Synchronous Protocol for Generators

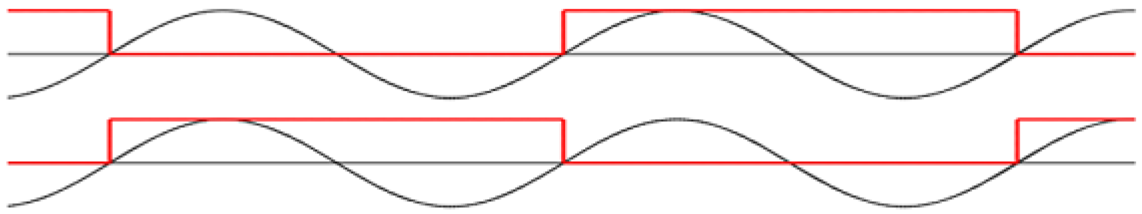
Author: Liyao Tang - u6142160 (21/10/2017)

Abstraction

In this assignment, two possible designs of protocol for synchronization are implemented and will be discussed. The first design is based on Master-Slave model and the second follows a Detect-React fashion. Based on the second design, a synchronous data transmitting protocol is designed and has been tested to work.

Design - Synchronization Protocol

The incoming signal to be mentioned below is a square wave, whose rising and falling edge denote the start of a sine wave cycle. In such encoding, the two different square waves shown in the graph below represent the same sine wave.



Master-Slave Model

In this model, the Master generate the source of the signal and slaves will take the incoming signal as clock source to generate and broadcast the signal to the rest of the network.

Assumption behind this design is that the delay time for transmitting one bit from the Master board to the end of the network is negligible.

The protocol consists of two basic phases, which are claiming-for-Master phase and stable phase. After the system startup, each board delay for a random time and the first board finishes its delay will claim itself as Master by output signal by toggling the ports. Board that receive the incoming signal before they finish their delay will realize that there is a Master in the network and start to synchronize with the Master and broadcasting the Master's signal. It is possible that two boards may claim to be Master at the same time, which will trigger another claiming-for-Master phase, until one is eventually selected.

To deal with cyclic network, we need to define a preference over the incoming signal such that slaves can broadcast the signal properly. For each slave, the first received incoming signal is selected as the signal from the Master and the board connected to that port is called direct master. The slave will then broadcast the signal to all its neighbours except for its direct master. In this way, the slave will stop redundant signals and make sure there is one and only one Master in the network. But such design still has constraint over the topology of network because it assumes each local port connects to different board. It also assumes that transmitting bit directly between board A and board B is strictly faster than transmitting a bit from A to B with some other boards in the middle. With those two assumptions, the protocol will eventually stabilize and the maximal drift between the sine waves on two different board is bounded by the maximal delay of transmitting one bit from the Master to the end of the network.

The design can be further extended to include self-adjustment strategy so that slaves can claim themselves as Master if they believe they are more accurate than the Master is. However, this extension need to assume the network will reach to a state where everyone is satisfied. In addition, our assumption makes the network hard to scale because the delay from Master to the slave will be accumulated as the network grows.

Detect-React Model

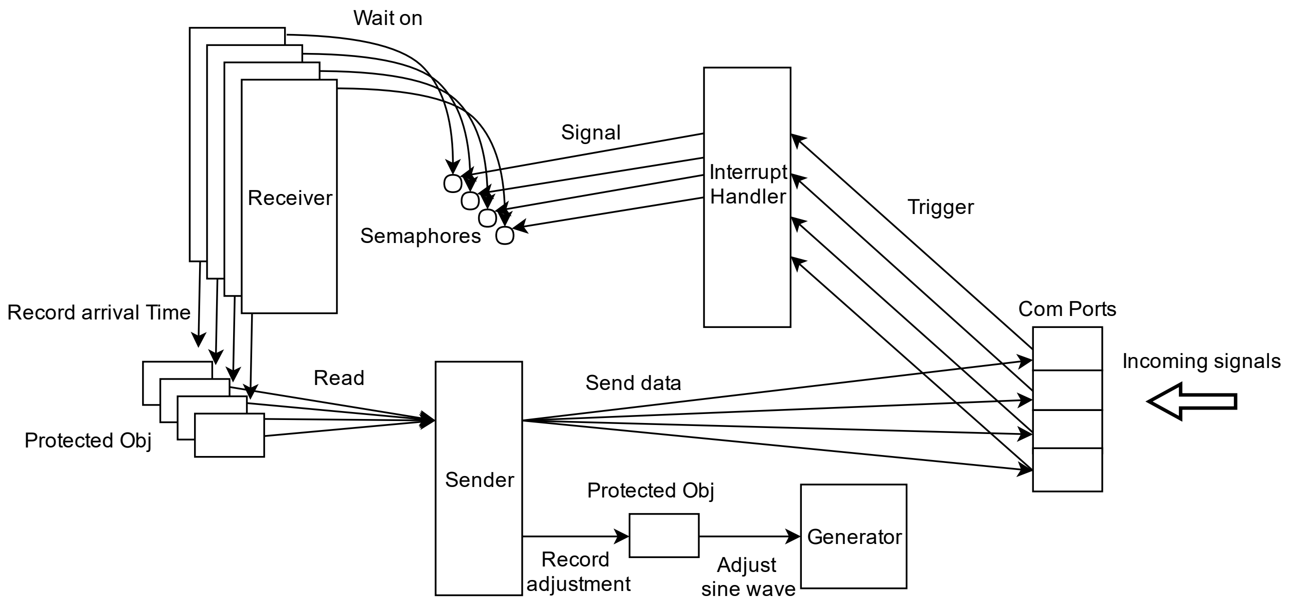
In this model, each board will generate their own signal and receive signals from all directly connected boards for synchronization.

Assumption behind this model is that the delay time for transmitting one bit from one board to its direct connected board is negligible to an extent. It is also assumed that each board has its own clock and measure of time are not consistent across boards.

To synchronize with incoming signal, the board performs two-step synchronization. The sender of the board first synchronizes with the incoming signal and then informs the generator to synchronize with itself after it becomes stable. In the first step, board records time for each incoming signal correspondingly and is thus able to calculate the period and timing behaviour of its direct neighbours and then adjust its behaviour accordingly. Each board will only synchronize with others when the selected incoming signal is stable, which provide a way to prevent two-way synchronization. That is, when board decides to synchronize with an incoming signal, it will toggle the selected port so that its output signal is disrupted and thus neighbour boards will not synchronize with it at the same time.

When boards form a network, boards need to define a preference over incoming signals. For example, the board can prefer lower frequency to higher frequency so that the network will eventually synchronize with the board with lowest frequency.

Each board consists of three modules, which are generator, receiver and sender and are separated into corresponding tasks. A graph is provided below to demonstrate their relation.



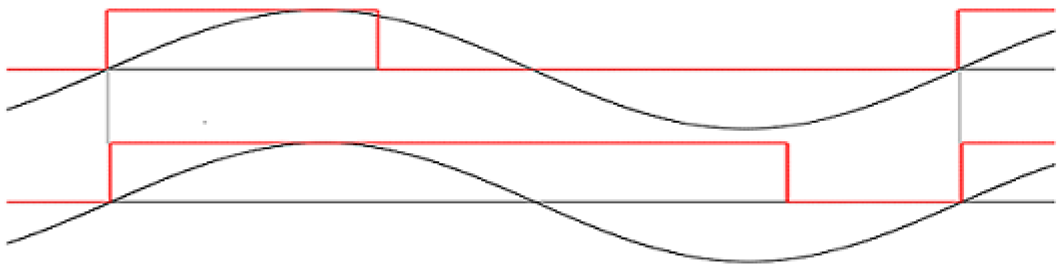
Because of the delay between the actual time when signals arrive and the time recorded locally, a certain drift between calculated incoming sine wave and local sine wave is allowed, which is 5 microseconds in the current design, with two generators on the same board. The delay is inevitable without further hardware support because the time need to be recorded in a protected object but touching protected object inside an interrupt handler will result in a frozen system. The allowed drifting time span is the maximal time for boards to handle interrupts for each incoming signal, context switch to the receiver, and record the time into the protected object.

Due to the limited time, some extension are not included in the current design, which includes hardware support from DMA and timer. With DMA, sender can be further decoupled and does not need to toggle the port directly but needs only to set up a DMA and corresponding timer. With timers, each port can have its own dedicated timer to record the arrival time of each rising and falling edge.

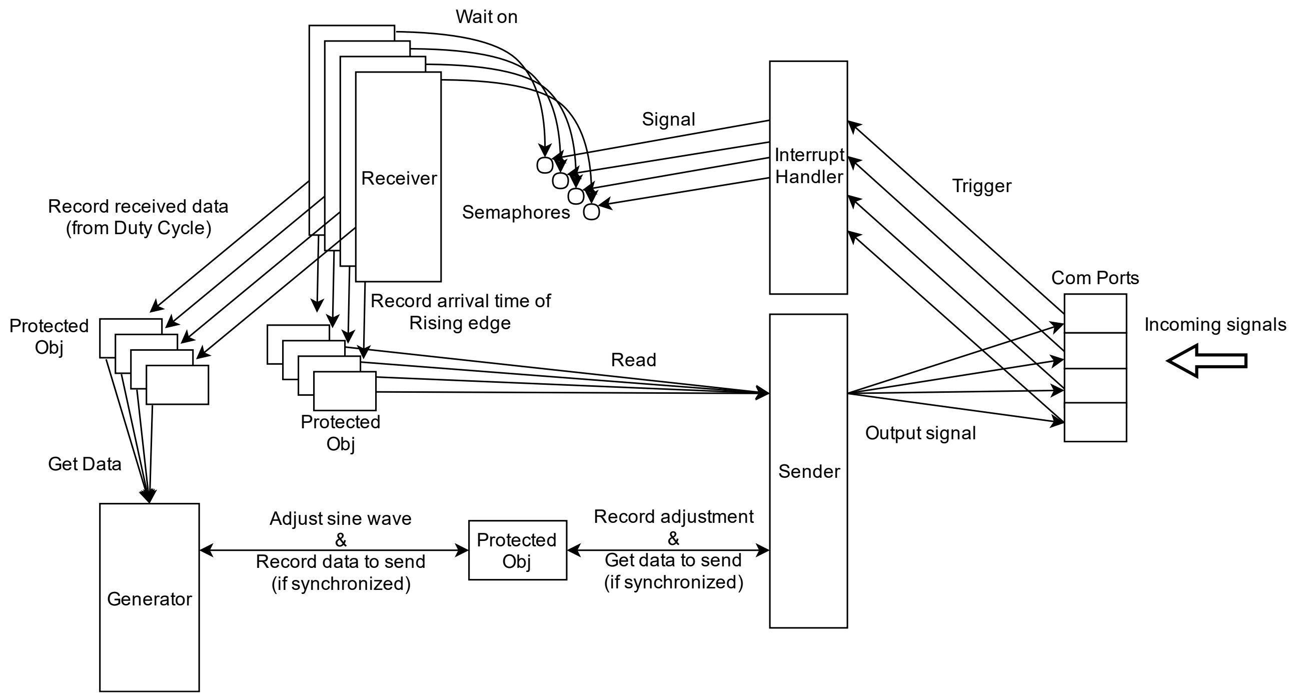
Duty Cycle

To implement duty cycle, I refer to Pulse Width Modulation (PWM), which inspires me to design my protocol for duty cycle. In the protocol, the rising edge of the square wave signal denotes the start and end of a sine wave cycle and the proportion of 1 in the square signal denotes the value of data.

To achieve this, it is important to synchronize on the rising edge of the square wave, which requires a different encoding of sine wave. The graph below shows an example encoding where two synchronized sine waves carries different data along.



Now the relation between tasks can be shown as the graph below.


Wait on
Signal
Receiver
Interrupt Handler
Semaphores
Trigger
Record received data (from Duty Cycle)
Com Ports
Incoming signals
Protected Obj
Record arrival time of Rising edge
Read
Protected Obj
Output signal
Get Data
Sender
Adjust sine wave & Record data to send (if synchronized)
Protected Obj
Record adjustment & Get data to send (if synchronized)
Generator

Compared to the design in previous graph, the logic to synchronize the base wave, sine wave in our case, is similar. But the fact limits the ability of sending data that it is inevitable to have a falling edge between two rising edge, which requires the protocol to define a certain behaviour pattern in synchronizing such that board can agree on an invalid number so that the board will not mistake those signals generated from synchronizing process as incoming data. This limitation is also caused by the design because the decoupling between receiver and sender makes receiver not able to recognize if signal is from unsynchronized port as receiver does not know its local frequency.

During the test, the effect of accumulated shift still exists but within the control as board can resynchronize within a few cycles and then continue to send data to each other. It is worth noting that the protocol for duty cycle still maintains an allowed detected delay down to 5 microseconds with two generators on the same board.

Conclusion

In this report, two protocol for synchronization are discussed. The Master-Slave model is more efficient and stable in terms of synchronizing and one-way message passing in a relative small network. However, although the Detect-React model is less stable as the phase shift can inevitably accumulate without further hardware support, it can scale to larger network because the delay of transmitting bits across the network does not accumulate. Furthermore, it is more efficient in transmitting message as it allows two-way transmission using duty cycle.