CHAPTER 1: INTRODUCTION

Dedicated short-range communications (DSRC) are one-way or two-way short-range to medium-range wireless communication channels specifically designed for automotive use and a corresponding set of protocols and standards.

The dedicated short range communication (DSRC) is an etiquette for one way or two way medium range communication mainly for the intelligent transportation system (ITS). The automobile to roadside and automobile to automobile are two important categories in DSRC. In automobile to automobile, the DSRC sends and receives messages from one automobile to other for safety issues and public information. The safety matters incorporate inter cars distance, collision alarm, intersection warning and blind spot. The automobile to roadside DSRC systems are mainly concentrating on the intelligent transportation service, such as the electronic toll collection (ETC) system. Using the ETC system, the toll collection is electrically consummate with the contactless IC-card platform. The payment for parking service also used the ETC system.

Parameter	Europe	America	Japan
Organization	European Committee for Standardization	America Society for Testing and Materials	Associate of Radio Industries and Business
Data rate	500kbps	27Mbps	4Mbps
Carrier Frequency	5.8GHz	5.9 GHz	5.8GHz
Modulation	ASK,PSK	OFDM	ASK
Encoding	FM0	Manchester	Manchester

Table I: Profile of DSRC Standards for America, Europe and Japan

The DSRC standards have been recognized by several organizations in various countries. Table I shows the DSRC standards of America, Japan and Europe. The transmitted signal is anticipated to have zero mean for vigor issue and this is called dc-balance. But the transmitted signal contains an arbitrary binary sequence, which is difficult to obtain the dc-balance. So, the encoders are used in communication to convert one form of information into another form, which is suitable for transmission. Different types of encoders are available in communication systems. The FM0 and Manchester encoders are mainly used in DSRC to attain the dc-balance.

Communications-based active safety applications use vehicle to vehicle and vehicle to infrastructure short-range wireless communications to detect potential hazards in a vehicle's path – even those the driver does not see. The connected vehicle provides enhanced awareness at potentially reduced cost, and offers additional functionality over autonomous

sensor systems available on some vehicles today. Communications-based sensor systems could potentially be a low-cost means of enabling hazard detection capability on all vehicle classes, but requires vehicles and infrastructure to be outfitted with interoperable communications capabilities.

1.1 FEATURES OF DSRC

Dedicated Short Range Communications (DSRC) are the communications media of choice for communications-based active safety systems research because:

- It operates in a licensed frequency band.
- It is primarily allocated for vehicle safety applications by FCC Report & Order – Feb. 2004 (75 MHz of spectrum).
- It provides a secure wireless interface required by active safety applications.
- It supports high speed, low latency, short-range wireless communications.
- It works in high vehicle speed mobility conditions.
- Its performance is immune to extreme weather conditions (e.g. rain, fog, snow, etc.).
- It is designed to be tolerant to multi-path transmissions typical with roadway environments.
- It supports both vehicle-to-vehicle and vehicle-to-infrastructure communications.

Desired features of DSRC for active safety systems are Communications-based active safety systems that need a tightly controlled spectrum for maximized reliability. DSRC communications take place over a dedicated 75 MHz spectrum band around 5.9 GHz, allocated by the US Federal Communications Commission (FCC) for vehicle safety applications.

1.2 DSRC and Wi-Fi

We use DSRC instead of Wi-Fi since DSRC is preferred over Wi-Fi because the proliferation of Wi-Fi hand-held and hands-free devices that occupy the 2.4 GHz and 5 GHz bands, along with the projected increase in Wi-Fi hotspots and wireless mesh extensions, could cause intolerable and uncontrollable levels of interference that could hamper the reliability and effectiveness of active safety applications.

DSRC was developed with a primary goal of enabling vehicular safety applications. DSRC is the only short-range wireless alternative today that provides:

1. Fast Network Acquisition: Active safety applications require immediate establishment of communication.
2. Low Latency: Active safety applications must execute in the smallest amount of time possible.

3. High Reliability when Required: Active safety applications require high level of link reliability.
4. Priority for Safety Applications: Safety applications on DSRC are given priority over non-safety applications.
5. Interoperability: DSRC ensures interoperability, which is the key to successful deployment of active safety applications.
6. Security and Privacy: DSRC provides safety message authentication and privacy.

The normal Wi-Fi means of recognizing nearby stations and associating with them (establishing a link between two or more devices) cannot be used for active safety applications because it can take multiple seconds to complete this association. Active safety applications require immediate establishment of communication. Several changes from the basic technology (Wi-Fi) were required to achieve this goal. The most significant change is to accommodate an extremely short time in which devices must recognize each other and transmit messages to each other. A large number of these safety applications require response times measured in milliseconds. For this reason, the periodic transmission of safety messages is used so that vehicles receiving the safety messages can immediately determine if they should respond or not.

CHAPTER 2: LITERATURE SURVEY

2.1 Data Encoding

Encoding is the process of using various patterns of voltage or current levels to represent 1s and 0s of the digital signals on the transmission link.

The data encoding technique is divided into the following types, depending upon the type of data conversion.

- Analog data to Analog signals – The modulation techniques such as Amplitude Modulation, Frequency Modulation and Phase Modulation of analog signals, fall under this category.
- Analog data to Digital signals – This process can be termed as digitization, which is done by Pulse Code Modulation PCM. Hence, it is nothing but digital modulation. As we have already discussed, sampling and quantization are the important factors in this. Delta Modulation gives a better output than PCM.
- Digital data to Analog signals – The modulation techniques such as Amplitude Shift Keying ASK, Frequency Shift Keying FSK, Phase Shift Keying PSK, etc., fall under this category.
- Digital data to Digital signals – There are several ways to map digital data to digital signals. Some of them are Non Return to Zero NRZ (NRZ-L and NRZ-I), Return to Zero RZ and Bi-phase Encoding (Manchester Encoding and Differential Manchester Encoding)

2.2 Manchester Encoding

In telecommunication and data storage,Manchester coding (also known as phase encoding, or PE) is a line code in which the encoding of each data bit has at least one transition and occupies the same time. It therefore has no DC component, and is self-clocking, which means that it may be inductively or capacitively coupled, and that a clock signal can be recovered from the encoded data. As a result, electrical connections using a Manchester code are easily galvanically isolated using a network isolator—a simple one-to-one isolation transformer. Manchester coding is one of the most common data coding methods used today. Similar to BiPhase, Manchester coding provides a means of adding the data rate clock to the message to be used on the receiving end. Also Manchester provides the added benefit of always yielding an average DC level of 50%. This has positive implications in the demodulator's circuit design as well as managing transmitted RF spectrum after modulation.

This means that in modulation types where the power output is a function of the message such as AM, the average power is constant and independent of the data stream being encoded. Manchester coding states that there will always be a transition of the message signal at the midpoint of the data bit frame. What occurs at the bit edges depends on the state of the previous bit frame and does not always produce a transition. A logical “1” is defined as a midpoint transition from low to high and a “0” is a midpoint transition from high to low. Encoding is the process of adding the correct transitions to the message signal in relation to the data that is to be sent over the communication system

The first step is to establish the data rate that is going to be used. Once this is fixed, then the mid-bit time can be determined as ½ of the data rate period. In our example we are going to use a data rate of 4 kHz. This provides a bit period of 1/f = 1/4000 = 0.00025s or 250 µs. Dividing by two gives us the mid-bit time (which we will label “T”) of 125 µs. Now let's look at how we use this to encode a data byte of 0xC5 (11000101b). The easiest method to do this is to use a timer set to expire or interrupt at the T interval. We also need to set up a method to track which ½ bit period we are currently sending. Once we do this, we can easily encode the data and output the message signal.

1. Begin with the output signal high.

2. Check if all bits have been sent, If yes, then go to step 7

3. Check the next logical bit to be coded

4. If the bit equals “1”, then call ManchesterOne(T)

5. Else call ManchesterZero(T)

6. Return to step 2

7. Set output signal high and return

The Manchester encoding is realized with a XOR operation for CLK and *X*. The clock always has a transition within one cycle, and so does the Manchester code no matter what the *X* is.

Manchester encoding is also referred to as phase encoding. This encoding technique is generally used for higher operating frequencies. In this coding, the signal is transmitted serially. The Manchester encoder is implemented with an exclusive OR (XOR) operation for X and CLK. The Fig.2.1. Shows the Manchester encoder structure.

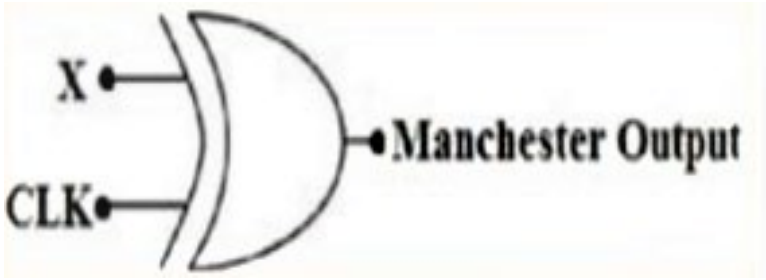


Fig 2.1 Manchester Encoder

$$X \oplus CLK. \qquad (1)$$

Example for Manchester encoding is given in Fig.2.2. For each X, Manchester code contains two parts: one for the former half cycle of clock signal and other for the latter half cycle of clock signal. Table II shows the operation of Manchester encoding.

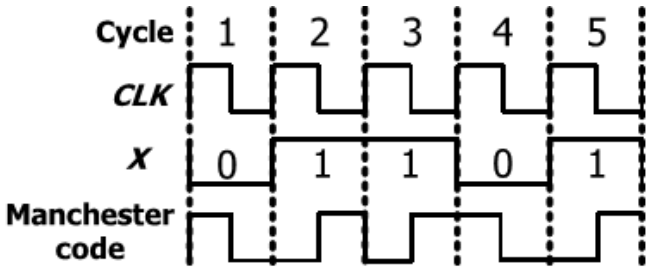


Fig. 2.2 Example of Manchester encoding

In Manchester encoding, if the input X is logic-0 and clock signal CLK is logic-1 then output signal is logic-1. If the input X is logic-0 and clock signal CLK is logic-0 then output signal is logic-0. If the input X is logic-1 and clock signal CLK is logic-1 then output signal is logic-0. If the input X is logic-1 and clock signal CLK is logic-0 then output signal is logic-1.

Input Data	Clock Signal	Manchester Code
0	1	1
	0	0
1	1	0
	0	1
1	1	0
	0	1
0	1	1
	0	0
1	1	0
	0	1

Table II: Operation of Manchester Encoding

2.2.1 Features Manchester Encoding:

1. Manchester code ensures frequent line voltage transitions, directly proportional to the clock rate; this helps clock recovery.

2. The DC component of the encoded signal is not dependent on the data and therefore carries no information, allowing the signal to be conveyed conveniently by media (e.g., Ethernet) which usually do not convey a DC component

2.3 FM0 encoding:

FM0 is also known as Biphase space encoding. A transition is present on every bit and an additional transition may occur in the middle of the bit. Here the data rate is twice. Sufficient clock information can be recovered from the data stream so that a separate clock is not needed. Therefore, for transmission, the number of wires is minimized. Logic 0 represents the transition in the center of the bit. Logic 1 represents there is no transition from the center of bit.

This encoding data contains sufficient information to recover a clock from the data. It has to reach the DC balance and enhance signal reliability. It is used to reduce noise and transmission power. The block diagram has an XOR gate, DFF, inverter, and MUX. For example, the XOR gate has one input as feedback that is 0, and another input as 1. This XOR output is given to DFF1, and it also has a CLK signal with an output of 1. Another DFF2 has an input as 1 and CLK. The output is 1. Both DFF outputs are given to MUX, and also it has a CLK with it that produces the output based on selection lines. If the selection line is 0, it

produces the output as DFF1 as FM0 output. Otherwise, the selection line is 0 and produces an output as DFF2 or FM0 output.

For each input , the FM0 code consists of two parts: one for the former-half cycle of CLK, A, and the other one for the later-half cycle of CLK, B. The coding principle of FM0 is listed as the following three rules.

1. If input data X is the logic-0, the FM0 code must demonstrate a transition between the former half cycle of CLK and later half cycle of CLK.
2. If X is the logic-1, no transition is permitted between half cycles of CLK. 3.
3. The transition is billed among each FM0 code, no matter what the X is.

Example for FM0 encoding is shown in Fig.2.3. For each X, the FM0 code contains two parts: one for the former half cycle of CLK and other for the later half cycle of CLK.

1. For simplicity, this transition is initially set from logic-0 to -1. According to rule 3, a transition is allocated among each FM0 code, and thereby the logic-1 is changed to logic-0 in the beginning of cycle 4

2. Then, according to rule 2, this logic-level is hold without any transition in entire cycle 2 for the X of logic-1. Thus, the FM0 code of each cycle can be derived with these three rules mentioned earlier.

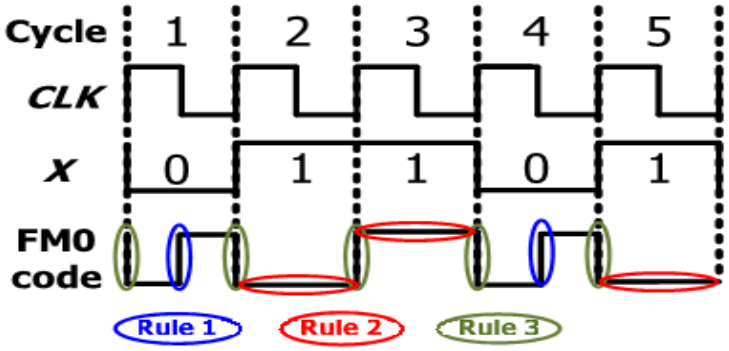


Fig.2.3 Example of FM0 encoding

The Manchester encoder hardware architecture contains an XOR operation as shown in Fig.2.1. The Hardware Architecture of the FM0 encoder is designed using the finite state machine (FSM). The FSM of FM0 code contains four states that can be shown in Fig.2.4.

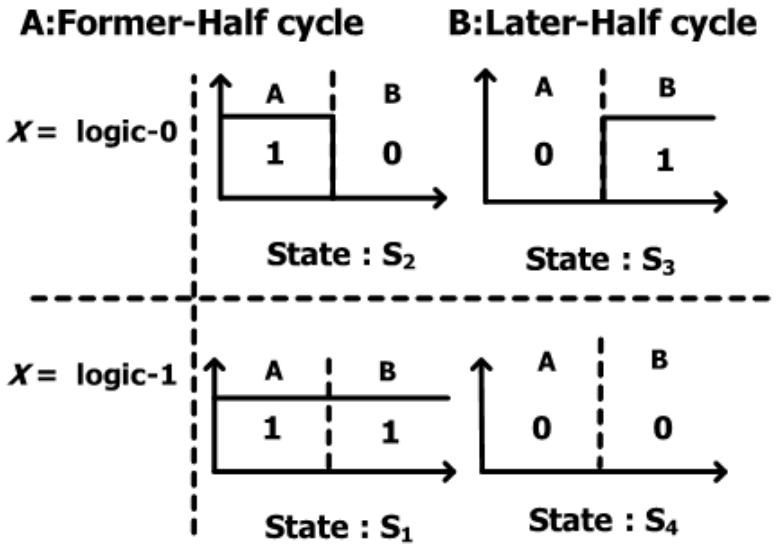


Fig 2.4 States definition for FM0 encoding

Each state is assigned to a state code and each state code contains A and B. Based on the FM0 contained coding principles, Fig.2.5 shows the FSM of FM0. Suppose the initial state is S1 and its state code is 11 for A and B. If the input signal X is logic-0, the state transition must pursue both rules 1 and 3. If the X is logic-1, based on rules 2 and 3 the state transition is done.

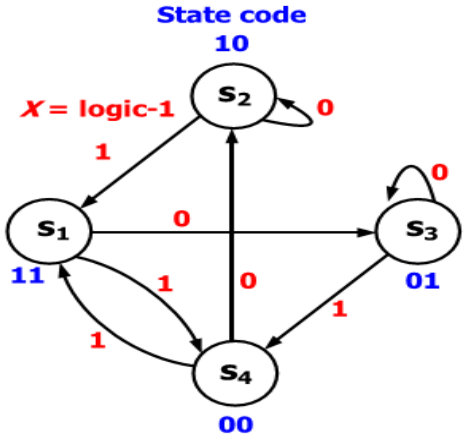


Fig 2.5 Finite State Machine(FSM) of FM0

Table III shows the state transition table of FM0. In Table III A(t) and B(t) represent the current states at time instant ‘t’. A(t-1) and B(t-1) represent the previous states.

Previous State		Current State			
A(t-1)	B(t-1)	A(t)		B(t)	
		X=0	X=1	X=0	X=1
1	1	0	0	1	0
1	0	1	1	0	1
0	1	0	0	1	0
0	0	1	1	0	1

Table III : State Transition table of FM0

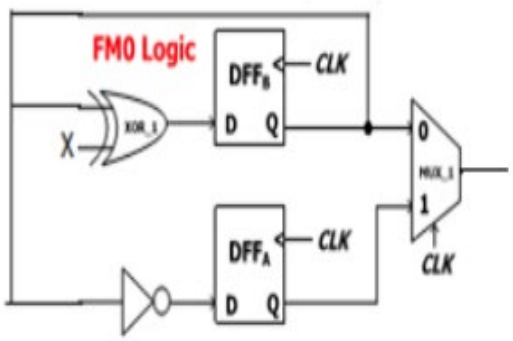


Fig. 2.6 FM0 Encoder

A(t) and B(t) represent the discrete-time state code of the current-state at time instant t. Their previous-states are denoted as the A(t − 1) and the B(t − 1), respectively. With this transition table, the Boolean functions.of A(t) and B(t) are given as

$$A(t) = B(t-1)' \qquad (2)$$

$$B(t) = X \oplus B(t-1) \qquad (3)$$

With both A(t) and B(t), the Boolean function of FM0 code is denoted as

$$\text{FM0 code} = \text{CLK } A(t) + \text{CLK } B(t). \qquad (4)$$

CHAPTER 3: EXISTING METHOD

This is the hardware architecture of the fm0/Manchester code. the top part is denoted the fm0 code and then the bottom part is denoted as the Manchester code .In fm0 code the DFFA and DFFB are used to store the state code of the fm0 code and also mux_1 and not gate is used in the fm0 code. when the mode=0 is for the fm0 code. With (1) and (4), the hardware architectures of FM0 and Manchester encoders are shown in Fig. 3.1. The top part is the hardware architecture of FM0 encoder, and the bottom part is the hardware architecture of Manchester encoder.

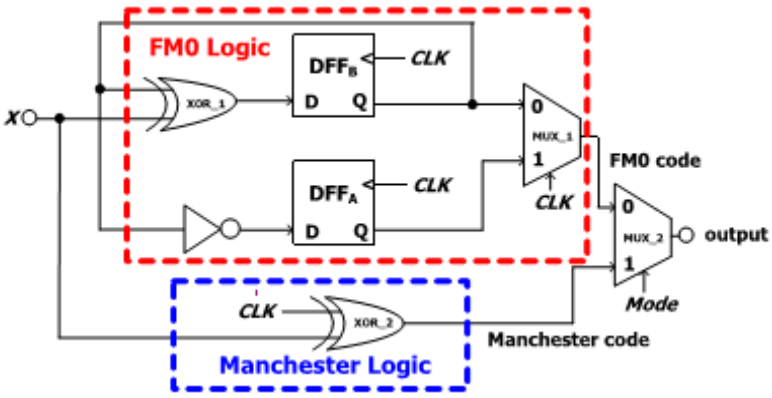


Fig.3.1 Existing hardware architecture of FM0 and Manchester encoding

As mentioned before, the Manchester encoder is as simple as a XOR operation for *X* and CLK. Nevertheless, the FM0 encoding depends on the code. The DFFA and DFFB store the state code of the FM0 code. The MUX −1 is to switch A(t) and B(t) through the selection of CLK signal. Both A(t) and B(t) are realized by (2) and (3), respectively. The determination of which coding is adopted depends on the Mode selection of the MUX−2, where the Mode = 0 is for FM0 code, and the Mode = 1 is for Manchester code. To evaluate the hardware utilization, the hardware utilization rate (HUR) is defined as

$$\text{HUR} = \frac{Active\ components}{Total\ components} \times 100\%. \qquad (5)$$

The component is defined as the hardware to perform a specific logic function, such as AND, OR, NOT, and flip flop. The active components mean the components that work for FM0 or Manchester encoding. The total components are the number of components in the entire hardware architecture no matter what encoding method is adopted. The HUR. of FM0 and

Manchester encodings is listed in Table IV. For both encoding methods, the total components are 7, including MUX-2 to indicate which coding method is activated. For FM0 encoding, the active components are 6, and its HUR is 85.71%. For Manchester encoding, the active components are 2, comprising XOR-2 and MUX-2, and its HUR is as low as 28.57%. On average, this hardware architecture has a poor HUR of 57.14%, and almost half of total components are wasted. The coding-diversity between the FM0 and Manchester codes seriously limits the potential to design a fully reused VLSI architecture.

Coding	Active components / Total components	HUR
FM0	6/7	85.71%
Manchester	2/7	28.57%
Average	4/7	57.14%

Table IV : HUR of FM0/Manchester encoder

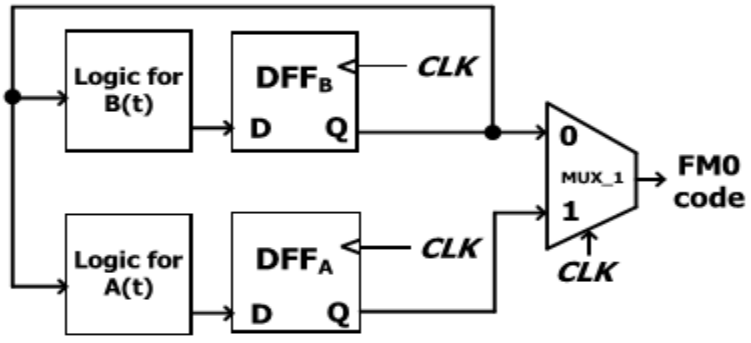
CHAPTER 4: THEORETICAL ASPECTS OF THE PROJECT

VLSI ARCHITECTURE DESIGN OF FM0 ENCODER AND MANCHESTER ENCODER USING SIMILARITY-ORIENTED LOGIC SIMPLIFICATION (SOLS) TECHNIQUE

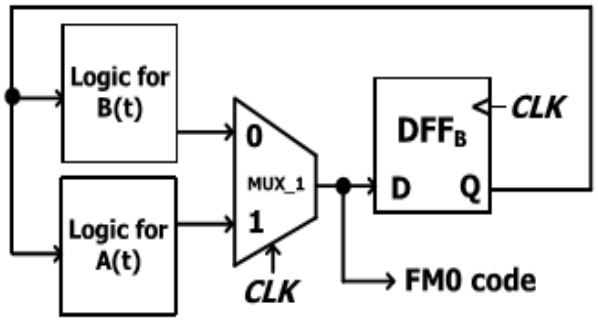
Encoding techniques are fetching important roles in communication. Techniques like Manchester, and FM0 encoding are used in numerous applications. Each technique has different operations depending on their needs. Each and every encoding scheme is used without losing any of its parameters. This paper adopts similarity oriented logic simplification technique (SOLS technique) which merges architecture together and synchronizes the operation and also DSRC technique is used to maintain the dc balance and signal reliability. By applying both techniques we can reduce the number of transistors and maintain the DC balance. The present work deals with obtaining an integrated architecture of FM0, Manchester to overcome several drawbacks of existing method. The purpose of SOLS technique is to design a fully reused VLSI architecture for FM0 and Manchester encodings. The SOLS technique is classified into two parts: area-compact retiming and balance logic-operation sharing. Each part is individually described as follows.

4.1 Area-Compact Retiming

The FM0 logic in Fig.3.1 is simply shown in Fig.4.1(a). The logic for B(t) and the logic for A(t) are the Boolean functions to obtain B(t) and A(t). For FM0, the DFFA and DFFB are used to store the state code of each state. From equations (2) and (3) the alteration of state code depends only on B(t-1) instead of both B(t-1) and A(t-1).



(a)



(b)

Fig. 4.1 Illustration of area-compact retiming on FM0 encoding architecture. (a) FM0 encoding without area-compact retiming. (b) FM0 encoding with area-compact retiming

A single one bit flip flop is only required for FM0 encoding to store the B(t-1). If the flip flop DFFA is removed directly, that presents non synchronization between A(t) & B(t) and causes the logic blunder of FM0 code. The logic fault is avoided by replacing the flip flop DFFB after the MUX_1 as shown in Fig.4.1(b), Where the DFFB is tacit to be positive edge triggered. At each cycle the FM0 code embraces A & B and is obtained from the logic of A(t) and B(t). The MUX_1 is used to switch the FM0 code alternately between A(t) and B(t) with control signal CLK. In Fig.4.1(a) the output of flip flop DFFB is directly given to the logic for B(t) with one cycle latency. If the CLK is logic-0, the MUX_1 selects the B(t) and conceded to the D of DFFB . Then the impending positive edge of the CLK revises it to the Q of DFFB. The timing diagram of FM0 encoding with area compact retiming as shown in Fig.4.2.

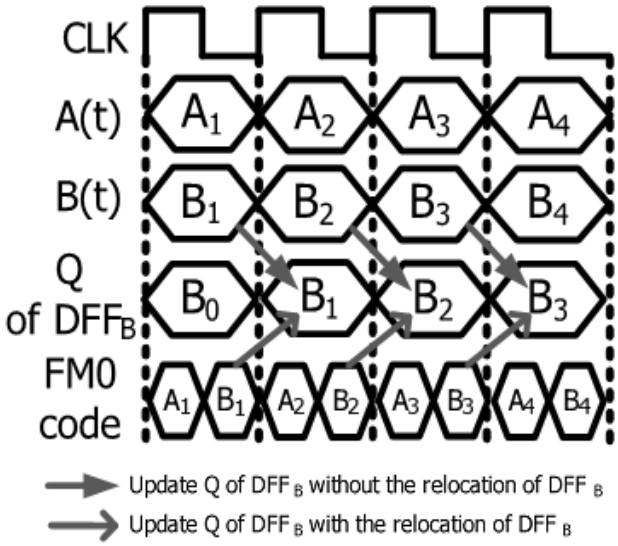


Fig 4.2 Timing diagram of FM0 encoder with area compact retiming

4.2 Balance Logic-Operation Sharing

As mentioned previously, the Manchester encoding can be derived from X ⊕ CLK, and it is also equivalent to

$$X \oplus CLK = X\,(CLK)' + X'\,CLK. \qquad (6)$$

The expression (6) can be realized using a multiplexer as shown in Fig.4.3(a). It is fairly related to the Boolean function of FM0 encoding in expression (4). From the expressions (4) & (5) the Manchester and FM0 logics have a multiplexer with control signal CLK. The topic of balance logic operation sharing is shown in Fig.4.3(b), in that to incorporate the X into A(t) and X into B(t).

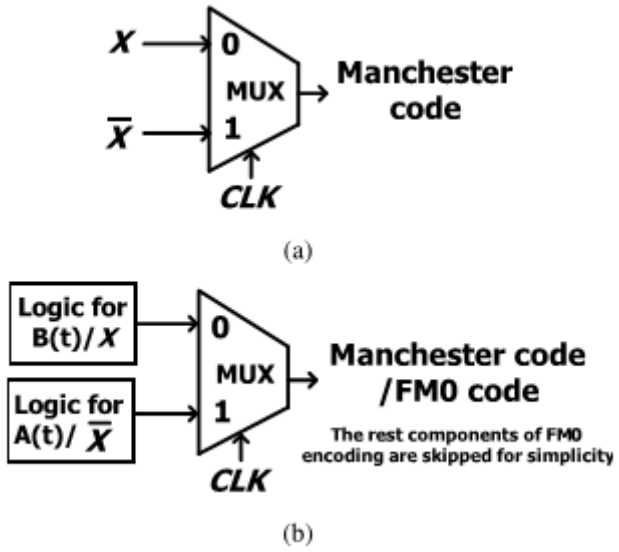


Fig. 4.3 Concept of balance logic-operation sharing for FM0 and Manchester encodings. (a) Manchester encoding in multiplexer. (b) Combines the logic-operations of Manchester and Fm0 encoding.

The fig.4.4 shows the logic for A(t)/ X . The inverter of B(t1) gives the A(t) and the inversion of X gives the X' . The logic for A(t)/ X' uses one inverter and a multiplexer is sited prior to the not gate to switch the operands of X and B(t-1). Based on mode signal to perform either Manchester or FM0 encoding.

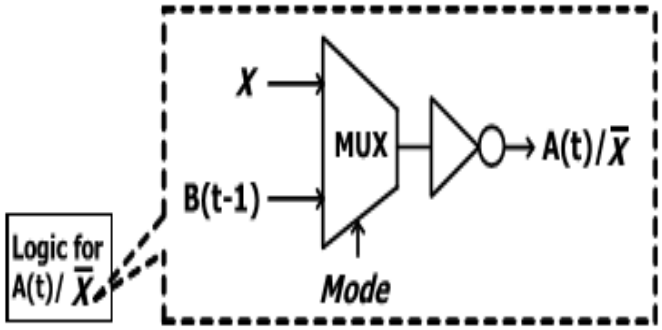
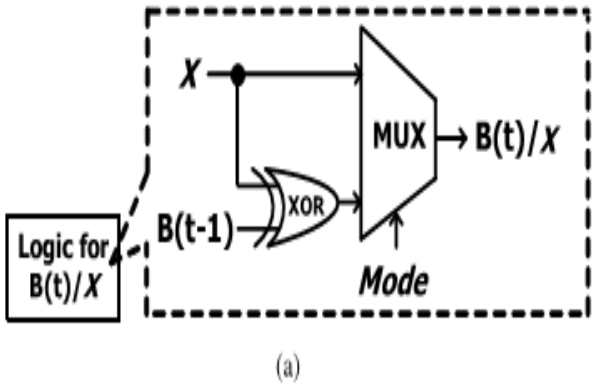


Fig. 4.4 Balance logic-operation sharing of A(t) and X'

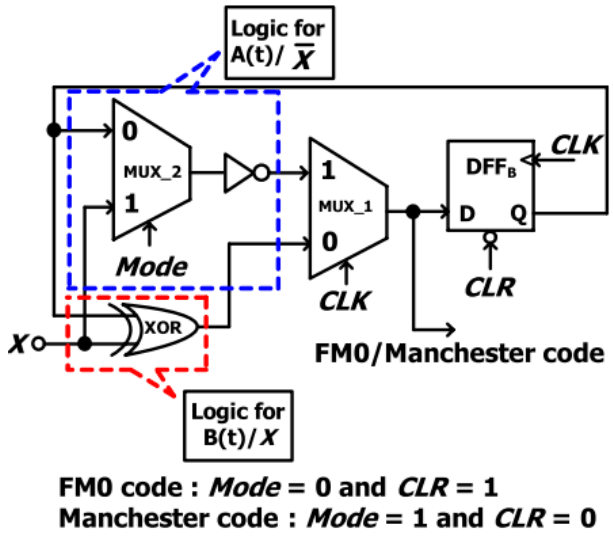
The logic for B(t)/X is also derived using a similar concept that can be shown in Fig.4.5(a). But the structure shown in Fig.4.5(a) has a drawback. The XOR gate is only used for FM0 encoding and not used for Manchester encoding. This results in the HUR of this architecture decreasing. The X can be interpreted as X ⊕ 0 and this XOR operation can be used for both FM0 and Manchester encodings. As a result, the Fig.4.4(b) shows the logic for B(t)/X . A multiplexer is used to switch the operands of B(t−1) and logic−0. Using the area compact retiming technique the multiplexer in Fig.4.5(b) is replaced with DFFB as shown in Fig.5.4(c). The CLR is the clear signal used to reset the DFFB content to logic−0. The Manchester encoding is performed by activating the CLR, this can set DFFB content to zero. The FM0 encoding is performed by disabling the CLR and DFFB gives the B(t−1).



(a)

Fig 4.5. Balance logic-operation sharing of B(t) and X. (a) Without XOR sharing. (b) With XOR sharing. (c) Sharing of the reused DFFB from area-compact retiming technique.

The Modified architecture of Manchester/FM0 encoder with SOLS technique is shown in Fig.4.6.



4.6 Modified structure of FM0/Manchester encoder with unbalance computation time between B(t)/X and A(t)/ *X*'

In Fig.4.6 the computation time of MUX_2 is the same as the computation time of XOR operation. An inverter is placed after the MUX_2, this causes unbalance computation time between B(t)/X and A(t)/X' that results in a glitch to MUX_1. To eliminate the unbalance computation time by using XNOR gate with an inverter in place of XOR in the logic for B(t)/X and an inverter is also used in logic for A(t)/ X' . So, an inverter is placed after the MUX_1 output. This balances the computation time between A(t)/X' and B(t)/X. The architecture of FM0/Manchester encoder using SOLS scheme with balance computation time between B(t)/X and A(t)/ *X*' is shown in Fig.4.7.

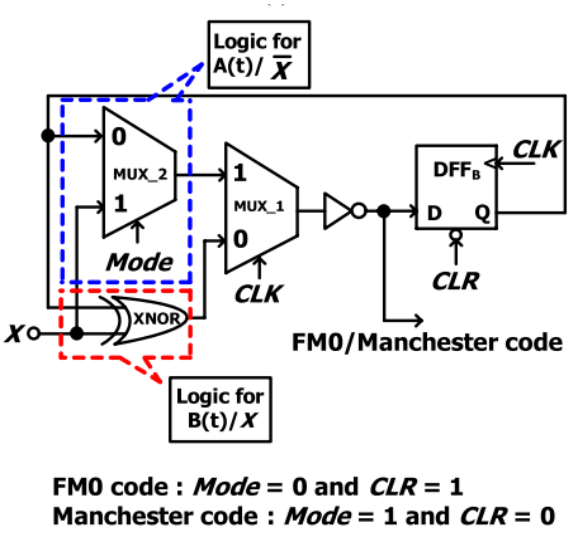


Fig.4.7 Modified structure of FM0/Manchester encoder with balance computation time between B(t)/X and A(t)/ *X*'

CHAPTER 5: PROPOSED METHOD

5.1 Flow chart of Proposed Method

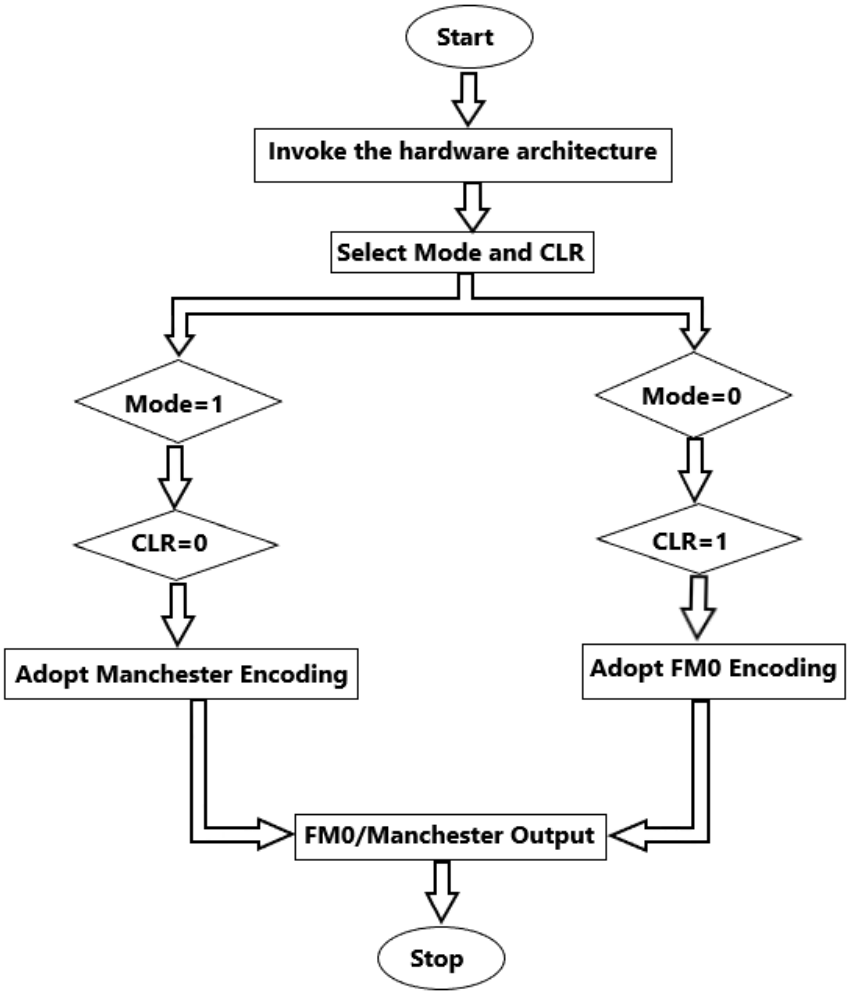


Fig 5.1 Flow chart of proposed method

5.2 Proposed Method

The SOLS technique integrates Manchester and FM0 encodings into fully reused hardware architecture. The coding procedure of FM0 is more complex than that of Manchester. Our work targets an efficient integration of hardware devices for Manchester encoding and FM0 encoding.

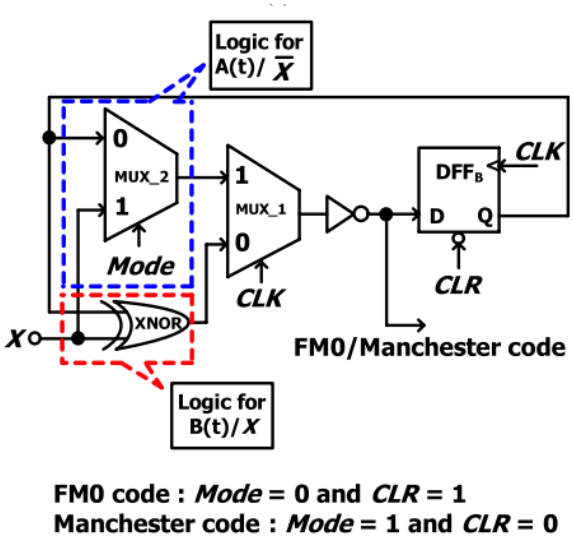


Fig 5.2 Modified structure of FM0/Manchester encoder with balance computation time between B(t)/X and A(t)/ *X*'

We have obtained the integrated architecture of both encodings using SOLS technique.The adoption of FM0 or Manchester code depends on Mode and CLR. In addition, the CLR further has another individual function of a hardware initialization. If the CLR is simply derived by inverting mode without assigning an individual CLR control signal, this leads to a conflict between the coding mode selection and the hardware initialization. To avoid this conflict, both Mode and CLR are assumed to be separately allocated to this design from a system controller. Whether FM0 or Manchester code is adopted, no logic component of the proposed VLSI architecture is wasted. Every component is active in both FM0 and Manchester encodings. Therefore, the HUR of the proposed VLSI architecture is greatly improved. It not only develops a fully reused VLSI architecture, but also has the capability to exhibit a reasonable performance compared with the existing works. The encoding capability can fully support the DSRC standards.

The techniques of Area-compact retiming are used to decrease the hardware problem like number of transistors and by the other techniques of balance logic operations sharing the helps for

recognizing the logic components which may efficiently combine with the FM0 and Manchester encoding techniques. Each section activates the FM0 and Manchester encoding techniques. It will considerably recover the utilization of hardware at the rate of 100% and it also reduces the consumption of power. Both the FM0 and Manchester encoding techniques are designed to attain the high speed and fully reconfigured VLSI design architecture for the system application.

Coding	Active components / Total components	HUR
FM0	5/5	100%
Manchester	5/5	100%
Average	5/5	100%

Table V : HUR of Manchester/FM0 encoder architecture

CHAPTER 6: OVERVIEW OF HARDWARE/SOFTWARE TOOLS USED

6.1 Introduction to VLSI

Very-large-scale integration (VLSI) is the process of creating integrated circuits by combining thousands of transistor-based circuits into a single chip. VLSI began in the 1970s when complex semiconductor and communication technologies were being developed. The microprocessor is a VLSI device. The term is no longer as common as it once was, as chips have increased in complexity into the hundreds of millions of transistors.

The first semiconductor chips held one transistor each. Subsequent advances added more and more transistors and, as a consequence, more individual functions or systems were integrated over time. The first integrated circuits held only a few devices, perhaps as many as ten diodes, transistors, resistors and capacitors, making it possible to fabricate one or more logic gates on a single device. Now known retrospectively as "small-scale integration" (SSI), improvements in technique led to devices with hundreds of logic gates, known as large-scale integration (LSI), i.e. systems with at least a thousand logic gates. Current technology has moved far past this mark and today's microprocessors have many millions of gates and hundreds of millions of individual transistors.

At one time, there was an effort to name and calibrate various levels of large-scale integration above VLSI. Terms like Ultra-large-scale Integration (ULSI) were used. But the huge number of gates and transistors available on common devices has rendered such fine distinctions moot. Terms suggesting greater than VLSI levels of integration are no longer in widespread use. Even VLSI is now somewhat quaint, given the common assumption that all microprocessors are VLSI or better.

Understanding why integrated circuit technology has such a profound influence on the design of digital systems requires understanding both the technology of IC manufacturing and the economics of ICs and digital systems.

6.2 Advantages of ICs over discrete components

While we will concentrate on integrated circuits, the properties of integrated circuits-what we can and cannot efficiently put in an integrated circuit-largely determine the architecture of the entire system. Integrated circuits improve system characteristics in several critical ways. ICs have

three key advantages over digital circuits built from discrete components:

- Size: Integrated circuits are much smaller-both transistors and wires are shrunk to micrometer sizes, compared to the millimeter or centimeter scales of discrete components. Small size leads to advantages in speed and power consumption, since smaller components have smaller parasitic resistances, capacitances, and inductances.
- Speed: Signals can be switched between logic 0 and logic 1 much quicker within a chip than they can between chips. Communication within a chip can occur hundreds of times faster than communication between chips on a printed circuit board. The high speed of circuits on-chip is due to their small size-smaller components and wires have smaller parasitic capacitances to slow down the signal.

- Power consumption: Logic operations within a chip also take much less power. Once again, lower power consumption is largely due to the small size of circuits on the chip-smaller parasitic capacitances and resistances require less power to drive them.

These advantages of integrated circuits translate into advantages at the system level:

- Smaller physical size: Smallness is often an advantage in itself-consider portable televisions or handheld cellular telephones.

- Lower power consumption: Replacing a handful of standard parts with a single chip reduces total power consumption. Reducing power consumption has a ripple effect on the rest of the system: a smaller, cheaper power supply can be used; since less power consumption means less heat, a fan may no longer be necessary; a simpler cabinet with less shielding for electromagnetic shielding may be feasible, too.

- Reduced cost: Reducing the number of components, the power supply requirements, cabinet costs, and so on, will inevitably reduce system cost. The ripple effect of integration is such that the cost of a system built from custom ICs can be less, even though the individual ICs cost more than the standard parts they replace.

Understanding why integrated circuit technology has such a profound influence on the design of digital systems requires understanding both the technology of IC manufacturing and the economics of ICs and digital systems.

6.3 Applications of VLSI

Electronic systems now perform a wide variety of tasks in daily life. Electronic systems in some cases have replaced mechanisms that operated mechanically, hydraulically, or by other means; electronics are usually smaller, more flexible, and easier to service. In other cases

electronic systems have created totally new applications. Electronic systems perform a variety of tasks, some of them visible, some more hidden:

- Personal entertainment systems such as portable MP3 players and DVD players perform sophisticated algorithms with remarkably little energy.

- Electronic systems in cars operate stereo systems and displays; they also control fuel injection systems, adjust suspensions to varying terrain, and perform the control functions required for anti-lock braking (ABS) systems.

- Digital electronics compress and decompress video, even at high-definition data rates, on-the-fly in consumer electronics.

- Low-cost terminals for Web browsing still require sophisticated electronics, despite their dedicated function.

- Personal computers and workstations provide word-processing, financial analysis, and games. Computers include both central processing units (CPUs) and special-purpose hardware for disk access, faster screen display, etc.

- Medical electronic systems measure bodily functions and perform complex processing algorithms to warn about unusual conditions. The availability of these complex systems, far from overwhelming consumers, only creates demand for even more complex systems.

The growing sophistication of applications continually pushes the design and manufacturing of integrated circuits and electronic systems to new levels of complexity. And perhaps the most amazing characteristic of this collection of systems is its variety-as systems become more complex, we build not a few general-purpose computers but an ever wider range of special-purpose systems. Our ability to do so is a testament to our growing mastery of both integrated circuit manufacturing and design, but the increasing demands of customers continue to test the limits of design and manufacturing

6.4 Software and Language used

Verilog is a Hardware Description Language (HDL). It is a language used for describing a digital system like a network switch or a microprocessor or a memory or a flip-flop. It means, by using a HDL we can describe any digital hardware at any level. Designs, which are described in HDL are independent of technology, very easy for

designing and debugging, and are normally more useful than schematics, particularly for large circuits. Verilog HDL modeling language supports three kinds of modeling styles: gate-level, dataflow, and behavioral. The behavioral modeling is used for both combinatorial and sequential circuits.The model which has been used is Behavioural Model.

Behavioral models in Verilog contain procedural statements, which control the simulation and manipulate variables of the data types. These all statements are contained within the procedures. Each of the procedures has an activity flow associated with it.The behavioral modeling is used for both combinatorial and sequential circuits.

The family which has been used is Spartan3E Family because of following reasons:

- Logic is Optimized
- For applications where logic densities matter more than I/O count
- Ideal for logic integration, DSP co-processing and embedded control, requiring significant processing and narrow or few interfaces

CHAPTER 7: RESULTS

RTL Schematic

RTL is an acronym for register transfer level. This implies that your Verilog code describes how data is transformed as it is passed from register to register. The transforming of the data is performed by the combinational logic that exists between the registers. Viewing an RTL schematic opens an NRG file that can be viewed as a gate-level schematic. This schematic is generated after the HDL synthesis phase of the synthesis process. It shows a representation of the pre-optimized design in terms of generic symbols, such as adders, multipliers, counters, AND gates, and OR gates, that are independent of the targeted Xilinx device. There are more components in the existing system. The RTL view of the existing system shows the following components: two XOR gates, two D flip flops, two multiplexers and inverter.

The RTL view of the proposed system shows the following components: one XOR gate, one D flip flop, two multiplexers and two inverters. This clearly represents that there are lesser components in the proposed system than the existing system.

Technology Schematic

Viewing a Technology schematic opens an NGC file that can be viewed as an architecture-specific schematic. This schematic is generated after the optimization and technology targeting phase of the synthesis process. It shows a representation of the design in terms of logic elements optimized to the target Xilinx device or "technology"; for example, in terms of LUTs, carry logic, I/O buffers, and other technology-specific components. Viewing this schematic allows you to see a technology-level representation of your HDL optimized for a specific Xilinx architecture, which might help you discover design issues early in the design process. In the existing system , there are three LUTs. In the proposed system, the number of luts are reduced. Hence the area is reduced.

7.1 RTL Schematic Results:

7.1.1 Existing method :

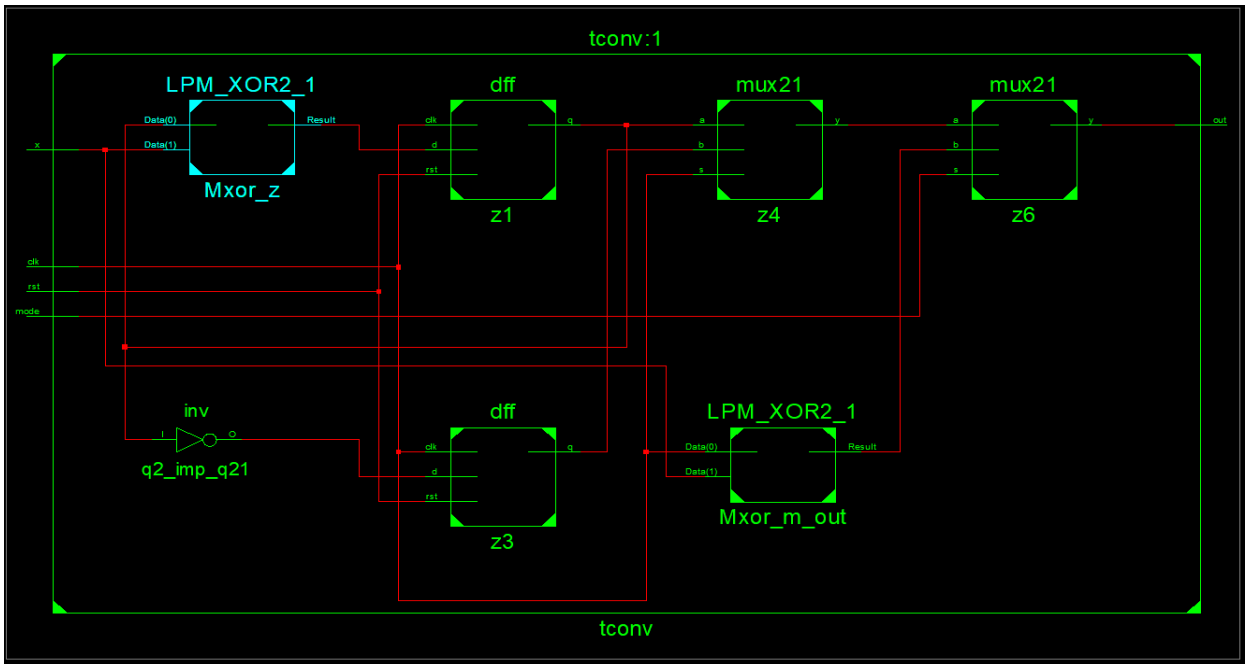


Fig. 7.1 RTL schematic of existing method

The hardware architecture of the existing system shows the following components: two XOR gates, two D flip flops, two multiplexers and inverter.The input X is given to XOR(Mxor_z) gate. In fm0 code the DFFA(z1) and DFFB(z3) are used to store the state code of the fm0 code and also MUX_1(z4) and NOT(q2_imp_q21) gate is used in the fm0 code.The adoption of FM0 or Manchester code depends on Mode and CLR.

When the mode=0, the fm0 code is selected. The Manchester encoder is as simple as a XOR operation for *X* and CLK. The MUX –1 is to switch outputs of D flip flops i.e A(t) (Output of DFFA) and B(t) (Output of DFFB) through the selection of CLK signal. When CLK= 1, A(t) is selected and when CLK=0, B(t) is selected. The determination of which coding is adopted depends on the Mode selection of the MUX–2(z6), where the Mode = 0 is for FM0 code, and the Mode = 1 is for Manchester code.

7.1.2 Proposed method :

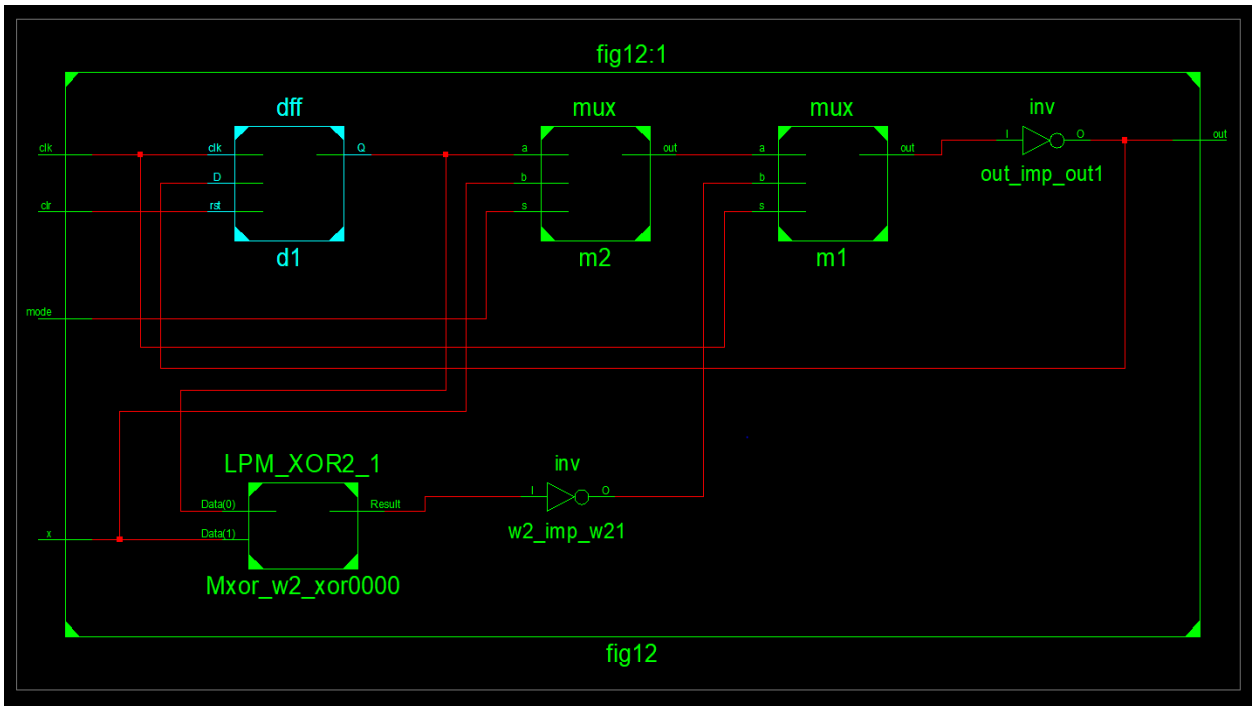


Fig.7.2. RTL schematic of proposed method

The hardware architecture of the proposed method shows the following components: one XOR gate, one D flip flop, two multiplexers and two inverters. For FM0, the state code of each state is stored into DFFA and

DFFB. The transition of state code only depends on B(t − 1) instead of both A(t − 1) and B(t − 1). Thus, the FM0 encoding just requires a single 1−bit flip−flop to store the B(t− 1). To avoid non synchronization between A(t) and B(t), the DFFB(d1) is relocated right after the MUX−1(m1), The FM0 code is alternatively switched between A(t) and B(t) through the MUX−1 by the control signal of the CLK. The Q of DFFB is directly updated from the logic of B(t) with 1−cycle latency. When the CLK is logic−0, the B(t) is passed through MUX−1 to the D of DFFB. Then, the upcoming positive−edge of CLK updates it to the Q of DFFB

7.2 Technology Schematic Results:

7.2.1 Existing method :

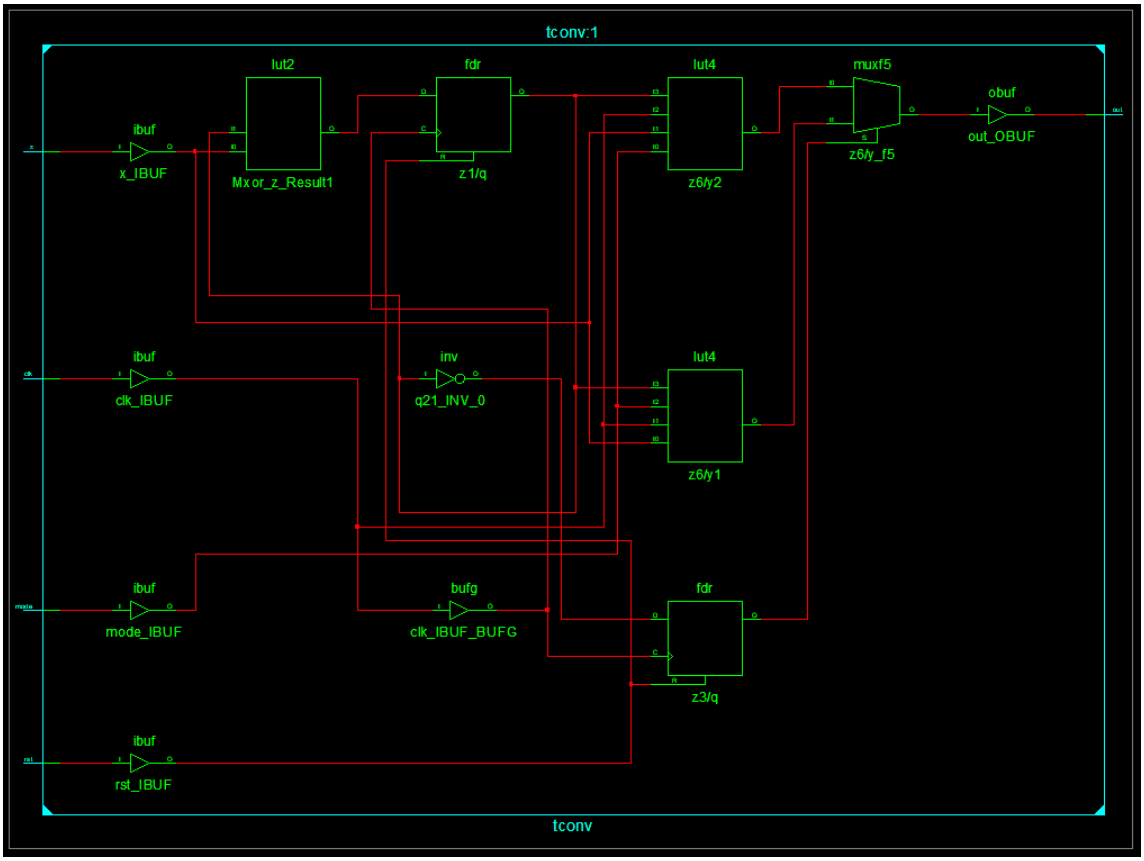


Fig.7.3. Technology schematic of existing method

7.2.2 Proposed method :

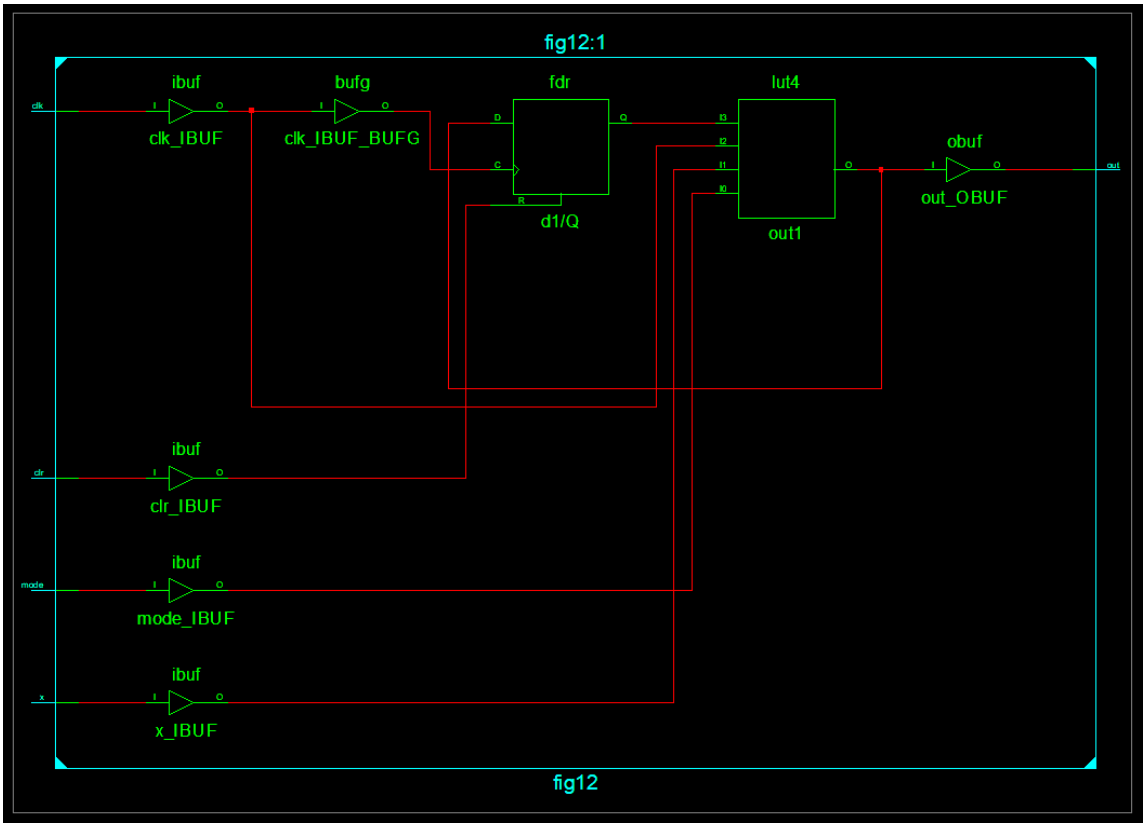


Fig. 7.4. Technology view of proposed method

Viewing this schematic allows you to see a technology-level representation of the HDL optimized for a specific Xilinx architecture, which might help you discover design issues early in the design process. The number of components in the existing method are more when compared to the number of components in the proposed method. In the proposed method, the number of luts are reduced which reduces the area.

7.3 Behavioral Simulation

7.3.1 Existing method :

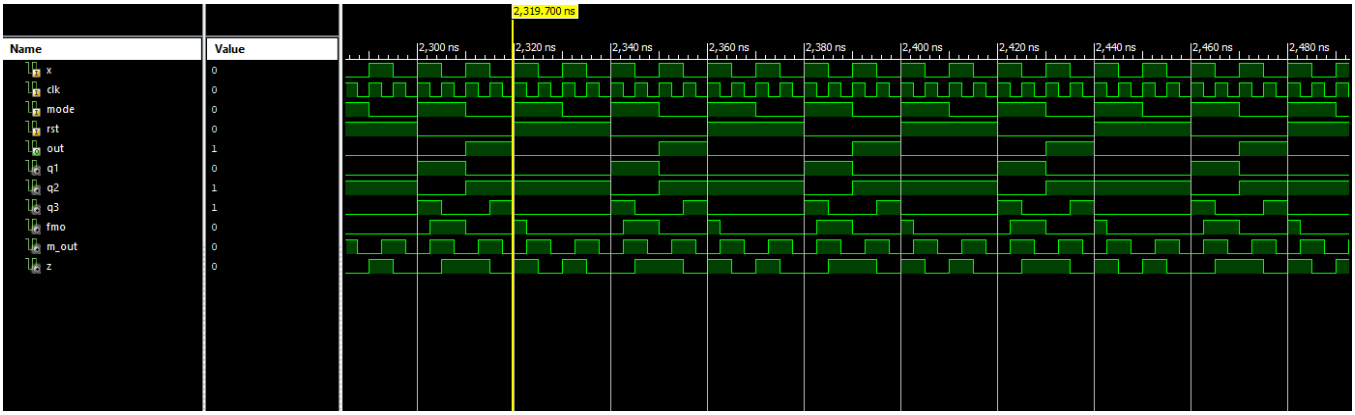


Fig.7.5. Behavioral simulation of existing method

7.3.2 Proposed method :

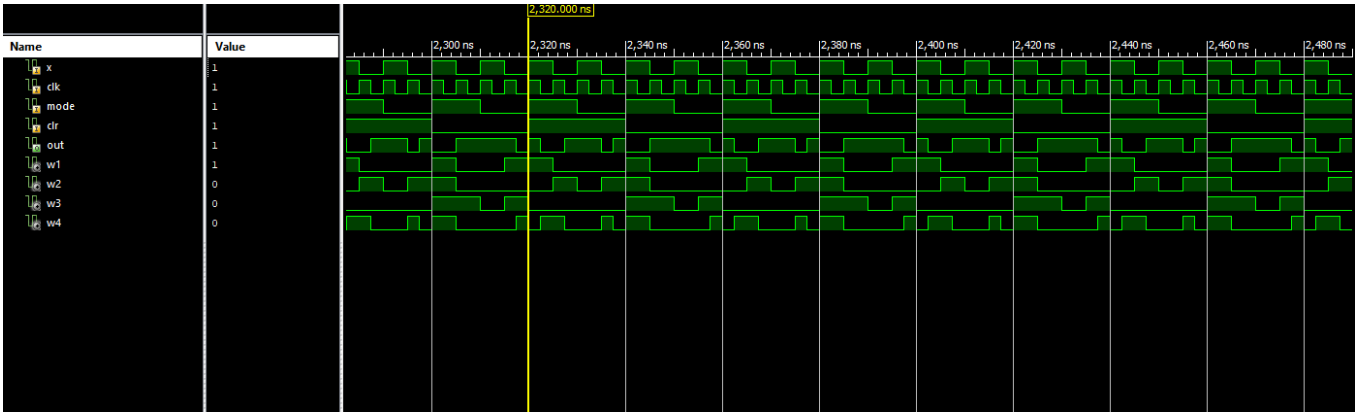


Fig.7.6. Behavioral simulation of proposed method

The coding-diversity between the FM0 and Manchester codes seriously limits the potential to design a fully reused VLSI architecture in existing method. We have obtained fully reused VLSI architecture using SOLS technique. The outputs of both the methods are the same, but by using the sols technique, the internal architecture of the proposed method is modified thereby making it more efficient. By using SOLS technique, the number of the components in the architecture are reduced. The two encoding techniques are integrated considering the similarities between them. By adopting this technique, we reduced the overall area which reduces the time taken by the architecture to perform the required task.

7.4 Device utilization summary :

7.4.1 Existing method :

Logic Utilization	Used	Available	Utilization
Number of Slices	2	960	0%
Number of Slice Flip Flops	2	1920	0%
Number of 4 input LUTs	3	1920	0%
Number of IOs	5		
Number of bonded IOBs	5	66	7%
Number of GCLKs	1	24	4%

Table VI: Utilization summary of Existing method

7.4.2 Proposed method :

Logic Utilization	Used	Available	Utilization
Number of Slices	1	960	0%
Number of Slice Flip Flops	1	1920	0%
Number of 4 input LUTs	1	1920	0%
Number of IOs	5		
Number of bonded IOBs	5	66	7%
Number of GCLKs	1	24	4%

Table VII: Utilization summary of Proposed method

From the above tables we can observe that, number of Slices, number of Slice Flip Flops and number of 4 input LUTs in the existing method are 2,2 and 3 respectively and in the proposed method they are 1,1 and 1. The number of components used in the proposed method are less compared to existing method and area is decreased.

7.5 Timing Summary :

7.5.1 Existed method :

Speed Grade: -4

Minimum period: 2.554ns (Maximum Frequency: 391.543MHz)

Minimum input arrival time before clock: 2.825ns

Maximum output required time after clock: 5.597ns

Maximum combinational path delay: 6.236ns

7.5.2 Proposed method :

Speed Grade: -5

Minimum period: 1.754ns (Maximum Frequency: 570.125MHz)

Minimum input arrival time before clock: 2.495ns

Maximum output required time after clock: 5.035ns

Maximum combinational path delay: 5.776ns

In Xilinx FPGAs, higher speed grades are faster. In the proposed method speed grade is -5 and in the existing method speed grade is -4 ,so we can infer that proposed method is faster compared to existing method. Similarly by comparing other time constraints we can observe that the proposed method requires less time to respond for a given signal.

CHAPTER 8: CONCLUSION AND FUTURE SCOPE

The coding-diversity between FM0 and Manchester encodings cause the limitations on

hardware utilization of VLSI architecture design. A limitation analysis on hardware utilization of FM0 and Manchester encodings is discussed in detail. In this report, the fully reused VLSI architecture using SOLS technique for both FM0 and Manchester encodings is proposed. The SOLS technique eliminates the limitation on hardware utilization by two core techniques: Area compact retiming and Balance logic-operation sharing. The balance logic-operation sharing efficiently combines FM0 and Manchester encodings with the identical logic components.

The encoding capability of this paper can fully support the DSRC standards of America, Europe, and Japan. This report not only develops a fully reused VLSI architecture, but also exhibits a competitive performance compared with the existing works.

In DSRC signals, consistency and DC-Stability are the requirements of the design which help to fulfil the demands of both the FM0 and Manchester encoding techniques. Manchester and FM0 coding are very popular codes, as these codes are level insensitive, self-clocking they also provide signal absence detection and have the encoding clock rate embedded within the transmitted data. Both the FM0 and Manchester encoding techniques are widely used to contribute to the short range communication. Reliability of signal is attained by DSRC for adopting both the FM0 and Manchester encoding. Besides, the logic delay and memory usage of the system also get reduced. FM0 and Manchester encoding architectures are combined together to form efficient compact architecture through SOLS. The future scope of design can be used to implement this by using high performance FPGA devices.

REFERENCES

[1] Yu-Hsuan Lee and Cheng-Wei-Pan ”Fully reused Vlsi architecture of FM0/Manchester encoding using sols technique for dsrc applications” in IEEE transactions on Very Large Scale Integration (VLSI) systems VOL 23, No.1, January 2015.

[2] J. B. Kenney, “Dedicated short-range communications (DSRC) standards in the United States,” Proc. IEEE, vol. 99, no. 7, pp. 1162–1182, Jul. 2011.

[3] M. A. Khan, M. Sharma, and P. R. Brahmanandha, “FSM based Manchester encoder for UHF RFID tag emulator,” in Proc. Int. Conf. Comput., Commun. Netw., Dec. 2008, pp. 1–6.

[4] M. A. Khan, M. Sharma, and P. R. Brahmanandha, “FSM based FM0 and Miller encoder for UHF RFID tag emulator,” in Proc. IEEE Adv. Comput. Conf., Mar. 2009, pp. 1317–1322.