ECE 5710-6710
10/18/2023

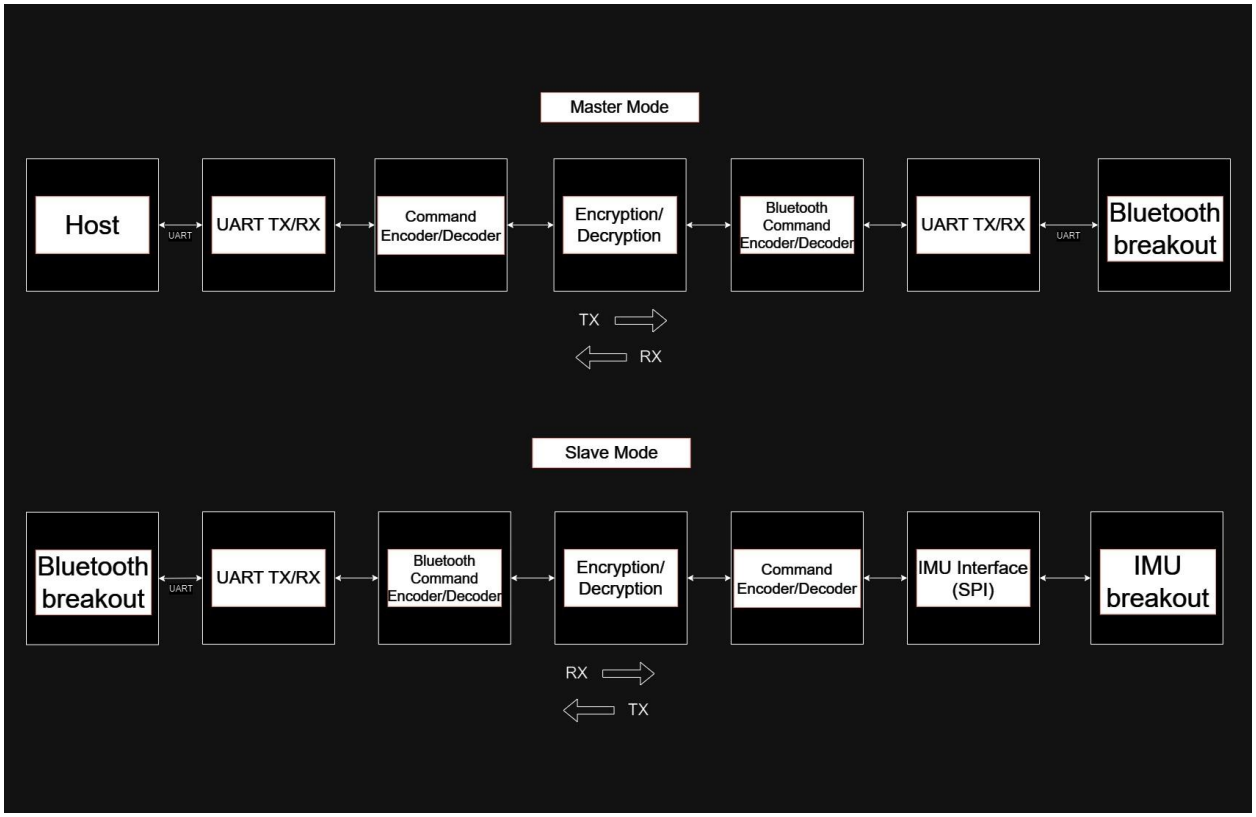
Project Proposal: IMU Bluetooth Communication Module

Group: Noah Mecham, Elmir Dzaka, Mike Mercer, Sean Koo

Motivation:
The idea for our module stems from a desire to implement a 1-to-many communication device. Devices that allow a 1-to-many connection are common in the radio and satellite communication world, however, we wanted to target something that would be able to be incorporated with COTS parts and could be integrated into a relatively simple commercial or personal system. The design that we lay out is for an ASIC that can be communicated with via UART, will transmit a message via bluetooth to any system using our ASIC in a slave mode, and interface with an IMU to retrieve some data. An application of our module would be in a swarm of drones, and would allow one master to obtain data about a group of drones.

High Level Description:
Our module will have two operating modes, a master and slave mode. For a module set in master mode a host device will be connected to the module via a UART connection. The host will send a command following a command structure we will provide. The command will be decoded, then encrypted using a one-time pad, and then re-encoded in a structure suitable for the bluetooth device connected. Following the command the module will go into a listen mode and would reverse the process upon receiving a response. In the slave configuration our module will wait for a command to come in from the bluetooth device. After receiving a command, the command will be decoded from the bluetooth packet, then decrypted using the same one-time pad, and then the original command can be decoded. Once the command is known we can retrieve the requested data from the IMU. The slave module will then reverse the process and send a response containing the IMU data back to the master.

Block Diagram:



The above block diagram depicts the various functional blocks that our design will require. A majority of the blocks work in both master and slave mode, reducing the amount of specialized blocks we need. Our UART block is already built thanks to the homeworks. Ideally, we should be able to find a SPI driver in the open source realm. The command encoder/decoder block will have to be custom. The encryption/decryption block could go either way. There are lots of open source implementations out there, but they could be too large for our chip. If an open source solution can't be found the idea is that we would implement a simple one-time pad encryption scheme that would provide basic encryption with minimal impact.

Our design is somewhat ambitious for a 6 week project. Due to time constraints we will have a minimum set of deliverables that consists of the UART, command encoder/decoder, bluetooth command encode/decoder, and SPI interface. The encryption block will be worked on after these other blocks are fully tested. If time permits we will include that block.

Testing options:

In the process of validating a Verilog master-slave architecture, it's essential to establish a robust testbench alongside the instantiation of the master and slave modules. This testbench is responsible for generating input stimuli, monitoring output responses, and assessing the design's correctness. Key components, such as clock generation and signal propagation, are integral to the

typical testbench structure. Using a simulation tool like ModelSim, the design is executed, and a waveform viewer is utilized to inspect the behavior, with a primary focus on confirming accurate data transfers between the master and slave components. Functional verification is a pivotal step in ensuring that the master-slave architecture operates as intended, encompassing checks for precise timing, synchronization, and data integrity during communication. Additionally, assertions are integrated into the testbench to automatically validate specific conditions, facilitating formal verification and bolstering confidence in the design's reliability and precision. This comprehensive testing approach ensures that the Verilog master-slave architecture fulfills its intended purpose, aligning with operational requirements and design specifications.

We estimate that we will need 16 pins. The pins will be for the following:
UART TX x2 (one for TX to host, one for TX to bluetooth board)
UART RX x2 (one for RX from host, one for RX from bluetooth board)
CTS (clear to send for bluetooth board)
RTS (read to send for bluetooth board)
CS (for SPI)
SPC (for SPI)
SDI (for SPI)
SDO (for SPI)
Ref Clock
Reset
Mode Select
Vdd - 3.3V (Needed for logic level of UART and SPI)
Vdd - 1.8V (Operating Vdd for chip)
Vss

Do you plan on fabricating your design: Yes.