Design of FM0/Manchester Encoding Using Sols Technique for Fully Reused VLSI Architecture

Neethu Susan Saji¹, Lijesh L²
M.Tech Scholar in VLSI and Embedded Systems¹, Associate Professor²
Department of ECE
Musaliar College of Engineering and Technology, Kerala, India

Abstract:
To promote intelligent and smart transportation services into our daily life the dedicated short range communication (DSRC) is an advanced technique. Its main purpose is to implement applications that reduce accidents and improve traffic flow. The vehicle safety issues demand not only the immediate connection but also higher signal reliability for delivering message correctly. The transmitted signal consists of arbitrary binary sequence, which is difficult to obtain dc-balance and hence not reliable. The DSRC standards generally adopt FM0 and Manchester codes to reach dc-balance, enhancing the signal reliability. Nevertheless, the coding-diversity between the FM0 and Manchester codes seriously limits the potential to design a fully reused VLSI architecture for both. In this paper, the similarity-oriented logic simplification (SOLS) technique is proposed to overcome this limitation. The SOLS technique improves the hardware utilization rate from 57.14% to 100% for both FM0 and Manchester encodings. This paper not only develops a fully reused VLSI architecture, but also exhibits an efficient performance compared with the existing works.

Index terms: DSRC-Dedicated short-range communication, FM0, Manchester.

I. INTRODUCTION
Now-a-days the use of automobiles for transportation has become crucial into our daily life, at the same time people face several problems due to traffic grids, accidents and vehicle damages. The dedicated short-range communication (DSRC) is the only solution to promote smart transportation services. The DSRC is a protocol for one- or two-way medium range communication especially for intelligent transportation systems. The DSRC can be briefly classified into two categories: automobile-to-automobile and automobile-to-road side. In automobile-to-automobile, the DSRC enables the message sending and broadcasting among automobiles for safety issues and public information announcement. The safety issues include blind-spot, intersection warning, inter cars distance, and collision-alarm. The automobile-to-roadside focuses on the intelligent transportation service, such as electronic toll collection (ETC) system. With ETC, the toll collecting is electrically accomplished with the contactless IC-card platform. Moreover, the ETC can be extended to the payment for parking-service, and gas-refueling. Thus, the DSRC system plays an important role in modern automobile industry. This paper proposes a VLSI architecture design using similarity-oriented logic simplification (SOLS) technique. The SOLS consists of two core methods: area-compact retiming and balance logic-operation sharing. With SOLS technique, this paper constructs a fully reused VLSI architecture of Manchester and FM0 encodings for DSRC applications.

II. CODING PRINCIPLES OF FM0 CODE AND MANCHESTER CODE
2.1 FM0 ENCODING
As shown in Figure 1, for each input signal X, the FM0 code consists of two parts: one for former-half cycle of CLK, A, and the other one for later-half cycle of CLK, B.

The coding principle of FM0 is listed as the following three rules.
1) If X is the logic-0, the FM0 code must exhibit a transition between A and B.
2) If X is the logic-1, no transition is allowed between A and B.
3) The transition is allocated among each FM0 code no matter what the X is.

Figure 1. Codeword structure of FM0
A FM0 coding example is shown in Figure. 2. At cycle 1, the X is logic-0; therefore, a transition occurs on its FM0 code, according to rule 1. For simplicity, this transition is initially set from logic-0 to -1. According to rule 3, a transition is allowed among each FM0 code, and thereby the logic-1 is changed to logic-0 in the beginning of cycle 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.

Figure 2: Illustration of FM0 coding example.
2.1 MANCHESTER ENCODING
The Manchester coding example is shown in Figure 3. The Manchester code is derived from
\[
X \oplus \text{CLK} = \begin{cases} 1 & (1) \\
0 & (2)
\end{cases}
\]
Figure 3: Illustration of Manchester coding example.

III. LIMITATION ANALYSIS ON HARDWARE UTILIZATION OF FM0 ENCODER AND MANCHESTER ENCODER
To make an analysis on hardware utilization of FM0 and Manchester encoders, the hardware architectures of both are conducted first. As mentioned earlier, the hardware architecture of Manchester encoding is as simple as a XOR operation. However, the conduction of hardware architecture for FM0 is not as simple as that of Manchester. How to construct the hardware architecture of FM0 encoding should start with the FSM of FM0 first. The FSM of FM0 code is classified into four states. A state code is individually assigned to each state, and each state code consists of A and B, as shown in Figure 1. According to the coding principle of FM0, the FSM of FM0 is shown in Figure 4.

Figure 4. Illustration of FSM for FM0.

Suppose the initial state is S1, and its state code is 11 for A and B, respectively. If the X is logic-0, the state-transition must follow both rules 1 and 3. The only one next-state that can satisfy both rules for the X of logic-0 is S3. If the X is logic-1, the state-transition must follow both rules 2 and 3. The only one next-state that can satisfy both rules for the X of logic-1 is S4. Thus, the state-transition of each state can be completely constructed. The FSM of FM0 can also conduct the transition table of each state, as shown in Table I.

Table I. Transition table of FM0.

<table>
<thead>
<tr>
<th>Previous-state</th>
<th>Current-state</th>
</tr>
</thead>
<tbody>
<tr>
<td>A(t − 1)</td>
<td>B(t − 1)</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

A(t) and B(t) represent the discrete-time state code of 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
\[
A(t) = B(t − 1) \\
B(t) = X \oplus B(t − 1)
\]
(2) (3)

With both A(t) and B(t), the Boolean function of FM0 code is denoted as
\[
\text{CLK} \ A(t) + \text{CLK} \ B(t)
\]
(4)

With (1) and (4), the hardware architectures of FM0 and Manchester encoders are shown in Figure 5. The top part is the hardware architecture of FM0 encoder, and the bottom part is the hardware architecture of Manchester encoder. The Manchester encoder is as simple as a XOR operation for X and CLK. Nevertheless, the FM0 encoding depends not only on the X but also on the previous-state of the FM0 code. The hardware architecture of FM0 encoding should start with the FSM of FM0. The DFFA and DFFB store the state code of the FM0 code. The MUX−1 is to switch A(t) and B(t) (discrete-time state code of current-state) through the selection of CLK signal. The determination of which coding is adopted depends on the Mode selection of MUX−2, where the Mode = 0 is for FM0 code, and the Mode = 1 is for Manchester code.

Figure 5. Hardware architecture of FM0 and Manchester encodings.

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 Hardware utilization rate (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.

IV. VLSI ARCHITECTURE DESIGN OF FM0 ENCODER AND MANCHESTER ENCODER USING SOLS TECHNIQUE
The SOLS technique is classified into two parts: area-compact retiming and balance logic-operation sharing.

4.1. AREA-COMPACT RETIMING
The FM0 logic in Figure 5 is simply shown in Figure 6(a). The logic for A(t) and the logic for B(t) are the Boolean functions to derive A(t) and B(t), where the X is omitted for a concise representation. For FM0, the state code of each state is stored into DFFA and DFFB. According to (2) and (3), 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). If the DFFA is directly removed, a non-synchronization between A(t) and B(t) causes the logic fault of FM0 code. To avoid this logic-fault, the DFFB is relocated right after the MUX−1, as shown in Figure 6(b), where the DFFB is assumed to be positive-edge triggered. At each cycle, the FM0 code, comprising A and B, is derived from the logic of A(t) and the logic of B(t), respectively. The FM0 code is alternatively switched between A(t) and B(t) through the MUX−1 by the control signal of the CLK.
4.2. BALANCE LOGIC-OPERATION SHARING

As mentioned previously, the Manchester encoding can be derived from \( X \oplus \text{CLK} \), and it is also equivalent to \( X \oplus \text{CLK} = X \cdot \text{CLK} + X \cdot \text{CLK} \). (6)

This can be realized by the multiplexer, as shown in Figure. 7(a). By comparing with (4) and (6), the FM0 and Manchester logics have a common point of the multiplexer like logic with the selection of CLK. As shown in Figure. 7(b), the concept of balance logic-operation sharing is to integrate the X into \( A(t) \) and X into \( B(t) \), respectively. The logic for \( A(t)/X \) is shown in Figure. 8. The \( A(t) \) can be derived from an inverter of \( B(t-1) \), and X is obtained by an inverter of X. The logic for \( A(t)/X \) can share the same inverter, and then a multiplexer is placed before the inverter to switch the operands of \( B(t-1) \) and X. The Mode indicates either FM0 or Manchester encoding is adopted.

The similar concept can be also applied to the logic for \( B(t)/X \), as shown in Figure. 9(a). Nevertheless, this architecture exhibits a drawback that the XOR is only dedicated for FM0 encoding, and is not shared with Manchester encoding. Therefore, the HUR of this architecture is certainly limited. The X can be also interpreted as the X \( \oplus 0 \), and thereby the XOR operation can be shared with Manchester and FM0 encodings. As a result, the logic for \( B(t)/X \) is shown in Figure. 9(b), where the multiplexer is responsible to switch the operands of \( B(t-1) \) and logic-0. This architecture shares the XOR for both \( B(t) \) and X, and thereby increases the HUR. Furthermore, the multiplexer in Figure. 9(b) can be functionally integrated into the relocated DFFB from area-compact retiming technique, as shown in Figure. 9(c). The CLR is the clear signal to reset the content of DFFB to logic-0. The DFFB can be set to zero by activating CLR for Manchester encoding. When the FM0 code is adopted, the CLR is disabled, and the \( B(t-1) \) can be derived from DFFB.

The proposed VLSI architecture of FM0 and Manchester encodings using SOLS technique is shown in Figure. 10(a). The logic for \( A(t)/X \) includes the MUX \( -2 \) and an inverter. Instead, the logic for \( B(t)/X \) just incorporates a XOR gate. In the logic for \( A(t)/X \), the computation time of MUX-2 is almost identical to that of XOR in the logic for \( B(t)/X \). However, the logic for \( A(t)/X \) further incorporates an inverter in the series of MUX-2. This unbalance computation time between \( A(t)/X \) and \( B(t)/X \) results in the glitch to MUX-1, possibly causing the logic-fault on coding. To alleviate this unbalance computation time, the architecture of the balance computation time between \( A(t)/X \) and \( B(t)/X \) is shown in Figure. 10(b). The XOR in
the logic for $B(t)/X$ is translated into the XNOR with an inverter, and then this inverter is shared with that of the logic for $A(t)/X$. This shared inverter is relocated backward to the output of MUX−1. Thus, the logic computation time between $A(t)/X$ and $B(t)/X$ is more balance to each other. The adoption of FM0 or Manchester code depends on Mode and CLR. 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.

V. SIMULATION RESULTS

VI. FUTURE WORK
Miller encoding can also be included in this technique along with FM0 and Manchester encoding. Miller encoding is the encoding of binary data to form a two-level signal where (a) a “0” causes no change of signal level unless it is followed by another “0” in which case a transition to the other level takes place at the end of the first bit period; and (b) a “1” causes a transition from one level to the other in the middle of the bit period. While using the Miller encoding, noise interference can be reduced.

VII. CONCLUSION
Without SOLS technique, the FM0 and Manchester encodings are performed on individual hardware architecture with a poor HUR of 57.14%. In this paper, the fully reused VLSI architecture using SOLS technique for both FM0 and Manchester encodings is proposed to improve the HUR to 100%. The SOLS technique eliminates the limitation on hardware utilization by two core techniques: area compact retiming and balance logic-operation sharing. The area-compact retiming relocates the hardware resource to reduce transistors. The balance logic-operation sharing efficiently combines FM0 and Manchester encodings with the identical logic components. Hence this work exhibits a competitive performance compared with the existing works.

VIII. REFERENCES


