State	Condition for transition	Next state
IDLE	reset is active or FIFO is empty: Rstb = 0 || fifo_empty =1 Reset is inavtive and FIFO not empty : Rstb = 1 && fifo_empty = 0	 IDLE SOP_S0
SOP_S0	FIFO is empty : fifo_empty =1 FIFO is not empty and Start of Packet is High: fifo_empty = 0 && sop=1 && … - Current message length < 6 - Current message length > 8 - Current message length == 7 - Current message length == 6	SOP_S0 S1 S2 S4 S6
S1	FIFO is empty : fifo_empty =1 always	S1 S1_P
S1_P	FIFO is empty : fifo_empty =1 FIFO is not empty: fifo_empty =0 && … - Current message length < 6 - Current message length > 8 - Current message length == 7 - Current message length == 6	S1 S1 S2 S4 S6
S2	FIFO is empty : fifo_empty =1 FIFO is not empty: fifo_empty =0 && … - Current message length < 6 - Current message length > 8 - Current message length == 7 - Current message length == 0 - Current message length == 6	S2 S1 S2 S4 S5 S6
S3	FIFO is empty : fifo_empty =1 FIFO is not empty: fifo_empty =0 && … - Current message length < 6 - Current message length > 8 - Current message length == 7 - Current message length == 6	S3 S1 S2 S4 S6
S4	FIFO is empty : fifo_empty =1 FIFO is not empty: fifo_empty =0	S4 S3
S5	FIFO is empty : fifo_empty =1 FIFO is not empty: fifo_empty =0 && … - Current message length < 6 - Current message length > 8 - Current message length == 7 - Current message length == 6	S5 S1 S2 S4 S6
S6	FIFO is empty : fifo_empty =1 FIFO is not empty: fifo_empty =0 current message length = current message length – length of data in current input	S6 S2

Outputs	**Description**
upd_mc	update message count
upd_ml[1:0]	update message length, upd_ml[1] → message_len[1], upd_ml[0] → message_len[0]
upd_len	update current message length Current message length -= length of data payload in current input
mc_st, mc_end	start and end index of message count field, updated only once
ml_st, ml_end	start and end index of message length field, updated for every new message
p_st, p_end	start and end index of data payload field, updated every clock
cur_len	length of data payload in current input
read_en	FIFO read enable

Outputs

State	Outputs	Description
	mc_st = 7 mc_end = 6	message count field always present in data_bytes[7:6], this field is only valid when upd_mc =1
IDLE	upd_mc = 0 upd_ml[1:0] = 0 ml_st = 0 ml_end = 0 upd_len = 0 p_st = 0 p_end = 0 cur_len = 0 read_en = 0 msg_end = 0	Clear all outputs
S0	If (sop == 1): upd_mc = 1 upd_ml[1:0] = 2'b11 ml_st = 5 ml_end = 4 upd_len = 1 p_st = 3 p_end = 0 cur_len = 4 read_en = 1 msg_end = 0	Set outputs as below if sop is HIGH, else clear all Update message count update message length, both bytes For the very first data input, message length field always present in data_bytes[5:4] update message length as data payload will be available For the very first data input, message data field always present in data_bytes[3:0] For the very first data input, current data payload length is 4 since FIFO is not empty, start reading from it
S1	upd_mc = 0 upd_ml[1:0] = 2'b11 ml_st = 8 – cur_msg_len - 1 ml_end = 8 – cur_msg_len -2 upd_len = 1 p_st = 7 p_end = 8 – cur_msg_len cur_len = 8 - (cur_msg_len +2) read_en = 1 msg_end = 1	No message count present in the middle of the packer update message length as data payload will be available Update new message length field indices based on where the current message data payload ends Update current message data payload Current data payload starts from index 7 Next message data payload length Read next data input from FIFO end of current message
S1_P	upd_mc = 0 upd_ml[1:0] = 2'b00 ml_st = ml_st ml_end = ml_end upd_len = 1 p_st = ml_end – 1 p_end = 0 cur_len = cur_len read_en = 1 msg_end = 0	Current message was previous message when in S1 state, message length field was already extracted in S1 state Do not change ml_st and ml_end value to save power Current data payload is available, so update current message length Data payload always starts one byte immediately after message length Current message length was previously updated in S1 state
S2	upd_mc = 0 upd_ml[1:0] = 2'b00 ml_st = ml_st ml_end = ml_end upd_len = 1 p_st = 7 p_end = 0 cur_len = 8 read_en = 1 msg_end = (cur_msg_len == 0)	This is an all data payload state If not more data payload , indicate end of message
S3	upd_mc = 0 upd_ml[1:0] = 2'b01 ml_st = ml_st ml_end = 7 upd_len = 1 p_st = 6 p_end = 0 cur_len = 7 read_en = 1 msg_end = 0	Partial message length state, only MessageLength[1] is available MessageLength[1] is present in data_bytes[7], MessageLength[0] will have been extracted in state S4 in previous clock
S4	upd_mc = 0 upd_ml[1:0] = 2'b10 ml_st = 0 ml_end = ml_end upd_len = 1 p_st = 7 p_end = 1 cur_len = 7 read_en = 1 msg_end = 1	Next message length field is present in data_bytes[0], only MessageLength[1] is available in this data input, MessageLength[0] will be available in next clock Current data payload is available, so update current message length end of current message
S5	upd_mc = 0 upd_ml[1:0] = 2'b11 ml_st = 7 ml_end = 6 upd_len = 1 p_st = 5 p_end = 0 cur_len = 6 read_en = 1 msg_end = 0	Start of new message length and data payload Next message length field is present in data_bytes[7:6]
S6	upd_mc = 0 upd_ml[1:0] = 2'b11 ml_st = 1 ml_end = 0 upd_len = 1 p_st = 7 p_end = 2 cur_len = 6 read_en = 1 msg_end = 1	Next message length field is present in data_bytes[7:6] end of current message

MC0	message count [0]
MC1	message count [1]
ML0	message length [0]
ML1	message length [1]
P	data payload

D7	D6	D5	D4	D3	D2	D1	D0
MC0	MC1	ML0	ML1	P	P	P	P
mc_st	mc_end	ml_st	ml_end	p_st			p_end

D7	D6	D5	D4	D3	D2	D1	D0
P	P/ML	P/ML	P/ML	P/ML	P/ML	P/ML	P
p_st	p_end						

FSM will only process P for current message

D7	D6	D5	D4	D3	D2	D1	D0
P	P/ML	P/ML	P/ML	P/ML	P/ML	P/ML	P
		ml_st	ml_end	p_st			p_end

Same as S1, FSM will process ML and P of next message

D7	D6	D5	D4	D3	D2	D1	D0
P	P	P	P	P	P	P	P
p_st							p_end

D7	D6	D5	D4	D3	D2	D1	D0
ML1	P	P	P	P	P	P	P
ml_end	p_st						p_end

D7	D6	D5	D4	D3	D2	D1	D0
P	P	P	P	P	P	P	ML0
p_st						p_end	ml_st

D7	D6	D5	D4	D3	D2	D1	D0
ML0	ML1	P	P	P	P	P	P
ml_st	ml_end	p_st					p_end

D7	D6	D5	D4	D3	D2	D1	D0
P	P	P	P	P	P	ML0	ML1
p_st					p_end	ml_st	ml_end

Max len of packet	1000	
Len of one data	8	
Burst len	125	
No of payloads in packet	99.8	98x(8+2) + (16+2)
		98 messages with 8B payload
		1 message with 16B payload

B0	*B1*	*B2*	*B3*	*B4*	*B5*	*B6*	*B7*	*no of clocks*
MC	MC	ML	ML	P	P	P	P	*1*
P	P	P	P	ML	ML	P	P	*2*
P	P	P	P	P	P	ML	ML	*1*
P	P	P	P	P	P	P	P	*1*
ML	ML	P	P	P	P	P	P	*1*
P	P	ML	ML	P	P	P	P	*2*
P	P	P	P	ML	ML	P	P	*2*
P	P	P	P	P	P	ML	ML	*1*
P	P	P	P	P	P	P	P	*1*
ML	ML	P	P	P	P	P	P	*1*
P	P	ML	ML	P	P	P	P	*2*
P	P	P	P	ML	ML	P	P	*2*
P	P	P	P	P	P	ML	ML	*1*
P	P	P	P	P	P	P	P	*1*
P	P	P	P	P	P	P	P	*1*

Each colored block contains 4 messages and takes 7 clocks to extract.
First 96 messages take 7/4 clocks each and the last 3 messages take 5 clocks.
The total number of clocks to extract all 99 messages is
96 x (7/4) + 5 = 173
FIFO depth = write burst – read burst
173-125 = 48