An Efficient FPGA Based HDB3 Decoding System Using Direct Digital Synthesis

Imran Ali and Ali Ahmed

Abstract—**The proposed design of HDB3 decoding system using FPGA implementation offers an efficient and unfailing decoding at receiving end by sustaining clock data recovery using Direct Digital Synthesis (DDS). The system captures E1/T1 HDB3 encoded tertiary level stream at input, converts it into binary level symbols, FPGA IO reconcilable, and decode and transforms it into synchronized NRZ output. The receiver end of FPGA based HDB3 decoding system has never been implemented using clock data recovery. The resource efficient implementation and synthesis outcome illustrate that the design of HDB3 decoder is very simple and fault tolerant and its ASIC design can easily be surrogate by the proposed idea.**

Index Terms—**AMI, clock data recovery, decoder, direct digital synthesis, E1, FPGA, HDB3, NRZ, T1**

I. Introduction

The voice over digital network uses different line encoding techniques. The HDB3 (High Density Bipolar-3) coding, a bipolar signaling technique is the refined version of AMI (Alternate Mark Inversion) and relies on the transmission of both the positive and negative pulses. The clock extraction from the data makes this coding technique distinguished among others. A number of various encoding methods are recommended [1] amongst which high density bipolar-3 is frequently used. The PCM Telephone Systems (E1/T1) over long distance commonly use line coding which helps resolve DC off set and synchronization challenges between two nodes. To decode an HDB3 encoded data stream, an efficient decoder is required which provide synchronized NRZ data with clock.

In the past, various designs of HDB3 decoder have been implemented. In [2], Wei Dang uses converters, shift registers, mono-stable trigger, multiplier and other combinational logic, making the design complex. It explains only simulation and does not explore the resource utilization and implementation results. M. V. Bhimrao and Ramesh T has implemented the HDB3 decoder as combination of violation, balance and polarity modules that outputs data only [3]. In [4]-[5], only NRZ data is recovered from two symbol HDB3 stream at simulation level only. The decoder design of [6] is implemented using VHDL for Altera device without clock extraction from decoded NRZ output and is relatively complex. The HDB3 decoder in previously implemented FPGA designs [2]-[6] and ASIC solutions never used clock data recovery technique at its output for synchronized data and more over ASIC realization have complex structure and long algorithmic delays. To cope up with these problems, a complete system which decodes the stream into NRZ format with clock data recovery at its output end is proposed in this design. The system uses distinctive approach of implementing this decoder in FPGA. Clock data recovery of stream uses the technique called Direct Digital Synthesis provided by the Xilinx® [7].

The system accepts standard HDB3 encoded E1 input from stream generator/medium and directs it to a voltage level translator circuit for converting the tertiary input to two symbol signals and to make it FPGA IO compliance. The proposed FPGA based HDB3 decoder interpret the data with its simple novice and unique algorithm which is then passed to the Clock Data Recovery module in FPGA for recovering NRZ data with clock. The synchronized clock along with decoded data is the final output of system and is the key characteristic of this proposed design. The overall system block diagram is depicted in Fig.1.

The HDB3 decoder is integrated with the clock data recovery of Xilinx reference design [2]. The FPGA implementation and simulation results reveal the simplicity, uniqueness and fault tolerance of the proposed design.

This paper is organized in the hierarchal order of system. Section II presents comparator based voltage level translation for FPGA. Section III reviews HDB3 Protocol. Section IV provides complete proposed HDB3 decoder design and architecture. Section V is dedicated to clock data recovery module which uses digital PLL for synchronization of data and clock. In section VI the simulation and FPGA implementation results are reported. The section VII covers the conclusion.

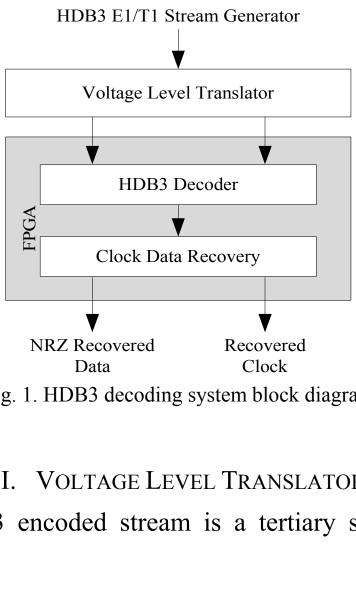


Fig. 1. HDB3 decoding system block diagram

II. Voltage Level Translator

The HDB3 encoded stream is a tertiary signal on one

Manuscript received November 29, 2012; revised February 5, 2013.

Imran Ali is with the Department of Electrical and Electronics Engineering, University of Engineering and Technology Taxila, Pakistan (e-mail: Engr.ImranAliMirza@ gmail.com).

Ali Ahmed is with the Department of Electronics and Communication, Hanyang University, South Korea (e-mail: aliahmedansari@gmail.com).

coaxial or symmetrical pair (HDB3+, HDB3-) having positive, negative and zero levels. Also, the maximum peak voltages are 2.37V and 3.0V for 75 Ohm and 120 Ohm load respectively [1]. Since, the voltage levels of HDB3 encoded signal are +V, 0V and –V, therefore, it is hazardous to interconnect the HDB3 input stream directly to FPGA. An interface is crucial for converting input data voltage levels to any of the FPGA IO Standard and to decode tertiary logic to two binary signals. For this propose, Voltage Level Translator Module is designed which successfully provides the necessary conversions before going into FPGA.

A. Voltage Translation Circuit Design

A simple circuit is designed with coupling transformer, two comparators and associated biasing components, e.g. capacitors and resistors for translating HDB3 tertiary logic to FPGA readable binary levels. The transformer provides electromagnetic inductive coupling in conjunction with galvanic isolation between HBD3 source and the conversion circuit, and plays an important role in filtering the power waveform. Due to high voltage transients, it saturates and acts as limiter to protect the circuit. It also equalizes the terminating impedances on both sides for maximum power transfer [8]. Due to low cost, a general purpose operational amplifier (Op-Amp) is utilized as a comparator [9]. The simplified block diagram of this comparator circuit is depicted in Fig. 2.

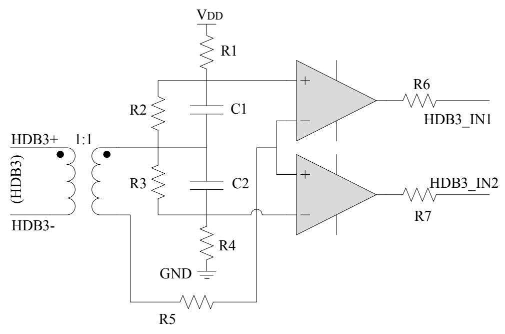


Fig. 2. Voltage translation circuit

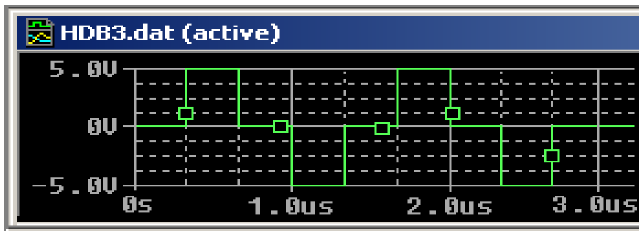
For the translation of HDB3 encoded data into the FPGA tolerant voltage levels, two back to back connected comparators sense the +V and -V pulses with the reference voltage 0V. This circuit takes *HDB3* encoded data and generates two outputs, named as *HDB3_IN1* and *HDB3_IN2*.

B. Working Principle

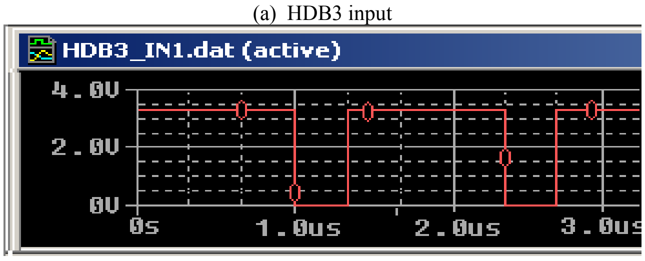
The unique combinations of outputs are generated as the HDB3 encoded signal appears at the primary side of transformer. Whenever, input of the designed circuit is +V, the outputs *HDB3_IN1* and *HDB3_IN2* are +3.3V (high) and 0V (low) respectively. The output *HDB3_IN1* is low while the output *HDB3_IN2* is high for -V input. Both the outputs are high for 0V input and can never be low at a time as there is no other input option.

C. PSpice® Simulation Analysis

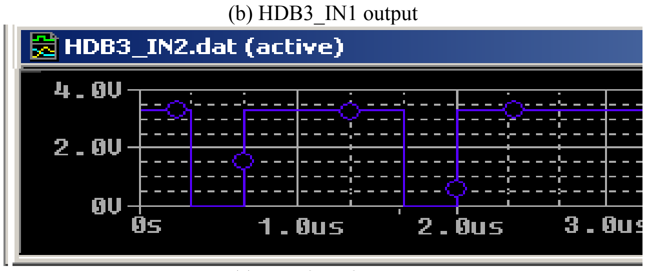
The comparator circuit is simulated for circumstantiate of its design using PSpice® Tool, a Windows based analog circuit and digital logic simulation program used for electronics circuit design and analysis [10]. The simulations results are analyzed to ensure the accuracy of the designed circuit as shown in Fig. 3. The tertiary input *HDB3* stream is depicted in Fig. 3(a) while the binary levels *HDB3_IN1* and *HDB3_IN2* outputs are presented in Fig. 3(b) and Fig. 3(c) respectively.



(a) HDB3 input



(b) HDB3_IN1 output



(c) HDB3_IN2 output

Fig. 3. Voltage comparator circuit pspice® simulation results

III. HDB3 Coding Protocol

HDB3 implies High Density Bipolarity with not more than three serial zeros to be transmitted consecutively. For encoding perspective, a special substitution code (0001 or 1001) has to be transmitted instead of four successive 0 bits. On the other side, the receiver/decoder has to differentiate between data bit and substitution code bit. The code violation could be introduced in substitution code, the code violation, as the name suggests to not following the protocol of sending a bit with alternate polarities (+V and –V) . For detection of special substitution code, the receiver has to replace four ongoing zeros at time of decoding for maintaining the integrity of the protocol. The detection of special substitution code follows two protocols at receiving/decoding end [11]. First, if the last bit of the encoded data and the last bit of the last encoded substitution code are of unchanged polarity, then B00V, substitution code is chosen. Here, B represents the bipolar 1 bit, and V represents the violating 1 bit. Secondly, if the last encoded bit of the data and the last bit of the previous encoded substitution code are of unlike polarities, then 000V substitution code is preferred.

IV. HDB3 Decoder Design

The decoder's working technique follows HDB3 encoding standard. There is a probability of six likely special patterns erratically transmitted contingent on formerly encoded data; these are P000P, N000N, PN00N, NP00P, P00PN00N and N00NP00P. Here, P and N, respectively typify the positive

(+V) and negative polarities (-V). The decoder is designed to detect, recognize and decode the special codes. The rules, created during decoder design are as follows:

1) Whenever, one of the special codes P000P, N000N, PN00N and NP00P is present in data, four zeros are transmitted.
2) For special code P00PN00N or N00NP00P, one 0 is transmitted.
3) Otherwise incoming data bit is transmitted.

The architecture of proposed HDB3 decoder is characterized at register transfer level (RTL) and contains a data path (DP) and a control unit (CU), depicted in Fig. 4, as required by the design specifications and constraints. With the emergence of the data path controller architecture, the constraints get effectively captured and the mapping of behavioral description to hardware results into high performance and space efficient machine [12].

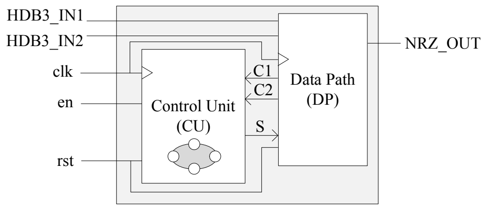


Fig. 4. HDB3 decoder datapath and control unit

Data inputs *HDB3_IN1* and *HDB3_IN2*, coming from voltage translation circuit, and data output *NRZ_OUT* going to CDR are connected to the data path unit, and the primary control input signal *en*, for enabling the decoder process is connected to control unit as clear from Fig. 4. There are also a control signal *S* from the CU to the DP and two status signals *C1* and *C2* from DP to CU. Every sequential module in the design incorporates an active high synchronous reset signal *rst* which is primary input for CU state initialization and DP operation control. Also, the clock *clk* is the primary input to both CU and DP.

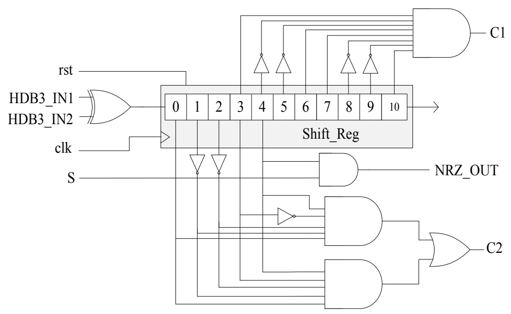


Fig. 5. Datapath design architecture

A. Datapath Unit Design

In general, the datapath unit includes arithmetic units, such as arithmetic and logic units (ALUs), adders, multipliers, shifters and digital signal processors. The datapath unit consists of computational resources e.g. ALUs and storage registers, logic for moving data through the system between the computation units and the internal registers, and to and from the external environment [13], [14]. However, the datapath design of HDB3 decoder is very simple and contains only a single 11-bit shift register *Shift_Reg* and a few gates as clear from Fig. 5. On power up or reset state, the shift register is initialized with zeros. With each clock cycle, the XORed value of *HDB3_IN1* and *HDB3_IN2* occupies the most significant bit of *Shift_Reg* and its remaining 10 bits are shifted right with flushing out its least significant bit. For each clock cycle, two conditions *C1* and *C2* are computed to decide the presence of special code substitution in the input encoded stream. The signal *S*, on basis of conditions *C1* and *C2* decides to out zero or data bit at *Shift_Reg[4]*. The signal *C1* becomes true when a stream of zeros is encoded in data while the presence of special substitution in encoded data asserts the signal *C2*.

B. Control Unit Design

Generally, the control unit orchestrates, coordinates and synchronizes the operations of datapath unit. The control unit of a machine generates the signals that load, read and shift the contents of storage registers; fetch instructions and data from memory; store data in memory; steer signals through multiplexers; control tri-state devices; controls the operations of ALUs and other complex datapath units [13], [15]. The design of proposed HDB3 Decoder control unit is very simple and is based on the synchronous Finite State Machine (FSM) model. Since, Moore and Mealy are the two fundamental types of finite state machines [13], Moore state machine is deliberately chosen for the CU design with output *NRZ_OUT* as the functions of current state only. The FSM can be delineated and designed systematically with the succor of Algorithmic State Machine (ASM) chart [16]. The ASM of CU, organized as set of seven states, termed as IDLE, TX0, TX1, TX01, TX02, TX03 and TX04 after the nature of operation, is rendered in Fig. 6. For simplicity, the en control signal is not included in ASM chart. As machine starts, it waits for de-assertion of *rst* and assertion of *en* controls. For a stream of encoded zeros, *C1* asserts high and machine remains in TX0 state while keeping *S* low. When there is 000V or B00V substitution in encoded data, *C1* is low and *C2* is high, resulting *S* low for successive four clock cycles.

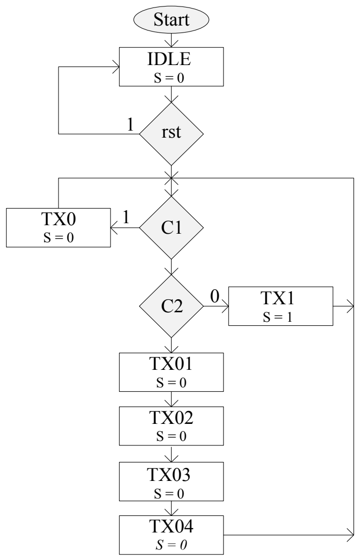


Fig. 6. Simple REPRESENTATION of decoder

In both two cases, the CU pushes the DP to generate zeros on *NRZ_OUTPUT*. For absence of more than three zeros in encoded input data, machine remains in TX1 state keeping *S* high and datapath XORs the *HDB3_IN1* and *HDB_IN2* to give *NRZ_OUT*.

The HDB3 Decoder module buffers input data in *HDB3_REG1* and *HDB3_REG2* registers. To discriminate between the data and special code novice approach is expressed. If the data is special code then *S* signal is 0 otherwise it is 1. The decoder idea and its implementation are uncomplicated as clear from its design. The output of HDB3 decoder is then passed to clock data recovery circuit that synchronizes the data with the clock for output. This integration has not been done in any other previously HDB3 decoder designs [2]-[6].

V. Clock Data Recovery

The Clock Data Recovery (CDR) hardware module is the final stage of the purposed HDB3 decoder design. It is implemented on FPGA using Direct Digital Synthesis (DDS) using Xilinx reference design [7]. The CDR is essential for recovering clock and NRZ data. This module is a digital counterpart of Phase Locked Loop (PLL), acquire following blocks as shown in CDR block diagram Fig. 7.

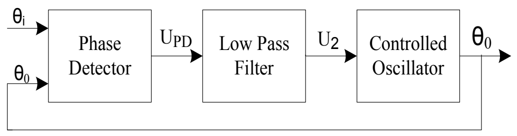


Fig. 7. CDR pll block diagram

The CDR's input ports require 150 MHz clock along with the data which is intended to be recovered. HDB3 decoder output is interfaced with CDR for synchronizing the NRZ Data at 2.048 MHz .This data is carried out through the desired PLL blocks as shown in Fig. 7. Finally, the NRZ Recovered Data along with the clock is properly synchronized. The output of phase detector is proportional to the difference between input and feedback phase coming from controlled oscillator. Equation (1) depicts the phase detector output.

$$v_{pd} = K_d(v_i - v_o) \tag{1}$$

Here, K_d is gain factor measured in radians per second. Since signal phase error from phase detector has to be low pass filtered so a stage of digital low pass filter is necessary. The transfer function of low pass filter is presented in (2).

$$F(s) = G1(G_2 + \frac{K_f.G_1}{s}) \tag{2}$$

The adder and 32 bit accumulator intermix to form a digital VCO. The digital VCO outputs deviation of frequency $\Delta\omega$ is proportional to its input controlling signal v_2. Equation (3) relates algebraically:

$$\Delta\omega = K_v v_2 \tag{3}$$

Here K_v is the gain constant of Controlled oscillator. The controlled oscillator output is the Laplace transform of signal phase as shown in (4) and (5).

$$\frac{d\theta_0}{dt} = K_v v_2 \tag{4}$$

$$L\left[\frac{d\theta_0}{dt}\right] = s\theta_0(s) = K_v v_2(s) \tag{5}$$

The relationship between control and phase is depicted in (6).

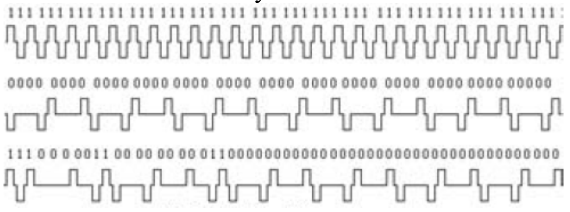
$$\theta_0(s) = \frac{K_v v_2}{s} \tag{6}$$

The central frequency f_c depends on low frequency output f_w and reference clock frequency f_{clk}. The central frequency is computed by (7).

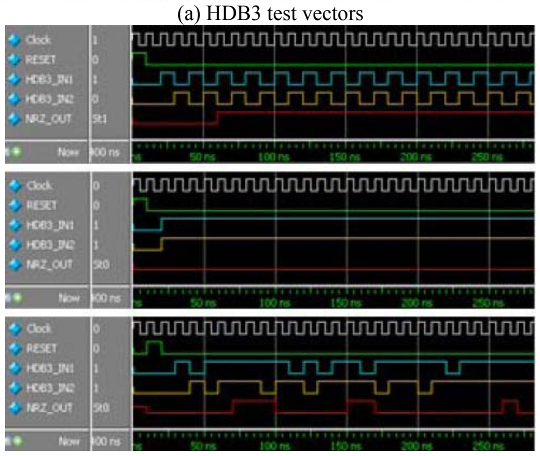
$$f_c = f_w \frac{2^{32}}{f_{clk}} \tag{7}$$

VI. Simulation and Implementation Results

After proposed protocol design, it is translated in Verilog HDL, standardized as IEEE 1364 and Mentor Graphics® simulator ModelSim® SE 6.5 is incorporated for debugging, analysis and simulation propose. The Xilinx® ISE 10.1 is used for implementation and hardware synthesis of simulated design to generate final bit stream for the target FPGA device. Both the simulation and implementation of HDB3 decoder uses test vectors with special codes. In this design an E1 stream as an HDB3 encoded data is utilized and the system is translating the data to NRZ format at 2.048 MHz clock cycles. The simulation of decoder algorithm produces the exact results of encoded data which is then implemented in Xilinx Vertex 5 FPGA successfully.



(a) HDB3 test vectors



(b) Simulation results

Fig. 8. HDB3 Test Vectors and Simulation Results

A. Simulation Analysis

The "HDB3 Decoder" hardware design is simulated independently to corroborate the purposed algorithm using the ModelSim®. For each HDB3 test vectors, *HDB3_IN1* and *HDB3_IN2* are derived first for test bench design. The *NRZ_OUT* is the decoded output of the hardware design in NRZ format. This output is always valid after four clock cycles. The design is verified with multiple test input patterns for long period. Some of the selected patterns for reference includes HDB3 encoded streams of all 1's, all zeros and random data, depicted in Fig. 8 (a). The decoded results in binary format corresponding to the input vectors are explained in Fig. 8 (b). It is clear that the original patterns are perfectly decoded for different inputs.

B. Implementation Analysis

The design is implemented using Xilinx® ISE 10.1. The implementation results confirm that the design with CDR can run at maximum clock frequency of 345.542 MHz and 2.894 ns clock period. The design takes 18 slice LUTs only. The system performance is absolute and no decadency is found in the design. The system is analyzed for different test vectors. Here some of them are presented Fig. 9(a, b and c).

The data output from HDB3 Decoder module implemented on Xilinx® FPGA contains the valid NRZ Data of all the input HDB3 encoded test vectors as expressed in the Fig. 9. The recovered clock and data from the CDR also analyzed for their synchronization. The Fig. 9(d) shows the final NRZ output along with the recovered clock from an FPGA based efficient HDB3 decoding system.

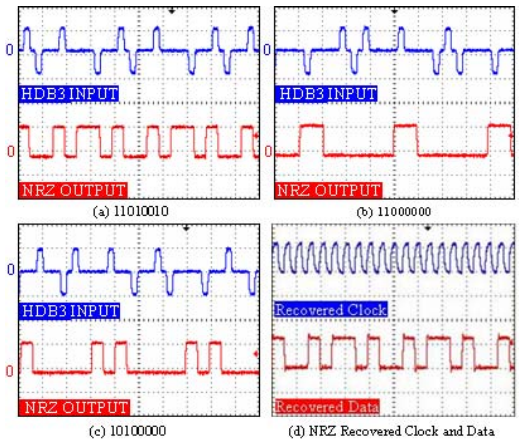


Fig. 9. Implementation results

VII. Conclusion

In a nut shell, system proposed is an efficient and uncomplicated one, the integration of clock data recovery using DDS module in HDB3 decoder is an exclusive technique that render the results that has never been apply so far. The scheme of dissecting substitution codes is novel, tricky and robust. It curtails the system delays while perceiving and imparting the data. From simplicity point of view it outperforms the ASIC realization of HDB3 codec and other FPGA based decoders. This efficient way is crucial for deploying hardware that entail synchronized clock with NRZ data. The precision and unfussiness of our proposed design offer the 0% error in simulation and hardware implementation results. The design reusability property of FPGA offers expediency for functional expansion and maintenance. It perks up the integration and reliability.

In future, this design can be tailored for any variety of G.703 encoding standards besides HDB3. The encoder can be amalgamated in this design to provide a comprehensive codec with the attribute of CDR. Even though, in this design the error adjustments are not anticipated, but for safe transmission an error rectification codes can also be used to shield the transmission error.

References

[1] ITU-T Recommendation G.703 "Physical/Electrical Characteristics of Hierarchical Digital Interfaces," November 2001.

[2] Wei Dang, "Implement of HDB3 Coder in Base-band System, " in *Proc. 2nd International Conference on Consumer Electronics, Communications and Networks (CECNet)*, pp. 3273-3274, 21-23 April 2012.

[3] M. V. Bhimrao and T. Ramesh, "ASIC Implementation of HDB3 Codec," *International Journal of Innovative Technology and Exploring Engineering (IJITEE)*, vol. 1, Issue 3, August 2012.

[4] C. S. Zhang and Q. Xu, "A design of HDB3 CODEC Based on FPGA," in *Proc. 2nd International Conference on Advanced Computer Control (ICACC)*, vol. 3, pp.75-78, 27-29 March, 2010.

[5] Y. Zhang, X. Wang, and Y. Wang, "A new design of HDB3 encoder and decoder based on FPGA," in *Proc. 9th International Conference on Hybrid Intelligent Systems*, vol.1, pp. 210-213, 12-14 August, 2009.

[6] S. Z. Wang and T. Wang, "A way to implement the HDB3 encoding-decoding based on VHDL language," *Journal of Inner Mongolia Normal University*, March 2006.

[7] P. Novellini and G. Guasti, "Clock data recovery design techniques for E1/T1 based on direct digital synthesis," *Xilinx Application Note XA*, pp. 868, vol. 1, January 29, 2008.

[8] P. A. Janse, V. Rensburg and H. C. Ferreira, "Design of a bidirectional impedance-adapting transformer coupling circuit for low-voltage power-line communications," in *Proc. IEEE Transactions on Power Delivery*, vol. 20, no.1, pp. 64- 70, Jan 2005.

[9] R. Moghimi, "Ask the application engineer-31," *Analogue Dialogue 3*, April 7, 2003.

[10] I. Chamas and M. A. E. Nokali, "Automated PSpice simulation as an effective design tool in teaching power electronics," *IEEE Transactions on Education*, vol. 47, no. 3, pp. 415- 421, August 2004.

[11] D. Oner, "Criteria for choosing line codes in data communications," *Journal of Electrical and Electrinics Engineering*, Istanbul University, vol. 3, pp. 843-857, 2003.

[12] R. K. Kamat, P. K. Gaikwad, and S. A. Shinde, "Implementation of FPGA based firewall using behavioral synthesis," *IJCSNS International Journal of Computer Science and Network Security*, vol. 10, no. 6, June 2010.

[13] M. Ciletti, *Advanced Digital Design with the Verilog HDL*, Prentice-Hall, 2002.

[14] S. A. Khan, *Digital Design of Signal Processing Systems: A Practical Approach*, 1st Edition, John Wiley & Sons, 2011.

[15] P. D. Minns and I. D. Elliott, *FSM based Digital Design using Verilog HDL*, John Wiley & Sons, April 2008.

[16] Jr. C. H. Roth , *Fundamentals of Logic Design*, Sixth Edition, 2009.

Imran Ali received his B.E degree in electrical engineering from University of Engineering and Technology Taxila Pakistan in 2008. He has also completed his M.E in Electronics Engineering from the same university. He is working as a research engineer in a private sector organization after bachelor degree with a working experience of about four years. His major areas of research include Advanced Digital Design, FPGA, Processor and Controller based Embedded Design and Digital Signal Processing. He is also working in domain of Cryptography, High Speed and Impedance Controlled Multilayer Board Design, System Testing and Verification, Single to Multilayer PCB Design and Application Development. He is committed deliberately to pursue for PhD from one of the reputed international university in *Advanced Digital Design of Signal Processing Systems* for the state of the art quality research.

Ali Ahmed received his B.Sc. degree in electronics engineering from the NED University of Engineering and Technology, Karachi, Pakistan in 2008. He is currently working toward the MS./Ph.D. degree in electronic engineering in the Department of Electronics and Communication Engineering, Hanyang University, South Korea. His current research includes Power analysis of SRAM-based architecture for ternary content addressable memory. He received the MS./Ph.D. Engineering Scholarship from the Higher Education Commission (HEC), Government of Pakistan under the HEC project titled HRDI/UESTP Scholarship scheme.