AeroSim FPGA HDL

System Verilog HDL design for Artix-7 chip on Cmod-A7 35T board.

Table of Contents

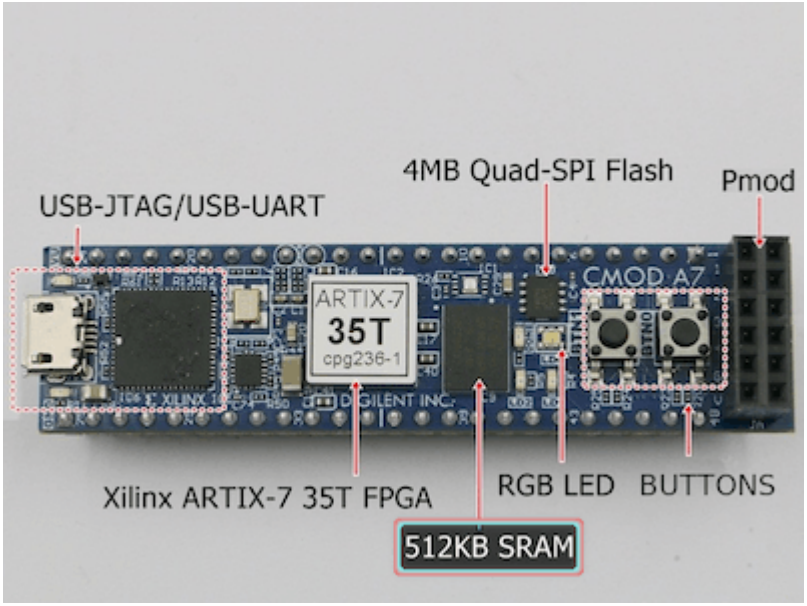
Description

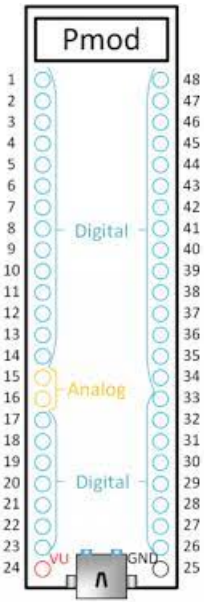
A Xilinx Artix-7 FPGA is used to implement ARINC 429 data management in the interface device. This FPGA IC is embedded on a Digilent Cmod A7-35T breadboard-compatible breakout board featuring a total of 48 I/O pins.

The FPGA board interfaces with 6 transmission (line driver) buses, each requiring 3 I/O pins for data and control, and 8 receiver (line receiver) buses each requiring 2 I/O pins for data.

Cmod-A7 Board Pinout

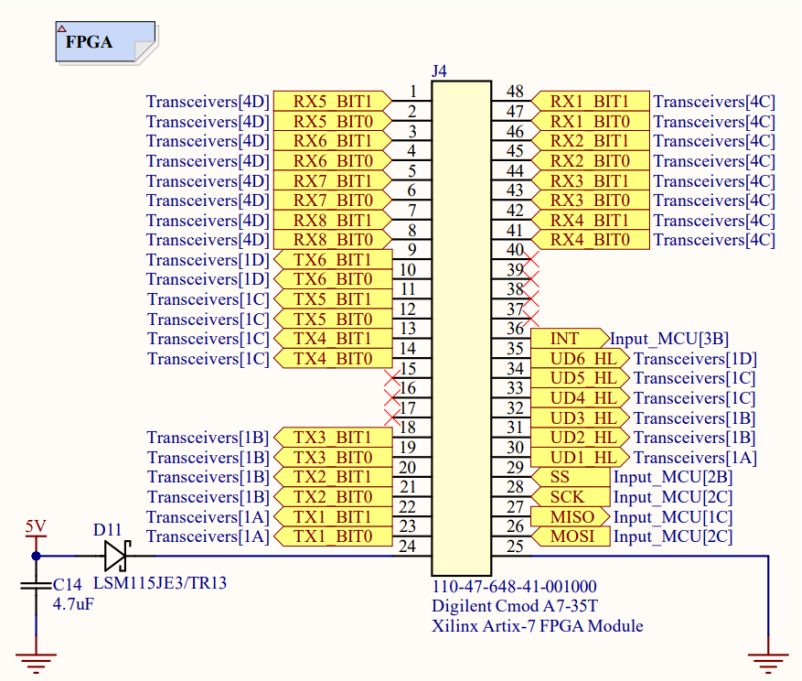
The Cmod-A7 board components and pinout are as shown below.





FPGA Board PCB Connections

The following pin connections between the FPGA board and the interface device are defined.



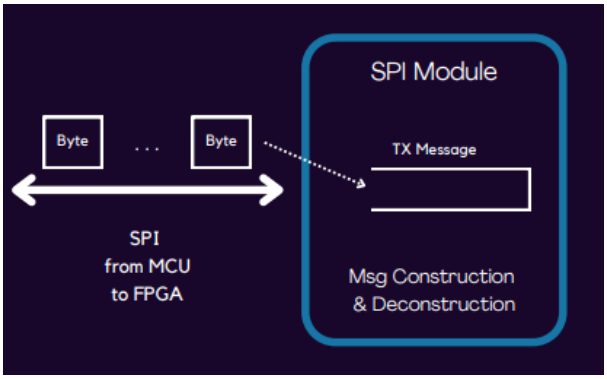
The inputs and output signals of the FPGA for the ARINC 429 interface device are defined in `xdc/ARINC420-Cmod-A7-Master.xdc`.

Design Blocks

Top Level Module

The Artix-7 FPGA top level module is defined in `ARINC429Adapter_top.sv`. This module includes the SPI slave communication logic with the external STM32 MCU as part of interface device, and reconstructs and deconstructs SPI messages. The top level module coordinates between receiving data from MCU and sending

deserialized data to MCU by asserting an interrupt when SPI communication available. Received messages from MCU are sent to the Message Decoder and messages from the RX Sending Queue are sent to MCU.



TX Datapath

The message transmission sequence from the GUI to the FPGA begins as the user toggles a global `Enable TX` GUI input after selecting one of the two ARINC 429 bitrates for each bus in use. This initializes the transmit datapath logic and configures the selected bitrates for each TX bus.

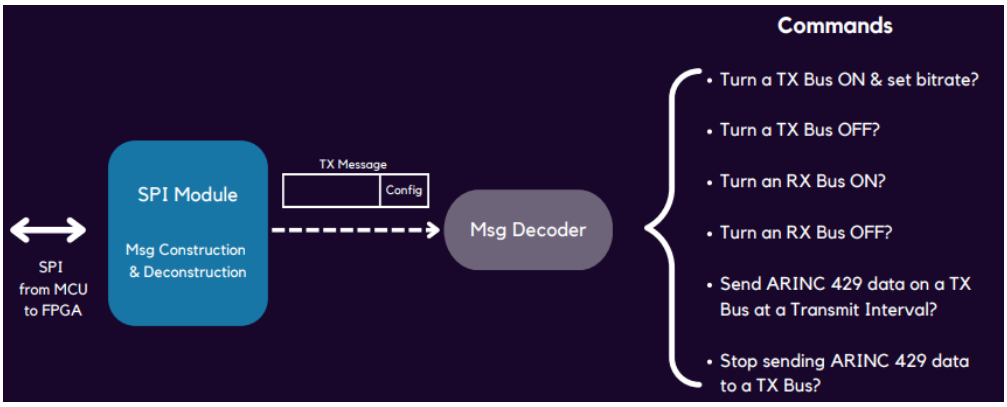
Once all TX buses are configured, the user is free to begin emulating data values for testing the connected avionics equipment.

Message Decoder

Various message types are transmitted from the user interface sofware to the FPGA such as:

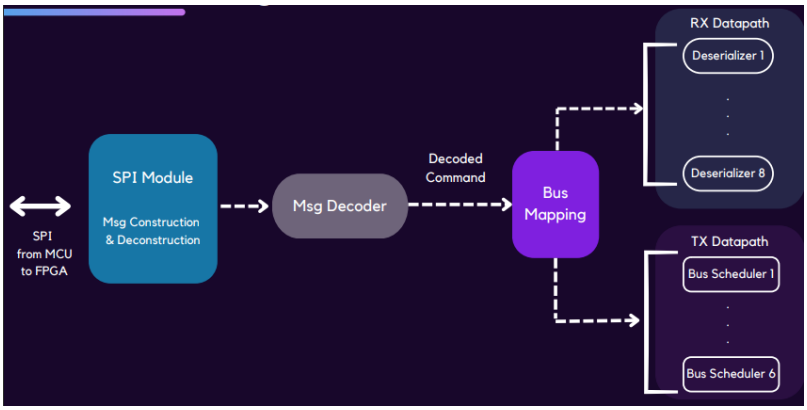
- TX ON/OFF
- Set Bus Bitrate
- Send TX Data (Continuous or Single Send)
- Stop TX Data

Each of these messages are sent to the Message Decoder module once reconstructed in the top level SPI module. Message types are identify using a defined configuration field at the start of the message. Once decoded, corresponding control signals are set for a defined number of clock cycles.



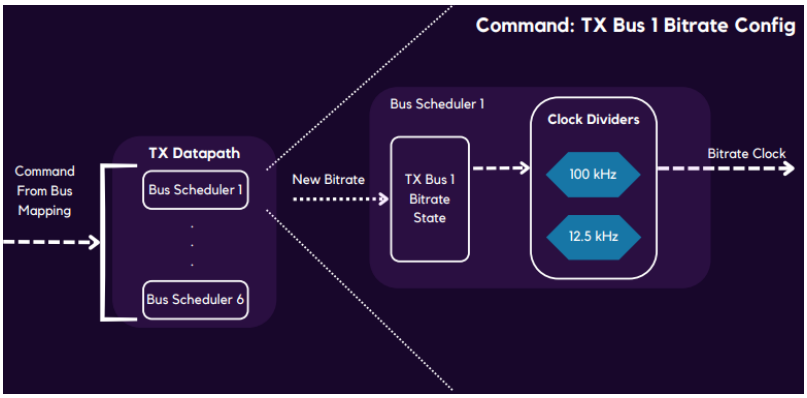
Bus Mapping

If the message has a destination TX bus specified, then it is mapped to the corresponding TX bus of interest in the TX Bus Mapper.

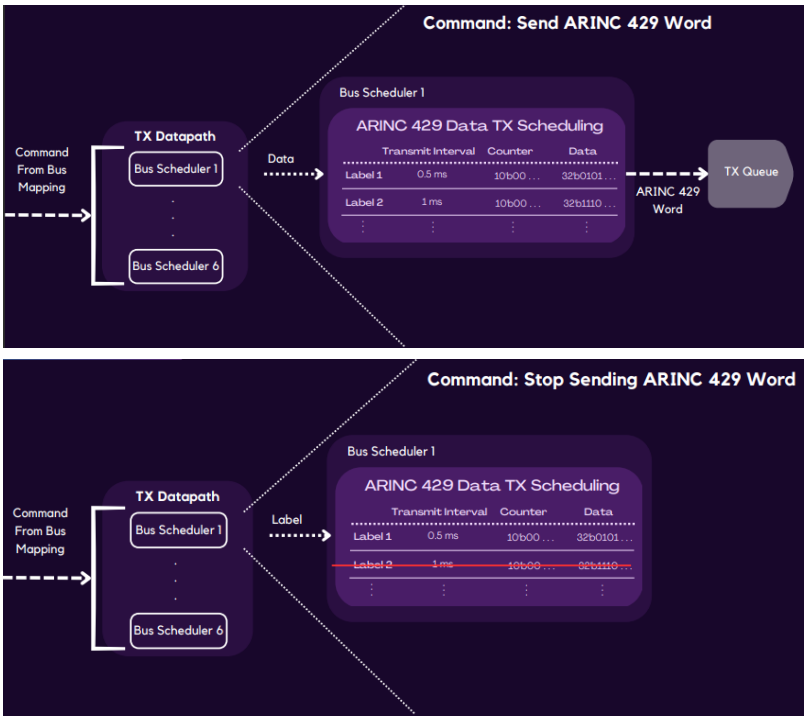


Bus Scheduling Unit

Once mapped to a TX Bus, the decoded message makes its way into the TX Bus Scheduler. The majority of the TX logic and timing of all ARINC 429 data transmitted is controlled from this module.
If the message is a *bus configuration* command, the new bitrate state is saved, and a clock divider is set ON so that upcoming data to be transmitted on this bus will be sent out at this frequency.



If the message is an *ARINC 429 data* message, the ARINC 429 word is stored in the bus scheduling unit along with its transmit interval. A counter begins, and data is sent out at interval defined by the transmit interval. If the message is a *Stop ARINC 429 data* message, the ARINC 429 data with the corresponding label from the message label field is removed from the bus scheduling unit.



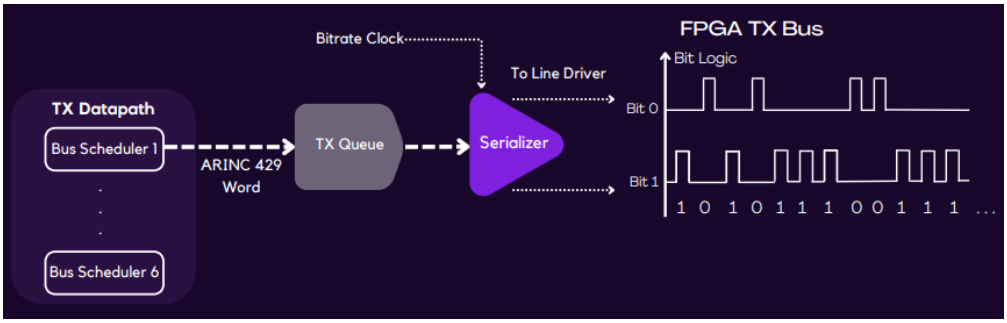
These actions are defined by control signals each asserted for 1 cycle. Control signals include: `reset_bus`, `reset_label`, `single_label`, and `add_label`. A TX bus scheduling unit may transmit up to **10 label data** at once.

TX Sending Queue

Once an ARINC 429 word in a bus scheduler is ready to be outputted, it is sent to the TX Bus Sending Queue. The Sending Queue acts as a buffer for ARINC 429 data to be outputed to avionics equipment. This queue enforces a minimum of **4 bit time delay** between ARINC 429 words sent as required by the ARINC 429 standard. ARINC 429 words are sent to the serializer when available. Control signals and communication to and from scheduling unit are also managed from this module.

Serializer

The TX Sending Queue outputted ARINC 429 word is serialized to a line driver. The serializer splits the 32-bit ARINC 429 word into 0 and 1 bits on a differential bus to flight instruments at the bitrate defined by TX Bus Scheduler. Serialization begins when the `start` flag is asserted and outputs a `processing` flag when word being processed. These control signals are set and read from the TX Sending Queue module.



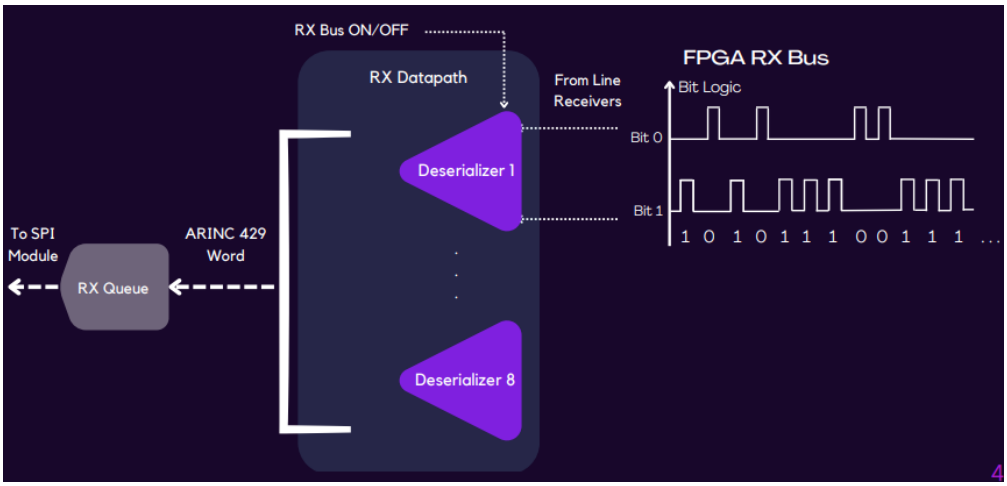
RX Datapath

In order to allow the reception of ARINC 429 data from avionics equipment, the user must first select which RX buses to listen to for received data, then toggle RX ON. A command is sent from the user interface to the

FPGA to start the intake of ARINC 429 words on the selected buses, initializing the RX datapath logic. The user may disable RX logging to stop the intake of data from avionics equipment, which is signalled by another command to the interface device.

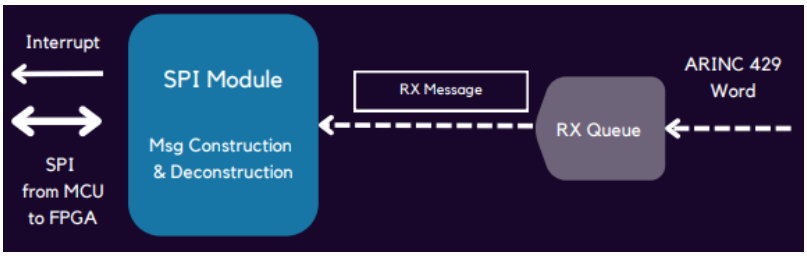
Deserializer

A deserializer is implemented for each RX Bus to accept differential bit inputs from Line receivers. This module deserializes 0 and 1 bits received from flight instruments and reconstructs received bits into 32-bit ARINC 429 word. A `data_ready` flag is asserted once 32 bits have been received. If the counter has not reached 32 bits and **2+ bit time** passed, the counter is reset to zero to avoid corrupting future data received in case of a missed bit error.
The Deserializer has single slot buffer to store a deserialized ARINC 429 word while wating for RX Queue to be ready to queue this data. Deserialized data must be queued in RX queue before next word arriving is done deserializing, otherwise will be dropped.



RX Sending Queue

Once ARINC 429 data is recorded from an RX bus and fully reconstructed into a 32-bit word, it is sent to a common RX Queue, encoded into an RX message with padding and origin RX port fields, then sent to the SPI module to be transmitted by SPI. The majority of the RX logic is controlled from this module. The RX Sending Queue coordinates the received ARINC 429 word storage in RX Queue from multiple buses at once and alternates between checking deserializer queues to store RX data. This queue also acts as a 24 slot buffer between small deserializer queues and the SPI module, while waiting for SPI access to MCU.
An interrupt line from the FPGA to the MCU is asserted as a notification that data has arrived and is ready to be transferred over the SPI bus. This ensures that the MCU, which is the master device on the SPI bus, runs a SPI transaction for the FPGA to transfer a data payload with the received data over the bus.



Dev Notes

This project was developed using the System Verilog HDL and the Vivado Design Suite 2022.2 with an IDE and simulation software of choice.

Project Structure

The suggested project directory structure is as follows:

```
AeroSimFPGA/
¦ README.md
¦ .gitignore
+---bit_bin/
+---sim/
+---src/
+---tb/
+---test/
+---vivado_proj/
+---xdc/
```

The `bit_bit/` folder contains files with .bit and .bin extensions to program the FPGA. The `sim/` folder contains any simulation related files and saved waveforms from your simulation software of choice. the `src/` folder contains all relevant System Verilog .sv files to the design. The `tb/` folder contains their corresponding testbench .sv files used for simulation purposes. The `vivado_proj/` contains your created Vivado project, containing all of the files above. As file path is of importance, the user must create a new Vivado project on their own device. The `/xdc` folder contains the pin and I/O definitions for the FPGA board.

Vivado

To install Vivado Design Suite, follow the installation instructions from Xilinx.
Note that Vivado Design Suite may require 20-60 GB of space to install depending on the version and packages installed. Ensure that the version is compatible with your version of Windows 10 or 11.

Project Setup

To create a new Vivado project for the Cmod-A7 board, create the new project in `vivado_proj/` and follow the Diligent Programming Guide.

Diligent Board Files

Digilent provides board files for each FPGA development board. These files make it easy to select the correct part when creating a new project and allow for automated configuration of several complicated components used in many designs.
To install Diligent's Board Files for the Cmod-A7 on Vivado, follow Diligent's Reference Manual.

Program Files

To program the FPGA, program files .bit and .bin must first be generated. Follow steps 2-2.10 from Diligent's Programming Manual to generate these files.

The Cmod-A7 can be programmed using JTAG by communicating through the micro-USB port, or by Quad SPI. For development purposes, JTAG is recommended as it is faster, and micro-USB connection supplies both power to the board, and the ability to program it. Step 3 of Diligent's Programming Manual describes the process to program the board by JTAG using the generated bitstream .bit file.

Programming the board by Quad SPI Flash non-volatile memory allows the FPGA chip to load the design automatically on startup without the need to connect the board by micro-USB. This programming method is best used when the design is finalized or ready to be demonstrated. Step 4 of Diligent's Programming Manual describes the process to program the board by Quad SPI Flash using the generated binary .bin file. Note that in part 4.2), the Configuration Memory part is `mx25l1327f` while the name is `n25q32-3.3v-spix1_x2_x4`. To boot the FPGA chip on start up, power cycling the board is required. The board will only load the design when connected to an external power source other than micro-USB. If the board must still be powered from the micro-USB, user will need to manually select the option to boot program from the configuration memory device on Vivado.

Design Simulation

To simulate the design and tesbenches, ModelSim 10.5b was used, however any prefered waveform simulator may be used. Vivado Design Suite also offers an internal waveform Simulator.
To use the Vivado Waveform Simulator:

- Add .tb files as simulation sources
- Add .sv files as design sources
- Hierarchy update - set top level module
- Run synthesis to remove errors
- To add signals to waveform window:
    - Click on DUT in scope window
    - Select signal in objects
    - Add to wave window

These steps are further described in Diligent's Vivado Simulator Manual

FPGA Tester

The file `\src\ARINC429FPGATester_top.sv` is an FPGA tester module unused in the hardware interface design. It is only used as a stand-alone FPGA module to test features on the FPGA in isolation without the rest of the interface device. When testing the FPGA using this module, the top-level module definition in the Vivado project must be updated to this file. A different pin definition file `/xdc/Cmod-A7-Master.xdc` is created to be used with this new top-level module if a different pinout is used. This file must also replace the original interface design pinout in the Vivado project.

Additional Resources

Getting Started With Vivado

Cmod-A7 Getting Started

Diligent Cmod-A7 Manual

Diligent Reference Manual: Vivdao and Board Files Installation

Diligent Reference Manual: Cmod-A7 Programming Guide

Diligent Reference Manual: Using Vivado Waveform Simulator