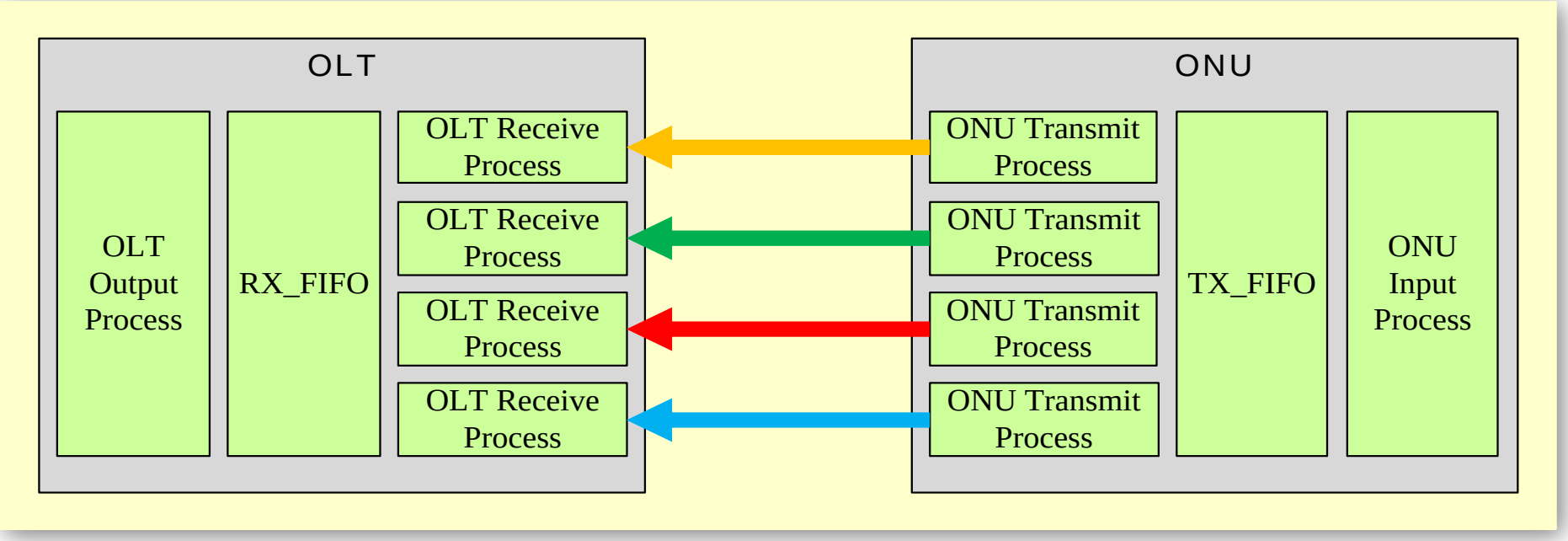
Multi-Point Reconciliation Sublayer (MPRS)
[OLT receive path]

Glen Kramer, Broadcom
Duane Remein, Huawei
Jean-Christophe Marion, Broadcom

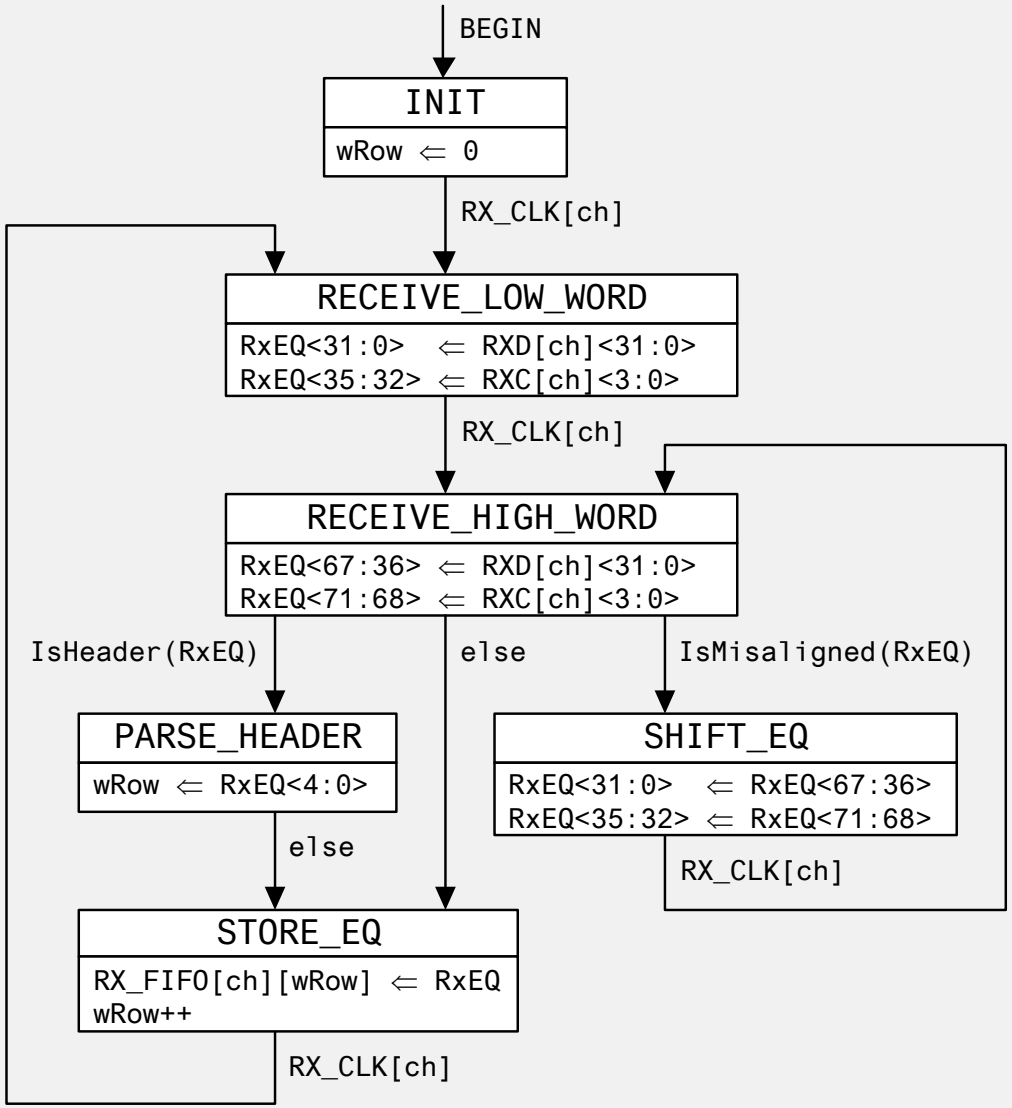
Scope of this presentation

- This presentation proposes the definition of OLT MPRS receive path operation, consisting of the following:
  1. ***OLT MPRS Receive Process*** (one instance per channel)
  2. 2D alignment buffer ***RX_FIFO***
  3. ***OLT MPRS Output Process*** (one instance per OLT)



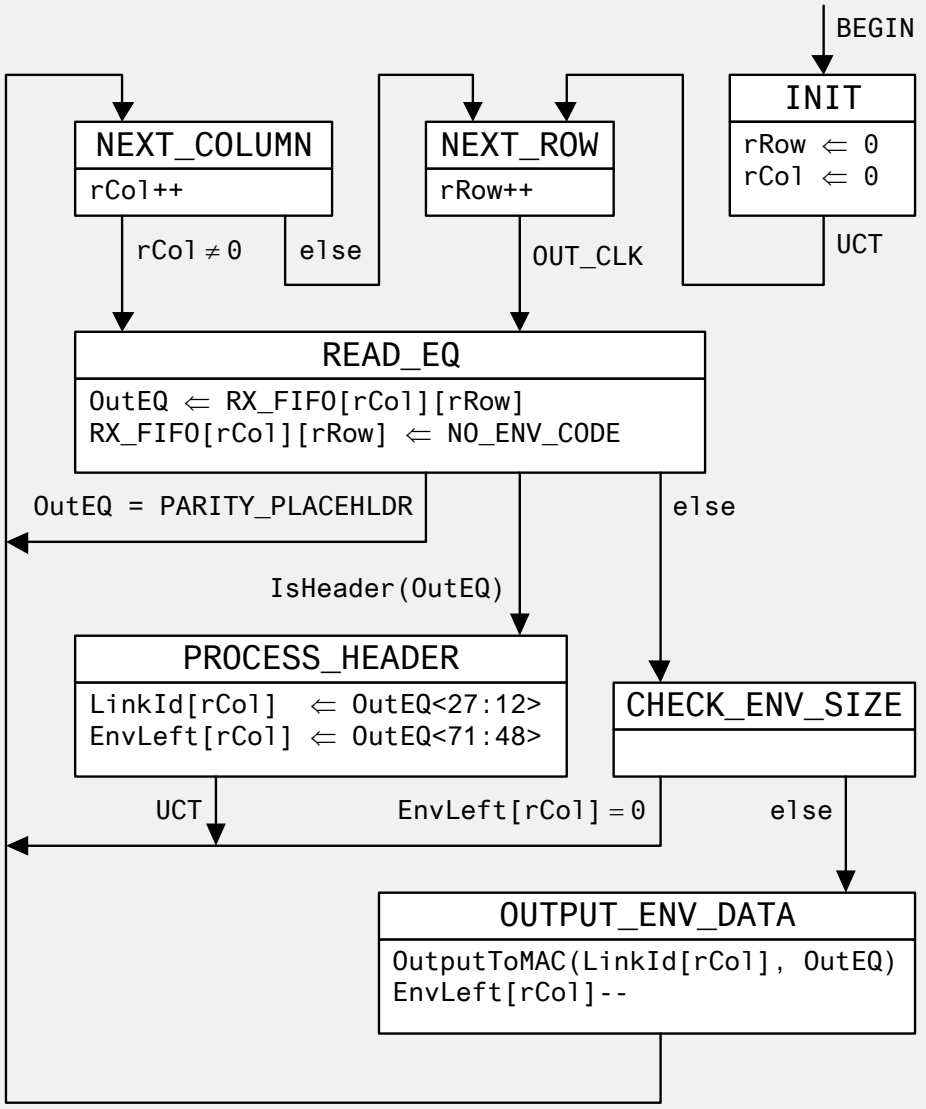
OLT MPRS Receive Process

- Forms an EQ from two 25GMII transfers
- Verifies proper alignment of EQ. If misaligned, shift the input by half-EQ
  - No other error checking is performed by this process.
- When a header is received, EPAM field is extracted and used as a write position for the RX_FIFO buffer.
- A separate instance of SD per channel is required because the phase of the receive clock (***RX_CLK[ch]***) in every channel is different
  - Different ONUs
  - Skew



OLT MPRS Output Process

- Outputs EQs to proper vMAC
  - In case of overlapping envelopes from the same LLID, data from multiple channels is properly serialized
- A corrupted header will lead to loss of envelope, but no chaining effects.



EPAM Synchronization

- To ensure that EPAM values in the received envelope headers can be used as write pointers for the RX_FIFO buffer (and not collide), the EPAM values across multiple ONUs shall be synchronized.
- EPAM synchronization mechanism:

1. MPCP ***LocalTime*** counter runs at the same frequency as EQ, i.e., TQ = 2.56 ns (the transmission duration of one EQ).
2. GATE's ***StartTime*** represents both the ONU local time at which it should start transmission AS WELL AS the OLT'S expected reception time for this burst.
    - This is easy to achieve by offsetting the GATE's ***Timestamp*** by RTT of a given ONU (instead of offsetting the ***StartTime*** as is done in 10G-EPON).
3. When ONU's MPCP Envelope Activation process passes an envelope information down to MPRS, as part of that message, it would pass the 5 LSB of its MPCP ***LocalTime***, which by definition also matches the ***StartTime***.
4. The MPRS Input Process stores these 5 bits in the EPAM field of the first envelope in the burst. The following envelopes in the same burst are automatically aligned to the first envelope.

State Diagram Variables

EnvLeft[rCol] - Number of EQs that remain to be transferred to MAC in the current envelope on channel *rCol*.

OUT_CLK - Clock signal that corresponds to even transitions of TX_CLK. Therefore OUT_CLK runs at half the frequency of TX_CLK.

RxEQ, OutEQ - Variables that hold temporary values of EQ.

rCol - A 2-bit integer that represents the column in RX_FIFO buffer currently being read. Each column corresponds to a separate reception channel, i.e., a separate 25GMII interface.

rRow - A 5-bit integer that represents the row in RX_FIFO buffer currently being read.

RX_CLK[ch] - This boolean variable represents the continuous clock that provides the timing reference for the transfer of the RXC[ch]<3:0> and RXD[ch]<31:0> signals received on channel *ch*. The variable RX_CLK[ch] becomes true on each edge of the clock and becomes false on read.

RXD[ch]<31:0> - 32-bit vector representing one data transfer on channel *ch* of 25GMII. See definition of 25GMII in 802.3by

RXC[ch]<3:0> - 4-bit vector representing one control transfer on channel *ch* of 25GMII. See definition of 25GMII in 802.3by.

wRow - A 5-bit integer that represents the row in TX_FIFO buffer currently being written.

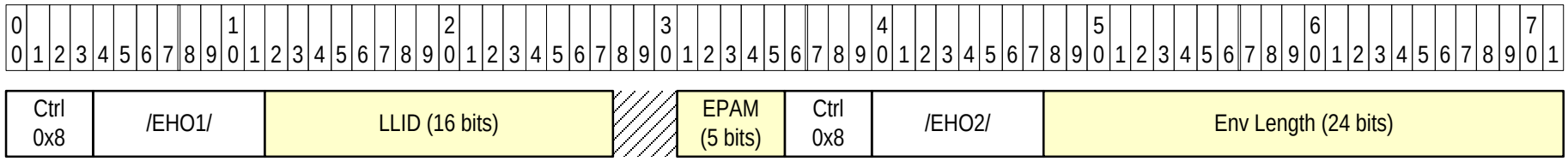
State Diagram Functions

bool IsHeader(eq) -

This function returns true is the parameter *eq* represents an envelope header. A header is an ordered set (see 802.3, clause x.x.x)

```
bool IsHeader( eq )
{
   return( eq<3:0>   == 0x8    AND    // Control bits (low half)
           eq<39:36> == 0x8    AND    // Control bits (high half)
           eq<11:4>  == OCODE1 AND    // Env. Header O-code #1 (TBD)
           eq<47:40> == OCODE2 );     // Env. Header O-code #2 (TBD)
}
```

Envelope Header EQ



State Diagram Functions

bool IsMisaligned(eq) -

This function returns true is the parameter *eq* is misaligned, i.e., shifted by half-eq.

```
bool IsMisaligned( eq )
{
   return(( eq<39:36> == 0xF AND              // Misaligned NO_ENV_CODE
            eq<71:40> == NO_ENV_CODE_LO )     // … s.b. NO_ENV_CODE_HI
          OR
          ( eq<39:36> == 0x8 AND              // Misaligned Env. Header
            eq<47:40> == OCODE1 ));           // … s.b. OCODE2
}
```

Thank You