



XAPP1130 (v1.0) March 20, 2009

Architecting ARINC 664, Part 7 (AFDX) Solutions

Author: Joel Le Mauff and Jeff Elliott

Summary

Each new generation of commercial aircraft has grown more complex, especially with the heavy reliance of fly-by-wire and the associated avionics. As more systems are designed into airframes, traditional point-to-point wiring schemes are no longer practical. The designers of the Airbus A380 searched for a solution to reduce the amount of wiring, increase bandwidth, and make use of commercial-off-the-shelf (COTS) technology where possible. ARINC Specification 664 (ARINC 664), Part 7 is the result of that search.

This application note provides users with a detailed overview of the architecture and function of avionics full-duplex switched Ethernet (AFDX) as defined in the specification ARINC 664, Part 7 ([\[Ref 1\]](#)). In addition, a detailed description of how various functional blocks required for an AFDX end system can be mapped to both the Virtex®-4 and Virtex-5 architectures is included.

ARINC 664 Overview

AFDX combines concepts taken from asynchronous transfer mode (ATM) and applies them to a variant of IEEE Std 802.3 (Ethernet). At the physical layer, AFDX consists of a star-topology, full-duplexed switched Ethernet (either 100BASE-TX or 100BASE-FX). This topology eliminates the collision issue found in half-duplexed Ethernet.

Note: For background on the development of AFDX, see [“Appendix A: Background,” page 24](#).

In addition, the network is profiled. In an airframe, all connection, addressing, and bandwidth requirements for the entire network are known in advance. Each part of the network can be tailored to the specific connection. The network profile is updated when there are any upgrades and changes to the electronics of the aircraft.

At the protocol level, AFDX creates the concept of a virtual link (VL) — a point-to-point or multicast connection through the network. The VL mimics the unidirectional connections found in ARINC Specification 429 [\[Ref 2\]](#) (see also [“ARINC 429,” page 24](#)). Again, as the network is profiled, the addressing and bandwidth requirements of each VL is defined in advance.

Moreover, the network is deterministic with the latency for each connection known in advance. The traffic flow and shaping mechanisms help guarantee the latency, jitter, and bandwidth for each link, providing the QoS required for avionics systems.

The last issue to be addressed is robustness. AFDX relies on parallel, redundant networks to provide an additional level of fault tolerance. Each data packet is sent across both networks simultaneously. Redundancy management mechanisms ensure that only one copy of each packet is transmitted, and that sequential order of the packets is maintained.

ARINC Specification 664 is divided into eight parts:

- Part 1, Systems Concepts and Overview [\[Ref 3\]](#)
- Part 2, Ethernet Physical and Data Link Layer Specifications
- Part 3, Internet-based Protocols and Services
- Part 4, Internet-based Address Structures and Assigned Numbers
- Part 5, Network Interconnection Services and Functional Elements
- Part 6, Reserved

- Part 7, Avionics Full Duplex Switched Ethernet (AFDX) Network
- Part 8, Upper Layer Services

AFDX is defined by Part 7, along with IEEE Std 802.3 (Ethernet standard).

AFDX – The Details

Network Topology

An AFDX network consists of up to 24 end systems connected to a switch (Figure 1). Switches can be cascaded to increase the capacity of the network. Total switch capacity is limited to 4,096 VLs (including the routing of VLs either originating or terminating beyond end systems connected to that switch).

Note: There is no explicit limit on the number of VLs an end system can support. The maximum number is a function of the amount of required bandwidth of each VL and its maximum frame length.

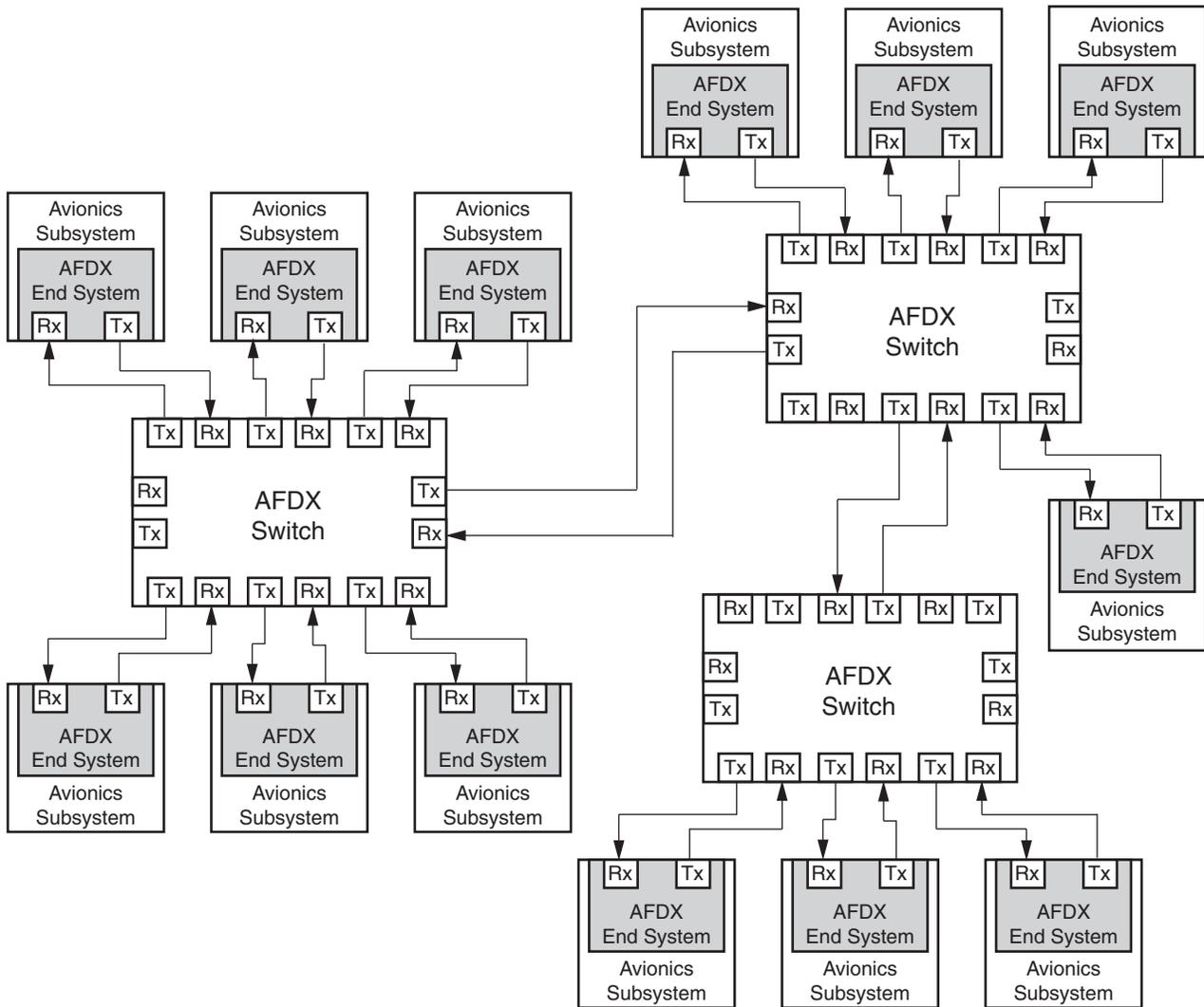


Figure 1: AFDX Topology (Redundancy Not Shown)

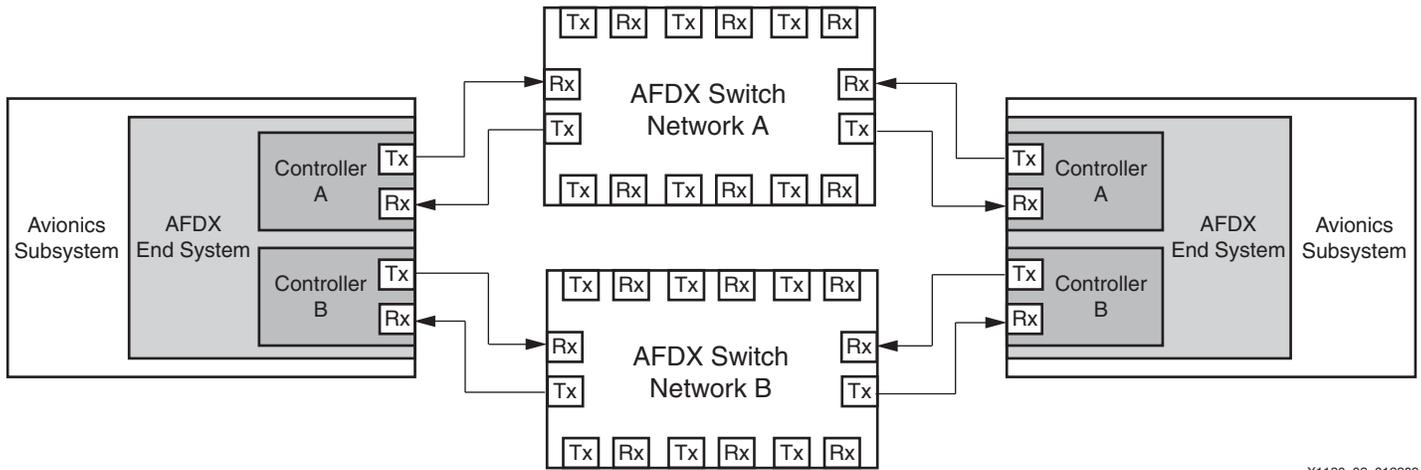
The network is profiled — all routes and addressing are predefined and contained in the configuration for both end systems and switches, simplifying network configuration. Transmitting end systems are responsible for enforcing bandwidth limits, and receiving end systems manage redundancy. Switches are responsible for routing frames, policing bandwidth, and shaping traffic.

There is no intra-switch communication (other than passing data frames) between redundant or cascaded switches. All routes are based upon the switch’s routing table.

The standard (ARINC 664, Part 7) also allows for the mapping of other protocols over AFDX. For example, ARINC 429 links can be built across an AFDX network through the use of concentrators/protocol conversion modules.

Redundancy

To increase the robustness of the system, an AFDX network consists of two redundant networks; each end system has two Ethernet ports (A and B), with A ports connected to switch A and B ports, which are connected to switch B (Figure 2). Identical frames are sent by the end system on both ports simultaneously. Each switch routes their frames independently to the destination end systems. The receiving end system is responsible for managing the reception of redundant frames, deleting duplicates and any out-of-order frames.



X1130_02_012309

Figure 2: AFDX Redundancy

Note: Redundancy is not required for all VLs and can be turned off for a given VL, provided a thorough evaluation of the impact is completed.

Frame Format

The AFDX frame format (Figure 3) is compliant with IEEE Std 802.3 (Ethernet). The frame contains addressing for identifying source and destination end systems as well as the assigned virtual link. AFDX frame length can vary from 64 to 1518 bytes (plus a 7-byte frame preamble, 1 frame start byte, and 12-byte interframe gap (IFG), with a data payload between 1 and 1471 bytes (payload must be padded to a minimum length of 17 bytes).

AFDX Frame											
7 Bytes	1 Byte	6 Bytes	6 Bytes	2 Bytes	20 Bytes	8 Bytes	1–1471 Bytes	0–16 Bytes	1 Byte	4 Bytes	12 Bytes
Preamble	Start Frame Delimiter	Destination Address	Source Address	0x800 IPv4	IP Structure	UDP Structure	AFDX Payload	Padding	SN	Frame Check Sequence	Inter-Frame Gap

X1130_03_030809

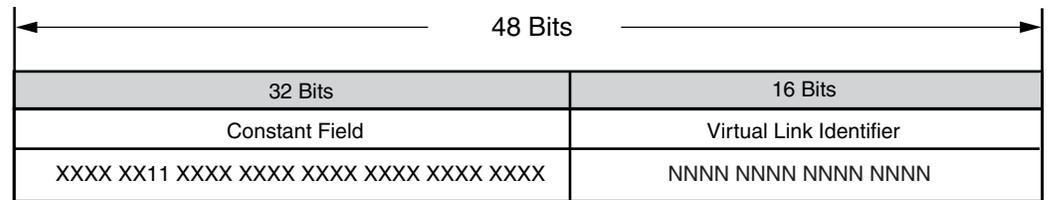
Figure 3: AFDX Frame

The one-byte frame sequence number is used to maintain ordinal integrity for frames of a given VL as well as assist in detecting missing frames. During transmission, the sequence number is incremented by one for each VL frame, starting at 0 and wrapping at 255 to 1.

Note: A sequence number of 0 is used to indicate a reset condition of the transmitting end system.

Addressing

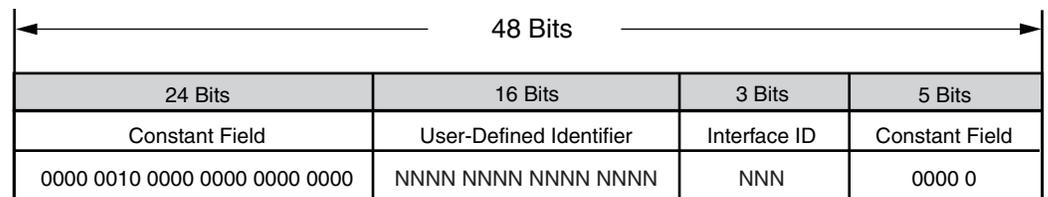
At the data link layer, each VL is assigned a MAC address by the system integrator. The 48-bit MAC destination address (Figure 4) consists of 32 bits to constant field (identical for all end systems in the network) and 16 bits to identify the VL. AFDX frames are routed by the switch to all destination end systems identified for the VL in the switch configuration.



X1130_04_012309

Figure 4: MAC Destination Address Format

The 48-bit MAC source address (Figure 5) identifies the Ethernet controller of the end system originating the frame. The first 24 bits of the address are set to a constant value. Following the constant value is a 16-bit unique identifier for the controller set by the system integrator (ARINC 664 provides only general guidance on setting this value). Following the 16-bit unique identifier is a 3-bit value used to identify which network the controller is connected to (001 for network A and 010 for network B — all other values are not used). The final 5 bits are set to a constant: 0 0000.



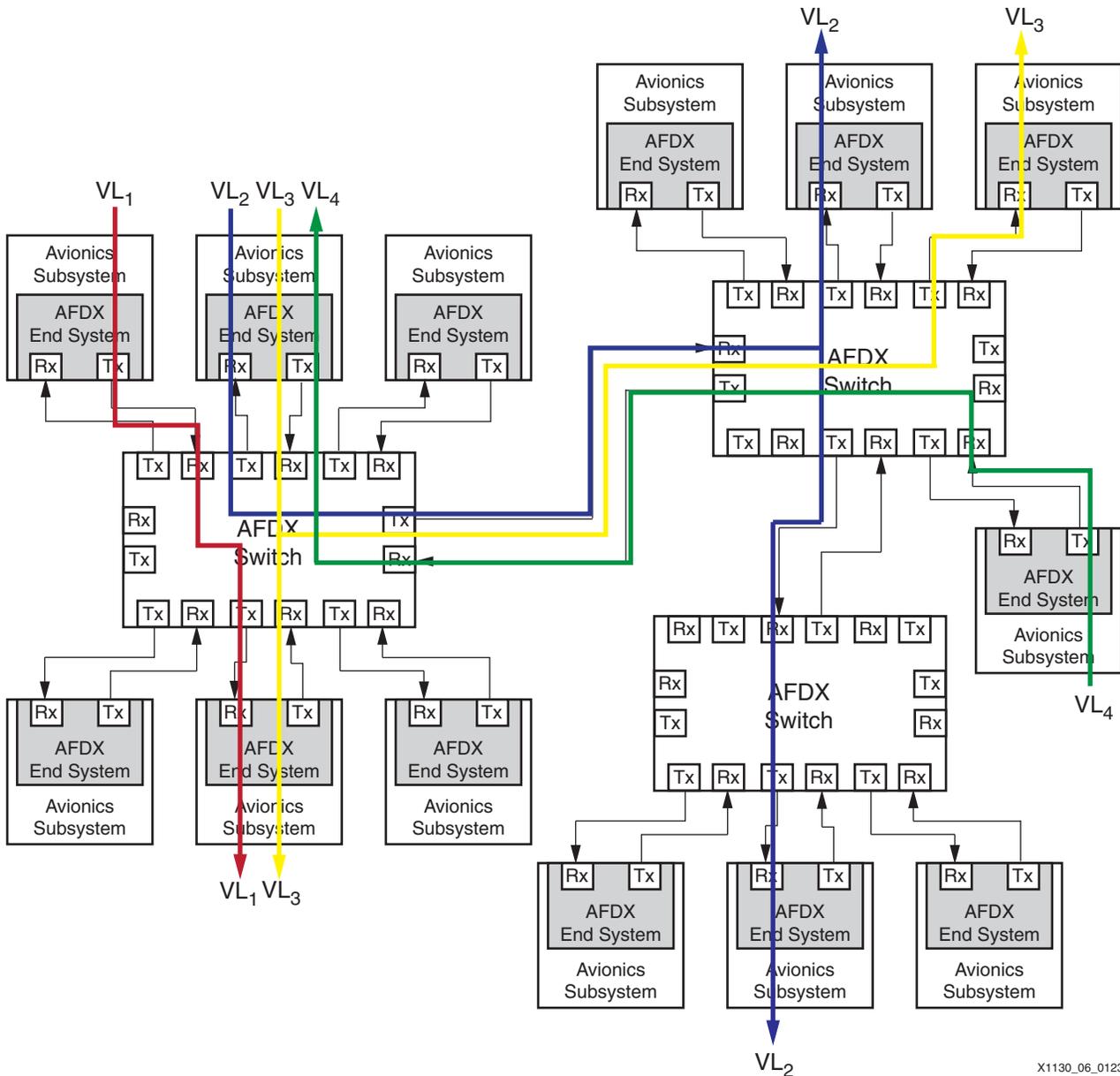
X1130_05_012309

Figure 5: MAC Source Address Format

Note: The focus of this application note is on the data link and physical layers. For more information about IP addressing within the network, refer to the ARINC 664, Part 7 standard.

Virtual Links

The goal of ARINC 664, Part 7, is to preserve point-to-point links while reducing the amount of wiring. The physical point-to-point links of ARINC 429 [Ref 2] are replaced by virtual links, connecting sensors and actuators with control units (Figure 6). VL links are time-division multiplexed at the end system for transmission over the network.



X1130_06_012309

Figure 6: Virtual Links over an AFDX Network

Each VL is guaranteed a specific *maximum* bandwidth as well as an end-to-end maximum latency. The assigned bandwidth is controlled by the end system and enforced in the switch, where the latency is defined by the system integrator, bounded by the limits set in the standard (see “[Latency](#),” page 8). In addition, a VL is assigned a maximum allowed frame size of L_{MAX} .

Note: In the switch specification section of the standard, L_{MAX} is referred to as S^{MAX} . In addition, that section specifies a minimum allowed frame size S^{MIN} for each VL (see “[Frame Filtering](#),” page 10).

The total of all bandwidth assigned to VLs cannot exceed the total bandwidth available in the network. Additionally, the demands on bandwidth at each switch must be known because each switch must handle VLs originating and terminating at attached end systems *and* any VLs being forwarded to other switches in the network.

Each VL can be composed of up to four sub-VLs. Sub-VLs are used to handle less critical data with less stringent bandwidth requirements (bandwidth guarantees apply only at the VL level). Data queues for each sub-VL are read in a round-robin fashion, with each frame containing data

only from one sub-VL queue (any fragmentation has to be handled at the IP layer). After a frame for a sub-VL is created, that frame is handled by the network no differently than a VL frame.

Note: Sub-VLs are an optional implementation, available to the end user as needed. Moreover, the standard does not specify how sub-VLs are identified. Possibly a unique VL identifier can be assigned to each sub-VL by the system integrator.

End Systems

Virtual Link Management

The primary responsibility is the management of transmitting and receiving data for the virtual links. An end system can handle a maximum of 128 VLs and can be built to any needed configuration, for example, to transmit four VLs and receive six VLs, with one receive VL being composed of three sub-VLs. A one-size-fits-all design is not required.

For each VL and sub-VL, the end system must maintain a FIFO queue (sub-VLs FIFO queues are read in a round-robin fashion to fill its assigned VL FIFO queue) — ordinal integrity of transmitted frames must be maintained. The size of the VL/sub-VL queues is not specified by the ARINC 664, Part 7, but the total of all queues for a given application (or partitions as defined by ARINC Specification 653 [Ref 4]) must be at least 8 kB (an application or partition can have one or more VLs).

For transmission, the end system is responsible for:

- Reading each VL queue.
- Incrementing the VL frame sequence number.
- Scheduling each frame for transmission to maintain the bandwidth guarantee within the allowed jitter.
- Transmitting redundant frames on both controllers A and B.

On reception, the end system is responsible for:

- Deleting redundant frames and policing ordinal integrity.
- Separating data by VL and writing received frames to the appropriate queue.

Note: The end system must continue to transmit frames even if there is a link failure.

For a redundant VL reception, an end system should:

- When redundancy management is active, pass one copy of redundant data to the partition (see “Redundancy Management”).
- When redundancy management is *not* active, pass both copies of redundant data to the partition.

For a non-redundant VL reception, the end system should pass data from either channel to the partition (redundancy management can be active or not).

Bandwidth Control

The bandwidth control mechanism varies the frame payload and frame transmission interval. Essentially, each VL is assigned a transmission time slot — a VL can transmit a frame within an assigned bandwidth allocation gap. A bandwidth allocation gap represents the *minimum* time interval (less allowed jitter) between the beginning of consecutive frames for a given VL (Figure 7); however, an end system can transmit frames from differing VLs within the limits defined by IEEE Std 802.3.

Note: If no data is available for a VL at the next available bandwidth allocation gap, the end system is not required to transmit any data (in other words, an empty frame). Moreover, the bandwidth allocation gap represents the minimum interval for transmission — a VL can transmit data at a longer interval than its assigned bandwidth allocation gap. Although not explicitly stated, the standard implies that frames exceeding the allocated bandwidth are dropped at the incoming AFDX port.

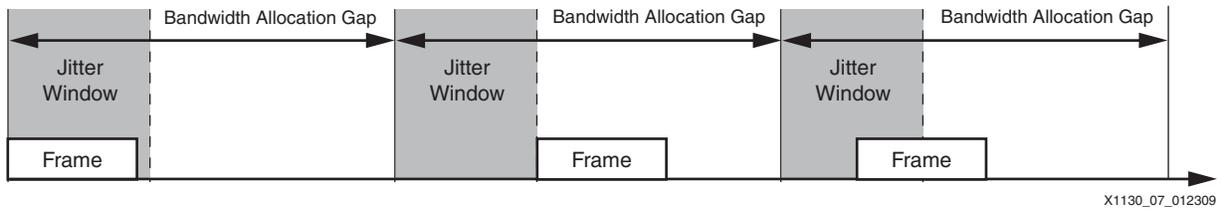


Figure 7: Single VL Transmission within Set Bandwidth Allocation Gap and Defined Jitter

Bandwidth allocation gaps range from a minimum of 1 ms to a maximum of 128 ms, the size determined by Equation 1.

$$\text{Bandwidth Allocation Gap Size} = 2^k \tag{Equation 1}$$

where k is an integer in the range of 0 to 7.

The bandwidth allocation gap value for each VL is assigned by the system integrator, based on the needs of the application, and stored in the configuration tables for the end system (and switch).

Via the traffic shaping function/scheduler, the end system reads each VL queue as needed, then determines the optional transmission order, taking advantage of the allowed jitter in scheduling frames. Each frame is transmitted outside the limits set by the bandwidth allocation gap (less jitter) for its VL, respecting the proper interframe gap between frames from differing VLs (Figure 8).

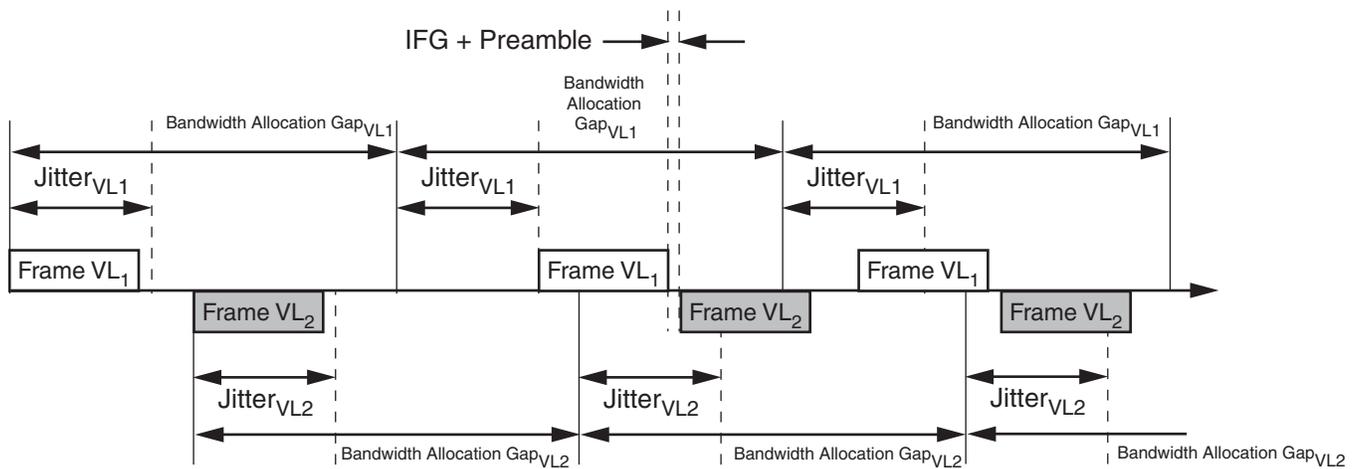


Figure 8: Scheduling Two VL Streams

An end system must be capable of transmitting data at the maximum frame rate supported by the medium. Conversely, the end system must be able to receive and process frames at that same maximum rate.

Jitter

The traffic shaping function is allowed to introduce jitter when transmitting frames. This jitter allows the end system flexibility when transmitting simultaneous (or near simultaneous) frames from differing VLs.

For AFDX, jitter is defined as the time between the beginning of the bandwidth allocation gap interval and the first bit of the frame to be transmitted in that bandwidth allocation gap interval, measured at the transmitting end system. The standard allows for 40 μ s of jitter as the result of the transmitting technology plus an amount based upon the bandwidth requirements of the VLs, limited to a maximum of 500 μ s. The maximum allowed jitter is shown in Equation 2.

$$Jitter_{MAX} \leq 40 + \frac{\sum_{i \in \{Set\ of\ VLs\}} (20 + L_{MAX}) \times 8}{N_{BW}} \quad \text{Equation 2}$$

where:

$Jitter_{MAX}$ is in μs , limited to a maximum of 500 μs .

L_{MAX} is in bytes.

N_{BW} is the bandwidth of the transmission medium in bits per second.

Latency

ARINC 664, Part 7 does not specify a system-wide latency but does provide some limits at the end system and switch level.

For an End System

For an end system, the standard limits the latency during reception to less than 150 μs . During transmission, the maximum latency for a VL is defined as:

$$Latency_{MAX} (frame_p) \leq p \times \text{bandwidth allocation gap} + Jitter_{MAX} + \text{technological_latency_in_transmission} \quad \text{Equation 3}$$

where p represents the number of the frame in a sequence of a data burst, or fragmented data. For a single frame with evenly spaced data, $p = 1$.

For a Switch

The standard defines latency for the switch as the elapsed time between the reception of the last bit of the frame until the transmission of the last bit of the frame. Switch latency is composed of three parts: technological latency of the switching function, the configuration latency due to switch loading, and the time required to transmit the frame on the medium.

The standard specifies a limit only for the technological latency (less than 100 μs).

Determining End System Capacity

The standard sets no limit on the number of VLs an end system can support and states that an end system must be able to transmit at the medium's maximum frame rate. However, the end system must respect bandwidth limits and L_{MAX} values for each VL as well as comprehend the total VL limit at the switch.

The worst-case (minimum) number of VLs occurs when L_{MAX} for each VL is 1,518 bytes and each VL is assigned the maximum bandwidth (bandwidth allocation gap = 1 ms). At 100 Mb/s, a frame of this size (1,518 bytes + 20 bytes overhead) takes 123.04 μs to transmit. With each VL respecting a bandwidth allocation gap of 1 ms, an end system could only handle eight VLs.

Note: This is based on a unit analysis only. It is doubtful that an end system could effectively schedule traffic from all eight, maximum-bandwidth, maximum-frame-length VLs. Thorough traffic modeling is required to determine the feasible maximum.

Without considering limitations at the switch level, the best case (maximum) number of VLs occurs when L_{MAX} for each VL is 64 bytes, and each VL is assigned the minimum bandwidth (bandwidth allocation gap = 128 ms). At 100 Mb/s, a frame of this size (64 bytes + 20 bytes overhead) takes 6.72 μs to transmit. With each VL respecting a bandwidth allocation gap of 1 ms, an end system could handle 19,047 VLs — far exceeding the capacity of the switch.

Limits on the number of VLs to be supported by an end system must be set by the system integrator.

Redundancy Management

During transmission, unless not required by the VL, the end system must simply transmit redundant frames via both controllers. The standard specifies that redundant frames must be sent within 0.5 ms of each other.

During reception, the end system must first check each incoming frame's integrity on both channels in parallel (irrespective of redundancy settings). For each received frame passed from the MAC, the Integrity Checker must verify that the frame received has the expected sequence number for its VL — the previous sequence number (PSN) received plus either one or two (taking into account that sequence numbers wrap from 255 to 1). If the Integrity Checker encounters an invalid frame, the frame is dropped and the system is notified of the error. The result of this check allows a single dropped frame in the data stream.

Note: This check is based upon the last frame received, even if it was discarded. This last requirement implies that the previous sequence number must be updated with the sequence number of the discarded frame and used to check the next frame, allowing for a dropped data stream to be resumed.

There are two special cases when the Integrity Checker must pass a frame that appears out of order (not equal to PSN +1 or PSN +2):

- A sequence number of zero is always accepted (indicates a transmitting end system reset).
- Any frame sequence number is accepted for the first valid frame received after a receiving end system reset.

Note: The specification requires that it is possible to disable integrity checking on a VL-by-VL basis. Integrity checking status is set at end system start-up via the configuration file.

After the Integrity Checker has a valid frame, it passes that frame to the Redundancy Manager, where the frames from both channels are compared, passing the first valid frame to the partition and dropping any redundant frames.

The Redundancy Manager operates in a two-step process (on a VL-by-VL basis):

- If a frame received on either channel is in ascending VL frame sequence number, then it is passed to the partition.
- Next, the Redundancy Manager looks for a duplicate frame from the other channel.

For AFDX, a duplicate/redundant frame is defined as a frame with the same identical VL sequence number as the last frame passed by the Redundancy Manager (for that VL), received with a specified time window (defined by the standard as $Skew_{MAX}$). A redundant frame received after $Skew_{MAX}$ is identified as a new frame and passed to the partition.

The values for $Skew_{MAX}$ (in ms) should be set by the system integrator based upon the network topology. The standard does not explicitly state but assumes that the values for $Skew_{MAX}$ are specific to a VL received at a specific end system.

Note: The standard also does not specify a maximum value for $Skew_{MAX}$; however, a maximum of 5 ms is implied in the commentary.

Switches

An AFDX switch consists of up to 24 full-duplex (but non-redundant) Ethernet ports, a central switch fabric, plus its own single-channel end system for data loading and monitoring functions. In the minimal system configuration, there are two redundant switches, one for network A and one for network B; however, there is no communication between switches, so each routes its traffic independently of the other.

In many ways, an AFDX switch resembles a commercial switch — frames are forwarded based on a static routing table, no redundancy, etc. However, the AFDX switch must perform two vital functions: frame filtering and traffic policing. The goal of these functions is to only pass valid frames (from both structure and bandwidth perspectives) to the switch fabric. This strategy isolates bad links from the rest of the network.

In addition, switch output ports must discard any frame that is older than a maximum delay value specified on a port-by-port basis in the configuration file. This requirement helps remove old data from the network (the assumption is that the redundant version of the frame was

transmitted successfully by the redundant switch, and the older redundant frame would be removed at the receiving end system anyway).

The standard requires that the switch provides a traffic prioritization mechanism, allowing high-priority traffic precedence over low priority traffic. The standard specifies that this prioritization be based on the destination end system, but specified on a VL basis in the configuration file.

Note: This standard seems to be in contradiction because a VL can have more than one destination end system. In other words, the situation could easily arise where a transmitting port must handle a mix of low and high priority VLs destined for the same end system.

Because design of an AFDX switch is beyond the scope of this document, not all aspects of the switch functionality are covered in detail. Refer to ARINC 664, Part 7 for details [Ref 1].

Frame Filtering

Upon frame reception at a switch port, the frame is filtered to ensure the validity of a frame based upon the parameters contained in the configuration table. For each received frame:

- The frame size is verified to be within the defined limits of VL length (between S^{MIN} and L_{MAX}/S^{MAX}).
- The frame is verified to have an integer number of bytes to check alignment.
- The FCS for the frame is calculated and verified against the value contained in the frame.
- The incoming switch port assignment for the VL is verified.
- The destination MAC addressed for the VL is verified as reachable.

Any frame not verified as valid is discarded and an entry is made into the management information base (MIB). Valid frames are passed to the traffic policing function.

Traffic Policing

Valid frames are then filtered for bandwidth. Any frame that exceeds the defined bandwidth limit for the VL is discarded. The standard specifies a token-bucket algorithm for policing bandwidth and allows the option of selecting either frame-based or byte-based policing.

Note: The standard does not explicitly state that an entry for any frame discarded by the traffic policing function needs to be logged to the MIB; however, best practices recommend logging.

Regardless of which version of the algorithm is chosen, discarded frames are not used in calculating bandwidth used. The goal is to enforce the bandwidth limit by blocking only those frames in excess of the defined limit.

Switch End System

In addition to the switch ports, each switch has its own end system. The design of the switch end system resembles the other end systems in the network except there is no requirement for redundancy.

The switch end system handles all direct communication with the switch and supports both data loading and network management.

Network Management

Management of an AFDX network is handled via a network management function that communicates with each AFDX network component (equipment, subscriber, and switch) to monitor the health and status of the network.

Note: The standard does not discuss where or how the network management function should be hosted.

Network health is monitored via simple network management protocol (SNMP) agents running on each subscriber (line-replaceable unit (LRU)/partition) and end system (including the switch

end system). Health status and errors are logged to the local MIBs, with status messages sent as requested by the network management function.

Application Level

In parallel with the development of AFDX is the rise of integrated modular avionic (IMA). Rather than having dedicated hardware for each onboard function (LRU), a standardized computing platform is used to run one or multiple avionics applications/subsystems (partitions). Each partition is assigned individual address spaces, and limits are set on their CPU usage to create isolation between partitions. Within an AFDX network, each partition is assigned an IP address.

Avionics subsystems communicate with the network via a standard application programming interface (API). Each partition can transfer data to the end system via either communication or service access point (SAP) ports. ARINC653 [Ref 4] defines two types of communication ports, sampling and queuing, both accessible via UDP. ARINC 664 allows for a third port type, SAP, to support legacy UDP/TCP traffic outside the API defined by ARINC653.

These ports are the communication points for VLs. Each VL or sub-VL is sourced by a single AFDX communication port; each VL and sub-VL terminates at one AFDX communication port per destination partition.

Solutions for Building End Systems

The basic building blocks required to build an AFDX end system are: two Ethernet controllers (MAC plus PHY), a processor, memory, and general-purpose logic. Because of the profiled and custom nature of an AFDX network, Xilinx® FPGAs with their flexibility are ideal solutions.

The Virtex-5 FXT and Virtex-4 FX families, with their embedded Ethernet MACs and PowerPC® processors, represent ideal solutions for AFDX end systems. In addition, devices from these families provide ample memory and logic resources to implement end system building blocks plus user logic. Both of these families are available in extended temperature ranges, making them suitable for avionics applications.

Given the compatibility between the architectures of both Virtex-5 FXT and Virtex-4 FX FPGAs, the two families can be viewed as a continuum of solutions — from the smallest Virtex-4 FX device for an end system with low bandwidth demands to the largest Virtex-5 FXT devices for an end system with high bandwidth demands and locally hosted AFDX partitions.

The Virtex-4 FX Family

Virtex-4 FX FPGAs extend the earlier Virtex series of devices, adding additional resources such as embedded Ethernet MAC. Family members are equipped with either two or four embedded Ethernet MACs and either one or two embedded PowerPC405 processors (PPC405 processor), making even the smallest member suitable for constructing an end system.

The Virtex-4 FPGA Tri-Mode Ethernet MAC supports 10/100/1000 Mb/s data rates and is designed to IEEE Std 802.3-2002 specifications. The Ethernet MAC can operate at single-speed (10, 100, or 1000 Mb/s) or in tri-mode, and in either full or half duplex. The embedded MAC supports Media Independent Interface (MII), Gigabit Media Independent Interface (GMII), and Reduced Gigabit Media Independent Interface (RGMII) for connecting to an external PHY.

The Virtex-4 FPGA Ethernet MAC block [Ref 5] contains two Ethernet MACs that share a single host interface (Figure 9). The host interface can use either the generic host bus or the DCR bus through the DCR bridge to communicate with the embedded PPC405 processor.

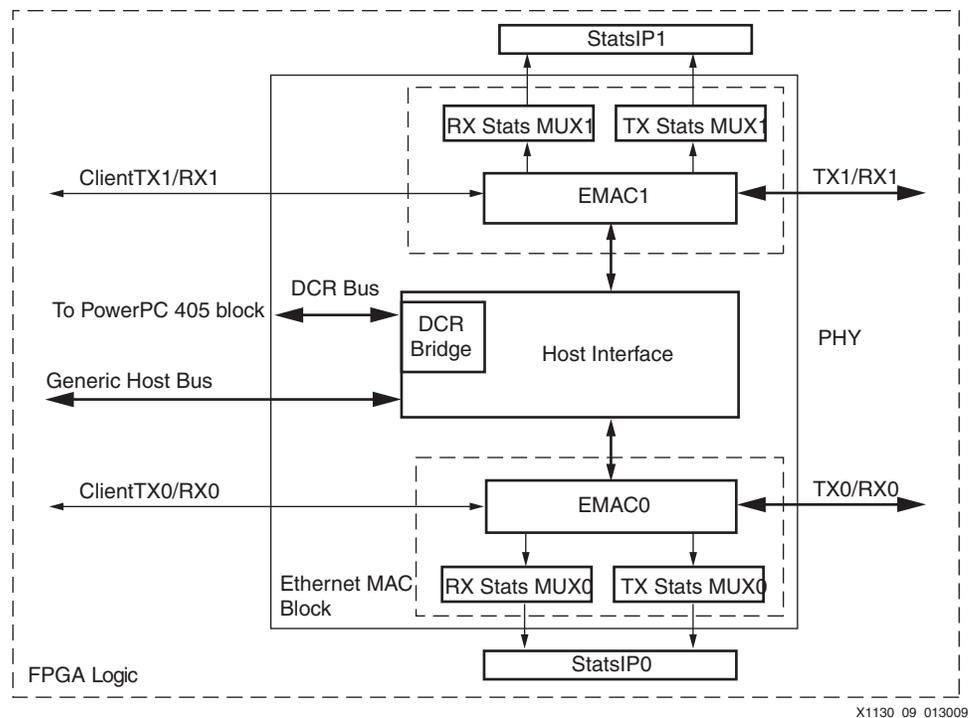


Figure 9: Virtex-4 FPGA Ethernet MAC Block

The PPC405 processor is a 32-bit implementation of the PowerPC embedded-environment architecture. The processor provides fixed-point embedded applications with high performance at low power consumption:

- Up to 450 MHz operation
- 1.5 DMIPs/MHz performance
- Five-stage datapath pipeline
- 16 KB instruction cache
- 16 KB data cache
- Enhanced instruction and data on-chip memory (OCM) controllers
- Auxiliary Processor Unit (APU) interface for direct connection from PPC405 to coprocessors in the FPGA logic

Refer to the *PowerPC Processor Reference Guide* [Ref 6], *Virtex-5 Family Overview* [Ref 7], and *Virtex-5 FPGA Embedded Tri-Mode Ethernet MAC User Guide* [Ref 8] for more details.

The Virtex-5 FXT Family

The Virtex-5 FXT family extends the capabilities found in Virtex-4 FPGAs, improving logic, memory, and DSP performance, and enhancing serial connectivity speeds, with enhanced processor performance and PCIe Endpoint capabilities. Family members are equipped with either four, six, or eight embedded Ethernet MACs and either one or two embedded PowerPC 440 processors (PPC440 processor).

Similar to the Virtex-4 FPGA Ethernet MAC block, the Virtex-5 FPGA Ethernet MAC block contains two Ethernet MACs that share a single host interface. The host interface can use either the generic host bus or the DCR bus through the DCR bridge to communicate with the embedded PPC440 processor. The Virtex-5 FPGA Ethernet MAC block adds additional flexibility in the processor interface, reduces clock resource requirements, and improves configuration capabilities over the Virtex-4 FPGA version.

The 32-bit embedded PPC440 processor contains a dual-issue, superscalar, pipelined processing unit, along with other functional elements required to implement embedded system-on-a-chip solutions:

- Up to 550 MHz operation
- Greater than 1000 DMIPS per processor (2.0 DMIPS/MHz)
- Seven-stage pipeline
- Multiple instructions per cycle
- Out-of-order execution
- 128-bit processor local buses (PLBs)
- Integrated scatter/gather DMA controllers
- Dedicated interface for connection to DDR2 memory controller
- Auxiliary processor unit (APU) interface and controller

Refer to *Embedded Processor Block in Virtex-5 FPGAs Reference Guide* [Ref 9], *First Portable AFDX Datasheet* [Ref 10], and *ML410 Embedded Development Platform User Guide* [Ref 11] for more details.

Building an End System

As with any design, deciding which functions are best handled by hardware and which are best handled by software is key. This application note presents two possible solutions: a more processor-centric solution and a more hardware-centric solution.

A processor-centric solution has the advantage of allowing the use of an off-the-shelf solution, which speeds development, but it lacks the ability to customize/differentiate a solution and can be limited in performance.

A hardware-centric solution allows for more customization and has obvious performance advantages, but it has a longer development cycle.

Note: Due to the similarities between the Virtex-4 and Virtex-5 FPGA architectures, the solutions described in this application note are applicable to both. Any differences in application between the families are highlighted.

Processor-Centric Solution

Because both Virtex-4 FX and Virtex-5 FXT FPGAs offer embedded PowerPC processors, they lend themselves to a processor-centric solution. In the processor-centric solution, the entire AFDX protocol stack is implemented in software — integrity checking, redundancy, VL, and bandwidth management are all hosted in software. In addition, the IP layer, AFDX ports, and network management functions are included. COTS solutions for a software-only AFDX protocol stack are available (see *ML510 Embedded Development Platform User Guide* [Ref 12] for an example); however, there can be limits to the number of VLs and total bandwidth available.

The processor-centric mode uses both Ethernet MACs in the MAC block, passing data to/from the processor via a packet FIFO. The protocol stack running in the mediated processor handles all aspects of the protocol, providing data to avionics partitions via AFDX communication ports. In addition, the processor handles the network management functions with a hosted SNMP agent, writing to the MIB contained in either internal or external memory (depending upon its required size and the amount of memory required by the protocol stack).

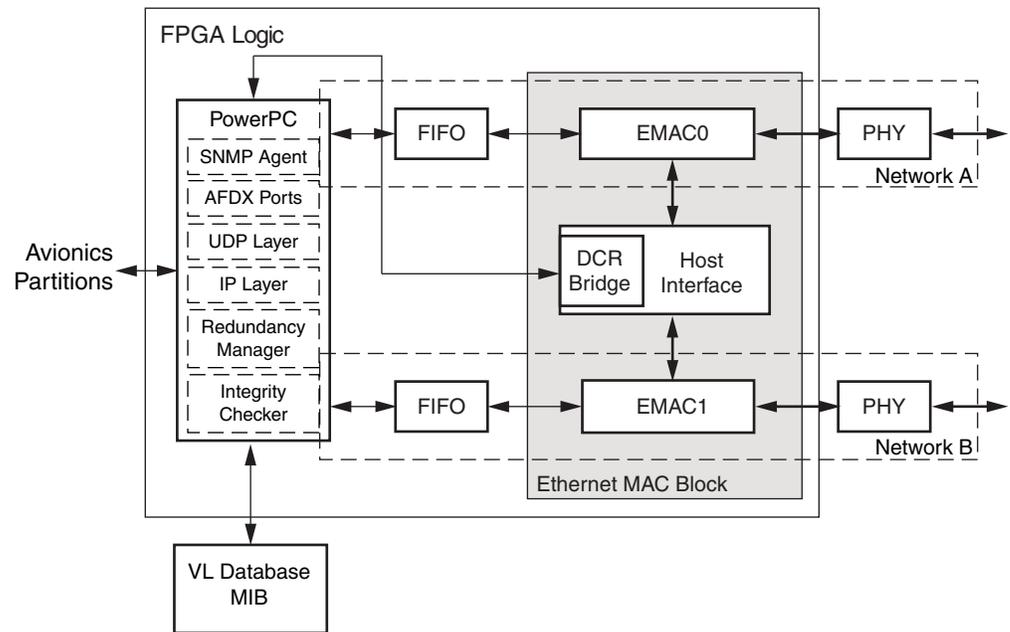


Figure 10: Processor-Centric AFDX Solution

Note: For this (and other AFDX solutions described in this application note), the Ethernet MACs must be placed in promiscuous mode because the MAC address in an AFDX frame represents the VL and not the controller — the AFDX switch filters any frame not intended for the end system.

This solution leaves room for other user applications as it places little demand on the basic FPGA logic.

Hardware-Centric Solution

In the hardware-centric solution, the parts of the AFDX protocol stack below the IP layer are implemented in hardware; integrity checking, redundancy management, VL transmit and decode, and bandwidth control are all implemented in hardware, offloading these tasks from the processor. The embedded processor handles the IP layer and above.

The advantage of a hardware-centric solution is performance and scalability. The hardware is designed to meet the needs of the specific application (number of receive and transmit VL/sub-VLs, number and size of transmit and receive frames, etc.).

Receive Path

The receive path (Figure 11) consists of an external 100BASE-TX PHY (on both networks A and B) connected to the pair of Ethernet MACs in the FPGA. These Ethernet MACs then pass data to dual integrity checkers (via a FIFO). A single redundancy manager block reads data from both integrity checkers (again via FIFO), and passes only unique valid frames in ordinal order for each VL to the PowerPC processor (unless otherwise configured).

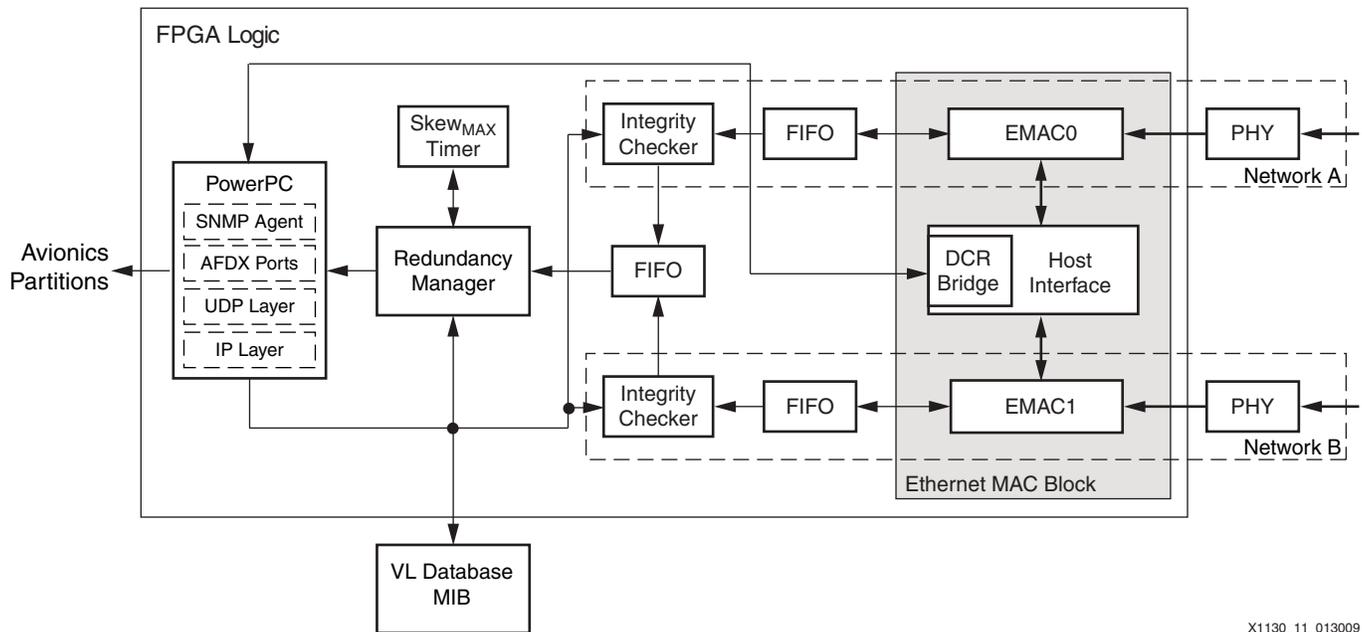


Figure 11: AFDX Receive Path

Ethernet MACs

For each of these operations, both the integrity checkers and redundancy manager must maintain a database for each VL. Because the end system is required to maintain statistics for the network management function, some or all of this data could be stored off-chip in the MIB.

The Ethernet MACs must be configured for:

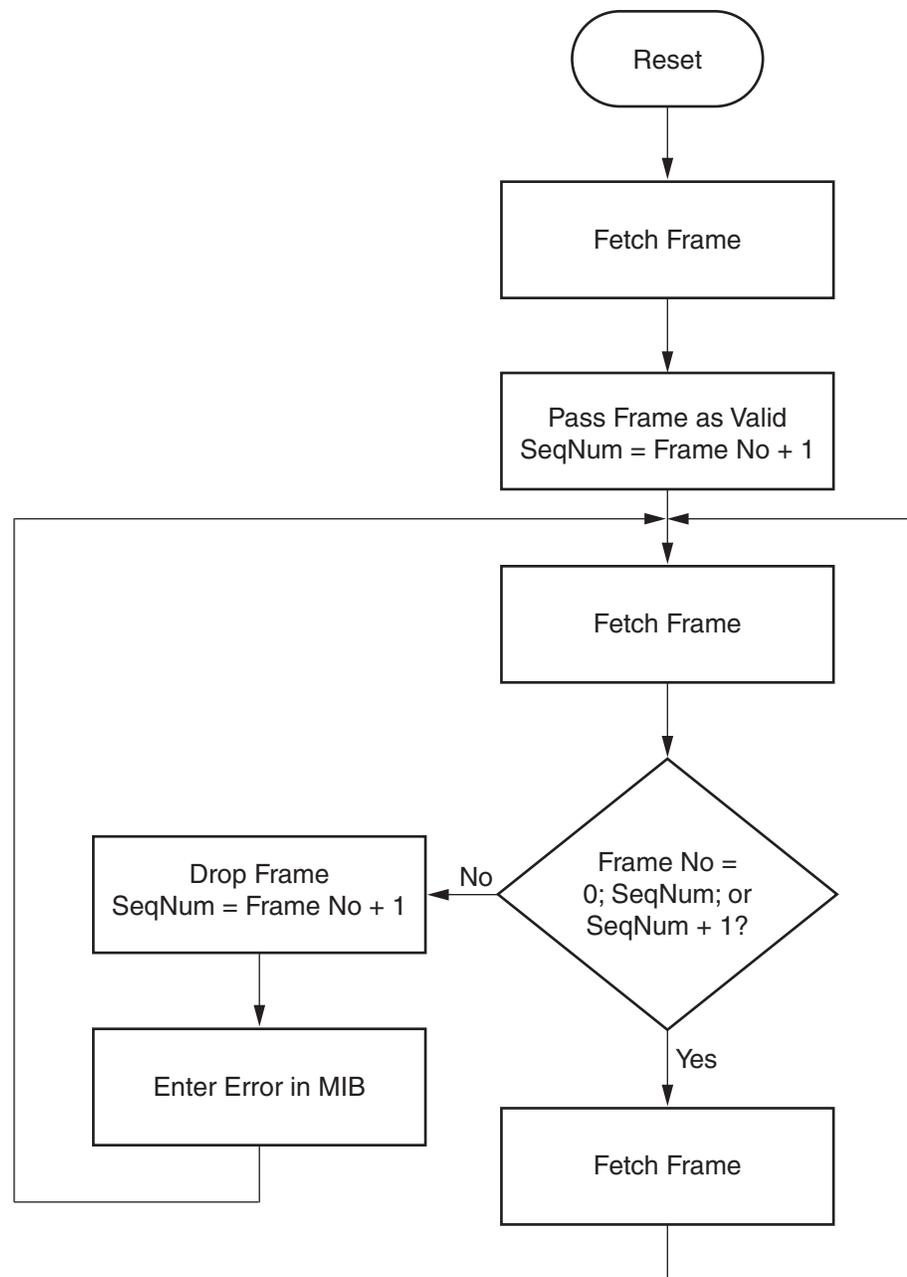
- Full-duplex operation
- 100 Mb/s operation only (no auto-negotiation)
- Address filtering disabled

Integrity Checker

For each frame passed to the integrity checker (Figure 12), the block must check the frame sequence number for that VL stream. Any frame with a sequence number that is not the next expected number or the next expected number plus one is dropped, and an entry is made into the MIB (unless the frame sequence number is zero or this is the first frame received after end system reset). Any frame found to be valid is passed to the redundancy manager (writing to a common FIFO for both channels). In both cases, the last sequence number entry for that VL must be updated in the database.

Note: The integrity checker can write valid frames to a common FIFO because the redundancy manager only needs a supply of valid frames and does not need to know which channel the frame is received on.

Rather than store the last sequence number, it can be more efficient to increment the last sequence number to allow for a unity comparison. After a match fail, the value can be incremented again and compared. If a second match fail is detected, the frame is discarded but the sequence number of that frame can be incremented and stored in the database.



X1130_12_012809

Figure 12: Integrity Checker Flowchart

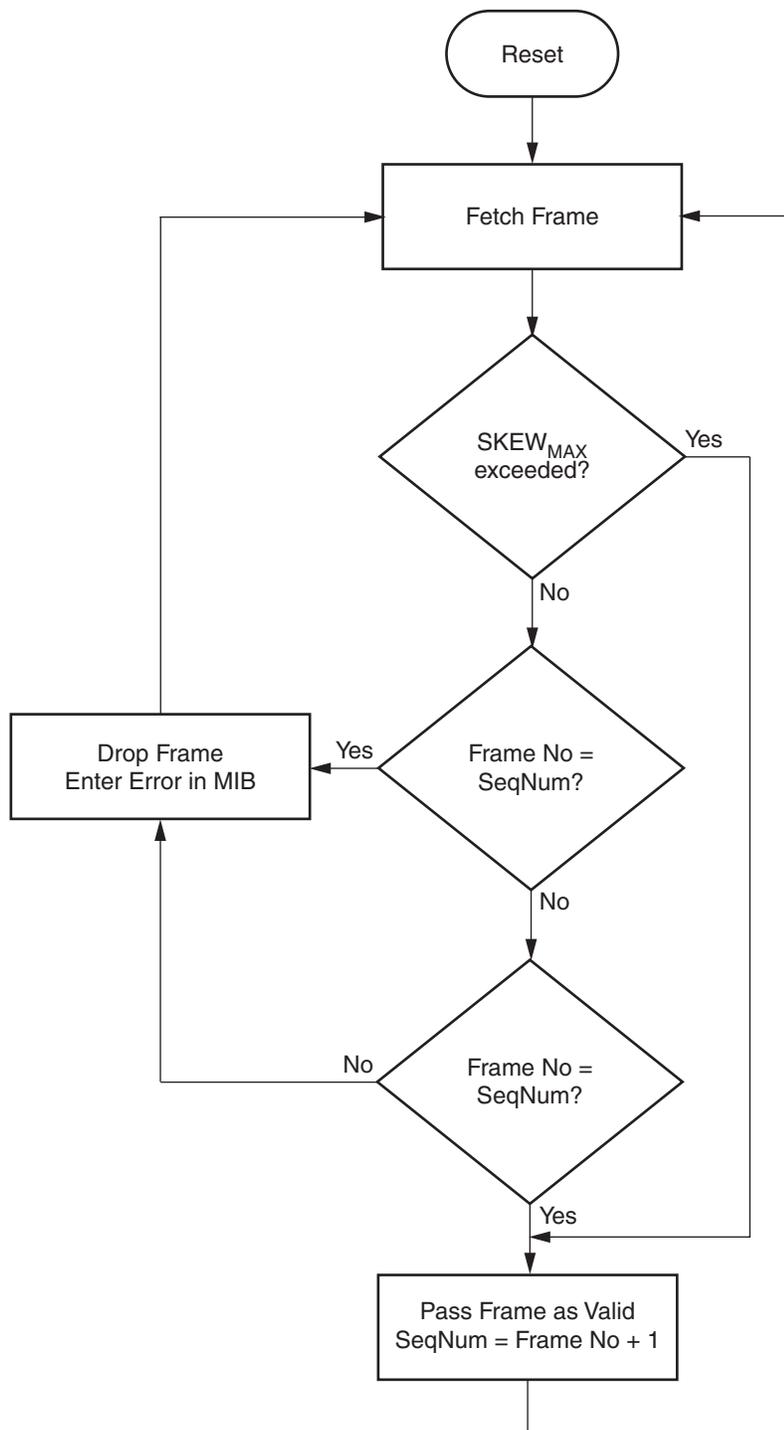
Because each VL is checked independently, integrity checker engines can run in parallel if more performance is needed (the standard requires that the end system is able to receive frames at the maximum frame rate specified by the standard, in other words, back-to-back with the required IFG). This option requires a demultiplexer block to segregate frames by VL, or the frames could be read in a round-robin fashion, but a contention-handling function is required (for example, allowing an integrity checker engine to lock a database entry for a given VL).

Redundancy Manager

The redundancy manager (Figure 13) reads the next frame from the common post-integrity-checker FIFO, reading both the VL and sequence number for the frame. The manager compares the sequence number of the received frame to the last sequence number passed by the manager

for that VL. If the received sequence number is greater than the sequence number of the last frame passed by the redundancy manager *or* if the last frame was received more than $Skew_{MAX}$ ms ago, the frame is passed to the processor and the database is updated. If the received sequence number is less than the last received frame sequence number, the frame is discarded.

Note: Although it is not clear from the standard whether $Skew_{MAX}$ is an end-system-wide or VL-specific value, it must be tracked for each VL, however, implying individual timers are required for each VL.



X1130_13_030809

Figure 13: Redundancy Manager Flowchart

PowerPC Processor

The redundancy manager passes valid frames to the PowerPC processor, which is responsible for handling the IP, UDP layers, plus the SNMP agent and AFDX ports. VL data is passed via the AFDX ports to the avionics partitions.

Transmit Path

The transmit path (Figure 14) consists of VL data streams from the avionics partitions being passed to the PowerPC processor, which in turn handles the AFDX ports, UDP and IP layers, and the scheduling of frames for transmission. Frames are written to a FIFO in transmission order. The regulator reads frames from the FIFO and then transmits each frame, respecting the VL's bandwidth allocation gap and jitter limits. Based on the redundancy settings, frames are passed to either one or both of the Ethernet MACs, which then transmit the frames via the external PHY.

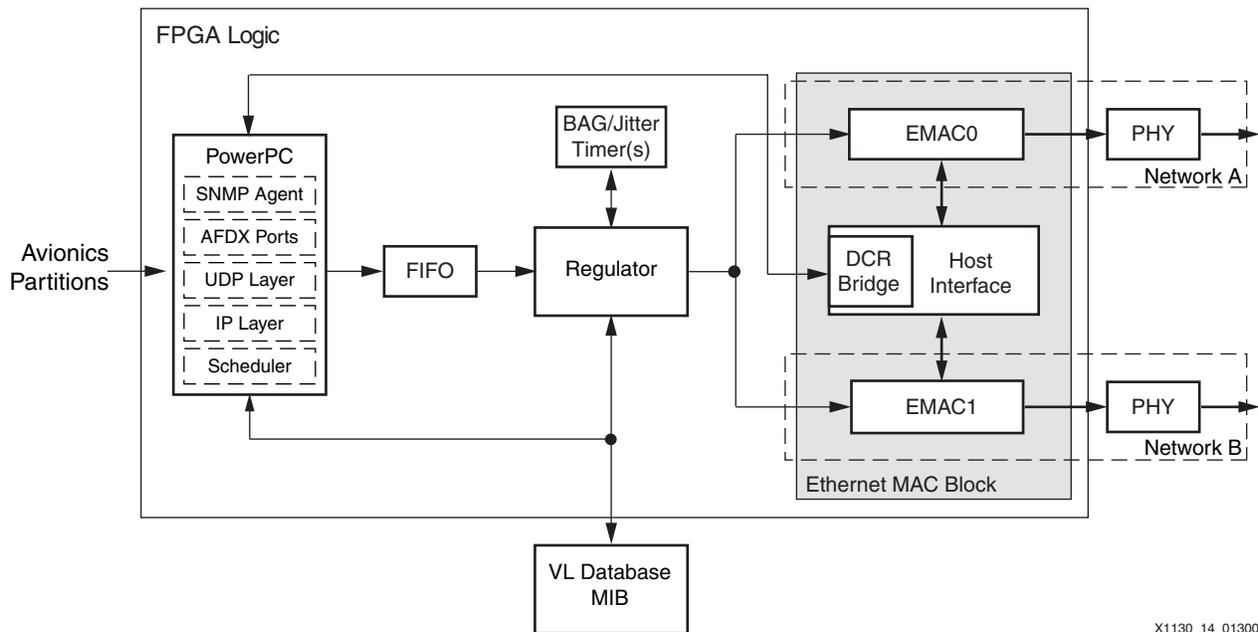


Figure 14: AFDX Transmit Path

PowerPC Processor

Aside from managing the incoming AFDX ports, SNMP agent and the UDP and IP layers, the PowerPC processor must handle scheduling (ordering) frames for transmission (the regulator handles transmission timing). The standard does not specify an algorithm or method for scheduling frames, but the scheduling function must take the following into account:

- The allowed bandwidth for each VL (in the sense of the VL's bandwidth allocation gap). Frames can be delayed but should not be scheduled, so that a frame of one VL does not block the transmission of another frame.
- The loading in each VL's incoming queue. VLs must be allowed to transmit at their maximum bandwidth. Frames can be delayed when needed, but the queue must be served.
- The required transmission time for the frame.
- The remaining transmission window for that frame.
- The priority of the VL.

Frames are written to the output FIFO in the order required for transmission.

The Regulator

After the frame ordering is determined, the regulator controls the exact timing of frame transmission, respecting the VL's bandwidth allocation gap limit and end system allowed jitter. One possible solution for the regulator is a by-the-book approach that transmits frames as soon as they are available, respecting the bandwidth allocation gap and jitter limits. This approach can ship consecutive frames separated only by the IFG.

The by-the-book approach depends on the PowerPC processor-based scheduler to order the frames properly. The regulator (Figure 15) reads a frame from the FIFO and reads the VL identifier to determine the proper bandwidth allocation gap. The frame can be transmitted after the bandwidth allocation gap has elapsed, and within the jitter window. If the jitter window expires, the frame can still be transmitted; however, the bandwidth allocation gap timer is reset (Figure 16). The frame is sent to one or both of the Ethernet MACs (depending on the redundancy settings), and the next frame in the FIFO is fetched.

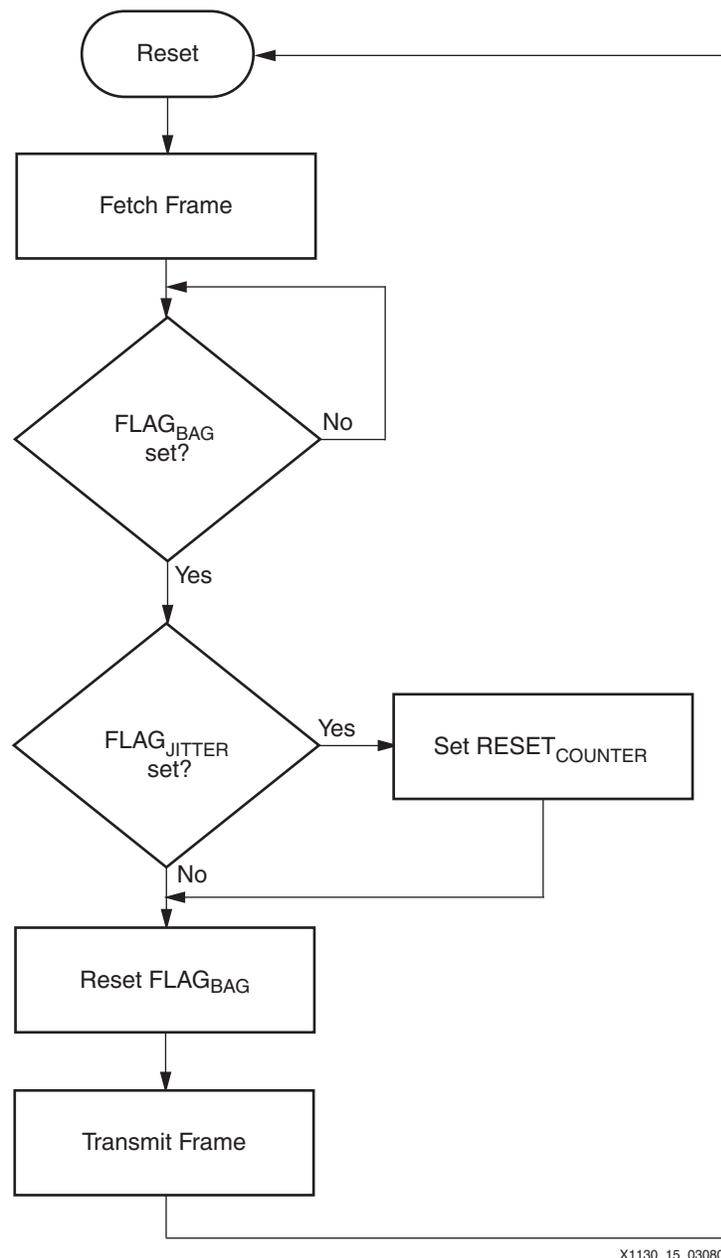
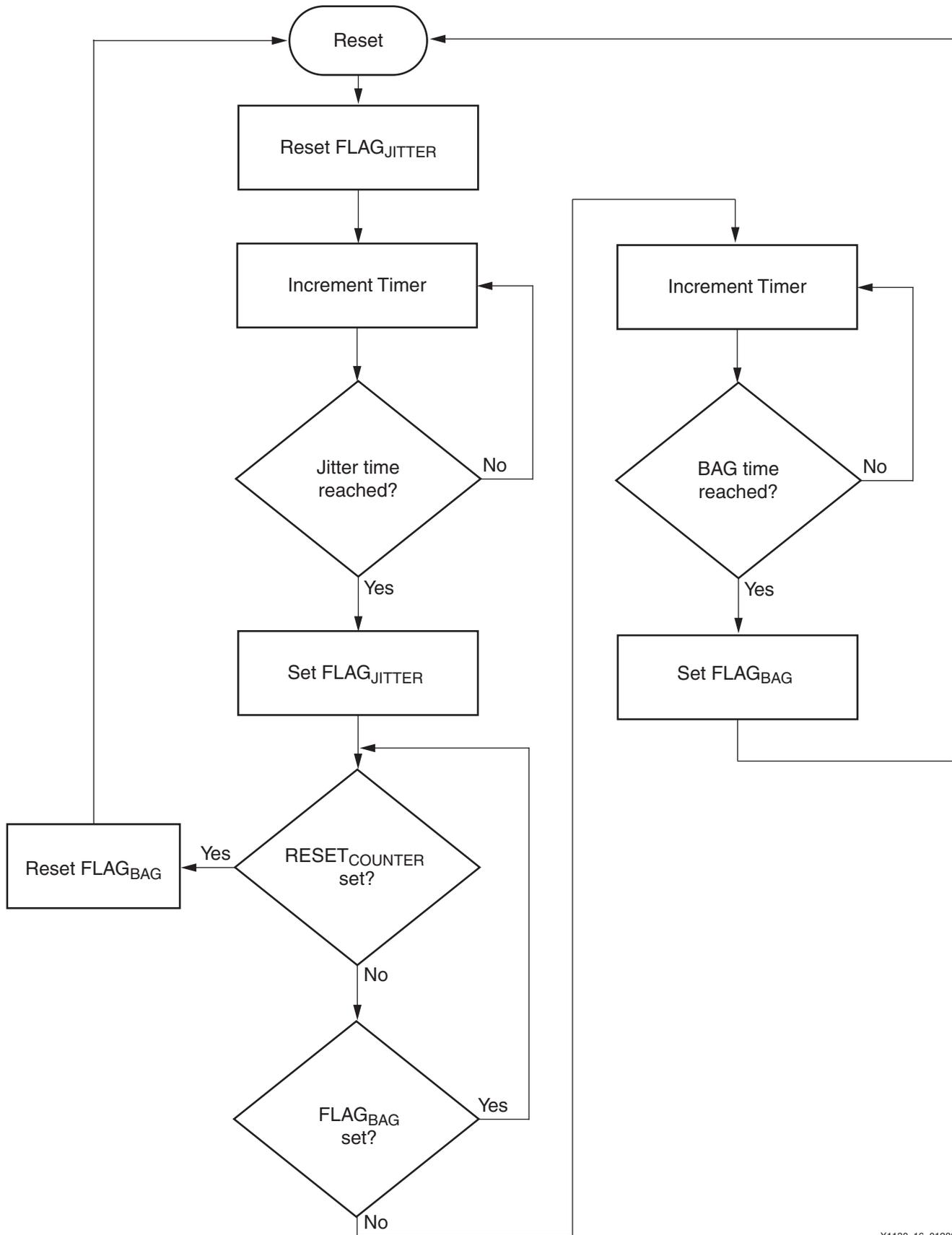


Figure 15: Regulator Flowchart (Per VL)

X1130_15_030809



X1130_16_012809

Figure 16: Bandwidth Allocation Gap/Jitter Timer Flowchart

For example, consider an end system with three VLs to transmit:

VL_A – bandwidth allocation gap = 1 ms; L_{MAX} = 1,518 bytes

VL_B – bandwidth allocation gap = 2 ms; L_{MAX} = 1,024 bytes

VL_C – bandwidth allocation gap = 2 ms; L_{MAX} = 512 bytes

For this end system, the allowed jitter is 289 μ s (per Equation 2).

With the by-the-book approach, frames are transmitted as soon as they are available. The stream shown in Figure 17 assumes that all three VLs send traffic at their maximum bandwidth and all have data available at the start. Frames are only separated by the IFG, bandwidth allocation gap, and allowed jitter.

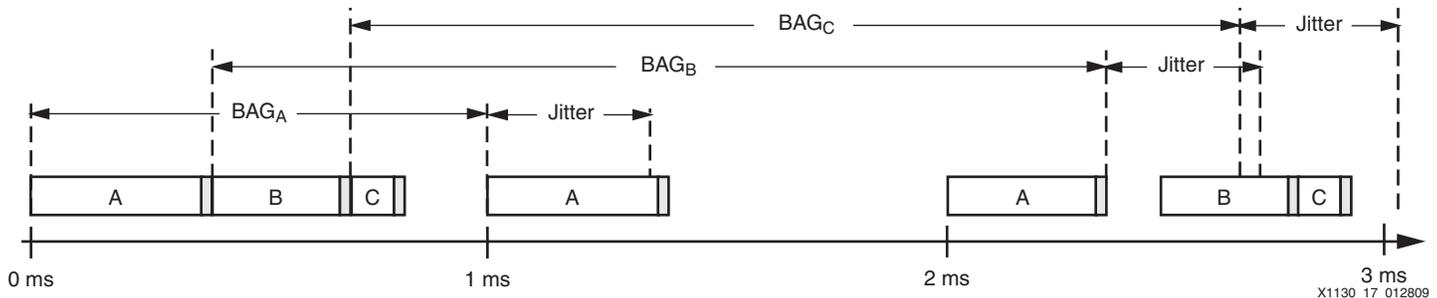


Figure 17: By-the-Book VL Transmission

The by-the-book approach:

- Requires a smart scheduler
- Individual bandwidth allocation gap/jitter timers for each VL

Ethernet MACs

Regardless of which approach is chosen for the regulator, output frames are transferred to one or both of the Ethernet MACs (depending on the redundancy settings for the VL). The Ethernet MACs transmit the frames on the networks via the external PHYs.

The Ethernet MACs, must be configured for:

- Full-duplex operation
- 100 Mb/s operation only (no auto-negotiation)

Prototyping Solutions

Xilinx provides demonstration/evaluation kits for developing AFDX end system solutions.

ML410 Embedded Development Platform for Virtex-4 FX FPGAs

The ML410 series of embedded development platforms [Ref 11] offers designers a versatile Virtex-4 FX device for rapid prototyping and system verification. In addition to the more than 30,000 logic cells, over 2,400 kb of block RAM, dual PPC405 processors, and RocketIO™ transceivers available in the FPGA, the ML410 provides two onboard Ethernet MAC PHYs, DDR memory, multiple PCI bus slots, and standard front panel interface ports within an ATX form-factor motherboard. An integrated System ACE™ tool CompactFlash controller is deployed to perform board bring-up and to load applications from the CompactFlash card.

ML510 Embedded Development Platform for Virtex-5 FXT FPGAs

The ML510 series of embedded development platforms [Ref 12] offer designers a versatile Virtex-5 FXT device for rapid prototyping and system verification. In addition to the more than 130,000 logic cells, over 10,700 kb of block RAM, dual PPC440 processors, and RocketIO transceivers available in the FPGA, the ML510 provides two onboard Ethernet MAC PHYs,

DDR2 memory, multiple PCI bus slots, and standard front panel interface ports within an ATX form-factor motherboard. An integrated System ACE tool CompactFlash (CF) controller loads applications from the CompactFlash card.

AFDX and DO-254

RTCA/DO-254 and its counterpart in Europe, EUROCAE/ED-80, are the guidelines for the design of complex electronic hardware (CEH) for use in avionics systems. FAA advisory circular AC 20-152 made DO-254 an official requirement for suppliers of civil aviation avionics systems. DO-254 is a collection of best industry practices for design assurance of airborne electronic hardware. These guidelines advocate a top-down approach for design and verification of safety critical electronics and other avionics systems and represent the consensus of the aviation community.

System Failure Levels

The FAA has defined a number of levels regarding the safety and criticality of an avionic system. For example, engineers designing to level A or B face a much more rigorous test, verification, and documentation process than for levels C, D, or E. All flight hardware needs to be classified as having one of these failure levels [Ref 13].

Within an AFDX network, not all end system applications carry the same level of criticality. Each end system application carries the criticality level associated with the avionics partitions it supports. The required level is set by the system integrator in consultation with the equipment supplier.

Network Safety versus Network Security

When discussing a network, the issue of network safety versus network security must be discussed. DO-254 addresses the safety of the network, in other words, the reliability of the network and its susceptibility to component failure.

Network security, or the network's susceptibility to viruses, and hacking are separate issues that are not addressed by DO-254.

A related issue, also not addressed by the standard, is bitstream security. While tampering with the FPGA bitstream is a remote possibility, Xilinx does provide effective bitstream security and encryption solutions to protect designs from malicious tampering [Ref 14].

Potential Methods for DO-254 with an ARINC 664 (AFDX) Solution Using Xilinx Devices

Generally, designers have a range of potential methodologies available to mitigate errors when designing solutions to meet DO-254 requirements [Ref 13]. Depending upon the design assurance level, designers can employ several fault mitigation schemes when implementing the design into an FPGA (in descending order of *strength*):

- Triple-FPGA redundancy with external voting circuits
- Dual-FPGA redundancy
- Triple-module redundancy (TMR) with voting circuits implemented in the FPGA
- Circuit redundancy with arbitration inside a single FPGA
- Bitstream scrubbing with error correction
- Periodic FPGA reconfiguration

AFDX presents a special case with respect to DO-254 — AFDX already includes redundancy as well as error mitigation. Given the redundant nature of AFDX, the range of additional, viable mitigation techniques is narrowed. Depending upon the design assurance level, possible techniques to increase fault immunity for an AFDX application fall into two categories:

- **Multiple-device solutions:** Networks A and B can be split between two FPGAs, placing network integrity, redundancy management, and transmit regulator in a third FPGA.
- **Single-device solutions:** The entire end system is implemented in a single FPGA, employing device-level mitigation techniques.

The strength of these solutions can be adjusted by implementing (in descending order of strength):

- TMR in all FPGAs (for both solutions)
- TMR only in the third FPGA supporting network integrity, redundancy management, and transmit regulator (for the multiple-device solution)
- Bitstream scrubbing with error correction
- Periodic FPGA reconfiguration

The exact technique employed should be determined in consultation with both the system integrator and the Designated Engineering Representative.

Conclusion

The increased complexity of today's commercial aircraft requires an enhanced data communication solution. By adapting commercial Ethernet technology, AFDX not only provides much higher data rates compared to earlier solutions but allows the leveraging of existing IP and silicon. The Virtex-5 FXT and Virtex-4 FX families, with their embedded Ethernet MACs and PowerPC processors, plus abundant resources for user logic, represent ideal solutions for AFDX end systems.

References

1. ARINC Specification 664, Part 7, *Aircraft Data Network, Avionics Full Duplex Switched Ethernet (AFDX) Network*.
2. ARINC Specification 429, *Mark 33 Digital Information Transfer System (DITS)*.
3. ARINC Specification 664, Part 1, *Aircraft Data Network, Systems Concepts and Overview*.
4. ARINC Specification 653, *Avionics Application Software Standard Interface*.
5. [UG074](#), *Virtex-4 FPGA Embedded Tri-Mode Ethernet MAC*.
6. [UG011](#), *PowerPC Processor Reference Guide*.
7. [DS100](#), *Virtex-5 Family Overview*.
8. [UG194](#), *Virtex-5 FPGA Embedded Tri-Mode Ethernet MAC User Guide*.
9. [UG200](#), *Embedded Processor Block in Virtex-5 FPGAs Reference Guide*.
10. *First Portable AFDX Datasheet*, SYSGO AG, 2007.
<http://www.sysgo.com>
11. [UG085](#), *ML410 Embedded Development Platform User Guide*.
12. [UG356](#), *ML510 Embedded Development Platform User Guide*.
13. [WP332](#), *Meeting DO-254 and ED-80 Guidelines when using Xilinx FPGAs*.
14. Design Security Solutions
www.xilinx.com/products/design_resources/security/index.htm.

Appendix A: Background

A modern commercial airframe has to connect thousands of sensors and actuators with a plane's control systems. Due to the critical nature of these systems, each of the sensors and actuators must be connected directly to their control systems. Although the bandwidth required for these connections is low (on the order of 100 kb/s or less), the connections must be robust, providing guaranteed delivery of data and offering no bus connection or data collisions.

In many aspects the requirements for an aircraft data network (ADN) mirror that of the public switched telephone network (PSTN), with its thousands of point-to-point, low-speed (64 kb/s) connections and its requirement for quality of service (QoS) and robustness — an analogy not lost on the architects of AFDX.

ARINC 429

For nearly 30 years, almost every commercial aircraft developed, starting with the Boeing 727 through to the 767 and Airbus A340, has made use of ARINC 429 data busing for connecting onboard electronics. Electrically, ARINC 429 buses are composed of a single twisted wire pair connecting one transmitter with up to 20 receivers (Figure 18). All communication is unidirectional, sending 32-bit data words at either low (12.5 kb/s) or high (100 kb/s) speed.

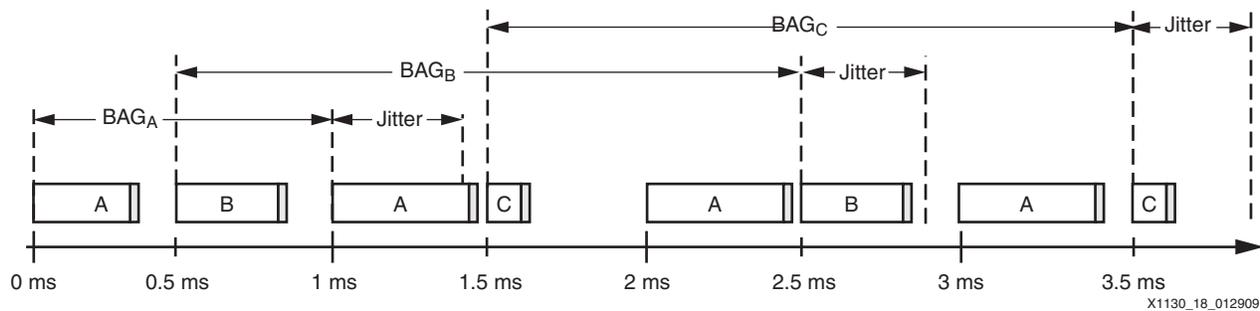


Figure 18: ARINC 429 Topology

Despite the robustness of ARINC 429, starting with the design of the first all-electronic, fly-by-wire system on the Airbus A320, it was clear that a replacement standard was needed. With the number of systems required by a modern airframe, the amount, size, and weight of wiring required to connect all of the sensors, controllers, and actuators (also known as Line Replaceable Units or LRUs) made ARINC 429 impractical for future design (already, the Boeing 747-400 aircraft has 171 miles of wiring). Designers of the A380 began looking for alternatives.

To avoid the expensive and lengthy development of a custom, aviation-only solution, A380 designers looked to leverage as much commercial-off-the-shelf (COTS) technology as possible. While most avionics systems do not require high data rates, a higher bandwidth solution was sought to support both newer technology but also allow for the multiplexing of connections. IEEE 802.3 Ethernet was chosen as the basis for a new ADN solution,

Ethernet

IEEE 802.3 Ethernet has an even longer history than ARINC 429, and has the advantage of being widely deployed and well understood. In addition, there are many commercial suppliers supporting all aspects of the standard, allowing a solution based on Ethernet to be built largely out of commercially available building blocks.

Despite the bandwidth and maturity, Ethernet, as typically deployed, has several drawbacks when compared to the needs of an ADN:

- **Broadband connection** – the standard handles the transfer of bulk data on network with many receivers and transmitters without central control. Any connection can use all of the available bandwidth.

- **Data collisions are allowed** – Ethernet does not prevent data collisions from occurring, but rather employs a technique for handling collisions when they occur using carrier sense multiple access with collision detection (CSMA/CD).
- **Bandwidth is shared** – there is no bandwidth guarantee for a single connection, the effective bandwidth being a function of the size of the network, its topology, and traffic.
- **Best-effort network** – there is no QoS requirement with Ethernet. All data is delivered based on current network traffic.
- **Vulnerability** – loss of a single wire severs the connection between data terminals; however, other network connections might not be impacted.

Clearly, an effective ADN cannot be built solely from commercial Ethernet concepts.

ATM

Given the similarity of ADNs with the publicly switched telephone network (PSTN), concepts from telephony can also be applied. The asynchronous transfer mode (ATM) is a cell-based packet-switched network protocol. The protocol supports virtual point-to-point connections (virtual circuits) with QoS guarantee (bandwidth, latency, and jitter). Within the PSTN, data for multiple channels is time-division multiplexed (TDM) over the ATM connection.

Each 53-byte ATM cell contains five header bytes containing addresses specifying the virtual circuit path and channel (the rest of the cell is data payload). The protocol allows each connection to specify its required QoS (traffic contract).

Revision History

The following table shows the revision history for this document.

Date	Version	Revision
03/20/09	1.0	Xilinx initial release.

Notice of Disclaimer

Xilinx is disclosing this Application Note to you “AS-IS” with no warranty of any kind. This Application Note is one possible implementation of this feature, application, or standard, and is subject to change without further notice from Xilinx. You are responsible for obtaining any rights you may require in connection with your use or implementation of this Application Note. XILINX MAKES NO REPRESENTATIONS OR WARRANTIES, WHETHER EXPRESS OR IMPLIED, STATUTORY OR OTHERWISE, INCLUDING, WITHOUT LIMITATION, IMPLIED WARRANTIES OF MERCHANTABILITY, NONINFRINGEMENT, OR FITNESS FOR A PARTICULAR PURPOSE. IN NO EVENT WILL XILINX BE LIABLE FOR ANY LOSS OF DATA, LOST PROFITS, OR FOR ANY SPECIAL, INCIDENTAL, CONSEQUENTIAL, OR INDIRECT DAMAGES ARISING FROM YOUR USE OF THIS APPLICATION NOTE.

XILINX PRODUCTS (INCLUDING HARDWARE, SOFTWARE AND/OR IP CORES) ARE NOT DESIGNED OR INTENDED TO BE FAIL-SAFE, OR FOR USE IN ANY APPLICATION REQUIRING FAIL-SAFE PERFORMANCE, SUCH AS IN LIFE-SUPPORT OR SAFETY DEVICES OR SYSTEMS, CLASS III MEDICAL DEVICES, NUCLEAR FACILITIES, APPLICATIONS RELATED TO THE DEPLOYMENT OF AIRBAGS, OR ANY OTHER APPLICATIONS THAT COULD LEAD TO DEATH, PERSONAL INJURY OR SEVERE PROPERTY OR ENVIRONMENTAL DAMAGE (INDIVIDUALLY AND COLLECTIVELY, “CRITICAL APPLICATIONS”). FURTHERMORE, XILINX PRODUCTS ARE NOT DESIGNED OR INTENDED FOR USE IN ANY APPLICATIONS THAT AFFECT CONTROL OF A VEHICLE OR AIRCRAFT, UNLESS THERE IS A FAIL-SAFE OR REDUNDANCY FEATURE (WHICH DOES NOT INCLUDE USE OF SOFTWARE IN THE XILINX DEVICE TO IMPLEMENT THE REDUNDANCY) AND A WARNING SIGNAL UPON FAILURE TO THE OPERATOR. CUSTOMER AGREES, PRIOR TO USING OR DISTRIBUTING ANY SYSTEMS THAT INCORPORATE XILINX PRODUCTS, TO THOROUGHLY TEST THE SAME FOR SAFETY PURPOSES. TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CUSTOMER ASSUMES THE SOLE RISK AND LIABILITY OF ANY USE OF XILINX PRODUCTS IN CRITICAL APPLICATIONS.