100 Gbit Ethernet PCS/PMA and MAC FPGA implementation

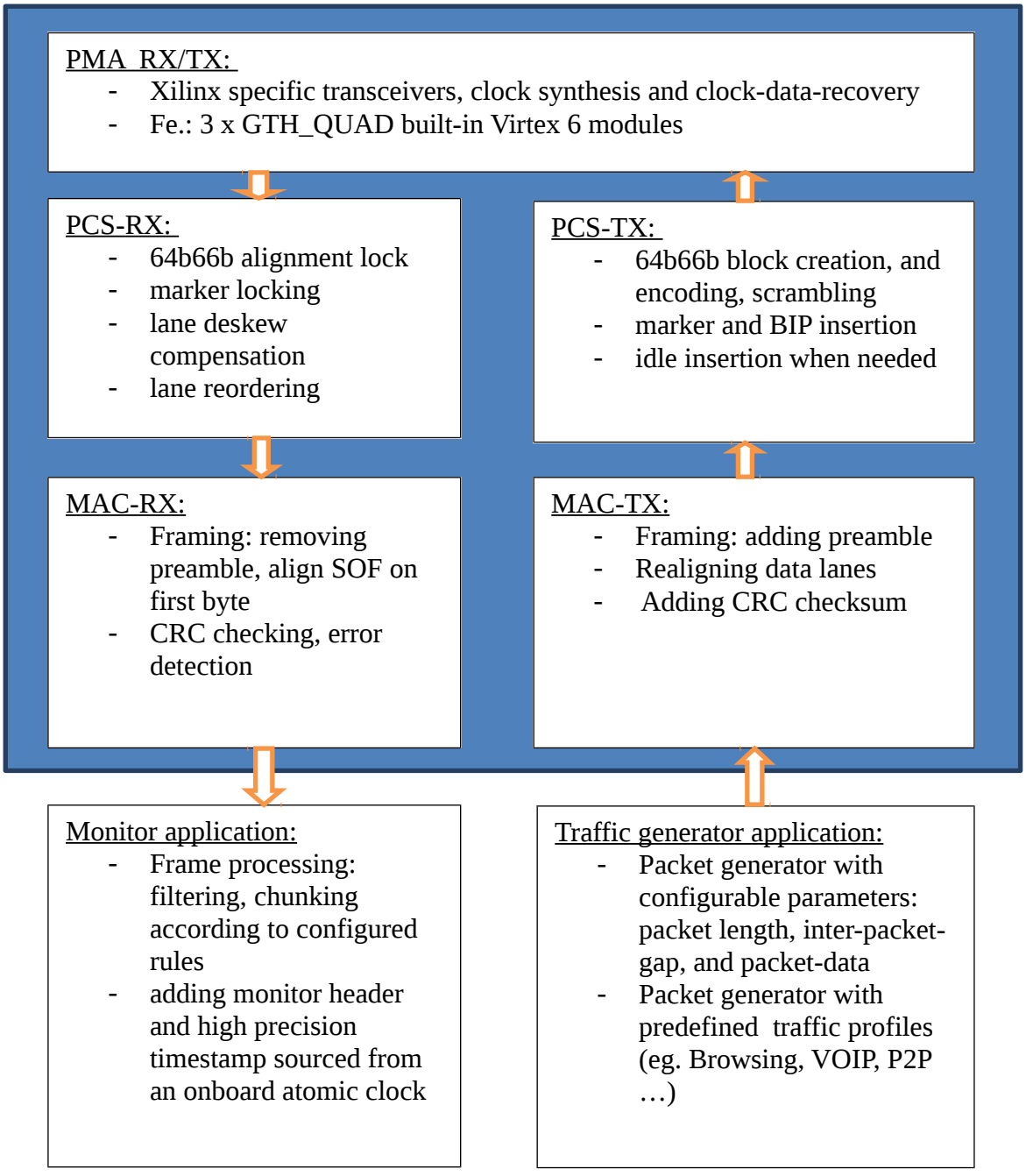
… optimized for your needs:

- 802.3ba-2010 compliant
- field tested for industrial usage
- full HDL source code available for developers
- tested on CGEP-100 devices

AITIA's 100Gbps IP core:

Features:

- IEEE 802.3ba-2010 full compliant implementation
- Source code available for further development!
- Uses Xilinx Virtex 6 device specific GTH transceivers, can be easily ported to other devices too
- 512bit//320bit Receive/Transmit MAC data-interface running at 312,5Mhz
- Optimized ethernet CRC checksum calculation on Rx and Tx MAC interfaces
- Tested on CGEP with CFP SR optical modules
- Simple and parameterizable traffic generator application available
- Sample application with 100G partitioned ISE project for fast development runtime, and guaranteed timing compliance.



100Gbps PCS/PMA functional description:

All major building modules of the 100Gbps ethernet core are well separated according to their functions. This results in code thats easier to understand and develop, desing placement and partitioning is also much more convenient.

The main components are:

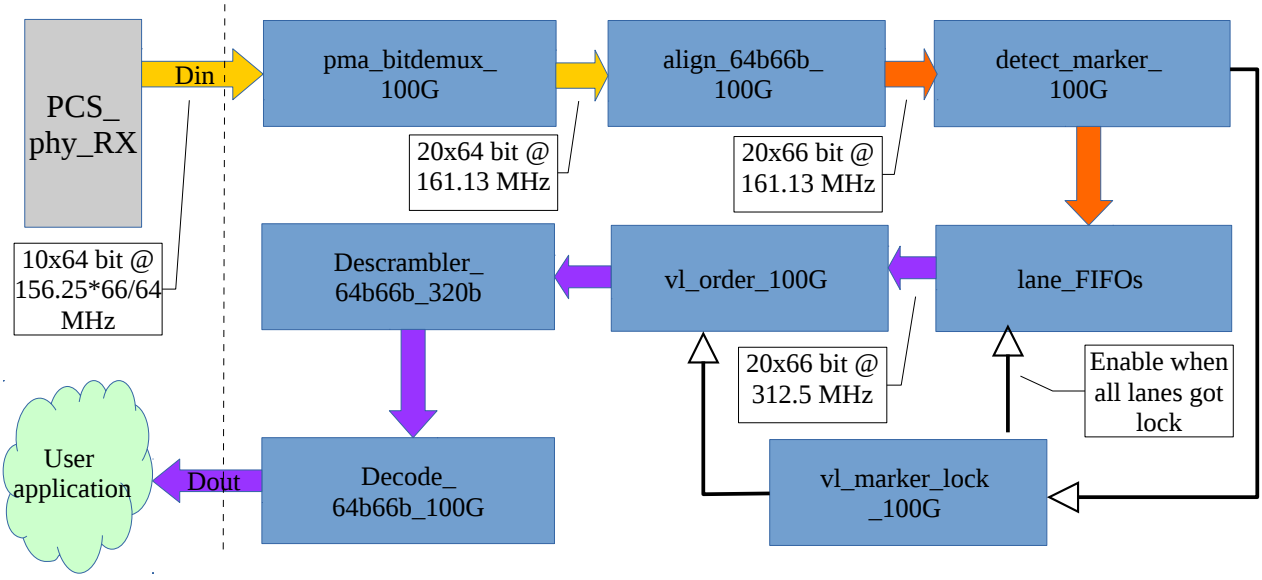
- ETH_100G_v6: this contains the Virtex 6 specific transcievers, the 100G RX/TX PCS/PMA, and MAC functions
    - pcs_phy_100G: this contains the V6 specific transcievers, and clocking modules
    - PCS_RX_100G: RX PCS/PMA functions
        - pma_bitdemux_100G
        - align_64b66b_100G
        - detect_marker_100G
        - lane_FIFOs
        - vl_marker_lock_100G
        - vl_order_100G
        - descrambler_64b66b_320b
        - decode_64b66b_100G
    - MAC_RX_100G: RX MAC functions
        - CRC32_512_wtable
    - MAC_TX_100G: TX MAC functions
        - CRC32_312_wtable
    - PCS_TX_100G: TX PCS/PMA functions
        - encode_64b66b_100G
        - scrambler_64b66b_320b
        - lane_FIFOs
        - insert_marker_100G
        - pma_bitmux_100G

User applications:

- CG_FrameGen_wBRAM: Simple parametrizable traffic generator.
  This module can send various types and numbers of predefined ethernet frames with variable length and interframe-gap
- CG_Monitor: Received ethernet frames are filtered, chunked, and time-stamped.
  Headered packet are written into DDR3 storage for speed compensation, and are then sent out over 4x10Gbps ethernet links for further analysis; like traffic mix, or deep packet inspection.
  A full functional application is avaliable for the CGEP_4G4X_1C reference board.

– *pcs_phy_100G:* This is the device specific physical interface of the 100G core. In our case its a Virtex 6 device, so 3 GTH transcievers are used to connect to and external optical CFP module over 10x10G electrical lines.
The GTH transcievers are configured in RAW 64 bit mode, no gearboxes are used (100G marker processing permits to use the built in 64b/66b encoder, or line scrambler).
If you want to use another device, for example an Altera chip, you only have to replace this module, the other parts of the core do not depend on the physical interface used.

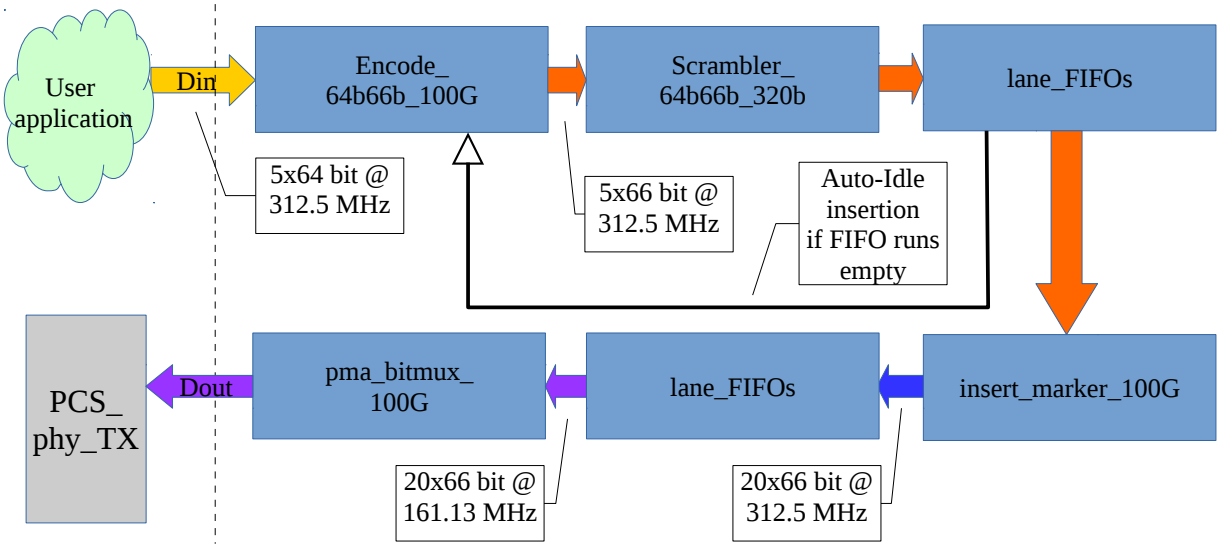
– *PCS_RX_100G:* This module implements the RX PCS/PMA functions according to IEEE 802.3ba-2010
The 20 virtual lanes are multiplexed into 10 physical lanes, so the interleaved bits have to be unfolded (pma_bitdemux_100G) to get the raw lane-data.
The GTH serdes has a 64 bit output, so we have to convert and lock on 64b/66b alignment too (align_64b66b_100G).
After that we can detect the markers that are inserted into the virtual lanes, to achieve cross-channel alignment (detect_marker_100G, vl_marker_lock_100G).
There can be a maximum of 180ns time delay between physical channels because of the optical transmission and electrical conversions, so we have to insert FIFOs on every virtual-lane to compensate delays (lane_FIFOs). These FIFO-s also take care of the PCS(161.13MHz) to MAC(312.5MHz) clock conversion.
After that virtual-lanes must be reordered as sent by the far end device (vl_order_100G).
Next is the self-synchronizing descrambler. This is based upon a simple linear feedback shift register (descrambler_64b66b_320b).
The last stage is the 66b decoder, which determines the ethernet frame boundaries, and other control codes (decode_64b66b_100G)



– *MAC_RX_100G:* This module aligns frame data to always start on the first byte, and cuts off ethernet preamble. This also means a 320 bit to 512 bit data-width conversion to compensate speed differences. Ethernet frame checksum is calculated and compared on the received data (CRC32_512_wtable). This is done in a sophisticated table-based checksum refactoring manner.

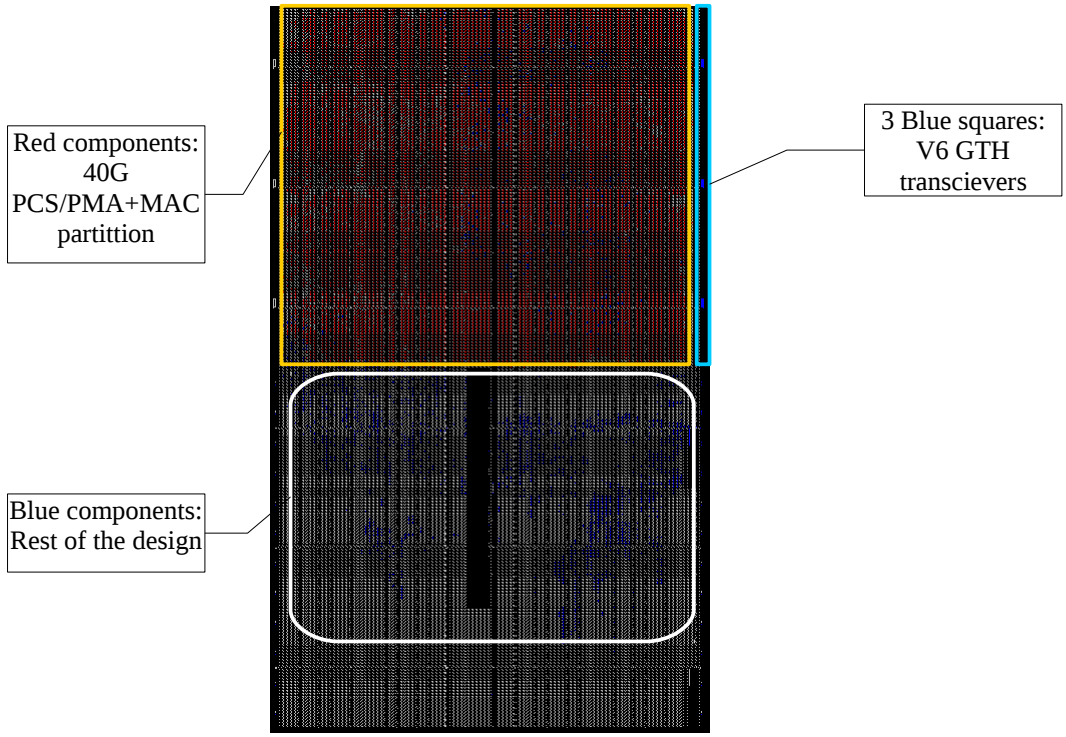
– *MAC_TX_100G:* This module is responsible for assembling the ethernet frame from the raw input data. Preamble is added on the start, and checksum on the end of frame(CRC32_312_wtable). By default MAC input data width is 320 bits, this can be changed if needed to 512 bit, for performace traffic generator applications.

– *PCS_TX_100G:* This module implements the TX PCS/PMA functions according to IEEE 802.3ba-2010
First input data is encoded into 64b format, frame control and data codes are added when needed (encode_64b66b_100G). If there is no input data, then idle codes are added to ensure a continuous dataflow.
After that encoded data is scrambled (scrambler_64b66b_320b) to dampen the DC component in the signal.
Lane FIFO-s (lane_FIFOs) are added to store the data to compensate speed differences from idle-code and lane-marker insertion. Also clock conversion from MAC(312.5MHz) to PCS(161.23MHz) clock is done in this component.
Virtual lanes are created by peridically adding a marker into the datastream (insert_marker_100G).
Finally lane bits are multiplexed to create physical lanes (pma_bitmux_100G).



Partitioning the design:

To achieve better timing closure and faster builds it is advised to partition the design into atleast two parts.
The partitioning is done in two steps. First we have to carefully choose the partitions, and placements for building the whole core as one.
This is the „reference build", which is used only to implement the 100G partition and its interfaces. We can simply reuse the fully implemented partition files in the following builds. The placement, internal wiring and timing closure of the 100G partitions remain untouched, so only the rest of the design has to be recompiled after that.



XC6VHX255T-2FF1155

The picture above shows a pre-partitioned design in FPGA-editor:
The red components are part of a fixed partition consisting of the 100G RX/TX PCS/PMA and MAC core, but without the physical transcievers. The blue components are the rest of the design interfacing to the 100G MAC core. The 3 GTH transcievers are on the right upper side, those are not part of the 100G partition.
Partitioned and implemented 100G reference design is avaliable for CGEP_4X4G_1C device in Xilinx ISE project format.

Interfacing to the core:

- Physical IO ports:

```
    Q0_REFCLK_N          : IN  STD_LOGIC;
    Q0_REFCLK_P          : IN  STD_LOGIC;
    Q1_REFCLK_N          : IN  STD_LOGIC;
    Q1_REFCLK_P          : IN  STD_LOGIC;
    Q2_REFCLK_N          : IN  STD_LOGIC;
    Q2_REFCLK_P          : IN  STD_LOGIC;
```

Those are the differential reference clocks for the device specific GTH Quad transcievers. These should be 3 independent 156.25 MHz crystal oscillator with atleast 100ppm stability. Input pin assignments are locked in the .ucf file.

```
    DRP_CLK              : IN  STD_LOGIC;
```

This clock input is needed for completing GTH initialization and reset. It must be a free running clock in range of 25 – 60 MHz.

```
    RXN_GTH              : IN  STD_LOGIC_VECTOR(12-1 downto 0);
    RXP_GTH              : IN  STD_LOGIC_VECTOR(12-1 downto 0);
    TXN_GTH              : OUT STD_LOGIC_VECTOR(12-1 downto 0);
    TXP_GTH              : OUT STD_LOGIC_VECTOR(12-1 downto 0);
```

These are the high-speed serial data lines of the GTH serdes. Only 10 out of the 12 pairs are used. The implementation tool assigns them automatically to the corresponding physical pins.

- User ports (fabric):

```
    CG_Clk_RX            : OUT STD_LOGIC;
    CG_Clk_TX            : OUT STD_LOGIC;
```

Two independent MAC clocks are synthetized from the PCS clocks for RX and TX. All user logic acts on the rising edge of those clocks.

```
    CG_RXData            : OUT STD_LOGIC_VECTOR(8*64-1 downto 0);
```

Receive Data from 100G Rx-MAC. Ethernet frames always start on the first byte.

```
    CG_RXMod64           : OUT STD_LOGIC_VECTOR(6-1 downto 0);
```

This shows the number of valid bytes at the end of the frame.
0 means all bytes are valid, b”100111” means 319 bytes are valid.

```
    CG_RXDv             : OUT STD_LOGIC;
```

Data valid signal: RXData, RXSof, RXEof is only valid when this signal is in high-state.

```
    CG_RXSof            : OUT STD_LOGIC;
```

Start of frame signal: This indicates the first valid data-nibble in an ethernet frame.

```
    CG_RXEof            : OUT STD_LOGIC;
```

End of frame signal: This indicates the last valid data-nibble in an ethernet frame. (RXMod64 is only valid when this signal is in high-state)

```
    CG_RXCRCErr         : OUT STD_LOGIC;
```

Checksum error signal: The indicates, that calculated and received ethernet checksums don't match.

```
    CG_RXErr            : OUT STD_LOGIC;
```

Receive error signal: This indicates a far-end receive error, or and invalid 64b/66b code.

```
    CG_TXAv             : OUT STD_LOGIC;
```

Transmit buffer available: This indicates, that the transmit buffer is ready to accept an ethernet frame for transmission (60 – 16kbyte in length)

```
    CG_TXData           : IN  STD_LOGIC_VECTOR(8*64-1 downto 0);
```

Transmit Data for 100G Tx-MAC. Ethernet frames must always start on the first byte.

```
    CG_TXMod64          : IN  STD_LOGIC_VECTOR(6-1 downto 0);
```

This shows the number of valid bytes at the end of the frame. 0 means all bytes are valid, b”111111” means 511 bytes are valid. (It is also possible to use the Tx-MAC in 320byte mode)

```
    CG_TXDv             : IN  STD_LOGIC;
```

Data valid signal: The Tx-MAC only accepts input data, when this signal is in high-state.

```
    CG_TXSof            : IN  STD_LOGIC;
```

Start of frame signal: This indicates the first valid data-nibble in an ethernet frame.

```
    CG_TXEof            : IN  STD_LOGIC;
```

End of frame signal: This indicates the last valid data-nibble in an ethernet frame. (TXMod64 is only valid when this signal is in high-state)

- Misc signals:

```
    PCS_RXSof            : OUT STD_LOGIC;
```

PCS start of frame: for debug purposes, and precise timestamping.

```
    CG_Rst               : IN  STD_LOGIC;
```

Reset signal: This signal resets the user portion of the 100G core (fabric reset)

```
    CG_CDR_Rst           : IN  STD_LOGIC;
```

GTH CDR Reset: This signal resets the Rx clock syntheser of the GTH transciever (phy reset)

```
    CG_signal_ok         : IN  STD_LOGIC;
```

Receive signal OK. Can be tied to high if no physical indication is available.

```
    CG_initdone          : OUT STD_LOGIC;
```

GTH initialization complete, core is ready.

```
    CG_aligned           : OUT STD_LOGIC;
```

All virtual lanes are aligned and locked, core is ready to receive frames.