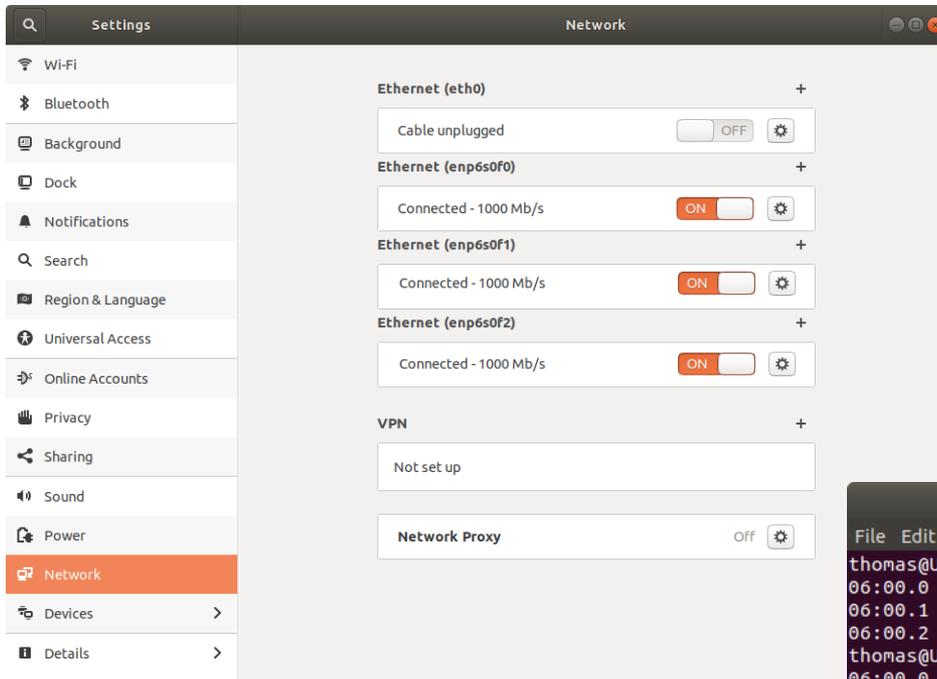


Realizing Ethernetlinks with the PCIe HCC DMA IP Core

Thomas Zerrer

V 1.2



This example shows 3 additional ethernet links visible in the device manager

3 PCIe Functions visible via lspci

```
thomas@Ubuntu1: ~/sleth/linux
File Edit View Search Terminal Help
thomas@Ubuntu1:~/sleth/linux$ lspci -d 1ad4:
06:00.0 Ethernet controller: Device 1ad4:2000
06:00.1 Ethernet controller: Device 1ad4:2000 (rev 01)
06:00.2 Ethernet controller: Device 1ad4:2000 (rev 02)
thomas@Ubuntu1:~/sleth/linux$ lspci -v -d 1ad4:
06:00.0 Ethernet controller: Device 1ad4:2000
    Flags: bus master, fast devsel, latency 0, IRQ 49, NUMA node 0
    Memory at fb100000 (32-bit, non-prefetchable) [size=64K]
    Capabilities: <access denied>
    Kernel driver in use: SLETH

06:00.1 Ethernet controller: Device 1ad4:2000 (rev 01)
    Flags: bus master, fast devsel, latency 0, IRQ 57, NUMA node 0
    Memory at fb110000 (32-bit, non-prefetchable) [size=64K]
    Capabilities: <access denied>
    Kernel driver in use: SLETH

06:00.2 Ethernet controller: Device 1ad4:2000 (rev 02)
    Flags: bus master, fast devsel, latency 0, IRQ 65, NUMA node 0
    Memory at fb120000 (32-bit, non-prefetchable) [size=64K]
    Capabilities: <access denied>
    Kernel driver in use: SLETH

thomas@Ubuntu1:~/sleth/linux$
```

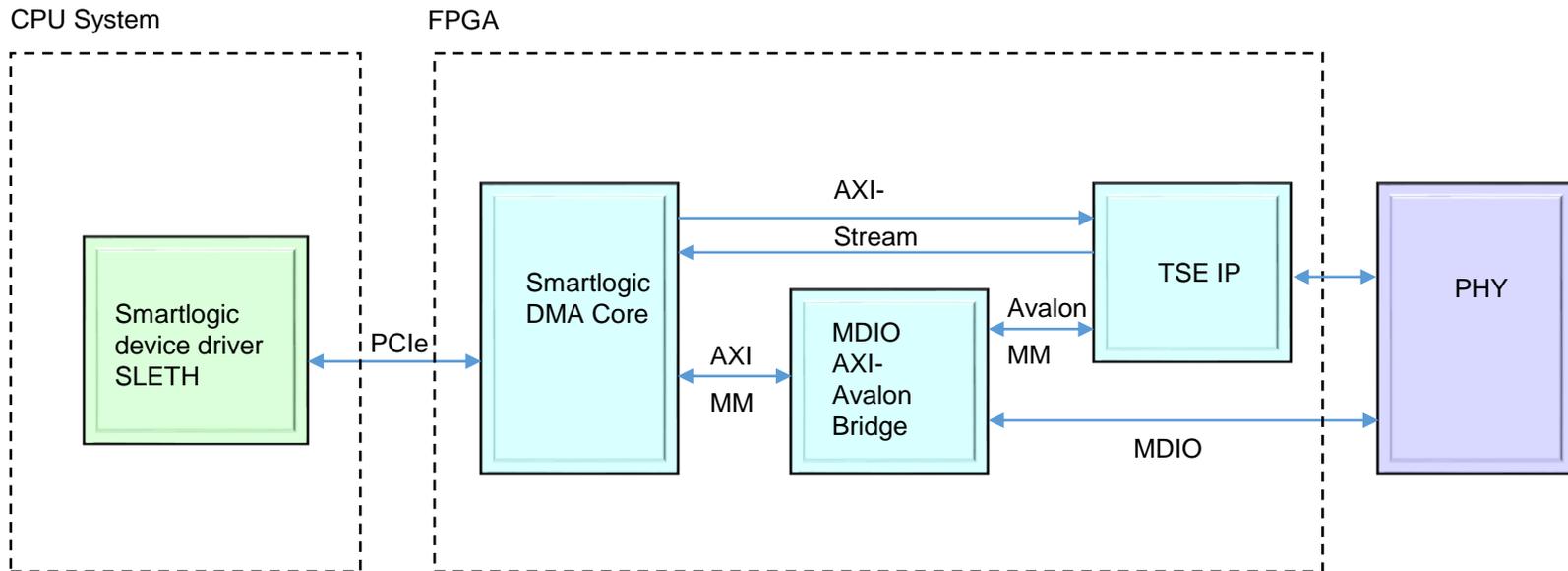
Product Highlights

- Fully tested package consisting of PCIe DMA HCC IP Core, PHY Bridge and device driver
- Available for Xilinx and Intel FPGAs
- Copper or Fiber based links supported
- Ethernet link speed is only dependent on PHY and limited by PCIe throughput
- Due to the PCIe multi function approach up to 4-8 ethernet links can be realized with one PCIe connection (The exact number of possible links is determined by the capabilities of the selected FPGA device family)
- Ethernet links and custom DMA transmission can be mixed

This AppNote describes how to successfully add one or more ethernetlinks to the FPGA

- Overview over the SLETH reference design
- Generating the Intel tripple speed ethernet IP core, the Intel PCIe IP core and the transceiver reconfiguration IP core
- Board layout and PHY selection
- Checklist when you connect more than one ethernet link to the FPGA (multifunction case)
- PCI Express IDs to ensure compatibility with the Smartlogic sleth device driver

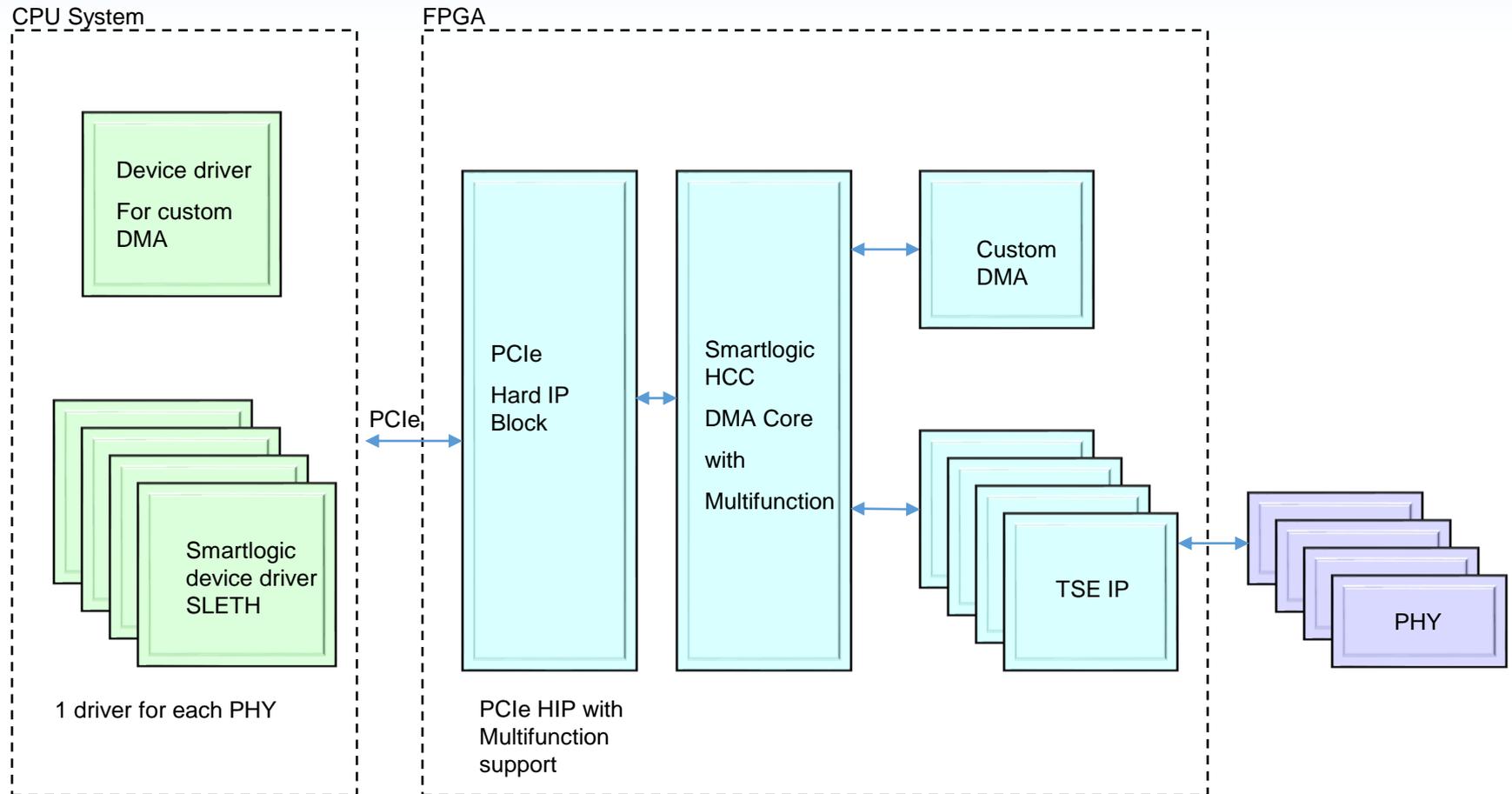
Overview of the ethernet reference design



The reference design is available along with a fully configured QuartusPrimeLite / Vivado project for the Cyclone 5 GT / KC705 development kit with a Marvell 88E1111 PHY. It can be easily modified to support other FPGA families.

The Intel tripple speed ethernetcore (TSE) requires a separate license from Intel / Xilinx which is not included in the SLETH package. Ordering Code is IP-TRIETHERNET / EF-DI-TEMAC-PROJ, price is at around 500 €. A free of charge evaluation version is included in the QuartusPrime / Vivado design suite.

Advantage of PCIe multi-function support



- 4-8 device drivers can be mapped (FPGA family dependent) to one PCIe HardIP Block
- Ethernetlinks can be freely mixed with custom DMA
- Ethernetlinks are visible in the OS as standard Ethernet socket

Details and parameters of the mdio axi avalon bridge

- This block realizes the AXI Lite to Avalon MM protocol conversion and includes an intelligent MDIO interface that automatically reads the PHYs registers.
- It is important to set the parameters for each instance correctly:

axi_data_width_g : 32 (do not change)
axi_addr_width_g : 15 (do not change)
mdio_freq_divider : x"xx" (divides the trn_clock to derive the mdc clock)
use_avalon_if_g : 1 enables Avalon MM interface, 0 enables m_axi interface
phy_addr_g : add the PHY's MDIO address here

For Cyclone V FPGAs three IP catalog cores have to be generated and added to the design:

1. The Transceiver Reconfiguration Controller Intel FPGA IP core with the IP variation filename „transceiver_reconfig.v“
2. The „Cyclone V Hard IP for PCI Express Intel FPGA“ IP core with the IP variation filename „intel_c5_pcie2.v“
3. The tripple speed ethernet IP core (must be separately licensed from Intel)

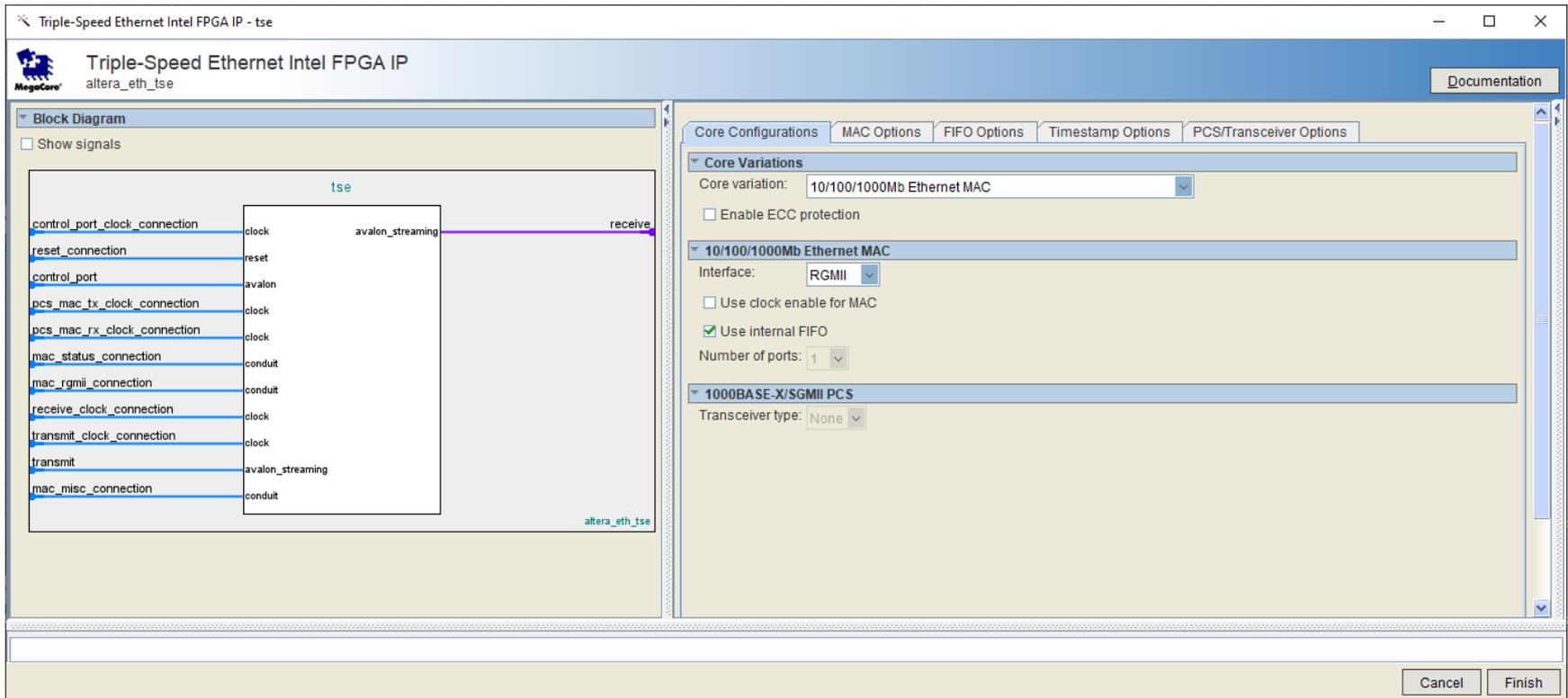
Select those IPs in the IP catalog and prepare them for configuration by double clicking.

Details for the first two cores can be found in a separate application note „AN_Intel_IP_catalog_flow.pdf“

Details on how to configure the Intel TSE ip core are shown on the next page.

Configuring the Intel TSE IP core

GUI of the TSE Ethernet IP Core:



Configuring the Intel TSE IP core

Important when you configure the Intel TSE IP core:

- The different flavours of the TSE IP Core serve various management interface types. Make sure to select only MII, GMII or RGMII interface types
- You can choose to work with or without internal FIFOs. However internal FIFOs with depth 4096 are recommended.
- Never activate the core internal MDIO interface. The MDIO interface is part of the Smartlogic IP core.
- Example of a working configuration :

The screenshot displays the configuration interface for the Triple-Speed Ethernet Intel FPGA IP core. The window title is "Triple-Speed Ethernet Intel FPGA IP - tse". The main area is divided into two panes:

- Block Diagram:** Shows a central block labeled "tse" with various input and output ports. On the left, ports include control_port_clock_connection (clock), reset_connection (reset), control_port (avalon), receive_fifo_status_clock_connection (clock), receive_fifo_status (avalon_streaming), transmit_0 (avalon_streaming), mac_misc_connection_0 (conduit), pcs_mac_tx_clock_connection_0 (clock), pcs_mac_rx_clock_connection_0 (clock), mac_status_connection_0 (conduit), and mac_rgmii_connection_0 (conduit). On the right, ports include mac_rx_clock_connection (clock), mac_tx_clock_connection (clock), receive (receive), and receive_packet_type (avalon_streaming).
- Configuration Options:** The "Ethernet MAC Options" tab is active. It contains several checkboxes:
 - Enable MAC 10/100 half duplex support
 - Enable local loopback on MII/GMII/RGMII*
 - Enable supplemental MAC unicast addresses
 - Include statistics counters
 - Enable 64-bit statistics byte counters
 - Include multicast hashtable
 - Align packet headers to 32-bit boundary
 - Enable full-duplex flow control
 - Enable VLAN detection
 - Enable magic packet detection

Below the Ethernet MAC Options, the "MDIO Module" section is visible, with the "Include MDIO module (MDC/MDIO)" checkbox unchecked and the "Host clock divisor" set to 35. The bottom of the window features "Cancel" and "Finish" buttons.

For the Marvell PHY sample constraint files are provided with the reference designs in the folder „sdc“.

Use the RGMII preconfigured files

`rgmii_input.sdc / rgmii_clock.sdc and rgmii_output.sdc`

If you are using another interface type (GMII or MII) you will have to adapt these constraints.

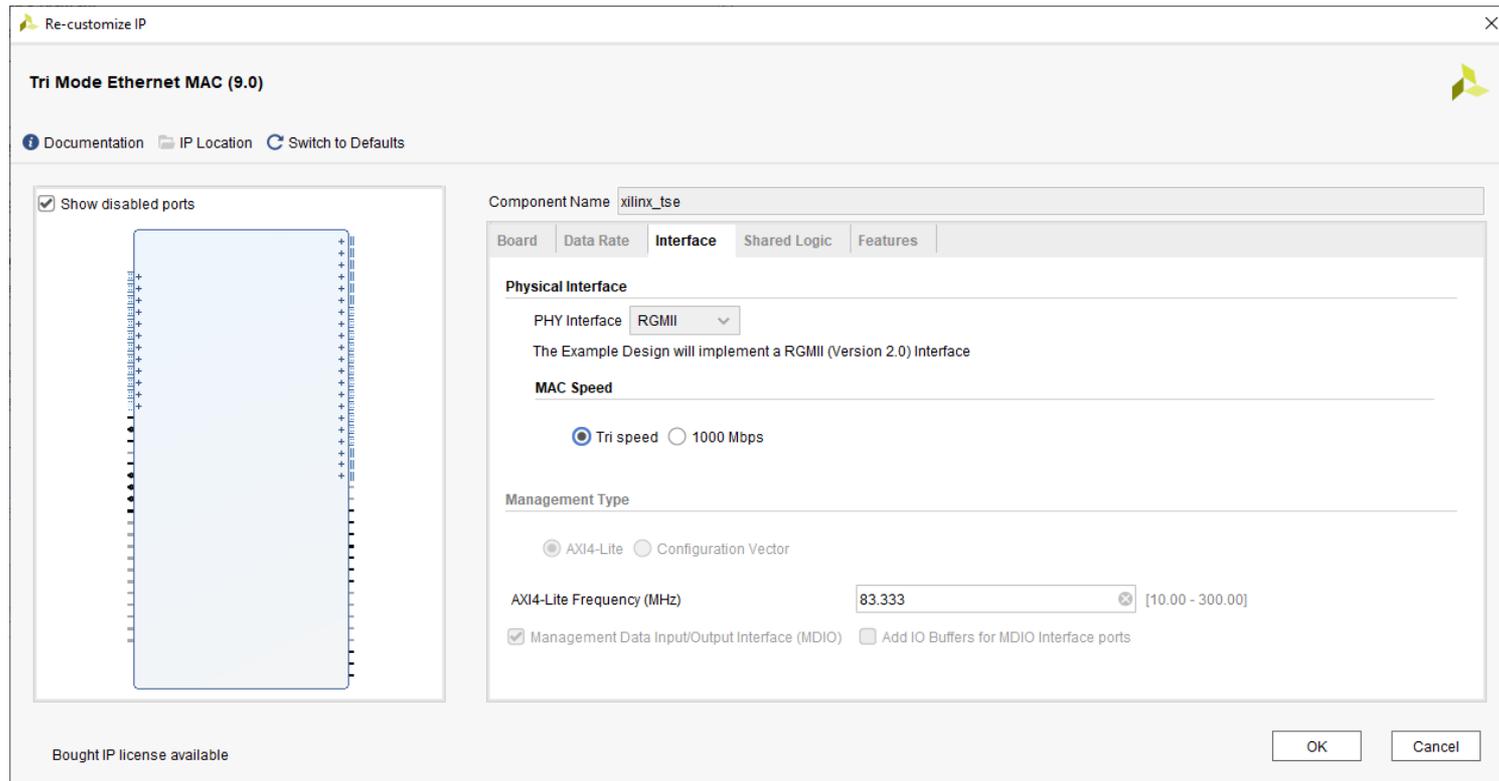
The mentioned constraint files are referenced out of the master constraint file.

For Xilinx FPGAs one IP catalog cores have to be generated and added to the design:

1. Tri Mode Ethernet MAC with name „xilinx_tse“

Configuring the Xilinx Tri Mode Ethernet MAC

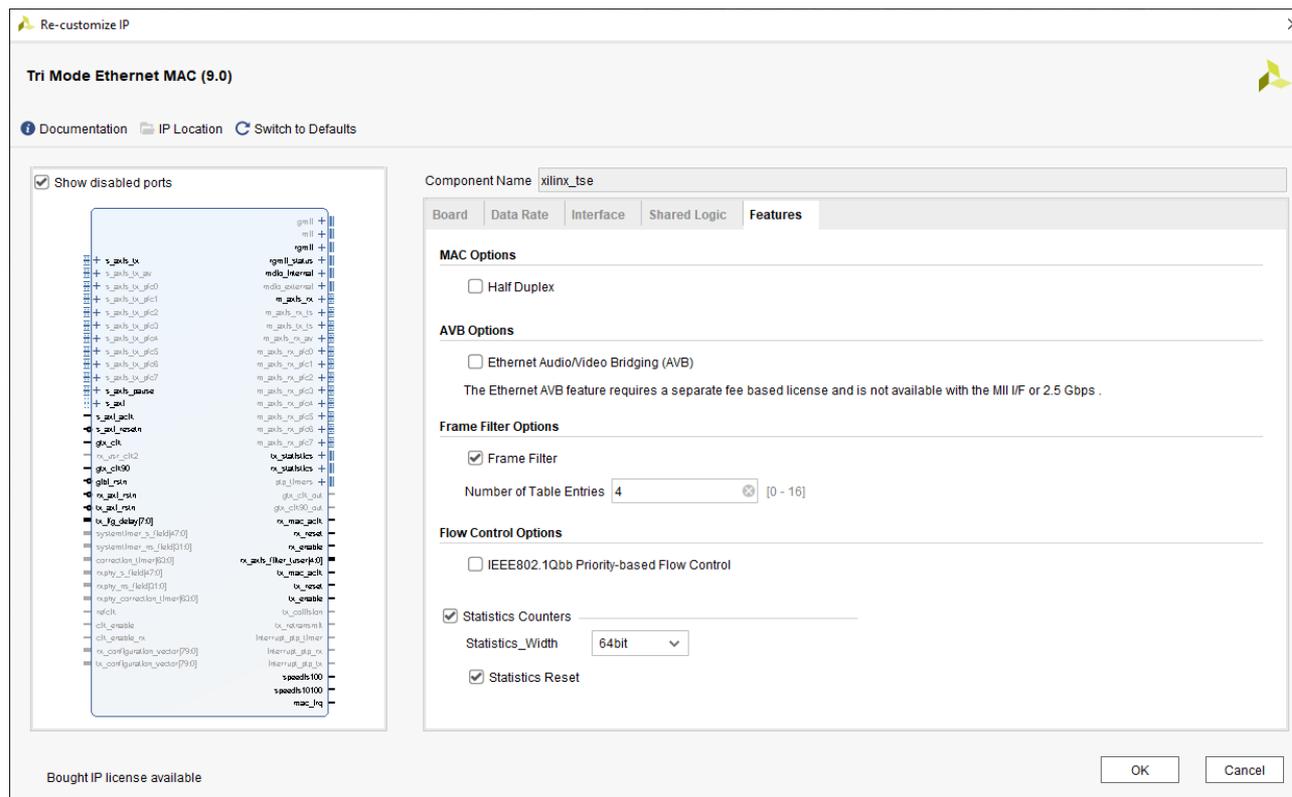
GUI of the Tri Mode Ethernet MAC:



Configuring the Xilinx Tri Mode Ethernet MAC

Important when you configure the Intel TSE IP core:

- The different flavours of the TSE IP Core serve various management interface types. Make sure to select only MII, GMII or RGMII interface types
- Never activate the core internal MDIO interface. The MDIO interface is part of the Smartlogic IP core.
- Example of a working configuration :



For the Marvell PHY, a sample constraint file is provided with the reference design.

It is „xilinx_tse_user_phytiming.xdc“.

Change this file according to the timing parameters of your selected PHY.

- Carefully decide which management interface type you want to use when you connect the PHY to the FPGA. Supported management interface types are currently MII, GMII and RGMII. Eventhough you might be able to configure other management interface types (like SGMII) for the FPGA, the Smartlogic device driver does currently only support MII, GMII and RGMII.
- Always choose dedicated clock inputs for the RX_CLK input. If you don't obey this rule, you will need one PLL for each Ethernetlink and may run out of PLLs. For each RX clock add the instance assignment „REGIONAL_CLOCK“ to your Quartus QSF file (for more details refer to the TSE User guide from Intel).
- Check, that the enet_rx* inputs are placed within the same clock region than the RX clock input
- Make sure to follow common layout guidelines (equal tracelengths, etc)
- RGMII Mode : The framework supports the programmable delays for the RX and TX Interface for each ethernetlink. This simplifies the TX clocking.
- The framework works with all Ethernet PHYs with the supported managment interface types. However make sure, that the PHY vendor supports a MDIO API device driver for your desired operating system.

Checklist when connecting more than one ethernetlink

- Ensure that you have licensed the multifunction IP core from Smartlogic
- The reference design for Intel requires that all Ethernet links run at the same data rate
- Provide a separate MDIO interface for each PHY
- Configure the PCIe IP for the desired amount of functions (1 PCIe function for one ethernet link) via the parameter editor. Each function requires one BAR and one AXI Master attached to this BAR.
- Details in order to understand the multifunction feature can be found in the Application Note [AN_working_with_multifunction.pdf](#)
- Ensure that the PCIe revision register is configured with the ethernet link index (e.g. 0x00, 0x01, 0x02 etc). The other PCIe IDs are identical for all functions. They are specified at the end of the document.
- Use 1 s_axis, 1 m_axis and 1 m_Axi interface of the PCIe DMA IP core exclusively for one instance of the TSE core (example design for 3 ethernetlinks provided)
- Configure the multifunction parameters in `pcie_ep_config_pkg.vhd` (see next page for an example)
- configure the plug & play parameters in the `dma_pkg.vhd`. These parameters contain important information for the device driver in order to configure each ethernetlink correctly (Management interface type, programmable delays, used interfaces, interrupts, etc)
- In order to minimize PLL usage in RGMII mode, it is possible to clock the RGMII TX data and GTX clock from the same clock with the same phase. In this case set the TX Delay bit in the Plug and Play parameters. See the reference designs for details.
- Use the provided constraints files `rgmii_input_mf.sdc` / `rgmii_clock_mf.sdc` and `rgmii_output_mf.sdc`
- Currently MSI-X is only supported for single functions. Multifunction designs must use MSI interrupts. (i.e. Use `msix_g` must be 0)

Parameter settings (pcie_ep_config_pkg.vhd):

New Parameter	Function	Example for 4 Ethernetlinks
ENABLE_MULTIFUNCTION_C	Enables or disables the multifunction feature	True
PCle_FN_NUMBER_C	Number of PCIe Functions (US / US+ support 1-4 Functions)	3 (=number of ethernetlinks)
AXIM_NUM_FN0_C	Number of AXI Masters mapped to Function 0	1
AXIM_NUM_FN1_C	Number of AXI Masters mapped to Function 1	1
AXIM_NUM_FN2_C	Number of AXI Masters mapped to Function 2	1
AXIM_NUM_FN3_C	Number of AXI Masters mapped to Function 3	0
AXIM_NUM_FN4_C	Number of AXI Masters mapped to Function 4	0
AXIM_NUM_FN5_C	Number of AXI Masters mapped to Function 5	0
PCle_DEV_CAP_EXT_TAG_SUPPORTED_C	Optimizes TX Performance when set to true. If more than 4 Functions or more than 4 m_axis interfaces are in use, this parameter must be set to „FALSE“	„TRUE“

Make sure to configure the following PCIe IDs in the parameter editor GUI of the PCIe HIP and in the pcie_ep_config_pkg.vhd VHDL package :

Parameter	Comment
PCIe_VEN_ID_C	0x1ad4
PCIe_DEV_ID_C	0x2000
PCIe_SUBSYS_VEN_ID_C	0x0
PCIe_SUBSYS_DEV_ID_C	0x0
PCIe_CLASS_CODE_C	0x20000

Make sure to set these upstream pnp parameters in the package dma_pkg.vhd (constant DMA_Write_pp_mem_c)

	Meaning of the bits	Comment
SL_ETH Header	Bit 2:0: PCIe function number (0-7) Bit 3 : RGMII : PHY Receive Control Delay (0 = no delay, 1=delay, RX_ID) SGMII / XGMII : (0 = no timer interrupt, 1 = timer interrupt (TX+1)) Bit 4 : RGMII : PHY Transmit Control Delay (0= no delay, 1=delay, TX_ID) SGMII / XGMII : (0 = no link interrupt, 1 = link interrupt (TX+2)) Bit 7:5: PHY Interface type Bit 31:8: 0xFFFF_FF identifier	Each PCIe function has its own header for identification Interface Type : 000 = RGMII (with MDIO) 001 = GMII (with MDIO) 010 = MII (with MDIO) 011 = SGMII (full, with MDIO) 100 = SGMII (full SGMII, no MDIO) 101 = 1000 BASE-X (SGMII) (no MDIO, fixed link) 110 = 10G-BASE-R (no MDIO, fixed link)
Entry 1 and 2	Bit 7:0: Logical channel Bit 15:8 : MSI Vector for Interrupt BIT 20:16 : Interrupt Enable Bit no (0-31) Bit 21: Mode (0= data, 1 = meta) Bit 22: Interrupt (0 does not trigger, 1 triggers) Bit 27:23: channel_index (0-31) Bit 31: 28: physical interface (0-15)	
Entry 3	Bit 31:0: 0xACAD_xxxx : Bit 15:0 of MAC address in reverse order	0xACAD is an identifier to mark the following 6 bytes as a MAC address
Entry 4	Bit 31:0: Bit 47:16 of MAC address in reverse order	Example MAC address (47 downto 0): 00-1C-23-17-4A-CB Entry 3 : 0xACAD_CB_4A Entry 4 : 0x17231c00

Make sure to set these downstream pnp parameters in the package dma_pkg.vhd (constant DMA_Read_pp_mem_c)

	Meaning of the bits	Comment
SL_ETH Header	Bit 2:0: PCIe function Number (0-7) Bit 7:3: PHY_Address (5 Bit) Bit 31:8: 0xFFFF_FF	Each PCIe function has its own header for identification
Entry 1	Bit 7:0: Logical channel Bit 15:8 : MSI Vector for Interrupt BIT 20:16 : Interrupt Enable Bit no (0-31) Bit 27:23: channel_index (0-31) Bit 31: 28: physical interface (0-15)	

Known limitations

- It is important to know, that the `m_axis_aclk` frequency should never be greater than $1.8 * trn_clk$. If for example 2.5 Gbit/s links are targeted with an 8 bit interface, it is recommended to clock `m_axis_aclk` with the `trn_clk` and to configure a 32 bit data width. By use of the `axis_stream_xy` ip core (part of this core) the data can be converted back to 8 bit with the 312.5 MHz.
- See also the user guide of the PCIe HCC IP core for more limitations
- Due to designflow restrictions, it is necessary to configure the PCIe link parameters (link speed and link width) at two different locations in exactly the same way. The PCIe HardIP is configured with GUI settings and they have to be entered also in `pcie_ep_config_pkg.vhd` (see application note AN_Intel_IP_catalog_flow) for details.

Linux Driver (I)

Installing the driver

- Within the folder sleth/linux is a makefile. Simply type make to compile and to get the sleth.ko file
- edit the linux shell script insmod.sh with the instructions given in the file
- run insmod.sh
- type ifconfig and see the status
- additional status output is available via dmesg. Search for entries with „sleth“

Optimizing performance – changing the number of RX/TX buffers

- Performance can be optimized regarding the number of buffers for RX and TX. sleth.h allows to define this number of buffers for RX and TX and must be a power of 2. It is important, that the address FIFO within the FPGA has the depth to store all buffer start addresses. Therefore the FPGA designer has to be make sure, that the parameter/generic scatter_gather_fifo_depth_g which defines the log2 of the fifo depth, is the same as SLETH_RX_BUFFER_COUNT / SLETH_TX_BUFFER_COUNT within sleth.h
- Valid settings for the two defines are currently : 64 or 128
- Make sure, that the constant Image_Format_fifo_depth_c (Line 177 in dma_pkg.vhd) is scatter_gather_fifo_depth_g + 1
- The type of the address FIFO can be either distributed/MLAB RAM or block RAM. The FPGA designer can select the type with the parameter/generic scatter_gather_fifo_bram_g. In most cases Block RAM is the right selection. Distributed/MLAB makes only sense for a single ethernet link where the parameter/generic scatter_gather_fifo_depth_g is equal or less than 7
- For more details on the parameters/generics above see the HCC IP core user guide