Mako FPGA Programming Test

Context

A stream of packetized data enters an FPGA system. The stream consists of a 64bit wide data path with valid, start of packet, and end of packet flags. These are accompanied by an 8bit wide vector indicating whether bytes are active on the 64bit data path. Data arrives synchronised to the rising edge of a clock.

The start and end of packet flags are active for one clock cycle at the beginning and end of packets respectively.

Packets can be a length of any whole number of bytes. Packets of length not a multiple of 8 bytes are indicated by the byte enable. Within a packet the byte enable is 0b11111111, except for the last 64bit word. Valid data is contained from the most significant side to the least significant. So, expected values for the byte enable are 0b10000000, 0b11000000, 0b11100000, 0b11110000 etc. The valid flag has its usual meaning. All signals are active high.

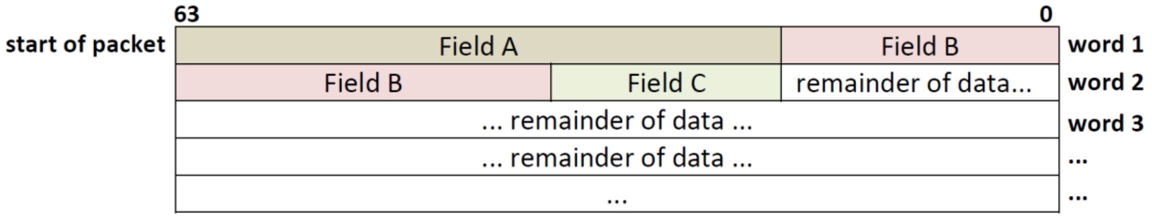
Problem

The incoming stream is to be parsed to remove data fields from a header present at the beginning of each packet. Every packet contains the header. The data is then to be realigned such that only the last word in each packet can have a byte enable not equal to 0b11111111. The other flags should be set correctly on the output.

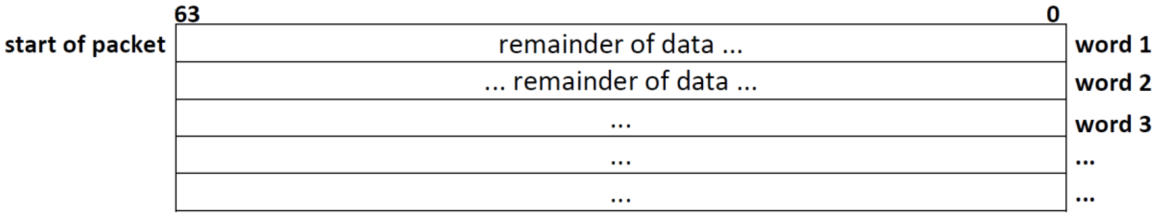
Header format

Field A - 48bits wide Field B - 48bits wide Field C - 16bits wide

Input data



Output data



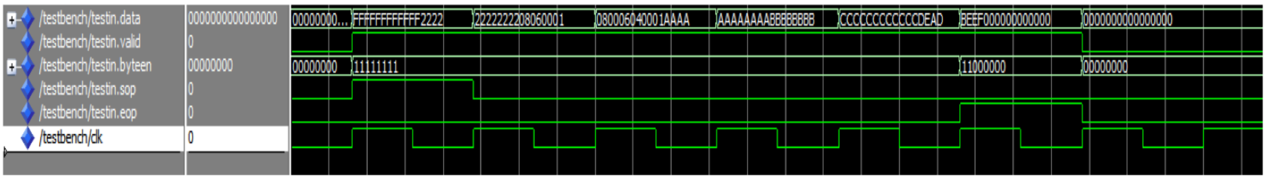
Deliverables

Preferably in VHDL or Verilog, deliver source for a synthesisable module performing the parsing described with any type definitions you create and an associated testbench and verification model. Input to the module should be the stream described in the context. Output of the module should be the same format as the input, with realigned data and a delay of any number of clock cycles, plus additional signals containing the parsed fields. The module should be capable of operating at 250MHz on a modern FPGA device. The module should be designed for low input to output latency.

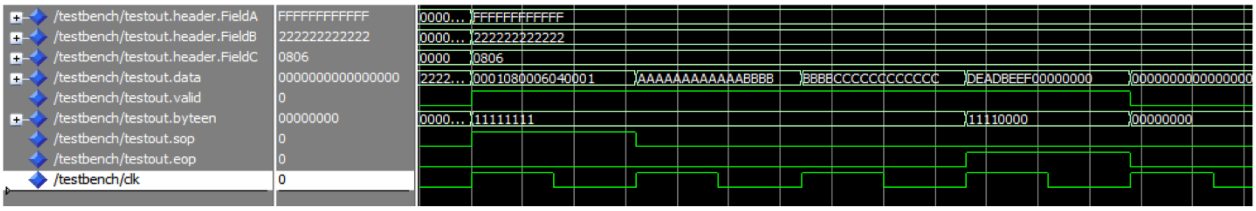
Example

The following waveforms are provided for a single example test packet. Your solution should operate with any number of input packets of any size.

Input



Output


/testbench/testout.header.FieldA
FFFFFFFFFFFF
0000...
FFFFFFFFFFFF
/testbench/testout.header.FieldB
222222222222
0000...
222222222222
/testbench/testout.header.FieldC
0806
0000
0806
/testbench/testout.data
0000000000000000
2222...
0001080006040001
AAAAAAAAAAAABBBB
BBBBCCCCCCCCCCCC
DEADBEEF00000000
0000000000000000
/testbench/testout.valid
0
/testbench/testout.byteen
00000000
0000...
11111111
11110000
00000000
/testbench/testout.sop
0
/testbench/testout.eop
0
/testbench/clk
0