LMS Hardware Engineer Assignment

Message Extractor

1. Draw a diagram of your chosen design (feel free to use https://asciiflow.com)

Please find the diagram in <msg_parser\doc\architecture.txt>.

2. Write an elegant, synthesizable solution for the message extractor in RTL/Verilog/SystemVerilog using the skeleton provided. And verify it against the given sample inputs. NOTE: If Verilog/SystemVerilog is too big of a step, please feel free to use VHDL, in that case the diagram and the explanation/comments of your code will have a bigger impact. The reference is the bare minimum that the design must be able to handle. The candidate is encouraged to add more test cases that cover as many scenarios they can think of to showcase their design skills.

Please find the solution developed in <msg_parser\src> and <msg_parser\sim>. If you have modelsim, you can launch the simulation using <msg_parser\scripts\msim_compil.bat>.

3. What is the bottleneck of your design/code? (what can limit the maximum frequency?)

This design doesn't have an important amount of combinational logic, so it should not have much impact on the maximum frequency, but actually the most critical path which can limits the maximum frequency is the path from the msg_length register to the payload counter register.

4. Please explain how would your design change if the range of message lengths change from min=8B max=32B to: a. min=1B ; max=32B b. min=8B ; max=256B

This design using FIFOs should be easily scalable for differents message length. The data FIFO is already read byte by byte, it should not be an issue to lower the minimum message length. The state machine should stay the same. For the b. option, where the msg_len can goes up to 256 bytes, it is the same thing, the design should not change, but it will change the size of signals such as the payload signal which will need 2048bits.

5. What are the trade-offs for the chosen approach?

The chosen approach implies another input clock, which has to be around 10 times faster to decode the data. Indeed, we receive a 8 bytes AXI DATA and we read it byte by byte in a FIFO, so we need to decode it faster than the receiving speed. It is the most simple and most scalable approach if the message length changes. As said in 4. this design should be easy to adapt for different message lengths. However, it makes this design dependent of the speed of the input data clock. If the input data speed is already high, it might be difficult to generate another clock which is 10 times faster. Also, the latency might not be the best as we depend on write/read latencies on the FIFO and on the decoding speed. On simulation, we have a latency of 1-2 cycles (cycle difference between the output and the input data).