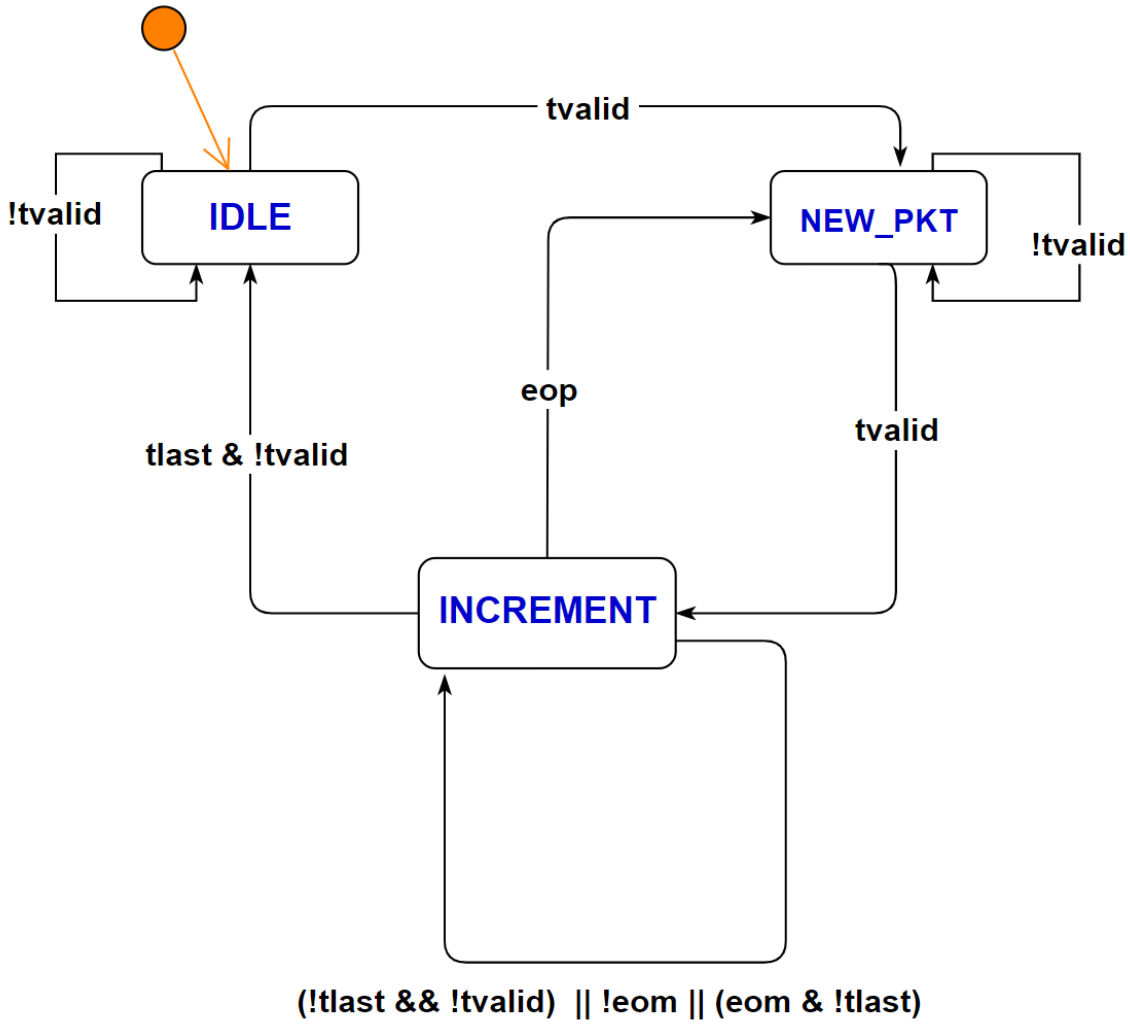
Message extractor assignment user guide

1. Overview
    - This message parser design is based on 3 states FSM:
        - IDLE: FSM goes to IDLE state if the reset is high, or the frame decoding is done
        - NEW_PKT: new packet state denotes the beginning of new list of messages
        - INCREMENT: The FSM goes to this state during the decoding process of messages of a given packet.
    - The state machine can be represented in more elegant way (by introducing a new state "NEW_MSG" for example). However, low latency was considered here.
      eom: end of message
      eop: end of packet

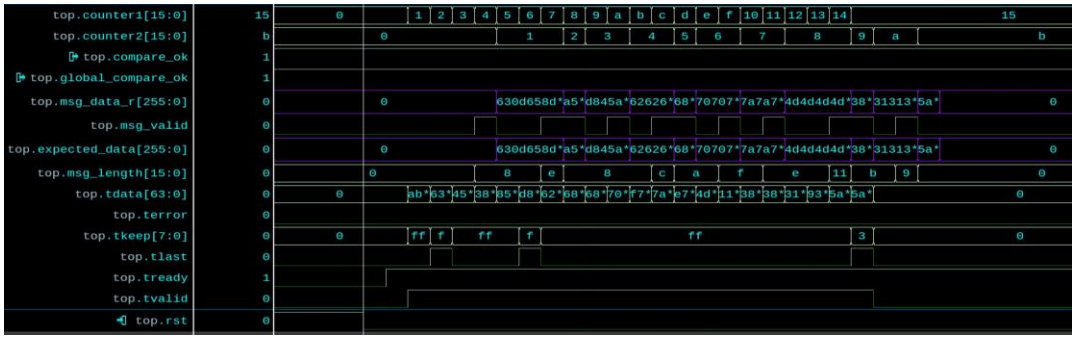


2. How to run simulation
   Modelsim/Questasim is needed to run the simulation
    - Go under <tb> directory
    - Run: make
    - Once the simulation is finished, you should see at the end the following:

```
# ============================
# Starting the 1st test
# ============================
# counter2 =     0  ;  compare_ok = 1  ;  global_compare_ok = 1
# counter2 =     1  ;  compare_ok = 1  ;  global_compare_ok = 1
# counter2 =     2  ;  compare_ok = 1  ;  global_compare_ok = 1
# counter2 =     3  ;  compare_ok = 1  ;  global_compare_ok = 1
# counter2 =     4  ;  compare_ok = 1  ;  global_compare_ok = 1
# counter2 =     5  ;  compare_ok = 1  ;  global_compare_ok = 1
# counter2 =     6  ;  compare_ok = 1  ;  global_compare_ok = 1
# counter2 =     7  ;  compare_ok = 1  ;  global_compare_ok = 1
# counter2 =     8  ;  compare_ok = 1  ;  global_compare_ok = 1
# counter2 =     9  ;  compare_ok = 1  ;  global_compare_ok = 1
# ============================
# Starting the 2nd test
# ============================
# counter2 =    10  ;  compare_ok = 1  ;  global_compare_ok = 1
# ** Warning: (vsim-PLI-8496) $fatal : Argument number 1 is invalid. Expecting 0, 1, or 2. Using default value of 1
#    Time: 2325 ns  Iteration: 1  Process: /top/msg_parser_inst/#ALWAYS#67 File: ../design/msg_parser.sv Line: 143
# ** Fatal: Message length is out of range: 8 <= msg_length <= 32. Received message length =     7
#    Time: 2325 ns  Scope: top.msg_parser_inst File: ../design/msg_parser.sv Line: 143
# ** Note: $finish    : ../design/msg_parser.sv(143)
#    Time: 2325 ns  Iteration: 1  Instance: /top/msg_parser_inst
# qwavedb_dumpvars : Simulation ending at [0 2325000] 0
# End time: 13:05:29 on Sep 18,2022, Elapsed time: 0:00:00
# Errors: 1, Warnings: 2
```

- **compare_ok** is comparing the decoded data against the expected data for each message
- **global_compare_ok** goes down if a single comparison fails
- The 1st test is the one provided along the assignment
- In the 2nd test, the message length is 7 which is out of range (message_length between 8B and 32B)

3. How to run FPGA compilation:
   - Go under hardware:
   - There, you should have a Quartus project to run full compilation
   - Pin assignment is left as it depends on the target device.
   - This design was tested on a Stratix10 1SG10MHN3F74C2LG_U1 FPGA device.
   - Quartus version used: 22.1
   - Design is timing clean at ~172 MHz.
   - Maximum number of logic levels is 9 (There are rooms for optimization to achieve higher frequency)
   - Waveform:



➢ The full waveform can viewed using gtkwave:
   - Go under hardware/wavefom
   - Run: sh view_waveform.sh

Questions

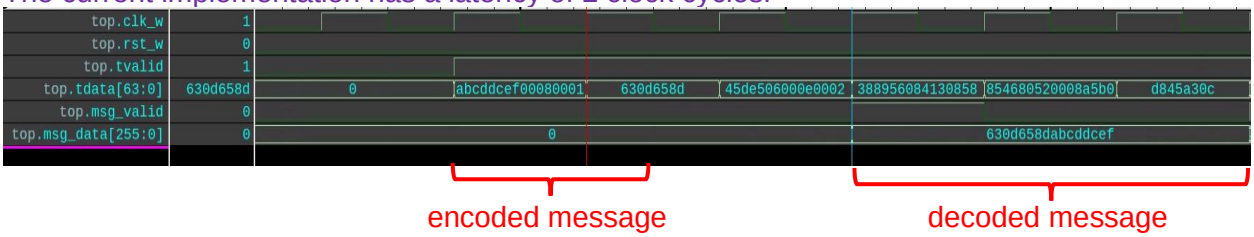
What is the bottleneck of your design/code? (what can limit the maximum frequency?)

The design was only timing clean at ~172 MHz. The max number of logic level is 9 LL, which are within the data decoding path:

```
Path #1: Setup slack is -0.780 (VIOLATED)
================================================================================
+-----------------------------------------------------------------+
; Path Summary                                                     ;
+---------------------------------+-------------------------------+
; Property                        ; Value                         ;
+---------------------------------+-------------------------------+
; From Node                       ; msg_parser_inst|data_counter[0] ;
; To Node                         ; msg_parser_inst|msg_data_r[40]  ;
; Launch Clock                    ; clk                           ;
; Latch Clock                     ; clk                           ;
; SDC Exception                   ; No SDC Exception on Path       ;
; Data Arrival Time               ; 8.327                         ;
; Data Required Time              ; 7.547                         ;
; Slack                           ; -0.780 (VIOLATED)             ;
; Worst-Case Operating Conditions ; Slow 850mV 100C Model          ;
+---------------------------------+-------------------------------+
```

One solution is to add a pipeline to reduce the number of LL which consequently should help increase the frequency. However, this would add another clock cycle of latency.

The current implementation has a latency of 2 clock cycles:



➔ latency = 2 * 5.8 = 11.6 ns

By adding 1 pipeline, latency become 3 cycles. Which means that the clock period needs to be below 11.6 / 3 = 3.87 ➔ the design needs to be timing clean at a frequency > 258 Mhz.

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

The State machine must be changed to consider the new constraint (min =1B). the current implementation (min = 8B) assumes that a given message is distributed on at least 2 transfers (msg_length + msg_data >=10B)

b. min=8B ; max=256B

No change at all. The current implementation can handle any max message length.

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

Max = 256B means that the design supports higher number of various messages lengthsOn the other hand, from implementation perspective, this introduces more complexity to meet the timing at higher frequency. As shown in the screenshot below, < min=8B ; max=256B> scenario is presenting higher number of fanout signals, with an increasing max fanout (4428 vs 538) compared to the <

min=8B ; max=32B> scenario?

```
+-----------------------------------------------------------------+
; Non-Global High Fan-Out Signals                                  ;
+----------------------------------------+---------+--------------+
; Name                                   ; Fan-Out ; Physical Fan-Out ;
+----------------------------------------+---------+--------------+
; rst~input                              ; 538     ; 125             ;
; msg_parser_inst|remaining_valid_bytes[0] ; 537     ; 537             ;
+----------------------------------------+---------+--------------+
```

min=8B ; max=32B

```
5593 +-----------------------------------------------------------------
5594 ; Non-Global High Fan-Out Signals
5595 +----------------------------------------+---------+--------------
5596 ; Name                                   ; Fan-Out ; Physical Fan-Ou
5597 +----------------------------------------+---------+--------------
5598 ; rst~input                              ; 4428    ; 701
5599 ; msg_parser_inst|state.INCREMENT        ; 3640    ; 3640
5600 ; msg_parser_inst|remaining_valid_bytes[0] ; 2090    ; 2090
5601 ; msg_parser_inst|i140606~0             ; 1995    ; 1995
5602 ; msg_parser_inst|i149082~0             ; 1943    ; 384
5603 ; msg_parser_inst|state.NEW_PKT         ; 1895    ; 1895
5604 ; msg_parser_inst|i140606~1             ; 1869    ; 1869
5605 ; msg_parser_inst|Select_5538~0         ; 1731    ; 1731
5606 +----------------------------------------+---------+--------------
5607
5608
```

min=8B ; max=256B