OA-II Backplane Bus System Design

DR00001

Rev: A03

Jinzhi Cai

2019–08–14

Contents

1 Introduction

1.1 Scope

This document analyzes the requirements for OA-II VEH system data transmission, and current bus technologies in the field, and to outline a system design to fullfill the needs of OA-II VEH system.

1.2 Purpose

The goal for the OA-II backplane bus system is to construct a high speed, high compatibility, and highly robust backplane data transmission system.

2 Revision History

Rev#	Editor	Delta	Date
A01	Jinzhi Cai	Initialize	2019-7-19
A02	Jinzhi Cai	Add detail Ethernet	2019-7-22

Table 1: Summary of Revision History

3 BUS System Requirement

3.1 Hardware Requirement

Backplane Bus The bus needs to suppport a swappable module.

Vribation-proof The bus needs to have good support to the module on the frame.

Size Needs to fit into the dimensions of the rocket.

Topology The hardware implementation needs to support out-of-order locating.

3.2 Software Requirement

Point to Point & Broadcast The bus need to support broadcasting data.

Bandwidth The bus needs to support the max bandwidth given.

Topology The bus needs to allow for changes in software topology.

Real Time The bus needs to support message priority levels.

Various Speed The bus needs to allow low end device connectivity into the system.

3.3 Bandwidth Calculation

Low Speed Payload Each low speed payload is sensing in 10kHz 16bit

- 4 high pressure sensors for propulsion system
- 2 low pressure sensors for pitot tube
- 4 high temperature sensors for propulsion system
- 4 low temperature sensors for electronics
- 4 low temperature sensors for batteries
- 2 low temperature sensor for ambient

$$4 + 2 + 4 + 4 + 4 + 2 = 20 chennals$$
$$10kHz = 10000Hz$$
$$16bit = 2byte$$
$$10000Hz \times 2byte = 20000byte/s = 20Kbyte/s$$
$$20Kbyte/s \times 20 = 400Kbyte/s$$

High Speed Payload

- 9 axis IMU
- GNSS
- 4x cameras

9 axis IMU in 10kHz is

$$9 \times 10000Hz \times 2byte = 180000byte/s = 180Kbyte/s$$

GNSS module[1]
UTC launch time 4byte
Latitude 4byte
Longitude 4byte
Height 4byte
Direction+Ground speed 4byte

$$4byte \times 5 = 20byte$$
$$10Hz \times 20byte = 200byte/s$$

Camera, set the bitrate to 8Mbps[2]

$$8Mbps = 1Mbyte/s$$
$$1Mbyte/s \times 4 = 4Mbyte/s$$

Total bandwidth

$$(180Kbyte/s + 4Mbyte/s + 200byte/s + 400Kbyte/s) \times 2 \approx 10Mbyte/s$$

Device	Number	Bandwidth per Device	Total Bandwidth
High Pressure Sensor	4	20KB/s	80KB/s
Low Pressure sensors	2	20KB/s	40KB/s
High Temperature Sensors	4	20KB/s	80KB/s
Low Temperature Sensors	10	20KB/s	200KB/s
9 axis IMU	1	180KB/s	180KB/s
GNSS	1	200B/s	200KB/s
Cameras	4	1MB/s	4MB/s
Estimate Bandwidth	-	-	10MB/s

Table 2: Summary of Estimate Bandwidth

[1] Did not include any fixing factor
[2] High bitrate is nessary for high virbation environment

4 Current Bus Analyze

4.1 I2C

I2C is a serial protocol for two-wire interface to connect low-speed devices like microcontrollers, EEPROMs, A/D and D/A converters, I/O interfaces and other similar peripherals in embedded systems. It was invented by Philips and now it is used by almost all major IC manufacturers.[1]

I2C is a great low speed communication bus, however it do not support hardware priority level and change software topology.

4.2 SPI

Serial Peripheral Interface (SPI) is an interface bus commonly used to send data between microcontrollers and small peripherals such as shift registers, sensors, and SD cards.[2]

Serial Peripheral Interface allows the device to increase its bandwidth by increasing the data clock rate. However, it also does not support hardware priority level and change software topology.

4.3 UART

A universal asynchronous receiver-transmitter is a computer hardware device for asynchronous serial communication in which the data format and transmission speeds are configurable.[3]

UART bus does not require clock line to transmit data. It also has different bitrate that is alterable by the device. However it is a point to point communication system, so it needs to switch for more than two devices. It is also too low bandwidth to meet the requirements.

4.4 CAN

A Controller Area Network (CAN bus) is a robust vehicle bus standard designed to allow microcontrollers and devices to communicate with each other in applications without a host computer.[4]

The CAN bus have hardware priority level and support 500kbps[3] bandrate. It also allows group broadcast as well as point to point communication.

[3] About 62.5Kbyte/s

4.5 PCIe

PCI Express, officially abbreviated as PCIe or PCI-e, is a high-speed serial computer expansion bus standard, designed to replace the older PCI, PCI-X and AGP bus standards. It is the common motherboard interface for personal computers' graphics cards, hard drives, SSDs, Wi-Fi and Ethernet hardware connections.[5]

The PCI Express is a common use buses in personal computer. However, the topology of this bus is mostly tree structure. It will increase difficulty when a second master need to add into the system.

4.6 RapidIO

The RapidIO architecture is a high-performance packet-switched interconnect technology. RapidIO supports messaging, read/write and cache coherency semantics.[6]

RapidIO is a high speed connection that supports up to 5Gbps[4] via a single lane. It also supports multi-master structure. By using a RapidIO switch, the system could change software topology. However, rapidIO is a new bus technology that is mainly used in DSP, high speed FPGA, and SoC. It require heavy hardware resources compared to the other buses.

4.7 SpaceWire

SpaceWire is defined in the European Cooperation for Space Standardization standard ECSS-E-ST-50-12C (replaces ECSS-E50-12A). The SpaceWire standard was authored by Steve Parkes, University of Dundee with contributions from many individuals within the SpaceWire Working Group from European Space Agency (ESA), European Space Industry, Academia and NASA.[7]

The SpaceWire is use LVDS voltage standard which is a commonly use voltage standard in FPGA. The PHY for SpaceWire is relativly simple and require less resource for constructe the PHY. The newest SpaceWire bus support 400Mbps for one lane[5].

4.8 Interlaken

Interlaken was invented by Cisco Systems and Cortina Systems in 2006, optimized for high-bandwidth and reliable packet transfers. It builds on the channelization and per channel flow control features of SPI-4.2, while reducing the number of integrated circuit (chip) I/O pins by using high speed SerDes technology.[8]

Interlaken is a bus deisgn for replacing ethernet. It supports port division and flow control,

[4] About 625Mbyte/s
[5] About 50Mbyte/s

however, the Interlaken PHY does not support in most of the FPGA outlined.This means that it will require an extra chip for PHY. This has similar speed to RapidIO[6].

4.9 Ethernet

Ethernet as a widely used bus has many benefits to the data bus system, as its standard is implemented in many devices and drivers, a router for it is easy to develop. It also can go up to 10MB/s with correct PCB design. However, it does not have a real time clock as SpaceWire does and is not as fast as the RapidIO or PCIe. Transmission via Ethernet is not designed for realtime applications.[9]

[6] About 625Mbyte/s

5 Recommend System Design

5.1 General Structure

Based on the requirements of the bus, the OA-II Bus System will include two parts, the command bus which is in charge of the initialization of another bus and then prioritizes data transmission, the data bus which incharge the transmission of a large amount of data and of programs. The job for those two buses will be different during each stage.[7]

5.1.1 Initialization

Command Bus **ON**
Data Bus **Configuration**

In the initialization stage, OA-II VEH COM MCU(Main Control Unit) will sense all the on board devices and run a check to examine all devices and ensure they are functioning correctly. During this stage, all the commands and the replies will happen in the Command bus, and each device will start to configuring the data bus interface. The final checking message will send from the MCU via the command bus and reply by each device via data bus.

5.1.2 Launch Ready

Command Bus **ON**
Data Bus **ON**

After the initialization stage, the whole vehicle will be into launch ready stage. In this stage, all the ignition process will download to all the unit via command bus and waiting for synchronize ignition signal. All the sensors will running at full speed and deliver data into MCU to monitor vehicle status via the data bus.

5.1.3 During Flight

Command Bus **StandBy**
Data Bus **ON**

After the ignition, all the sensors will be running at full speed and deliver data into MCU to monitor vehicle status and recording via the data bus. Command Bus will be in the standby mode to be ready to send an emergency message to passby.

[7] More detail in this section at ES00007 OA-II Payload Bus Specifications

5.1.4 Post-Flight

Command Bus **StandBy**
Data Bus **OFF**

OA-II VEH COM MCU(Main Control Unit) After landing, the MCU will again do a final check of all existing units via the Command bus before turning all sensor units' power off. The data bus will be off during the whole landing process.

5.1.5 Emergency Backup

Scenario 1 Command Bus Failure
When COM FRU detects a command bus failure, it will cutoff the control of the COM MCU and use data bus send out a Failure Alert to all the existing units. From this moment, the data bus will only transmiting critical data and commands. Other data units will be powered off or save to unti internal storage.

Scenario 2 Data Bus Failure
When COM FRU detect a data bus failure, it will use command bus to communicate with MCU and checking the MCU status. The mission will continue and all units will try to switch to secondary data bus which will running at a lower speed.

5.1.6 Bus Data Summery

	From: COM	From: TAM	From: PAM
To: COM	X	Data Bus	Data Bus
To: TAM	Command Bus/Data Bus	X	Data Bus
To: PAM	Command Bus	Data Bus	X

Table 3: Communication route

Command Bus	Data Bus
Need realtime transmission	Not mission critical
Mission critical	high bitrate data
Small size data	Research data
Send from COM	Base station information

Table 4: Bus Data Catalog

5.2 System Redundancy

The command bus redundancy is accomplished by adding an additional bus. The primary and secondary command will send the same command at the same time and each device will only receive the one that reaches them first and use the second one for error checking.

Each unit will have a two lane connection to each data bus router. Those two buses will send and receive the same information, each device will only receive the one that reached them first and use the second one as error checking.
During the Initialization, the primary data router will synchronize configuration with the second data routing core. When the configuration is finished, the link will be disabled.

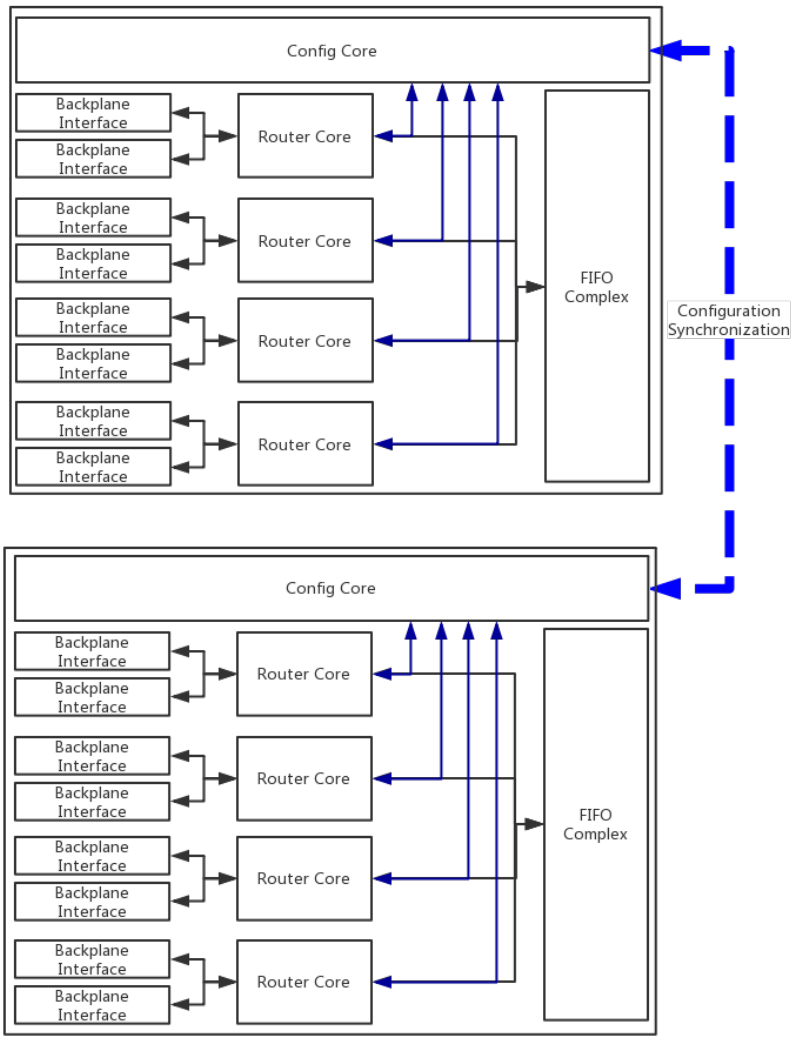

Figure 1: Data Router Structure

5.3 CAN + SpaceWire

Command Bus CAN

Data Bus SpaceWire

In this plan, the command bus is using CAN. It provide prioritized data transmission for small amounts of data transmission, up to 500kbps bandwidth, which is enough bandwidth for command data transmission. The data bus is using SpaceWire which provides enough bandwidth for this high speed data transmission. It also includes a timestamp feature that will help in future sensor fusion. The whole system is build based on two routers, one is the main router that will be used for the major data transmission and the other is a backup router which can take over in the event the main router fails.

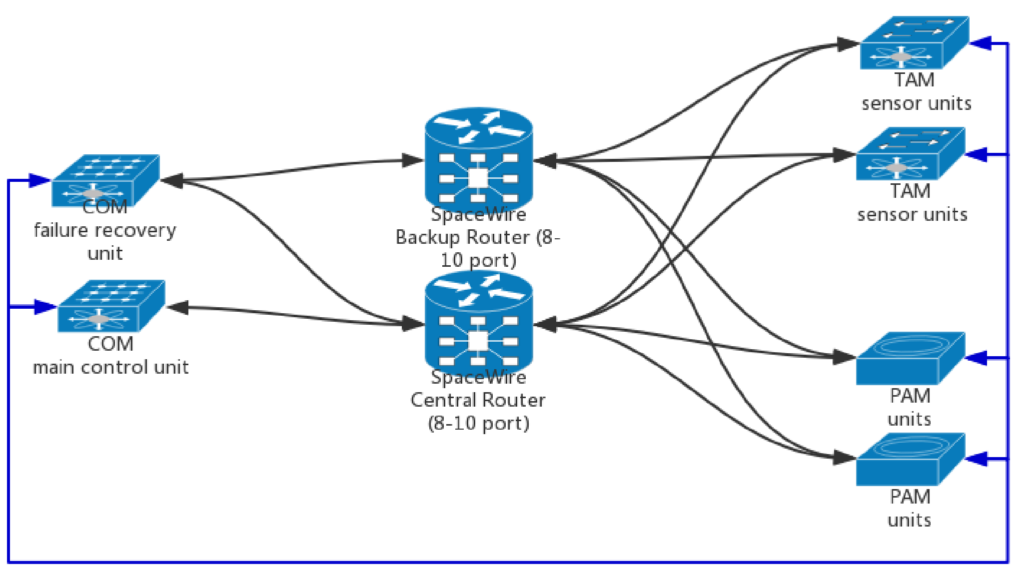

Figure 2: Recommand Design A

5.4 CAN + RapidIO

Command Bus CAN

Data Bus RapidIO

In this plan, CAN bus still is the main command bus due to its prioritized data transmission. In the data bus, RapidIO each lane can carry 10 times data rate compared to SpaceWire. It can handle most future advanced missions without any hardware upgrades. A company will also be providing ASIC for rapidIO bus router.

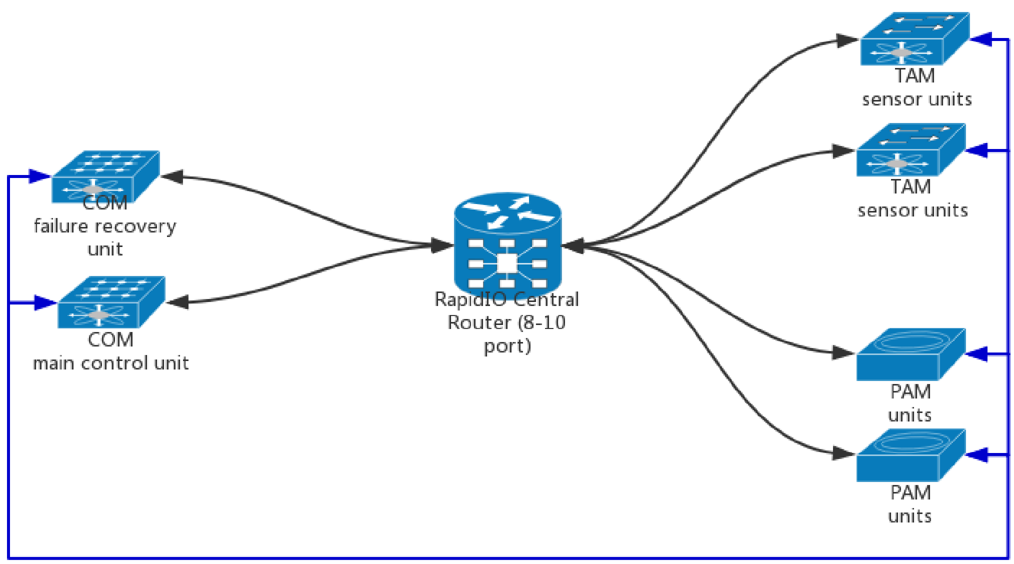

Figure 3: Recommend Design B

6 Bibliography

[1] aaronRong, *I2C*, 2019. [Online]. Available: `https://www.cnblogs.com/aaronLinux/p/6218660.html`.

[2] Deng Tao, *SPI*, 2019. [Online]. Available: `https://www.cnblogs.com/deng-tao/p/6004280.html`.

[3] zjy900507, *UART*, 2019. [Online]. Available: `https://blog.csdn.net/zjy900507/article/details/79789671`.

[4] CANbus, *CAN*, 2019. [Online]. Available: `https://www.canbus.us/`.

[5] Margaret Rouse, *Peripheral Component Interconnect Express*, 2019. [Online]. Available: `https://searchdatacenter.techtarget.com/definition/PCI-Expressd`.

[6] RapidIO.org, *RapidIO*, 2019. [Online]. Available: `http://www.rapidio.org/#`.

[7] ESA, *SpaceWire - Key Features*, 2017. [Online]. Available: `http://spacewire.esa.int/content/Home/KeyFeatures.php`.

[8] honeypot, *Interlaken*, 2019. [Online]. Available: `http://blog.sina.com.cn/s/blog_43c625f101017otd.html`.

[9] qq_5072517, *Ethernet*, 2019. [Online]. Available: `https://blog.csdn.net/qq_25072517/article/details/78478968`.