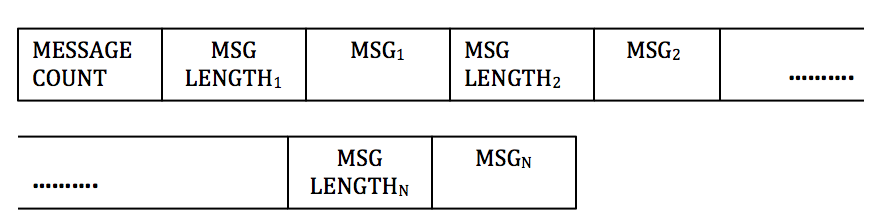
**FPGA Developer Assignment**

**Message Decoder**

Exchanges disseminate data in a protocol specific to them. A sample exchange protocol is defined below:



|  |  |  |
| --- | --- | --- |
| **Field** | **Length (bytes)** | **Description** |
| Message count | 2 | Number of message in the payload |
| Msg Length | 2 | Length of the following message |
| Msg | Variable | Message description |

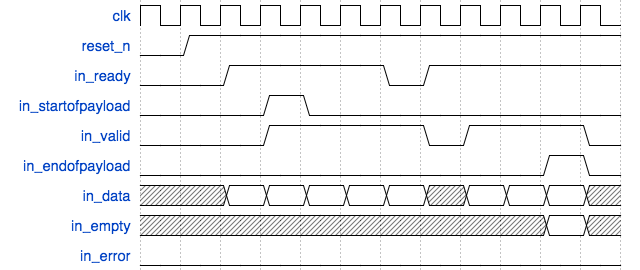
The FPGA receives data in this format from the exchange. **The objective of this assignment is to design and implement a module that extracts messages from this stream and outputs it in the interface defined.**

**Input interface**

The module to be designed receives the data stream in Avalon streaming interface defined below:

|  |  |  |  |
| --- | --- | --- | --- |
| **Signal Name** | **Width (bits)** | **Direction** | **Description** |
| clk | 1 | input | Positive edge triggered clock |
| reset\_n | 1 | input | Active low reset |
| in\_valid | 1 | input | High when incoming data is valid, low other wise |
| in\_startofpayload | 1 | input | High for 1 cycle, marks the beginning of incoming payload; should be qualified with in\_valid |
| in\_endofpayload | 1 | input | High for 1 cycle, marks the end of incoming payload; should be qualified with in\_valid |
| in\_ready | 1 | output | Asserted by the module being design to indicate that it is ready to accept data. **Read Latency=1** |
| in\_data | 64 | input | Data |
| in\_empty | 3 | input | Always qualified when in\_endofpacket is high. Indicates the number of empty bytes in the last cycle of the incoming payload |
| in\_error | 1 | input | Used to indicate an error in the incoming data stream |

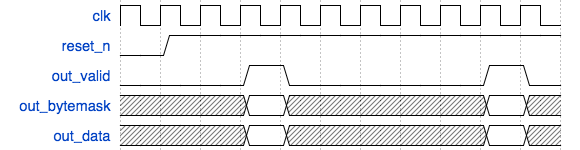
Timing diagram is given below



**Output Interface**

|  |  |  |  |
| --- | --- | --- | --- |
| **Signal Name** | **Width (bits)** | **Direction** | **Description** |
| clk | 1 | input | Positive edge triggered clock |
| reset\_n | 1 | input | Active low reset |
| out\_data | 256 | output | Extracted message |
| out\_valid | 1 | output | High when out\_data is valid; low otherwise |
| out\_bytemask | 32 | output | Used to indicate the number of valid bytes in out\_data. For example, if out\_data has 10 valid bytes, then out\_bytesmask is 32’b0000\_0000\_0000\_0000\_0000\_0011\_1111\_1111 and so on. |

The timing diagram is given below:



**Assumptions**

1. The maximum number of bytes in a single payload will not exceed 1,500 bytes.
2. The minimum size of a message is 8 bytes and the maximum size of a message is 32 bytes.
3. in\_error is always 0.

**Questions**

1. Draw the finite state machine for your design
2. Write synthesizable, elegant RTL for your module in Verilog/SystemVerilog or VHDL.
3. How would your design if the range of message changed from {8,32} bytes to:
   1. {1,32} bytes
   2. {8, 64} bytes

You don’t need to write the RTL for question 3, a brief description will suffice.

1. What is the critical path in your design?
2. What do you think is the Fmax of your design? An exact number is not required; a ballpark number with an explanation is enough.
3. What are the trade-offs of your design approach?

Please write down all the assumptions you make.

All the best!

**Sample Input**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **In\_data** | **In\_valid** | **In\_endofpayload** | **In\_startofpayload** | **In\_empty** |
| 0008000962626262 | 1 | 0 | 1 | X |
| 6262626262000b43 | 1 | 0 | 0 | X |
| 4343434343434343 | 1 | 0 | 0 | X |
| 4343000e72727272 | 1 | 0 | 0 | X |
| 7272727272727272 | 1 | 0 | 0 | X |
| 7272000856565656 | 1 | 0 | 0 | X |
| 5656565600118989 | 1 | 0 | 0 | X |
| 8989898989898989 | 1 | 0 | 0 | X |
| 8989898989898900 | 1 | 0 | 0 | X |
| 0a30303030303030 | 1 | 0 | 0 | X |
| 3030300010282828 | 1 | 0 | 0 | X |
| 2828282828282828 | 1 | 0 | 0 | X |
| 2828282828000d54 | 1 | 0 | 0 | X |
| 5454545454545454 | 1 | 0 | 0 | X |
| 5454545400000000 | 1 | 1 | 0 | 4 |

**Sample Output**

|  |  |  |
| --- | --- | --- |
| **Out\_data(hex)** | **Out\_valid** | **Out\_bytemask(binary)** |
| 626262626262626262 | 1 | 32’b0000\_0000\_0000\_0000\_0000\_0001\_1111\_1111 |
| 4343434343434343434343 | 1 | 32’b0000\_0000\_0000\_0000\_0000\_0111\_1111\_1111 |
| 7272727272727272727272727272 | 1 | 32’b0000\_0000\_0000\_0000\_0011\_1111\_1111\_1111 |
| 5656565656565656 | 1 | 32’b0000\_0000\_0000\_0000\_0000\_0000\_1111\_1111 |
| 8989898989898989898989898989898989 | 1 | 32’b0000\_0000\_0000\_0001\_1111\_1111\_1111\_1111 |
| 30303030303030303030 | 1 | 32’b0000\_0000\_0000\_0000\_0000\_0011\_1111\_1111 |
| 28282828282828282828282828282828 | 1 | 32’b0000\_0000\_0000\_0000\_1111\_1111\_1111\_1111 |
| 54545454545454545454545454 | 1 | 32’b0000\_0000\_0000\_0000\_0001\_1111\_1111\_1111 |