International Journal of Computer Applications (0975 – 8887)
Volume 88 – No.14, February 2014

Virtex 4 FPGA Implementation of Viterbi Decoded 64-bit RISC for High Speed Application using Xilinx

Ritam Dutta
Dept. of Electronics & Communication Engineering
SIEM, West Bengal University of Technology
Siliguri, India

Charles Van Der Mast
Dept. of Electrical Engg., Mathematics & Comp. Sc
Delft University of Technology
Delft, The Netherlands

ABSTRACT

In the modern era of electronics and communication decoding and encoding of any data(s) using VLSI technology requires low power, less area and high speed constrains. The viterbi decoder using survivor path with necessary parameters for wireless communication is an attempt to reduce the power and cost and at the same time increase the speed compared to normal decoder. This paper presents three objectives. Firstly, an orthodox viterbi decoder is designed and simulated. For faster process application, the Gate Diffused Input Logic (GDIL) based viterbi decoder is designed using Xilinx ISE, simulated and synthesized successfully. The new proposed GDIL viterbi provides very less path delay with low power simulation results. Secondly, the GDIL viterbi is again compared with our proposed technique, which comprises a Survivor Path Unit (SPU) implements a trace back method with DRAM. This proposed approach of incorporating DRAM stores the path information in a manner which allows fast read access without requiring physical partitioning of the DRAM. This leads to a comprehensive gain in speed with low power effects. Thirdly, all the viterbi decoders are compared, simulated, synthesized and the proposed approach shows the best simulation and synthesize results for low power and high speed application in VLSI design. The Add-Compare-Select (ACS) and Trace Back (TB) units and its sub circuits of the decoder(s) have been operated in deep pipelined manner to achieve high transmission rate. Although the register exchange based survivor unit has better throughput when compared to trace back unit, but in this paper by introducing the RAM cell between the ACS array and output register bank, a significant amount of reduction in path delay has been observed. All the designing of viterbi is done using Xilinx ISE 12.4 and synthesized successfully in the FPGA Virtex 4 target device operated at 64.516 MHz clock frequency, reduces almost 41% of total path delay.

Keywords

Viterbi decoder, GDIL technique, SPU, DRAM, Add Compare Select (ASC), Trace Back (TB), Xilinx, FPGA.

1. INTRODUCTION

The Viterbi decoding algorithm, proposed by Viterbi, is a decoding process for convolutional codes in memory-less noise. The algorithm can be applied to a host of problems encountered in the design of communication systems. The Viterbi Algorithm finds the most-likely state transition sequence in a state diagram, given a sequence of symbols. The Viterbi algorithm is used to find the most likely noiseless finite-state sequence, given a sequence of finite-state signals that are corrupted by noise.

Generally, a viterbi decoder consists of three basic computation units: Branch Metric Unit (BMU), Add-Compare-Select Unit (ACSU) and Trace Back Unit (TBU). The BMU calculates the branch metrics by the hamming distance or Euclidean distance and the ACSU calculates a summation of the branch metric from the BMU and previous state metrics, which are called the path metrics.

After this summation, the value of each state is updated and then the survivor path is chosen by comparing path metrics. The TBU processes the decisions made in the BMU and ACSU and outputs the decoded data. The feedback loop of the ACSU is a major critical path for the viterbi decoder. The operations of the convolutional encoder and decoder for constraint length of 3 are explained as follows.

The convolution encoder has a rate of ½ (k/n) with a constraint length of 3. With an encoder, the 3 bit shift register provides the memory and two modulo-2 adders provide convolution operations. For each bit in the input sequence, two bits are output, one from each of the two modulo-2 adders. The decoding procedure compares the received sequence with all the possible sequences that may be obtained with the respective encoder and then selects the sequence that is closest to the received sequence. There are always two paths merging at each node and the path selected is the one with the minimum hamming distance, the other is simply terminated. The retained paths are known as survivor paths and the final path selected is the one with the continuous path through the trellis with a minimum aggregate hamming distance.

2. EARLIER RESEARCH WORK

Y. Zhu and M. Benaissa (2003) presented a novel ACS scheme that enables high speeds to be achieved in area efficient viterbi decoders without compromising for area and power efficiency. Multilevel pipelining has been introduced into the ACS feedback loop.

Arkadiy Morgenshtein et al (2004) used Gate Diffusion Input circuits for asynchronous design and compared the designs with CMOS asynchronous design [1]. Dalia A. El-Dib and Mohamed I. Elmasry (2004) discussed the implementation of a viterbi decoder based on modified register-exchange (RE) method [2]. Song Li and Qing-Ming Yi (2006) proposed a scheme based on Verilog language for the implementation of high-speed and low power consumption bi-directional viterbi decoder [3]. The decoding was done in both positive and negative direction and the delay was half of that of the unilateralism decoder and the decoding speed was greatly improved. Yun-Nan Chang and Yu-Chung Ding (2006) presented a low power design for viterbi decoder based on a novel survivor path trace mechanism. Lupin Chen et al (2007) presented a low-power trace-back (TB) scheme for high constraint length viterbi decoder. Xuan-zhong Li et al (2008) discussed a high speed viterbi decoder which was based on parallel radix-4 architecture and bit level carry-save algorithm. Seongjoo Lee (2009) presented an efficient implementation method for parallel processing viterbi decoders in UWB systems.

3. ORTHODOX VITERBI LIMITATION

From the literature survey, viterbi decoder is mainly used in all communication techniques. Logic styles like CMOS, Pseudo NMOS and Dynamic logic design of circuits at ACS level are done [4] but the switching activity in these logic styles are high and hence lead to high power dissipation.

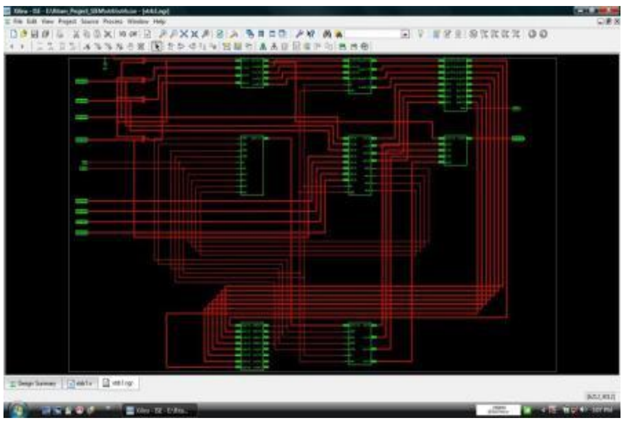

Fig 1: The RTL design of orthodox viterbi decoder

In figure 1 the RTL logic design of the old orthodox viterbi is simulated. The backend coding is done in VHDL and the circuit is synthesized using Xilinx ISE. This viterbi provides the delay report and power analysis report, which clearly depicts the high power dissipation that is undesired for any FPGA based VLSI circuit design.

4. GDIL BASED VITERBI

Gate Diffusion Input Logic (GDIL) is a technique of low power digital for circuit design which allows reducing power consumption, delay and area of the digital circuit. The basic GDIL cell is similar to the standard CMOS inverter, the differences are: (1) GDIL cell contains three inputs (2) Bulks of both NMOS and PMOS are connected to N or P, so it can be randomly biased at contrast with CMOS inverter. The GDIL contains four terminals – G (common gate input of the nMOS and pMOS transistors), P (outer diffusion node of the pMOS transistor), N (outer diffusion node of the nMOS transistor) and D node (common diffusion of both transistors). The GDIL approach [5] allows implementation of a wide range of complex logic functions using only two transistors.

This GDIL method is suitable for design of fast, low-power circuits using a reduced number of transistors (as compared to CMOS and existing Pass Transistor Logic techniques), while improving logic level swing and static power characteristics and allowing simple top-down design by using small cell library. A simple change of the input configuration of the simple GDI cell corresponds to very different Boolean functions [6]. This GDIL undoubtedly reduces area as lesser number of LUTs and CLBs are used in FPGA prototyping [7].

GDIL viterbi decoder consists of three blocks. They are Branch Metric Unit (BMU), Add Compare Select Unit (ACSU) and Survivor Memory Unit (SMU). All these blocks are designed using GDIL technology, simulated and synthesized using Xilinx ISE.

4.1 GDIL based Branch Metric Unit(BMU)

The branch metric computation block compares the received code symbol with the expected code symbol and counts the number of differing bits. It consists of EXOR gate and counter. The Branch Metric Unit is designed using the EXOR gate and the 3-bit counter. The output of the EXOR gate is fed as the clock input to the 3-bit counter. 3-bit counter is designed by cascading the D FF and the output of the one flip flop is given as clock input for the next flip flop. Further the D input for all the flip flops are tied to HIGH input. The preset and clear input is used to make the counter working as asynchronous counter. The RTL schematic diagram of the Branch Metric Unit is shown in figure 2.

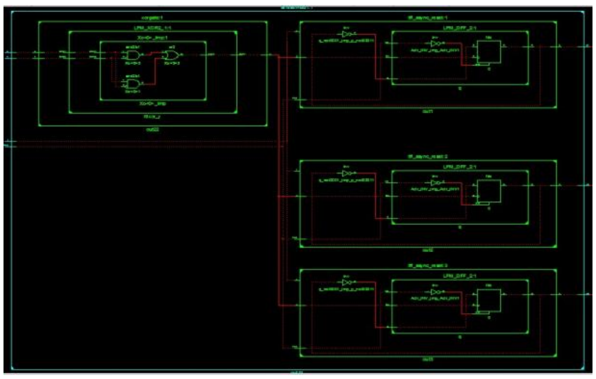

Fig 2: The RTL design of Branch Metric Unit

4.2 Add Compare Select Unit (ACSU)

The Add Compare Select Unit (ACSU) which adds the Branch Metrics (BM) to the corresponding Path Metrics (PM) compares the new PMs and then stores the selected PMs in the Path Metric Memory (PMM). At the same time, the ACSU stores the associated survivor path decisions in the Survivor Memory Unit (SMU).The PM of the survivor path of each state is updated and stored back into the PMM. The Block diagram of the Add Compare and Select unit is shown in the figure 3.

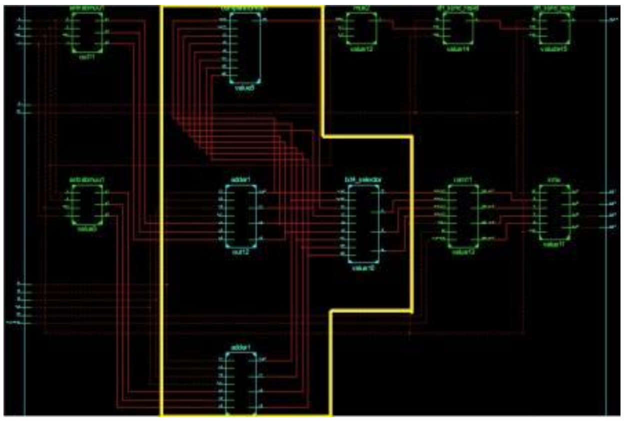

Fig 3: The RTL design of Add Compare Select Unit

State metric (SMi,j) and Branch Metric (BMi,j) are the two inputs to the Adder unit. Each butterfly wing is usually implemented by a module called ACS module. The two adders compute the partial path metric of each branch. The comparator compares the two partial metrics and the selector selects an appropriate branch. The new partial path metric updates the state metric of state p and the survivor path-recording block records the survivor path.

The adder unit which is proposed in the design consists of two full adders and one half-adder. Output of the Branch metric unit (BMU) is added with the previous path metric and the obtained output is the new path metric for the next branch. The input sequence from a0 (1100), b0 (1001), a1 (1011), b1 (1101), a2 (0111) and b2 (0011) are added to the adder unit. The 1st bit of a_0 and b_0, i.e. 11, is the input of half-adder and produces the output of the half adder operation sum1 as ‘0’ and carry as ‘1’.

4.3 Selector Unit (SU) using GDIL

The output of the comparator is given as the select signal for the multiplexer which is used to select the minimum path metric of the decoded message bit in the Viterbi decoder. The selector unit consist of four 2x1 MUX and the select signal for all the multiplexers are from the A<B output of the comparator. Hence the selector selects the minimum path metric value. The 4-bit input is a_0=1101, b_0=1111, a_1=1101, b_1=1000, a_2=1111, b_2=1110, a_3=1001, b_3=1110 and select line value is 1010. When the select line value is high i.e. ‘1’, the output value z_0 will be b_0, i.e. ‘1’. When the select line value is ‘0’, the output value z_0 will be a_0. Likewise all other input values z_1, z_2 and z_3 are taken and the outputs are obtained.

4.4 Survivor Memory Unit (SMU) using GDIL

The Survivor Memory Unit is designed by using the serial-in-serial-out shift register and the length of the shift register depends on the length of the convolution encoder. The figure 4 shows the 4x4 memory unit to store the minimum surviving path and the clock signal of the SMU are the ACSU output for a constraint length of 3.

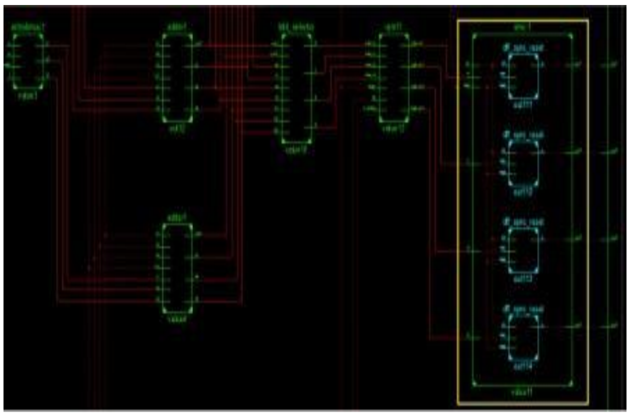

Fig 4: The RTL design of Survivor Memory Unit

The 4-bit output of the selector unit is input of the SMU. The SMU was designed as 4x4 shift register using D flip-flop. Each bit is stored in each of the D flip-flop. Similarly all the 4 shift registers store one bit each. The input of D flip-flop and clock is D flip-flop GND PULSE and CLK GND PULSE and the output for the q_1, q_2, q_3 and q_4 are shifting depending on the input value of D flip-flop and clock pulse. Number of memory stage depends on 2^{k-1} where k is the constraint length. In this GDIL method it varies from 4 to 128 stages.

4.5 Complete Integration of GDIL based Viterbi decoder

The viterbi decoder using GDIL technique is designed by integrating all the units like BMU, ACSU and SMU. The proposed design using GDIL is shown in the figure 5.

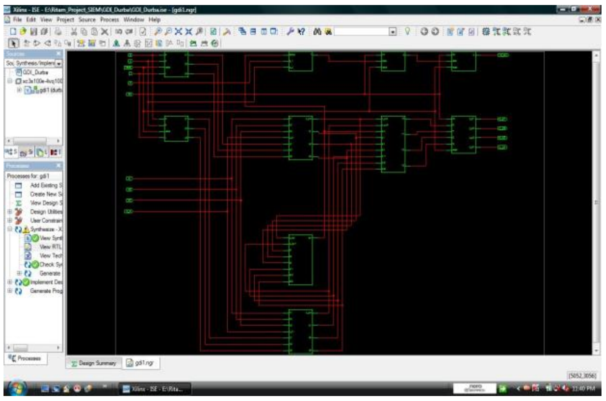

Fig 5: The RTL design of GDIL Viterbi Decoder

Here two Branch Metric Units are used since two possible changes from one state to another. This BMU calculates the branch metric between the expected sequence (original random input which is ‘a’) and the received sequence (introduced errors which is ‘b’). Then adder unit adds branch metric with the previous path metric and comparator compares the two paths and select the least path using selector. The survivor memory unit stores the path metric value and its corresponding states using the 2x1 multiplexer and 2 bit shift register to get the decoded output.

5. PROPOSED VITERBI USING DRAM

The GDIL viterbi is also compared with our proposed technique, which comprises a Survivor Path Unit (SPU) implements a trace back method with DRAM. This proposed approach of incorporating DRAM stores the path information in a manner which allows fast read access without requiring physical partitioning of the DRAM. This leads to a comprehensive gain in speed with low power effects.

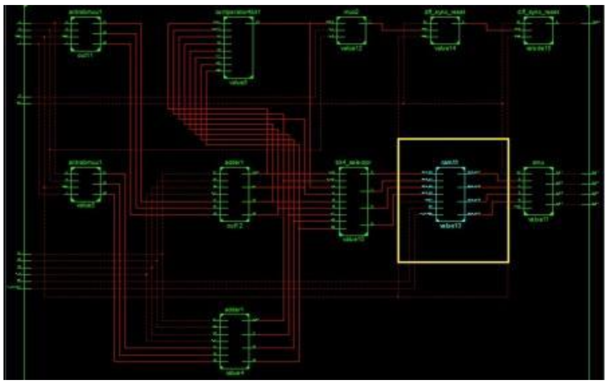

Fig 6: The RTL design of Proposed Viterbi using DRAM

As shown in figure 6 the proposed DRAM based GDIL viterbi is designed in Register Transfer Logic style. As in earlier inventions of viterbi, the main block is divided into two main sub-units, i.e; Add-Compare-Select (ACS) array and a Survivor Path Unit (SPU). Here in this proposed system the metric calculation, addition, weight comparison and survivor path selection everything take place in the ACS array (ACSU). Thus ACS array contains the weight values at each state, as a progression along the survivor path. The weight values are very necessary when comparing with other weights of other paths, to determine the survivor path. The compilation and comparison functions, which takes place within the ACS array actually determines the path extensions. Signals indicating these extensions to the survivor paths are passed from the ACS array to the SPU, which then updates the survivor paths.

Till now two methods exist for implementing the SPU: the register exchange method and the trace back method. Though the register exchange based survivor unit has better throughput when compared to trace back unit, but in this project by introducing the RAM cell between the ACS array and output register bank, a significant amount of reduction in path delay (almost 41%) has been observed.

The Trace Back (TBU) Unit operates in the following manner. At each time step, the chosen transitions are stored in a column. One state is chosen as a starting point and then trace back begins. Here in this proposed system the TBU consists of repeated reads from the RAM cell. Each read accesses a column which precedes the column last accessed in same clock cycle. This would cause very demanding requirements of GDIL viterbi using DRAM for high speed applications though it requires larger silicon chip area.

6. VITERBI IMPLEMENTATION ON RISC PROCESSOR

Finally the fastest GDIL based viterbi decoder is implemented on the 64-bit RISC (Reduced Instruction Set Computer) processor. RISC is basically a modern style of microprocessor with a few old tricks from the mainframe world. In todays market most microprocessors are based on either RISC or CISC (Complex Instruction Set Computer) architecture technologies. Research has shown that RISC architecture immensely boost up computer speed by using simplified machine instructions for frequently used functions. In this case the instruction set is etched into logic circuits using HDLs like Verilog HDL. This instruction set is reduced to basic, often used commands that can be executed in a single machine cycle. For general purpose computers the data collection shows that up to 85% is spent executing simple instructions like LOAD, STORE and BRANCH. For further complexity the compiler should generate several simple instructions. The more complex instructions consist of a very small amount of overall execution time of the CPU. Therefore for optimization of RISC processing power and to avoid complex instructions, no complex addressing is allowed and all instruction sets act on internal register set. These fruitful factors make RISC an irresistible choice and most modern processors are built on this form factor.

6.1 RISC Design Architecture

The ultimate goal of the project is to design and implement a RISC (Reduced Instruction Set Computer) using a FPGA (here Virtex 4 – Target device) for high speed application. The RISC is characterized by 64 bit architecture having 64 bit registers, ALU, RAM, Counters, Control Unit, Display Unit and most importantly Decoders. Here the proposed GDIL based Viterbi decoder is replaced by ordinary decoder, and then implemented and synthesized by Xilinx 12.4 ISE software.

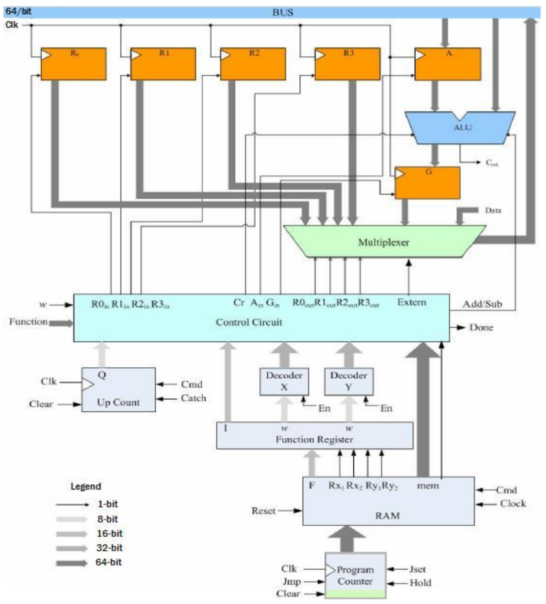

Fig 7: Processor Design Architecture

The following section would provide the background information of RISC architecture for best target processor. The figure 7 illustrates the processor's 64-bit architecture and the ability to READ and WRITE to the external memory. The designed processor core consists of Registers, ALU, RAM, GDIL Viterbi Decoders (Proposed), Counters, Control Unit and Display Unit connected by a central bus denoted by Bus wires. Control Unit (CU) provide necessary control signals that allow data to be moved or copied over the Bus wires as well as for performing an ALU operation on the data.

The RISC is designed using Verilog HDL. Machine instructions are directly implemented in hardware. For complex tasks to execute in a single cycle is performed by executing a series of basic instructions, either as in-line code or by calling a sub-routine.

6.2 Complete RTL Design of RISC with GDIL Implementation

Finally the target device of 64-bit RISC is designed in block level and synthesized. The control unit having orthodox decoders is replaced by proposed Viterbi using proposed GDIL technology.

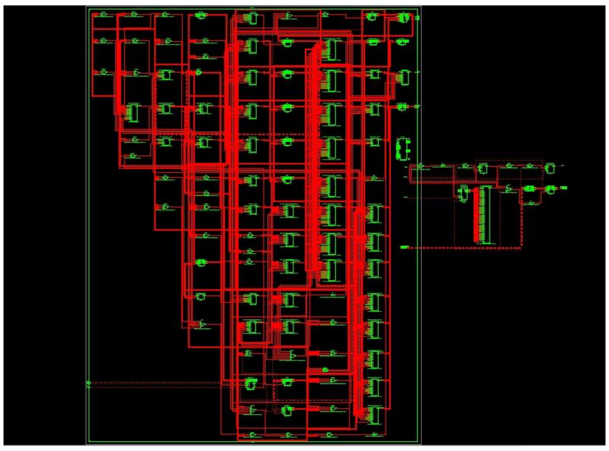

Fig 8: RTL Design of proposed RISC Architecture

The figure 8 shows the complete RTL design of the RISC processor with modifications of viterbi decoder for high speed applications. The backend coding is done using Verilog HDL and the layout is extracted by Xilinx ISE. Finally the proposed processor is successfully implemented in Virtex 4 FPGA.

7. SIMULATION RESULTS

The simulation results of orthodox viterbi, GDIL viterbi and proposed GDIL viterbi using DRAM are shown below. The simulation is mainly performed by Xilinx ISE 12.4 and synthesized in the FPGA [8, 9] target device: Virtex 4. The major constrains for VLSI design is speed, power and area. In this whole project all three constrains have been taken into account. The detail timing report and power report describes the delay analysis for high speed applications and low power for proposed Virtex [10] based viterbi decoder.

7.1 Timing Analysis

Table 1. Performance Summary (Timing)

Timing Report	Orthodox Viterbi	GDIL Viterbi	Proposed GDIL Viterbi with DRAM
Target Device	XA9536XL (Virtex 4)	XA9536XL (Virtex 4)	XA9536XL (Virtex 4)
Max Clock Frequency	64.516 MHz	64.516 MHz	64.516 MHz
Min Clock Period	21.400 ns	15.500 ns	15.500 ns
Clock to Set up Cycle Time	21.400 ns	15.500 ns	15.500 ns
Set up to Clock Pad Delay	87.663 ns	52.600 ns	20.800 ns
Clock Pad to Output Pad Delay	16.635 ns	5.800 ns	5.800 ns

7.2 Power Analysis

Table 2. Performance Summary (Power)

Power Report	Orthodox Viterbi	GDIL Viterbi	Proposed GDIL Viterbi with DRAM
Target Device	XA9536XL (Virtex 4)	XA9536XL (Virtex 4)	XA9536XL (Virtex 4)
On-Chip IO Pads	0.982 µw	0.001 µw	0.001 µw
Leakage Power	3.791 µw	1.010 µw	1.007 µw
Quiescent Power	2.315 µw	1.110 µw	1.007 µw
Dynamic Power	0.739 µw	0.022 µw	0.003 µw

8. SYNTHESIS RESULTS OF VITERBI DECODED 64-BIT RISC

The developed proposed viterbi decoded RISC architecture is simulated and verified its functionality. Once the functional verification is done, the RTL model [11] is taken to the synthesis process using the Xilinx ISE 12.4. In synthesis process, the RTL model will be converted to the gate level netlist mapped to a specific technology library [12]. This modified viterbi decoded 64-bit RISC design is implemented on FPGA (Field Programmable Gate Array) family of Virtex 4. Here in this Virtex 4 family many different devices were available in the Xilinx ISE tool. In order to implement this modified viterbi with DRAM, the device named as "XA9536XL" has been chosen and the package as "FG320" with the device speed as "– 4 ". The design of modified viterbi decoded RISC for low power [13] and high speed [14] is synthesized successfully and its results are analyzed as shown in the figure 9 below.

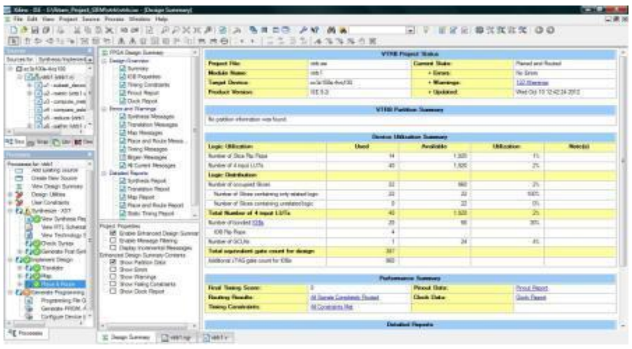

Fig 9: Synthesize Report of Proposed Viterbi Decoded RISC

After completion of synthesize the entire circuit model is processed through Translate, Map, Place and Route successfully. Finally a UCF file of the target object is created which is prototyped with hardware FPGA Virtex 4 hardware [15], through parallel connection using JTAG. Boundary Scan is performed followed by generation of PROM file. Finally the modified viterbi with RAM cell is successfully configured into FPGA Virtex 4 shown in figure 10.

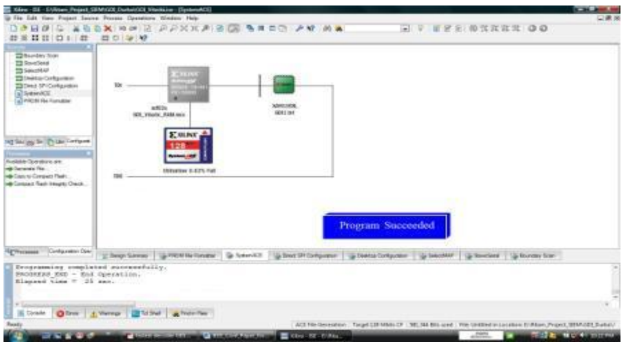

Figure 10: FPGA prototyping of Proposed Viterbi decoded 64-Bit RISC

The layout of the proposed modified GDIL Viterbi decoded RISC is obtained shown in figure 11.

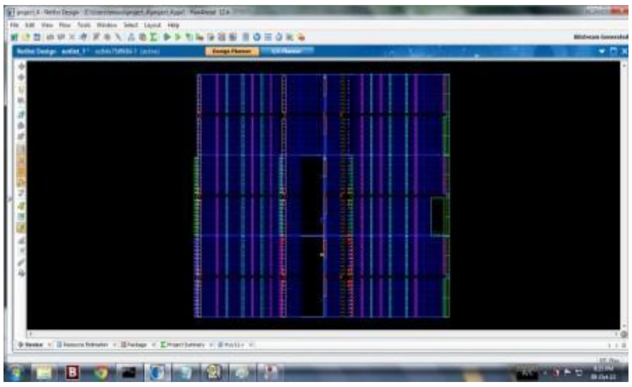

Figure 11: Layout of proposed GDIL Viterbi decoded RISC

The graphical analysis is also performed for all three viterbi decoders to have a complete performance summary. The timing - delay analysis and power analysis are shown in figure 12 (a) and 12 (b) respectively.

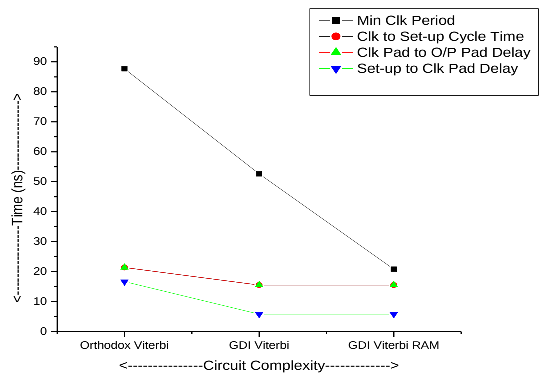

Figure 12(a): Timing & Delay (Graphical) Analysis

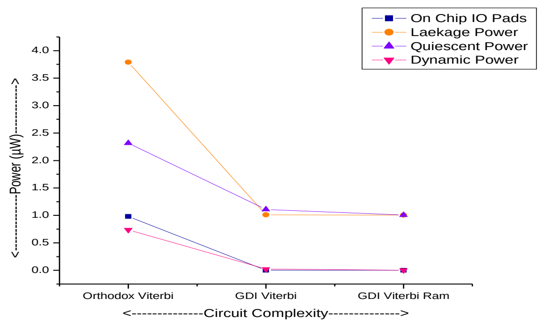

Figure 12(b): Power (Graphical) Analysis

9. CONCLUSION & FUTURE SCOPE

The Add-Compare-Select (ACS) and Trace Back (TB) units and its sub circuits of all the three type decoder(s) have been operated in deep pipelined manner to achieve high transmission rate. Although the register exchange based survivor unit has better throughput when compared to trace back unit, but in this paper by introducing the RAM cell between the ACS array and output register bank, a significant amount of reduction in path delay has been observed. All the designing of viterbi is done using Xilinx ISE 12.4 and synthesized successfully in the FPGA Virtex 4 target device operated at 64.516 MHz clock frequency and reduces almost 41% of total path delay with considerable low power results. The fast decoding can be achieved by our modified viterbi approach, which shows significant results in high speed applications. In future this research work can be extended by 32nm CMOS process technology for more precise fast decoding results.

10. REFERENCES

[1] Arkadiy Morgenshtein, M Moreinis and R Ginosar, 2004. "Asynchronous Gate-Diffusion-Input (GDI) Circuits" IEEE Transactions on VLSI Systems, 12, pp.847-856.

[2] El-Dib D. A. and M. I. Elmasry.2004, "Modified register-exchange Viterbi decoder for low power wireless communications", IEEE Trans. Circuits Syst. I, Reg. 51, pp. 371–378.

[3] Song li and qing-ming yi., 2006. 'The Design of High-Speed and Low Power Consumption Bidirectional Viterbi Decoder". Fifth International Conference on Machine Learning and Cybernetics. pp. 3886-3890.

[4] Mohammad K.Akbari and Ali Jahanian, 2004, "Area efficient, Low Power and Robust design for Add Compare and Select Units," Proceedings of the IEEE Conferecne on EUROMICRO Systems on Digital System Design (DSD '04).

[5] Arkadiy Morgenshtein, Fish, and I. A. Wagner 2001, "Gate-diffusion input (GDI) – A novel power efficient method for digital circuits: A detailed methodology" in Proc. 14th IEEE Int. ASIC/SOC Conf., pp. 39–43.

[6] Arkadiy Morgenshtein, M Moreinis and R Ginosar, 2004. "Asynchronous Gate-Diffusion-Input (GDI) Circuits" IEEE Transactions on VLSI Systems, 12, pp.847-856.

[7] F. Chan and D. Haccoun. Adaptive Viterbi Decoding of Convolutional Codes over Memoryless Channels. IEEE Transactions on Communications, 45(11):1389–1400, Nov. 1997.

[8] http://www.xilinx.com/products/silicondevices/fpga/virtex-6

[9] Denton J. Daily (2004), "Programming Logic Fundamentals using Xilinx ISE and CPLDs," in Prentice Hall, 203 pages.

[10] www.xilinx.com/training/languages/designing-with-vhdl.htm

[11] Chu C.-Y., Y.-C. Huang and A.Y. Wu, 2008, "Power Efficient Low Latency Survivor Memory Architecture for Viterbi Decoder". IEEE International Symposium on VLSI Design Automation, and Test, pp. 228-231.

[12] www.digilentinc.com/Products/Detail.cfm?NavPath=ATLYS

[13] Man Guo, M. Omair Ahmad, M.N.S. Swamy, and Chunyan Wang , "A Low-Power Systolic Array-Based Adaptive Viterbi Decoder and its FPGA Implementation", International Symposium on Field-Programmable Technology 2003, Vol 2, Page(s)- 276 - 279, 25-28 May 2003.

[14] Abdulfattah M. Obeid, Alberto Garcia, Mihail Petrov, Manfred Glesner ,"A Multi – path high speed Viterbi decoder" , Proceedings of the 2003 10th IEEE International Conference on Electronics, Circuits and Systems, 2003. ICECS 2003, Vol 3, Issue, 14-17 Page(s): 1160 – 1163, December 2003.

[15] www.xilinx.com/products/boards/v6conn/reference_designs.html