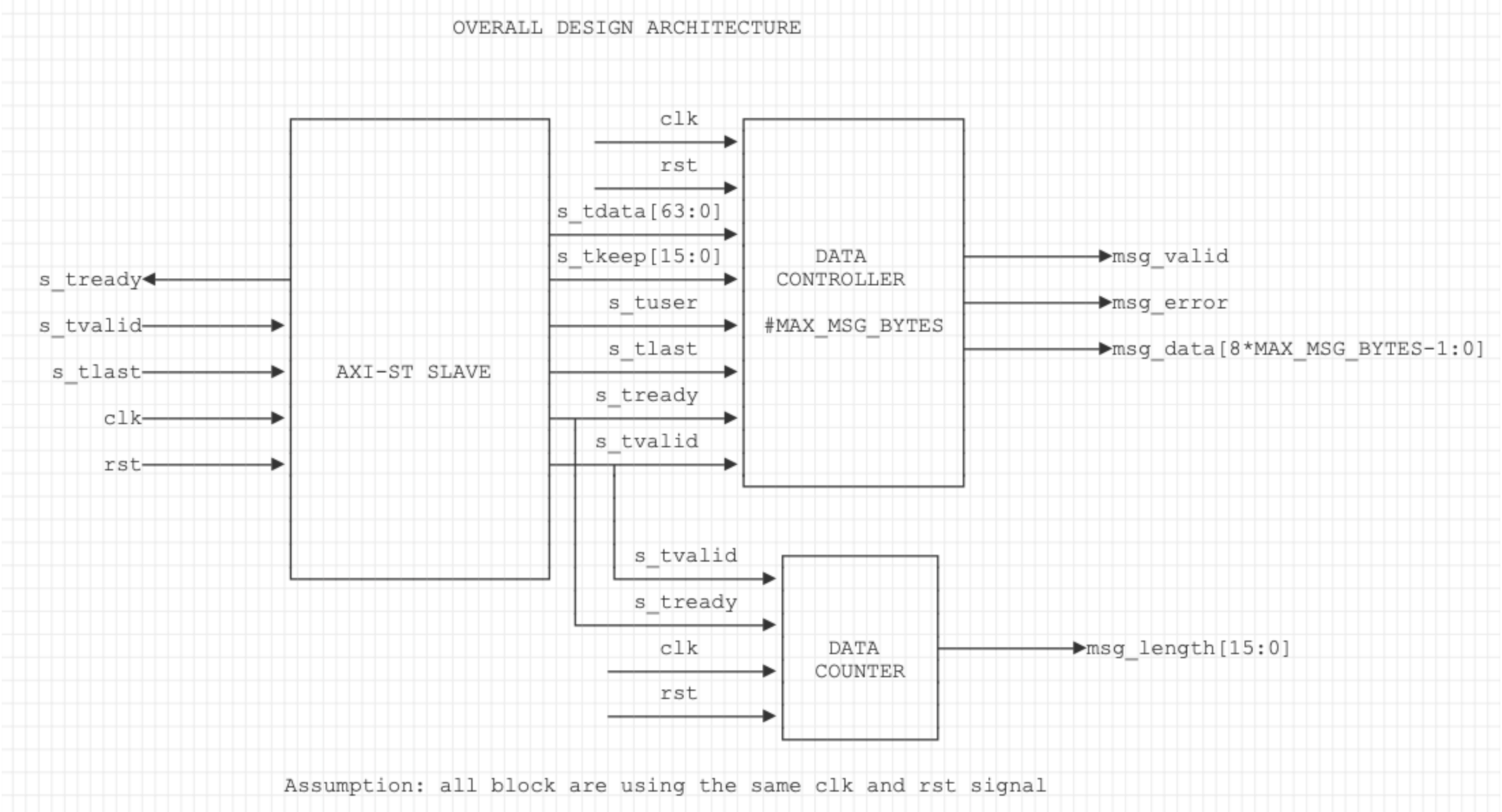
1. Draw a diagram of your chosen design (feel free to use https://asciiflow.com)

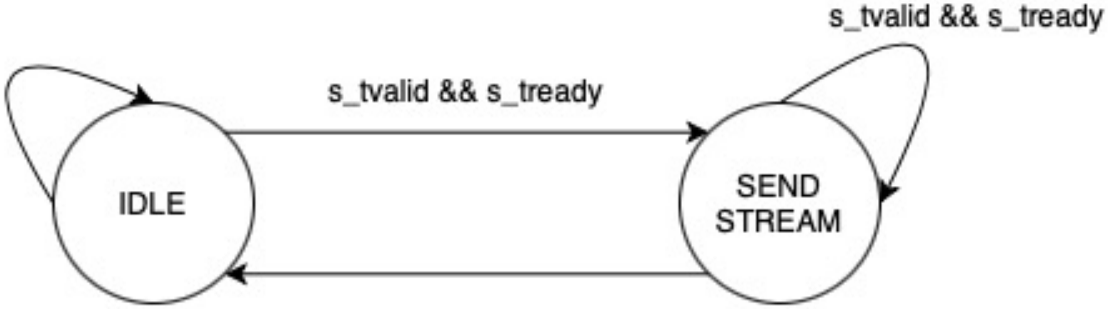
   Link: Overall top architecture & Data Buffer



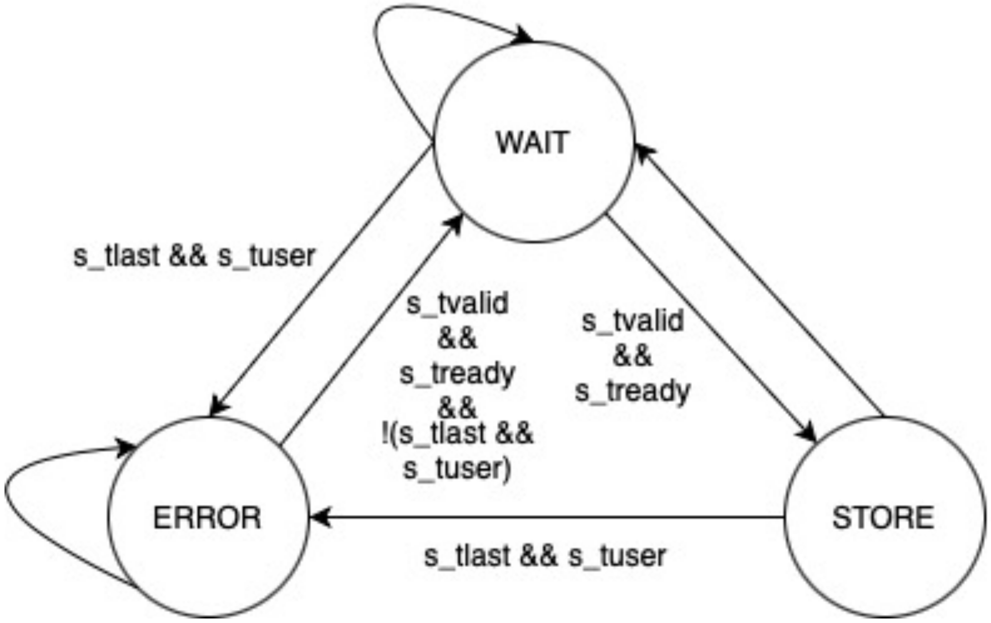
FSM Diagram

FSM Diagram

AXI Slave
FSM



Data
Controller
FSM



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.

   https://github.com/nadflop/LMS-Technical-Assesment

   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.

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

   If I were to make an educated guess, the bottleneck of my design would be the Data Controller. This is where most of the logic in determining the output for msg_data, msg_valid, msg_error are and some of these mentioned output are being used for the Data Counter.

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

      Need to add a data 'downsizing' logic at the Data Buffer.

   2. min=8B ; max=256B

      Nothing since my current design already has an 'upsizing' logic at the Data Buffer.

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

   I used a behavioral modelling in my design approach for the best behavior accuracy. Also, I separated all of my logic and registers in my design which helped me to give a good timing reference. However, this definitely added more cost in logic and eventually execution time.