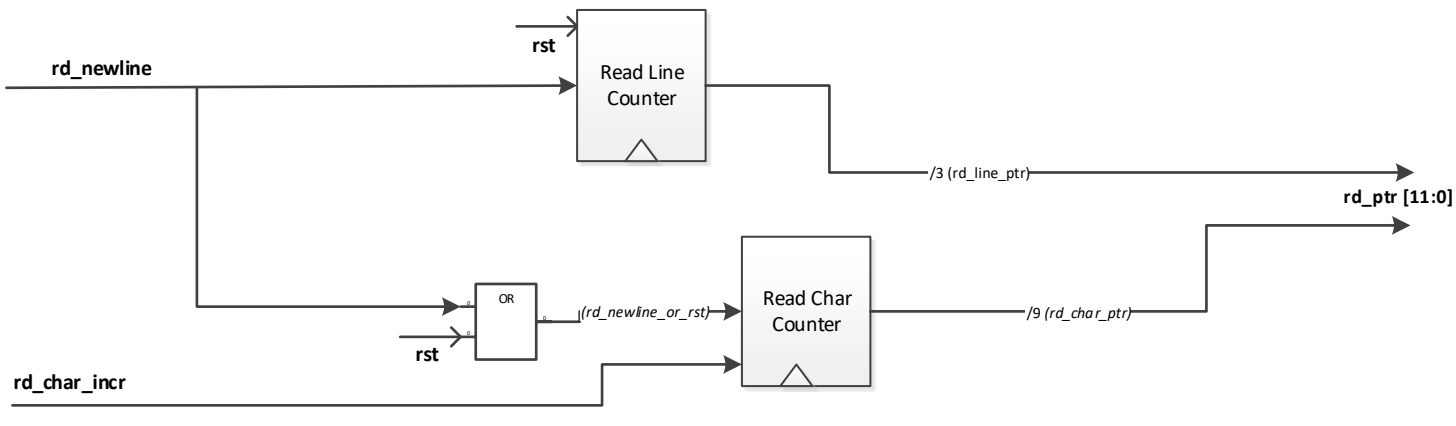
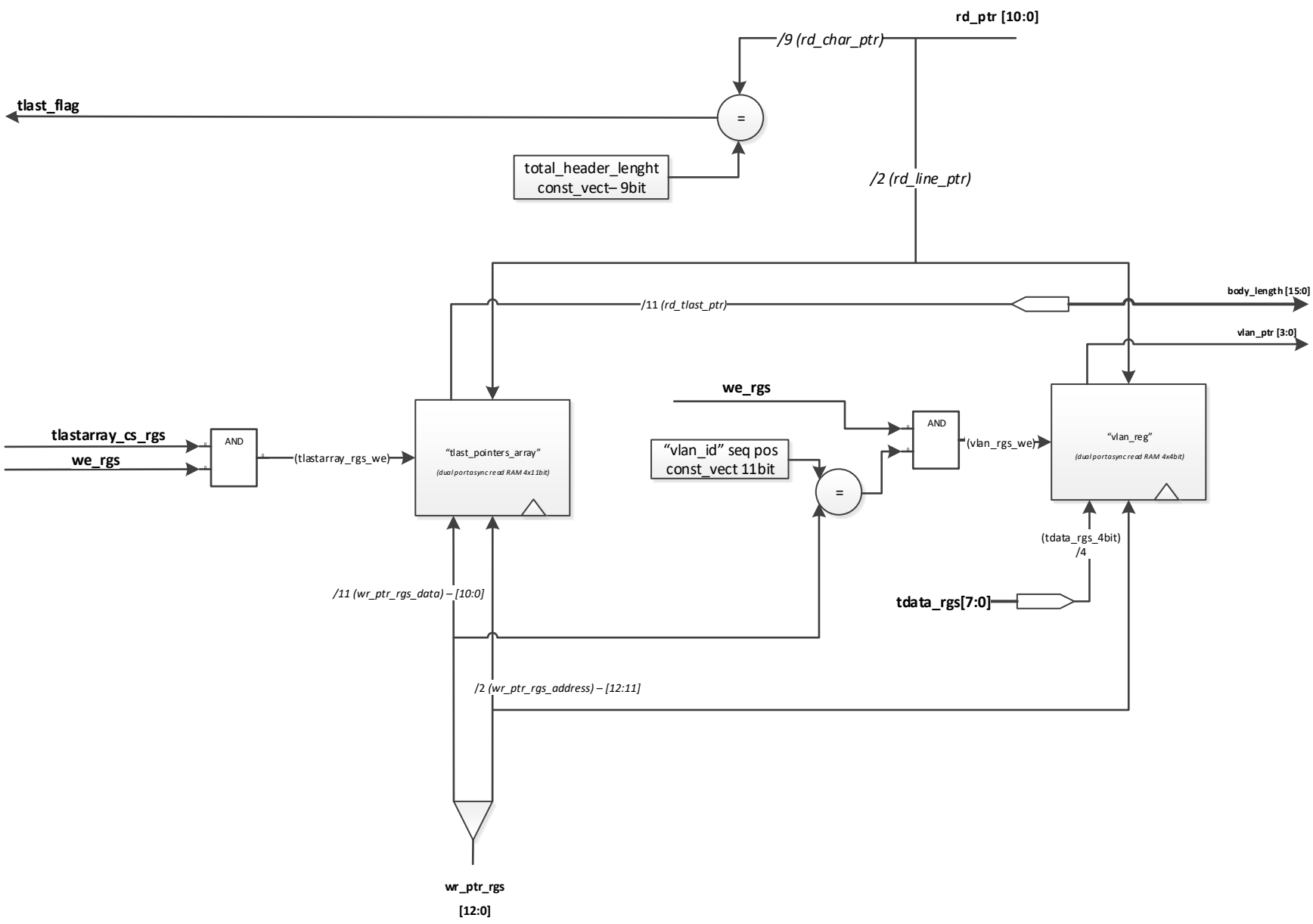
“READ_LOGIC_COUNTERS”
module



“READ_LOGIC_REGS_HEADER”
module



“READ_LOGIC_HEADER”
module

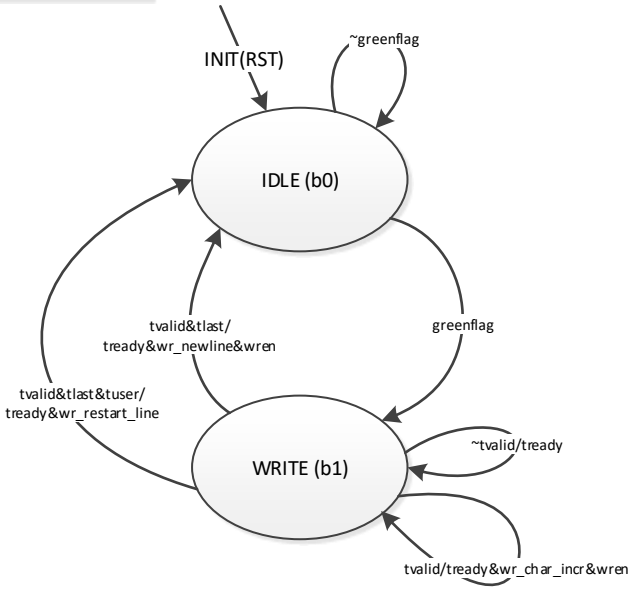
REF ONLY
(as in EHA_v1.1)

Write and Read Finite state machines

INPUTs	OUTPUTs
greenflag tvalid tlast tuser	tready wren wr_newline wr_char_incr wr_restart_line

WRITE TO BUFFER FSM

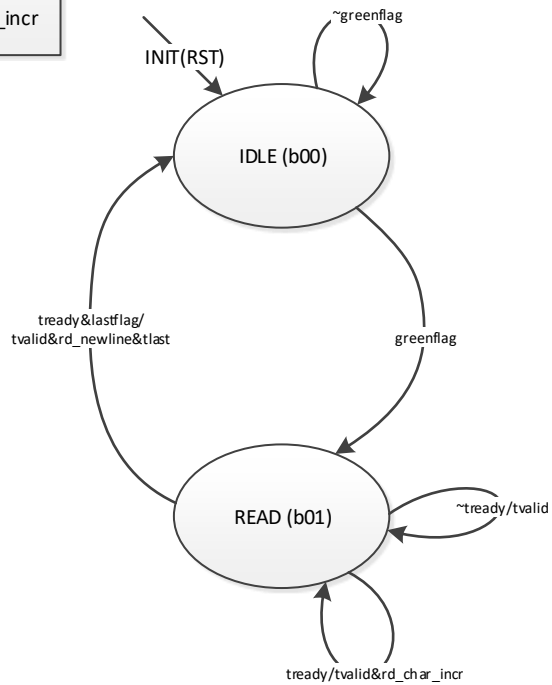
(writebuf_fsm)



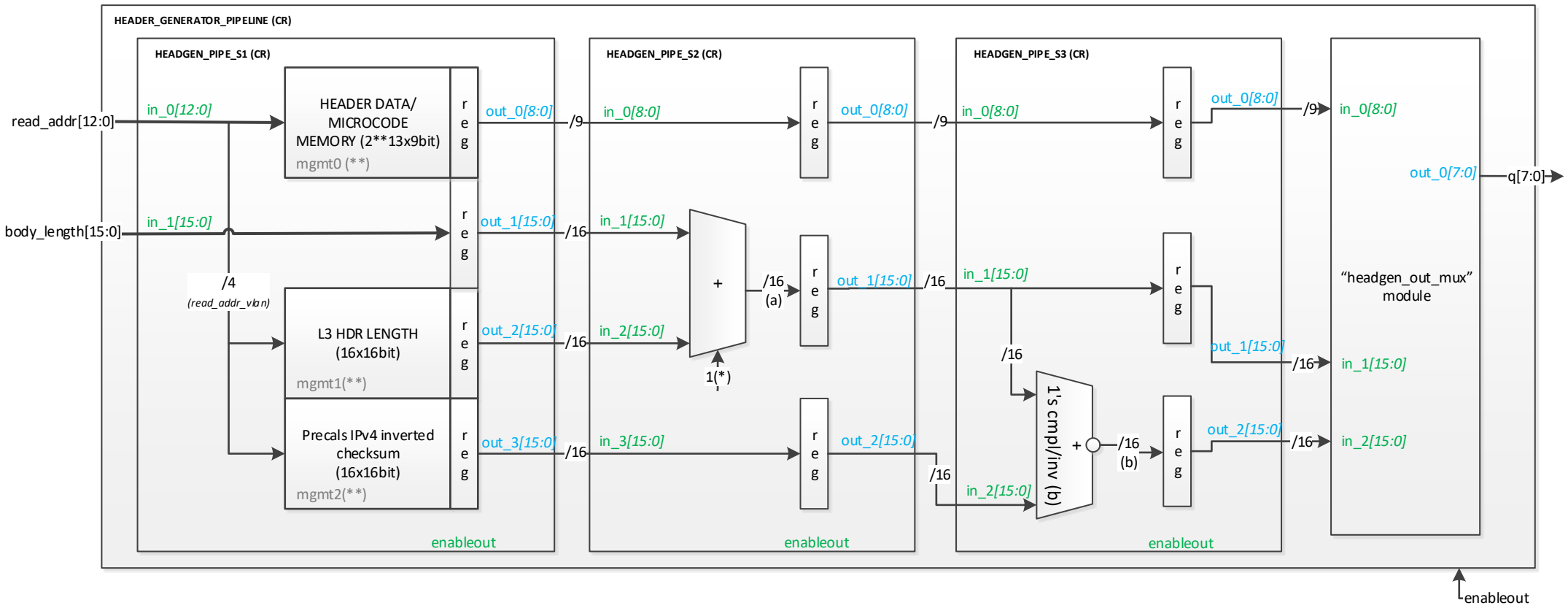
INPUTs	OUTPUTs
greenflag lastflag tready	tvalid tlast rd_newline rd_char_incr

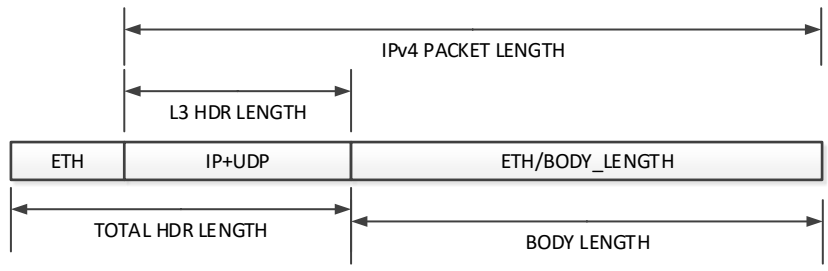
READ FROM BUFFER FSM

(readbuf_fsm)

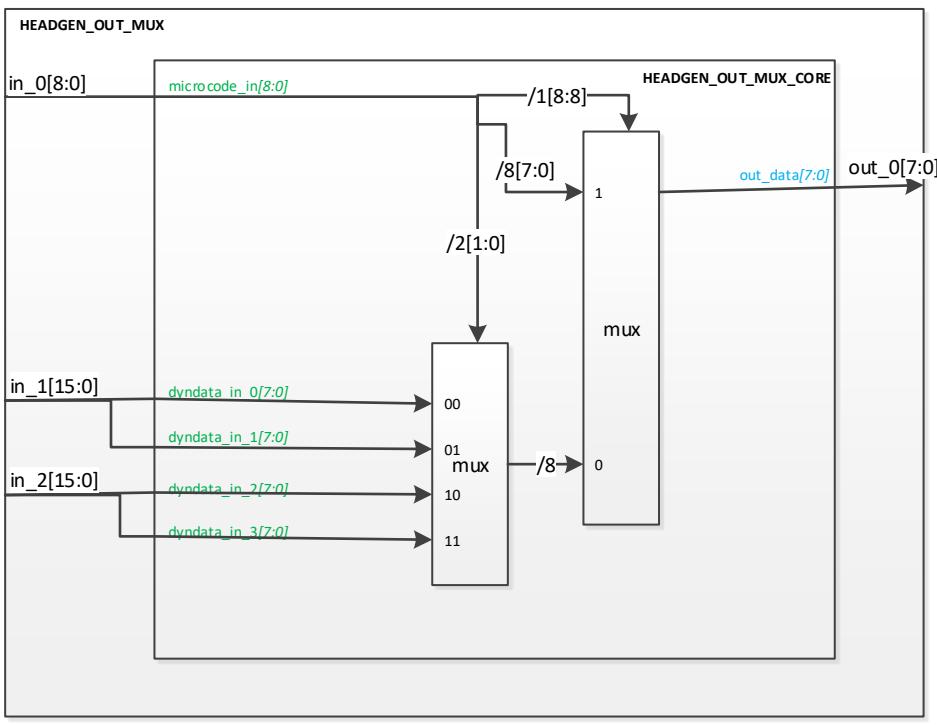


“HEADER_GENERATOR_PIPELINE” module





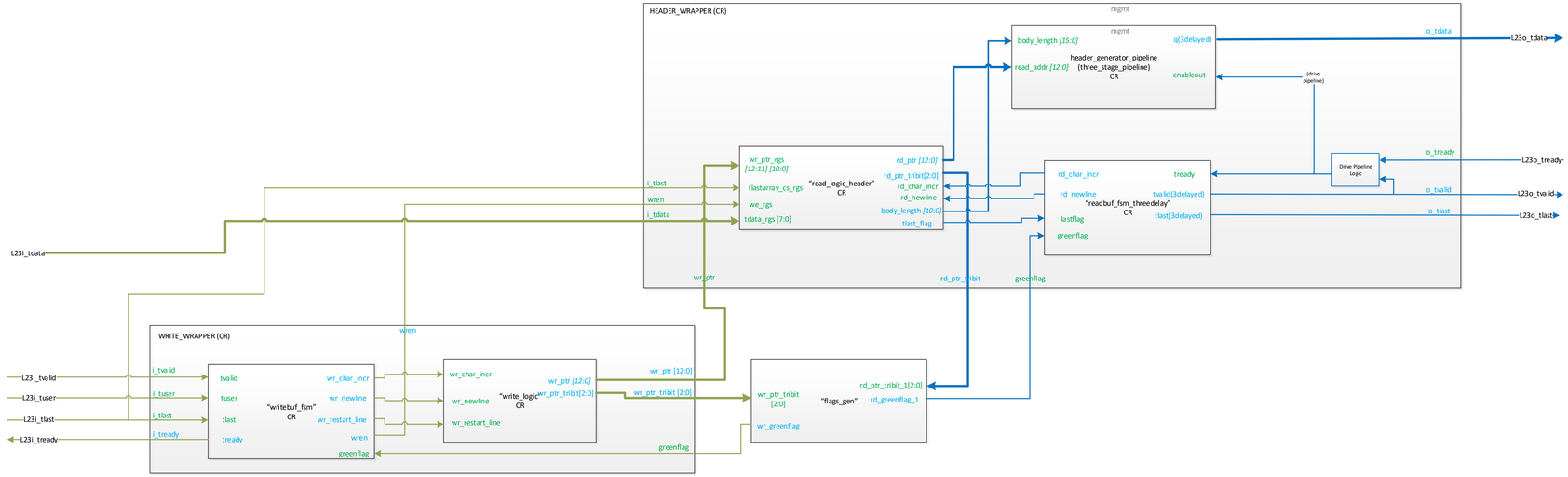
a) Calculation of “IPv4 packet length/IPv6 payload length” as a sum of “body_length” and “L3_HDR_LENGTH” which in case of IPv6 is 0.
b) Calculation of “IPv4 checksum” as 1's complement addition of “precalculated IPv4 inverted checksum” and “IPv4 packet length”. To produce an actual checksum value a result value of this 1s complement addition must be inverted.
*) See fragmentation support proposals for next versions TABLE_A. In the current version without fragmentation this carry_in signal is always 1.
**) Mgmt is a write interface to the RAM memory block, required to program the content of the memory and it is presented externally (on the “HEADER_GENERATOR_PIPELINE” level) so it can be accessible from the processor system (control plane).



HEADER STATIC/DYNAMIC DATA GENERATION:
ADDRESS: “13'bvvvv_dddddddd” as “13'b(VLAN_NUM)_(OFFSET)”
MICROCODES DATA:
9'bc_dddddddd, if c=1 then the following bytes (dddddddd) are microcode instruction, otherwise they are data itsef.
00 – IPv4 packet length/IPv6 payload length MAJOR-BYTE
01 – IPv4 packet length/IPv6 payload length MINOR-BYTE
10 – IPv4 checksum MAJOR-BYTE
11 – IPv4 checksum MINOR-BYTE

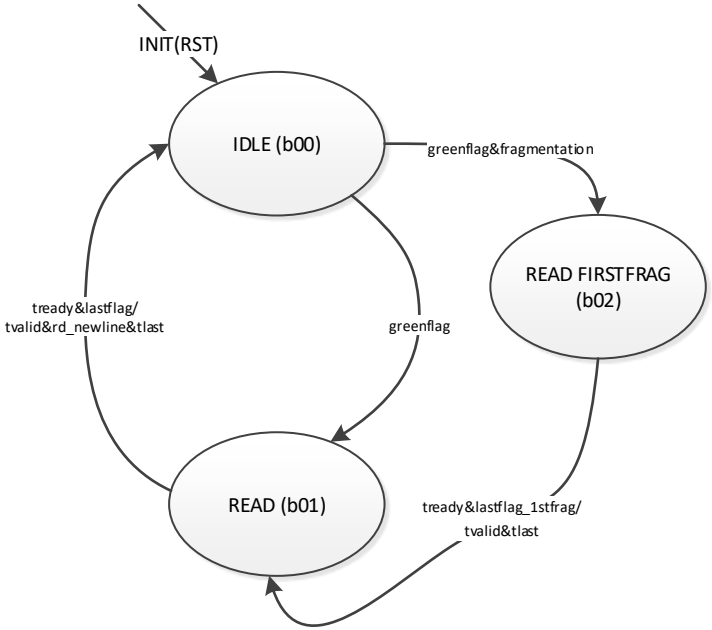
Example: microcode_data of 9'1_00000010 stored in 13'b0011_000000011 ensures that 4th byte of L3 header of vlan3 contains “IPv4 checksum MAJOR-BYTE”.

“L23_buffer” module



FRAGMENTATION SUPPORT PROPOSAL FOR NEXT VERSIONS

FRAGMENTATION (happens is “BODY_LENGTH”>1024):
STORAGE_WRAPPER: CHAR_counter increases until BODY_LENGTH/2 value producing first fragment.
Then till BODY_LENGTH producing second fragment.
HEADER_WRAPPER: Calculate “Packet_length” values for both first and second fragments simply having “BODY_LENGHT/2” and adjusting it plus/minus 1 byte using TABLE_A.
READFSM for body generator will look as follows:



TABLE_A (IPv4 packet length adjustment)

(*) – calculation of “NORMAL LENGTH” by Adding “ADDbit” to “BODY LENGTH” or “HALF_BODY_LENGHT” depending on fragmentation parameters			
Fragmented	LastFrag	LSBofBodyLength	ADD Bit
0	0	0	1 (this_ver)
0	0	1	1 (this_ver)
0	1	0	Illegal comb
0	1	1	Illegal comb
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	1

IPv4 Checksum calculation (INCREMENTAL UPDATE):
Checksum is calculated as 1's complement addition of all 16bit fields in the header with the inversion (1's complement) of derived value.
The precalculated “incomplete” checksum value stored in “PRECALC IPv4 INVERTED CHSUM” memory is the value calculated with “16bit(length) value” not taken into account and without final inversion.
The hardware takes precalculated checksum value and completes calculation of final checksum using this formula (1's complement addition and inversion):
HC’=~(C+length)
where: HC’ - final checksum value going to Ipv4 header;
C – “PRECALC IPv4 INVERTED CHSUM” - uncomplited checksum (without “length” value);
length – 16bit IPv4 packet length value.