Area and Energy Efficient Viterbi Accelerator for Embedded Processor Datapaths

Abdul Rehman Buzdar*, Liguo Sun*, Muhammad Waqar Azhar‡, Muhammad Imran Khan†§, Rao Kashif†
*Department of Electronic Engineering and Information Science
†Micro/Nano Electronic System Integration R & D Center (MESIC)
University of Science and Technology of China (USTC), Hefei, China
‡Department of Computer Science and Engineering, Chalmers University of Technology, Gothenburg, Sweden
§Department of Electronics Engineering, University of Engineering and Technology Taxila, Pakistan

***Abstract*—Viterbi algorithm is widely used in communication systems to efficiently decode the convolutional codes. This algorithm is used in many applications including cellular and satellite communication systems. Moreover, Serializer-deserializers (SERDESs) having critical latency constraint also use viterbi algorithm for hardware implementation. We present the integration of a mixed hardware/software viterbi accelerator unit with an embedded processor datapath to enhance the processor performance in terms of execution time and energy efficiency. Later we investigate the performance of viterbi accelerated embedded processor datapath in terms of execution time and energy efficiency. Our evaluation shows that the viterbi accelerated Microblaze soft-core embedded processor datapath is three times more cycle and energy efficient than a datapath lacking a viterbi accelerator unit. This acceleration is achieved at the cost of some area overhead.**

***Keywords*—Viterbi decoder; Codesign; FPGA; MicroBlaze; Embedded Processor**

I. Introduction

Channel coding is used in wireless communication systems for reliable data transfer over noise prone communication channels. Various forward error correction (FEC) schemes e.g. Low-density parity-check (LDPC), Reed Solomon, Viterbi and Turbo codes are used to meet the growing need to improve the spectrum efficiency [1], [2], [3], [4], [5]. In FEC schemes the encoding of data is done using convolutional encoding and at the receiver end the decoding process is done by viterbi or turbo decoders [21-31]. The viterbi decoder is suitable in wireless communication systems in which the transmitted signals are corrupted by additive white Gaussian noise [6].

The decoding process in FEC schemes is computationally intensive and power hungry. The hand held devices are battery powered, so they must be energy efficient. The customized hardware implementation of these FEC decoders are performance and power efficient but lacks flexibility. As the wireless standards evolve with time, so the hardware needs to be flexible. The viterbi decoder can be implemented in software and executed on an embedded processor but it will require a lot of clock cycles. The viterbi decoder can be implemented more efficiently in dedicated hardware which will require few clock cycles at the cost of flexibility. The high speed communication systems today requires fast data rates which can only be delivered using dedicated hardware solutions.

Different hardware modules like USB, Ethernet, TCP/IP, CRC and CAN protocol are included in modern embedded processors [7], [8], [9] to speedup certain parts of application in areas like signal processing, communication and control systems. This provides effective use of viterbi accelerator in programming systems where a series of viterbi decoding is required to be computed.

II. Convolutional Encoding and Viterbi Decoding

Convolutional encoding of data is implemented with a shift register having $K-1$ memory elements and cascaded network of exclusive-or gates. Here K is the constraint length and having 2^{K+1} encoder states. The shift register is a chain of flip-flops and the output of nth flip-flop goes as input into the $(n+1)th$ flip-flop. The data in the registers is shifted to the next register and the value in the last register gets discarded. The combinational logic consisting of exclusive-or gates is used to perform modulo-2 addition. The encoder outputs n symbols using generator polynomials and values in the shift register. Fig. 1 shows a convolutional encoder for $K = 3$, $R = 1/2$ and generator polynomials $G1 = (1, 1, 1)$ and $G2 = (1, 0, 1)$. The code rate is the ratio of the number of input bits to the number of output bits $(R = m/n)$. The reason for the convolutional codes being efficient compared to block codes is the fact that every input bit has an impact on K successive output symbols [10]. The value of K is directly proportional to the code complexity and error correction capability. The decoder complexity and memory requirements increases with increasing K.

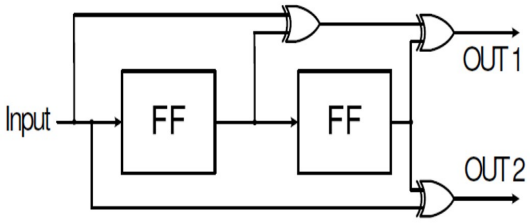

Figure 1: Convolutional Encoder general architecture.

Trellis diagram is used to visualize the state transitions of an encoder, as shown in Fig. 2. The black lines represent input bit 0 and the dotted lines represent input bit 1. The trellis path of input sequence is represented by the red lines. The basic concept is that the valid path through trellis diagram is

generated by the sequence of input bits from left to right. The viterbi decoder is able to find the valid path on trellis which is closest match when some transmission error occurs [11]. In start the reset state of encoder is "00". If the input is 0 the encoder state will become 00 as shown by the black line. The encoder will transmit 00 as output. The viterbi decoder reconstructs the valid input bit. If the input bit is 1 the decoder goes to state "10" and "11" is transmitted.

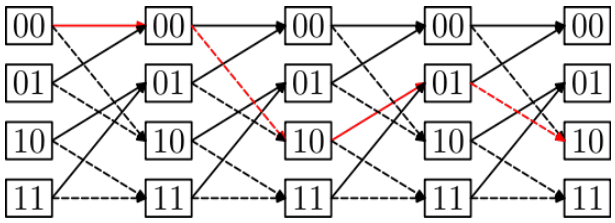

Figure 2: Trellis diagram.

The main concept of viterbi decoder is mapping received symbols to most likely valid sequence. The decoding process consist of following steps.

1) Branch Metric Unit: In this step difference is calculated between received symbol and every possible encoder output combinations. In hard decision decoder the difference is the Hamming distance and in case of soft decision decoder the Euclidean distance is used. There can be 2^K output combinations for an encoder with 2^{K+1} states and 0 or 1 as input bit.

2) Path Metric Unit: This step is very computationally intensive. It performs add compare select (ACS) operation on branch metric which comes from previous step to calculate path metric which is accumulated distance. The branch having biggest accumulated distance gets discarded.

3) Trace Back Unit: In trace back unit accumulated column error metrics are traced back beginning from the last smallest metric value. The next step in trace back unit is finding previous two possible states and the state with smallest entry is picked. They are stored in survivor state table. These steps are continued until metric table's first column is reached. The survivor table state transitions are used to recreate original message in the last step of the viterbi decoding process.

The decoder output table size is $2^{K+1}.2^m.n$ bits. Where as the size of metric table is $2^{k-1}.b.(5k+1)$ bits. Here b represent number of bits of every entry in metric table. Implementing viterbi decoder in a memory efficient way is a challenging task. For every symbol output table's each entry is accessed once. During decoding process the output table is accessed 2^k times and every entry of metric table is accessed two times for each symbol. The calculation of one entry of the metric table requires two distance calculations and one ACS operation. To calculate one metric table column for each received symbol 2^{k-1} ACS operations and 2^k distance calculations are needed. This process is done for every symbol.

III. Viterbi Accelerator Unit

The aim of this paper is to design and integrate a viterbi accelerator unit with Microblaze soft-core processor datapath. To enhance the processor performance in terms of execution time and energy efficiency. The integration of accelerator will have an impact on the performance of processor. So the accelerator unit should be area, timing and power efficient.

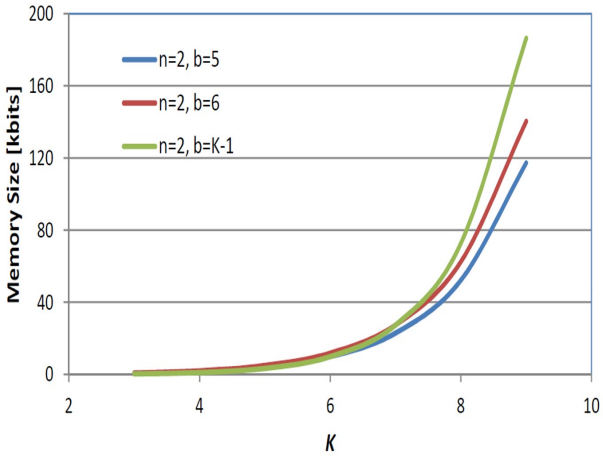

Figure 3: Metric Table memory size variation.

A. Initial Viterbi Decoder

The initial viterbi decoder consist of one ACS unit and one hamming distance calculation unit. For every ACS operation hamming distance unit is used twice sequentially. The control unit is implemented as a state machine. The initial viterbi decoder is shown in Fig. 4. Fig. 3 shows the impact of increasing constraint length K. Here b represents each metric table entry bits and n represent number of output bits. This initial implementation of viterbi decoder with different constraint lengths was synthesized on 7vx485tffg1157-3 Virtex-7 FPGA device which is based on a 28nm technology. Fig. 7 shows the area of implementation for three different constraint lengths. As can be seen the area of decoder increases exponentially with increasing constraint length K. It is observed that major portion of decoder area is consumed by metric table.

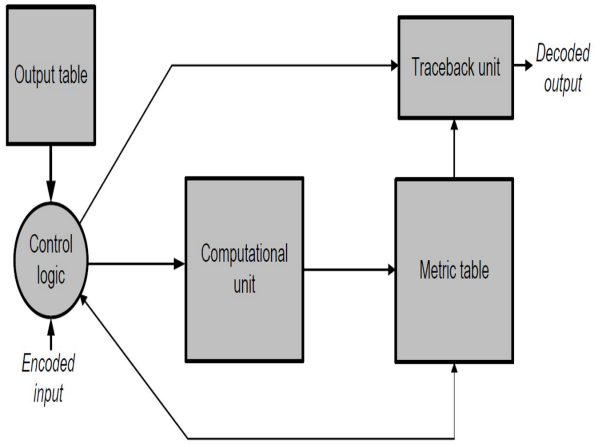

Figure 4: Initial Viterbi decoder architecture.

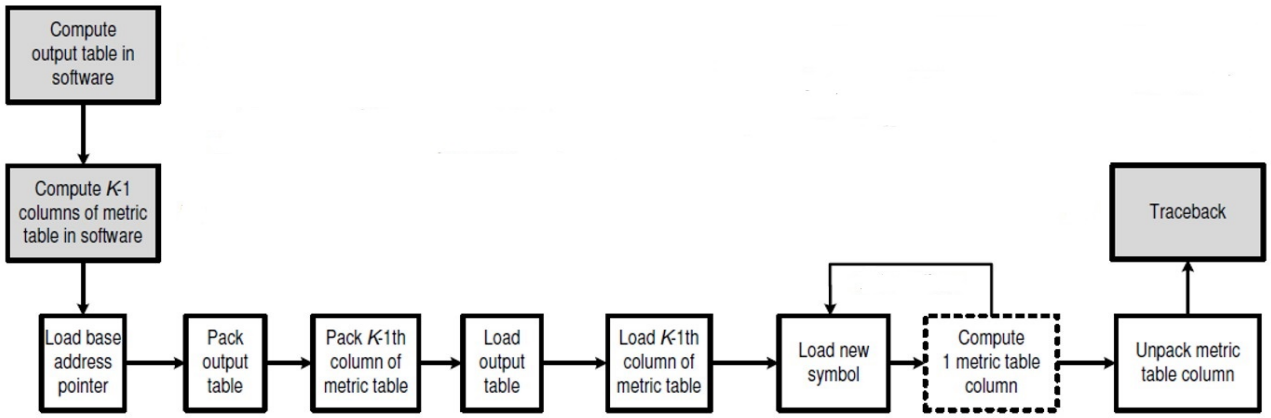

Figure 5: Flow chart for Viterbi Decoder in Full mode.

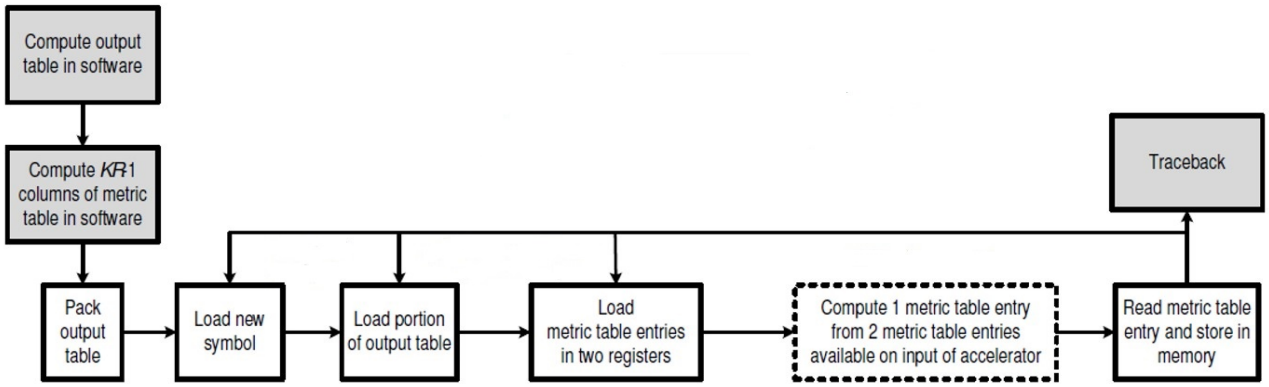

Figure 6: Flow chart for Viterbi Decoder in Sub-State mode.

B. Mixed HW/SW Viterbi Accelerator

The mixed hardware/software approach helps to achieve a good balance between flexibility and performance. We intend to implement the portion of viterbi code in part of accelerator which is computational and memory intensive. The remaining portion of the code which is not frequently executed will be handled by the processor. The decision to define a suitable boundary between hardware and software in designing accelerator-centric heterogeneous systems is a challenging task. Based on the analysis done in the previous section we have made the following conclusions:

- The branch metric and path metric are computational intensive calculations and repeating steps.
- Output table is initialized once.
- After the computation of complete metric table, Trace-back is needed once.
- Output table and previous column of metric table is needed for the computation of new column.

Based on these observations we intend to perform branch metric and path metric calculations in hardware part of mixed hardware/software viterbi accelerator. The output table and traceback computations are done in software and executed on the Microblaze soft-core processor. As the metric table is very computation-intensive and its parallelism can be exploited in hardware implementation. The last metric table column is important for the computation of next column that is why they are stored in the local memory of hardware accelerator. The full metric table is kept in main memory of processor which is needed for trace-back operation. Fig. 8 shows the mixed Hardware/Software viterbi decoder having four computational blocks. Every computational block has one ACS unit and two Euclidian distance computation units for increasing the throughput. This viterbi accelerator is capable to support any

Table I: Synthesis Results of Viterbi Accelerator

Power	241mW
Max Freq	179.808MHz
Latency	5.562ns
Slice Registers	1087
Slice LUTs	3079
Occupied Slices	1038

constraint lenght K. The full metric table computations are performed in hardware when applications constraint length is less than or equal to viterbi accelerator constraint length. Fig. 5 shows flow chart for viterbi decoding in Full mode. Whereas in situations in which the constraint length of applications is greater than viterbi hardware accelerator constraint length and accelerator memory is not enough then Sub-State mode is used. In this mode metric table value is received from the MicroBlaze processor register file. Fig. 6 shows the flow chart for the steps performed in Sub-State mode by the mixed Hardware/Software viterbi decoder. The gray boxes in Fig. 5 and Fig. 6 represent part of the code that is executed in software, whereas the transparent boxes show the steps done in hardware accelerator.

IV. Integration of Viterbi Accelerator Unit with MicroBlaze Processor

We have implemented the viterbi accelerator unit in VHDL hardware description language and verified it using Xilinx ISE design suit [12]. We used Xilinx Spartan-6 FPGA SP605 Evaluation Kit [14] and Xilinx Embedded Development Kit(EDK) [12] for the implementation. Xilinx Microblaze soft-core processor [13] was used to run the software implementation of viterbi decoder. The Hardware/Software co-design is a well established technique which improves the performance of the system [16-20]. There are two ways to integrate a hardware accelerator core into a MicroBlaze-based embedded soft processor system. One way is to connect the accelerator through the Processor Local Bus (PLB). The second way is to connect it using MicroBlaze dedicated Fast Simplex Link (FSL) bus system [15]. First PLB was tried but it was taking a lot of cycles. Because it is a traditional memory mapped transaction bus. Then it was decided to integrate our viterbi accelerator unit using a dedicated FIFO style FSL Bus with the MicroBlaze processor system, shown in Fig. 9.

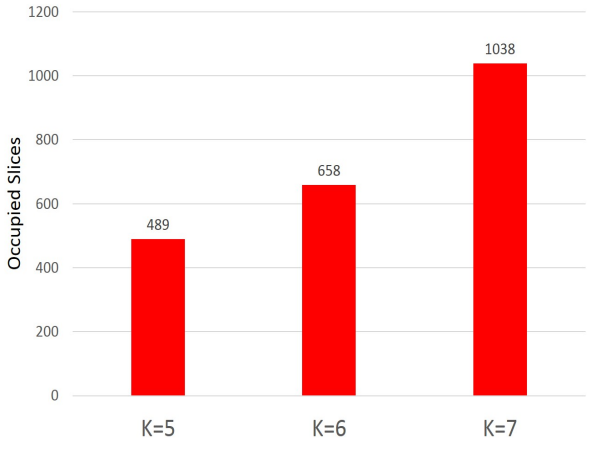

Figure 7: Total area for different constraint lengths.

The software only C code for viterbi decoder was implemented and verified. Later this C code was executed on the MicroBlaze processor using Xilinx Software Development Kit (SDK) [12]. The cycle count for the complete software implementation of viterbi was measured using the XPS hardware timer block, shown in Table II. Fig. 10 and 11 shows the cycle

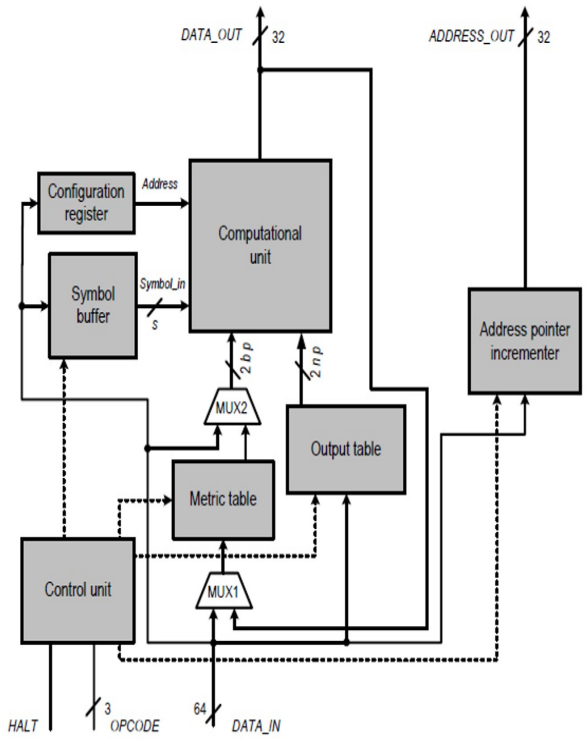

Figure 8: Mixed HW/SW Viterbi decoder architecture.

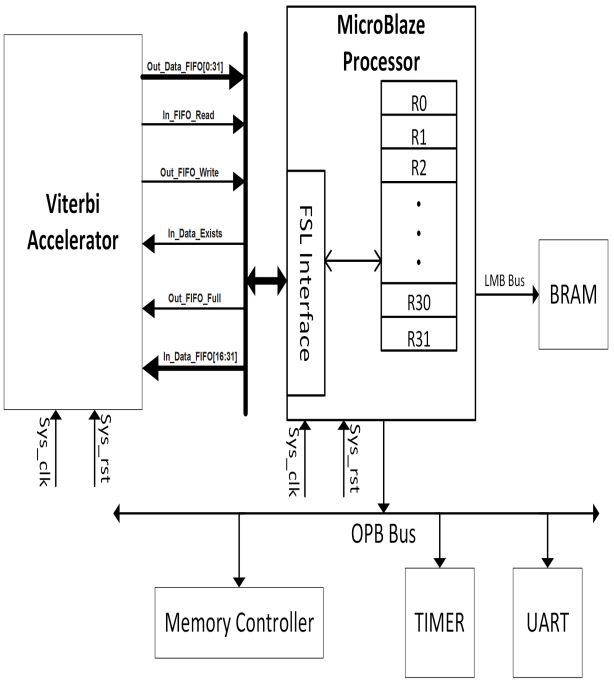

Figure 9: Viterbi Accelerator Unit with MicroBlaze Processor System

count and energy dissipation of two Viterbi implementations, respectively.

The viterbi accelerator unit was attached with the Mi-

Table II: Cycle Count and Energy Dissipation at Clock Period 20ns

Architecture	#Cycles	Power (mW)	Energy* (μJ)
Software Only	8312	178	29.590
Accelerated	2518	185	9.3166

*: Energy = #cycles $\times$ clock period $\times$ power.

croblaze processor system via FSL bus using Xilinx Platform Studio (XPS) [12]. The software part of viterbi accelerator unit was implemented in C programming with Xilinx SDK. The predefined C functions of SDK were used to communicate with hardware part of viterbi accelerator unit via FSL bus. Our evaluation shows that an accelerated MicroBlaze processor datapath is three times more cycle and energy efficient than a datapath lacking viterbi accelerator. This acceleration is achieved at some area overhead.

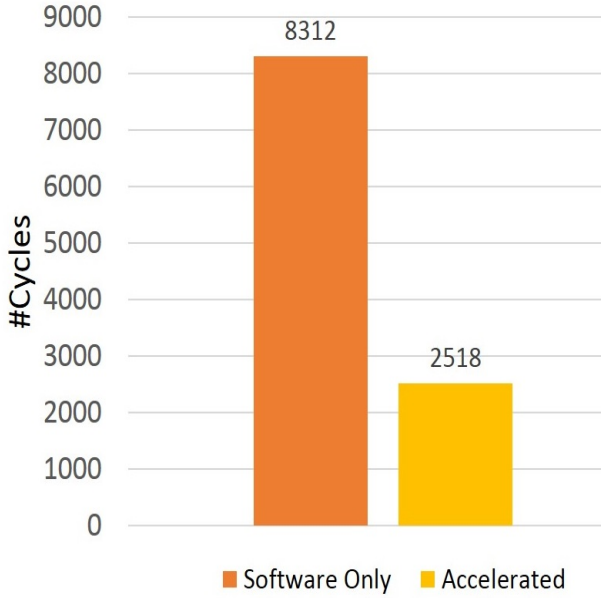

Figure 10: Cycle count of two Viterbi implementations.

V. Conclusion

In this paper, we have designed a mixed hardware/software viterbi accelerator unit using VHDL. We have integrated the viterbi accelerator unit with the Microblaze soft-core processor system using FSL Bus to enhance the processor performance in terms execution time and energy efficiency. We used Xilinx Spartan-6 FPGA Evaluation Kit and Xilinx Embedded Development Kit (EDK) for the implementation. We have shown that a viterbi accelerated Microblaze embedded processor datapath is three times more cycle and energy efficient that a datapath lacking a viterbi accelerator. This acceleration is achieved at the cost of some area overhead.

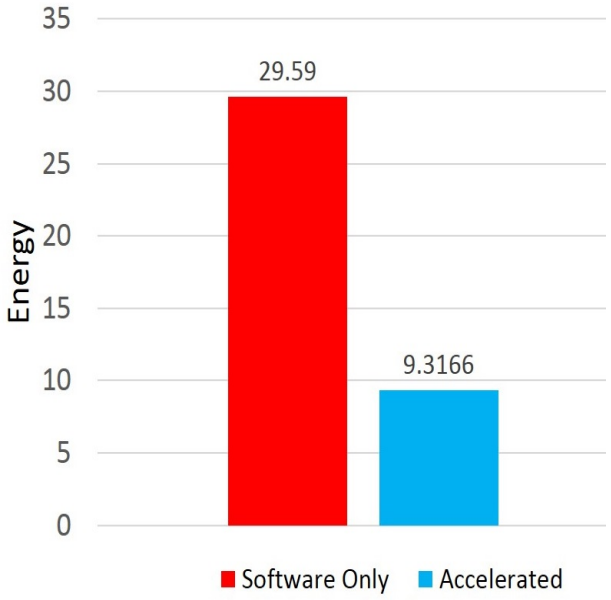

Figure 11: Energy dissipation of two Viterbi implementations.

Acknowledgment

This work is partially supported by the Chinese Academy of Sciences and The World Academy of Sciences CAS-TWAS President's Fellowship 2013-2017.

References

[1] M. F. Brejza, L. Li, R. G. Maunder, B. Al-Hashimi, C. Berrou, L. Hanzo, "20 years of turbo coding and energy-aware design guidelines for energy-constrained wireless applications", IEEE Commun. Surveys Tuts., vol. 18, no. 1, pp. 8-28, 1st Quart. 2016.

[2] Mehran Mozaffari Kermani, Vineeta Singh, Reza Azarderakhsh, "Reliable Low-Latency Viterbi Algorithm Architectures Benchmarked on ASIC and FPGA," IEEE Transactions on Circuits and Systems I: Regular Papers, vol. 64, no. 1, pp. 208-216, 2017.

[3] Linjia Chang, Avhishek Chatterjee, Lav R. Varshney, "Performance of LDPC Decoders With Missing Connections," IEEE Transactions on Communications, vol. 65, no. 2, pp. 511-524, 2017.

[4] Salvatore Pontarelli, Pedro Reviriego, Marco Ottavi, Juan Antonio Maestro, "Low Delay Single Symbol Error Correction Codes Based on Reed Solomon Codes," IEEE Transactions on Computers, vol. 64, no. 5, pp. 1497-1501, 2015.

[5] G. Krishnaiah, N. Engin, and S. Sawitzki, "Scalable Reconfigurable Channel Decoder Architecture for Future Wireless Handsets," in IEEE Design, Automation Test in Europe Conference, Apr. 2007, pp. 1-6.

[6] R. Johannesson and K. S. Zigangirov, "Fundamentals of Convolutional Coding". Wiley-IEEE Press, 1999.

[7] Atmel, "Secure microcontroller for smart cards." [Online]. Available: http://www.atmel.com

[8] Freescale, "MAPLE hardware accelerator and SC3850 DSP core." [Online]. Available: http://www.freescale.com

[9] Microchip, "PIC32mx775f512l datasheet." [Online]. Available: http://www.microchip.com

[10] O. O. Khalifa, T. Al-Maznaee, M. Munjid, and A.-H. A. Hashim, "Convolution Coder Software Implementation Using Viterbi Decoding Algorithm," J. Computer Science, vol. 4, no. 10, pp. 847-856, 2008.

[11] G. D. Forney, Jr., "The Viterbi Algorithm," Proceedings of the IEEE, vol. 61, no. 3, pp. 268278, Mar. 1973.

[12] Xilinx Inc. FPGA Design Tools. Silicon Devices. [Online]. Available: http://www.xilinx.com

[13] Xilinx MicroBlaze [Online] www.xilinx.com/tools/microblaze.htm

[14] Xilinx Spartan-6 FPGA SP605 Evaluation Kit. [Online] Available: www.xilinx.com/products/boards-and-kits/ek-s6-sp605-g.html

[15] Xilinx Fast Simplex Link (FSL). [Online] Available: http://www.xilinx.com/products/intellectual-property/fsl.html

[16] Abdul Rehman Buzdar, Liguo Sun, Azhar Latif and Abdullah Buzdar, "Distance and Speed Measurements using FPGA and ASIC on a high data rate system" International Journal of Advanced Computer Science and Applications(IJACSA), 6(10), 2015, pp.273-282.

[17] Abdul Rehman Buzdar, Liguo Sun, Azhar Latif and Abdullah Buzdar, "Instruction Decompressor Design for a VLIW Processor", Informacije MIDEM-Journal of Microelectronics, Electronic Components and Materials Vol. 45, No. 4 (2015), pp.225-236.

[18] Abdul Rehman Buzdar, Azhar Latif, Liguo Sun and Abdullah Buzdar, "FPGA Prototype Implementation of Digital Hearing Aid from Software to Complete Hardware Design" International Journal of Advanced Computer Science and Applications(IJACSA), 7(1), 2016, pp.649-658.

[19] Abdul Rehman Buzdar, Liguo Sun, Shoab Ahmed Khan, Abdullah Buzdar, "Area and Energy efficient CORDIC Accelerator for Embedded Processor Datapaths" Informacije MIDEM-Journal of Microelectronics, Electronic Components and Materials Vol. 46, No. 4(2016), pp.197-208

[20] Abdul Rehman Buzdar, Liguo Sun, Rao Kashif, Muhammad Waqar Azhar, Muhammad Imran Khan, "Cyclic Redundancy Checking (CRC) Accelerator for Embedded Processor Datapaths" International Journal of Advanced Computer Science and Applications(IJACSA), 8(2), 2017, pp.321-325.

[21] Muhammad Waqar Azhar, Magnus Sjlander, Hasan Ali, Akshay Vijayashekar, Tung Thanh Hoang, K. K. Ansari, and Per Larsson-Edefors, "Viterbi Accelerator for Embedded Processor Datapaths," in Proc. of IEEE Int. Conf. on Applicationspecific Systems, Architectures and Processors, 2012.

[22] J. Heller and I. Jacobs, Viterbi Decoding for Satellite and Space Communication, IEEE Trans. Communication Technology, vol. 19, no. 5, pp. 835848, Oct. 1971.

[23] T. Gemmeke, M. Gansen, and T. G. Noll, "Implementation of Scalable Power and Area Efficient High-Throughput Viterbi Decoders," IEEE J. Solid-State Circuits, vol. 37, no. 7, pp. 941-948, Jul. 2002.

[24] M. Kawokgy and C. A. T. Salama, "A Low-Power CSCD Asynchronous Viterbi Decoder for Wireless Applications," in Proc. Int. Symp. Low Power Electronics and Design, 2007, pp. 363-366.

[25] M. Kamuf, V. wall, and J. B. Anderson, "Optimization and Implementation of a Viterbi Decoder Under Flexibility Constraints," IEEE Trans. Circuits and Systems I: Regular Papers, vol. 55, no. 8, pp. 2411-2422, Sep. 2008.

[26] M. A. Anders, S. K. Mathew, S. K. Hsu, R. K. Krishnamurthy, and S. Borkar, "A 1.9 Gb/s 358 mW 16-256 State Reconfigurable Viterbi Accelerator in 90 nm CMOS," IEEE J. Solid-State Circuits, vol. 43, no. 1, pp. 214-222, Jan. 2008.

[27] C.-C. Lin, Y.-H. Shih, H.-C. Chang, and C.-Y. Lee, "A Low Power Turbo/Viterbi Decoder for 3GPP2 Applications," IEEE Trans. Very Large Scale Integration (VLSI) Systems, vol. 14, no. 4, pp. 426-430, Apr. 2006.

[28] M. A. Bickerstaff et al., "A Unified Turbo/Viterbi Channel Decoder for 3GPP Mobile Wireless in 0.18-m CMOS," IEEE J. Solid-State Circuits, vol. 37, no. 11, pp. 1555-1564, Nov. 2002.

[29] J. R. Cavallaro and M. Vaya, "Viturbo: A Reconfigurable Architecture for Viterbi and Turbo Decoding," in Proc. IEEE Int. Conf. Acoustics, Speech, and Signal Processing, vol. 2, Apr. 2003, pp. 497-500.

[30] D. E. Hocevar and A. Gatherer, "Achieving Flexibility in a Viterbi Decoder DSP Coprocessor," in Proc. 52nd IEEE Vehicular Technology Conf., vol. 5, 2000, pp. 2257-2264, vol.5.

[31] A. Niktash, H. Parizi, and N. Bagherzadeh, "A Reconfigurable Processor for Forward Error Correction," in Proc. Int. Conf. on Architecture of Computing Systems, 2007, pp. 1-13.