High-Speed Zero-Jitter IO Management And Network-Based Solutions

John C. Glisson

Piedmont Automation Inc.

Introduction

While the development of network-based distributed IO management has greatly simplified the design, construction, and operation of industrial process control, digital IO applications requiring the highest-speed zero-jitter low-latency response times still require costly and complicated FPGA-based designs. By design, fieldbuses reduce multiple parallel input and output channels into a single serial channel for communication with a central IO processing unit responsible for management of the serial bus, processing task timing and interrupts, and combining data from multiple fieldbus systems. Unfortunately, this model introduces constraints involved in serialization of the input and output data that may render network-based solutions incapable of handling certain tasks or improperly implemented networking solutions that leave ambiguity about the machines response times. This paper describes the crafting of fieldbus systems for minimal response timeouts by describing multiple models of IO-bus management, illustrating their comparative advantages and disadvantages, presenting formulae for calculation of their latency and jitter, and validating the formulae and models with a use-case implementation using B&R hardware.

Background

A. Simple model of IO management over a fieldbus

The following steps are necessary for a Digital Input (DI) to be registered on a PLC-

1. The physical event to be measured happens and is observed in a register on a DI card
2. A software interrupt triggers the input registers on the remote IO module to copy into a communications buffer, from which the SI frame is built
3. The bus "handshakes" with the network master on the CPU at its assigned network timing slot and transfers the SI frame to the CPU
4. The network master CPU copies the data from the bus module to the input buffer for application code access

Further, the following steps are required to set a Digital Output (DO)-

1. Copy of data from the appropriate internal variable to the output register buffer
2. Software interrupt triggers sending the SO frame from the master network interface on the CPU the remote IO module
3. Copying of data from SO buffer on IO module to appropriate register for DO
4. Triggering of releasing the DO register to the physical outputs

The speed of which can be affected by the following factors-

1. Number of DI channels per remote IO module
2. Time required to create the buffer per channel
3. Number of remote IO channels to be read
4. Size of remote IO channels

The simplest model of IO communication makes the following assumptions-

- SI frame generation is finished by the time the bus master is ready for the SI frame transmission to ensure minimal latency reading the inputs. In cases where this cannot be assumed, the SI frame communication must be delayed one cycle after SI frame generation
- The master is ready for receiving the SI frame at the same time it is ready to send the previous cycle's SO frame
- Time between the receipt of the module's SO frame and setting the DO register is minimal

Note: citation needed for all of the above!

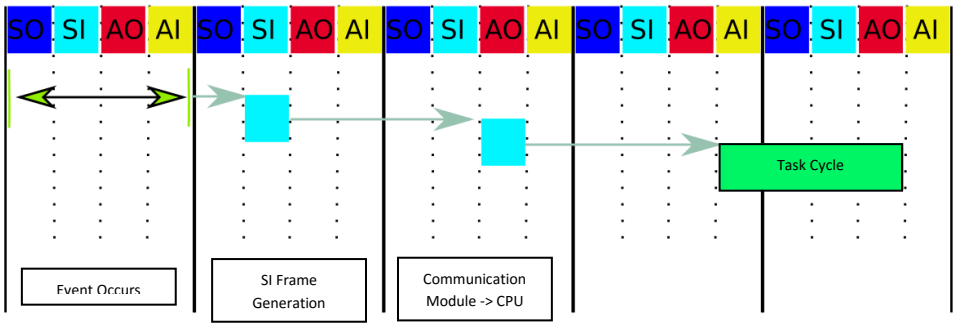
The advantages of the model presented above are as follows-

- Multiple modules' input data can be ensured to be available at the beginning of the processing task cycle
- Processing of input registers into communication frames can be done in parallel
- After processing, data can be sent in a synchronized fashion to the appropriate output channels
- The fieldbus master can synchronize the modules such that each input frame is guaranteed to represent identical discrete-time segments. This means jitter-free communication between modules.

The disadvantages of the model presented are as follows-

- At least a bus cycle will be required between reading the DI and setting the appropriate DO
- Digital signals too short for the module to register (shorter than the bus cycle time) cannot be measured reliably
- The bus cycle time cannot be shorter than the amount of time required to process the data (to ensure that the appropriate registers have been set before transmitting the SO frames).
- Time delays between inputs and outputs are relegated to specific intervals.

Altogether, A chart describing the IO gathering and bus communication looks as follows->



The advantages of the previous model are as follows-

- Being the most basic model, it can be implemented on nearly every existing fieldbus platform possible
- Maximum amount of time is guaranteed to the module for frame generation
- The only source of unpredictability between the input event and output event is the actual time between cycles that the event occurred.

The disadvantages-

- Seven cycles of latency are required between inputs and outputs

The model for the total latency is then

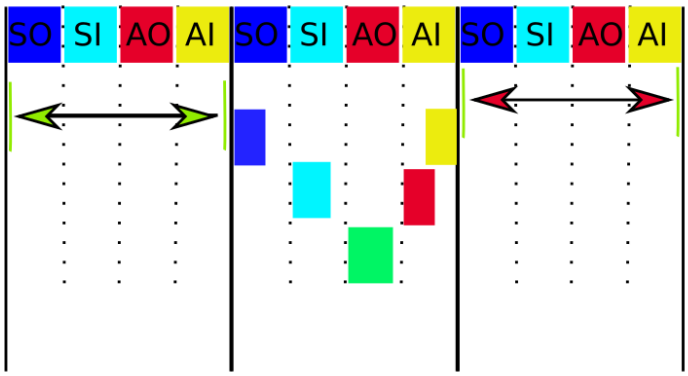
$$L_{Total} = [T_{Event\ to\ Bus} + T_{SI\ frame\ Cycle} + T_{Bus\ Cycle} + T_{Bus-Task\ offset} + T_{task}]$$

For systems where the bus, module, and task are synchronized, this becomes

$$L_{Total} = [T_{Event\ to\ Bus} + 4 * T_{cycle}]$$

[More charts and equations explaining each step go here]

Finally, minimum latency can be achieved by staggering the tasks as follows-



Disadvantages-

- Only the second half of the task cycle is available for the task to complete
- If the task has not completed in the appropriate amount of time, old data is sent to the outputs

Advantages-

- Minimum latency is achieved in this fashion

The total latency is then

$$L_{Total} = [T_{Event\ to\ Bus} + T_{cycle}]$$

Solution

A. Optimization of Input Latency

Moving the start time of the task class up by ½ of the bus cycle by delaying the SI frame until the middle of the cycle means that the SI frame information can be received and processed earlier than it would be otherwise. The downsides to this are that you must rely on the SI frame to be available at that time and that you only have half of the cycle to process and build the SO frame.

[Add diagram here showing comparison of SI frame a beginning and middle of cycle]

The resulting latency between the signal being read and processed on the PLC is the input frame generation + ½ of the bus cycle + ½ of the system task cycle, the sum of which is usually just one task cycle

Equipment:

- X20CP1584 Firmware V1.2.0.0
    - RS-232 Interface IF1
    - Ethernet Interface IF2
    - Powerlink Interface IF3
    - USB Interfaces IF4 & IF5
    - X2X Interface IF6
        - X20DI9371
            - Hardware Filter <= 100us
            - Software Input Filter 0-25ms
            - Current Sinking
        - X20DO9322
            - <300us Switching delay
            - Current Sourcing

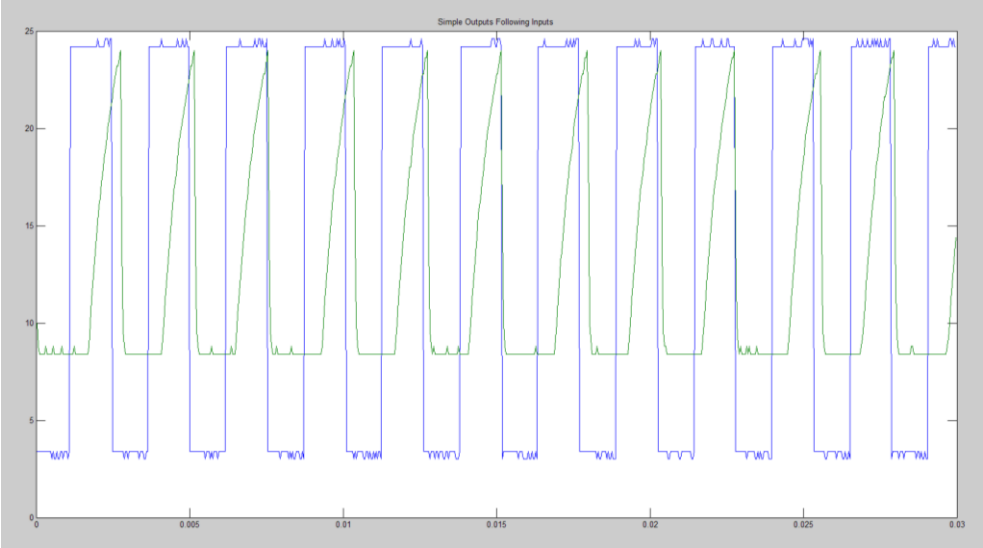
Settings->

- X2X bus cycle timer used as task timer = 400us
- Inputs delayed to middle of cycle
- Outputs delayed to end of cycle
- Interface optimized for minimum latency

Experiment->

- Using a simple function generator, generate multiple frequencies of square waves
- Document minimum, maximum, and average latency for multiple settings of the input/output delays and interface optimization settings

Results->



Bus Optimization	Inputs Delay	Outputs Delay	Average Latency(us)	Min. Latency(us)	Max Latency(us)
Min. Latency	No Delay	No Delay	424	318	732
	Delay to Middle	Delay to End	412	244	652
Data Throughput	No Delay	No Delay	652	468	824
	Delay to Middle	Delay to End	496	352	740

For n=12, the result is that it the model presented above minimizes total input-output latency.