Essential SpaceWire Hardware Capabilities for a Robust Network

Session: SpaceWire Networks and Protocols, Short Paper

Michael Birmingham
GOES-R Embedded Software Engineer
NASA Goddard Space Flight Center
Denver, CO USA
mike.j.birmingham@lmco.com

William Anderson
GOES-R Flight Data System Lead Engineer
NASA Goddard Space Flight Center
Greenbelt, MD USA
william.h.anderson@nasa.gov

Alexander Krimchansky
GOES-R Mission Systems Manager
NASA Goddard Space Flight Center
Greenbelt, MD USA
alexander.krimchansky@nasa.gov

Matthew Lombardi
GOES-R Simulation and Test Engineer
Lockheed Martin
Denver, CO USA
matthew.s.lombardi@lmco.com

***Abstract*— The Geostationary Operational Environmental Satellite R-Series Program (GOES-R) mission is a joint program between National Oceanic & Atmospheric Administration (NOAA) and National Aeronautics & Space Administration (NASA) Goddard Space Flight Center (GSFC). GOES-R project selected SpaceWire as the best solution to satisfy the desire for simple and flexible instrument to spacecraft command and telemetry communications. GOES-R development and integration is complete and the observatory is scheduled for launch October 2016.**

The spacecraft design was required to support redundant SpaceWire links for each instrument side, as well as to route the fewest number of connections through a Slip Ring Assembly necessary to support Solar pointing instruments. The final design utilized two different router designs.

The SpaceWire standard alone does not ensure the most practical or reliable network. On GOES-R a few key hardware capabilities were identified that merit serious consideration for future designs. Primarily these capabilities address persistent port stalls and the prevention of receive buffer overflows. Workarounds were necessary to overcome shortcomings that could be avoided in future designs if they utilize the capabilities, discussed in this paper, above and beyond the requirements of the SpaceWire standard.

I. Introduction

This paper seeks to describe some of the pitfalls encountered during the design and integration of major components for the Geostationary Operational Environmental Satellite-R Series (GOES-R) program [1]. An awareness of those pitfalls may prevent a similar experience in future designs.

The GOES-R spacecraft uses European Cooperation for Space Standardization (ECSS) SpaceWire [2] for the transfer of sensor, telemetry, ancillary, command, time code, and time synchronization information between instruments and the spacecraft. Capabilities beyond those specified in the standard are offered in the interest of providing a more robust system.

This paper describes four instances where considerable effort was expended to avoid or mitigate problems concerning persistent port stall, receive buffer overflow, pipeline side-effects, and a situation where buffer depth configuration of a node exposed a router defect that locked out further transfers. This specific configuration can be avoided given the details in that section.

A. Background Information

GOES-R uses Reliable Data Delivery Protocol (GRDDP [3]) which specifies that Reset packets are transmitted at that channel's transmit timeout rate from the time that the channel is placed into an Enabled state, until an Acknowledge packet is received. The transmit timeout is specified in an instrument Interface Control Document (ICD), and is on the order of 100ms for the instruments described in this paper. During instrument power-on, the spacecraft will begin transmitting Reset packets (9 bytes in length) to the instrument at a 100ms rate until the instrument responds.

The spacecraft transition to Enabled state is delayed from the application of instrument power to coincide with the point at which the instrument enters Run Mode, and is able to process GRDDP messages. If the instrument indeed enters Run Mode at about the expected time, few Reset packets will be transmitted. Problems may arise, however, if there is a problem with either the instrument or the link.

GOES-R also specifies that instruments shall transition to a Safe Mode if time ticks or time-of-day messages are absent for 10 consecutive seconds.

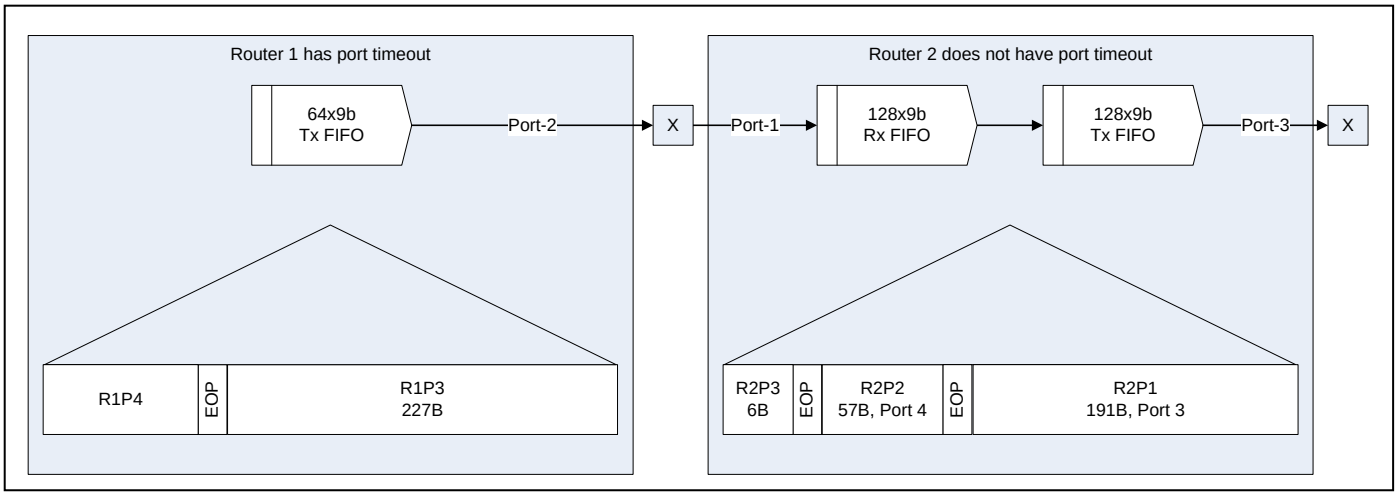

Figure 1 Example Routers and Packets for Transmit Timeout Discussion

I. Persistent port Stall Prevention

The first capability to be discussed is a mechanism to prevent an indefinite network stall. This is especially important when routers are employed between nodes. The need will be illustrated in the following sections by way of an example.

A. Routing example

The following is a simplified example of a real-world condition encountered on GOES-R during instrument emulator integration. Router 1 in the following example is implemented as a Goddard Space Flight Center (GSFC) developed core [4] which is part of the BAE Systems SpaceWire ASIC [5]. Router 2 is an Aeroflex 4-port router [6].

Details such as Lookup Tables (LUTs), registers, arbiters and other router components are not included since it assumed that the reader has a working knowledge of those mechanisms.

B. Initial Condition

In Figure 1 above, Router 1 has port transmit timeout capabilities, while Router 2 does not. Router 2 Port 3 is in disconnect due to an instrument or cable failure, and cannot reconnect. This condition may be present prior to instrument power-on or may occur during operation.

R2P1 is Router 2 Packet 1; it is 192 bytes including the End of Packet marker (EOP) and its destination is Port 3. No part of Packet 1 has been transmitted yet, in this example.

R2P2 is Router 2 Packet 2; it is 58 bytes including the EOP and will be routed to Port 4 (not shown).

R2P3 is the leading portion of Packet 3, while R1P3 is the trailing portion of Packet 3. Packet 3 is 234 bytes in total, including the EOP.

R1P4 is the final packet to be queued up for Router 1, but neither the length nor the EOP is indicated because it is not relevant for this example.

C. Stall Condition

R2P1 will not be delivered due to the disconnect condition on Router 2 Port 3. Since R2P1 exceeds the size of Port 3's transmit (Tx) First In First Out memory (FIFO), it will block Port 1's arbiter. The trailing portion of Packet 1 and all of Packet 2 will occupy all but 6 bytes of Port 1's Receive (Rx) FIFO. The remaining free space on Router 2's Port 1 Rx FIFO will be filled with the leading 6 bytes of Packet 3.

D. Timeout Condition

Router 1 Port 2 has not completed transmitting Packet 3 within the programmed timeout limit, and disconnects Port 2. Pursuant to ECSS error recovery specifications, Router 1 will spill the trailing 228 bytes of Packet 3. Router 2 will not append an EEP to partial packet 3 because there is no space in the receive buffer.

E. Link Recovery

Both Router 1 and Router 2 will issue NULL characters in an attempt to re-establish the link. Assuming Router 1 Port 2 Rx FIFO (not shown) has at least 8 bytes free, it will also issue one or more Flow Control Token (FCT) characters. Router 2, on the other hand, will not issue an FCT because there are no bytes free in its Rx FIFO.

The ECSS standard does not have a remedy for this situation. It is assumed that there are no hard link errors, and that eventually data will flow through Router 2 Port 3. If the failure is not recoverable with an instrument power cycle (if it can even be identified by the host system) then the failure will persist ad infinitum.

F. GOES-R Configuration

On GOES-R, only the first router in the chain is capable of disconnect on a transmit timeout, and it is not on a per-port basis; the timeout applied to all ports equally. The routers downstream (Aeroflex 4-port routers) of that router had no transmit timeout capability. The indefinite stall cannot be avoided unless all routers have the transmit timeout capability.

G. GOES-R Stall Consequences

The perpetual stall means that all instruments downstream of Router 2 Port 1 will be unable to communicate. Instrument telemetry will not be acknowledged, and instruments will no longer receive commands, time messages or time ticks. Within 10 seconds instruments fall into Safe Mode. All GOES-R GRDDP transmit channels to those instruments close and numerous error events result. Unless the condition was present during the power-on process, there is no way of knowing which port of which router was in disconnect.

H. GOES-R Recovery Method

The GOES-R recovery method begins by powering off all instruments downstream of Router 1 port 2. A hard reset is then required of the routers downstream of Router 1 (there are four on GOES-R). Each hard reset clears the FIFOs and all router registers are returned to default values. The reset does not affect the LUT contents. Next, the registers have to be re-configured for each router. As each instrument is powered up, their router port status is examined. If not in Run State, the instrument is swapped to the redundant side.

I. Recommended Design Solution

On any network involving one or more nodes, a programmable transmit timeout feature on **every** router port in the chain is essential to preventing a perpetual stall somewhere in the chain. Of course the timeout must apply to any packet that stalls the transmitter, even if the port is in disconnect and no part of the packet has been sent. The ECSS standard specifies only that a partial packet be spilled when the link error is reported (transition to disconnect).

The transmit timeout feature on all routers will clear the stall but as long as the point of origin continues transmitting packets to the node in disconnect then the behavior repeats indefinitely. Best practice would be to check port status prior to and following instrument power-on, as well as periodic monitoring.

There may be considerable packet jitter with this solution, caused by the timeout that must expire before a packet is spilled. When using GRDDP, the port timeout setting must be much less than the shortest re-transmit timeout, since Reset packets have priority over all but Ack packets and Reset packets will likely be prevalent in this situation.

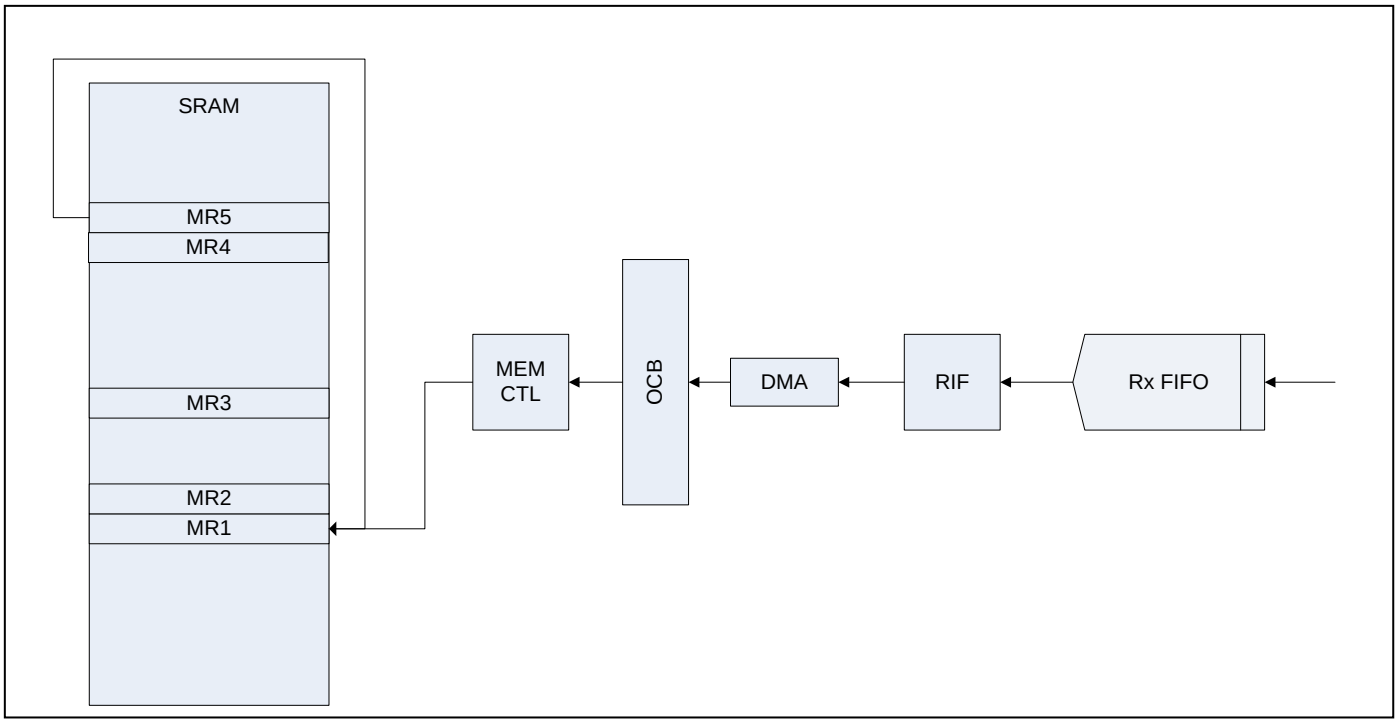

Figure 2 Example Receive Buffer Chain

II. Receive buffer overflow prevention

The next capability to be discussed is a mechanism to prevent receive buffer overflow. The ECSS standard does not limit the length of data packets, but practical applications should limit the size of packets, as does the GRDDP protocol. On GOES-R, each instrument also had constraints on the size of packets that were to be transmitted or received, which were equal to or less than what the protocol allowed.

The ECSS standard assumes that FCT messaging will prevent receive buffer overflow (section 8.3). In reality, receive FIFO overflow is prevented, but not necessarily receive buffers. Rx FIFOs are the domain of hardware and credit counts and FCT transmit is the purview of that lower level of the system. The practical problem is that the receive front end may have no idea of the size of the buffer in system memory. FCTs are issued when there is room in the FIFO, without consideration for the size of the host system buffer allocated for data transfer from the FIFO.

A. Packet Overflow

Receive buffers on link end points have high and low memory limits, whether that memory is statically or dynamically allocated, and whether a linked list is contiguous or scattered in physical memory. Receive buffer overflow is very damaging, so a high-availability system should seek to avoid that situation with a hardware mechanism of some sort. The host system may be removed from the receive front end by several layers complicating the connection between receive FIFO and receive buffer. The complexity of the chain may be inadequate to prevent receive buffer overflow or to avoid an intricate recovery.

B. GOES-R Spacecraft Receive Buffer Chain

The BAE SpaceWire ASIC is used in the GOES-R spacecraft to interface to the instruments, and Figure 2 illustrates the Application Specific Integrated Circuit (ASIC) cores involved in transferring incoming telemetry to system Static Random Access Memory (SRAM).

The Rx FIFO is connected to a Receive Interface (RIF), which controls a Direct Memory Access (DMA) engine to transfer data through the On-Chip Bus (OCB), to a Memory Controller (MEMCTL) which ultimately writes the packet into system SRAM.

Working from left to right in the Figure 2 example, there are 5 equally sized memory regions, MR1 through MR5, in SRAM. Each region has been sized to accommodate the maximum packet expected to be received. In this example, the memory is contiguous for two of the regions but is otherwise scattered. Incoming packets will be written first to MR1, then MR2 and so on, with MR5 linking back to MR1.

In typical producer-consumer fashion, each region would not be overwritten until consumed by the host system. Simple linked-list buffers should not be employed if there's any possibility of overwriting a buffer until it has been completely consumed. A non-linked list of descriptors, albeit with host software intervention, can be fashioned into a scatter-gather controller.

On the BAE SpaceWire ASIC, a receive descriptor is constructed by the host software to point to the target buffer in SRAM, by address and by length. The address of that descriptor is written to a RIF register and the RIF starts the process, which terminates when either the specified buffer length is reached or an end-of-packet marker is received. This

effectively prevents the designated receive buffer from overflow but does not terminate or spill the packet. Remaining packet data will consume additional buffers until an end of packet marker is received, complicating recovery.

C. Programmable Per-Port Maximum Packet Length

On GOES-R, the GSFC-designed router core embedded within the BAE SpaceWire ASIC includes an additional feature to prevent overflow on receive, and to prevent a stall due to blabbering transmit; a maximum packet length feature. This feature, if enabled, will disconnect the link and append an Error End of Packet (EEP) to any packet that exceeds the programmable maximum length. This will of course also spill the remainder of the errant packet. With the BAE SpaceWire ASIC, the limit applies to all ports, but ideally each port would have separate limits. The other router used on GOES-R, the Aeroflex 4-port router, has no such feature.

III. Receive Buffer Depth & Router Defect

Receive FIFO depth of a router may be configurable within a soft core for an FPGA. During development, pipeline side-effects should be taken into consideration to avoid potential stalls and data dropout. Router defects may exist which may cause a stall which will only clear when the receive buffer depth is adequate to compensate for the defect.

A. Logic Value vs. On-The-Wire Value

The value of a transmitter's credit count may be different in the logic of a transmit block then the value on the wire due to pipeline delays, synchronization delays and logic delays. A receive FIFO configured to a depth of only 16 bytes, and with a one byte pipeline delay, may initially transmit 2 FCT's, but not issue a successive FCT until 9 bytes are transmitted to it, and remain one byte delayed thereafter.

The transmitter may also have logic delays that cause its internal credit count to fall behind the value on-the-wire.

B. Router Transmit Block Defect

The Aeroflex 4-port router designed into GOES-R had a latent defect that was not exposed until integration testing with an instrument that had configured a receive FIFO depth of only 16 bytes. When the router's internal credit count transitioned to zero on the same cycle that an FCT was received, the router would stall due to the defect. The router would resume transmission when another FCT was received.

C. Indefinite Stall Condition

Although the router could break the stall when yet another FCT was received, the instrument configuration prevented further FCT transmit due to the shallow receive buffer depth, combined with the receive pipeline delay, causing the stall to last indefinitely.

D. GOES-R Stall Resolution

To avoid a reconfiguration of the instrument's FPGA, the transmit speed through the router was slowed to avoid the stall condition, which occurred only when the internal credit count transitioned to zero on the same clock as when an FCT was received. By slowing the transmit clock, the router's internal logic no longer lagged behind the on-the-wire value. Subsequent builds of the instrument did incorporate a deeper buffer to further mitigate the problem.

IV. PIPELINING PITFALLS

Pipeline stages were in part responsible for the problem described above, and is the main culprit in another issue encountered with GOES-R.

To avoid buffer overrun a producer-consumer buffering model was employed by the GOES-R spacecraft. Unlike linked-list operation, there is a time gap (latency) from DMA completion until the RIF is programmed with the next receive buffer. The bug, described below, was never observed when in linked-list mode.

A. Problem Description

Under nominal telemetry conditions a receive buffer was made available via the RIF (see Figure 2 above) prior to filling the Rx FIFO. Telemetry bursts, however, would exhaust the supply of receive buffers in SRAM until the downlink (consumer) could catch up. On occasion, the Rx FIFO and a 4-byte pipleline stage (not shown) would fill before a newly-freed buffer could be assigned to accept the packet via the RIF. Data did not overflow from the Rx FIFO because credit count depletion would stall the packet. There was a bug, however, in the pipeline stage that could drop those leading 4 bytes from the incoming packet. The GRDDP CRC would match, but half of the GRDDP header would be missing from the receive buffer. Several methods were utilized to address the bug.

B. GRDDP Transmit Retry

The GRDDP retry mechanism, for normal data packets, assures that those cropped packets will be retried, since header checks fail and the packets would not be acknowledged. Network traffic is increased, however, and dropouts of urgent message packets are possible since they are not retried.

C. Buffer Utilization

There wasn't enough physical memory to allow linked-list operation, but all remaining SRAM was dedicated for receive buffers, which helped quite a bit, but was not enough. Another technique considered was to dynamically utilize the allocated receive buffer space vs. a ring of fixed-size buffers. With this method, the start address for the next packet would depend on the size of the current packet, rounded up per DMA constraints. Pending on downlink transfer completion could be reduced or eliminated given the increase in number of buffers. While a sound idea, it was more complex, and would lead to additional processing latency.

D. Processing Delay Reduction

The assignment of a buffer freed by the downlink to the RIF had been a function of the main processing loop. By moving that function to an ISR context the mechanism behaved more like a linked-list. The addition of this latency reduction proved a sufficient workaround.

Conclusion

Real-world systems may be vulnerable to serious faults that can result even when there is no apparent violation of the ECSS standard. Additional capabilities are required of routers and nodes to avoid these pitfalls. All components in a system must be thoroughly researched, including the experience gained with those components by others.

References

[1] A. Krimchansky, W. Anderson, C. Bearer, “The Geostationary Operational Satellite R Series SpaceWire Based Data System Architecture”, NASA Goddard Space Flight Center, 2010

[2] European Cooperation for Space Standardization, ECSS-E-50-12A, “SpaceWire – Links, Nodes, Routers and Networks”, 2003

[3] NASA Goddard Space Flight Center GOES-R Project "GOES-R Reliable Data Delivery Protocol", 417 - R - RTP - 0050 Version 2.1, 2008

[4] L. Haynes, G. Rakow, “BAE SYSTEMS SpaceWire Router Specification: 4 Port, 2 External Interface”, 2005

[5] J. Marshall, S. Santee, M. Hanley, J. Robertson, D. Stanley, “Leveraging SpaceWire Network Prototyping to Create Flexible SpaceWire Components and Support Software”, 2011

[6] Aeroflex, “UT200SpW4RTR 4-Port SpaceWire Router Preliminary Datasheet”, 2013