Design and Implementation of an ARINC 429 Transmitter on FPGA

Author
Kouzmane Saad

Abstract

In the field of avionics, reliable and standardized communication between various subsystems is critical for the safe and efficient operation of aircraft. The ARINC 429 standard, developed by Aeronautical Radio, Inc. (ARINC), is one of the most widely used data bus protocols in commercial aviation for the exchange of information between avionics components. This standard specifies the physical and electrical interfaces, data format, and protocol for serial data transmission in aircraft systems.

1 Introduction

The ARINC 429 protocol operates using a unidirectional data bus, where data is transmitted in 32-bit words that include a label, Source/Destination Identifier (SDI), data field, Sign/Status Matrix (SSM), and a parity bit for error checking. The simplicity and reliability of the ARINC 429 bus have made it a preferred choice in avionics systems, where robust and deterministic communication is essential.

This project aims to design and implement an ARINC 429 transmitter on a Field-Programmable Gate Array (FPGA). The transmitter is responsible for encoding and serially transmitting 32-bit ARINC 429 data words in compliance with the protocol specifications. Given the high reliability requirements of avionics systems, the project incorporates a First-In, First-Out (FIFO) buffer to manage data flow and ensure that data is transmitted in an orderly and timely manner.

This report details the design methodology, implementation steps, and testing procedures, providing a comprehensive overview of the ARINC 429 transmitter development. The successful implementation on the FPGA demonstrates the feasibility and effectiveness of using FPGAs for avionics communication systems.

2 ARINC 429 Frame Format

The ARINC 429 data word is structured as follows:

ARINC 429 Word Format																														
P	SSM		MSB										Data								LSB	SDI		LSB						MSB
32	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1

Figure 1: ARINC 429 Frame Format

- **Label (8 bits)**: Identifies the type of data being transmitted. Each label corresponds to a specific type of information.
- **SDI (2 bits)**: Used to specify the source or destination of the data within the ARINC 429 system. It helps to identify which device or system the data is coming from or going to.
- **Data (19 bits)**: Contains the actual data being transmitted. The format of the data field varies depending on the information type.
- **SSM (2 bits)**: These bits may have various encodings dependent on the particular data representation applied to a given word: In all cases using the SSM, these bits may be encoded to indicate, Normal Operation (NO) - Functional Test (FT) - Failure Warning (FW) - No Computed Data (NCD)
- **Parity (1 bit)**: Used to verify that the word was not damaged or garbled during transmission. Every ARINC 429 channel typically uses "odd" parity

3 Design Overview

The ARINC 429 transmitter is designed in VHDL to encode and transmit data based on the ARINC 429 standard. This design includes a state machine that governs the different phases of data transmission according to the ARINC 429 protocol.

3.1 Interface Inputs and Outputs

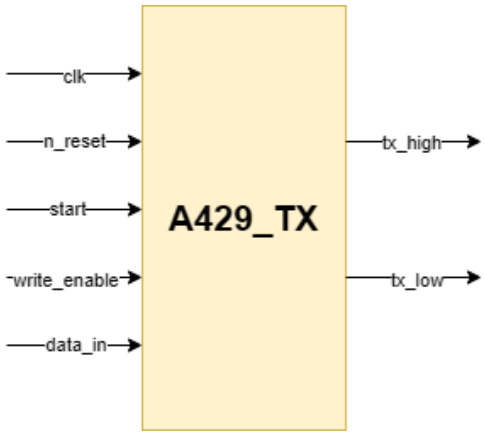

Figure 2: ARINC 429 Transmitter Interface

Inputs

- **clk:** The clock signal that synchronizes all operations within the ARINC 429 transmitter. It regulates state transitions and data encoding processes.
- **n_reset:** Initializes the transmitter to a known state, typically the idle state. When de-asserted, it clears all internal registers and the state machine.
- **start:** Initiates the data transmission process. When asserted, the transmitter transitions from the idle state to the read_data state and begins transmitting the encoded ARINC 429 word.
- **data_in:** The primary data input to the transmitter, carrying the 19-bit data word that is to be transmitted. This data includes the payload that the ARINC 429 transmitter encodes and sends as part of the ARINC 429 word format.
- **write_enable:** Controls the writing of data to the FIFO buffer. When asserted, the data from the data_in signal is written into the FIFO for subsequent transmission.

Outputs

- **tx_high:** Outputs the high level of the differential signal according to the ARINC 429 protocol during transmission. This signal forms one part of the balanced differential pair used for data transmission.
- **tx_low:** Outputs the low level of the differential signal according to the ARINC 429 protocol during transmission. This signal forms the other part of the balanced differential pair used for data transmission.

4 Generic Parameters

Clock Frequency

- **Default Value**: 50,000,000 (50 MHZ)
- **Description**: This parameter defines the clock frequency of the component in Hertz (Hz). It determines the rate at which the component's internal logic operates. Adjusting this parameter allows the component to match the clock speed of the overall system, facilitating integration with different clock domains.

Transmission Speed

- **Default Value**: 100,000 (100 kb/s - high speed)
- **Description**: Specifies the data transmission speed in bits per second (bps). This parameter is crucial for configuring how rapidly data is sent or received by the component. It enables the component to be used in systems with varying data rates.

Frame Label

- **Default Value**: X"4F"
- **Description**: Represents an 8-bit identifier or label used in framing data packets. This vector is instrumental in distinguishing different types of frames or messages within the system. The default value X"4F" can be customized based on specific protocol requirements or operational needs.

SDI

- **Default Value**: "00" (All systems)
- **Description**: 2-bit vector used to specify the source or destination of the data being transmitted. This vector identifies the particular source or target component within a system, allowing for precise routing and handling of data.

SSM

- **Default Value**: "11" (BNR format - Normal operation)
- **Description**: A 2-bit vector used to specify the status or mode of the component. This parameter can control different operational modes or statuses, offering versatility in how the component is utilized.

5 State Machine

The state machine in the ARINC 429 transmitter is designed to control the sequence of operations required for the correct transmission of the ARINC 429 frame. It ensures that the data, including the label, Source/Destination Identifier (SDI), data bits, Sign-Status Matrix (SSM), and parity bit, are transmitted in the proper order. The state machine operates through several distinct states, each responsible for a specific part of the frame.

- **idle**: The initial state where the system waits for the 'start' signal. If 'start' is asserted and the FIFO is not empty, the state machine transitions to the 'read_data' state. In this state, output signals 's_tx_high' and 's_tx_low' are set to '0', and the clock count is initialized.
- **read_data**: This state enables reading data from the FIFO buffer. It prepares the data for transmission and then transitions to the 'label_state'. The FIFO read enable signal ('fifo_read_enable') is asserted during this state.
- **label_state**: In this state, the transmitter sends the frame label. It transmits each bit of the 'frame_label' sequentially. The 'data_bit' signal keeps track of the current bit being transmitted. The state machine transitions to the 'sdi_state' after the entire label has been sent.

- **sdi_state**: This state handles the transmission of the Source/Destination Identifier (SDI). Similar to the 'label_state', it transmits each bit of the 'sdi' signal. Once all SDI bits are transmitted, the state machine moves to the 'data_state'.
- **data_state**: This state manages the transmission of the data bits. It reads the data from the FIFO output and sets the 's_tx_high' and 's_tx_low' signals accordingly. After transmitting all data bits, it transitions to the 'ssm_state'.
- **ssm_state**: In this state, the Sign-Status Matrix (SSM) is transmitted. Each bit of the 'ssm' signal is sent sequentially. The state machine transitions to the 'parity' state once the SSM transmission is complete.
- **parity**: This state is responsible for transmitting the parity bit. It calculates the parity based on previously transmitted bits and sets the 's_tx_high' and 's_tx_low' signals accordingly. After sending the parity bit, the state machine transitions to the 'four_cycles_idle' state.
- **four_cycles_idle**: The transmitter remains idle for four cycles after transmitting the parity bit. This state ensures a delay before returning to the 'idle' state. It provides a pause to ensure that the transmitted frame is properly spaced out.

6 Simulation and Verification

After detailing the state machine and its operation, it's crucial to verify that the ARINC 429 transmitter functions correctly in practice. This is achieved through comprehensive simulation and verification with visual inspection of the wave and check the bit timings.

6.1 Purpose of Simulation

The primary aim of the simulation is to ensure that the ARINC 429 transmitter adheres to the ARINC 429 protocol and performs its intended functions accurately. The simulation allows us to:

- Verify that the transmitter progresses through the correct state sequence.
- Ensure that data, frame labels, source/destination identifiers (SDI), and Sign-Status Matrix (SSM) are transmitted as expected.
- Validate the timing of data bits and the accuracy of the parity bit.

6.2 Simulation Approach

To achieve these objectives, a testbench was designed to simulate the ARINC 429 transmitter's behavior. Key elements of the testbench include:

- **Clock Signal Generation**: Provides the necessary timing for the transmitter.
- **Reset and Control Signals**: Initialize the transmitter and control its operation.

- **Data Input**: Simulates various data values to be transmitted.
- **Output Monitoring**: Observes the transmitter's output signals to verify correct data transmission.

The testbench was configured with the following generic parameter values for the ARINC 429 transmitter: f

```
generic(
clk_frequency : integer := 50_000_000; -- 50mhz
tranmssion_speed : integer := 100_000;  -- 100 kb/s
frame_label : std_logic_vector(7 downto 0) := X"4F";
ssm : std_logic_vector(1 downto 0) := "11"; -- Normal Operation BNR format
sdi : std_logic_vector(1 downto 0) := "00" -- All systems
);
```

Figure 3: Testbench Configuration with Generic Parameters

6.3 Results and Observations

6.3.1 State Transitions

The state machine's transitions were monitored to ensure it correctly moved through the states (`idle`, `read_data`, `label_state`, `sdi_state`, `data_state`, `ssm_state`, `parity`, and `four_cycles_idle`). The state machine functioned as expected, confirming that the transmitter processes data in the correct sequence.

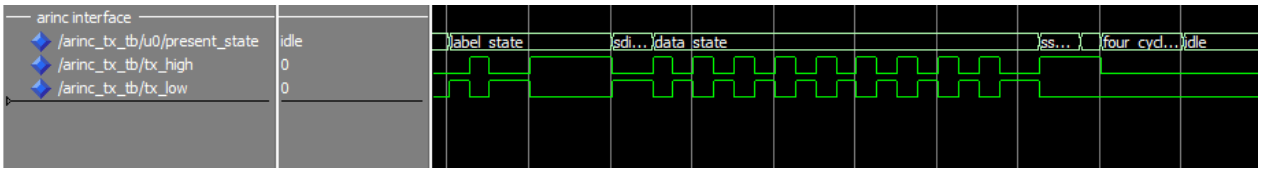

Figure 4: State Machine Transitions

6.3.2 Label Transmission

The simulation verified that the frame label was transmitted correctly. The generic parameter for the frame label was set to `0X4F`, which was transmitted with the most significant bit (MSB) first order. The transmitted label matched the expected value, demonstrating that the transmitter correctly encodes and sends the frame label according to the specified format.

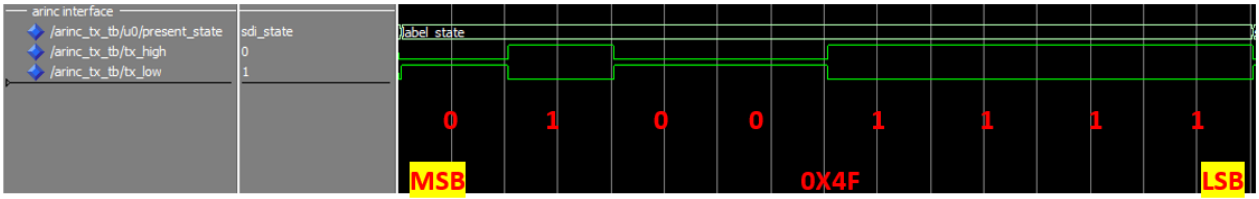

Figure 5: Frame Label Transmission

6.3.3 Source/Destination Identifier (SDI)

The Source/Destination Identifier (SDI) was set to '00', indicating that the data is intended for all systems. The simulation results showed that the SDI was transmitted correctly as '00', validating that the transmitter is configured to broadcast data to all systems or is not targeting a specific one.

Figure 6: Source/Destination Identifier (SDI) Verification

6.3.4 Data Bits and Timing

The data bits were checked to ensure they were transmitted correctly according to the ARINC 429 protocol. The bit timing was also analyzed to confirm that data bits were sent with the correct timing intervals.

Given a transmission speed of 100 kb/s, each bit takes 10 μs to transmit. With a clock frequency of 50 MHz, the correct timing of 10 μs per bit was confirmed during simulation.

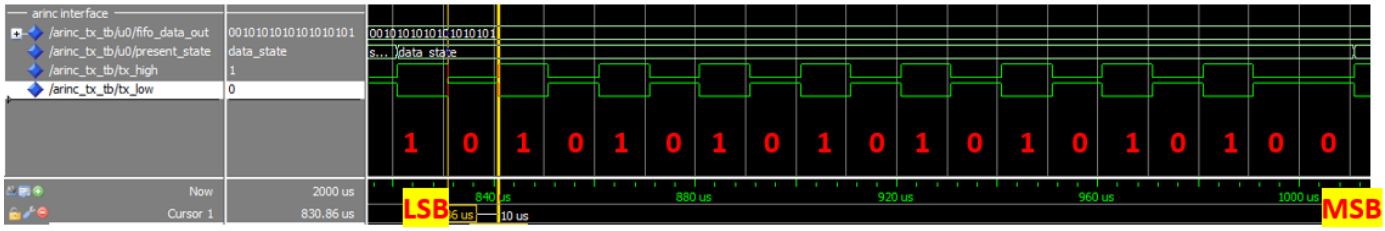

Figure 7: Data Bits Transmission

6.3.5 Sign-Status Matrix (SSM)

In the ARINC 429 transmitter, the Sign-Status Matrix (SSM) is used to indicate the status of the transmitted data. For normal operation, the SSM was set to '11' in the generic parameters. The simulation verified that the SSM field was transmitted correctly as '11', confirming that the transmitter operates under normal conditions without any error or special status.

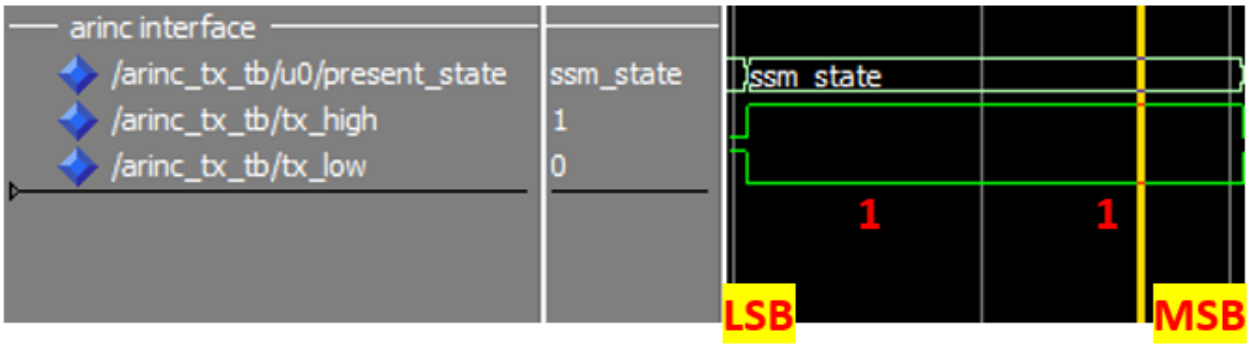

Figure 8: Sign-Status Matrix (SSM) Verification

6.3.6 Parity Bit

The parity bit is used to ensure data integrity by making the number of '1's in the transmitted data odd or even, depending on whether odd or even parity is used. In this case, odd parity is used, which means that the total number of '1's, including the parity bit, should be odd.

Let's calculate the parity bit for the given data:

- **Frame Label**: '4F' in hexadecimal translates to '01001111' in binary. This has 5 ones.
- **SDI**: '00' in binary has 0 ones.
- **Data Bits**: '001' followed by '5555' in hexadecimal translates to '00101010101010101010101010101' in binary. This has 9 ones.
- **SSM**: '11' in binary has 2 ones.

Adding these together:

- Total number of ones = 5 (Frame Label) + 0 (SDI) + 9 (Data Bits) + 2 (SSM) = 16 ones

Since the total number of ones is 16 (an even number) and we are using odd parity, the parity bit should be set to '1' to make the total number of ones odd.

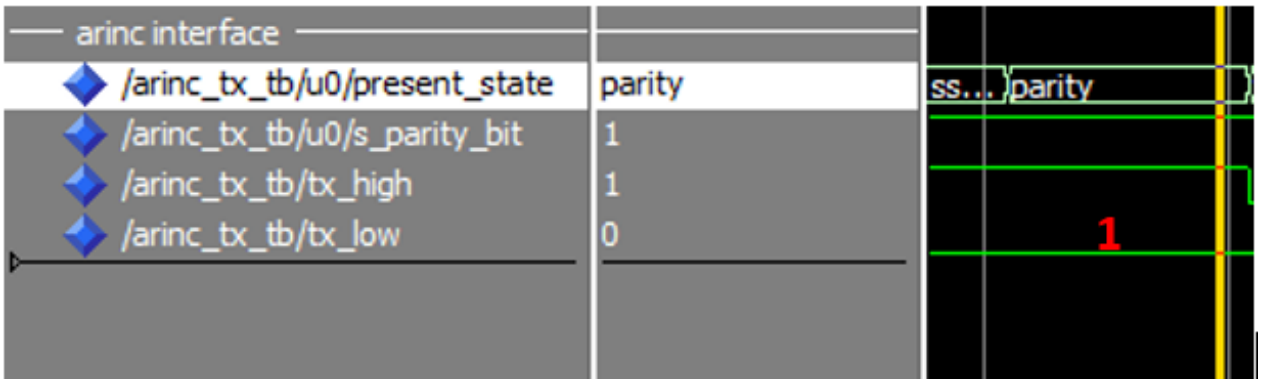

Figure 9: Parity Bit Transmission

7 Implementation on Spartan-6 FPGA

7.1 Overview

The ARINC 429 transmitter has been implemented on a Spartan-6 FPGA to test its functionality in a real hardware environment.

7.2 Hardware Setup

- **FPGA Model**: Spartan-6 (Xilinx)
- **Clock Frequency**: 50 MHz
- **I/O Pins**: Used for transmitting data (TX_HIGH and TX_LOW), clock, reset, and control signals.

7.3 Design Implementation

7.3.1 Synthesis

The VHDL code for the ARINC 429 transmitter was synthesized using Xilinx ISE. The synthesis process translates the high-level VHDL code into a netlist of logic gates.

7.3.2 RTL Schematic

The RTL schematic provides a graphical representation of the top-level design. It shows how the various components of the design are interconnected.

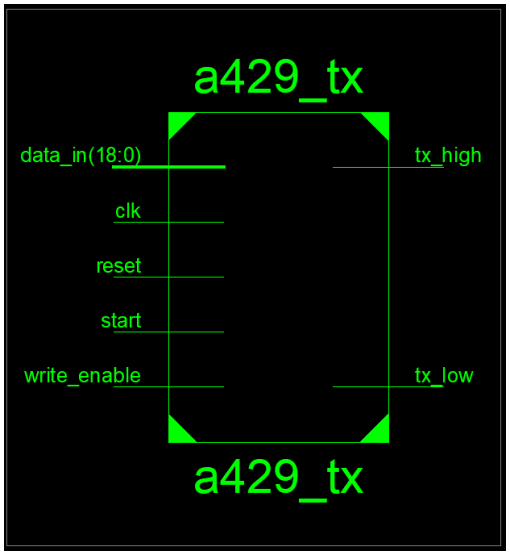

Figure 10: RTL Schematic of the Top-Level Design with Netlist

7.3.3 Netlist

The netlist provides a detailed view of the synthesized design, showing the logical gates and connections that make up the design. This is essential for understanding how the high-level VHDL code maps onto the FPGA's resources.

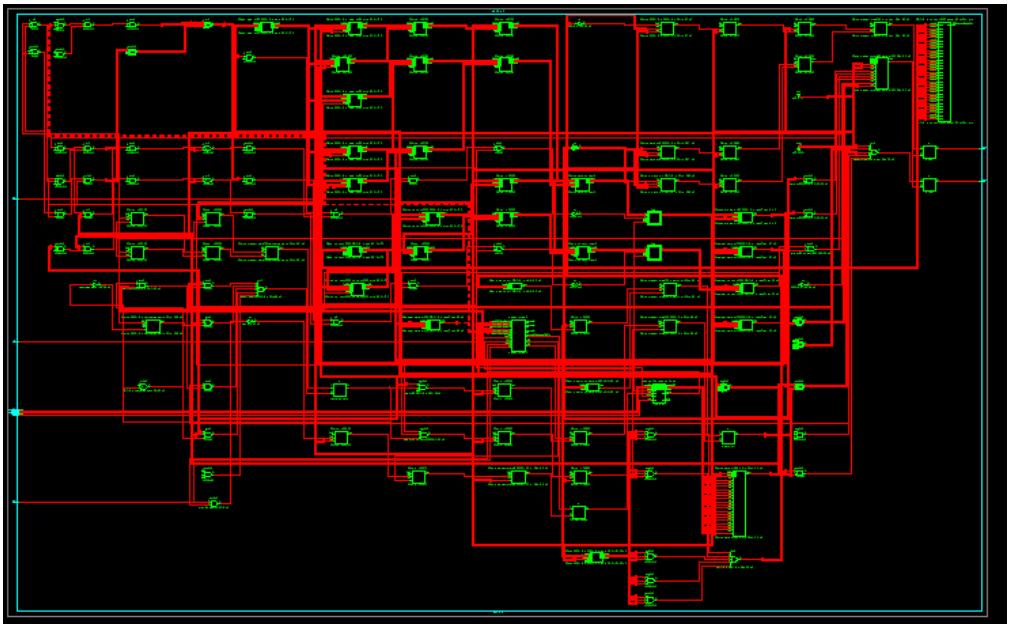

Figure 11: Netlist of the ARINC 429 Transmitter Design

7.3.4 Implementation

The implementation process maps the synthesized design onto the FPGA's resources. This step includes placement and routing, optimizing the design to fit the FPGA's architecture.

The routed design shows the final placement and routing of the FPGA resources. This step ensures that all components are correctly connected and that the design meets the timing requirements.

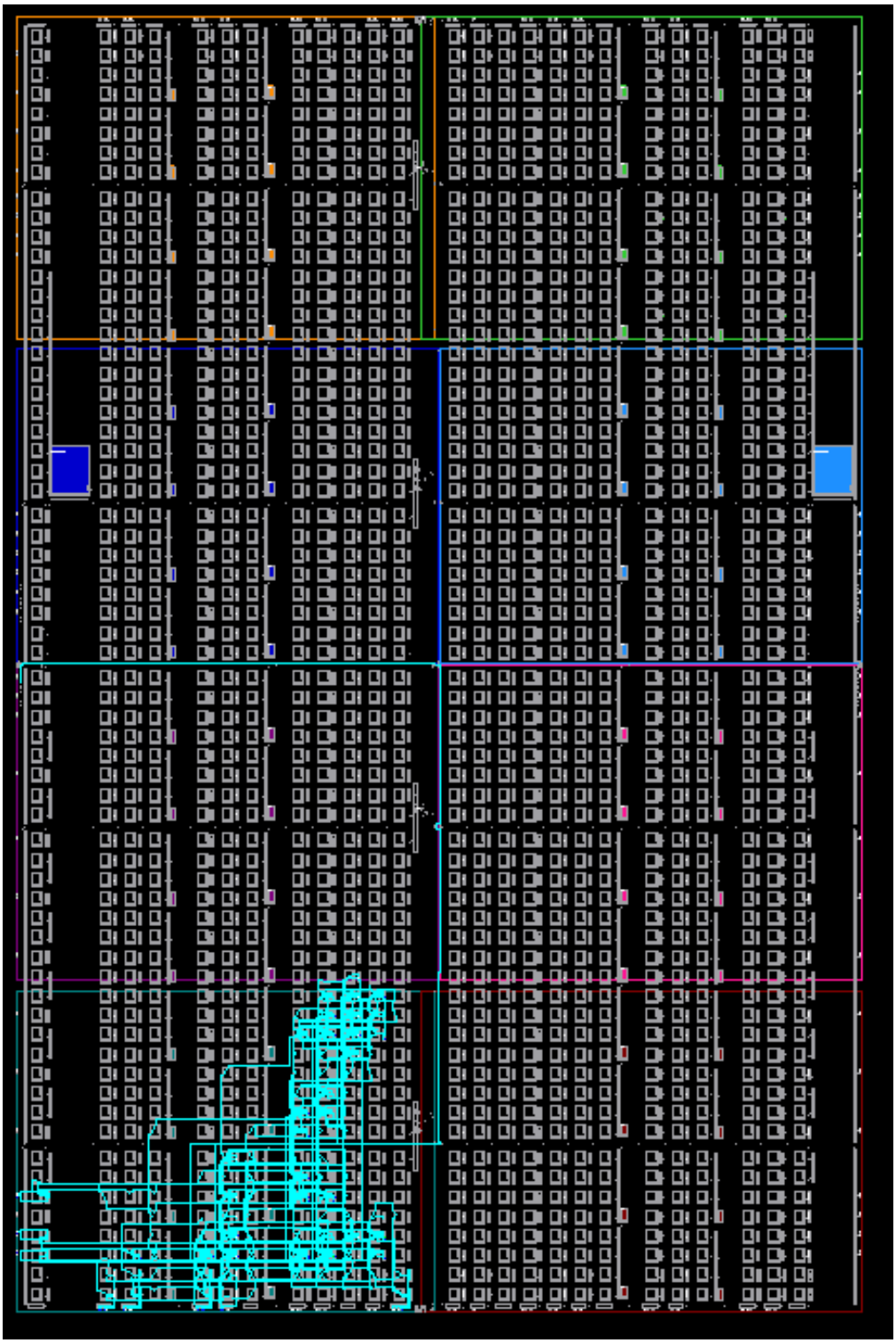

Figure 12: Routed Design in Xilinx ISE

7.3.5 Binary File Generation

After implementation, a binary file was generated for programming the FPGA. This file contains the bitstream necessary to configure the FPGA.

7.4 Summary and Resource Utilization

7.4.1 Summary

The following summary was generated by Xilinx ISE after the implementation of the ARINC 429 transmitter on the Spartan-6 FPGA:

```
a429_tx Project Status (08/14/2024 - 19:33:35)
Project File: arinc.xise Parser Errors: No Errors
Module Name: a429_tx Implementation State: Programming File Generated
Target Device: xc6slx16-2ftg256
Errors:
No Errors
Product Version: ISE 14.7
Warnings:
93 Warnings (7 new)
Design Goal: Balanced
Routing Results:
All Signals Completely Routed
```

This summary confirms that the implementation was successful, with no errors and all signals completely routed.

7.4.2 Resource Utilization

The resource utilization for the Spartan-6 FPGA is as follows:

- Registers: 57 used of 18,224 available (1% utilization)
- LUTs: 124 used of 9,112 available (1% utilization)
- slices: 45 used of 2,278 available (1% utilization)

a429_tx Project Status (08/14/2024 - 19:33:35)			
Project File:	arinc.xise	**Parser Errors:**	No Errors
Module Name:	a429_tx	**Implementation State:**	Programming File Generated
Target Device:	xc6slx16-2ftg256	• **Errors:**	No Errors
Product Version:	ISE 14.7	• **Warnings:**	93 Warnings (7 new)
Design Goal:	Balanced	• **Routing Results:**	All Signals Completely Routed
Design Strategy:	Xilinx Default (unlocked)	• **Timing Constraints:**	All Constraints Met
Environment:	System Settings	• **Final Timing Score:**	0 (Timing Report)

Device Utilization Summary				[-]
Slice Logic Utilization	**Used**	**Available**	**Utilization**	**Note(s)**
Number of Slice Registers	57	18,224	1%	
Number used as Flip Flops	54			
Number used as Latches	3			
Number used as Latch-thrus	0			
Number used as AND/OR logics	0			
Number of Slice LUTs	124	9,112	1%	
Number used as logic	109	9,112	1%	
Number using O6 output only	88			
Number using O5 output only	7			
Number using O5 and O6	14			
Number used as ROM	0			
Number used as Memory	14	2,176	1%	

Figure 13: Snapshot of FPGA Resource Utilization Report

7.5 Conclusion

The ARINC 429 transmitter design is successfully implemented on the Spartan-6 FPGA, demonstrating efficient use of FPGA resources and meeting all design requirements. Future work may explore additional features or optimizations.