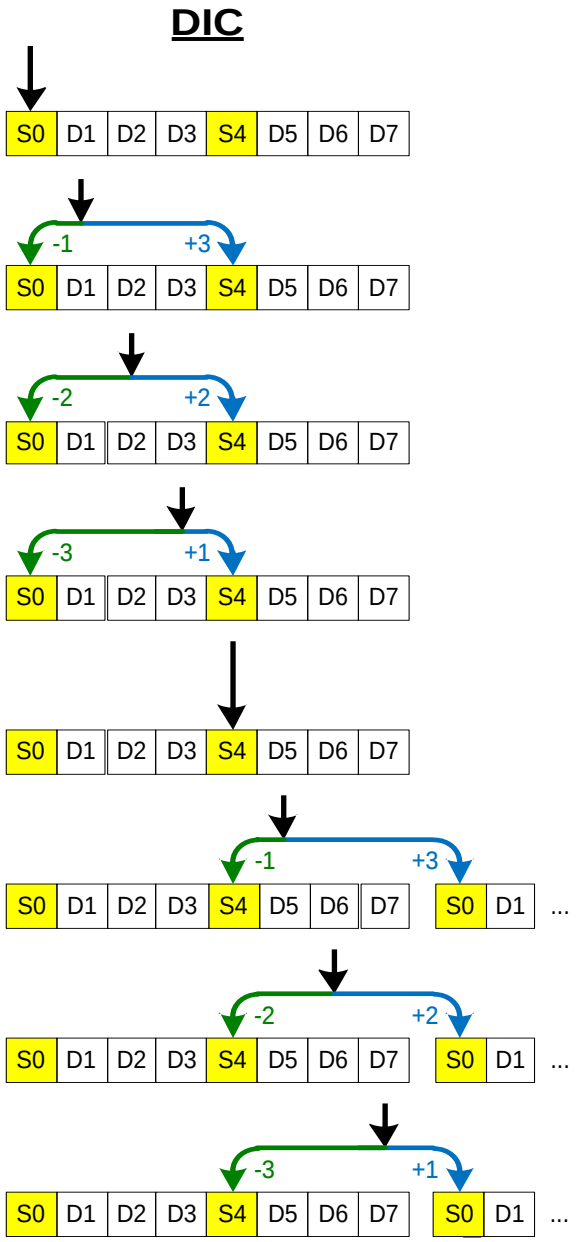
# /S/ Character Alignment

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

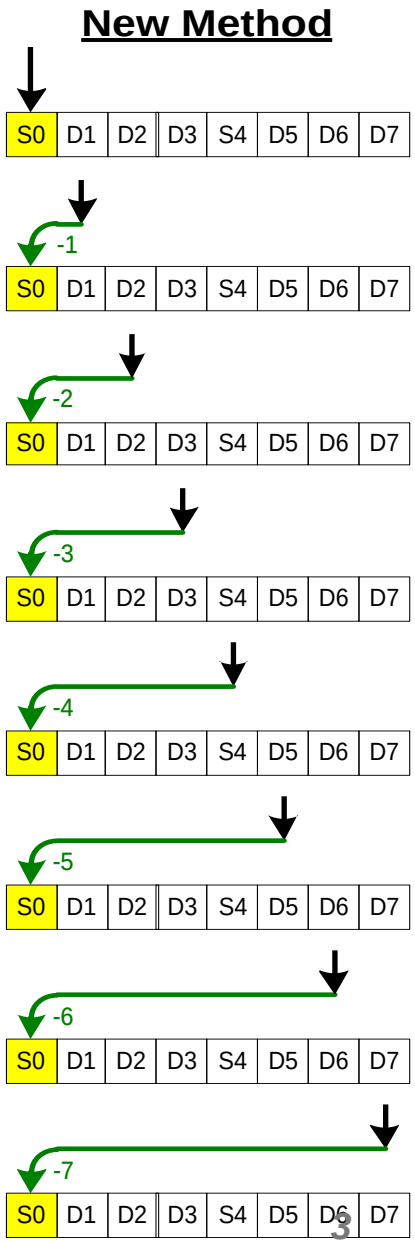
# 64b/66b Start-of-Frame Alignment

- In 64b/66b systems, a frame can start in octet 0 or octet 4.
- Generally, Deficit Idle Count (DIC) method is used to maintain the effective data rate by sometimes inserting and sometimes deleting idle characters to align the Start control character.



# MPRS Start-of-Frame Alignment

- In 25G-EPON we will have a strong FEC, which consumes 17-18% of bandwidth. We do not ever need to extent the idle gap to maintain the effective data rate.
- We need to leave at least 4 idle characters between frames to ensure proper start-of-frame detection.
- **Proposal: idle gap shall be unchanged or reduced (but never increased) to ensure that the next frame starts at octet 0.**

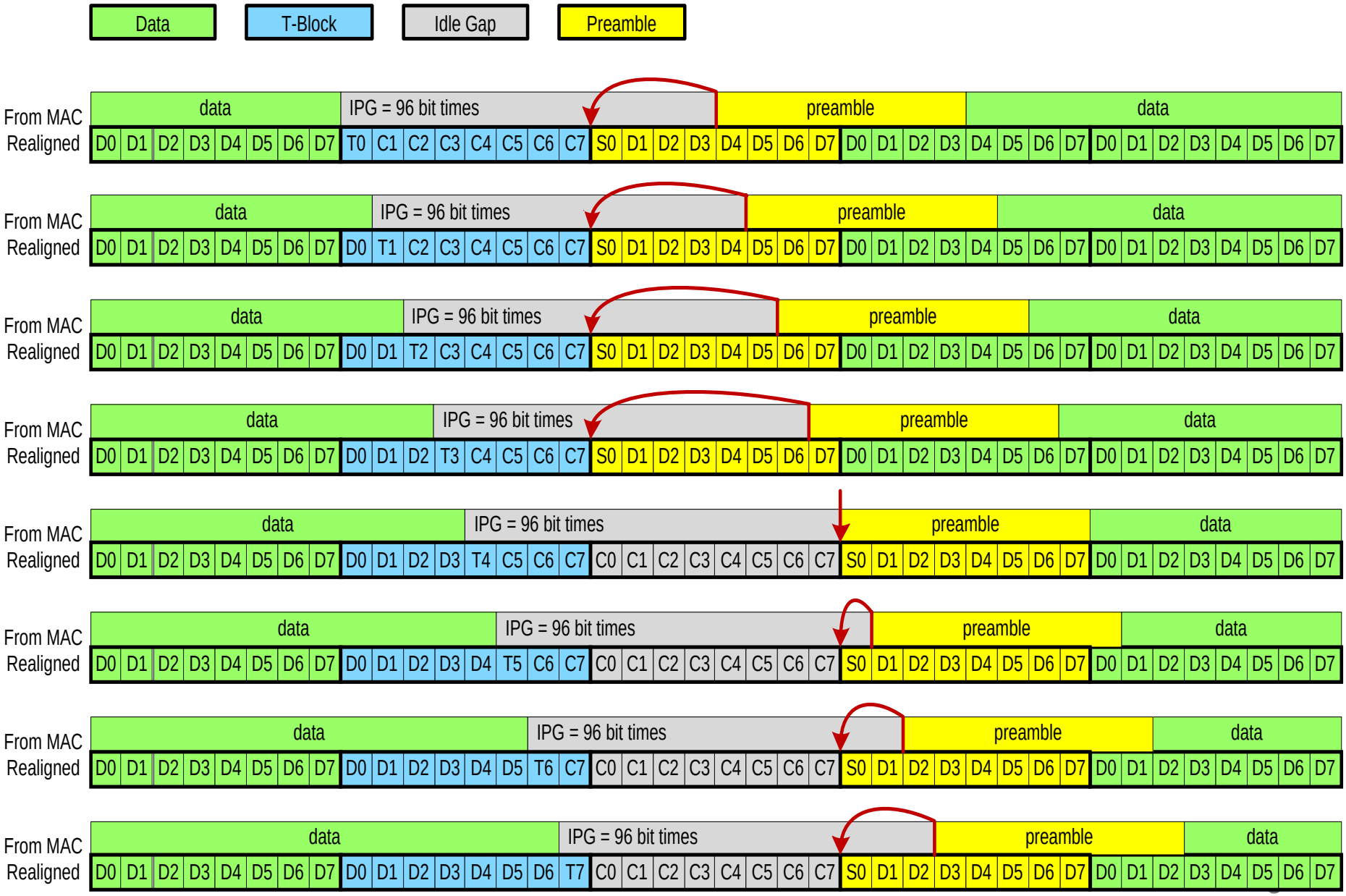


# Start-of-Frame Alignment & IFG

- For back-to-back frames, depending on the length of previous frame there are 8 possible adjustments of the IPG:
  - $\boldsymbol{NewIPG = IPG - \left((FrmSize + Pre + IPG)\ \mathbf{MOD}\ 8\right)}$
    $\boldsymbol{= 12 - \left((FrmSize + 20)\ \mathbf{MOD}\ 8\right)}$
  - *NewIPG* ranges from **5** to **12** octets

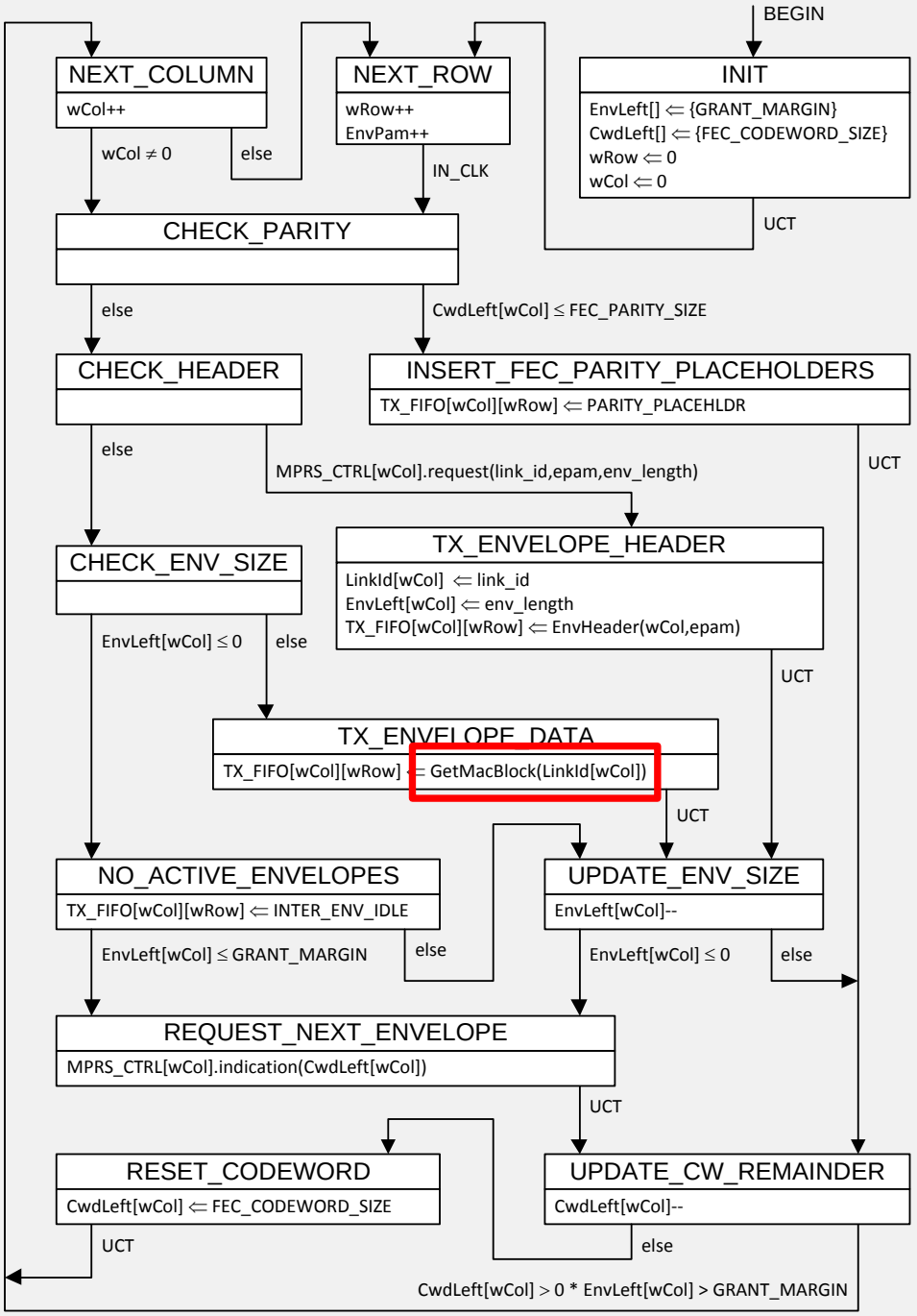
- Maximum throughput gain:
  - **8.75%** if all frames happen to be 67 bytes long
- Typical throughput gain:
  - **2.28%** for real downstream packet mix
  - **2.78%** for real upstream packet mix

# Start-of-Frame Realignment



# MPRS Control

- The entire Start-of-Frame Alignment mechanism is confined to the function **`GetMacBlock(link_id)`** called by the MPRS Input Process



# GetMacBlock(link_id) function

```
PrevIdle[] = {true};  // Previous octet from a given MAC (link_id) was an Idle
                      // Global array of booleans that retain their values
                      // between successive calls to GetMacBlock()

EQ GetMacBlock(link_id)
{
    EQ eq;  // consists of 8 elements in the format { 1b:CTRL/DATA, 8b:value }

    for( octet_index = 0; octet_index < 8, octet_index++ )
    {
        tx_data = GetMacOctet( link_id );              // Get 8 bits from MAC

        if( IsIdle(tx_data) AND !PrevIdle[link_id] )   // The first Idle after Data
        {
            PrevIdle[link_id] = true;
            eq[octet_index] = { CTRL, 0xFD };          // Store /T/-character
        }
        else if( IsIdle(tx_data) )                     // Idle after Idle
            eq[octet_index] = { CTRL, 0x07 };          // Store /I/-character

        else if( PrevIdle[link_id] )                   // The first Data after Idle
        {
            PrevIdle[link_id] = false;
            octet_index = 0;                           // Shift to octet 0
            eq[octet_index] = { CTRL, 0xFB };          // Store /S/-character
        {
        else                                           // Data after Data
            eq[octet_index] = { DATA, tx_data };       // Store regular Data octet
    }
    return eq;
}
```

# Function Definitions

**GetMacOctet(link_id)**

This function returns eight bits of data from the MAC entity designated by link_id. All eight bits in the returned octet either represent a data octet (i.e., all have values of ONE or ZERO), or all have value of DATA_COMPLETE (see 6.3.1.1). If the PLS_DATA.request (OUTPUT_UNIT) initially returns DATA_COMPLETE and transitions to data (ONE or ZERO) the first data bit is justified to bit position 0 in the octet and a full octet of data is returned.

**IsIdle(tx_data)**

This function returns ‘true’ if the octet represented by tx_data contains DATA_COMPLETE values, otherwise the function returns ‘false’.

# Thank You