

See discussions, stats, and author profiles for this publication at: <https://www.researchgate.net/publication/220338037>

# Design of a Multicode Bi-phase Encoder for Data Transmission.

Article in *Journal of Circuits Systems and Computers* · February 2006

DOI: 10.1142/S0218126606002897 · Source: DBLP

CITATIONS

0

READS

3,194

2 authors:



Raida Al Alawi  
Ahlia University

15 PUBLICATIONS 99 CITATIONS

[SEE PROFILE](#)



A-Imam Al-Sammak  
Ahlia University

22 PUBLICATIONS 90 CITATIONS

[SEE PROFILE](#)

# DESIGN OF A MULTICODE BI-PHASE ENCODER FOR DATA TRANSMISSION

RAIDA AL-ALAWI

Department of Computer Engineering  
College of Information Technology, University of Bahrain,  
P.O. Box 32038, Isa Town, Bahrain.

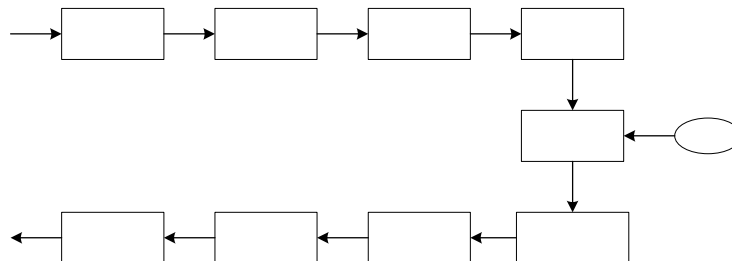
A. J. AL-SAMMAK

Department of Electrical and Electronics Engineering,  
College of Engineering, University of Bahrain,  
P.O. Box 32038, Isa Town, Bahrain.

In this paper, we present a versatile Multicode Bi-Phase Encoder (MBPE) circuit capable of encoding five different Bi-Phase line codes, namely: Bi-Phase-Level (Bi- $\Phi$ -L), Bi-Phase-Mark (Bi- $\Phi$ -M), Bi-Phase-Space (Bi- $\Phi$ -S), Differential Manchester (DM) and Inverse Differential Manchester (IDM) codes. The design methodology is based on a new definition of these codes in terms of encoding rules and state diagrams, instead of the traditional way of representing them in terms of their bit transition. The operation mode of the MBPE is set by three selection lines, which can be either hardware or software controlled. This will facilitate the process of altering the data transmission protocol without the need of changing the encoder hardware. The functionality and design of the MBPE is outlined. VHDL has been used to describe the behavior of the MBPE whose operation was verified using the *ModelSim XE II* Simulation tools. Implementation and testing of the MBPE on XILINX Spartan-II FPGA showed that the MBPE circuit is capable of encoding NRZ data into any of the five codes.

## 1. Introduction:

Advances in data transmission and various digital communications network architectures have resulted in an increased demand to multimedia applications where data, voice, images and video signals are to be transmitted. These networks are becoming very large, and often their nodes have several different connections between them utilizing more than one protocol. Different protocols normally employ different line (modulation) codes, and hence, it is important to develop a controller in the physical layer that can cope with multiple protocols. It is common nowadays to have off-the-shelf multi-protocol communication boards operating several line codes in order to minimize parts swapping [1]. A typical digital transmission system is shown in figure 1. Source encoding reduces the average number of data bits for a given message to the minimum possible value while channel coding is used for error control. Line coding is used to transform blocks or sequences of symbols into physical signals well matched to the characteristics of the physical channel of the system. After passage through the channel, the decoding processes are in the reverse order. This letter is concerned with the line encoder.



Several line coding techniques suitable for data transmission have been devised. Among these codes are Bi-Phase-Level (Bi- $\Phi$ -L), Bi-Phase-Mark (Bi- $\Phi$ -M), Bi-Phase-Space (Bi- $\Phi$ -S), Differential Manchester (DM) and Inverse Differential Manchester (IDM) codes. These widely used codes belong to the Bi-Phase code family which is characterized by having at least one transition per bit time. They share many desirable properties such as DC free component, self clocking and error detection capability. DC free prevents the transmission line from being saturated, while self clocking is an attractive feature in several applications (such as wireless communications and local area networks "LAN") where a separate clock line is not possible and / or not practical. The absence of a transition within one bit can be used for error detection. This family of Bi-Phase codes is very popular in many data communication and data storage applications. The Bi- $\Phi$ -L code has been specified for the IEEE 802.3 standard in the Carrier Sense Multiple Access/Collision Detection (CSMA/CD) protocol used in Ethernet networks [2]. It has also been specified in the MIL-STD-1553B bus standard, which is a shielded twisted-pair bus LAN designed to ensure high integrity data exchange between unattended equipment in military airplanes, ships and land vehicles [3]. Bi- $\Phi$ -M and Bi- $\Phi$ -S are part of the specification in amateur satellite system [4]. Bi- $\Phi$ -M is used as an industry standard for single density magnetic floppy disk recording [5]. It is also used in some optical communications and satellite telemetry applications [6]. The LocalTalk® LAN system that implemented in Apple Macintosh uses Bi- $\Phi$ -S encoding as a transmission format [7]. DM code is popular in data communications and has been specified for use as part of the physical layer in the IEEE 802.5 Token Ring local area network [8].

Currently, there is very little work in the open literature concerning the circuit implementation for encoding these codes, and if they exist, they generally encode a single code. Some of the design presented is based on the use of one shot and/or delay circuits [9]. These devices are considered inferior (more expensive and could introduce noise) to the completely synchronous circuit presented here. This work develops a design of a versatile Bi-Phase encoder circuit that can handle the change in the baseband encoding scheme without the need to swap the hardware. The circuit can encode any NRZ serial bit stream into any one of the popular Bi-Phase codes, namely, Bi- $\Phi$ -L, Bi- $\Phi$ -M, Bi- $\Phi$ -S, DM and IDM.

The paper is organized as follows. Section 2 gives a brief classical description of the five Bi-Phase line codes. In section 3, a new description for Bi-Phase codes in terms of state diagrams is presented. Section 4 shows the MBPE hardware implementation and section 5 illustrates its VHDL simulation results.

## 2. Bi-Phase Codes Overview

Bi-Phase codes are self clocking codes in which the clocking information is embedded within the serial bit stream and can be recovered without the need for a separate clocking signal, as there is at least one transition per bit period. These codes are briefly described below.

Bi- $\Phi$ -S: This code is also called FM0 [11] and classified as IFM-V6 [12]. In this code a transition is present at the start of every transmitted bit, with "0" represented by an additional transition at the middle of the bit period.

Bi- $\Phi$ -M: This code is also known as FM1 code [11] and classified as FM-V6 code [12]. In this code a transition is present at the start of every transmitted bit, with "1" represented by an additional transition at the middle of the bit period.

Bi- $\Phi$ -L: This code is also known as Manchester and split-phase code [10]. In this code there is always a transition at the middle of each bit, with "0" encoded as low-to-high transition and "1" encoded as high-to-low transition.

DM: This code is also referred to as Differential Bi-Phase Space [13]. In this code a transition always occurs at the middle of the transmitted bit period with "0" represented by an extra transition at the beginning of the bit period.

IDM: In this code a transition always occurs at the middle of the transmitted bit period with "1" represented by an extra transition at the beginning of the bit period. By analogy to DM this code can

also be referred to as Differential Bi-Phase Mark. A summary of operation of all of the five codes is given in table 1.

Table 1: Codes summary of operation  
 $\uparrow = 0$  to  $1$  transition,  $\downarrow = 1$  to  $0$  transition,  $\updownarrow =$  transition,  
 $x =$  (may / may not be a transition),  $- =$  no transition

| Code                    | '0' trans. at  |                | '1' trans. at  |                |
|-------------------------|----------------|----------------|----------------|----------------|
|                         | Beg            | Mid            | Beg            | Mid            |
| Bi-Phase-Space          | $\updownarrow$ | $\updownarrow$ | $\updownarrow$ | -              |
| Bi-Phase-Mark           | $\updownarrow$ | -              | $\updownarrow$ | $\updownarrow$ |
| Bi-Phase-Level          | x              | $\up$          | x              | $\downarrow$   |
| Differential Manchester | $\updownarrow$ | $\updownarrow$ | -              | $\updownarrow$ |
| Inverse Diff Manchester | -              | $\updownarrow$ | $\updownarrow$ | $\updownarrow$ |

### Encoding rule definition

In order to lay the ground for our synchronous design (section 3) new representations of the five biphasic codes were developed. These representations are based on a new definition of the five biphasic codes in terms of encoding rules as well as their state diagrams, apparently for the first time in the open literature. To accommodate middle and beginning of bit transitions, the encoded output is divided into two half bits. The current output is represented by  $y_{1k}$  and  $y_{2k}$ , representing the first and second half bits respectively. The next pulse output half bits are given by  $y_{1(k+1)}$  and  $y_{2(k+1)}$ . The next pulse output depends on the current output ( $y_{1k}$ ,  $y_{2k}$ ) and the current input ( $x_k$ ). The output symbols during each half bit can take the values -1 or +1, are recorded as {- or +} for simplicity. To satisfy the characteristics of the code (always transition in the middle), Bi- $\Phi$ -L, DM and IDM encoder do not produce {++} or {- -} outputs, but state {- -} occurs only once as initial condition. The standard definitions of all 5 codes, presented in table 1, are translated to the coding rule listed in Tables 2.

The coding rules for all five codes are translated into equations (1 - 5) respectively.

$$\text{Bi-}\Phi\text{-S: } y_{1(k+1)} = -y_{2k}, \quad y_{2(k+1)} = -y_{2k} (2x_k - 1) \quad (1)$$

$$\text{Bi-}\Phi\text{-M: } y_{1(k+1)} = y_{2k}, \quad y_{2(k+1)} = y_{2k} (2x_k - 1) \quad (2)$$

$$\text{Bi-}\Phi\text{-L: } y_{1(k+1)} = x_k, \quad y_{2(k+1)} = -x_k \quad (3)$$

$$\text{DM: } y_{1(k+1)} = y_{2k} (2x_k - 1), \quad y_{2(k+1)} = -y_{2k} (2x_k - 1) \quad (4)$$

$$\text{IDM: } y_{1(k+1)} = -y_{2k} (2x_k - 1), \quad y_{2(k+1)} = y_{2k} (2x_k - 1) \quad (5)$$

Table 2: Bi-Phase coding rule.

$y_1 = y_{1k}$ ,  $y_2 = y_{2k}$ ,  $y_1^+ = y_{1(k+1)}$ ,  $y_2^+ = y_{2(k+1)}$  xx = not defined

| $y_1$ | $y_2$ | Bi- $\Phi$ -S |         |           |         | Bi- $\Phi$ -M |         |           |         | Bi- $\Phi$ -L |         |           |         | DM        |         |           |         | IDM       |         |           |         |
|-------|-------|---------------|---------|-----------|---------|---------------|---------|-----------|---------|---------------|---------|-----------|---------|-----------|---------|-----------|---------|-----------|---------|-----------|---------|
|       |       | $x_k = 0$     |         | $x_k = 1$ |         | $x_k = 0$     |         | $x_k = 1$ |         | $x_k = 0$     |         | $x_k = 1$ |         | $x_k = 0$ |         | $x_k = 1$ |         | $x_k = 0$ |         | $x_k = 1$ |         |
|       |       | $y_1^+$       | $y_2^+$ | $y_1^+$   | $y_2^+$ | $y_1^+$       | $y_2^+$ | $y_1^+$   | $y_2^+$ | $y_1^+$       | $y_2^+$ | $y_1^+$   | $y_2^+$ | $y_1^+$   | $y_2^+$ | $y_1^+$   | $y_2^+$ | $y_1^+$   | $y_2^+$ | $y_1^+$   | $y_2^+$ |
| -     | -     | +             | -       | +         | +       | +             | +       | +         | -       | -             | +       | +         | -       | +         | -       | -         | +       | -         | +       | +         | -       |
| -     | +     | -             | +       | -         | -       | -             | -       | -         | +       | -             | +       | +         | -       | -         | +       | +         | -       | +         | -       | -         | +       |
| +     | -     | +             | -       | +         | +       | +             | +       | +         | -       | -             | +       | +         | -       | +         | -       | -         | +       | -         | +       | +         | -       |
| +     | +     | -             | +       | -         | -       | -             | -       | -         | +       | xx            | xx      | xx        | xx      | xx        | xx      | xx        | xx      | xx        | xx      | xx        | xx      |

### 3. Design of the Multi-Code Bi-Phase Encoder (MBPE)

In this section the design of a MBPE digital circuit is presented. The circuit is capable of encoding NRZ data format into five different Bi-Phase codes, namely, the Bi- $\Phi$ -L, Bi- $\Phi$ -M, Bi- $\Phi$ -S, DM and IDM line codes. Three *code select* inputs are used to alter the encoding scheme.

#### 3.1. Functional Description

The block diagram of the Bi-Phase encoder is shown in figure 2.

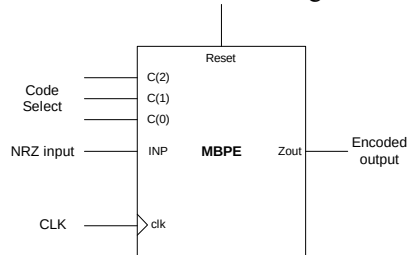


Fig. 2: MBPE block diagram

The encoder accepts as an input: the three code select lines ( $C_2$   $C_1$   $C_0$ ), the serial NRZ bit stream (INP), an asynchronous RESET input and a clock signal. The encoder generates a single output ( $Z_{out}$ ) that represents one of the Bi-Phase codes listed in table 1, depending on the settings of the code select lines. The clock frequency used in this encoder circuit is double the frequency of the incoming NRZ bit stream.

#### 3.2. Behavioral Description of the Encoder

The behavior of the encoder circuit can be best described by a Moore type state machine. For clarity, the state diagram for each code is drawn separately in Figure 3.

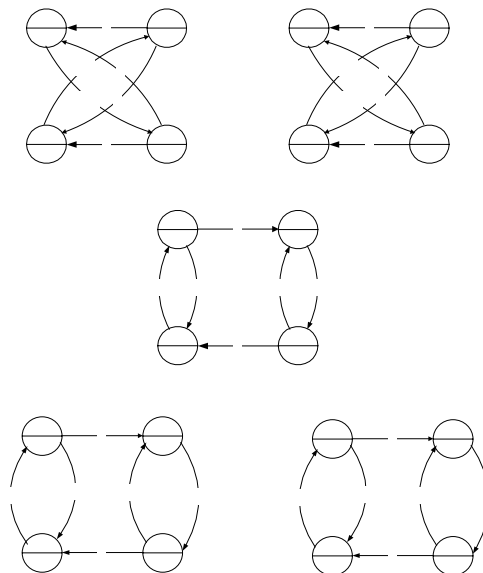


Fig. 3: State diagrams for all five Bi-Phase codes.

By inspecting these state machines, it is noticed that they share a common feature where they all consist of 4 states, with two states having an output of "0" and the other two having an output of "1". The two states with "0" outputs are named  $S_0$  and  $S_1$  while the other two states (with "1" outputs) are  $S_2$  and  $S_3$ . These five state graphs are translated into a single state table as shown in table 3. Table 3 also shows how the control lines  $C_2$ ,  $C_1$  and  $C_0$  are assigned to the different Bi-Phase codes.

Table 3: State assignment for all five codes

| State<br>Assignment | Present<br>State | Next State  |        |    |        |    |        |    |     |    |     |    | Output |
|---------------------|------------------|-------------|--------|----|--------|----|--------|----|-----|----|-----|----|--------|
|                     |                  | $C_2C_1C_0$ | Bi-Φ-S |    | Bi-Φ-M |    | Bi-Φ-L |    | DM  |    | IDM |    |        |
|                     |                  |             | 000    |    | 001    |    | 010    |    | 011 |    | 100 |    |        |
|                     |                  |             | IN     | 0  | 1      | 0  | 1      | 0  | 1   | 0  | 1   | 0  |        |
| 00                  | S0               |             | S2     | S2 | S2     | S2 | S1     | S3 | S3  | S1 | S1  | S3 | 0      |
| 01                  | S1               |             | S3     | S0 | S0     | S3 | S2     | -  | S2  | S2 | S2  | S2 | 0      |
| 10                  | S2               |             | S0     | S3 | S3     | S0 | S1     | S3 | S1  | S3 | S3  | S1 | 1      |
| 11                  | S3               |             | S1     | S1 | S1     | S1 | -      | S0 | S0  | S0 | S0  | S0 | 1      |

The next step toward the design of the encoder circuit is to decide on state encoding. Research has shown that changing state assignment can dramatically affect the performance of code targeted to a specific device or architecture. Therefore, a poor state assignment will result in a state machine that uses too much logic, or is too slow, or both. Various approaches have been developed for finding the “optimal” state assignment [14]. It must be mentioned here that the method of assigning binary codes to the five Bi-Phase codes (i.e.; binary assignment for inputs  $C_2$ ,  $C_1$ ,  $C_0$ ) also affects the realization cost of the encoder. By carrying on an exhaustive search to the minimum circuit realization, it was found that almost all possible combination of code-selection and state assignments lead to logic circuits having a comparable realization cost that lies between a total gate cost of 16 to 20 and a total number of input cost between 72 to 74. Straight binary assignment is among one of the possible state assignments that led to a minimum realization cost with a gate cost of 16 gates and total number of gate’s inputs of 72. Therefore, for simplicity, straight binary assignment for the states is selected, with the code selection assignment shown in table 3. With this state assignment, the D flip-flops input equations are given by:

$$D_1 = \overline{C_2} \overline{C_0} \text{INP} \overline{Q_0} + \overline{C_1} C_0 \text{INP} \overline{Q_0} + C_1 \text{INP} Q_1 \overline{Q_0} + C_2 \text{INP} Q_1 \overline{Q_0} + C_1 C_0 \text{INP} \overline{Q_1} + \overline{C_0} \text{INP} \overline{Q_1} Q_0 + \overline{C_2} \overline{C_1} \overline{Q_1} \overline{Q_0} + C_2 \text{INP} \overline{Q_1} + C_0 \text{INP} \overline{Q_1} \overline{Q_0} \quad (6)$$

$$D_0 = C_1 \overline{Q_0} + C_2 \overline{Q_0} + \overline{C_2} \overline{C_1} \overline{C_0} \text{INP} Q_0 + \overline{C_2} \overline{C_1} \overline{C_0} \text{INP} Q_1 + C_2 \text{INP} Q_1 \overline{Q_0} + \overline{C_1} C_0 \text{INP} Q_0 + \overline{C_1} C_0 \text{INP} Q_1 \quad (7)$$

The encoder output equation is given by:

$$Z_{\text{out}} = Q_1 \quad (8)$$

#### 4. MBPE Hardware Implementation

Various digital techniques for implementing the MBPE circuit in hardware exist. The simplest is based on the derived equations presented in the previous section (equations 6, 7 and 8) using discrete logic gates and D flip flops. In this case, the circuit requires two D flip-flops, four

inverters, seventeen 4-inputs AND gates and six 4-inputs OR gates. A total of 14 SSI (Small Scale Integration) ICs is needed for the complete design.

A more efficient implementation of the MBPE circuit can be achieved by using a Programmable Array Logic (PAL) such as the 22V10 PAL [15]. The 22V10 is a versatile programmable logic device that has a programmable AND array feeding ten OR gates. The output of each OR gate is connected to programmable Macrocells that contains a D flip flop. The logic described by equations 6-8 can be easily accommodated into a single 22V10 PAL. This will lead to a simpler design that is smaller in size, cheaper, uses less power and more reliable. Figure 4, shows a generic connection matrix for a PAL device implementing the MBPE circuit.

An alternative approach for the implantation of the encoder is to use VHDL (VHSIC Hardware Description Language) for describing the behavior of the encoder state diagram.

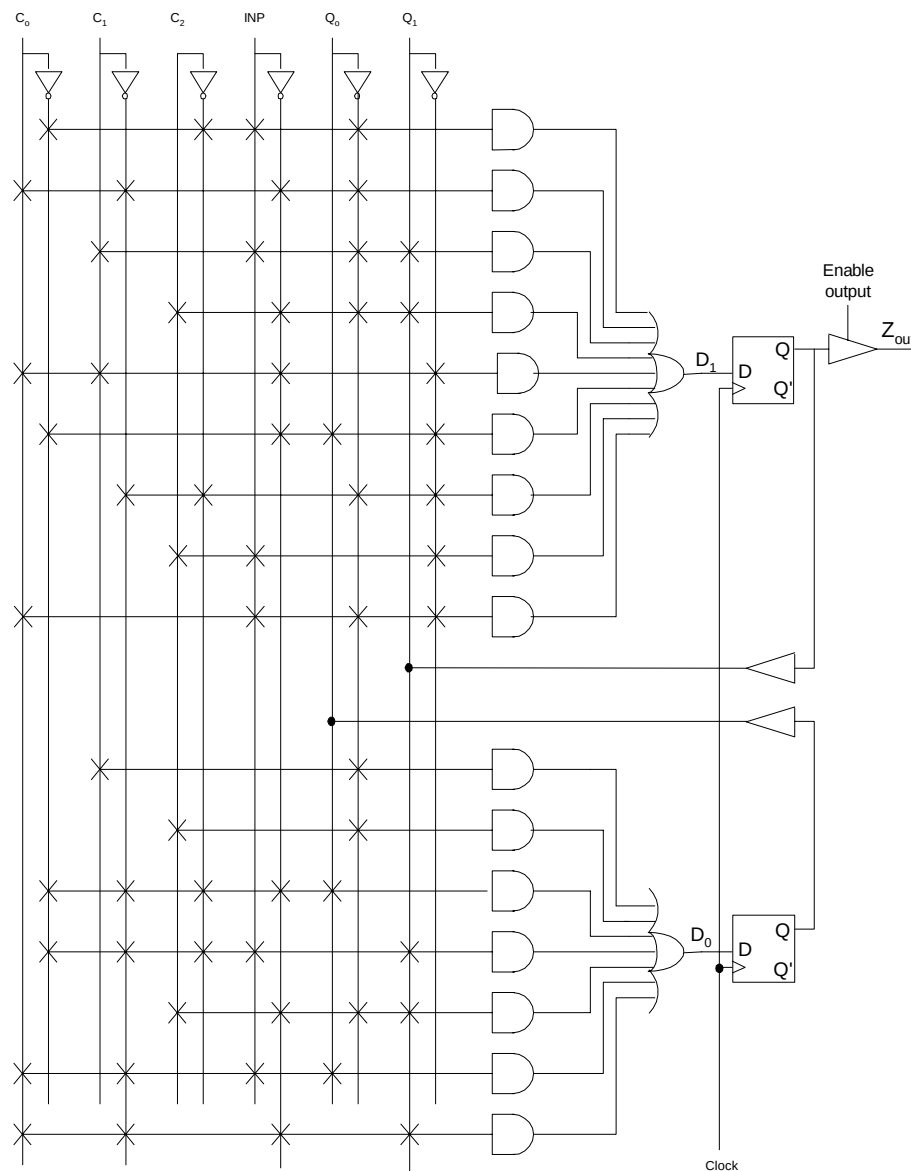


Fig. 4: Connection matrix for a PAL device realizing the MBPE circuit

VHDL is a technology independent design tool that permits the creation of a portable design and simulating it without the need of specifying the target hardware [16]. The above five state machines are combined into a single state machine whereby the transition from one state to another depends on the incoming NRZ code as well as on the code select inputs. The combined MBPE state machine has been described and simulated in VHDL.

## 5. VHDL Simulation Results

The MBPE circuit is coded in VHDL then simulated, tested and implemented into XILINX Spartan-II FPGA. To verify the behavior of VHDL source code of the MBPE, a VHDL test bench is generated. This test bench is applied as a stimulus to the MBPE. The design is simulated and tested using *ModelSim XILINX Edition* Simulation Environment [17]. Various test patterns are tried and it is found that the MBPE can encode the five different Bi-Phase line codes successfully. The VHDL code verification results are shown in Figures 5 – 9 for the Bi- $\Phi$ -S, Bi- $\Phi$ -M, Bi- $\Phi$ -L, DM and IDM respectively. In these figures, the waveforms represent clock (CLK), reset, NRZ data input (INP) and encoded output (Zout) signals respectively from top to bottom. The results shown in these figures are for the test pattern 011010011 represented by the serial NRZ bit stream (INP). This test pattern is selected to assess the encoders under various conditions including consecutive ones and consecutive zeros. Each bit last for two clock cycles since the clock frequency (CLK) used in this encoder circuit is double the frequency of the incoming NRZ data input. The generated encoder's output Zout is synchronized with the positive edge of the CLK pulse and the generated Zout waveform agrees with the expected encoder results for the five bi-phase codes.

When implemented on XILINX Spartan-II FPGA, the MBPE has occupied 6 slices out of the 2,352 total slices on chip with an equivalent gate count equals to 85.

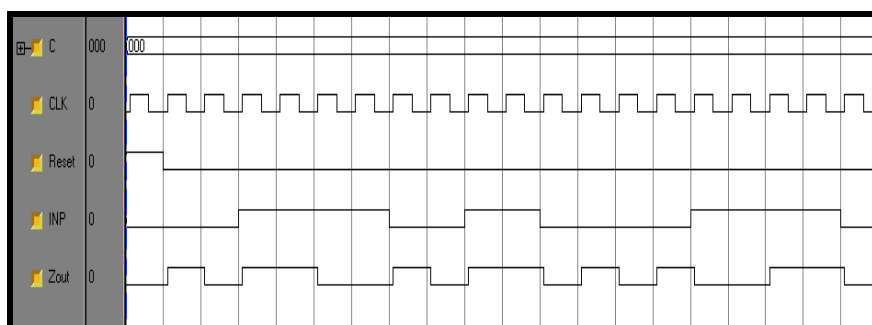


Fig. 5: Bi- $\Phi$ -S encoder output

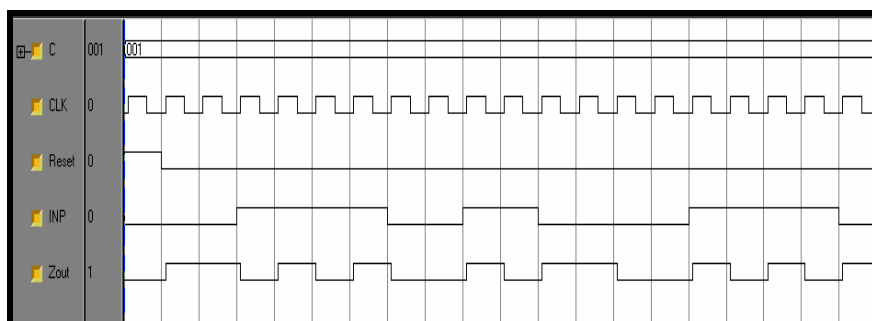


Fig. 6: Bi- $\Phi$ -M encoder output

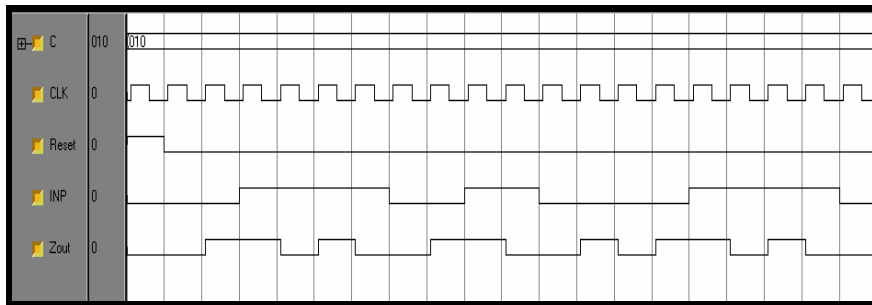


Fig.7: Bi- $\Phi$ -L encoder output

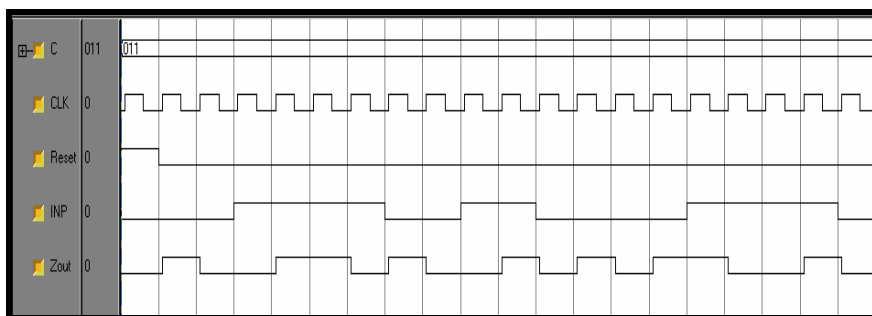


Fig. 8: DM encoder output

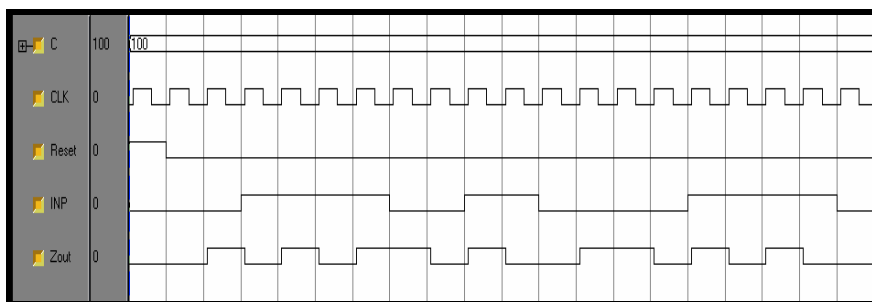


Fig. 9: IDM encoder output

## 6. Conclusions:

In this paper, a new approach for designing a Muticode Bi-Phase Encoder (MBPE) is presented. The encoder is capable of encoding any NRZ bit stream into one of five Bi-Phase codes, namely, Bi-Phase-level, Bi-Phase-Mark, Bi-Phase-Space, Differential Manchester and Inverse Differential Manchester codes. The encoder design methodology is based on defining the state machine of the five codes and combining them into a single machine with code selection capability. The Behavior of the encoder is described in VHDL and simulated using *ModelSim XILINX Edition* Simulation tools. Simulation results shows that the encoder has successfully encoded the NRZ code into any of the selected Bi-Phase codes. As a result, this design can be targeted without modifying the VHDL source code towards a range of PLD devices. The significance of the proposed design is the ability to change the encoding scheme without the need for swapping hardware. Switching from one encoding scheme to another is performed by simply changing the inputs at the code selection lines.

The proposed network can be inserted in larger communication systems that can handle multiple protocols and use bi-phase data encoding formats

## 7. References:

1. Zilog Inc., "Design a serial board to handle multiple protocols", *Zilog application note* document ID: AN97USC0200, 1997.
2. Schwartz, M., *Information Transmission, Modulation and Noise*, McGraw Hill, 1990 ( 4<sup>th</sup> Edn).
3. Killough R.L., "Designing command and telemetry systems using MIL-STD-1553 and CCSDS", *AIAA/USU Conference on Small Satellites in Logan*, Utah, August 2000.
4. Amateur satellite, [www.scit.wlv.ac.uk/vhfc/iaru.r1.vhfm.4e/7A.html](http://www.scit.wlv.ac.uk/vhfc/iaru.r1.vhfm.4e/7A.html).
5. Moore, S.J., "A Formal model of asynchronous communication and Its use in mechanically verifying a bi-phase mark protocol", *Formal Aspect of Computing*, Vol. 6, No. 1, pp. 60-69, 1994.
6. Sklar, B., *Digital Communications Fundamentals and Applications*, Prentice Hall, 2000 (2<sup>nd</sup> Edn).
7. Gursharan S. Sidhu, R.F. Andrews, and Oppenheimer A.B., *Inside AppleTalk*, Don Mills, ON, Addison-Wesley Publishing Co., 1990. (2<sup>nd</sup> Edn)
8. IEEE, "Token Ring Access Method and Physical Layer Specifications", *ANSI/IEEE Standard 802.5*, 1998 Edition, ISO/IEC 8802-5:1998E.
9. Al-Sammak A.J., "Encoder Circuit for inverse differential Manchester Code" Operating at any Frequency", *IEE Electronics Letters*, Vol. 38, No. 12, 6<sup>th</sup> June 2002, pp. 567 – 568.
10. Hayken, S., *An Introduction to analog and digital communication*, John Wiley, 2001.
11. Han E.: "Implementing SDLC on the Am186 CC or Am186CH Microcontroller", *application note (publication # 24945)*, *Advanced Micro Devices (AMD)*, 2001.
12. Tsuzuki, S., Osaka, S., Tazaki, S. and Yamada, Y., "Periodic correlation properties of FM coded M Sequences having DC free spectrum", *IEICE trans. Fundamentals*, Vol. E78-A, No. 9, September 1995.
13. Sorrells, P.: "Passive RFID basics", *AN680 application note*, *Microchip Technology*, 1998.
14. De Micheli G., Brayton R.K. and Sangiovanni-Vincentelli A., "Optimal state assignment for finite state machines", *IEEE Trans. Computer-Aided Design*, vol. CAD-4, no. 3, pp. 269-285, July 1985.
15. Bolton, M., *Digital System Design with Programmable Logic*, Addison-Wesley, 1990.
16. Yalamanchili, S., *Introductory VHDL: From Simulation to Synthesis*, New Jersey, Prentice Hall, 2001.
17. Model Technology: "ModelSim Xilinx Edition II V5.6e", [www.model.com](http://www.model.com), U.S.A., Model Technology , 2003