tx-Rx system is also considered for verifying the low-power performance of VD. A power analysis is conducted using the DVB standard, as shown in Fig. 3, as a benchmark revealed that the VD is responsible for approximately 50% of the system's overall power dissipation.

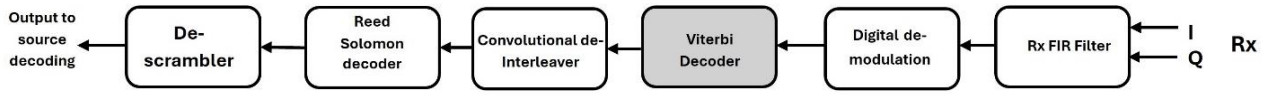

Fig. 3: Block diagram of Rx DVB scheme [2].

3. Memories in VD Architecture

This work investigates the impact of diverse hardware validation memory architectures (CLB flip-flops, BRAM, LUTRAM, URAM) on power consumption within the Viterbi decoder. The work specifically incorporates both coarse and fine selection of memory architectures for the SMU and PMM. Coarse selection mandates the use of an identical memory architecture for all instances within the SMU and PM. Fine selection permits the employment of heterogeneous memory architecture combinations specifically configured for individual SMU and PMM units. By examining the VD across these varied memory architectures, the work aims to elucidate the associated power consumption and hardware resource utilization.

VD is synthesized and implemented, written in Verilog, on Zynq UltraScale+^{TM} MPSoC using VIVADO HLx tool. VD receives a symbol (2-bit) input at a clock frequency of 125 MHz with a stream length of 1638 based on DVB (1632 DVB bits plus 6 initialization bits). For error computation, each PM utilizes 9-12 bits. These PM values, calculated for all 64 states, are instantaneously stored in the PMM, demanding a memory size of 576-768 bits. The SMU, which stores survivor paths for 64 states across 1638 clock cycles, requires 104,832 memory units.

VD is implemented using parallel computing for all states that SMU and PMM should be accessed for write and read operations instantaneously for all 64 states. This requires dual-port memories and no latency in the reading operation.

An evaluation of the Viterbi decoder's power efficiency and computational complexity is conducted. This assessment relies on several key metrics: maximum operating frequency (F_{max}), which directly influences dynamic power consumption; FPGA resource utilization, a measure of decoder complexity; decoding latency, defined as the number of clock cycles required to process an input and produce the corresponding decoded output (a parameter common to all designs, except for variations in memory read latency); and decoder

throughput, quantified as the rate at which the decoder processes the input data stream (bits per second). Significant power consumption within the VD is attributed to the large memory capacity and the switching activity of these memories during the scan time.
Functional verification is performed, for each memory selection, based on test vectors derived from the DVB standard (as detailed in Section 5), with maximum frequency assessed using the VIVADO timing analysis tool, and power calculated using the VIVADO Power Analyzer tool.

3.1. Coarse selection of VD memories

A consistent implementation approach is adopted, employing CLB flip-flops, BRAM, LUTRAM, and URAM for the SMU and PMM on the VD. Four distinct VD designs are implemented as follows:

1. CLB Flip-Flop VD

The CLB FF VD design is realized through synthesizable Verilog guidelines, such as using 'always@' procedural blocks, ensuring that the synthesis tool primarily inferred and mapped the SMU and PMM components to CLB FFs.

2. LUTRAM VD

The realization of the SMU and PMM components of the VD using LUTRAM is achieved. Developing RTL guidelines to implement memory with the specific width and depth designed by Xilinx for LUTRAMs on the FPGA. As mentioned in the background section, LUTRAM is a 64-bit single-port. The requirement for dual-port memory, driven by simultaneous read and write operations, is developed in RTL. To preclude different memory resource mapping and mitigate interconnect resource congestion on the FPGA, synthesis tool optimization is carefully constrained through the application of attributes, including dont_touch="true", ram_style="distributed", and cascade_height = N, where N represents the number of cascaded LUTRAM blocks [27].

3. BRAM VD

A BRAM-based implementation of the SMU and PMM in the VD is realized through a detailed RTL specification, necessitating dual-port memory for simultaneous read and write operations. The 'no change' mode, known for its power efficiency, is employed. The timing diagram depicted in Fig. 4, confirms that dout is held constant during write operations, effectively reducing output switching activity when read operations are not performed at certain time intervals. The inherent one-clock-cycle read latency of BRAM poses a challenge, particularly given the 1638 read cycles required by the VD. Implementation of the VD in TB mode introduces substantial latency. To address this, the BRAM read operation is synchronized to the negative clock edge via the insertion of an inverter at the clock port of the read registers. This makes VD write in the positive half cycle and read in the negative half cycle, which overcomes one cycle latency on BRAM. The constraints and attributes applied to the LUTRAM configuration are replicated in this design.

4. URAM VD

The URAM implementation of the SMU and PMM within the VD follows a methodology analogous to that employed for both BRAM and LUTRAM. However, URAM exhibits an inherent read latency of at least three clock cycles, resulting in a significantly increased overall latency. Attempts to modify the HDL representation of the URAM to fix the latency, while maintaining the attributes to prevent mapping to LUTRAMs or flip-flops, proved ineffective. Furthermore, unlike BRAM, URAM's architecture utilizes a single clock port for both reading and writing operations; the clock inversion technique, which is applied to BRAM, is infeasible. URAM VD is implemented with the inherent latency of URAM.

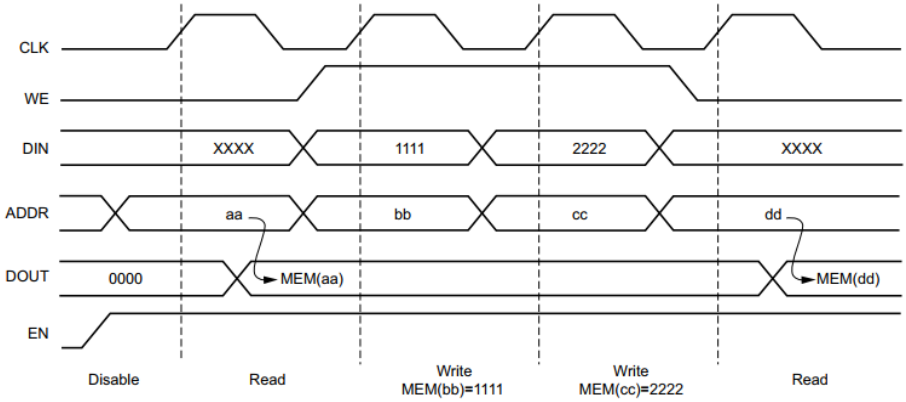

Fig. 4: Timing diagram of no-change mode of BRAM

3.2. Fine selection of VD memories

To analyze the effect of memory architecture on power dissipation and hardware resources on FPGA, a combinatorial selection strategy is employed for the SMU and PMM. This involves varying the memory architecture of one component (e.g., implementing the PMM with CLB FFs) while iterating through all four available memory architectures for the SMU. This procedure is then repeated for each possible architecture of the former. The resulting architectural variations impact directly the overall VD performance.

4. Convolutional encoder and decoder in DVB

4.1. Convolutional encoder

The proposed Tx-Rx employs a (2, 1, 6) convolutional code, as shown in Fig. 5, primarily for bit-level error correction, as its inner coding component. Convolutional encoders enhance transmission reliability by introducing redundant bits. Functionally, a convolutional encoder operates as a shift register Fig. 5, where the input bit represents the transmitted bits. The output codewords, output bit_1 and output bit_2, are generated by performing a modulo-2 sum (XOR operation) on the current input bit and current state of the registers. Input bits are shifted right to the registers in the most significant bit (MSB) of the current state. This process results in the dropping of the least significant bit (LSB),

which can be either '1' or '0'. Consequently, the current memory state is dependent upon two possible preceding states, contingent on the value of the LSB.

The structure of this convolutional code is defined by the parameters (n, k, m), where:

- n: represents the number of output bits.
- k: denotes the number of input bits.
- m: denotes the number of memory elements, corresponding to the quantity of flip-flops within the encoder's shift register. This configuration yields 64 possible encoder states.

4.2. Viterbi decoder

VD employs a ML approach to decode the most probable transmitted frame. VD evaluates the ML based on a trellis diagram, as depicted in Fig. 6 which is formed as groups of butterfly structures, as shown in Fig. 7 Each butterfly structure consists of two ACSU. The number of butterfly structures is determined based on the number of states in the convolutional encoder.

To achieve low-power implementation, VD is designed using TB method. VD scans all 64 states forward and backward to extract the ML path. The computation of this path involves summing the BM and the PM of the converging states, followed by a comparison. The BM quantifies the number of corrupted bits in the received symbol by comparing it to the expected 2-bit encoder output. As established in Section 2.1, each current state is derived from two preceding states which is illustrated in Fig. 7 The BM is added to the PM of the preceding states, which are stored in the PMM. Subsequently, the minimum of the two possible paths converging on each state is determined, as expressed in Equations (1) and (2).

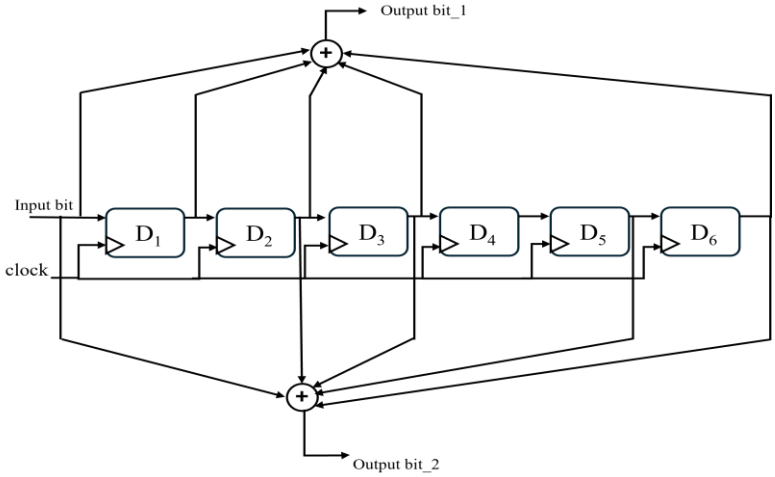

Fig. 5: Convolutional Encoder (2, 1, 6) [2]

VD calculates the PM for all 64 states, iteratively updating the PMM with the minimum PM values. The VD maintains a record of the survivor paths for all 64 states within the SMU during the scanning process. Following a complete scan of the input stream, the VD selects

the path exhibiting the minimum error among the PMs. A subsequent traceback along this selected path yields the most likely sequence of the transmitted frame.

$$PM_{p,t+1} = \min \{PM_{i,t} + BM_{i1} , PM_{j,t} + BM_{j1}\} \quad (1)$$

$$PM_{q,t+1} = \min \{PM_{i,t} + BM_{i0} , PM_{j,t} + BM_{j0}\} \quad (2)$$

Where $PM_{i/j,t+1}$ represents the next PM of state i or j which is stored in PMM for next-time calculations, $PM_{i/j,t}$ denotes the current PM of state i or j, and $BM_{i/j1}$ ($BM_{i/j0}$) signifies the error of expected and received bits when the input bit to convolutional encoder is '1' ('0').

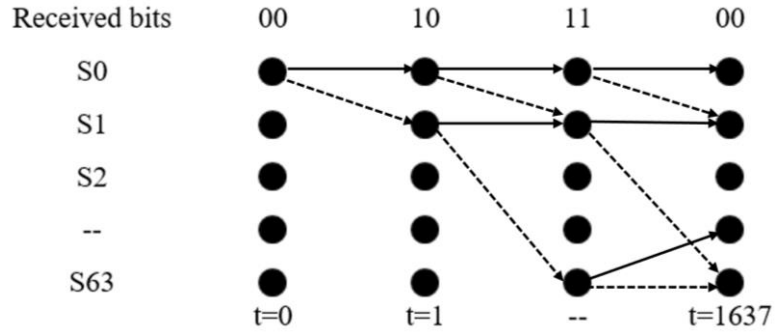

Fig. 6: Trellis diagram of Viterbi decoder.

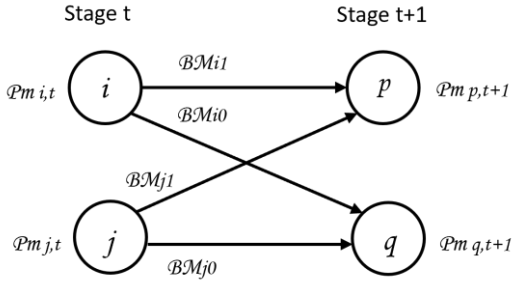

Fig. 7: Butterfly architecture of Viterbi decoder.

5. Results and Discussion

5.1. Power, Performance, and Utilization Analysis

It is demonstrated that employing BRAM for the SMU and CLB flip-flops for the PMM yields the lowest power consumption and minimizes FPGA resource utilization among the evaluated memory architectures on the xczu7ev-2ffvc1156 FPGA. As depicted in Fig. 8 and 9, this configuration achieves a 71% reduction in power consumption and resource utilization, compared to the default mapping produced by the VIVADO synthesis tool, thus underscoring the significant influence of memory configuration on these metrics [1]. Power consumption of each memory architecture is reported on the power analyzer tool as shown in Fig. 9 These results have been obtained with FPGA core voltage (Vccint) of 0.825V, a 125 MHz clock frequency, and a 0 dB SNR.

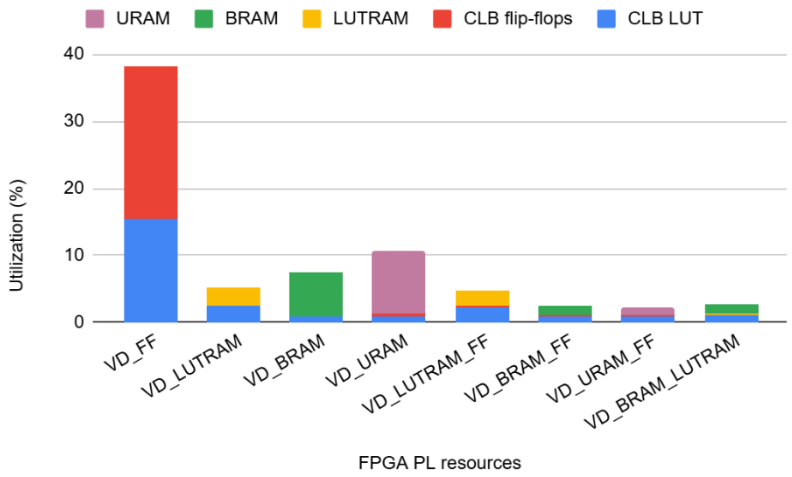

Fig. 8: Utilization of VD using different on-chip FPGA memory resources

Table 2 presents a comparative power analysis of VD designs implemented using the four memory architectures, highlighting key metrics such as latency, throughput, and maximum frequency. For the TB method, the BRAM-based VD achieves a throughput of 62.3 Mbps, significantly outperforming the URAM-based design with 10.41 Mbps. While URAM can operate at higher frequencies (up to 285.7 MHz) and achieve a throughput of 23.8 Mbps, this comes at the cost of increased power consumption. URAM's minimum three-cycle read latency negatively impacts VD performance, particularly in forward and backward scanning, making it a poor architectural choice despite potential power benefits. The read latency presented in Table 2 is the sum of the single memory read latency in the forward and backward scanning stages.

Table 3 presents an analysis of power reduction across various VD designs: LUTRAM-FF, BRAM-FF, URAM-FF, and BRAM-LUTRAM, denoted as SMU-PMM configurations. Notably, the table demonstrates a 40% power reduction in the VD when the PMM is implemented using CLB FFs, combined with BRAM for the SMU. This result is attributed to the non-standard architecture of the PMM, which requires 9 x 64 bits and simultaneous access for all states, leading to high memory resource utilization on the FPGA. Consequently, CLB FFs provide an optimal solution for minimizing resource utilization and routing congestion.

Table 2: Comparative analysis of VD Implementations using CLB FF, LUTRAM, BRAM, and URAM memories at 125 MHz

	Total power (W)	Dynamic power (W)	Total power (%)	Dynamic power (%)	Read Latency (cycles)	Throughput (Mbps)
CLB FF VD	2.229	1.619	--	--	0	62.30
LUTRAM VD	0.74	0.137	-67%	-91%	0	62.30
BRAM VD	0.648	0.045	-71%	-97%	0	62.30
URAM VD	0.66	0.058	-70%	-96%	6	10.41

Table 3: Dynamic power reduction results from optimizing the SMU and PMM with diverse memory architectures

SMU-PMM VD	Dynamic power (W)	Dynamic power (%)	Read Latency (cycles)	Throughput (Mbps)
LUTRAM-FF VD	0.128	-6.5%	0	62.30
BRAM-FF VD	0.027	-40%	0	62.30
URAM-FF VD	0.029	-50%	3	20.83
BRAM-LUTRAM VD	0.037	-17%	0	62.30

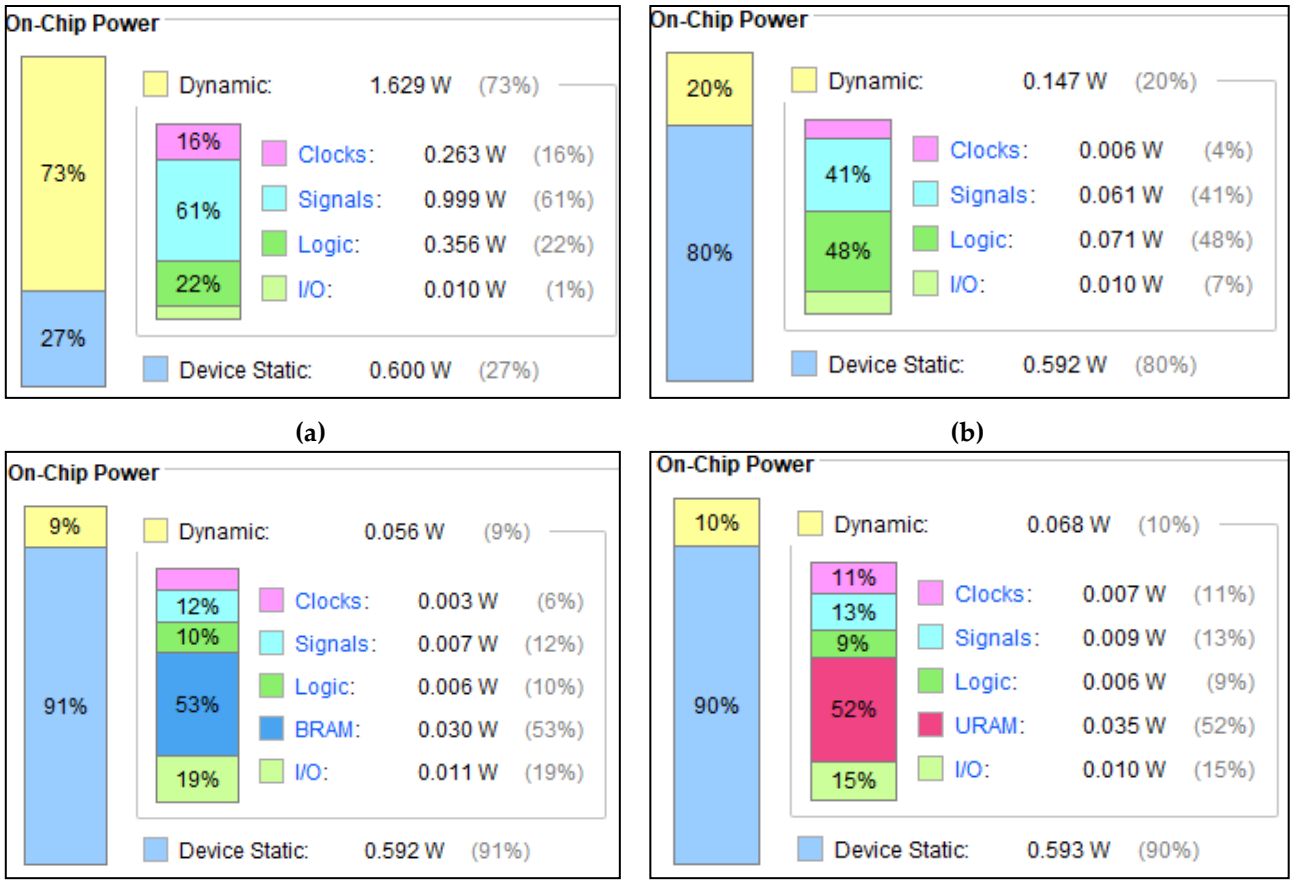

Fig. 9: Power consumption of (a) VD with FF, (b) VD with LUTRAM, (c) VD with BRAM, and (d) VD with URAM.

Fig. 10 and Fig. 11 illustrate the power report after the implementation of the baseband transceiver, which is represented in Fig. 3 Power report includes static and dynamic power of FPGA. The xczu7ev-2ffvc1156 FPGA is fabricated on 16nm which has a significant static power consumption of 0.592 mW. Table 4 provides a detailed breakdown of resource utilization, highlighting the specific FPGA resources consumed by the design. Tx-Rx timing is met for the operating clock frequency of 125 MHz.

As demonstrated in Table 5, the implementation of BRAM in the Viterbi decoder of the baseband transceiver system results in a 50% reduction in power consumption compared to the default mapping produced by the VIVADO synthesis tool on the xczu7ev-2ffvc1156

FPGA. Considering power consumption, resource utilization, and latency, BRAM emerges as the more optimal choice compared to URAM.

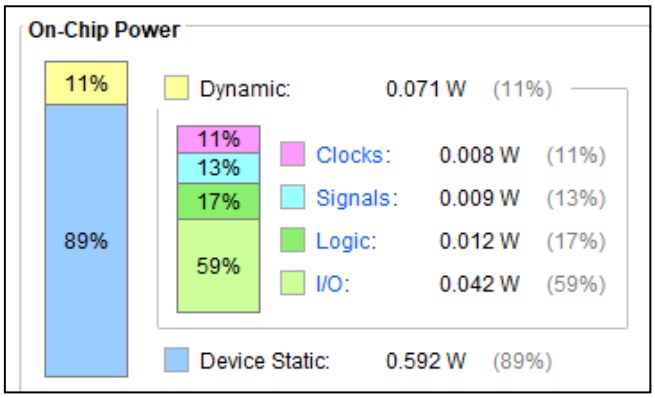

Fig. 10: Power consumption of Tx blocks.

A comparison between four previous designs of VD is shown in Table 6, considering factors such as speed, throughput, power consumption, technology node, and SNR for a constraint length of 7 and a code rate of ½. Our proposed design demonstrates superior performance, achieving low power consumption while maintaining a suitable throughput of 62.30 Mbps for DVB applications.

In comparison to Namratha and Bakhar's design [28], which consumes 89 mW and offers a throughput of 87.8 Mbps, our design exhibits a 29% reduction in throughput but a substantial 70% decrease in power consumption at 0 dB SNR. Akash et al.'s design [29] achieves a power consumption of 37.62 mW but with a significantly lower throughput. Our design offers a 28% reduction in power consumption but a 389x improvement in throughput compared to [29].

This work offers a 44% reduction in power consumption and a higher throughput compared to the work of Devi1 et al. [12]. However, it operates at a lower speed. It is important to note that this work is evaluated under 0 dB AWGN conditions while [12] is tested at 6 dB.

When compared to the design presented in [30], our design offers a significant 61% reduction in power consumption, speed, and throughput, despite being implemented on a 16 nm technology node.

Table 4: Resource utilization of Tx and Rx modules on ZCU104.

Design/Utilization	LUT	LUTRAM	FF	BRAM	URAM	BUFG
Available	230400	101760	460800	312	96	544
Tx	2527	318	1943	--	--	1
Rx with CLB FF	46916	320	113529	--	--	4
Rx with LUTRAM	17254	3024	8003	--	--	3
Rx with BRAM	13997	320	7999	20	--	3
Rx with URAM	13779	320	9285	--	9	3

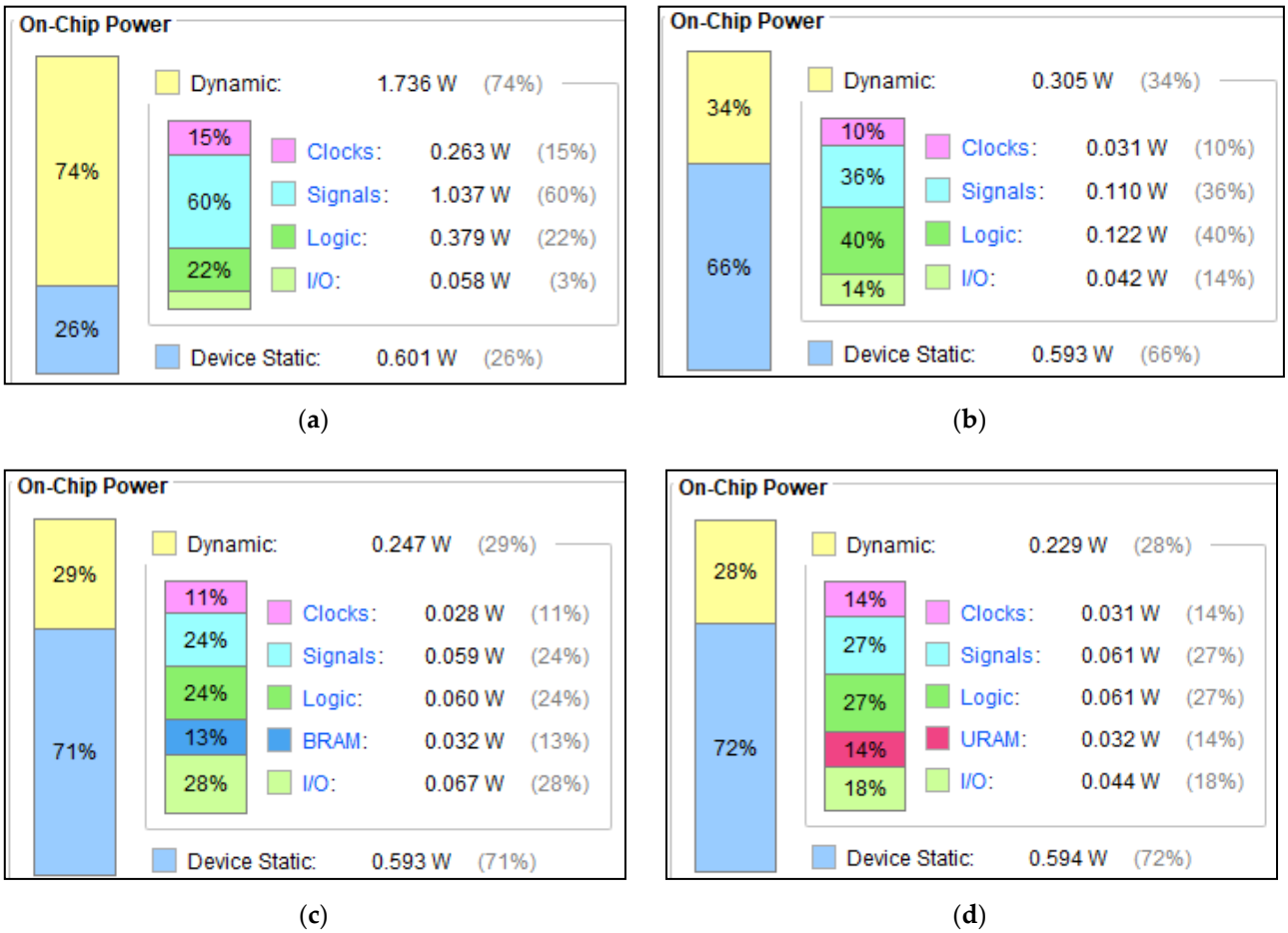

Fig. 11: Power consumption of (a) Rx with FF, (b) Rx with LUTRAM, (c) Rx with BRAM, and (d) Rx with URAM.

Table 5: Transceiver power consumption for various memory architectures implemented on a zcu104 FPGA for VD

	Total power (W)	Dynamic power (W)	Total power (%)	Dynamic power (%)
Tx-Rx with CLB FF	3	1.716	--	--
Tx-Rx with LUTRAM	1.561	0.293	-48%	-83%
Tx-Rx with BRAM	1.503	0.210	-50%	-88%
Tx-Rx with URAM	1.485	0.213	-50%	-87.6%

Table 6: Comparative analysis of proposed VD design and existing methods in terms of key performance metrics.

Design/Parameter	Speed	Throughput	Power	Node	SNR
[28]	401.8 MHz	87.8 Mbps	89 mW	40 nm	--
[29]	221.9 MHz	20.0 KBps	37.62 mW	28 nm	--
[12]	25 GHz	9.0 Mbps	48.56 mW	90 nm	6 dB
[30]	50 MHz	50 Mbps	70 mW	180 nm	--
This work	125.0 MHz	62.30 Mbps	27 mW	16 nm	0 dB

5.2. Functional Verification

Design verification is a crucial phase of complex systems such as DVB transceivers, often consuming up to 80% of the total product development time [31]. To ensure the proposed

design's correctness, a robust verification flow is essential. In this work, a MATLAB-based verification environment is employed to generate "golden test vectors" for a DVB-compliant VD core.

The verification flow initiates with MATLAB code which provides a high-level architectural representation of the transceiver system's blocks. Pseudo-random input test vectors, conforming to DVB standard, employ stimuli for system blocks in MATLAB environment. Output vectors and random input from MATLAB are designated as "golden test vectors". In VIVDO environment design under test (DUT) is stimulated by the same inputs, and its outputs are validated against the golden test vectors.

Functional verification of VD has been successfully proved for an input length of 3,264 bits and code rates of ½, ⅔, and ¾. Simulation is performed for AWGN channel with SNR of 0 dB. Fig. 12 shows a simulation waveform for VD at a code rate of ½, demonstrating the accuracy of the RTL implementation.

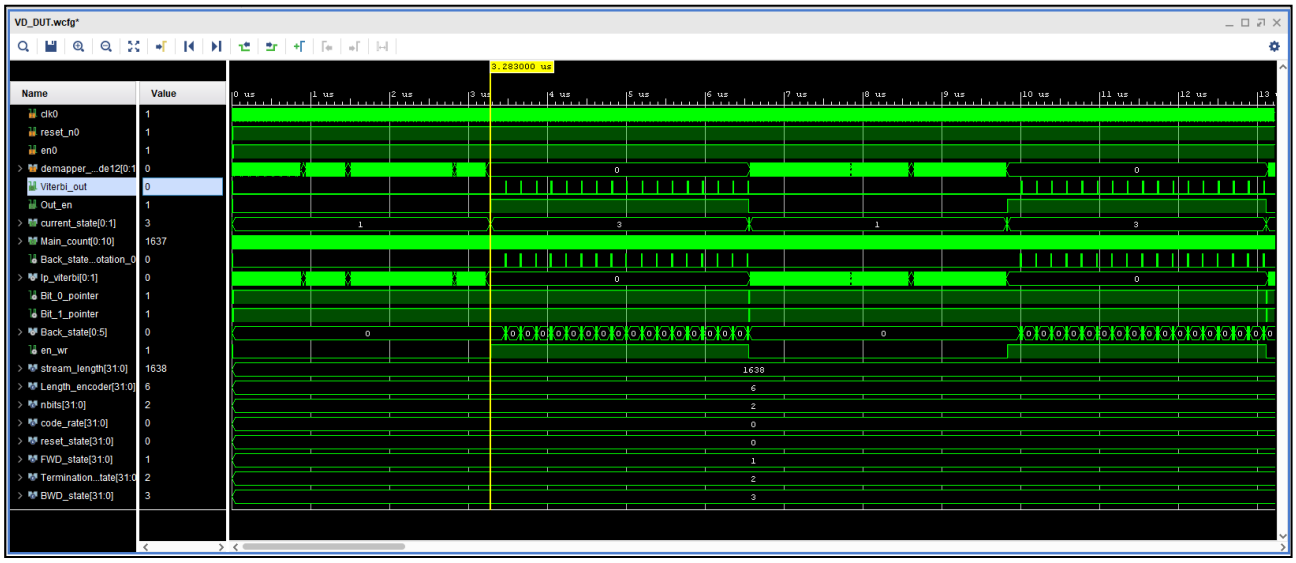

Fig. 12: Functional simulation waveform of a 1/2 code rate Viterbi decoder for a DVB Tx-Rx.

5.3. Hardware Validation

To validate the correct functionality of digital modules on the FPGA, a comprehensive hardware validation process is implemented. This involves downloading bit streams for the modules and comparing their actual behavior to simulated results. As outlined in the functional verification section, golden reference test vectors are utilized. However, for hardware validation, these vectors are stored in on-FPGA memories to serve as both input stimuli and expected output references. The DUT is then triggered by using AMD virtual input/output (VIO) IP.

The input vector from memory stimulates the DUT, and its output is compared to the expected output references using a comparator. An error signal is generated to indicate any discrepancies between the actual and expected outputs. Integrated logic analyzer (ILA) cores are employed to monitor critical signals within the DUT, enabling detailed analysis of

its behavior. Fig. 13 visualizes the output signals from Tx blocks including the Reed Solomon encoder, scrambler, and convolutional interleaver. The error signal generated by the comparator for each DUT signifies the correctness of its output. A zero-valued error signal indicates a match between the actual and expected outputs, while a non-zero value signifies an error. The test_done signal confirms the completion of the test vector sequence. Both the error_signal and test_done signals are connected to LEDs DS 38.2 and DS 40.2, respectively, for visual feedback. Fig. 14 illustrates the successful download of the bitstream for the entire Tx/Rx system, providing empirical evidence that the design methodology, RTL coding guidelines, and functional. verification procedures employed in this work are effective in hardware implementation. The illumination of LED DS 40.2 signifies the successful transmission of a 1504-bit packet, adhering to DVB standards.

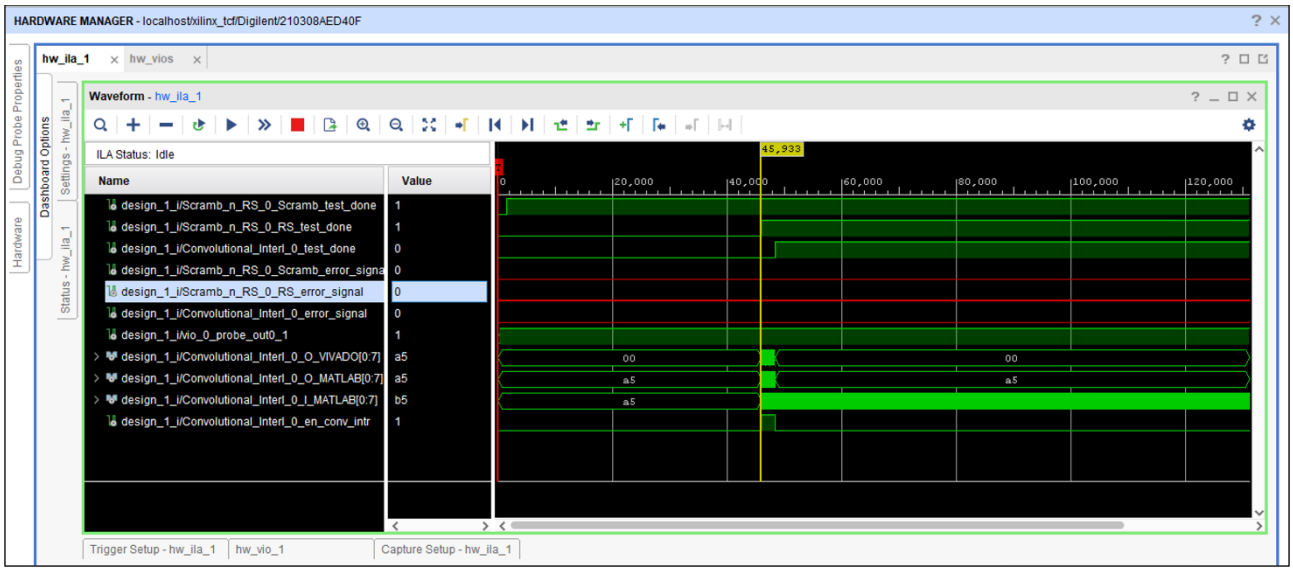

Fig. 13: Validation of hardware functionality using VIO and ILA cores on ZCU104 FPGA.

Fig. 14: FPGA-based implementation of a DVB Tx/Rx system utilizing a ZCU104 platform. The figure illustrates the bitstream download process for Tx/Rx modules.

6. Conclusions

This paper presents an investigation into how on-chip FPGA memory configuration influences power consumption and hardware complexity in Viterbi decoders within DVB systems, which necessitate substantial memory resources to handle high data rates.
It's proved that employing BRAM for the SMU and CLB flip-flops for the PMM minimizes both power consumption and FPGA resource utilization, compared to configurations using URAM, LUTRAM, and CLB flip-flops in the SMU and BRAM, URAM, and LUTRAM in the PMM on the xczu7ev-2ffvc1156 FPGA. This BRAM-FF configuration achieves a 71% reduction in power consumption and resource utilization relative to the default VIVADO synthesis tool mapping, highlighting the substantial impact of memory configuration on these metrics. The VD is implemented and validated on a 16 nm xczu7ev-2ffvc1156 FPGA, operating at a clock frequency of 125 MHz and achieving a throughput of 62.3 Mbps. The design is evaluated for a code rate of 1/2 and 16-APSK modulation scheme under AWGN channel conditions with a 0 dB SNR. Implementing this configuration in the Viterbi decoder of the DVB system yields a 50% reduction in power consumption.

References

[1] P. Garcia, D. Bhowmik, R. Stewart, G. Michaelson, and A. Wallace, "Optimized Memory Allocation and Power Minimization for FPGA-Based Image Processing", Journal of Imaging, vol. 5, no. 1, 2019.
[2] Digital Video Broadcasting (DVB); framing structure, channel coding and modulation for digital terrestrial television, ETSI EN 300 744, January 2009.
[3] P. J. Black and T. H. Meng, "A 140-Mb/s, 32-state, radix-4 Viterbi decoder," in IEEE Journal of Solid-State Circuits, vol. 27, no. 12, pp. 1877-1885, Dec. 1992.
[4] M. Véstias, H. Neto and H. Sarmento, "Design of High-Speed Viterbi Decoders on Virtex-6 FPGAs," 2012 15th Euromicro Conference on Digital System Design, Cesme, Turkey, 2012, pp. 938-945.
[5] C. -J. Chen, C. Yu, M. -H. Yen, P. -A. Hsiung and S. -J. Chen, "Design of a low power viterbi decoder for wireless communication applications," IEEE International Symposium on Consumer Electronics (ISCE 2010), Braunschweig, Germany, 2010, pp. 1-4.
[6] R. Bhattacharjya et al., "Locate: Low-Power Viterbi Decoder Exploration using Approximate Adders," Proceedings of the Great Lakes Symposium on VLSI 2023, TN, Knoxville, USA, 2023, pp. 409-413.
[7] AMD, " UltraScale Architecture Configurable Logic Block," UG574 (v1.5) [online], September 24, 2021. Available: https://docs.amd.com/r/en-US/ug574-ultrascale-clb
[8] AMD, " UltraScale Architecture Memory Resources User Guide," UG573 (v1.13) [online], September 24, 2021. Available: https://docs.amd.com/v/u/en-US/ug573-ultrascale-memory-resources.
[9] AMD, "ZCU104 Evaluation Board User Guide," UG1267 (v1.1) [online], October 9, 2018. Available: https://docs.amd.com/v/u/en-US/ug1267-zcu104-eval-bd.
[10] AMD, "Device Reliability Report," UG116 (v10.18) [online], June 25, 2024. Available: https://docs.amd.com/r/en-US/ug116.
[11] J. Jin and C. Y. Tsui, "Low-power limited-search parallel state Viterbi decoder implementation based on scarce state transition," IEEE Transaction on Very Large-Scale Integration (VLSI) System, Vol. 15, no. 10, October 2007.
[12] Kalavathi Devi, T., Priyanka, E.B., Sakthivel, P. et al., "Sleepy keeper style based Low Power VLSI Architecture of a Viterbi Decoder applying for the Wireless LAN Operation sustainability," Analog Integr Circ Sig Process, vol. 109, no. 3 , pp. 487-499, 2021.
[13] M. J. Flynn, "Area - time - power and design effort: the basic tradeoffs in application specific systems," 2005 IEEE International Conference on Application-Specific Systems, Architecture Processors

(ASAP'05), Samos, Greece, 2005, pp. 3-6.
[14] A. Mosbeh, A. A. Y. Ibraheem, H. Mostafa, and K. Yousef, "Low Power Microarchitecture Designs of ACS Block in Viterbi Decoder: A Review", in Proceedings of the 2023 13th International Conference on Information Communication and Management, Cairo, Egypt, 2024, pp. 16–20.
[15] L. M. G. Rocha et al., "Physical implementation of an ASIC-oriented SRAM-based viterbi decoder," 2017 24th IEEE International Conference on Electronics, Circuits and Systems (ICECS), Batumi, Georgia, 2017, pp. 526-529.
[16] N. Prasad, I. Chakrabarti and S. Chattopadhyay, "An Energy-Efficient Network-on-Chip-Based Reconfigurable Viterbi Decoder Architecture," in IEEE Transactions on Circuits and Systems I: Regular Papers, vol. 65, no. 10, pp. 3543-3554, Oct. 2018.
[17] C. Yu, Chien-Hung Kuo, Chen-Hen Sung, Mao-Hsu Yen and Sao-Jie Chen, "Design of a low-power OFDM baseband receiver for wireless communications," 2012 IEEE International Conference on Consumer Electronics (ICCE), Las Vegas, NV, 2012, pp. 548-549.
[18] K. Cholan, "Design and Implementation of Low Power High Speed Viterbi Decoder", Procedia Engineering, vol. 30, pp. 61–68, 2012.
[19] Varada, S., Katpally, S. & Thiruveedhi, S.S.L. "Comprehensive Analysis and Optimization of Reliable Viterbi Decoder Circuits Implemented in Modular VLSI Design Logic Styles," J Electron Test, vol.36, pp. 343–363, 2020.
[20] D. A. F. Ei-Dib and M. I. Elmasry, "Low-power register-exchange Viterbi decoder for high-speed wireless communications," 2002 IEEE International Symposium on Circuits and Systems (ISCAS), Phoenix-Scottsdale, AZ, USA, 2002, pp. V-V.
[21] R. Liu and K. K. Parhi, "Low-Latency Low-Complexity Architectures for Viterbi Decoders," in IEEE Transactions on Circuits and Systems I: Regular Papers, vol. 56, no. 10, pp. 2315-2324, Oct. 2009.
[22] R. A. Rashid, H. Harun, Z. Mansor, N. Shamsudin and S. M. Nor, "Pruning the algorithm complexity of the Add-Compare Select Unit (ACSU) for the Viterbi Decoder - A Review," 2018 IEEE 5th International Conference on Smart Instrumentation, Measurement and Application (ICSIMA), Songkhla, Thailand, 2018, pp. 1-5.
[23] M. Irfan, Z. Ullah, M. H. Chowdhury and R. C. C. Cheung, "RPE-TCAM: Reconfigurable Power-Efficient Ternary Content-Addressable Memory on FPGAs," in IEEE Transactions on Very Large-Scale Integration (VLSI) Systems, vol. 28, no. 8, pp. 1925-1929, Aug. 2020, doi: 10.1109/TVLSI.2020.2993168.
[24] P. Musil, R. Juránek, M. Musil and P. Zemčík, "Cascaded Stripe Memory Engines for Multi-Scale Object Detection in FPGA," in IEEE Transactions on Circuits and Systems for Video Technology, vol. 30, no. 1, pp. 267-280, Jan. 2020, doi: 10.1109/TCSVT.2018.2886476.
[25] Tessier, R., et al, "Power-aware RAM mapping for FPGA embedded memory blocks," in Proceedings of the 2006 ACM/SIGDA 14th International Symposium on Field Programmable Gate Arrays, 2006, pp. 189–198.
[26] R. Tessier, V. Betz, D. Neto, A. Egier and T. Gopalsamy, "Power-Efficient RAM Mapping Algorithms for FPGA Embedded Memory Blocks," in IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, vol. 26, no. 2, pp. 278-290, Feb. 2007, doi: 10.1109/TCAD.2006.887924.
[27] AMD, " Vivado Design Suite User Guide: Synthesis," UG901 [online], December 24, 2024. Available: https://docs.amd.com/r/en-US/ug901-vivado-synthesis.
[28] Namratha and Bakhar, "Power and area optimized adaptive Viterbi decoder for high-speed communication applications", International Journal of Information Technology, vol. 15, no. 1, pp. 45–52, Jan. 2023.
[29] B. N. Akash, K. M. Amogh, S. Sridhar, and P. Agarwal, "Viterbi Decoder with Configurable Constraint Length with Bit Error Correction for Satellite Communication", in Information and Communication Technology for Competitive Strategies (ICTCS 2020), 2022, pp. 697–707.
[30] M. A. Abu, H. Harun, M. Yazdi Harmin, N. I. Abdul Wahab, and M. K. Abdul Kadir, "The design of Viterbi decoder for low power consumption space time trellis code without adder architecture using RTL model", World Journal of Engineering, vol. 13, no. 6, pp. 540–546, Jan. 2016.
[31] C. E. Stroud, L.-T. (L-T.). Wang, and Y.-W. Chang, "Chapter 1 - Introduction," in Electronic Design Automation, L.-T. Wang, Y.-W. Chang, and K.-T. (tim) Cheng, Eds. Boston: Morgan Kaufmann, 2009, pp. 1–38.