Application Note

Binary Parity Generator and Checker

AN-CM-242

Abstract

This app note implements a Binary Parity Generator and Checker with two data input variants, a parallel data input and a serial data input. It describes the implemented logic, GreenPAKs implementation and the obtained results.

This application note comes complete with design files which can be found in the References section.

Contents

Figures

Tables

1 Terms and Definitions

IC	Integrated circuit
I/O	Input/output
MSB	Most significant bit
PCI	Peripheral component interconnect
SCSI	Small computer system interface
SPI	Serial peripheral interface
XOR gate	A digital logic gate that gives a true (1 or high) output when the number of true inputs is odd

2 References

For related documents and software, please visit:

GreenPAK™ Programmable Mixed-Signal Products | Renesas

Download our free GreenPAK™ Designer software [1] to open the .gp files [2] and view the proposed circuit design. Use the GreenPAK development tools [2] to freeze the design into your own customized IC in a matter of minutes. Renesas Electronics provides a complete library of application notes [3] featuring design examples as well as explanations of features and blocks within the IC.

[1] GreenPAK Designer Software, Software Download and User Guide, Renesas Electronics

[2] AN-CM-242 Binary Parity Generator and Checker.gp, GreenPAK Design File, Renesas Electronics

[3] GreenPAK Development Tools, GreenPAK Development Tools Webpage, Renesas Electronics

[4] GreenPAK Application Notes, GreenPAK Application Notes Webpage, Renesas Electronics

[5] SLG46536V, Datasheet, Renesas Electronics

[6] SLG46620V, Datasheet, Renesas Electronics

[7] AN-1120 Bluetooth-Controlled Car/Robot, Application Note, Renesas Electronics

3 Introduction

Binary serial transmissions are among the most widely used techniques for sharing information between devices by using wired or unwired transmissions. Within these transmissions, data errors are one of the most important problems that must be analyzed to obtain a reliable communication system.

The parity generating/checking method is one of the most widely used error detection techniques for data transmission; a parity bit is appended to the transmitted data to make the binary data's sum of 1s either even or odd. This bit is used to detect errors during the transmission of binary data.

The message containing the data bits, along with parity bit, is transmitted from transmitter node to receiver node. In the receiver node, the number of high bits in the message is counted. If this number doesn't match with the parity bit transmitted it means there is an error in the received data.

There are several different brands of commercial IC's (CD40101, 74HC/HCT280) that implement the parity generator/checker. Supplementing a GreenPAK design can positively affect the affordability, size, and modularity of the design. For example, the same general GreenPAK design can be used whether the intended I/O is active-high, active-low, or a mix. In this application note, the digital logic required to implement an integrated parity generator/checker managed by control signals is implemented. To do this, the app note implements two variants of parity checking. The first variant has a parallel input so that the data bits to be verified are loaded simultaneously. The second variant implements a serial input, loading the data with an asynchronous serial data transmission. To do this, a serial to parallel conversion is implemented within the GreenPAK.

To implement the parallel input binary parity generator and checker a SLG46536V is used. To implement the serial input variant a SLG46620V is used.

4 Digital Communications and Parity Bit

In digital communications, a parity bit is a bit added to a binary stream to ensure that the total number of 1-valued bits is even or odd. This technique is a simple and widely used method for detecting errors. There are two types of parity bit methods, called even parity bit and odd parity bit.

The odd parity bit system consists of counting the occurrences of bits whose value is 1 in the data stream. If the number is even, the parity bit value is set to 1, so the total count of occurrences of high bits in the entire stream including the parity bit is odd. If the count of high bits is odd, the parity bit value is 0. An example is shown in Figure 1.

Data Stream	Odd Parity
01100110	1

Figure 1: Odd Parity Binary Stream

The even parity bit method employs inverse logic. If the count of bits with a value of 1 is even in the data stream, the parity bit value is set to 0 making the total count of high bits in the entire stream including the parity an even number. If the count of bits with a value of 1 is odd, the parity bit is set to 1 so the entire stream has an even number of high bits.

Data Stream	Even Parity
01100110	0

Figure 2: System Diagram

To detect errors, a receiver must calculate the parity bit of the received binary data stream and compare it with the received parity bit. If parity bits are the same, an error is not detected. If they are different, an error is detected.

The parity bit is only useful for detecting errors. It cannot correct any errors, because it is not possible to determine which bit is incorrect within the stream. If a binary stream with errors is received, the receiver must discard it.

This makes the parity bit error method not suitable for high noise to signal ratio mediums, because a successful transmission can take a long time. The advantage of this method is that it only needs a single bit to detect errors, which can increase the number of transmissions within a period.

As an example, an odd parity bit transmitter transmits the previously analyzed stream. If a bit of the stream is changed, the receiver obtains a different parity bit if it is compared with the transmitted one. This effect is shown in Figure 3.

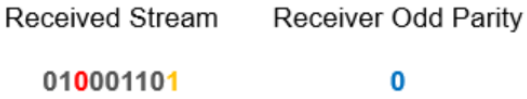

Figure 3: Odd Parity Check of an Erroneous Stream

The parity bit is used in applications where a simple error detector is needed and the transmission can be repeated if an error occurs. The most important application is in serial data transmission. It is based on a common format of 7 or 8 data bits, an even parity bit, and one or two stop bits.

Other applications of parity bits are SCSI buses, PCI buses, and many microprocessor instruction caches. Because the L-cache data is just a copy of main memory, it can be disregarded and re-fetched if it is found to be corrupted.

5 Logic Implementation

One of the main advantages of the parity bit for error detection is the simplicity of its calculation. To obtain even parity it is necessary to only perform the Modulo-2 sum, or XOR, of the data bits in the binary stream to obtain the parity bit.

Once the even parity is obtained, the odd parity can be obtained as the inverse of the even one.

As stated before, this application note implements two variants of the binary parity generator and checker. Both have an even output bit and an odd output bit, which is set to a high level if the corresponding parity is detected. Also, they have an enable input. If the enable is high, the parity is calculated. Otherwise, both parity outputs are set to a low level.

In the parallel variant, the generator or checker obtains the parity bit of a 9-length binary stream. With this length, it can be used by the generator as a processor of 9 bits or as a processor of more than 9 bits by only using the MSB (the 9th bit) as the cascaded input of another processor.

In Figure 4, the logic diagram of the 9-bit length parallel input parity checker is shown.

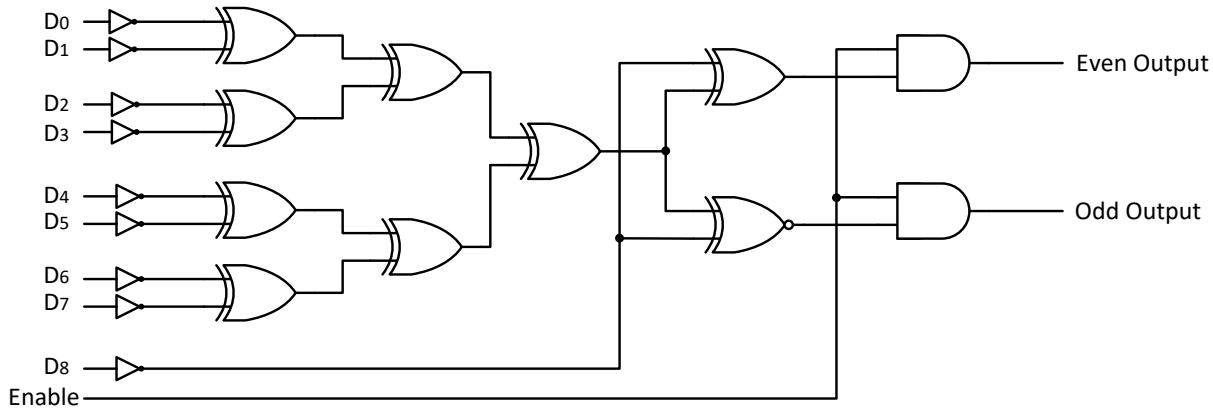

Figure 4: Parity Generator Logic Diagram

Table 1 shows a functional table of the parity generator and checker.

Table 1: Parity Generator Functional Table

D0-D8	Enable	Even Output	Odd Output
XXXX	0	0	0
Even Input	1	1	0
Odd Input	1	0	1

In the serial variant, the input stage includes a serial to parallel conversion, so the output of the converter is connected to the parity generator circuit. This scheme is shown in Figure 5.

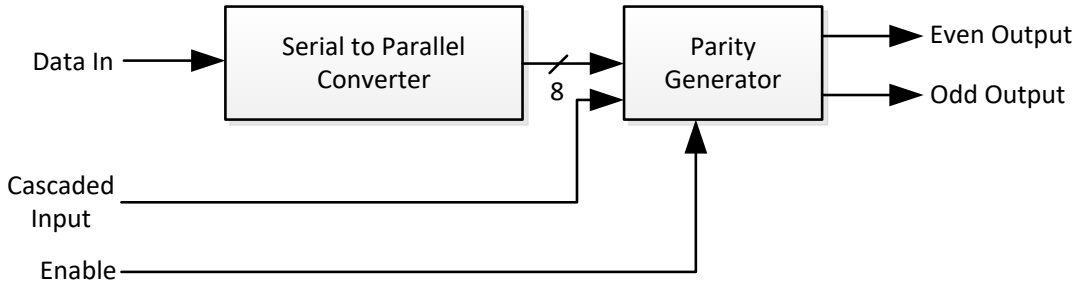

Figure 5: Serial Input Parity Generator Schematic Diagram

This variant also includes an additional cascaded input, so more bits can be processed by using several 8-bit parity checkers.

Serial to parallel data conversion is based on Renesas' AN-1120.

When there's no data on the serial input pin the serial bus is held high. When a byte is going to be sent, a logic low start bit is sent before the byte to indicate a transmission. After that, the eight data bits are sent and finally, a stop high-level bit is sent. This sequence can be seen in Figure 6.

Figure 6: Serial Data Frame

Within several GreenPAK ICs the SPI block can be used to implement the serial-to-parallel conversion. The serial communication must have a 9600 baud rate.

A falling edge detection is implemented to detect the start bit. When it is detected, a connection flag bit is set so two counters/delays are triggered. One of them, titled Bit Timer, is configured to have a period equal to the bit time duration (1/9600). The other counter, titled Frame delay, is configured to have a delay time equal to the 10-bit frame period (10/9600).

With these timers, the SPI block is connected so that the serial data input pin is connected to the MOSI input and the Bit Timer output is connected to CLK. The eight data bits are received by the SPI block.

Additional logic is used for controlling the clock signal, so when the frame period has elapsed, the SPI clock stops and the data is held on the register.

More details of the SPI to Parallel converter can be found in Renesas' AN-1120.

6 Implementation and Configuration

As described before, there are two variants of parity generator and checker, implemented with two different GreenPAKs.

The parallel input variant is implemented with SLG46536V.

To implement the bit inversion, as shown in Figure 4, 9 LUT's were used, configured as inverters. This can be seen in Figure 7.

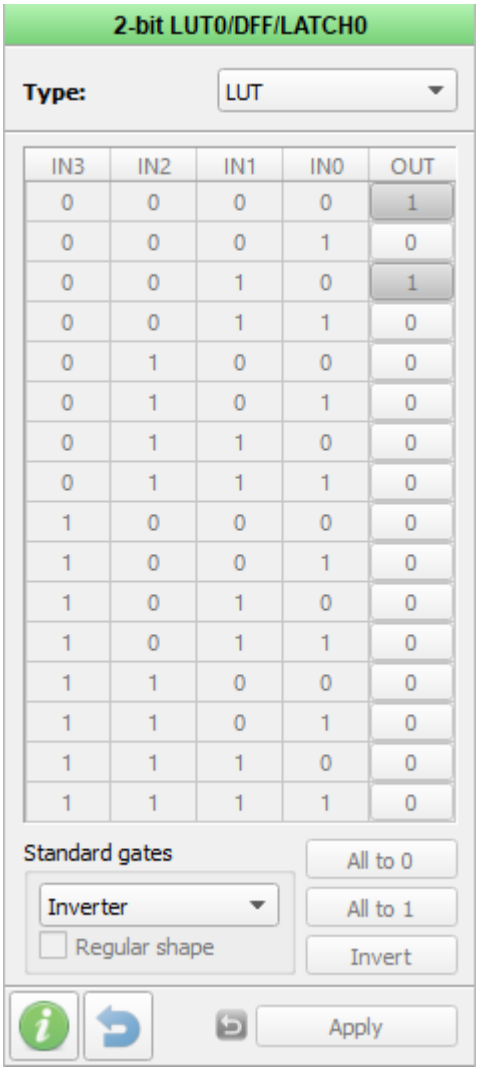

Figure 7: Bit Inverter

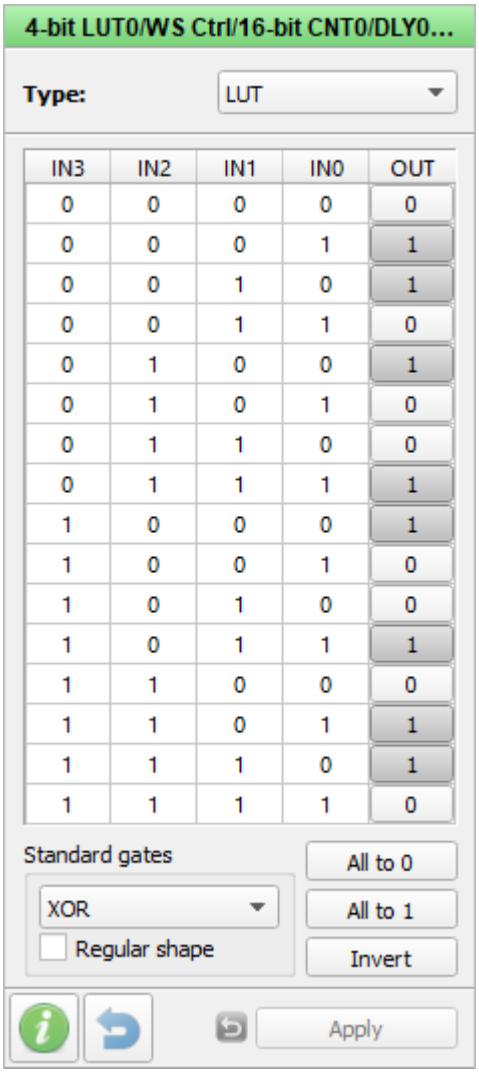

Figure 8: XOR Processor

The XOR is implemented to obtain the resulting bit for each nibble of data by using two 4-bit LUTs. They are configured as shown in Figure 8. As there aren't more 2-bit LUTs available, the XOR between the two nibbles is processed with the 3-bit LUT3 with the third input connected to GND.

To obtain the resulting bits of processing the 9-th bit input, 3-bit LUT11 and 3-bit LUT12 are used by connecting input 2 to ground. They are configured as shown in Figure 9 and Figure 10 to process the XOR and XNOR respectively.

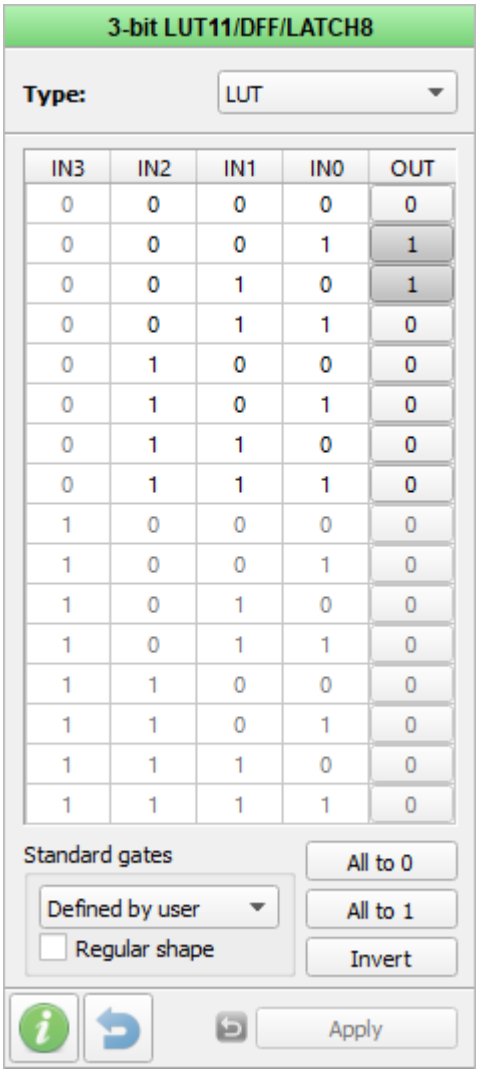

Figure 9: 9-th Bit XOR

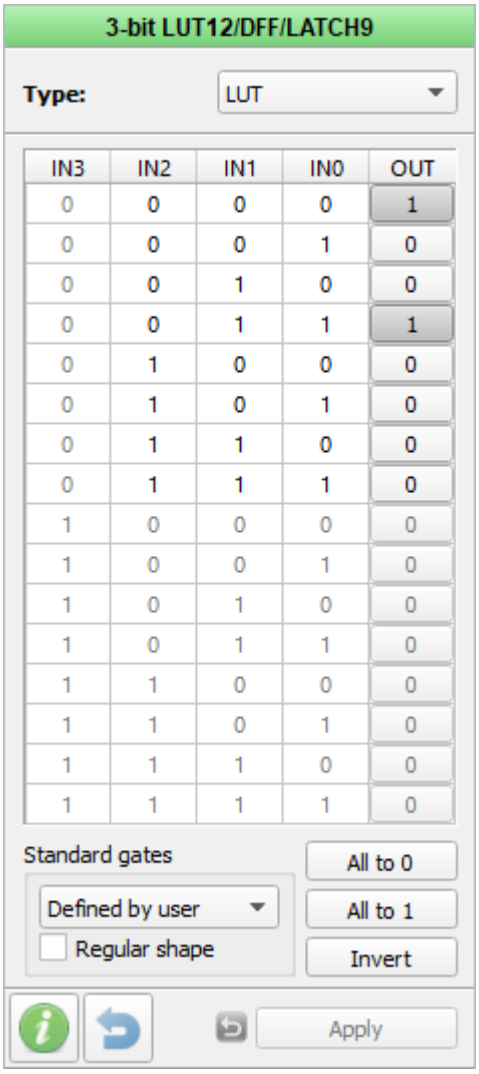

Figure 10: 9-th Bit XNOR

Finally, the even bit and odd bit are individually AND'ed to the enable bit and VDD using 3-bit LUT14 and 3-bit LUT15 respectively. These configurations can be seen in Figure 11 and Figure 12.

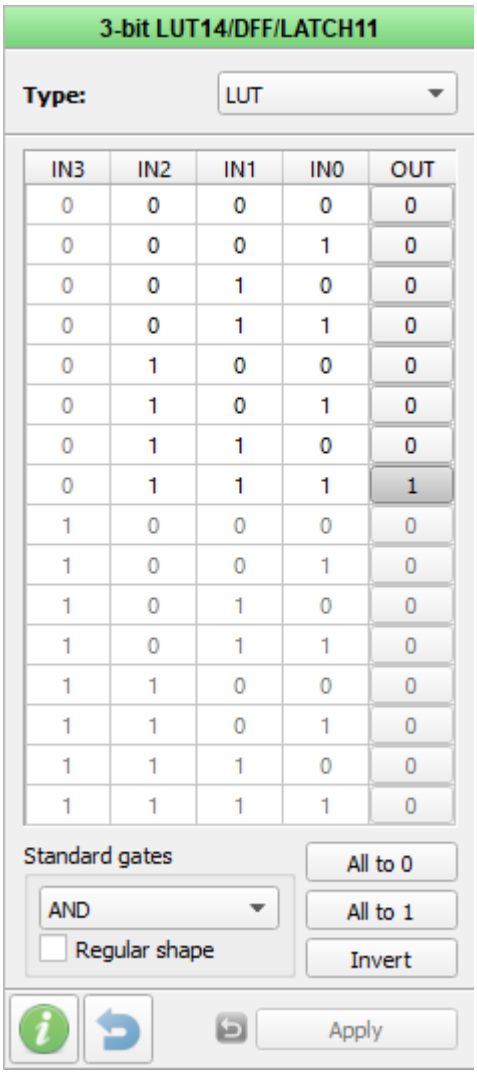

Figure 11: Even Output Control

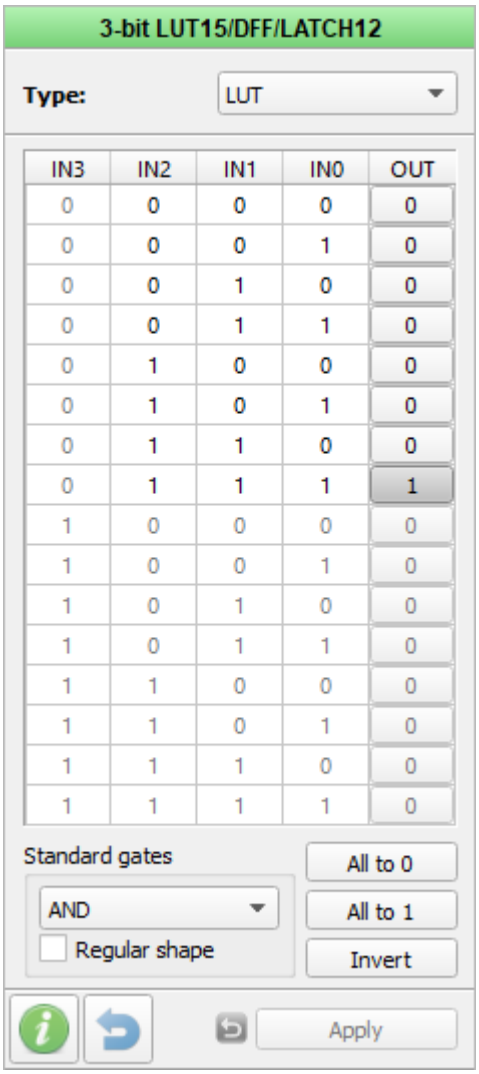

Figure 12: Odd Output Control

The entire parallel input Parity Generator and checker design is shown in Figure 13.

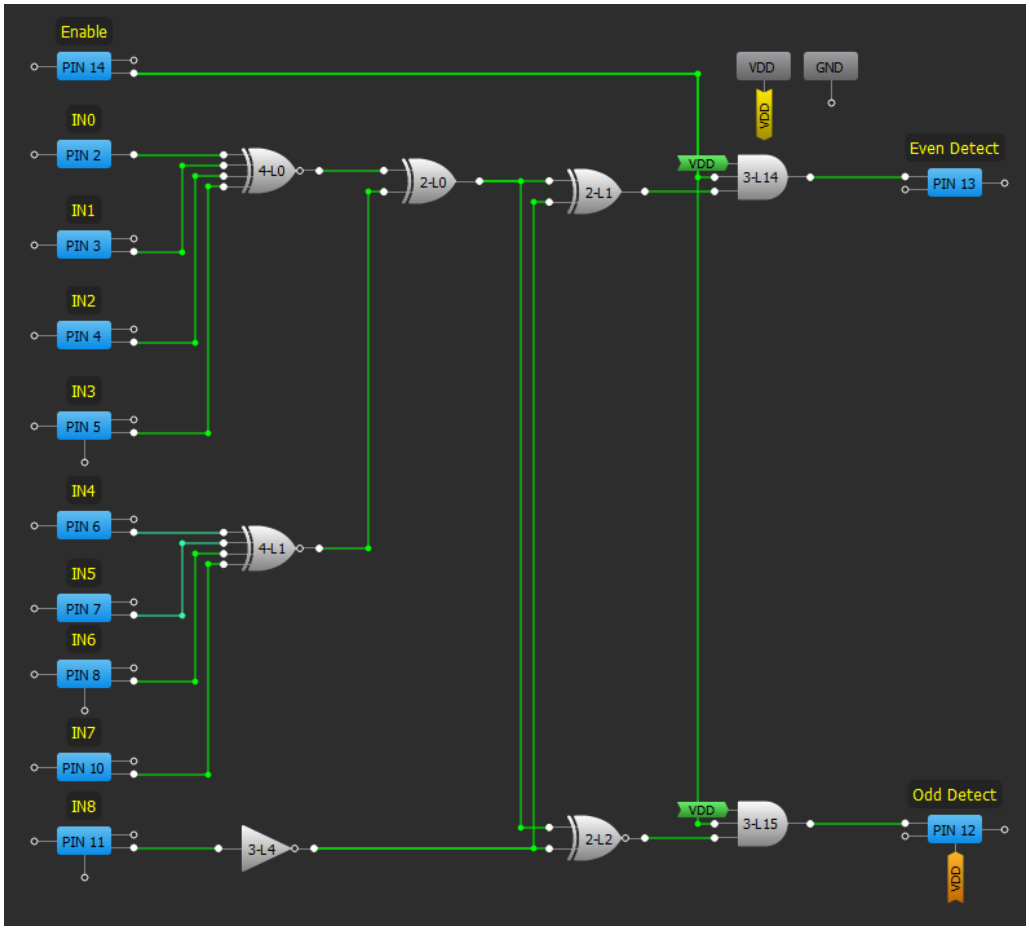

Figure 13: Parallel Input Parity Generator and Checker

The serial input variant is implemented with the SLG46536V. It has two matrixes that can be interconnected, so one of them was used to implement the serial to parallel converter and the other to implement the parity logic.

In Figure 14 and Figure 15, the Matrix 0 of the SLG46536V can be seen with the implemented serial to parallel converter.

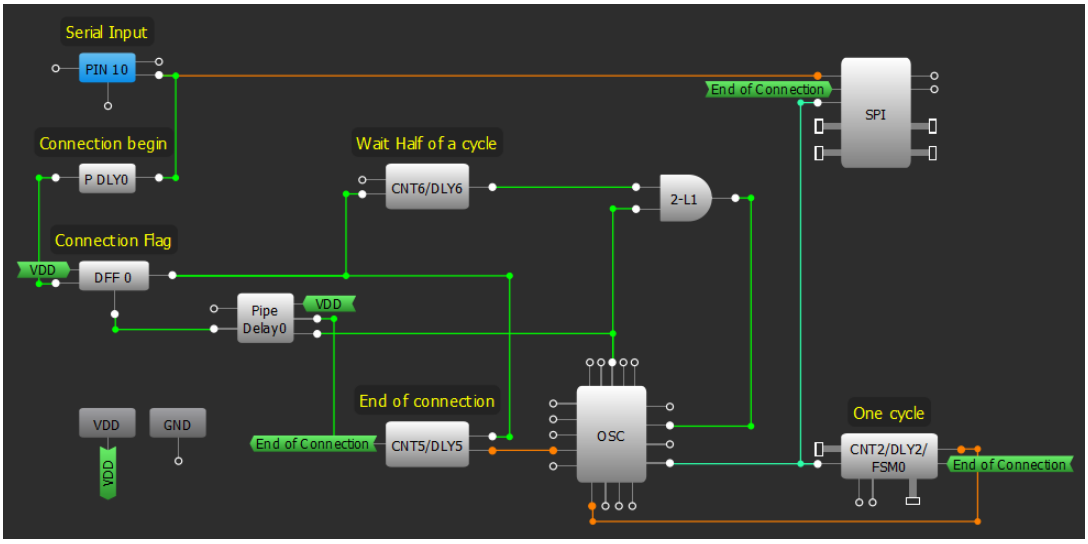

Figure 14: Serial to Parallel Converter (Matrix 0)

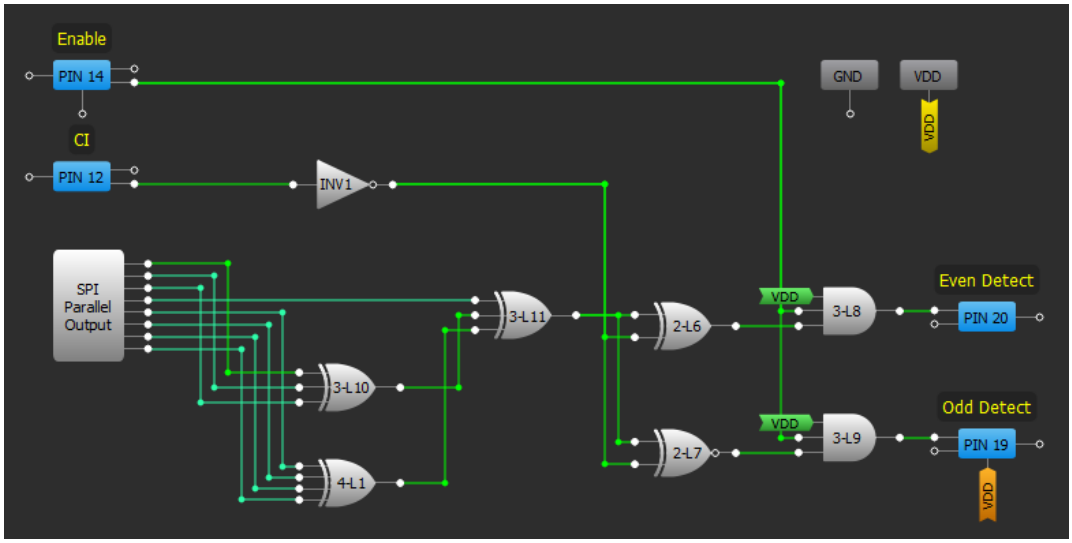

Figure 15: Serial to Parallel Converter (Matrix 1)

Pin 10 is used as the serial data input. As mentioned before, a falling edge detector with a delayed output is implemented with P DLY0. This signal is used to indicate the start of reception, held by DFF0 and DLY6.

Once the transmission has started, CNT2 generates a signal with a frequency equal to 9600. This is done by dividing the output clock of the Oscillator, which corresponds to the internal Ring Oscillator controlled by the 2-bit L1. CNT2 configuration can be seen in Figure 16.

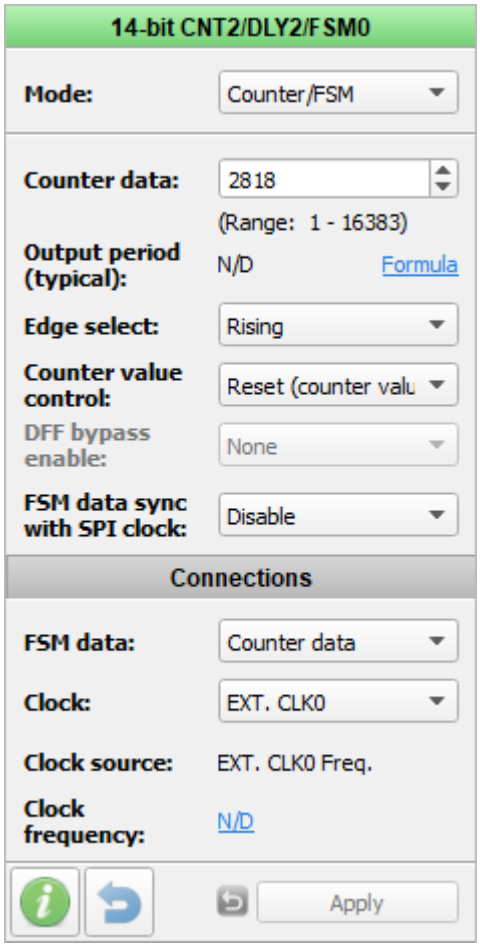

Figure 16: CNT2 Configuration

The data is received by the SPI block, configured in S2P mode and with an 8-bit data length. This can be seen in Figure 17.

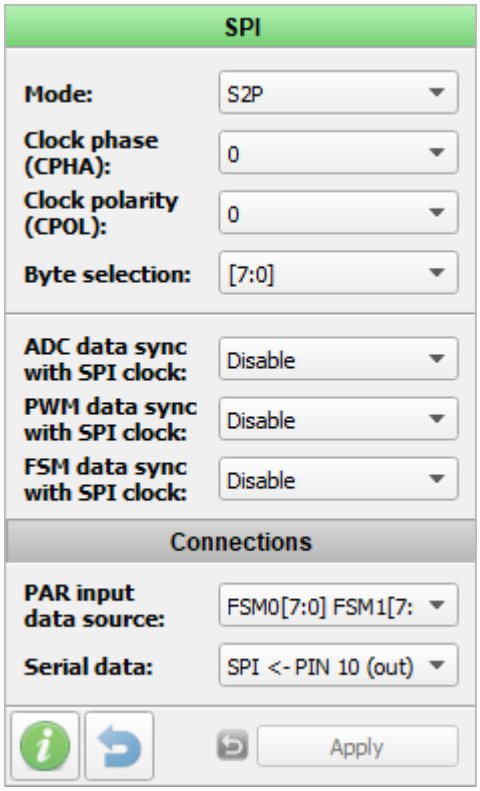

Figure 17: SPI Configuration

The reception is enabled until CNT5 reaches the maximum count, which is configured to be 8/9600 to receive the eight data bits. The counter uses the internal RC oscillator (2 MHz) divided by 24. Its configuration can be seen in Figure 18.

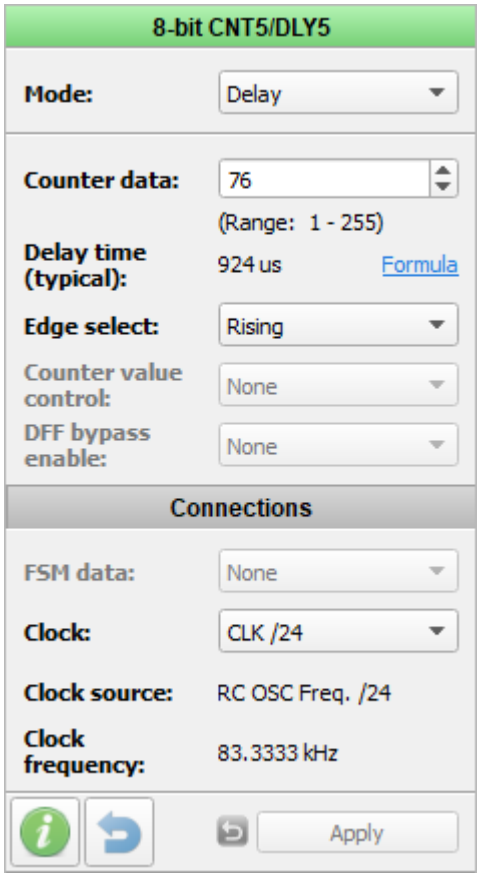

Figure 18: CNT5 Configuration

Once the data is received, it is processed by the logic implemented in Matrix 1. Matrix 1 can be seen in Figure 19.

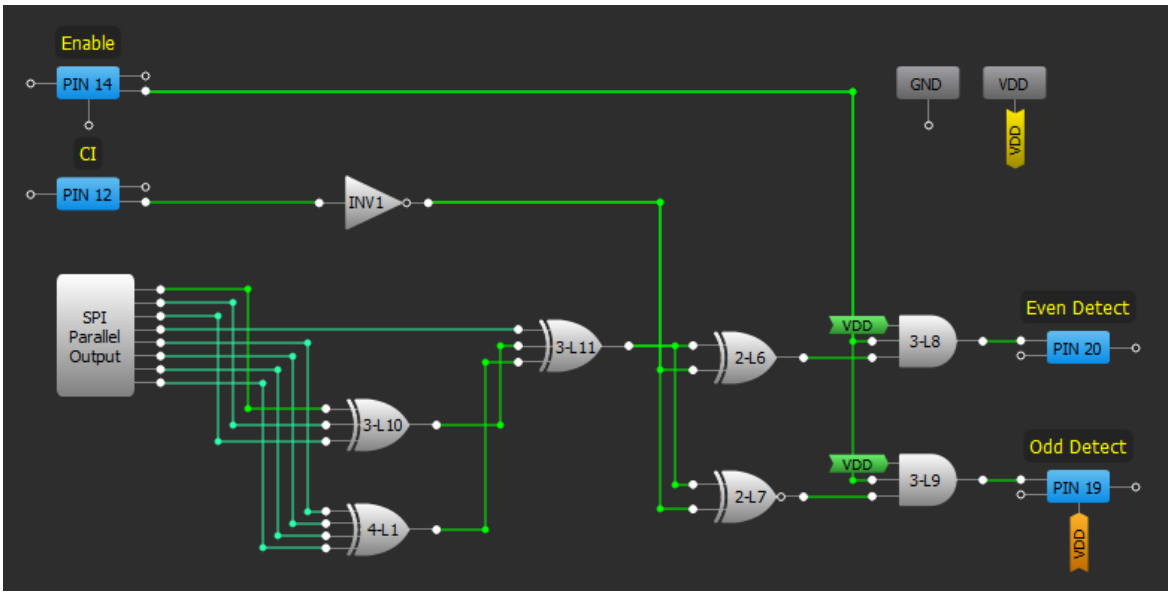

Figure 19: Serial Input Parity Checker

Data bits are obtained from the parallel output of the SPI Module. The XOR of the eight bits is implemented with 3-bit LUT10, 4-bit LUT1, 2-bit LUT4 and 2-bit LUT5. Finally, 2-bit LUT6 and 2-bit LUT7 implement the XOR and XNOR with the cascaded input (Pin 12) respectively. The enable controls are AND'ed by 3-bit LUT8 and LUT9.

7 Results

To test the implementation, the two variants of binary parity generator and checker were analyzed separately.

The parallel input Parity Generator was tested by generating known data to be processed by the Generator so the output can be checked. In this case, the used data was

Data

0X1110011

X was a bit changing from 0 to 1 periodically. This way, an Odd result was expected when X is 0 and Even result was expected when X is 1.

The input signals and the Odd and Even Detect outputs where measured with a logic analyzer. In Figure 20, the logged signals are shown to behave correctly.

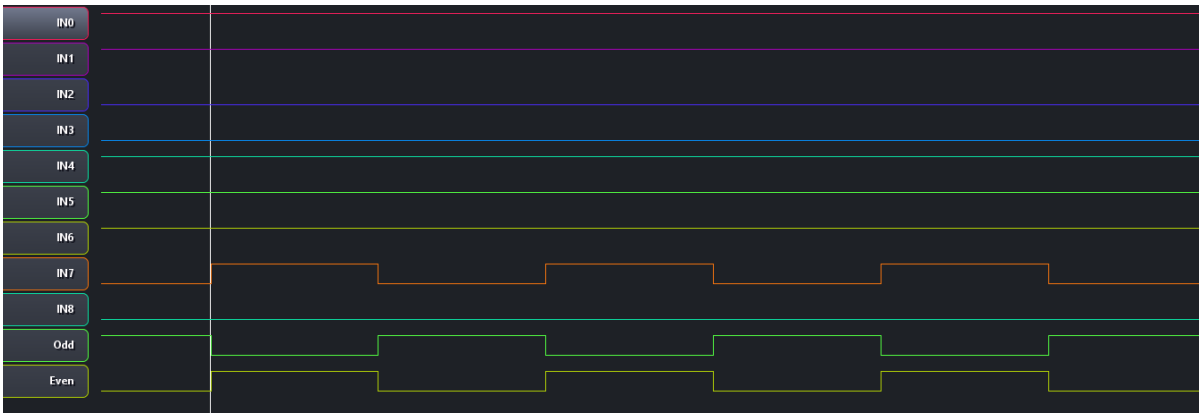

Figure 20: Parallel Input Parity Generator Test

The serial input parity generator variant was tested by independently transmitting two bytes, processing them, and verifying the result. The transmitted bytes were chosen to analyze an odd byte and an even byte.

In the odd data case, the used data was

Odd Data

11001101

In Figure 21, the serial input data and the Odd and Even detect Outputs, logged with a logic analyzer, are shown.

Figure 21: Serial Input Parity Generator Odd Test

The odd output is low until the Odd data is received. After that, the Odd detect output is high and the Even detect output is low.

In the even data case, the used data was 10011001.

In Figure 22, the serial input data and the Odd and Even detect Outputs, logged with a logic analyzer, are shown.

Figure 22: Serial Input Parity Generator Even Test

In this case, the previous odd data was re-sent so the Odd output is high, and the Even output is low prior to receiving the even data. After the even data is received, the Odd detect output is low and the Even detect output is high.

8 Conclusion

In this application note, we implemented two variants of a binary parity generator and Checker to be used as an error detection technique for data transmission. A parity bit is added to the transmitted data to make the number of 1s either even or odd. This bit is used to detect errors during the transmission of binary data. Several commercial IC's can be replaced with GreenPAKs so that the application size and cost can be reduced. The two variants show how the data input method can be either parallel or serial. This is useful for applying the parity generator in different applications.

Revision History

Revision	Date	Description
1.0	20-Jun-2018	Initial Version

IMPORTANT NOTICE AND DISCLAIMER

RENESAS ELECTRONICS CORPORATION AND ITS SUBSIDIARIES (“RENESAS”) PROVIDES TECHNICAL SPECIFICATIONS AND RELIABILITY DATA (INCLUDING DATASHEETS), DESIGN RESOURCES (INCLUDING REFERENCE DESIGNS), APPLICATION OR OTHER DESIGN ADVICE, WEB TOOLS, SAFETY INFORMATION, AND OTHER RESOURCES “AS IS” AND WITH ALL FAULTS, AND DISCLAIMS ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING, WITHOUT LIMITATION, ANY IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, OR NON-INFRINGEMENT OF THIRD-PARTY INTELLECTUAL PROPERTY RIGHTS.

These resources are intended for developers who are designing with Renesas products. You are solely responsible for (1) selecting the appropriate products for your application, (2) designing, validating, and testing your application, and (3) ensuring your application meets applicable standards, and any other safety, security, or other requirements. These resources are subject to change without notice. Renesas grants you permission to use these resources only to develop an application that uses Renesas products. Other reproduction or use of these resources is strictly prohibited. No license is granted to any other Renesas intellectual property or to any third-party intellectual property. Renesas disclaims responsibility for, and you will fully indemnify Renesas and its representatives against, any claims, damages, costs, losses, or liabilities arising from your use of these resources. Renesas' products are provided only subject to Renesas' Terms and Conditions of Sale or other applicable terms agreed to in writing. No use of any Renesas resources expands or otherwise alters any applicable warranties or warranty disclaimers for these products.

(Disclaimer Rev.1.01)

Corporate Headquarters

TOYOSU FORESIA, 3-2-24 Toyosu,
Koto-ku, Tokyo 135-0061, Japan
www.renesas.com

Contact Information

For further information on a product, technology, the most up-to-date version of a document, or your nearest sales office, please visit www.renesas.com/contact-us/.

Trademarks

Renesas and the Renesas logo are trademarks of Renesas Electronics Corporation. All trademarks and registered trademarks are the property of their respective owners.