# **Communication Network**

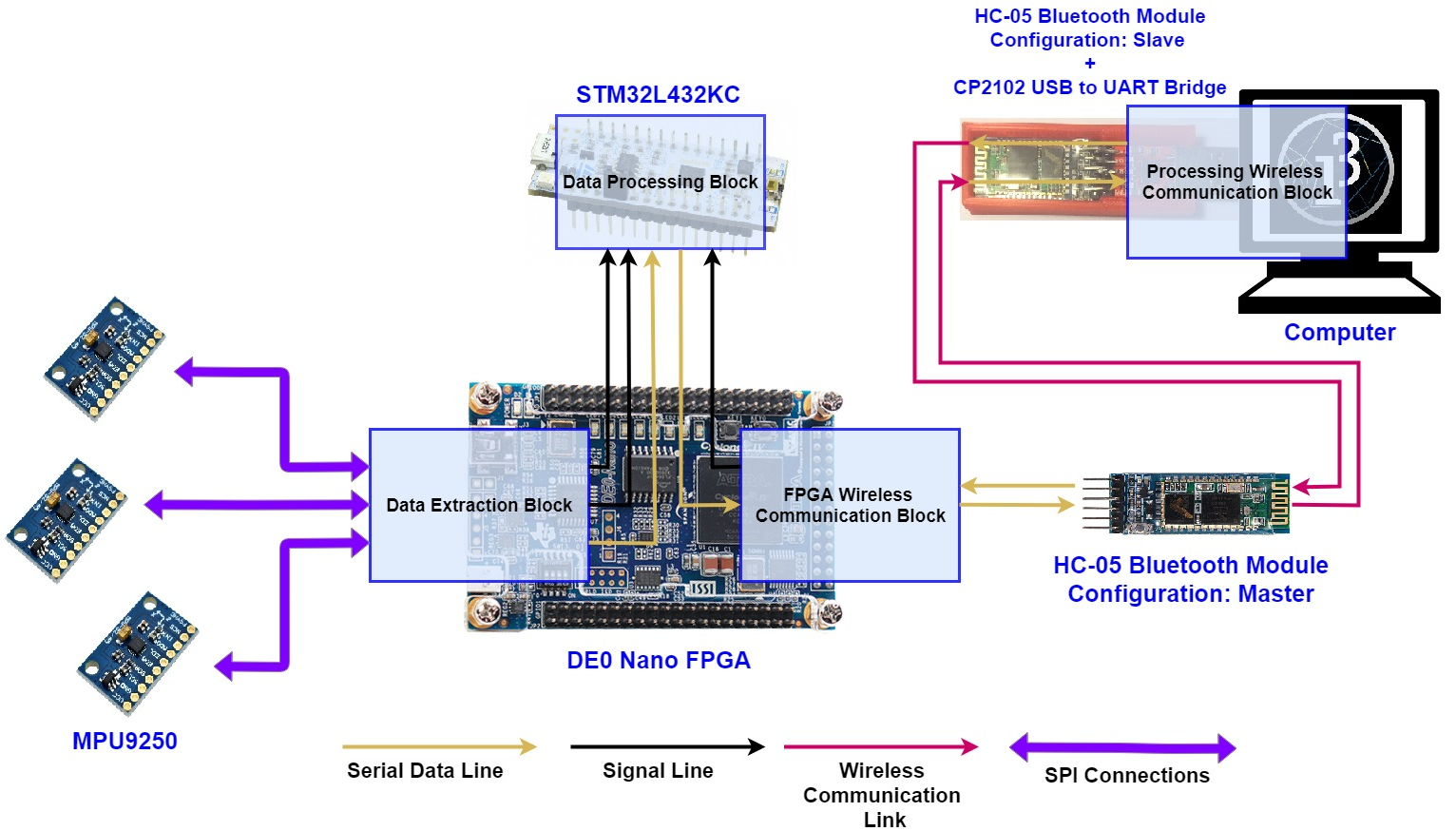
The purpose of this communication network is to provide a means of moving data from three internal measurement units (IMUs), through the system, to the destination which is the processing simulation. This communication network consists of 4 distinct sections; the data extraction block, data processing block, FPGA wireless communication block and the processing communication block. Figure 1 provides an overview of how the data moves across hardware before it arrives in the processing simulation. This part of the design section is concerned with the design of the network and how it enables the data to move before it reaches the processing simulation. The Blocks inside of the FPGA function at a clock speed of 1MHz and each component is synchronised through the same clock. Because each component in the system is connected to this clock, the connections to the clock have not been shown to maintain clarity in the upcoming diagrams.

Figure 1: Diagram depicting the flow of data across hardware.

## **Data Extraction Block**

The ‘Data Extraction Block’, as the name suggests, is responsible for the extraction of data from the internal measurement units. The data is extracted using the serial peripheral interface and then is stored in a buffer where it waits to be transmitted to the ‘Data Processing Block ‘using the serial peripheral interface once again. To achieve parallel data extraction, three IMU extraction components are present, each consisting of the above mentioned serial peripheral interface component, buffer, and also an address loader, data extraction controller and data transmission controller. The address loader is used to load addresses to the SPI such that it can request data from registers inside the MPU9250 which contain the gyroscope and accelerometer data. The two controllers are used to synchronise the extraction and transmission process.

Because the there are three IMU extraction blocks within the data extraction block but there is only one serial peripheral interface component that communicates with the data processing block inside of the STM32L432KC board, the ‘FPGA-STM32L432 SPI Access Controller’ is implemented to control the access such that data can be sent without any conflict. Figures 2 and 3 show all the components that can be found inside the data extraction block and how they are connected.

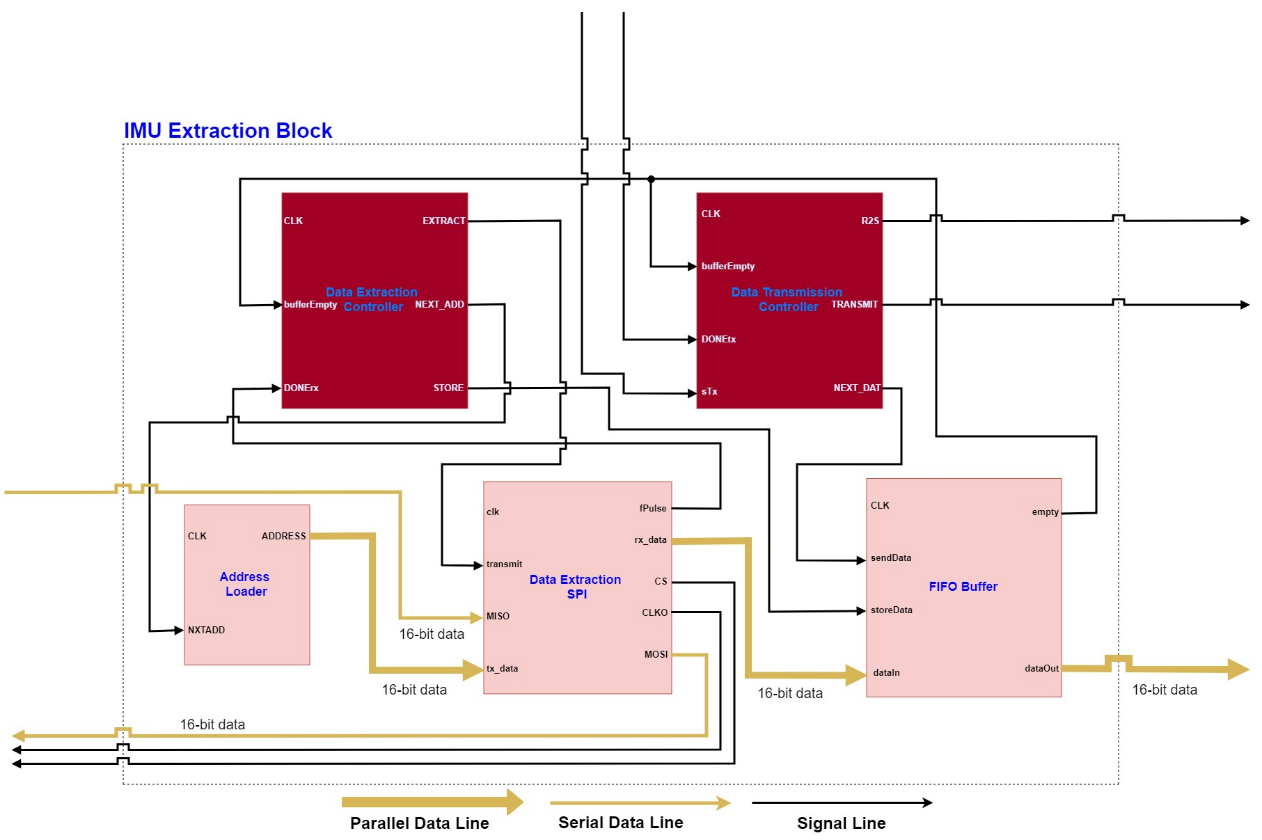
Machine generated alternative text:
Data Extraction 
Block 
CLK 
r2slMU1 
txlMU1 
datalMU1 
16-bit data 
dataOut 
TRANSMIT 
FPGA-STM32 
clk 
MISO 
cs 
CLKO 
FPGA-STM32rx data 
r2slMU2 
SPI 
Access 
txlMU2 
Controller 
datalMU2 
r2slMU3 
txlMU3 
datalMU3 
DONEtx 
IMU Extraction 
strtTx3 
strtTx2 
strtTx1 
R2S 
SPI 
tx data 
transmit 
DONEtx 
MOSI 
fPulse 
R2S 
DONEtx 
16-bit data 
R2S 
Text 
Blockl TRANSMI 
IMIJ Extraction 
Block2 TRANSMIT 
MISO 
IMLJ Extraction 
Block3 TRANSMIT 
dataOut 
16-bit data 
MISO 
16-bit data 
O 
x 
dataOut 
o 
16-bit data 
MISO 
16-bit data 
16-bit data 
Serial Data Line 
O 
x 
dataOut 
o 
16-bit data 
O 
x 
o 
16-bit data 
16-bit data 
Parallel Data Line 
16-bit data 
Signal Line 

Figure 2: Diagram depicting the component composition and connections between the inside the Data Extraction Block.

Figure 3: Diagram depicting the component composition and connections inside the IMU Extraction Block that is part of the Data Extraction Block.

### **Data Extraction Controller**

The extraction controller is responsible for the extraction and storage of the IMU data that it extracts. The controller is a state machine that consist of several states which it transitions between based on the input in order to accomplish its task. Table 1 describes the functions that the inputs and outputs of the extraction controller serve to drive the state transitions. The following descriptions of the states explain how the controller operates.

|  |  |  |  |
| --- | --- | --- | --- |
| **Pin Name** | **Input/Output** | **Usage** | **Signal Type** |
| CLK | Input | The clock input allows the component to be synchronised to a common clock across the system. Every component in the system has an input clock. | Clock |
| bufferEmpty | Input | The FIFO buffer can signal to the controller through this pin that it is either full (0) or empty (1). | High/Low |
| DONErx | Input | The SPI can signal the controller that a new sample has been extracted and can be stored. | One clock cycle pulse |
| EXTRACT | Output | This pin is used to command the SPI component to start the extraction of another sample. | One clock cycle pulse |
| NEXT\_ADD | Output | This pin is used to command the address block to load the next address to be transmitted to the IMU through the | One clock cycle pulse |
| STORE | Output | This pin is used to command the FIFO buffer to store the newly extracted data piece. | One clock cycle pulse |

Table 1: Description of inputs and outputs of the Data Extraction Controller Component.

* When in the ‘IDLE’ state, the component waits for the bufferEmpty signal to go high in order to start the data extraction process. The controller transitions to the ‘NXTADDR’ state when the bufferEmpty signal goes high.
* When in the ‘NXRADDR’ state, the controller sends the ‘NEXT\_ADD’ signal to command the Address Loader component to load the new address on its output for the SPI master component to use. The controller then automatically transitions to the ‘STRTrx’ state.
* When in the ‘STRTrx’ state the controller sends a signal using the ‘EXTRACT’ pin to command the SPI component to start the transaction. The controller then automatically transitions to the ‘WAITINGrx’ state.
* When in the ‘WAITINGrx’ state, the controller waits for the ‘DONErx’ input to go high. This signal through this input notifies the controller that an SPI transaction has been completed. When this signal is received, the controller transitions to the ‘STOREstate’.
* When in the ‘STOREstate’, the controller sends a one clock cycle pulse using the ‘STORE’ output pin to command the FIFO buffer to store the data that the SPI extracted in the latest transaction. The controller then increments counter which tracks the number of transactions; this way the controller knows how many more data samples need to be extracted to have a complete set of gyroscope and accelerometer data samples (see Address Loader component). If the counter value is less than 11 then the state transitions back ‘NXTADDR’ state, otherwise the controller resets the counter to zero and transitions to the ‘W4BFFRrx’ state.
* When in the ‘W4BFFRrx’ state, the controller waits for the bufferEmpty signal to go low. This state is used to let the FIFO buffer finish its internal processes before it sends the signal as the buffer is designed to only holds as many samples as there are in a complete set of gyroscope and accelerometer data before it becomes full. If the controller transitioned to the IDLE state before the bufferEmpty signal transitioned to logic 0, the controller would continue initiating new sets of transactions causing data corruption in the later stages of operation such as data transmission. Adding this state between the IDLE state and the ‘STOREstate’ prevents this from happening. When the bufferEmpty input transitions to logic 0, the controller transitions to the IDLE state.