2022 IEEE 4th International Conference on Civil Aviation Safety and Information Technology (ICCASIT) | 978-1-6654-6766-7/22/$31.00 ©2022 IEEE | DOI: 10.1109/ICCASIT55263.2022.9986576

Design of serial-parallel converter based on FPGA

Jiahui Yang
Hohai University
Changzhou, China
2468237566@qq.com

Peixuan Li
Space Engineering University
Beijing, China
2223156872@qq.com

Yihan Li
Taiyuan University of Technology
Taiyuan, China
2291979372@qq.com

Qingxin Cao*
Beijing Jiaotong University
Beijing, China
19723042@bjtu.edu.cn

***Abstract*—In this paper, the design of an 8-bit serial-parallel conversion system which can be applied to FPGA is presented. Serial-parallel conversion is a very common design and be widely used in digital electronic systems. In communication systems, using parallel data bytes and transmitting them over a single line can be efficient in saving wire area, reducing crosstalk effects, increasing clock speeds, or maintaining compatibility between devices. However, with the upgrading of the processing speed and frequency of FPGA, the serial-parallel conversion using the basic chip seems to be no longer reliable. Therefore, based on the existing problems, in this paper, the development and simulation process of a PFGA-based serial-parallel conversion system is introduced. The system consists of a 4-bit counter module, Series/parallel conversion module, which is composed of multiplexers and flip-flops, combination module. The counter is used to control the time of the overall system and provide the function of reset. The series/parallel conversion module convert an 8-bit parallel signal to a serial signal firstly, after the transmission and processing, it is restored to the original parallel signal again. The combination module makes the whole system work orderly. The system is implemented with VHDL language. The rationality of the system is verified by experiments.**

Keywords—FPGA, Serial-parallel convertion, VHDL

I. Introduction

FPGA (Field Programmable Gate Array) is a digital integrated circuit chip. It is one of the physical implementations of FPGA digital circuits. Compared with another implementation of digital circuits, an important feature of FPGA is its programmability, users can specify FPGA to implement a specific digital circuit through programs. VHDL is one of the common programming languages of FPGA. Parallel/serial conversion is widely used in design of FPGA. It has been shown [1] that when data is transferred outside, a common method is to convert the data from parallel signal to a serial one. Conversely, when data is transferred inward to a computer, it should be converted from a serial signal to a parallel signal, this process can efficiently save power and raise processing speed [2].

The design of an 8-bit serial-parallel conversion system in FPGA is introduced. It is designed to be implemented through component instantiation, dividing the complex system into three modules, namely Counter module, Serial/Parallel conversion module and Combination module respectively. In this paper, firstly, the methods of designs for each single module and the whole implement system will be described in detail respectively, as well as work performed, results obtained and analysis. Moreover, a general conclusion and the process of physical experiment will be presented. Based on the analysis of the strengths and weaknesses of this system in conclusion part, the future recommended work will also be indicated.

II. Method of Design and Implementation

A. Counter module

A 4-bit counter is created in this module, which acts as a timer for the serialization and de-serialization modules. Signal *clk* and *rst* are contained as the inputs, 4-bit signal vector *outputc* acts as output.

The code piece is shown below as Figure 1. The counter is initialized first. Then judge the value of *rst*. When *rst*=1, the counter is reset. If *rst* ≠ 1, the counter starts counting sequentially (grow 1 each time) at the rising edge of the clock signal *clk*. Since in the later modules, only 8 bits of data need to be converted, however there are 2^4=16 possible count results of a 4-bit counter, in order for each bit of data to correspond to a unique count result, it is judged that when the counter counts to "1000" (the 9th value), reset the counter to "0000". So, for the counter, there are 8 valid count results ("0000" to "0111") and 8 invalid count results ("1000" to "1111").

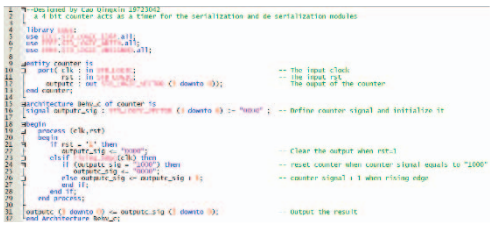

Fig.1. VHDL code for counter module

The results in RTL viewer are shown in Figure 2. The counter is connected by adder, mux and a D flip-flop.

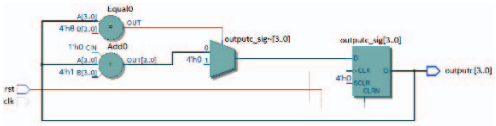

Fig.2. RTL diagram code for counter module

Analysis: Every time the counter outputs a number, it will be input into the adder again to add 1 and continue to accumulate. While the multiplexer is used to judge whether the counter exceeds the counting range. When "1000" is reached, the multiplexer will no longer output the result of the adder, thus achieving the effect of reset. The new result will be sent to the D flip-flop then, each time the clock reaches the rising edge, the value stored in the D flip-flop is output as a result of the count. When *rst*=1, the counter is reset.

B. Parallel-to-Serial converter

A Parallel-To-Serial converter is created in this module,

which takes an 8-bit parallel input signal and convert it into a serial output format, such that each bit will be transmitted on a 1-bit wide output wire over 8 clock cycles. Signal *clk* and *rst* and an 8-bit signal vector *Parallel* are contained as the inputs, single signal *Serial* acts as output.

The code piece is shown below as Figure 3. In order to sort out the required timing for the parallel-to-serial conversion, the counter module previously established is declared and instantiated for the converter. Additionally, the converter uses case statements to switch between the output states by defining and selecting different binary patterns, which is similar to a state machine with 8 possible output states.

The counter will start count sequentially from "0000", increase by 1 every time the clock signal rises. At the same time, the converter will follow the case select statement, according to different count result of the counter, assign an output state to each bit of the 8-bit input *Parallel*, finally output a serial signal *Serial* in 8 clock cycles, then the counter is reset in the 9th clock cycle.

Fig.3. VHDL code for Parallel-To-Serial converter

The results in RTL viewer are shown in Figure 4. The Parallel-To-Serial converter is connected by mux and latch. Analysis: signal of eight valid count results ("0000" to "0111") and signal of eight invalid count results ("1000" to "1111") defined in the counter enter Mux0 and Mux1 respectively in order to make selection in the next step. Mux0 is also connected with input signal. When the input clock is one of eight valid count results, 8-bit parallel input signal will be correspondingly output, then the output serial signals will be obtained through a latch. Mux1 is also connected to a null signal. When the input clock is one of eight invalid count results, the null signal will enter the enable terminal of latch, and no output signal will be obtained.

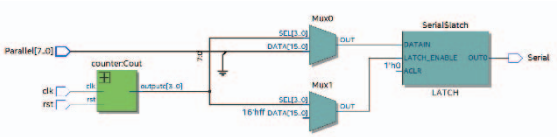

Fig.4. RTL diagram for Parallel-To-Serial converter

A Testbench is created to simulate the output waveform, as shown in Figure 5.

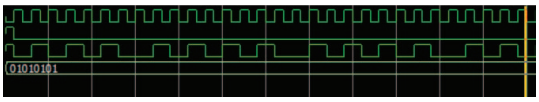

Fig.5. Testbench result for Parallel-To-Serial converter

For the Testbench result, set the parallel input signal to "01010101". In the first cycle, let *rst*=1, observe that the counter is cleared. Then set *rst*=0, let the converter start working formally. It's find that the 8-bit parallel input signal is successfully converted into a serial signal in 8 clock cycles, which is triggered at every rising edge of the clock, and the output sequence is from LSB to MSB of the 8-bit parallel input signal. In the 9th clock cycle, it's reset. Thus, the Parallel-To-Serial converter runs normally.

C. Serial-to-Parallel converter

A Serial-To-Parallel converter is created in this module, which almost the opposite of the Parallel-to-Serial converter created before. The serial output signal obtained in B is took as input here, and get an 8-bit output signal through this converter, same as the 8-bit parallel input signal in B. Signal *clk* and *rst* and *Serial* are contained as the inputs, an 8-bit signal vector Parallel acts as output.

The code piece is shown below as Figure 6. Similarly, the counter module is declared and instantiated to act as a timer for the converter. Moreover, the converter uses case statements to switch between output states by defining and selecting different binary patterns, which is similar to the state machine with 8 possible output states.

The counter will start count sequentially from "0000", increase by 1 every time the clock signal *clk* rises. At the same time, the converter will follow the case select statement, according to each current bit of serial input signal Serial and different count result of the counter, transfer corresponding output bits into a new declared 8-bit signal vector reg (prevent the output port from presenting incomplete data) firstly. In 8 clock cycles, the input Serial is completely converted into reg. Then the reg will be transferred into 8-bit parallel output signal Parallel in the 9th clock cycle. The first eight bits of this output will be consistent with the input signal. The counter is reset in the 9th clock cycle.

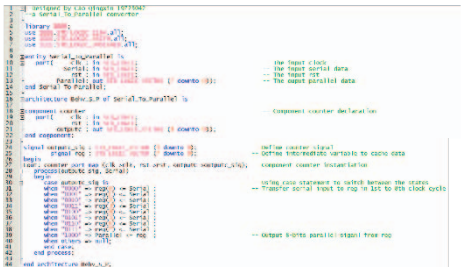

Fig.6. VHDL code for Serial-To-Parallel converter

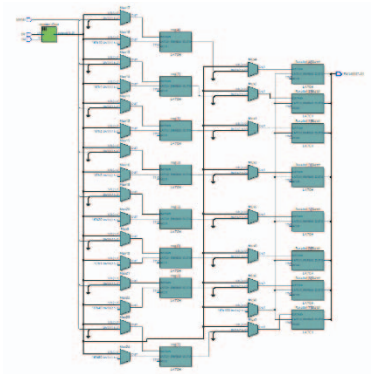

Fig.7. RTL diagram for Serial-To-Parallel converter

The results in RTL viewer are shown in Figure 7. The Serial-To-Parallel converter is connected by mux and latch.

It is observed that there are two groups of mux and latch connected in RTL diagrams. The first group of mux and latch are similar with the one in *B*, different count results are used to match the corresponding signals: when input 8 valid count results ("0000" to "0111") to select, each bit of serial input

signal *Serial* will be correspondingly output as an 8-bit parallel signal vector in *reg*. when input 8 invalid count results ("1000" to "1111") to select, no output signal will be obtained. The second group of mux and latch is used to transfer the value in *reg* to the output 8-bit parallel vector signal *Parallel* at the correct clock.

A Testbench is created to simulate the output waveform, as shown in Figure 8.

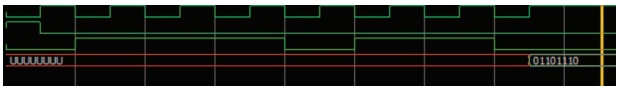

Fig.8. VHDL Testbench result for Serial-To-Parallel converter

At the beginning of the Testbench code, set the serial input signal to "01110110". Firstly, let *rst* signal =1 to reset the counter. Then set *rst* =0, let the converter start working formally. Observing the output results then, it's found that the serial input signal Serial is successfully converted into an 8-bit parallel output signal *Parallel* in 8 clock cycles, which is triggered at every rising edge of the clock, the output sequence is from LSB to MSB of the 8-bit parallel output signal. This is actually an opposite process of *B*. Thus, the Serial-To-Parallel converter runs normally.

D. Combination

The module of Combination is used to implement all the previous modules together to finish the overall function, which is shown as Figure 9. Input signal *clk_Comb* and *rst_Comb* gives clock for the whole system and provide the function of reset. An 8-bit signal vector *Parallel_in* is input to this system, then getting a same 8-bit signal vector output *Parallel_out*.

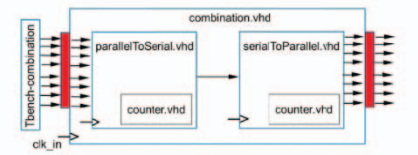

Fig.9. Overall view of the Parallel/Serial Conversion System

The code piece is shown as Figure 10. The Parallel-To-Serial converter component created in *B*, the Serial-To-Parallel converter component created in *C*, are declared and instantiated respectively. The Combination module is used to effectively connect the previously declared modules together.

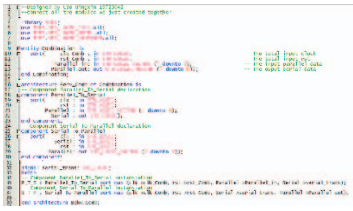

Fig.10. VHDL code for Combination module

The results in RTL viewer are shown in Figure 11. Compare Figure 9 and Figure 11, it's found that the RTL viewer output a same structure with the concept map. The 8-bit parallel input signal *Parallel_in* firstly enters Parallel-To-Serial converter, be converted into a serial signal, then enters Serial-To-Parallel converter, be restored into an 8-bit parallel signal *Parallel_out*. Input signal *clk_Comb* controls the clock of the overall system and *rst_Comb* is used to control the reset function of the counter.

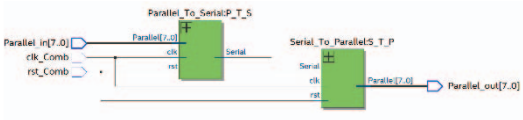

Fig.11. RTL diagram for Combination module

A Testbench is created to simulate the output waveform, as shown in Figure 12.

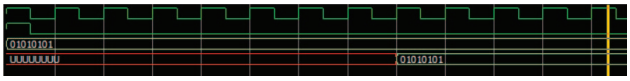

Fig.12. VHDL Testbench result for Combination

Observe the testbench result for this Combination module, when a "01010101" is given as input, the same signal is output after 8 cycles. Thus, the Combination module is successfully set up and the overall system rans normally.

III. Experiment Results and Conclusion

Observe the simulation result in Testbench (Figure 5, 8, 12), the design of an 8-bit serial-parallel conversion system is built successfully. Use FPGA to verify the correctness of this system again, download the program and check the results. When input a serial signal "01010011" to the FPGA, after a period of conversion, the output result of the FPGA is exactly the same as this input, so this system is completely correct, as shown in Figure 13.

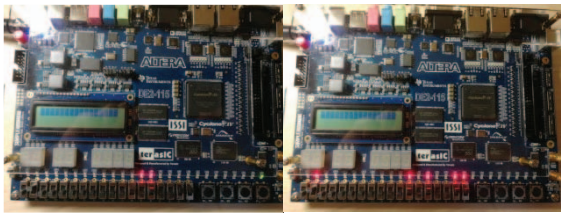

Fig.13. The rationality verification using FPGA

Thus, the Counter module can provide clock for other modules and achieve reset function, both Parallel-to-Serial and Serial-to-Parallel modules can follow the case statement, output correctly according to different count results of Counter module. The functions of converting an 8-bit parallel input signal to a serial format, and restoring the serial format signal to the original 8-bit parallel signal are implemented respectively. The Combination module combines all modules together. Each 8-bit parallel input signal is observed to be successfully converted to the same 8-bit parallel output signal after 8 clock cycles.

However, considering the deficiencies of this design, there are still three important points. First, the actual transfer period of this Parallel-to-Serial and Serial-to-Parallel converter is not 8 cycles, the 9th clock cycle is required to reset the counter and output data. In addition, the system cannot guarantee whether the transmission is reliable. In other words, the system implements the function of converting the parallel signal to serial and then restoring it back, however, the whole system does not know if the information it is transmitting is correct. Last but not least, the circuit modules utilized in the whole system is relative complex, which is probable to unintentionally increase the consumption, it's shown that [6] one of the reasons of the designed converter is to reduce the consumption in transmission, and the complexity of the combination system may be contrary to the design concept.

IV. Recommendations for Future Work

Firstly, the limitations and shortcomings of this design will be summarized, some of which have been presented in the previous sections. The corresponding possible solutions will then be presented.

A. Limitation 1 & Probable solution

There is no pause function in this design, it stops only when a complete transmission process is finished. Besides, as mentioned before, the reliability of the transmission is not guaranteed. To solve this, a pause key can be designed and used to control the whole process [4]. In addition, an intermediate buffer variable *reg* can also be set to increase the reliability of the Parallel-to-Serial converter, which is similar with the Serial-to-Parallel converter, thus ensuring more stable output results.

B. Limitation 2 & Probable solution

The actual transfer period of this design is not 8 cycles, the 9th clock cycle is required to reset the counter and output data. To solve this, deleting redundant signals between inputs, additionally, optimizing the counter module and redesigning it, is also a possible approach [5].

C. Limitation 3 & Probable solution

The design code is relatively complex, leading to the circuit is cumbersome, which may lead to more consumption. To solve this, The use of shift registers consisting of D flip-flops with an enable signal may be a good approach, which allows the code to be greatly simplified and makes greater use of FPGA resources [5]. Based on this design concept, a similar system is designed by instantiating declared D-flipflops, which is simply shown in Figure 14.

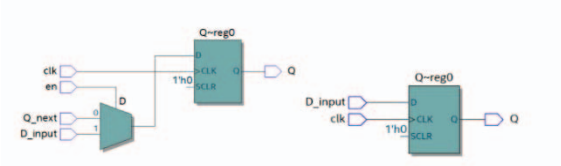

Fig.14. RTL viewer for 2 D flip-flops (with enable and without enable)

References

[1] Wilson, Peter. (2016). Design recipes for PFGAs. 2nd ed., Amsterdam, Netherlands: Newnes, 2016.

[2] Stackler, Marc. "Interfacing FPGA with High-speed Data Converter Using Parallel and Serial Interface." Signal Integrity Journal, 19 Feb. 2018,

[3] Nitin Sharma. (2020). Difference between Serial and Parallel Transmission. Tutorialspoint

[4] Guo-Ming, Sung. "USB Transceiver With a Serial Interface Engine and FIFO Queue for Efficient FPGA-to-FPGA Communication." IEEE Xplore, IEEE.

[5] Bing, Shen. "Research on Video Remote Transmission Technology Based on FPGA." NASA/ADS, IOP Conference Series: Materials Science and Engineering, Volume 382, Issue 4, pp. 042031 (2018), July 2018

[6] Prajapati, Hardik. "Efficient FFT/IFFT Implementation Technique for OFDM on FPGA." Department of Electronics & Communication Engineering, vol. 5, no. 08, Aug. 2017, pp. 35-37