



IFI Nios[®]II GMACII

User Guide

Core Version:	2009.09
Document Version:	2010.01 rev 9.1
Document Date:	01 2010

IFI GMACII

- High Performance Gigabit Ethernet MAC
 - Up to 114 MByte/s UDP Data
- Easily integrated into Nios II systems using SOPC Builder
- Avalon interface for Nios II processor
- Independent clock domains for Nios II and GMACII
- Royalty free
- Reference Software shipped with IP
- Verified on Nios II development boards
- Gigabit Ethernet Phy module available
 - (only from www.devboards.de)
- Evaluation version available (OCP Open Core Plus)

IFI GMACII

- Jumboframe support (compiletime parameter)
 - Receivebuffer 4 (standard),8,16,32,64 or 128 kByte (ringbuffer)
 - Transmitbuffer 2*2 (standard), 2*4, 2*8, 2*16, 2*32, 2*64 kByte (double buffer)
- Transmitbuffer-Readback for easy software debugging
- Automatic frameextension to meet the 64Bytes minimum frame length
- Multicast support with separate MAC_ID and IP filters
- IGMP filter
- All filters can be switched off
- DMA masteraddress support for no increment
- Statusbits receivebuffer overrun, CRC-error
- Selectable PHY interface
 - GMII/MII, MII, RGMII, RMII
- Automatic SDC file generation for TimeQuest

Contents

- *Overview*
- *Install*
- *Integrating the Core using SOPC Builder*
- *Reference Designs*
- *Using the Core without Nios*
- *Necessary Assignments*
- *Detailed Information*
- *Ethernet Background*
- *Revision History*
- *License Agreement*

Overview

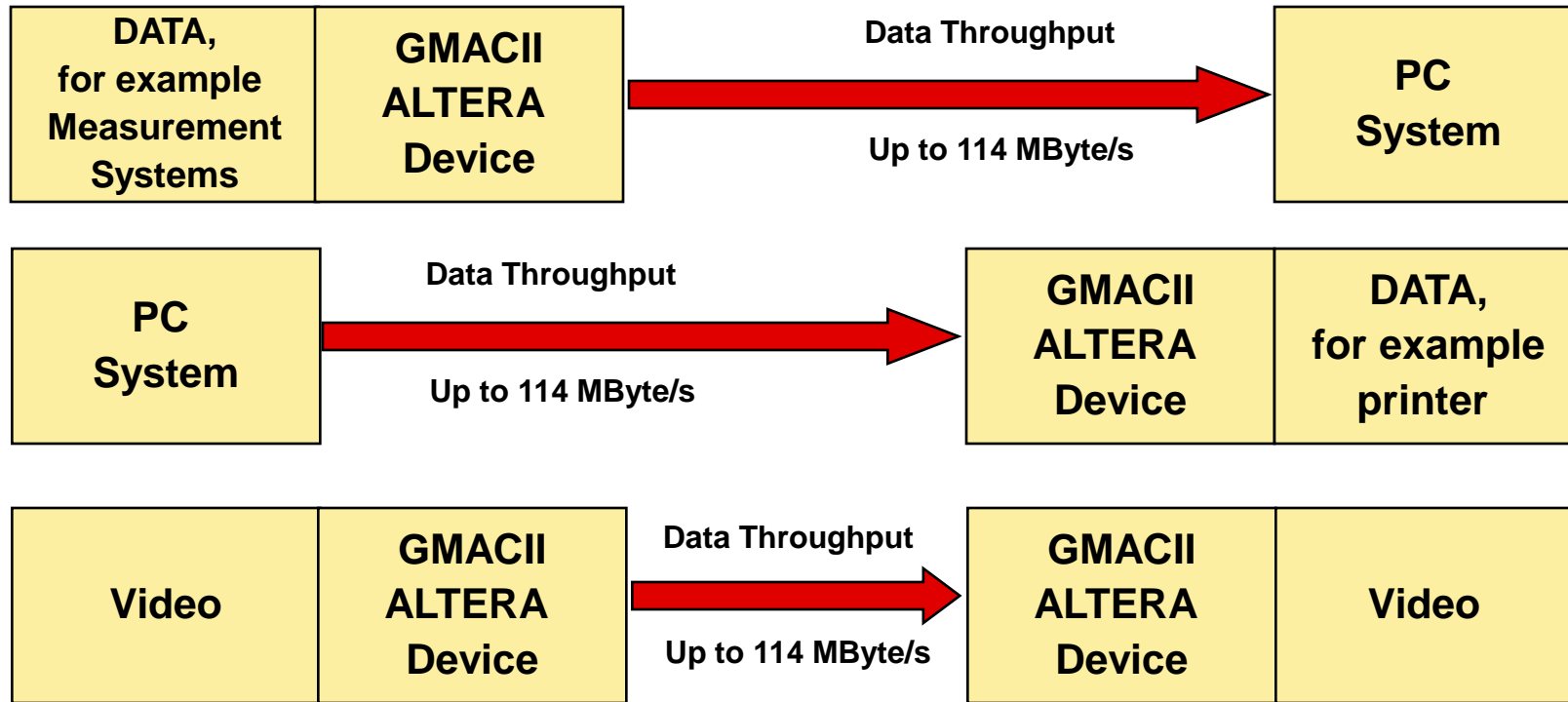
- *Brief Description*
- *Suitable Applications*
- *Block Diagram*
- *Feature List*
- *Altera Implementation*
- *Contacting Technical Support*

Brief Description

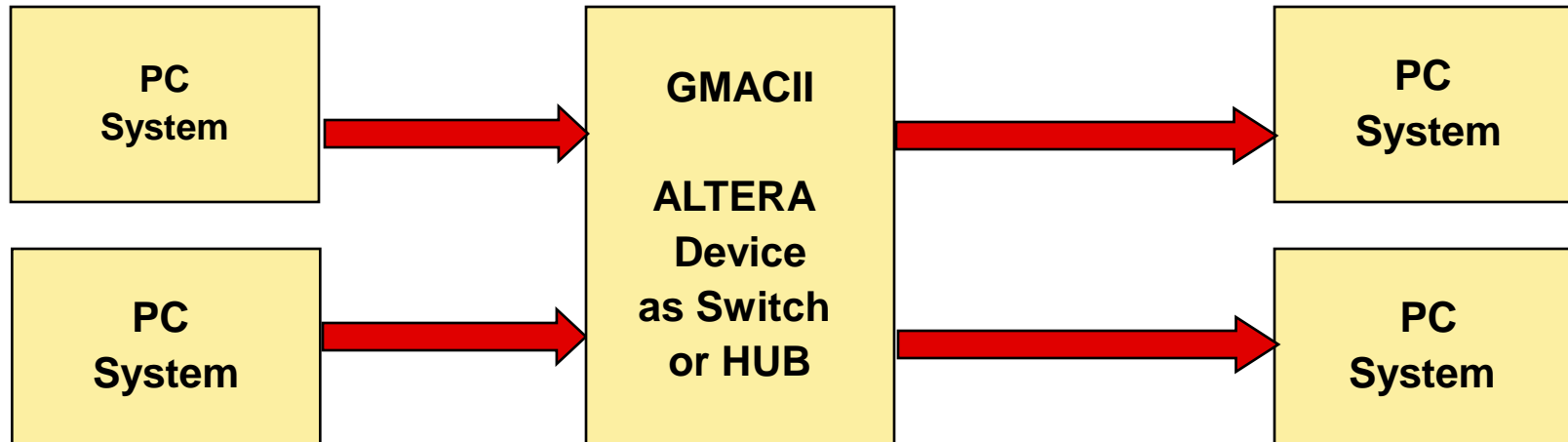
■ IFI_GMACII

- This IP combines the advantages of the softwareflexibility with the high performance of a hardware solution
 - Advantages: medium size, software controlled, high performance

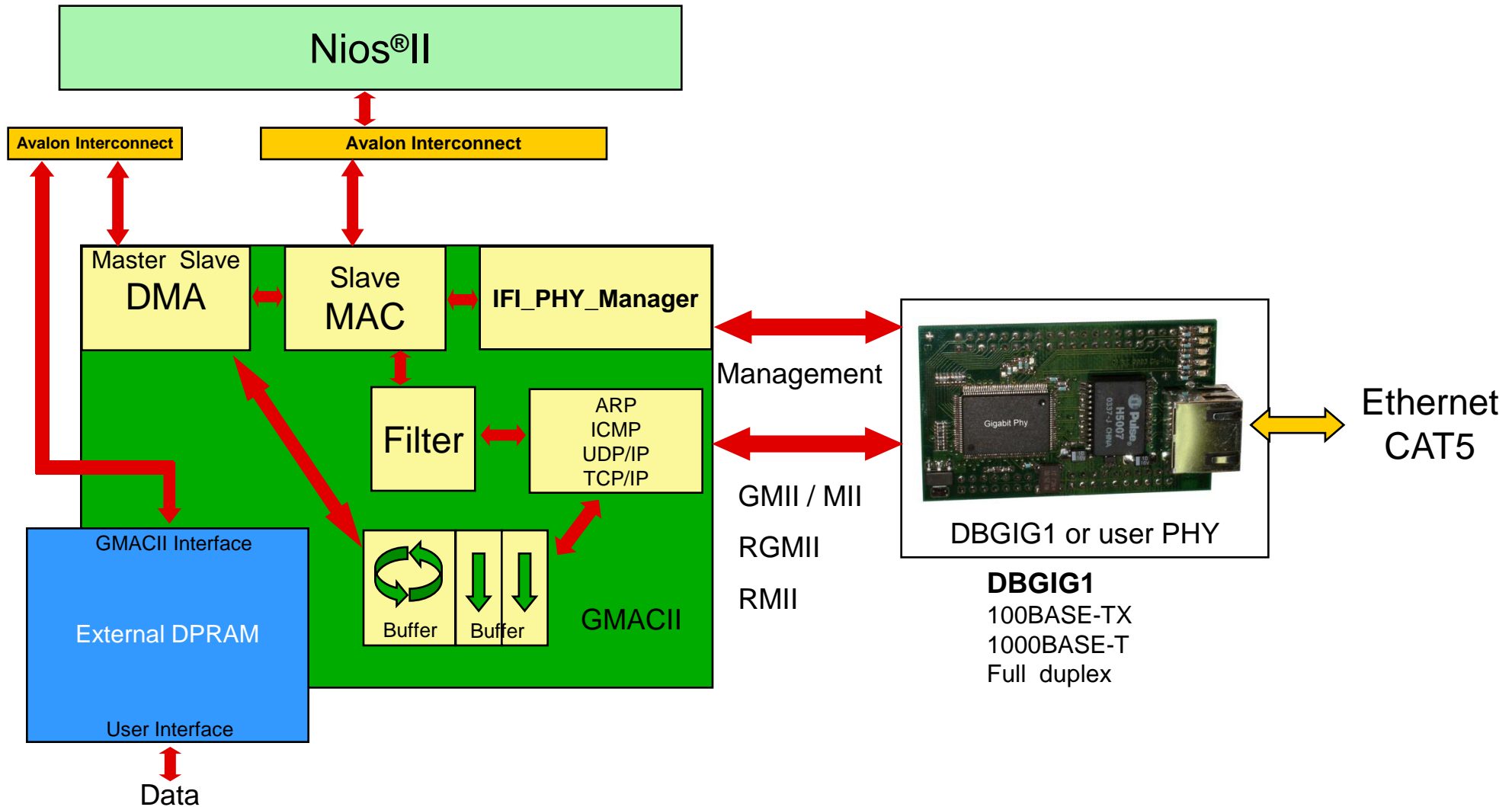
Suitable Applications for GMACII



Not Suitable Applications for GMACII



Block Diagram



IFI GMACII Feature List

- 1000 Base-T
 - Full duplex
- 100 Base-TX
 - Full duplex
- Filter
 - MAC ID
 - MAC IP
- Integrated DMA controller
 - uses pipeling on both ends
 - generates checksum on the fly
 - alignment aware
- IFI_PHY_Manager included

Standard buffers

- Transmitbuffer
 - Double buffer
 - 2000 Byte each
 - checksum advance logic
- Receivebuffer
 - Ring buffer
 - 4 kByte total

With buffers for jumbo frames

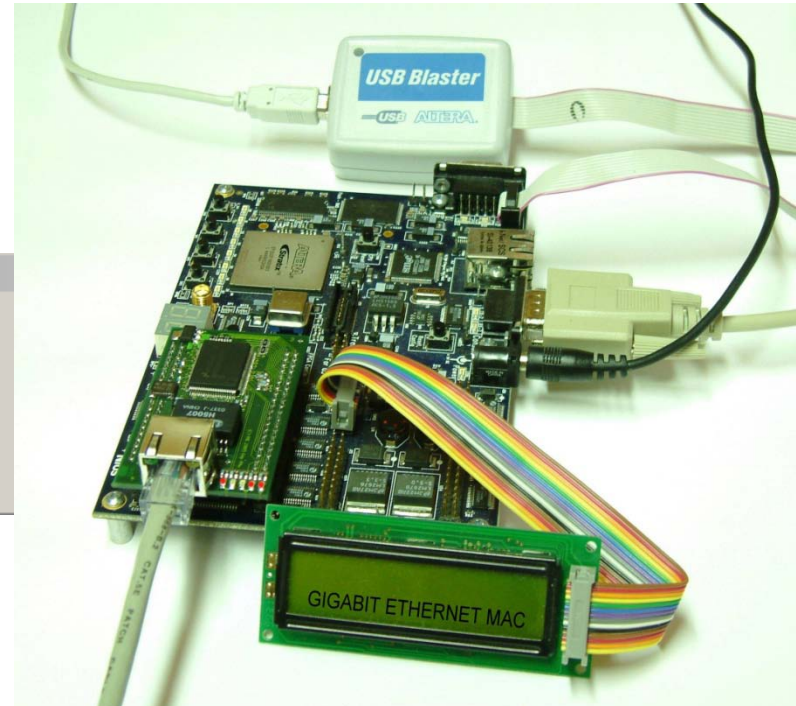
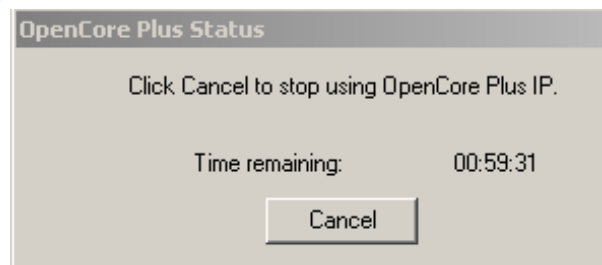
- Transmitbuffer
 - Double buffer with parametrized size
 - 2*2kB, 2*4kB, 2*8kB, 2*16kB, 2*32kB, 2*64kB
 - checksum advance logic
- Receivebuffer
 - Ring buffer with parametrized size
 - 4kB, 8kB, 16kB, 32kB, 64kB, 128kB

IFI GMACII Implementation

- Design Flows supported
 - SOPC Builder
 - Encrypted VHDL
- Software examples
- Device families supported
 - all CYCLONE, STRATIX and ARRIA families
- Device resource utilization
 - about 3000 LEs for CYCLONEIII
 - RAM: 8 M9K Blocks (standard buffers)

OpenCore Plus Feature

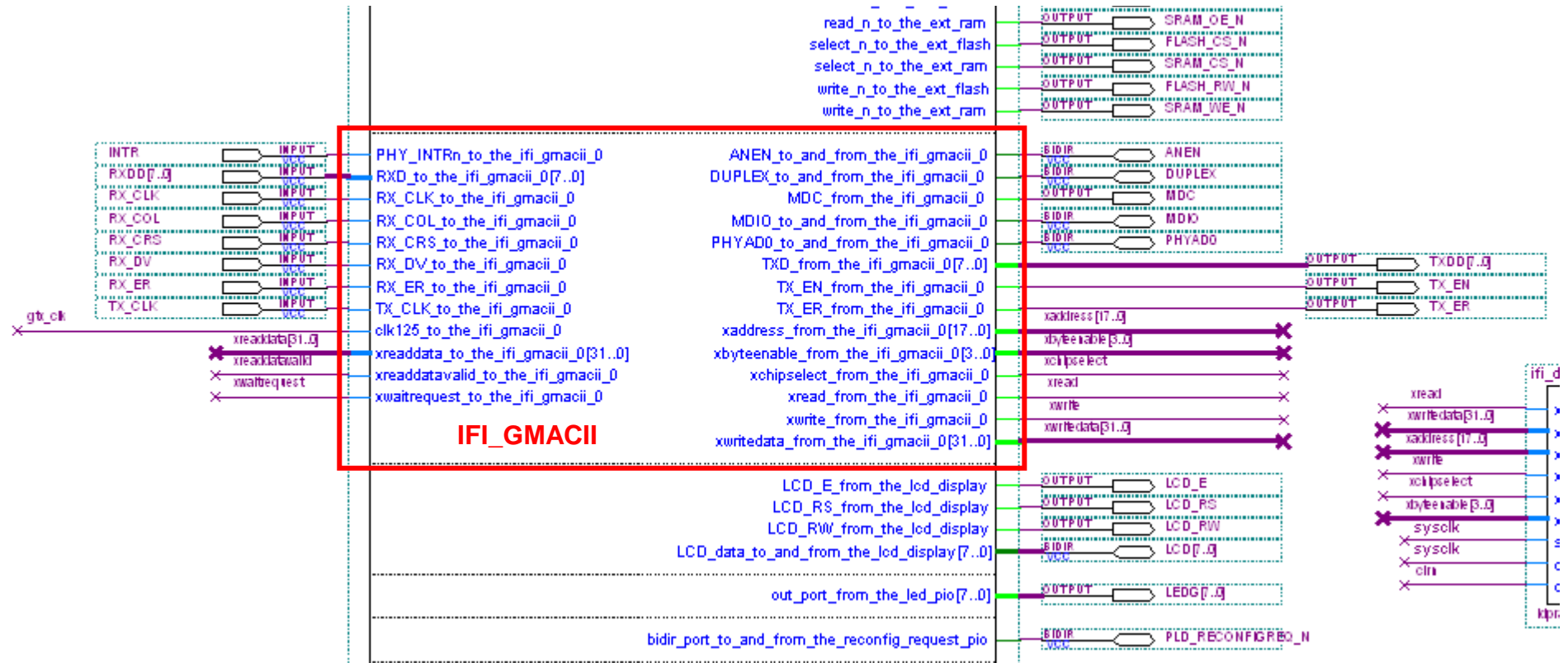
- Evaluate the IFI GMACII on your board or the Altera Nios II development boards
 - Stand alone evaluation times out after ~1 hour
 - If there is a connection between the device and the Quartus programmer the evaluation time is unlimited.



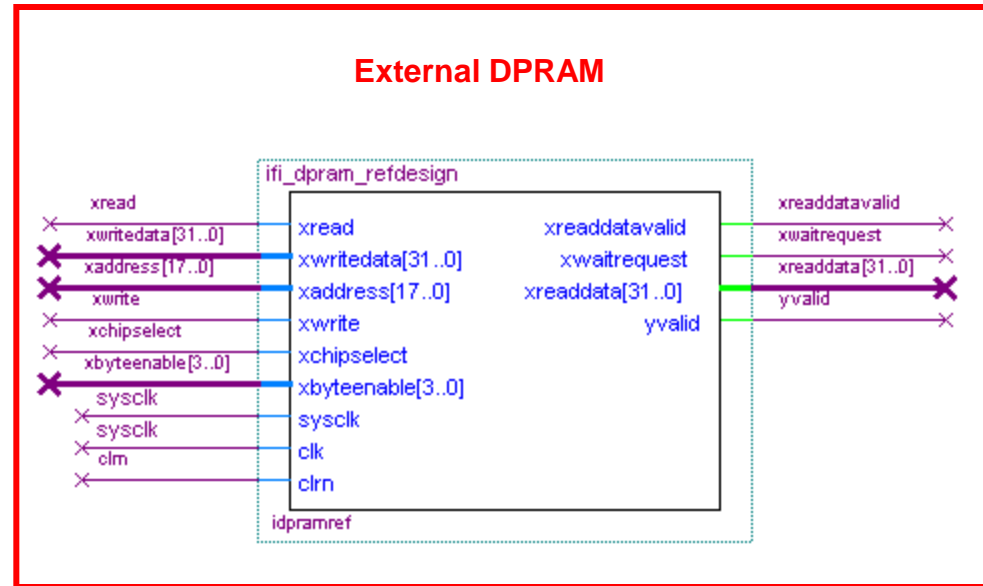
IFI GMACII Reference Design

- CycloneII 2C35 Development Board Reference Design
 - With GMII Interface
 - For the DBGIG1 Gigabit Ethernet Phy Module
 - National Semiconductor DP83865 GigPhy
 - 100Mb/Gigabit capability
 - More information: www.devboards.de
- DBC3C40 Development Board Reference Design
 - With RMII Interface
 - For 2 National Semiconductor DP83848 Phyter (100 Mb)
 - More information: www.devboards.de
- CycloneIII 3C120 Development Board Reference Design
 - With RGMII Interface
- CycloneIII 3C25 NEEK Reference Design
 - With MII Interface
- DBM3CXXX Development Board Reference Design
 - With GMII Interface
 - More information: www.devboards.de

IFI GMACII Reference Design



IFI GMACII Reference Design



- This example is included as source code
- It can be used as starting point for the user data interface

IFI GMACII Pricing

- Encrypted Netlist node locked: 5000 EURO
- Encrypted Netlist Floating License: 6250 EURO
- Additional node locked (same location)
 - License: 1250 EURO
- 1 year maintenance included
- Maintenance: 10 % from the netlist-price / year

- Licensing:
 - Unlimited T-Guard License
 - Multiproject
 - Royalty Free

IFI GMACII Verification

■ Hardware Tested on

- Cyclone/II/III Nios II Development Kits
- StratixII Nios II Development Kit
- DBC2C20/DBC3C40 Cyclone Development Boards

■ Tested with different PHYs

- National Semiconductor DP83865 GigPhy (100 Mb/Gigabit)
- National Semiconductor DP83847 DsPhyter (100 Mb)
- National Semiconductor DP83848 Phyter (100 Mb)
- National Semiconductor DP83640 Phyter (100 Mb)
- Marvell 88E1111 (100 Mb/Gigabit)

Contacting Technical Support

- Although we have made every effort to ensure that this SOPC Builder Ready OpenCore Package works correctly, there might be problems that we have not encountered.
- For questions about the core's features, functionality, and parameter settings please contact:

IFI Ingenieurbüro Für Ic-Technologie
P. Riekert & F. Sprenger
Kleiner Weg 3 -- 97877 Wertheim -- Germany
Phone: (+49)9342/96080
E-Mail: ifi@ifi-pld.de
<http://www.ifi-pld.de>

Install

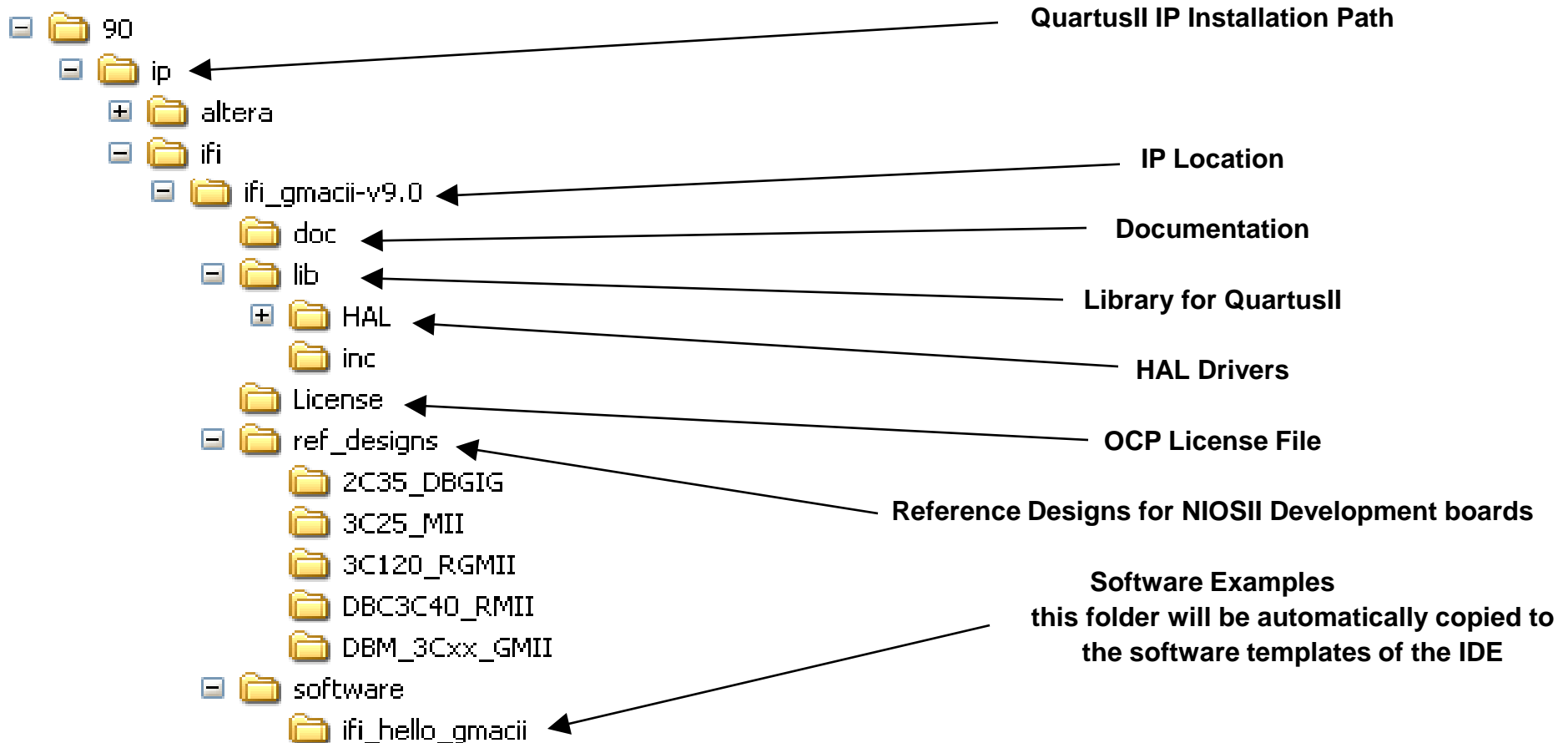
- *How to install*
- *SOPC Builder Ready OpenCore Package*
- *Software Examples Installation*
- *Install the Driver Library*
- *Licensing*
- *Set up Licensing*

Install the IFI GMACII

- Before you can start using Altera IFI GMACII functions, you must install the IFI GMACII files on your computer. The following instructions describe this process for the IFI GMACII.
- **Close Quartus and IDE.**
- **The installed QuartusII version must be 9.1 or newer**
- **Install the IFI GMACII Files**
 - The following instructions describe how you install IFI GMACII on computers running the Windows operating system.
 - If you don't change the installation path, the SOPC Builder and the Megawizard will find the IP automatically
 - Windows
 - Follow these steps to install the IP on a PC running a supported version of the Windows operating system:
 - Choose Run (Windows Start menu).
 - Type <path name>\<filename>.exe, where <path name> is the location of the downloaded IP function and <filename> is the filename of the IP function.
 - Click OK. The IP Installation dialog box appears. Follow the on-screen instructions to finish installation.
 - Solaris & Linux
 - Follow these steps to install the IP on a computer running supported versions of the Solaris and Linux operating systems:
 - Decompress the package by typing the following command:
 - `gzip -d<filename>.tar.gz`
 - where <filename> is the filename of the IFI NIOSII Advanced CAN function.
 - Extract the package by typing the following command:
 - `tar xvf <filename>.tar`

SOPC Builder Ready OpenCore Package

The SOPC Builder Ready OpenCore Package contains all files required for plug-and-play integration of this core into Altera's SOPC Builder tool, allowing the user to easily evaluate the core within his Avalon-based system. (example screenshot from older version)



Licensing

■ OpenCorePlus License

This package is shipped with a OpenCorePlus license,
<Core installation directory>\license\license_GMACII_OCP.dat.

When the FEATURE line from this license is appended to the user's Quartus II license file, the encrypted VHD file can be read into Quartus II and place and route can be performed.

The license permit generation of <revision_name>_time_limited.sof files.

The hardware evaluation feature will run during you have an established connection between your board and the QuartusII programmer (do not close the programmer, else the core stops immediate).

If you remove the connection it will stop working after 1 hour.

(Refer to the messages created by the programmer)

■ Full License

If you purchased a FULL LICENSE you receive an additional license file,
license_???.dat.

Use this instead of the license_GMACII_OCP.dat. When the FEATURE line from this license is appended to the user's Quartus II license file, the encrypted VHD file can be read into Quartus II and place and route can be performed. The license permit generation of <revision_name>.pof files and gate-level simulation netlists.

- One FEATURE line can span more than one line

Set Up Licensing

- To install your license, you can either append the license to your **license.dat** file or you can specify the IFI GMACII 's **license_GMACII_ocp.dat** file in the Quartus II software.
 - Before you set up licensing for the IFI NIOSII GMACII , you must already have the Quartus II software installed on your computer with licensing set up.
- **Append the License to Your license.dat File**
 - To append the license, follow these steps:
 - Open the IFI GMACII license file in a text editor.
 - Open your Quartus II **license.dat** file in a text editor.
 - Copy all lines from the license file and paste it into the Quartus II license file.
 - Do not delete any FEATURE lines from the Quartus II license file.
 - Save the Quartus II license file.
 - When using editors such as Microsoft Word or Notepad, ensure that the file does not have extra extensions appended to it after you save (e.g., **license.dat.txt** or **license.dat.doc**). Verify the filename in a DOS box or at a command prompt. Also, make sure that the file is saved in plain-text format without formatting characters.
- **Specify the License File in the Quartus II Software**
 - To specify the IFI GMACII license file in Quartus II, follow these steps:
 - Altera recommends that you give the file a unique name, e.g., *<core name>*_license.dat.
 - Run the Quartus II software.
 - Choose **License Setup** (Tools menu). The **Options** dialog box opens to the **License Setup** page.
 - In the **License file** box, add a semicolon to the end of the existing license path and filename.
 - Type the path and filename of the IFI GMACII function license file after the semicolon.
 - Do not include any spaces either around the semicolon or in the path/filename.
 - Click **OK** to save your changes.

Integrating the Core using SOPC Builder

- *Prerequisites*
- *Adding the Core to your System*
- *Using IPToolBench*
- *About*
- *Documentation*
- *Add/Update Component*

Integrating the Core with your System using SOPC Builder

- This section contains instructions on the following:
 - Adding the Core to your System
 - Running the Reference Design
- These instructions assume that the user is familiar with the
 - Altera OpenCore evaluation process,
 - Altera's Quartus II development software,
 - and the Altera SOPC Builder tool.
- For more information on these prerequisites, please visit www.altera.com.

Adding the Core to your System

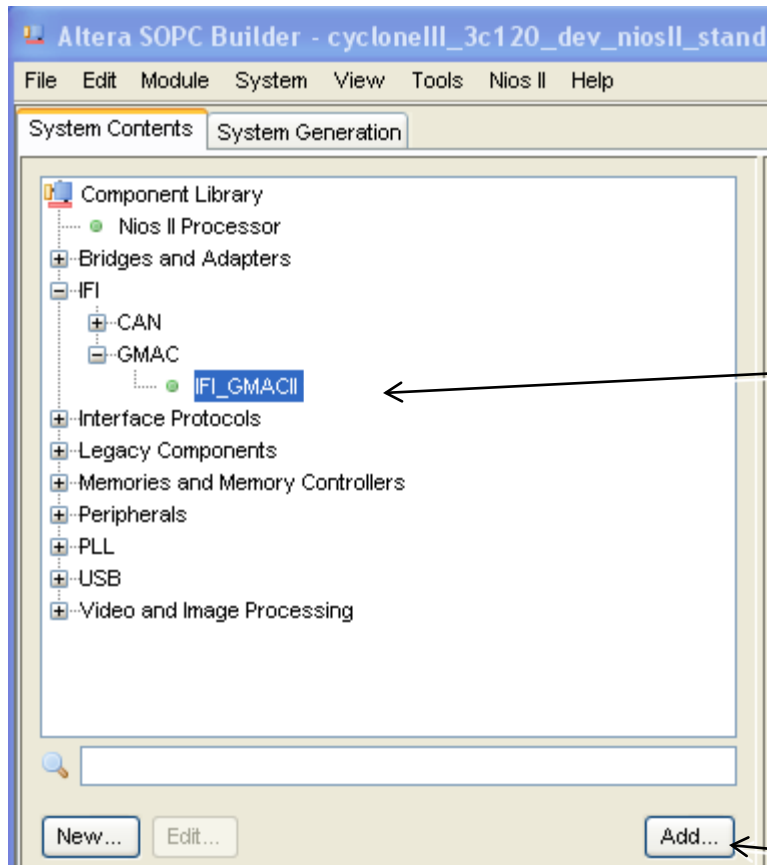
■ This walkthrough involves the following steps:

- Create a New Quartus II Project
- Create a New SOPC Builder Design
- Launch IP Toolbench
 - Step 1: Generate

■ Create a New Quartus II Project

- Before you begin, you must create a new Quartus II project. With the New Project wizard, you specify the working directory for the project, assign the project name, and designate the name of the top-level design entity. You will also specify the IFI GMACII user library. To create a new project, follow these steps:
- Choose **Programs > Altera > Quartus II <version>** (Windows Start menu) to run the Quartus II software.
- Choose **New Project Wizard** (File menu).
- Click **Next** in the introduction (the introduction will not display if you turned it off previously).
- Specify the working directory for your project. This walkthrough uses the directory **c:\qdesigns\myproject**.
- Specify the name of the project. This walkthrough uses **myproject**.
- Click **Next**.
- Click **User Libraries...**
- Type **<path>\IP_CORE_DIR-v<version>\lib** into the **Library name** box, where **<path>** is the directory in which you installed the IFI GMACII .
- Click **Add**.
- Click **OK**.
- Click **Next**.
- Choose the target device family in the **Family** list.
- Click **Finish**.
- You have finished creating your new Quartus II project.

Adding the Core to your SOPC System



- Launch SOPC Builder from Quartus II (Tools menu).

- Select the core by clicking on the core name

- Click "Add" to add the core to your system.

IFI_GMACII v9.1 - ifi_gmacii_0

IFI
P. Riekert & F. Sprenger

IFI_GMACII v9.1
ifi_gmacii

Info

Block Diagram

GMACII Timer Information
System Clock in MHz: 75

Transmitbuffer Read
Transmitbuffer Read: OFF

DPRAM Interface
DPRAM Interface: USED

Advanced Features
Advanced Features: OFF
Select Receive Buffersize(Ringloutter): 4_kByte
Select Transmit Buffersize(Doublebuffer): 2*2_kByte

Phy Interface
Select Phy Interface type:: GMII

Phy Speed
Select Phy Speed:: 100/1000

Show Phy support Pins
Show Phy support Pins: OFF

External PHY Timing
Select the external PHY Timing:: DEFAULT

Default External PHY Timing
Setup time in ns: 2.0
Hold time in ns: 0.0
Maximum Tco time in ns: 5.5
Minimum Tco time in ns: 0.5

User Defined External PHY Timing
Setup time in ns: 0.0
Hold time in ns: 0.0
Maximum Tco time in ns: 0.0
Minimum Tco time in ns: 0.0

Info: ifi_gmacii_0: DPRAM interface enabled.
Info: ifi_gmacii_0: Advanced Features Disabled
Info: ifi_gmacii_0: Transmitbuffer Read Disabled
Info: ifi_gmacii_0: GMII/MII Interface automatic switching

Cancel Finish

Info + Documentation

Parameters

Information

IFI_GMACII v9.1 - ifi_gmacii_0

IFI
P. Riekert & F. Sprenger

IFI_GMACII v9.1
ifi_gmacii

Info

Block Diagram

GMACII Timer Information

System Clock in MHz: 75

Transmitbuffer Read

Transmitbuffer Read: OFF

DPRAM Interface

DPRAM Interface: USED

Advanced Features

Advanced Features: OFF

Select Receive Buffersize(Ringbuffer): 4_kByte

Select Transmit Buffersize(Doublebuffer): 2*2_kByte

Phy Interface

Select Phy Interface type: GMII

Phy Speed

Select Phy Speed: 100/1000

Show Phy support Pins

Show Phy support Pins: OFF

External PHY Timing

Select the external PHY Timing: DEFAULT

Default External PHY Timing

Setup time in ns: 2.0

Hold time in ns: 0.0

Maximum Too time in ns: 5.5

Minimum Too time in ns: 0.5

User Defined External PHY Timing

Setup time in ns: 0.0

Hold time in ns: 0.0

Maximum Too time in ns: 0.0

Minimum Too time in ns: 0.0

Info: ifi_gmacii_0: DPRAM interface enabled.
Info: ifi_gmacii_0: Advanced Features Disabled
Info: ifi_gmacii_0: Transmitbuffer Read Disabled
Info: ifi_gmacii_0: GMII/MII Interface automatic switching

Cancel Finish

IFI_GMACII v9.1 Info

IFI_GMACII v9.1

Class Name ifi_gmacii

Version 9.1

Author IFI

Description IFI GMACII v9.1

Group IFI/GMAC

Data Sheet file:///D:/altera/91/ip/ifi/ifi_gmacii-v9.1/doc/IFI_GMACII_docu.pdf

GMACII Timer Information

System Clock in MHz System Clock in MHz (used for computing the performance only)

Transmitbuffer Read

Transmitbuffer Read Transmitbuffer-Readback for easy software debugging

DPRAM Interface

DPRAM Interface Use of the Dualport RAM Interface

Advanced Features

Advanced Features Use of the Advanced Features

Select Receive Buffersize(Ringbuffer) Select the Receive Buffersize(Ringbuffer)

Select Transmit Buffersize(Doublebuffer) Select Transmit Buffersize(Doublebuffer)

Phy Interface

Select Phy Interface type: Selection of the Interface typ (GMII/MII, MII, RGMII, RMII)

Phy Speed

Select Phy Speed: Selection of the PHY Speed (100/1000, 100, 1000)

Show Phy support Pins

Show Phy support Pins Use of the PHY support pins

IFI_GMACII v9.1 - ifi_gmacii_0

IFI
P. Riekert & F. Sprenger

IFI_GMACII v9.1
ifi_gmacii

Info

Block Diagram

```

graph LR
    ifi_gmacii_0[ifi_gmacii_0]
    clock --> gmacii_slave_clock[gmacii_slave_clock]
    conduit --> gmacii_slave_export[gmacii_slave_export]
    conduit --> gmacii_slave_import[gmacii_slave_import]
    conduit --> gmacii_slave_irq[gmacii_slave_irq]
    conduit --> dpram_slave_export[dpram_slave_export]
    conduit --> dpram_slave_import[dpram_slave_import]
    gmacii_slave_clock --> gmacii_master[gmacii_master]
    gmacii_slave_export --> gmacii_master
    gmacii_slave_import --> gmacii_master
    gmacii_slave_irq --> gmacii_master
    dpram_slave_export --> gmacii_master
    dpram_slave_import --> gmacii_master
    gmacii_master --> avalon[avalon]
  
```

GMACII Timer Information

System Clock in MHz: 75

Transmitbuffer Read

Transmitbuffer Read: OFF

DPRAM Interface

DPRAM Interface: USED

Advanced Features

Advanced Features: OFF

Select Receive Buffersize(Ringbuffer): 4_kByte

Select Transmit Buffersize(Doublebuffer): 2*2_kByte

Phy Interface

Select Phy Interface type: GMII

Phy Speed

Select Phy Speed: 100/1000

Show Phy support Pins

Show Phy support Pins: OFF

External PHY Timing

Select the external PHY Timing: DEFAULT

Default External PHY Timing

Setup time in ns: 2.0

Hold time in ns: 0.0

Maximum Tco time in ns: 5.5

Minimum Tco time in ns: 0.5

User Defined External PHY Timing

Setup time in ns: 0.0

Hold time in ns: 0.0

Maximum Tco time in ns: 0.0

Minimum Tco time in ns: 0.0

Info: ifi_gmacii_0: DPRAM interface enabled.
Info: ifi_gmacii_0: Advanced Features Disabled
Info: ifi_gmacii_0: Transmitbuffer Read Disabled
Info: ifi_gmacii_0: GMII/MII interface automatic switching.

Cancel Finish

- Selected system frequency
- Transmit Buffer Readback for easy software debugging(yes/no)
- Dualport RAM Interface(yes/no)
- Advanced Features(yes/no)
- Receive Buffer Size(Ringbuffer)
- Transmit Buffer Size(Doublebuffer)
- PHY Interface(GMII/MII, MII, RGMII, RMII)
- PHY speed(100/1000, 100, 1000)
- Show the PHY support pins (yes/no)
- External PHY Timing
- Use DEFAULT/USER_DEFINED)
- Setup time in ns
- Hold time in ns
- Maximum Tco in ns
- Minimum Tco in ns

Adding the Core to your System

The screenshot shows the Altera SOPC Builder interface. The 'System Components' window is open, displaying a tree view of components on the left, a detailed description of selected components in the middle, and a table of component parameters on the right. The components listed are 'onchip_ram', 'onchip_memory_dpr...', and 'ifi_gmacii_0'. The 'ifi_gmacii_0' component is selected, showing its description as 'IFI_GMACII' and its parameters, including 'clk' and 'dpram_slave'. The bottom of the window shows a status bar with information about the 'ext_flash' and 'onchip_ram' components, and a 'Generate' button.

Component	Parameter	Value
onchip_ram	clk	0x00000000
	clk	0x00000000
onchip_memory_dpr...	clk	0x00044000
	clk	0x00044000
ifi_gmacii_0	clk	0x00040000
	dpram_slave	0x00200000

Info: **ext_flash**: Flash memory capacity: 64,0 MBytes (67108864 bytes).
 Info: **onchip_ram**: User is required to provide memory initialization files for memory .
 Info: **ifi_gmacii_0**: DPRAM interface enabled.
 Info: **ifi_gmacii_0**: Advanced Features Disabled

- Specify desired instance name, base address, and IRQ.
- Connect your Avalon interfaces as necessary
- Add additional components as required by your design.
- Complete system generation as described in the Altera SOPC Builder documentation.

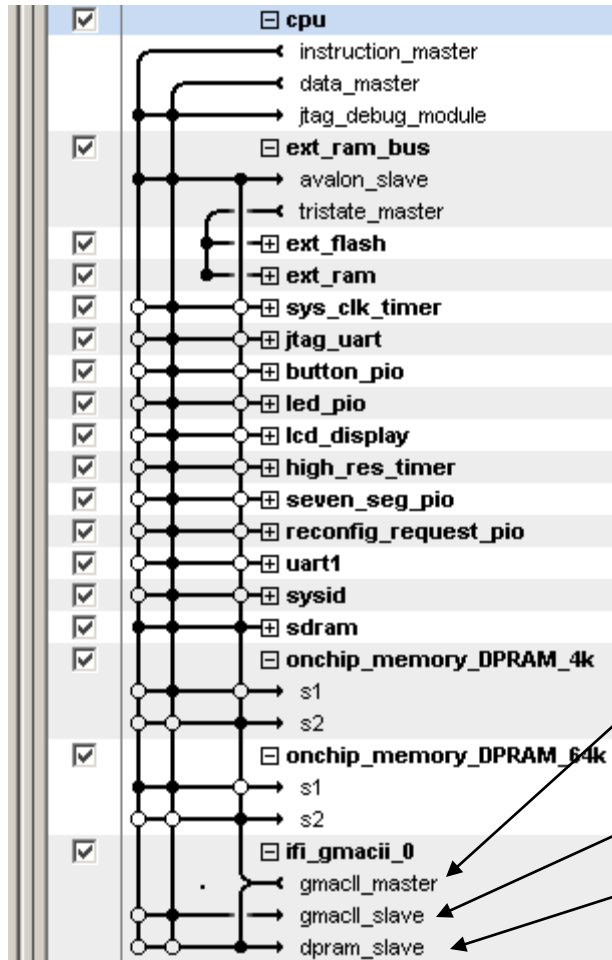
Reference Designs

- *Running a Reference Design*
- *Creating a Software Project*
- *Run a Hardware Configuration*

Running a Reference Design

- Start Quartus II, version 9.0 or higher.
- Open the Quartus II project <Core installation directory>\reference_designs\xxx\IFI_GMACII_Reference_design.qpf
- Launch SOPC Builder from Quartus II (Tools menu).
- The Nios CPU has been parameterized and added to the system for you, as have the program and data memories, JTAG_UART and the core itself.

SOPC Connections



GMACII_master is master for

- onchip_memory_DPRAM_64k slave s2
- onchip_memory_DPRAM_4k slave s2
- sdram slave
- ext_ram_bus slave
- dpram_slave

GMACII_slave is mastered by CPU
data_master

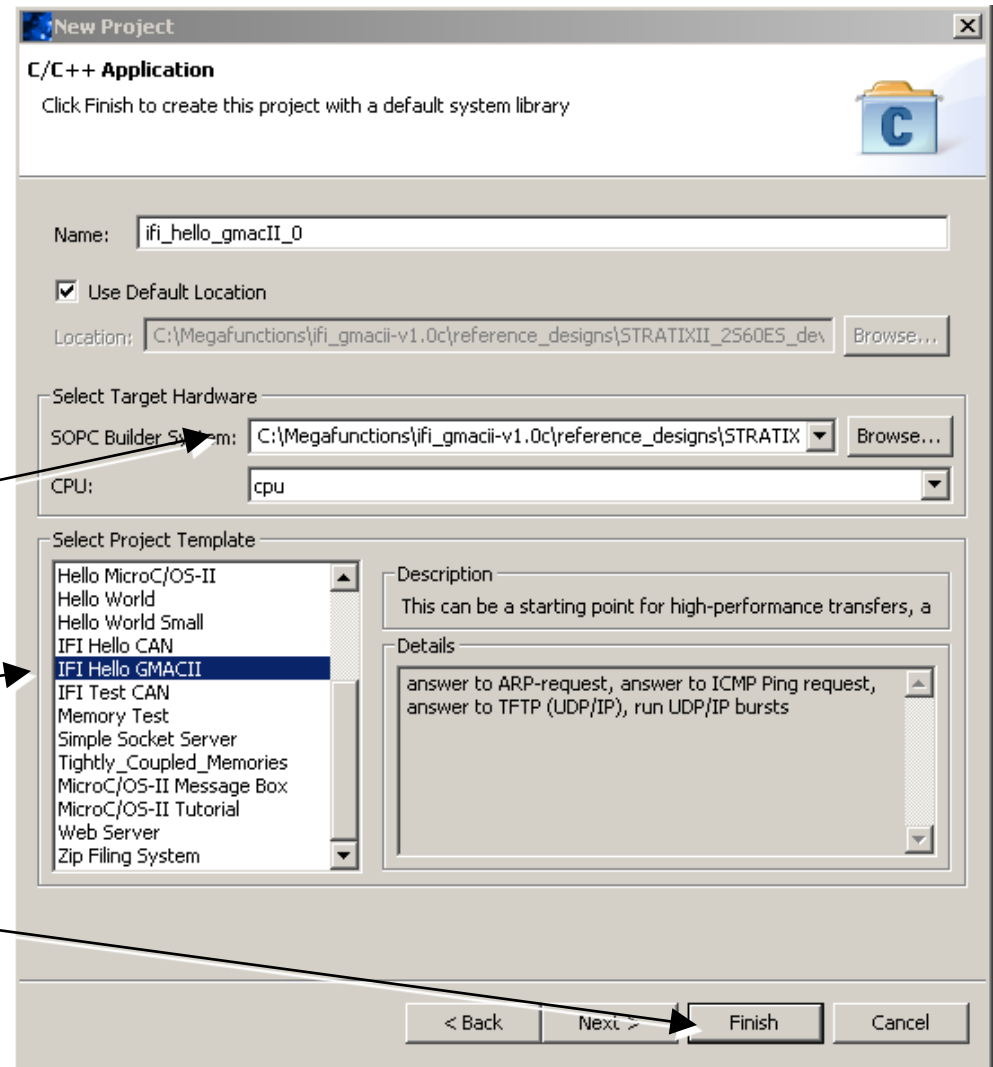
dpram_slave is mastered by
GMACII_master

Running a Reference Design

- Click "Generate" to generate the HDL files.
- Click "Exit" to go back to Quartus and compile the design.
- Launch the IDE for creation of software projects or InstructionSetSimulation.

Creating a Software Project

- File → New → Project
 - Select C/C++ Application
 - Click Next

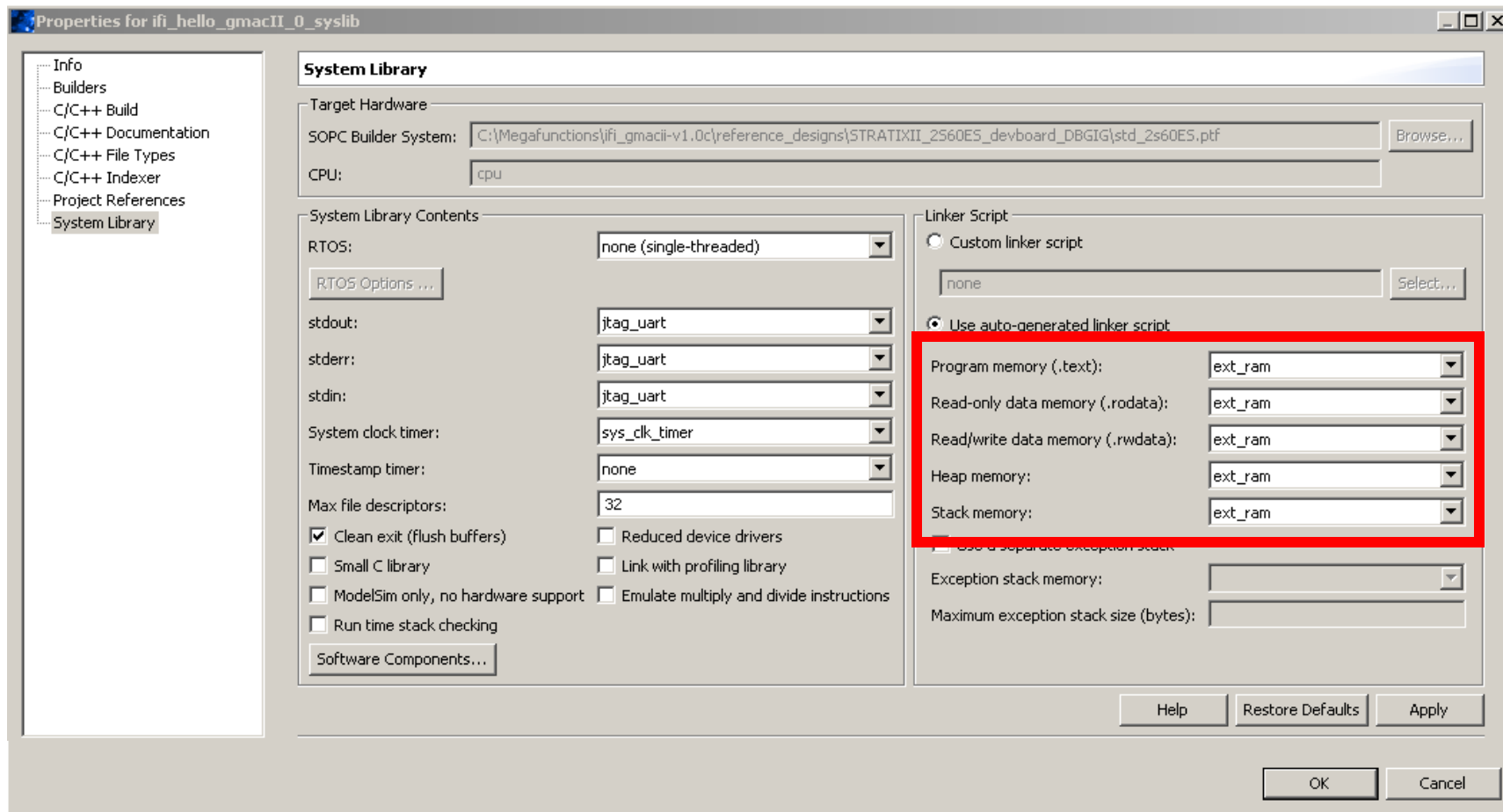


Select the PTF of your project

Select Project Template

Click on Finish

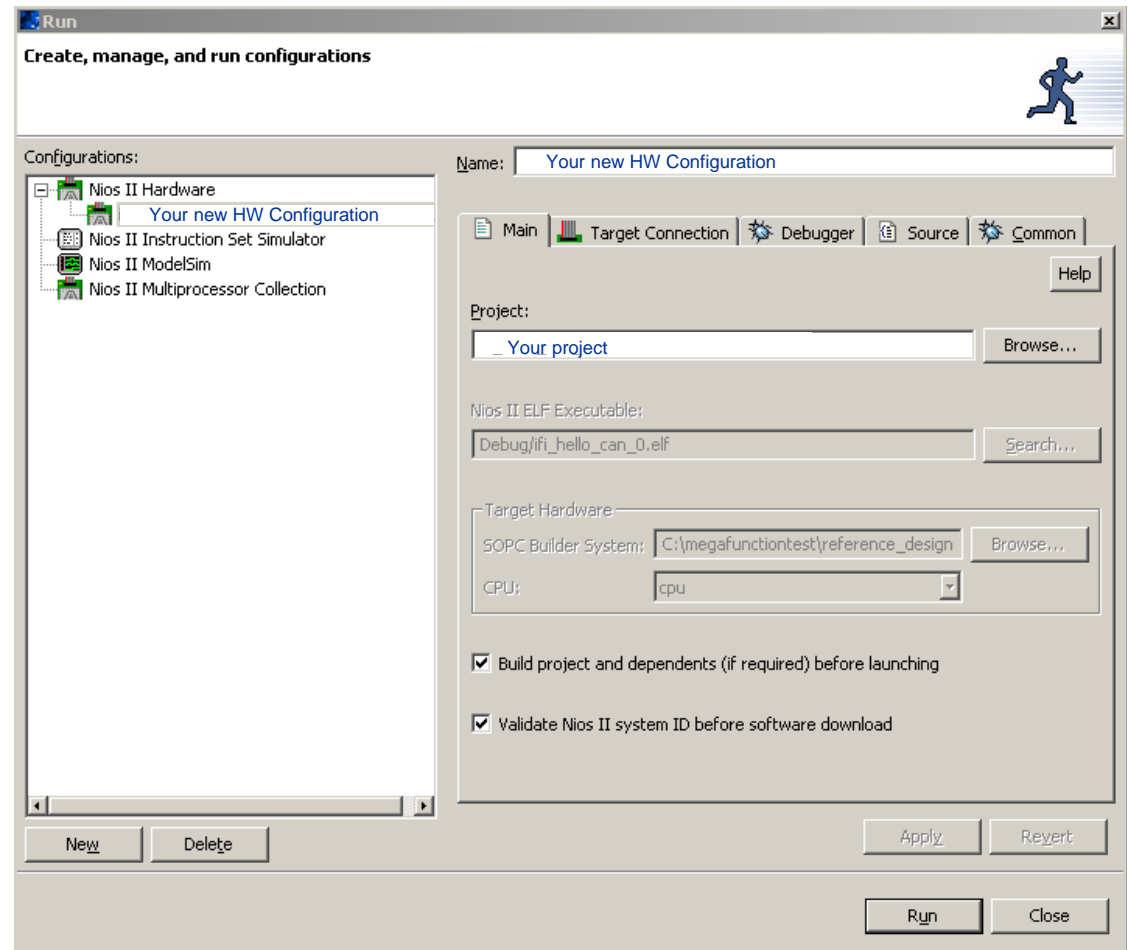
Syslib Settings



- Change the memory settings to any wished RAM which is big enough and fast enough

Run a Hardware Configuration

- Select your Project within the C/C++ Projects View
- Run → Run..
- Select NiosII Hardware
- Click on New
- Click on Run



Port description

Portname	Direction	Usage	Description
clk125	input	External	125 MHz clock
TX_CLK	input	External	TX Clock from PHY
RX_CRS	input	External	RX_CRS from PHY
RX_COL	input	External	RX_COL from PHY
RX_CLK	input	External	RX Clock from PHY
RX_DV	input	External	RX_DV from PHY
RX_ER	input	External	RX_ER from the PHY
RXD[7..0]	input	External	RXD from the PHY
TX_EN	output	External	TX_EN to the PHY
TX_ER	output	External	TX_ER to the PHY
TXD[7..0]	output	External	TXD to the PHY
PHY_INTRn	input	External	Interrupt input (low active)
DUPLEX	bidir	External	Connect to PHY (optional)
PHYADO	bidir	External	Connect to PHY (optional)
ANEN	bidir	External	Connect to PHY (optional)
MDC	output	External	Connect to PHY
MDIO	bidir	External	Connect to PHY

Port description

Portname	Direction	Usage	Description
xreaddata[31..0]	input	External	External DPRAM readdata
xreaddatavalid	input	External	External DPRAM readdatavalid
xwaitrequest	input	External	External DPRAM waitrequest
xaddress[17..0]	output	External	External DPRAM address
xbyteenable[3..0]	output	External	External DPRAM byteenable
xchipselect	output	External	External DPRAM chipselect
xread	output	External	External DPRAM read
xwrite	output	External	External DPRAM write
xwritedata[31..0]	output	External	External DPRAM writedata

Tip

- when having no PHY-Board available, you can test the GMACII transmitter with STP (Signaltap) when connecting the clk125 also to the RX_CLK port

Necessary Assignments

- *Assumptions*

Assumptions

- Depending on the used PHY Interface
- You have to provide a 125MHz Clock
 - This 125MHz clock is required in 100Mb and 1Gb mode
 - Use a PLL within the device
 - The external oscillator has to be better than 100ppm frequency deviation
 - An example of this can be found in the reference designs
- For the RGMII interface you have to provide an additional 125MHz Clock with a 90° Degrees shift

Necessary Assignments

- The necessary timing assignments are automatically written in SDC files for you.
- You have to provide a user SDC file which contains the clock settings for the system
- You have to activate TimeQuest Timing analysis and to add your user SDC as the first file and then the automatically generated ifi_gmacii_XXXXX.sdc files.

Detailed Information

- *Address map standard buffers*
- *Address map jumbo buffers*
- *Registers*
- *DMA*
- *Transmitter*
- *Filters*
- *Reference software*
- *Reference software flow*

Address map 1 standard buffers

byte address	dword address	register name
0x00000000	0x00000000	Receive Count
0x00000004	0x00000001	Receive Buffer
0x00002000	0x00000800	Transmit Buffer
0x00003F70	0x00000FDC	TBD checksum
0x00003F74	0x00000FDD	TCP/IP checksum
0x00003F78	0x00000FDE	UDP/IP checksum
0x00003F7C	0x00000FDF	IP checksum
0x00003F80	0x00000FE0	MAC ID low
0x00003F84	0x00000FE1	MAC ID high
0x00003F88	0x00000FE2	MAC IP
0x00003F8C	0x00000FE3	Command/Status/IFG
0x00003F90	0x00000FE4	Transmit Control
0x00003F94	0x00000FE5	Transmit Count
0x00003F98	0x00000FE6	Receive Control
0x00003F9C	0x00000FE7	Frame Count
0x00003FA0	0x00000FE8	Version

Address map 2 standard buffers

byte address	dword address	register name
0x00003FC0	0x00000FF0	DMA Control
0x00003FC4	0x00000FF1	Receive Destination (avalon)
0x00003FC8	0x00000FF2	Receive Source GMACII
0x00003FCC	0x00000FF3	Receive Length
0x00003FD0	0x00000FF4	Receive Checksum
0x00003FD4	0x00000FF5	Transmit Source (avalon)
0x00003FD8	0x00000FF6	Transmit Destination GMACII
0x00003FDC	0x00000FF7	Transmit Length
0x00003FE0	0x00000FF8	Transmit Checksum
0x00003FE4	0x00000FF9	Timer
0x00003FF8	0x00000FFE	PHY MANAGER IO
0x00003FFC	0x00000FFF	PHY MANAGER MIO

Address map 1 Advanced Features ON

byte address	dword address	register name
0x00000000	0x00000000	Receive Count
0x00000004	0x00000001	Receive Buffer
0x00010000	0x00004000	Transmit Buffer
0x0001FF70	0x00007FDC	TBD checksum
0x0001FF74	0x00007FDD	TCP/IP checksum
0x0001FF78	0x00007FDE	UDP/IP checksum
0x0001FF7C	0x00007FDF	IP checksum
0x0001FF80	0x00007FE0	MAC ID low
0x0001FF84	0x00007FE1	MAC ID high
0x0001FF88	0x00007FE2	MAC IP
0x0001FF8C	0x00007FE3	Command/Status/IFG
0x0001FF90	0x00007FE4	Transmit Control
0x0001FF94	0x00007FE5	Transmit Count
0x0001FF98	0x00007FE6	Receive Control
0x0001FF9C	0x00007FE7	Frame Count
0x0001FFA0	0x00007FE8	Version

Address map 2 Advanced Features ON

byte address	dword address	register name
0x0001FFA4	0x00007FE9	Configuration rd only
0x0001FFA8	0x00007FEA	Multicast MAC ID low
0x0001FFAC	0x00007FEB	Multicast MAC ID high
0x0001FFB0	0x00007FEC	Multicast MAC IP
0x0001FFB4	0x00007FED	reserved
0x0001FFB8	0x00007FEE	reserved
0x0001FFBC	0x00007FEF	reserved

Address map 3 Advanced Features ON

byte address	dword address	register name
0x0001FFC0	0x00007FF0	DMA Control
0x0001FFC4	0x00007FF1	Receive Destination (avalon)
0x0001FFC8	0x00007FF2	Receive Source GMACII
0x0001FFCC	0x00007FF3	Receive Length
0x0001FFD0	0x00007FF4	Receive Checksum
0x0001FFD4	0x00007FF5	Transmit Source (avalon)
0x0001FFD8	0x00007FF6	Transmit Destination GMACII
0x0001FFDC	0x00007FF7	Transmit Length
0x0001FFE0	0x00007FF8	Transmit Checksum
0x0001FFE4	0x00007FF9	Timer
0x0001FFF8	0x00007FFE	PHY MANAGER IO
0x0001FFFC	0x00007FFF	PHY MANAGER MIO

Define macros 1

ifi_gmacii_regs.h

```
C:\megafunctions\lfi_gmacii-v1.7\lib\sopc_builder\lfi_gmacii\inc\lfi_gmacii_regs.h
1:  /*
2:  *   Copyright (c) of IFI
3:  *
4:  *   this file belongs to the IFI_GMACII
5:  *   it defines the IO-macros to access the IP-core
6:  *
7:  *   Date:   January 12 2007
8:  *   Author: IFI/Sr
9:  *
10:  */
11: /* Code accessing the Bits, all listed Bits are Software readable others 0
12:  */
13: /*
14:  *   w   software writeable
15:  *   s   software write 1 to Set,   hardware clears, write 0 no operation
16:  *   c   software write 1 to Clear, hardware sets,   write 0 no operation
17:  *   h   hardware updated periodically
18:  */
19: #include <io.h>
20: #include "system.h"
21:
22: #ifndef __IFI_GMACII_REGS_H_
23: #define __IFI_GMACII_REGS_H_
24:
25:
26: // starting with revision 1.7 we have two different addressmaps, to keep the
27: // macros identical we need some help
28: // if jumbo is 1 we have the new addressmap
29: #ifndef jumbo
30: #define jumbo ((IFI_GMACII_0_GMACII_SLAVE_SPAN > 0x4000) ? 1 : 0)
31: #endif
32: // adders when having jumbo addressing
33: #define ja (jumbo * 0x00007000u)
34: // DW adder for jumbo
35: #define jb (jumbo * 0x0001c000u)
36: // Byte adder for jumbo
37: #define jt (jumbo * 0x00003800)
38: // DW adder transmit buffer
39: #define jbt (jumbo * 0x0000e000u)
40: // byte adder transmit buffer
41:
42:
43: // the Receive_buffer
44: #define IORD_IFI_GMACII_RBUFFER(base, index) IORD(base, ((0x00000001u)+index))
45:
46: /* MAC-ID, MAC_IP Register */
47: #define IORD_IFI_GMACII_MACIDL(base) IORD(base, (ja+0x00000fe0u))
48: #define IOWR_IFI_GMACII_MACIDL(base, data) IOWR(base, (ja+0x00000fe0u), data)
49: //ex: 0xdeadlab2c3
50: #define IORD_IFI_GMACII_MACIDH(base) IORD(base, (ja+0x00000fe1u))
51: #define IOWR_IFI_GMACII_MACIDH(base, data) IOWR(base, (ja+0x00000fe1u), data)
52: //ex: 0x00000007
53:
54: #define IORD_IFI_GMACII_MACIP(base) IORD(base, (ja+0x00000fe2u))
55: #define IOWR_IFI_GMACII_MACIP(base, data) IOWR(base, (ja+0x00000fe2u), data)
56: //ex: 0xc0a8642c for 192.168.100.44
57: 
```

Define macros 2

ifi_gmacii_regs.h

```
58: /* CMD Control/Status Register */
59: #define IORD_IFI_GMACII_CMD(base) IORD(base, (ja+0x00000fe3u))
60: #define IOWR_IFI_GMACII_CMD(base,data) IOWR_16DIRECT(base, (jb+0x000003f8cu), data)
61:
62: // bit 0 :w : disable ARP request frames when 1
63: // bit 1 :w : disable ICMP (PING) request frames when 1
64: // bit 2 :w : disable MCF receive frames when 1
65: // bit 3 :w : disable UDP receive frames when 1
66: // bit 4 :w : disable TCP receive frames when 1
67: // bit 5 :w : accept all types of IP-Frames 1 (not tested)
68: // bit 6 :w : (1.7) CRC checking off when set
69: // bit 7 :w : (1.7) use Multicast ID-IP for SRC filter when set
70: // bit 8 :h : Ethernet-speed is 100 Mbit when 1, else Gbit
71: // all 9..15 (1.7)
72: // bit 9 :w : length checking off when set (ARP is without lengthcheck)
73: // bit 11..10 : w : 00 accept UDP/IP, TCP/IP, ICMP(echo request), ARP
74: //
75: // 01 accept UDP/IP, TCP/IP, ICMP(echo request), ARP
76: // accept IGMP
77: //
78: // reserved 10 accept UDP/IP, TCP/IP, ICMP(echo request), ARP
79: // accept IGMP
80: //
81: //
82: // 11 accept frame types
83: //
84: // bit 13..12 : w : 00 accept broadcast IP
85: // accept DST MAC-IP that match
86: //
87: // 01 accept broadcast IP
88: // accept DST MAC-IP that match
89: // accept Multicast MAC_IP that match
90: //
91: // 10 accept broadcast IP
92: // accept DST MAC-IP that match
93: // accept all Multicast MAC_IPs
94: //
95: // 11 accept all MAC-IPs
96: //
97: // bit 15..14 : w : 00 accept broadcast ID
98: // accept DST MAC-ID that match
99: // when IFI_GMACII_CMD_MC_SRC_MSK set
100: // accept only when SRC MAC_ID match too
101: //
102: // 01 accept broadcast ID
103: // accept DST MAC-ID that match
104: // accept Multicast MAC_ID that match
105: //
106: // 10 accept broadcast ID
107: // accept DST MAC-ID that match
108: // accept all Multicast MAC_IDS
109: //
110: // 11 accept all MAC-IDS
111:
112: #define IFI_GMACII_CMD_ARP_MSK (0x00000001u)
113: #define IFI_GMACII_CMD_ARP_OFST (0)
114: #define IFI_GMACII_CMD_ICMP_MSK (0x00000002u)
115: #define IFI_GMACII_CMD_ICMP_OFST (1)
116: #define IFI_GMACII_CMD_MCF_MSK (0x00000004u)
117: #define IFI_GMACII_CMD_MCF_OFST (2)
118: #define IFI_GMACII_CMD_UDP_MSK (0x00000008u)
119: #define IFI_GMACII_CMD_UDP_OFST (3)
120: #define IFI_GMACII_CMD_TCP_MSK (0x00000010u)
121: #define IFI_GMACII_CMD_TCP_OFST (4)
122: #define IFI_GMACII_CMD_PI_MSK (0x00000020u)
123: #define IFI_GMACII_CMD_PI_OFST (5)
124: #define IFI_GMACII_CMD_CRCOFF_MSK (0x00000040u)
125: #define IFI_GMACII_CMD_CRCOFF_OFST (6)
126: #define IFI_GMACII_CMD_MC_SRC_MSK (0x00000080u)
127: #define IFI_GMACII_CMD_MC_SRC_OFST (7)
128: #define IFI_GMACII_CMD_I100_MSK (0x00000100u)
129: #define IFI_GMACII_CMD_I100_OFST (8)
130: #define IFI_GMACII_CMD_LEN_OFF_MSK (0x00000200u)
131: #define IFI_GMACII_CMD_LEN_OFF_OFST (9)
132: #define IFI_GMACII_CMD_FTyp_MSK (0x00000c00u)
133: #define IFI_GMACII_CMD_FTyp_OFST (10)
134: #define IFI_GMACII_CMD_FIP_MSK (0x00003000u)
135: #define IFI_GMACII_CMD_FIP_OFST (12)
136: #define IFI_GMACII_CMD_FID_MSK (0x0000c000u)
137: #define IFI_GMACII_CMD_FID_OFST (14)
```


Define macros 3

ifi_gmacii_regs.h

```
139: /* IFG Inter Frame Gap Register */
140: #define IORD_IFI_GMACII_IFG(base) IORD(base, (ja+0x00000fe3u))
141: #define IOWR_IFI_GMACII_IFG(base,data) IOWR_16DIRECT(base, (jb+0x000003f8eu), data)
142: // default and minimum is 12 Byte
143: #define IFI_GMACII_IFG_MSK (0x000000ffu)
144: #define IFI_GMACII_IFG_OFST (0)
145:
146: /* Transmit Control Register */
147: #define IORD_IFI_GMACII_TCR(base) IORD(base, (ja+0x00000fe4u))
148: #define IOWR_IFI_GMACII_TCR(base,data) IOWR_8DIRECT(base, (jb+0x00003f90u), data)
149:
150: // bit 0 :w : reset transmitter, set this bit to 1 to disable transmit-actions
151: // bit 1 :w : enable transmitter interrupt
152: // bit 2 :w : disable transmitter start, setting this bit disables the start
153: // of a new transmit frame
154: // bit 3 :w : reserved
155: // bit 4 :w : reserved
156: // bit 5 :c : transmitter interrupt pending, clear with writing 1
157: // bit 6 :c : packet transmitted, acknowledge transmitter, clear with writing 1
158: // bit 7 :s : transmit request, running
159: // to start a transmitframe set this bit to 1,
160: // when transmit is done this bit goes to 0
161: // when using pre_transmit_request (see below)
162: // this bit is set automatically
163: #define IFI_GMACII_TCR_TXRES_MSK (0x00000001u)
164: #define IFI_GMACII_TCR_TXRES_OFST (0)
165: #define IFI_GMACII_TCR_TIENA_MSK (0x00000002u)
166: #define IFI_GMACII_TCR_TIENA_OFST (1)
167: #define IFI_GMACII_TCR_DIST_MSK (0x00000004u)
168: #define IFI_GMACII_TCR_DIST_OFST (2)
169: #define IFI_GMACII_TCR_R3_MSK (0x00000008u)
170: #define IFI_GMACII_TCR_R3_OFST (3)
171: #define IFI_GMACII_TCR_R4_MSK (0x00000010u)
172: #define IFI_GMACII_TCR_R4_OFST (4)
173: #define IFI_GMACII_TCR_IACK_MSK (0x00000020u)
174: #define IFI_GMACII_TCR_IACK_OFST (5)
175: #define IFI_GMACII_TCR_TACK_MSK (0x00000040u)
176: #define IFI_GMACII_TCR_TACK_OFST (6)
177: #define IFI_GMACII_TCR_TRANS_MSK (0x00000080u)
178: #define IFI_GMACII_TCR_TRANS_OFST (7)
179:
180: // pre_transmit_request, request next transmit, even actual is stillrunning
181: #define IORD_IFI_GMACII_PRETCR(base) IORD_8DIRECT(base, (jb+0x00003f91u))
182: #define IOWR_IFI_GMACII_PRETCR(base,data) IOWR_8DIRECT(base, (jb+0x00003f91u), data)
183: // bit 0 :s : pre_transmit_request, request pending when 1
184: // to start transmitframes with minimum gap use this pre_transmit_request
185: // set bit 0 to 1, then wait until this bit is 0,
186: // this signals the request is accepted and the transmitbuffers are swapped
187: // now the next transmitframe can be put together
188: #define IFI_GMACII_TCR_PRETRANS_MSK (0x00000001u)
189: #define IFI_GMACII_TCR_PRETRANS_OFST (0)
190:
191: /* Transmit Count Register */
192: #define IORD_IFI_GMACII_TCNT(base) IORD(base, (ja+0x00000fe5u))
193: #define IOWR_IFI_GMACII_TCNT(base,data) IOWR(base, (ja+0x00000fe5u), data)
194: // number of bytes the transmitter has to send (without the CRC)
195: // minimum is 60 Byte,
196: // (1.7) starting with revision 1.7 the GMACII extends a frame with zeros
197: // when tcnt is smaller than 60
198: // because the CRC with 4 bytes is added to this, we get the minimum framelength of 64bytes
199: // on the wire
200: // the maximum TCNT depends on the transmit buffer size
201: // for the old addressmap (jumbo = off,0) we have a value of
202: // 1904 bytes !!
203: // for the new addressmap (jumbo = on,1) we have the values
204: // TAW = 10 => 2048 bytes (2052 on the wire)
205: // TAW = 11 => 4096
206: // TAW = 12 => 8192
207: // TAW = 13 => 16384
208: // TAW = 14 => 32768
209: // TAW = 15 => 65392 !! (not 65536)
```

Define macros 4

ifi_gmacii_regs.h

```
210:
211:
212: /* Receive Control Register */
213: #define IORD_IFI_GMACII_RCR(base) IORD(base, (ja+0x0000fe6u))
214: #define IOWR_IFI_GMACII_RCR(base,data) IOWR(base, (ja+0x0000fe6u), data)
215:
216: // bit 0 :w : reset receiver, set this bit to 1 to disable receive-actions
217: // bit 1 :w : enable receiver interrupt
218: // bit 2 :w : CRC error interrupt enable (1.7)
219: // bit 3 :w : ReceiveBuffer error interrupt enable (1.7)
220: // bit 4 :c : CRC error interrupt (1.7)
221: // bit 5 :c : clear receiver interrupt
222: // bit 6 :c : packet pending, acknowledge receiver
223: // bit 7 :c : ReceiveBuffer interrupt (1.7)
224: // a Receive buffer error allways implies a crc error !!
225: #define IFI_GMACII_RCR_RXRES_MSK (0x00000001u)
226: #define IFI_GMACII_RCR_RXRES_OFST (0)
227: #define IFI_GMACII_RCR_RIENA_MSK (0x00000002u)
228: #define IFI_GMACII_RCR_RIENA_OFST (1)
229: #define IFI_GMACII_RCR_CRCIENA_MSK (0x00000004u)
230: #define IFI_GMACII_RCR_CRCIENA_OFST (2)
231: #define IFI_GMACII_RCR_RBUFIENA_MSK (0x00000008u)
232: #define IFI_GMACII_RCR_RBUFIENA_OFST (3)
233: #define IFI_GMACII_RCR_CRCERR_MSK (0x00000010u)
234: #define IFI_GMACII_RCR_CRCERROFST (4)
235: // pre 1.7
236: // #define IFI_GMACII_RCR_R2_MSK (0x00000004u)
237: // #define IFI_GMACII_RCR_R2_OFST (2)
238: // #define IFI_GMACII_RCR_R3_MSK (0x00000008u)
239: // #define IFI_GMACII_RCR_R3_OFST (3)
240: // #define IFI_GMACII_RCR_R4_MSK (0x00000010u)
241: // #define IFI_GMACII_RCR_R4_OFST (4)
242: #define IFI_GMACII_RCR_IACK_MSK (0x00000020u)
243: #define IFI_GMACII_RCR_IACK_OFST (5)
244: #define IFI_GMACII_RCR_RACK_MSK (0x00000040u)
245: #define IFI_GMACII_RCR_RACK_OFST (6)
246: #define IFI_GMACII_RCR_RBUFERR_MSK (0x00000080u)
247: #define IFI_GMACII_RCR_RBUFERR_OFST (7)
248: // working with interrupts is not recommended because of low performance !
249: //
250: // after clearing the receiver interrupt read the framecounter
251: // and process all pending frames, a new interrupt is only
252: // generated when a new frame is received and not on pending frames !
253:
254:
255: /* Frame Count Register */
256: #define IORD_IFI_GMACII_FCNT(base) IORD(base, (ja+0x0000fe7u))
257: // pending number of frames in receive buffer
258:
259: /* Receive Count Register */
260: #define IORD_IFI_GMACII_RCNT(base) IORD(base, (0x00000000u))
261: // number of correctly received bytes in first frame
262: // it is necessary to read the receive count register at least once
263: // before setting the IFI_GMACII_RCR_RACK_MSK in the IOWR_IFI_GMACII_RCR
264: // this keeps the ringbuffer in shape !
265:
266: /* version read only */
267: #define IORD_IFI_GMACII_VER(base) IORD(base, (ja+0x0000fe8u))
268: #define IFI_GMACII_VER_VER_MSK (0x000000ffu)
269: #define IFI_GMACII_VER_VER_OFST (0)
270: #define IFI_GMACII_VER_QII_MSK (0x0000ff00u)
271: #define IFI_GMACII_VER_QII_OFST (8)
272: #define IFI_GMACII_VER_YEAR_MSK (0x00ff0000u)
273: #define IFI_GMACII_VER_YEAR_OFST (16)
274: #define IFI_GMACII_VER_MONTH_MSK (0xff000000u)
275: #define IFI_GMACII_VER_MONTH_OFST (24)
276: // ex: 0x07055010
277: // ex: 0x04065115 for April, 2006, required QuartusII rev5.1+, GMACII rev1.5
278:
```


Define macros 5

ifi_gmacii_regs.h

```
278: /* configuration read only */
279: #define IORD_IFI_GMACII_CFG(base) IORD(base, (ja+0x00000fe9u))
280: #define IFI_GMACII_CFG_RAW_MSK (0x0000000fu)
281: #define IFI_GMACII_CFG_RAW_OFST (0)
282: #define IFI_GMACII_CFG_TAW_MSK (0x000000f0u)
283: #define IFI_GMACII_CFG_TAW_OFST (4)
284: #define IFI_GMACII_CFG_TDP_MSK (0x00000100u)
285: #define IFI_GMACII_CFG_TDP_OFST (8)
286: #define IFI_GMACII_CFG_NEWADR_MSK (0x80000000u)
287: #define IFI_GMACII_CFG_NEWADR_OFST (31)
288: // RAW :10 => 4 kbyte receive ringbuffer (default)
289: // RAW :11 => 8 kbyte receive ringbuffer
290: // RAW :12 => 16 kbyte receive ringbuffer
291: // RAW :13 => 32 kbyte receive ringbuffer
292: // RAW :14 => 64 kbyte receive ringbuffer
293: // RAW :15 => 128 kbyte receive ringbuffer
294: // TAW :10 => 2+ 2 kbyte transmit doublebuffer (default)
295: // TAW :11 => 4+ 4 kbyte transmit doublebuffer
296: // TAW :12 => 8+ 8 kbyte transmit doublebuffer
297: // TAW :13 => 16+16 kbyte transmit doublebuffer
298: // TAW :14 => 32+32 kbyte transmit doublebuffer
299: // TAW :15 => 64+64 kbyte transmit doublebuffer
300: // TDP : 0 => transmit buffer write only
301: // TDP : 1 => transmit buffer readback enabled (only the actual transmitbuffer)
302: // NEWADR : 0 => we have the old addressmap (total 16 kbyte)
303: // NEWADR : 1 => we have the new addressmap (total 128 kBytes)
304:
305: /* Multicast MAC-ID, Multicast MAC_IP Register (1.7) */
306: #define IORD_IFI_GMACII_MCMACIDL(base) IORD(base, (ja+0x00000feau))
307: #define IOWR_IFI_GMACII_MCMACIDL(base, data) IOWR(base, (ja+0x00000feau), data)
308: //ex: 0xdeadlab2c3
309: #define IORD_IFI_GMACII_MCMACIDH(base) IORD(base, (ja+0x00000febu))
310: #define IOWR_IFI_GMACII_MCMACIDH(base, data) IOWR(base, (ja+0x00000febu), data)
311: //ex: 0x00000007
312:
313: #define IORD_IFI_GMACII_MCMACIP(base) IORD(base, (ja+0x00000fecu))
314: #define IOWR_IFI_GMACII_MCMACIP(base, data) IOWR(base, (ja+0x00000fecu), data)
315: //ex: 0xc0a8642c for 192.168.100.44
316:
317: // write into transmitbuffer *****
318: // index in DWORD
319: #define IOWR_IFI_GMACII_TBUFFER(base, index, data) IOWR(base, (jt+(0x00000800u)+index), data)
320: #define IORD_IFI_GMACII_TBUFFER(base, index) IORD(base, (jt+(0x00000800u)+index))
321: // index in BYTE
322: #define IOWR_IFI_GMACII_TBUFFER32(base, index, data) IOWR_32DIRECT(base, (jbt+(0x000002000u)+index), data)
323: #define IOWR_IFI_GMACII_TBUFFER16(base, index, data) IOWR_16DIRECT(base, (jbt+(0x000002000u)+index), data)
324: #define IOWR_IFI_GMACII_TBUFFER8(base, index, data) IOWR_8DIRECT(base, (jbt+(0x000002000u)+index), data)
325:
326: // convert and store checksum into transmitbuffer
327: // the written 32bit checksum is converted to 16bit and
328: #define IOWR_IFI_GMACII_CSIP(base, data) IOWR(base, (ja+0x00000fd0u), data)
329: // stored to the IP-header-checksum place (offset is Byte 24)
330: #define IOWR_IFI_GMACII_CSUDP(base, data) IOWR(base, (ja+0x00000fdeu), data)
331: // stored to the UDP-data checksum place (offset is Byte 40)
332: #define IOWR_IFI_GMACII_CSTCP(base, data) IOWR(base, (ja+0x00000fddu), data)
333: // stored to the TCP checksum place (offset is Byte 50)
```

Define macros 6

ifi_gmacii_regs.h

```
337: /* for GMACII DMA *****/
338: /* Destination address on Avalon Bus for received data */
339: #define IORD_IFI_GMACII_DMCR(base) IORD(base, (ja+0x0000ff0u))
340: #define IOWR_IFI_GMACII_DMCR(base, data) IOWR(base, (ja+0x0000ff0u), data)
341: // bit 0 :w : reset receiver DMA, set this bit to 1 to disable DMA-actions
342: // bit 1 :w : enable receiver DMA interrupt
343: // bit 2 :s : request DMA receive-part (enable)
344: // bit 3 :h : done DMA receive-part ready
345: // clear bit 3 and set bit 2 to 1 => start the DMA receive-part
346: // when DMA has finished it sets bit 3 and clears bit 2
347: // so, wait until bit 3 is 1, now DMA is free and can be reloaded
348: // we have only one DMA engine, so receive and transmit at
349: // the same time is not possible
350: // bit 4 :w : avalonaddress is constant on receive when set (1.7)
351: #define IFI_GMACII_DMCR_RXRESET_MSK (0x00000001u)
352: #define IFI_GMACII_DMCR_RXRESET_OFST (0)
353: #define IFI_GMACII_DMCR_RXIENA_MSK (0x00000002u)
354: #define IFI_GMACII_DMCR_RXIENA_OFST (1)
355: #define IFI_GMACII_DMCR_RXENA_MSK (0x00000004u)
356: #define IFI_GMACII_DMCR_RXENA_OFST (2)
357: #define IFI_GMACII_DMCR_RXDONE_MSK (0x00000008u)
358: #define IFI_GMACII_DMCR_RXDONE_OFST (3)
359: #define IFI_GMACII_DMCR_RXDESTCONST_MSK (0x00000010u)
360: #define IFI_GMACII_DMCR_RXDESTCONST_OFST (4)
361:
362: // bit 16 :w : reset transmitter DMA, set this bit to 1 to disable DMA-actions
363: // bit 17 :w : enable transmitter DMA interrupt
364: // bit 18 :s : request DMA transmitter-part (enable)
365: // bit 19 :h : done DMA transmitter-part ready
366: // clear bit 19 and set bit 18 to 1 => start the DMA transmitter-part
367: // when DMA has finished it sets bit 19 and clears bit 18
368: // so, wait until bit 18 is 1, now DMA is free and can be reloaded
369: // we have only one DMA engine, so receive and transmit at
370: // the same time is not possible
371: // bit 20 :w : avalonaddress is constant on transmit when set (1.7)
372: #define IFI_GMACII_DMCR_TXRESET_MSK (0x00010000u)
373: #define IFI_GMACII_DMCR_TXRESET_OFST (16)
374: #define IFI_GMACII_DMCR_TXIENA_MSK (0x00020000u)
375: #define IFI_GMACII_DMCR_TXIENA_OFST (17)
376: #define IFI_GMACII_DMCR_TXENA_MSK (0x00040000u)
377: #define IFI_GMACII_DMCR_TXENA_OFST (18)
378: #define IFI_GMACII_DMCR_TXDONE_MSK (0x00080000u)
379: #define IFI_GMACII_DMCR_TXDONE_OFST (19)
380: #define IFI_GMACII_DMCR_TXSRCCONST_MSK (0x00100000u)
381: #define IFI_GMACII_DMCR_TXSRCCONST_OFST (20)
```


Define macros 7

ifi_gmacii_regs.h

```
382: /* Destination address on Avalon Bus for received data */
383: #define IORD_IFI_GMACII_RXDEST(base) IORD(base, (ja+0x0000ff1u))
384: #define IOWR_IFI_GMACII_RXDEST(base, data) IOWR(base, (ja+0x0000ff1u), data)
385:
386: /* Source address for GMACII core receivebuffer */
387: #define IORD_IFI_GMACII_RXSRC(base) IORD(base, (ja+0x0000ff2u))
388: #define IOWR_IFI_GMACII_RXSRC(base, data) IOWR(base, (ja+0x0000ff2u), data)
389: // ex: 0x00000004 for Receivebuffer-data
390: #define IFI_GMACII_RXSRC_MSK (0x0000ffffu)
391: #define IFI_GMACII_RXSRC_OFST (0)
392:
393: /* Bytecount to be transfered from receivebuffer to avalon bus*/
394: #define IORD_IFI_GMACII_RXLEN(base) IORD(base, (ja+0x0000ff3u))
395: #define IOWR_IFI_GMACII_RXLEN(base, data) IOWR(base, (ja+0x0000ff3u), data)
396: #define IFI_GMACII_RXLEN_MSK (0x0000ffffu)
397: #define IFI_GMACII_RXLEN_OFST (0)
398:
399: /* Checksum of received data, read only */
400: #define IORD_IFI_GMACII_RXCS(base) IORD(base, (ja+0x0000ff4u))
401: #define IFI_GMACII_RXCS_MSK (0x0000ffffu)
402: #define IFI_GMACII_RXCS_OFST (0)
403: // each DMA transfer generates the checksum of all transported bytes
404: // the carry is already included, so checksum has 16 bit
405:
406: /* Source address on Avalon Bus for transmit data */
407: #define IORD_IFI_GMACII_TXSRC(base) IORD(base, (ja+0x0000ff5u))
408: #define IOWR_IFI_GMACII_TXSRC(base, data) IOWR(base, (ja+0x0000ff5u), data)
409:
410: /* Destination address on GMACII core transmit buffer for transmit data */
411: #define IORD_IFI_GMACII_TXDEST(base) IORD(base, (ja+0x0000ff6u))
412: #define IOWR_IFI_GMACII_TXDEST(base, data) IOWR(base, (ja+0x0000ff6u), (jbt+(data)))
413: // ex: 0x00002000 first byte of transmit buffer
414: #define IFI_GMACII_TXDEST_MSK (0x0000ffffu)
415: #define IFI_GMACII_TXDEST_OFST (0)
416:
417: /* Bytecount to be transfered from avalon bus to transmitbuffer*/
418: #define IORD_IFI_GMACII_TXLEN(base) IORD(base, (ja+0x0000ff7u))
419: #define IOWR_IFI_GMACII_TXLEN(base, data) IOWR(base, (ja+0x0000ff7u), data)
420: #define IFI_GMACII_TXLEN_MSK (0x0000ffffu)
421: #define IFI_GMACII_TXLEN_OFST (0)
422:
423: /* Checksum of transmitted data, read only */
424: #define IORD_IFI_GMACII_TXCS(base) IORD(base, (ja+0x0000ff8u))
425: #define IFI_GMACII_TXCS_MSK (0x0000ffffu)
426: #define IFI_GMACII_TXCS_OFST (0)
427: // each DMA transfer generates the checksum of all transported bytes
428: // the carry is already included, so checksum has 16 bit
429:
430: /* systemtimer in us since reset, read only */
431: #define IORD_IFI_GMACII_SYSTIME(base) IORD(base, (ja+0x0000ff9u))
432: // attention: rollover after about 71 minutes depending on sys_clk !!
433: // GMACII parameter sys_clk must be set in SOPC-Builder / Megawizard
434: // to the used system clock rate in MHz
435:
436:
```


Define macros 8

ifi_gmacii_regs.h

```
437: /*
438: /* *****
439: /* IFI PHY MANAGER inside the GMACII DMA part *****
440: /* MIO Control Register */
441: #define IORD_IFI_PHY_MANAGER_MIO(base)      IORD(base, (ja+0x00000fffu))
442: #define IOWR_IFI_PHY_MANAGER_MIO(base,data) IOWR(base, (ja+0x00000fffu), data)
443: // bit 0 : w : MDC Clock
444: // bit 1 : w : Output enable for MDIO 1 => Z, 0 => 0
445: // bit 2 : w : Interrupt Enable
446: // bit 3 : h : MDIO in, data from Phy
447: // bit 4 : h : Interrupt pin from Phy(not masked)
448: #define IFI_PHY_MANAGER_MIO_MDC_MSK      (0x00000001u)
449: #define IFI_PHY_MANAGER_MIO_MDC_OFST      (0)
450: #define IFI_PHY_MANAGER_MIO_MOE_MSK      (0x00000002u)
451: #define IFI_PHY_MANAGER_MIO_MOE_OFST      (1)
452: #define IFI_PHY_MANAGER_MIO_IENA_MSK      (0x00000004u)
453: #define IFI_PHY_MANAGER_MIO_IENA_OFST      (2)
454: #define IFI_PHY_MANAGER_MIO_MDI_MSK      (0x00000008u)
455: #define IFI_PHY_MANAGER_MIO_MDI_OFST      (3)
456: #define IFI_PHY_MANAGER_MIO_INT_MSK      (0x00000010u)
457: #define IFI_PHY_MANAGER_MIO_INT_OFST      (4)
458:
459: /* IO Control Register */
460: #define IORD_IFI_PHY_MANAGER_IO(base)      IORD(base, (ja+0x00000ffeu))
461: #define IOWR_IFI_PHY_MANAGER_IO(base,data) IOWR(base, (ja+0x00000ffeu), data)
462: // bit 0 : w : DUPLEX output enable 0 => Z, 1 => DUPLEX output value
463: // bit 1 : w : DUPLEX output value
464: // bit 2 : w : PHYAD0 output enable 0 => Z, 1 => PHYAD0 output value
465: // bit 3 : w : PHYAD0 output value
466: // bit 4 : w : ANEN output enable 0 => Z, 1 => ANEN output avlue
467: // bit 5 : w : ANEN output value
468: // bit 6 : h : DUPLEX input value
469: // bit 7 : h : PHYAD0 input value
470: // bit 8 : h : ANEN input value
471: #define IFI_PHY_MANAGER_IO_DOE_MSK      (0x00000001u)
472: #define IFI_PHY_MANAGER_IO_DOE_OFST      (0)
473: #define IFI_PHY_MANAGER_IO_DUP_MSK      (0x00000002u)
474: #define IFI_PHY_MANAGER_IO_DUP_OFST      (1)
475: #define IFI_PHY_MANAGER_IO_POE_MSK      (0x00000004u)
476: #define IFI_PHY_MANAGER_IO_POE_OFST      (2)
477: #define IFI_PHY_MANAGER_IO_PHY_MSK      (0x00000008u)
478: #define IFI_PHY_MANAGER_IO_PHY_OFST      (3)
479: #define IFI_PHY_MANAGER_IO_AOE_MSK      (0x00000010u)
480: #define IFI_PHY_MANAGER_IO_AOE_OFST      (4)
481: #define IFI_PHY_MANAGER_IO_ANE_MSK      (0x00000020u)
482: #define IFI_PHY_MANAGER_IO_ANE_OFST      (5)
483: #define IFI_PHY_MANAGER_IO_DI_MSK      (0x00000040u)
484: #define IFI_PHY_MANAGER_IO_DI_OFST      (6)
485: #define IFI_PHY_MANAGER_IO_PI_MSK      (0x00000080u)
486: #define IFI_PHY_MANAGER_IO_PI_OFST      (7)
487: #define IFI_PHY_MANAGER_IO_AI_MSK      (0x00000100u)
488: #define IFI_PHY_MANAGER_IO_AI_OFST      (8)
489:
490: #endif /* __IFI_GMACII_REGS_H */
491: */
```

Version

Byte 3	Bit	31	30	29	28	27	26	25	24
	read	Month 7	Month 6	Month 5	Month 4	Month 3	Month 2	Month 1	Month 0
	write								
Byte 2	Bit	23	22	21	20	19	18	17	16
	read	Year 7	Year 6	Year 5	Year 4	Year 3	Year 2	Year 1	Year 0
	write								
Byte 1	Bit	15	14	13	12	11	10	9	8
	read	Quartus 7	Quartus 6	Quartus 5	Quartus 4	Quartus 3	Quartus 2	Quartus 1	Quartus 0
	write								
Byte 0	Bit	7	6	5	4	3	2	1	0
	read	Core rev 7	Core rev 6	Core rev 5	Core rev 4	Core rev 3	Core rev 2	Core rev 1	Core rev 0
	write								

■ month - year - quartus - core revision

– Example: 0x01076117 → 01.2007 Quartus 6.1 Core 1.7

Details DMA

the GMACII includes an internal DMA controller with special features:

- alignment aware and byte exact
the GMACII-DMA is able to copy an exact number of bytes from the source (byteaddress) to the destination (byteaddress), with this feature we get the high copy-performance of up to 4 Bytes/clock_cycle even when the databytes need to be shifted 1,2 or 3 bytes (limitation: we read some bytes more than we need)
- checksum logic
all written bytes are going into the checksum-adder, so after each copying is done the checksum can be read, there is no overhead, when not used, the checksum-adder is cleared with each new copy request, the receive part and the transmit part have their own checksum registers.
- pipelinesupport
the GMACII-DMA has pipeline support on both ends, that means we can reach the maximum possible throughput
- separate register for receiver and transmitter
we have separate registers for the receiver part of the DMA (reading data from the receive-buffer, writing to the avalon-bus) and the transmitter part of the DMA (reading from the avalon-bus, writing to the transmit-buffer), this give better performance, and less overhead
the receiver-DMA can only read the receive-buffer !
the transmit-DMA can only write the transmit-buffer !

Details transmitter

■ padding

starting with revision 1.7 the GMACII transmitter makes the padding to extend a frame that it meets the minimum framelength of 64 bytes (for older versions that was business of the software)

example: the TCNT was set to 60 bytes, this tells the transmitter to send 60 bytes from the transmit-buffer, append the 4 bytes from the automatically generated CRC, and 64bytes are on the line

setting the TCNT to less than 60 Bytes, a frame with TCNT+ 4 Bytes was send !!, but that frame could be removed from a switch or a networkcard, because of violating the minimum length requirement

With the new feature the transmitter appends X"00" bytes up to byte 60, when the TCNT is shorter than 60, this behavior can not be switched off

■ The transmit-buffer can be filled in any order with the exact bytes needed

■ We have two possibilities to start the transmission

1. Wait until the IFI_GMACII_TCR_TRANS_MSK is 0, load the TCNT, start the transmission with setting IFI_GMACII_TCR_TRANS_MSK, the transmit-buffer is switched immediately and is sending, the next transmit-buffer can be loaded now
 2. Load the transmit-buffer, set the IFI_GMACII_TCR_PRETRANS_MSK, that is the prestart for the transmitter, then wait until this bit is cleared, the transmit starts at the earliest possible time (after the interframe gap) to send, and the buffer is switched, we can now load the next transmit frame, with this pre-start it is possible to reach the minimum interframe-gap of 12 bytes
- mixing both versions to start transmission is not recommended

Filter Details MAC-ID

Filtering the MAC-ID, the MAC-IP and IP-Header is implemented to reduce the overhead in comparing the received frames for the cpu in embedded systems

■ MAC-ID filter:

Each device has to have it's own MAC-ID, which has to be loaded into the GMACII-core by software (MAC ID low - the last 32 bit and MAC ID high - the first 16 bit).

The MAC-ID filter decodes the first 6 bytes of each received frame (called Destination Address) and compares to that loaded value.

If all 48bit match, the received frame is accepted.

Additionally a Destination Address of "FF FF FF FF FF FF" is accepted as broadcast.

■ MCMAC-ID filter for Multicast (Advanced Features ON):

- For receiving multicast frames we have an additional filter set. This filter works:
 - In parallel to the MAC-ID filter, so we can receive both kinds of destination IDs
 - Or work as SRC Address filter when not used as Multicast
 - filtering the SRC address can increase the security level, example:
 - after a communication is set up, the local software can load the MCMAC-ID with the SRC address of the partner, and enable that filter, so only IP communication with that partner gets processed, other partners can not get through
- Additionally a Destination Address of "01 00 5E 0xxxxxxx XX XX" is accepted as multicast-broadcast.
 - 01-00-5e-00-00-00 ... 01-00-5e-7f-ff-ff

Filter Details MAC-IP

- MAC-IP filter:

Each device has to have its own MAC-IP, which has to be loaded into the GMACII-core by software. This IP can be fixed, or received from a DHCP-server depending on the system's structure.

The MAC-IP filter decodes the Destination IP Address and compares it to the loaded value.

If all 32 bits match, the received frame is accepted.

Additionally a Destination Address of "FF FF FF FF" is accepted as broadcast.

- MCMAC-IP filter for Multicast (Advanced Features ON):

For receiving multicast frames we have an additional filter set. This filter works in parallel to the MAC-IP filter, so we can receive both kinds of destination IPs

Additionally a Destination Address of "Exxxx.XX.XX.XX" is accepted as multicast-broadcast.

224.0.0.0 ... 239.255.255.255

Filter Details frame types and length

- We have a filter to detect the type of the frame
 - ARP
 - IP which is divided in
 1. ICMP internet control message protocol
 2. IGMP internet group management protocol
 3. UDP/IP user datagram protocol
 4. TCP/IP Transmission control protocol
- For IP we can test if the announced total length is matching the real frame length with revisions until 1.7 we did this length test also on ARP, this failed on some systems using a special driver from Intel, that driver did not extend a ARP frame to the minimum of 64 bytes on the wire, it extended to 108 bytes (or so), and the GMACII rejected that ARP
- All these filters can be combined or disabled, please see the `ifi_gmac_regs.h` for details, in the reference software you can find a subroutine which prints the actual setting

Summary for older GMACII revisions

- ARP request filter:
Only valid ARP-requests are accepted (length and type)
- ICMP request filter:
Only valid ICMP-requests (ping request) are accepted (length and type, echo request)
- IP filter:
Only UDP/IP frames and TCP/IP frames are accepted, other types are discarded (length and type)
- LEN filter:
To minimize the test-overhead for the cpu, the length of ARP-requests (fix), ICMP-requests and IP-frames is validated, this can not be switched off.
Frames can not use options in the protocols or other header types.
Example: an incorrect IP-frame tells a length of 100 byte but the real received frame has 1300 bytes, this frame is discarded by this filter.
- CRC filter:
The crc filter is always running and accepts frames only, when the correct crc is found in the last 4 bytes of a frame. This filter can not be switched off.

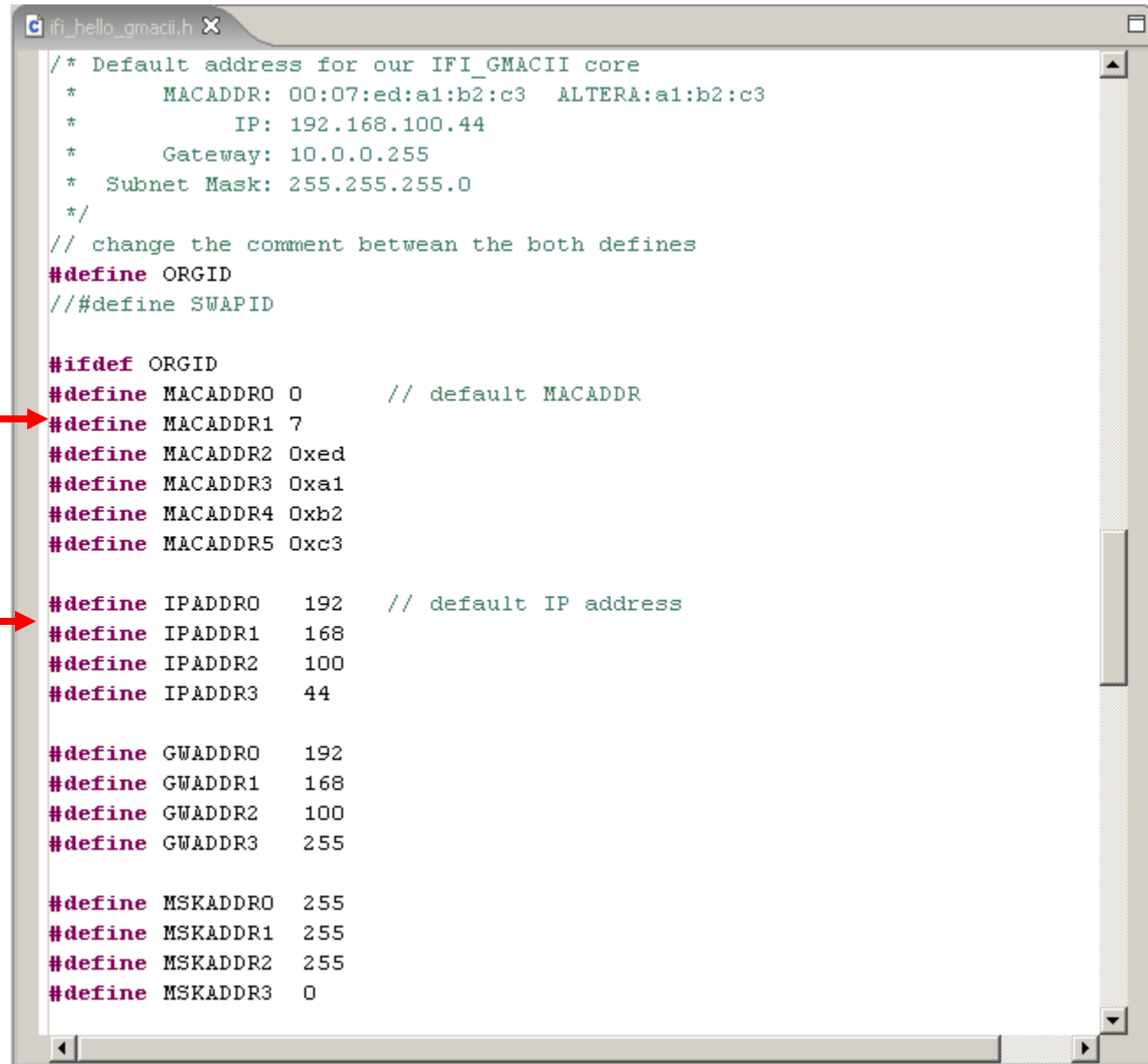
Reference-software for NIOS II and IDE

- project template in the IDE
 - IFI_hello_GMACII
- Files
 - ifi_hello_gmacii.h
 - Settings
 - ifi_hello_gmacii.c
 - Example application
 - ifi_phy_manager.c
 - Configuration and communication with the PHY Manager
 - ifi_tftp_server.c
 - TFTP Server routine
 - ifi_tftp_client.c
 - TFTP client routine
 - ifi_udp_burst.c
 - UDP burst routine
 - ifi_arp_reply.c
 - ARP reply routine
 - ifi_arp_request.c
 - ARP request routine
 - ifi_ping_reply.c
 - PING reply routine
 - ifi_printfilter.c
 - Print the actual filter settings
 - ifi_ip_copycheck.c
 - Example for checking the checksum, during DMA copy

- Open ifi_hello_gmacii.h

- Modify MAC ADDR

- Modify IP address



```
ifi_hello_gmacii.h x
/* Default address for our IFI_GMACII core
 *   MACADDR: 00:07:ed:a1:b2:c3   ALTERA:a1:b2:c3
 *   IP: 192.168.100.44
 *   Gateway: 10.0.0.255
 *   Subnet Mask: 255.255.255.0
 */
// change the comment between the both defines
#define ORGID
//#define SW&PID

#ifdef ORGID
#define MACADDR0 0           // default MACADDR
#define MACADDR1 7
#define MACADDR2 0xed
#define MACADDR3 0xa1
#define MACADDR4 0xb2
#define MACADDR5 0xc3

#define IPADDR0 192         // default IP address
#define IPADDR1 168
#define IPADDR2 100
#define IPADDR3 44

#define GWADDR0 192
#define GWADDR1 168
#define GWADDR2 100
#define GWADDR3 255

#define MSKADDR0 255
#define MSKADDR1 255
#define MSKADDR2 255
#define MSKADDR3 0
```

■ Burst Destination MAC



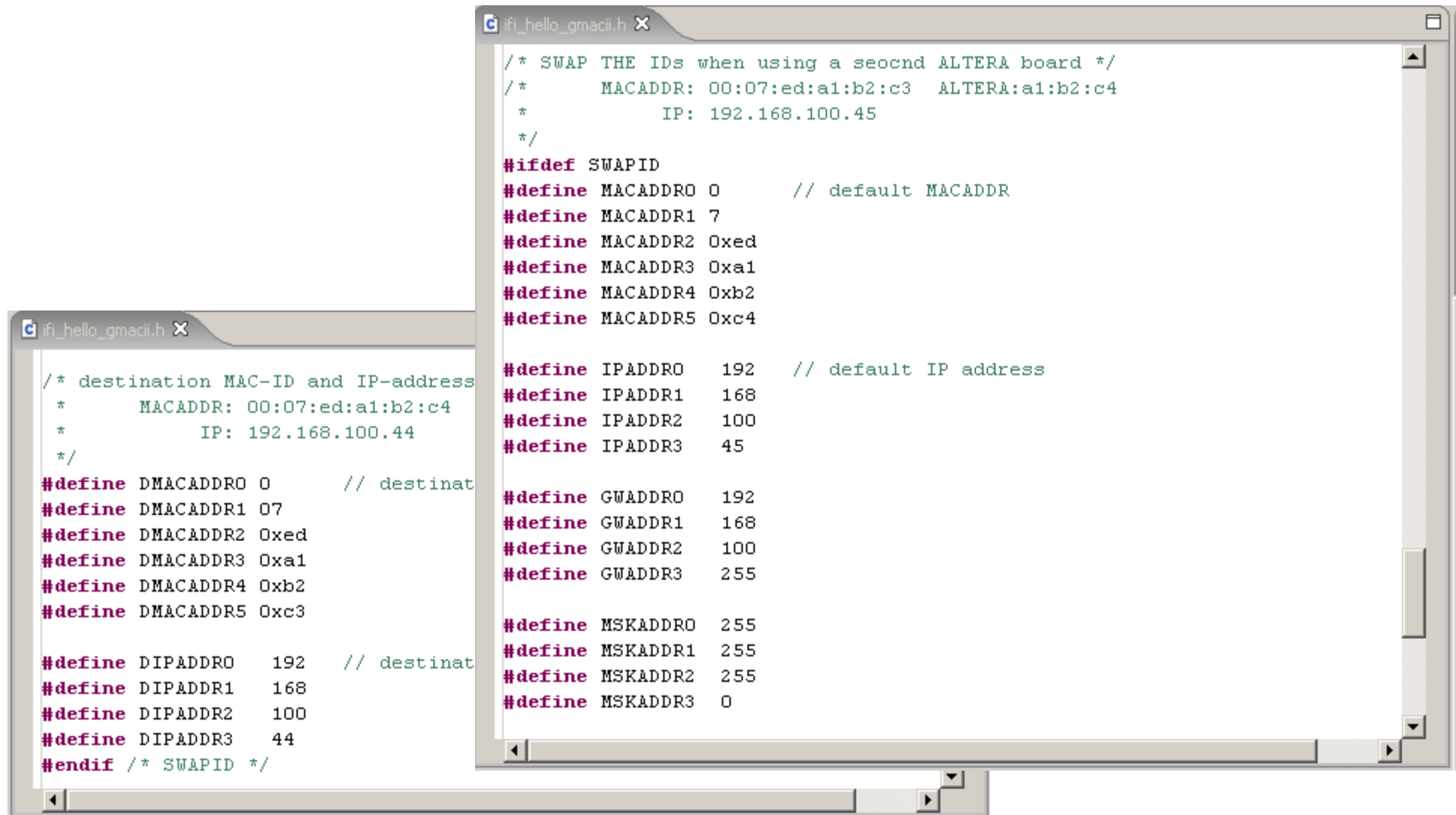
■ Burst Destination IP



```
ifi_hello_gmacii.h x
/* destination MAC-ID and IP-address for our partner we send our bursts
 *      MACADDR: 00:0b:db:5c:be:d9
 *      IP: 192.168.100.110
 */
#define DMACADDR0 0      // destination MACADDR
#define DMACADDR1 0xb
#define DMACADDR2 0xdb
#define DMACADDR3 0x5c
#define DMACADDR4 0xbe
#define DMACADDR5 0xd9

#define DIPADDR0 192     // destination IP address
#define DIPADDR1 168
#define DIPADDR2 100
#define DIPADDR3 110
#endif /* ORGID */
```

- Second set of MAC and IP address
 - Change the comments to use this



The image shows two overlapping windows displaying C preprocessor code for configuring MAC and IP addresses. The top window, titled 'ifi_hello_gmacii.h', contains a block of code that is commented out, indicated by '/*' and '*/' at the beginning and end of the section. This commented-out code defines MAC addresses (MACADDR0 to MACADDR5) and IP addresses (IPADDR0 to IPADDR3) for a second ALTERA board. The bottom window, also titled 'ifi_hello_gmacii.h', shows the same code but with the comments removed, making it active. The code defines destination MAC addresses (DMACADDR0 to DMACADDR5) and destination IP addresses (DIPADDR0 to DIPADDR3) for a second board. The code is as follows:

```
/* SWAP THE IDs when using a second ALTERA board */
/*     MACADDR: 00:07:ed:a1:b2:c3   ALTERA:a1:b2:c4
 *     IP: 192.168.100.45
 */
#ifdef SWAPID
#define MACADDR0 0          // default MACADDR
#define MACADDR1 7
#define MACADDR2 0xed
#define MACADDR3 0xa1
#define MACADDR4 0xb2
#define MACADDR5 0xc4

#define IPADDR0 192        // default IP address
#define IPADDR1 168
#define IPADDR2 100
#define IPADDR3 45

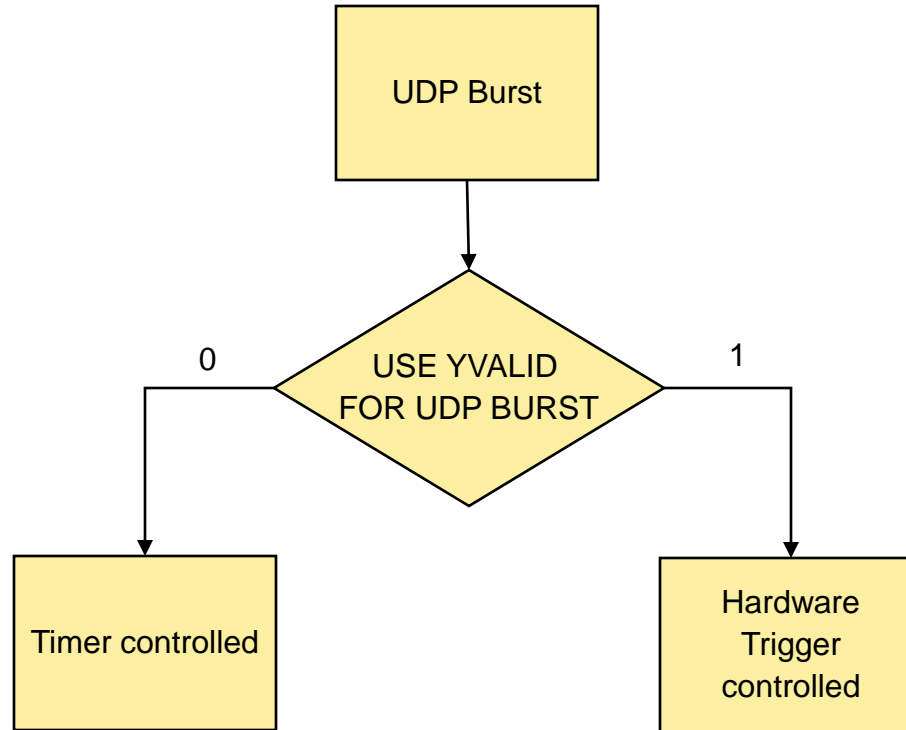
#define GWADDR0 192
#define GWADDR1 168
#define GWADDR2 100
#define GWADDR3 255

#define MSKADDR0 255
#define MSKADDR1 255
#define MSKADDR2 255
#define MSKADDR3 0
#endif

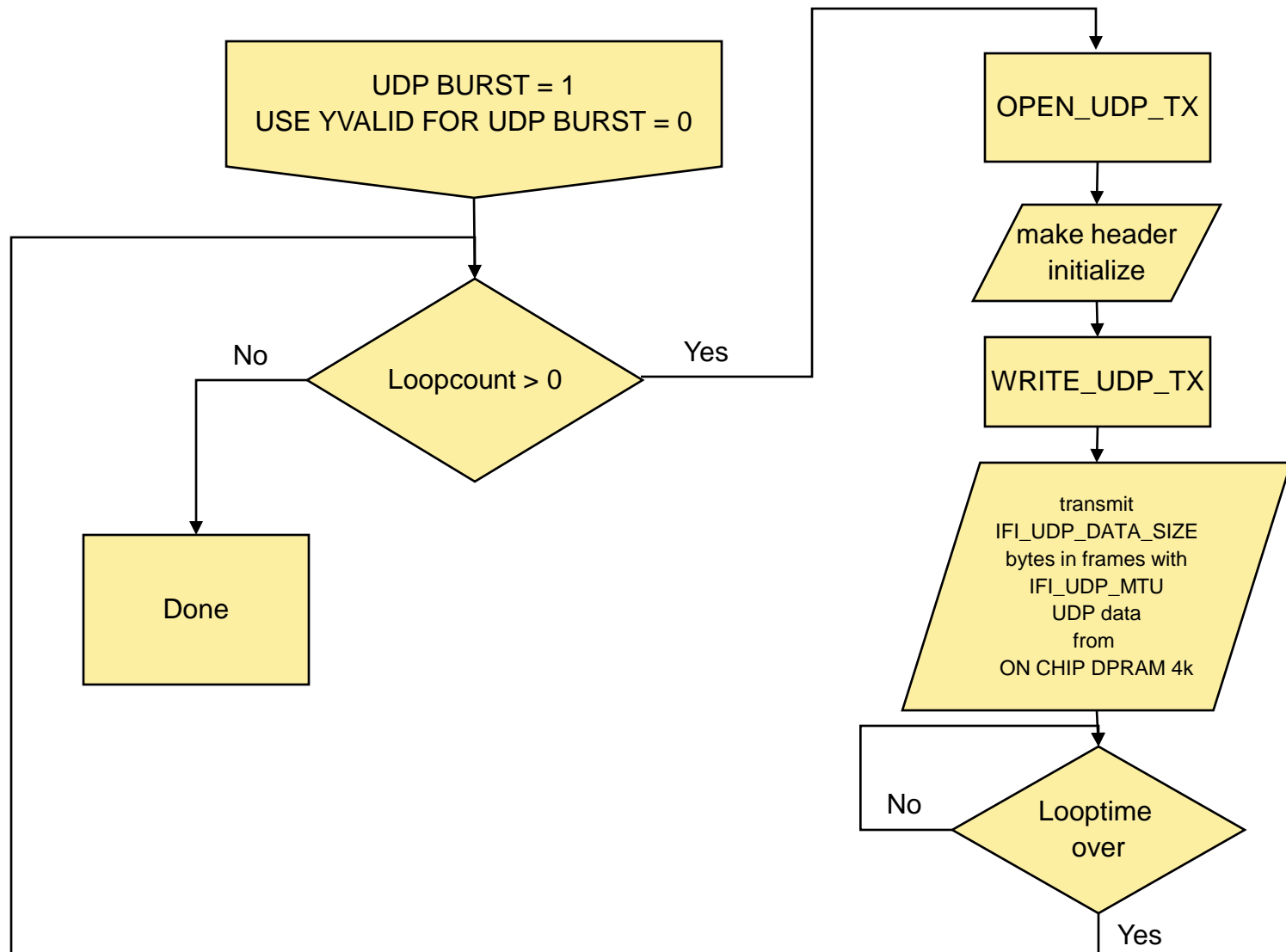
/* destination MAC-ID and IP-address
 *     MACADDR: 00:07:ed:a1:b2:c4
 *     IP: 192.168.100.44
 */
#define DMACADDR0 0          // destination MACADDR
#define DMACADDR1 07
#define DMACADDR2 0xed
#define DMACADDR3 0xa1
#define DMACADDR4 0xb2
#define DMACADDR5 0xc3

#define DIPADDR0 192        // destination IP address
#define DIPADDR1 168
#define DIPADDR2 100
#define DIPADDR3 44
#endif /* SWAPID */
```

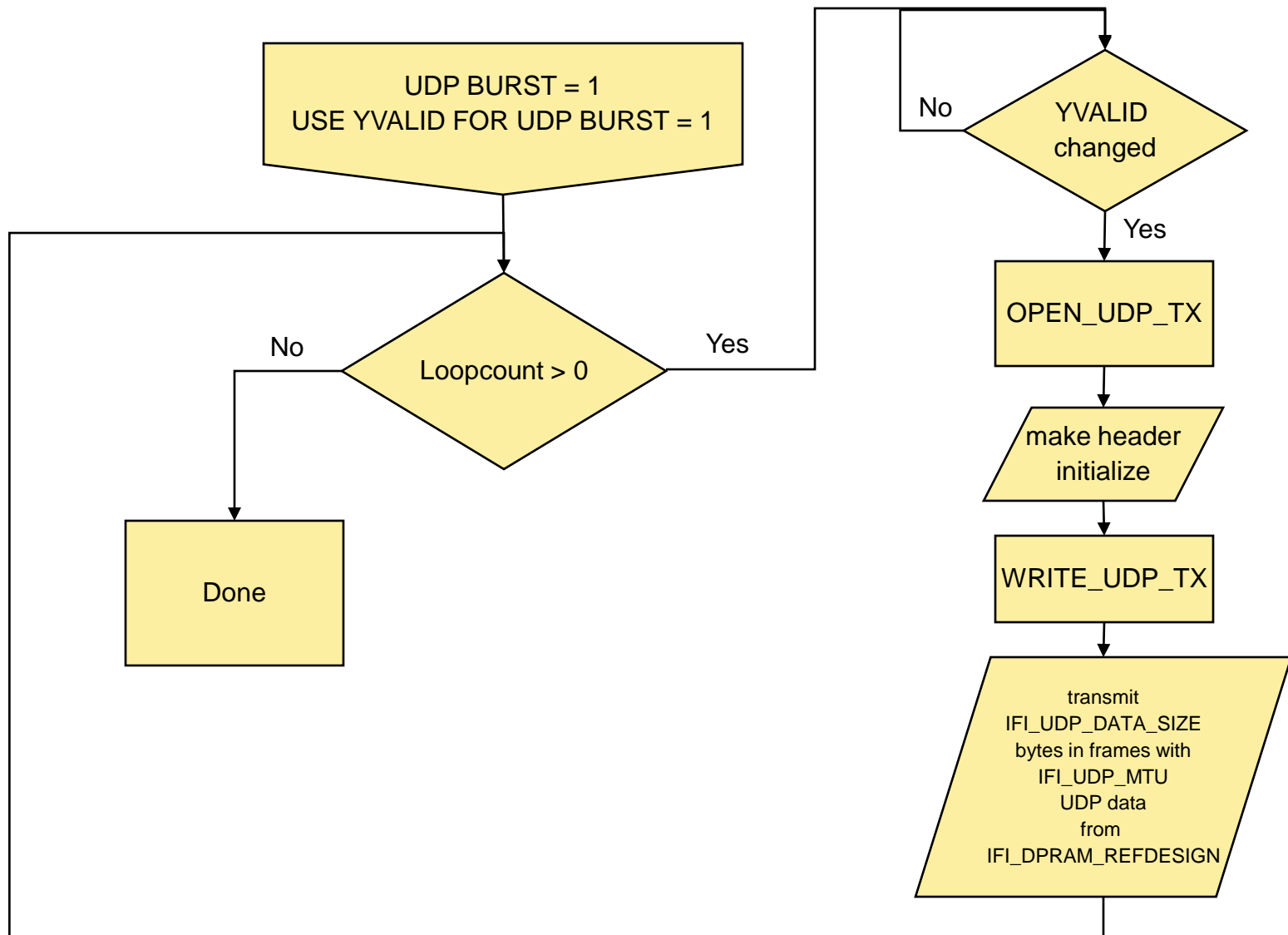
Reference Software Flow



Timer controlled



Hardware Trigger controlled



Ethernet Background

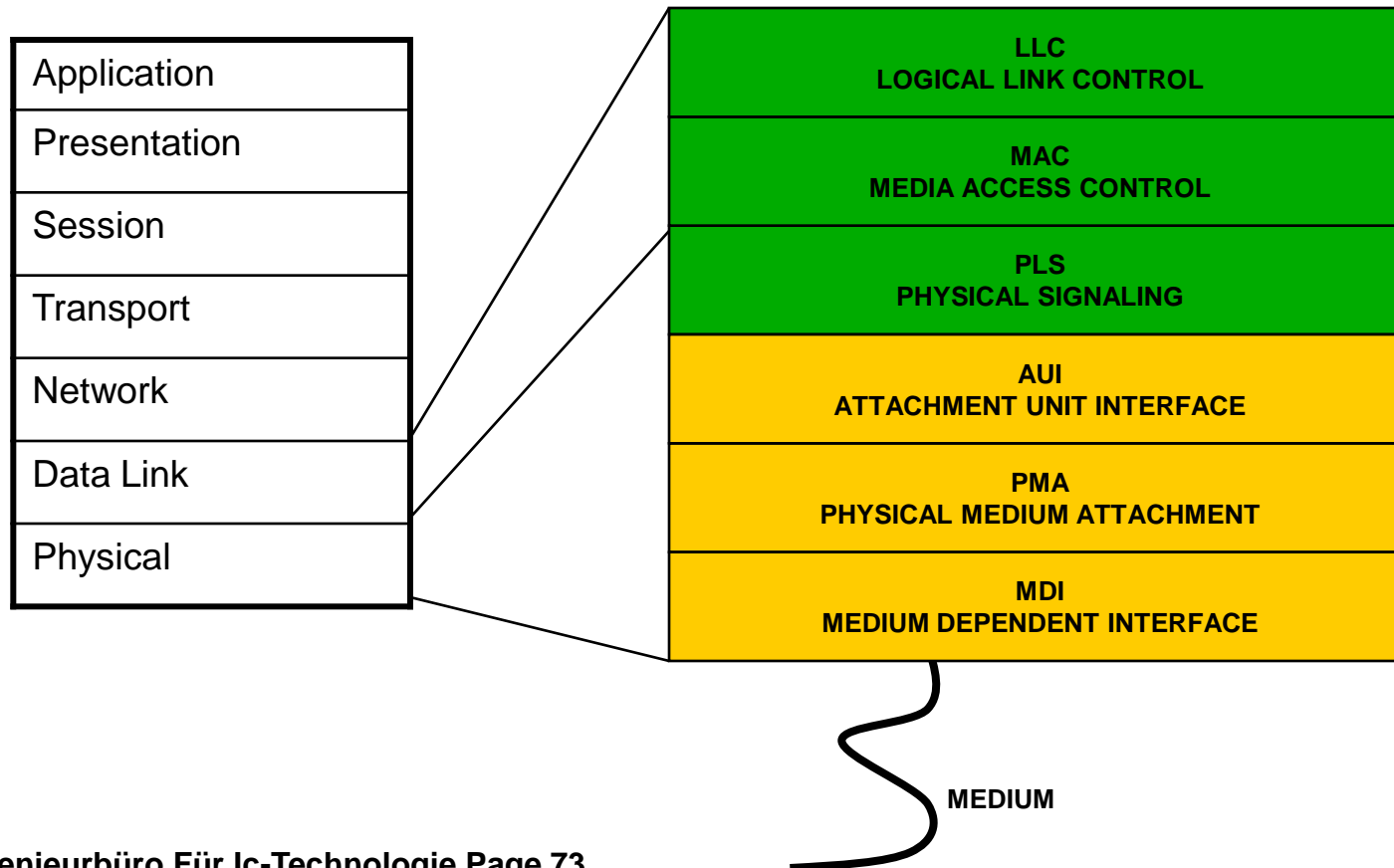
- *Protocol Stack Fundamentals*
- *UDP*
- *TCP*
- *what is theoretical possible*

Protocol Stack Fundamentals

■ The layered model

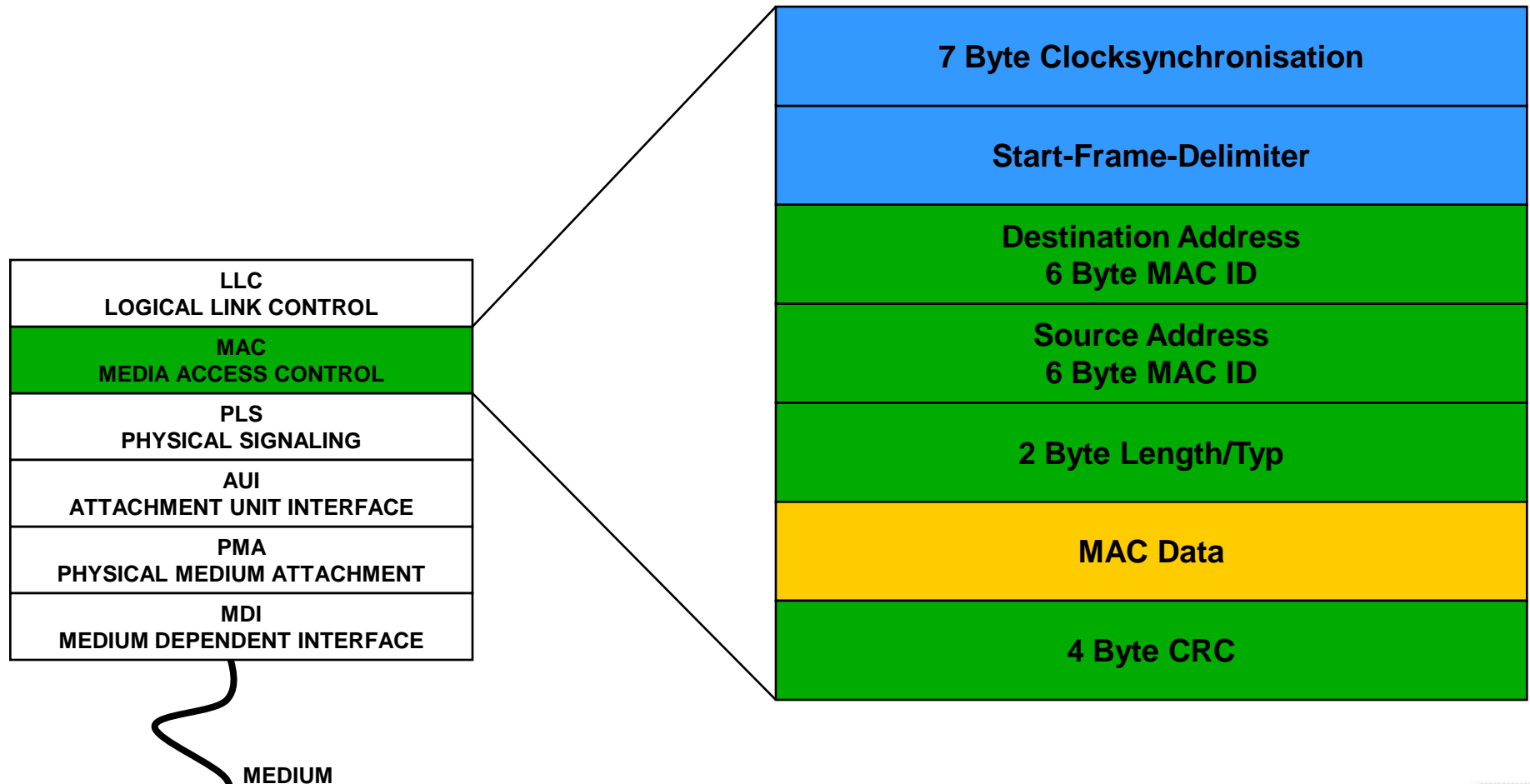
Protocol stacks are made up of layers

OSI 7 layer model of a network is a common representation



Protocol Stack Fundamentals

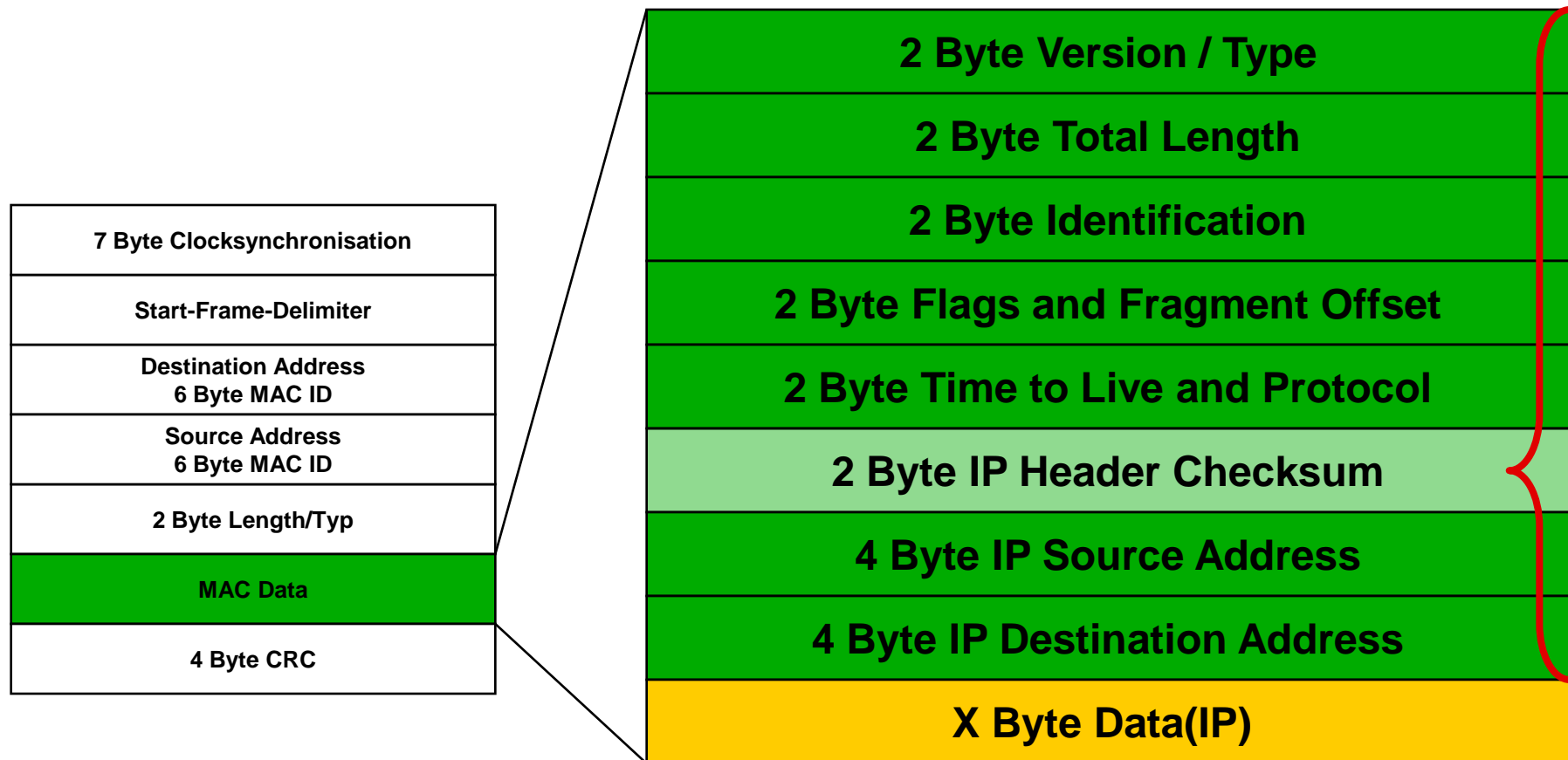
■ The MAC Frame



Protocol Stack Fundamentals

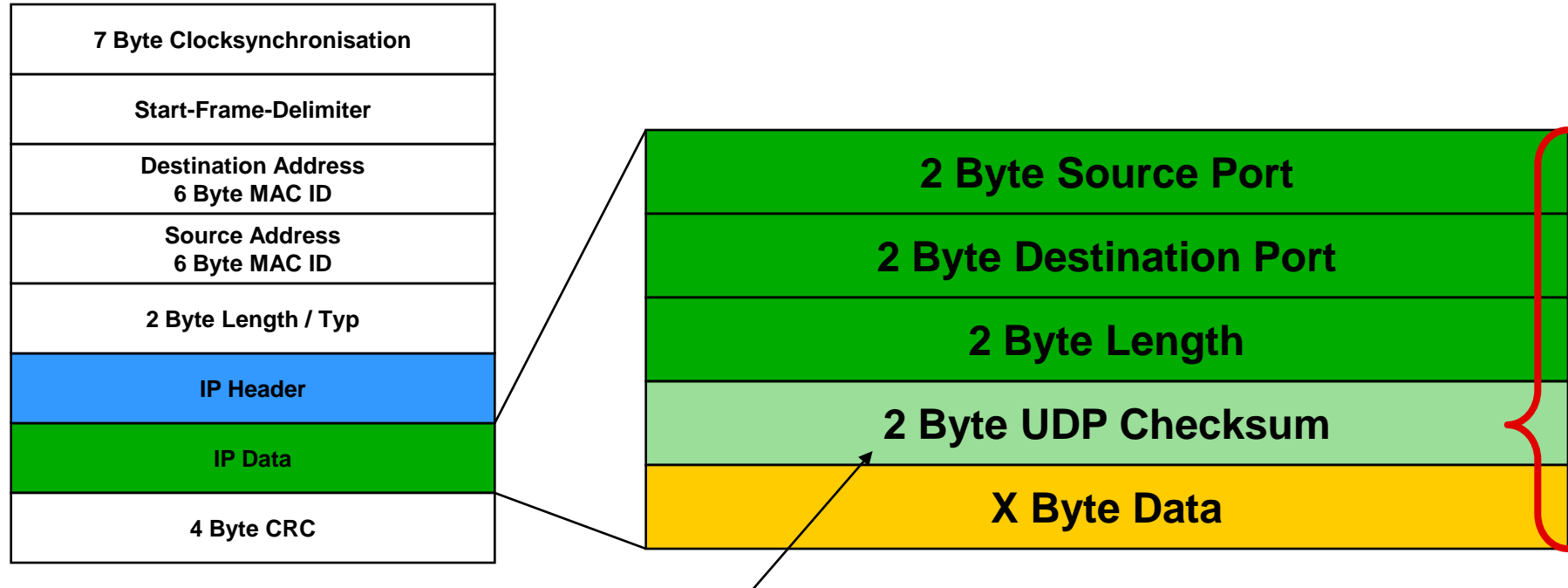
■ Example: IP for TCP/IP or UDP/IP

- Internet Protocol - IP



Protocol Stack Fundamentals

- Example: UDP for UDP/IP
 - User Datagramm Protocol - UDP



You have to process all of your data to calculate the checksum

UDP – User Datagram Protocol

- Simplest IP protocol for applications

- Limited reliability

- No guarantee of delivery
- No Handshake
- No Timeout

- Ports

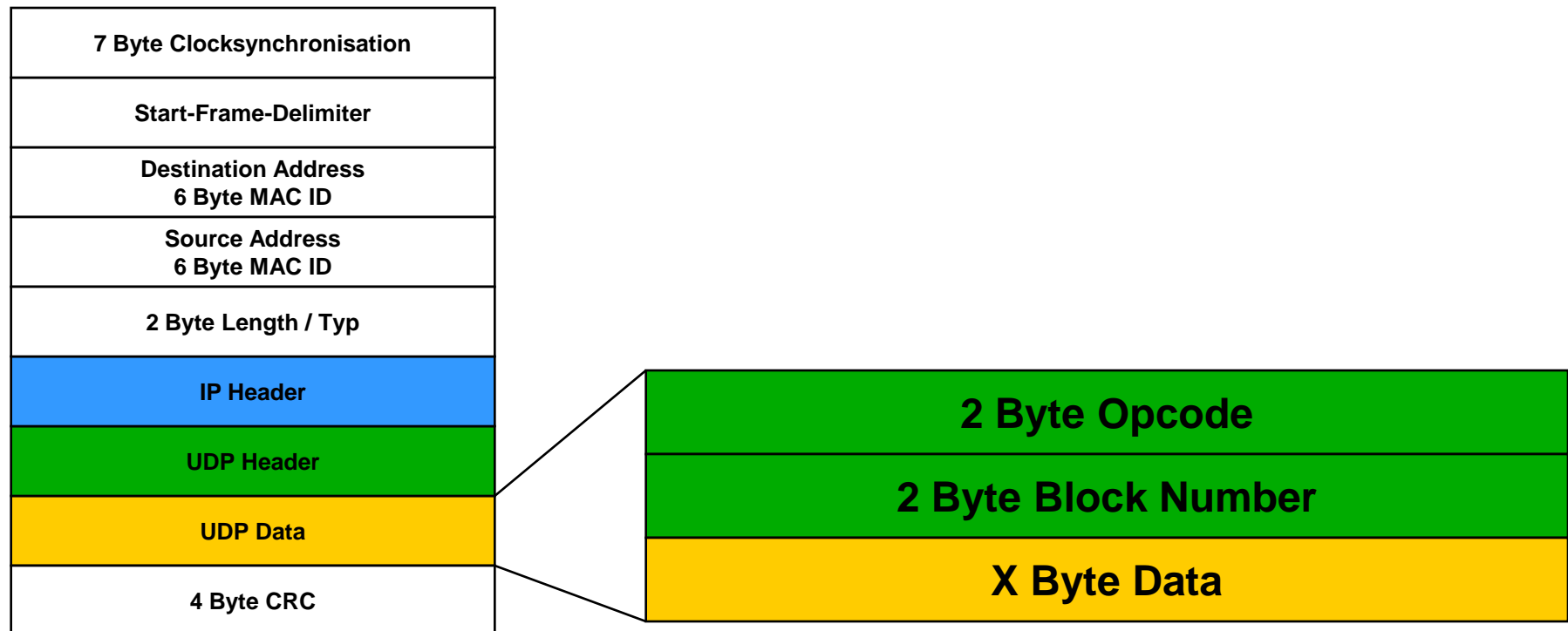
- Each application uses different port(s)

- Checksums the data

- Optional

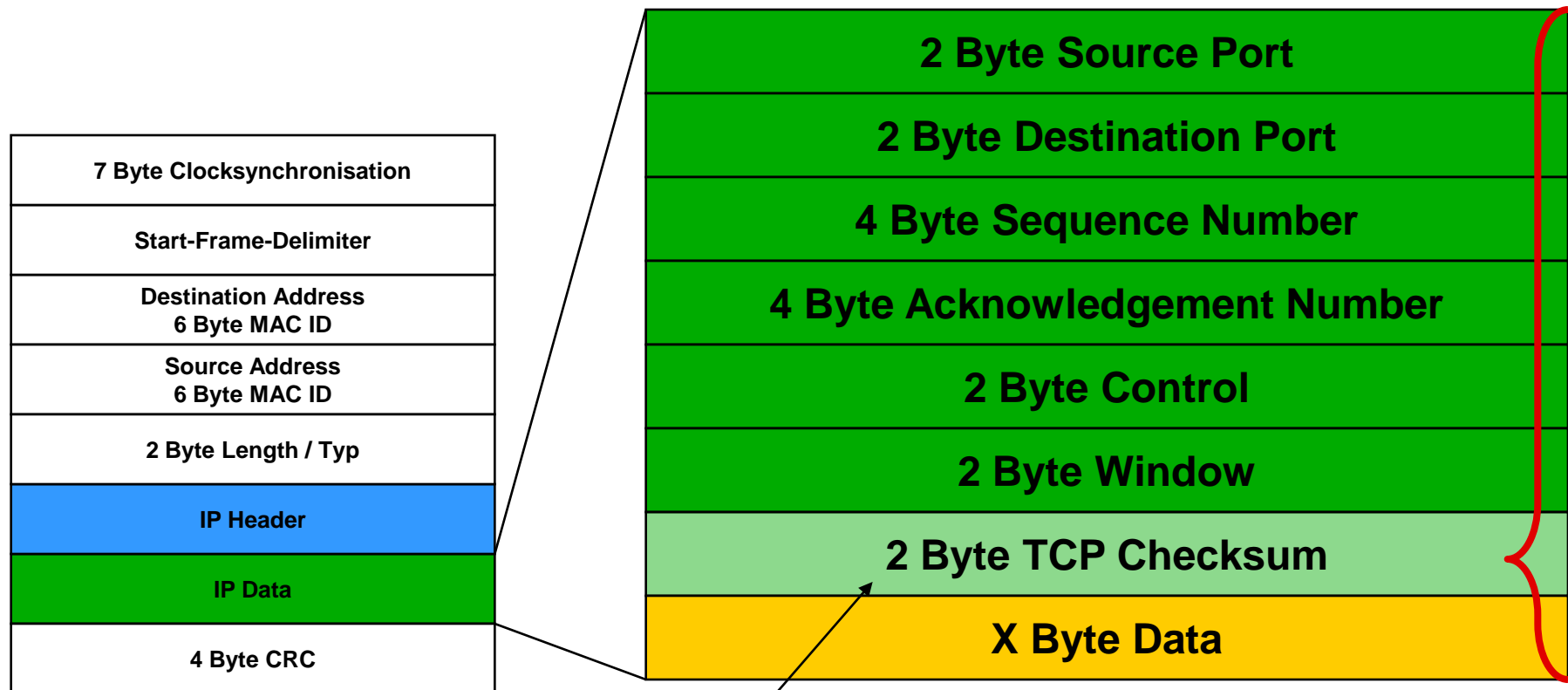
Protocol Stack Fundamentals

- Example: TFTP for UDP/IP
 - Trivial File Transfer Protocol - TFTP



Protocol Stack Fundamentals

- Example: TCP for TCP/IP
 - Transmission Control Protocol - TCP



TCP – Transmission Control Protocol

■ Reliable

- Guaranteed delivery – using sequence numbers and acknowledgements
- Protocol has to exchange sequence numbers at startup
- Checksums the data

■ Flow control

- Stacks advertise their available buffer space
- Algorithms to minimise congestion

■ Also provides ports

Other protocols

■ DHCP - Dynamic Host Configuration Protocol

- Automatically assign an IP Address

■ ARP – Address Resolution Protocol

- Get the MAC address for a known IP Address

■ ICMP – Internet Control Message Protocol

- Echo Request / Echo Reply
- Used by Ping

■ IGMP – Internet Group Management Protocol

- Used to manage multicasting

What's theoretical possible

■ 1000 Million Bit/s on the line

- 119,2 MByte/s gross amount (1 MByte = 1048576 Byte)
- without the minimum Interframe Gap (12 byte)
 - For a 1518 Byte Frame the rate drops to → 118 MByte/s
- without the Preamble, Address, Typ, CRC (another 26 byte)
 - For a 1518 Byte Frame the rate drops to → 116 MByte/s
- without the IP Header (20 byte)
 - For a 1518 Byte Frame the rate drops to → 115 MByte/s
- without the UDP Header (8 byte)
 - For a 1518 Byte Frame (1472 byte load)
 - the rate drops to → **114 MByte/s UDP-Data**
 - For a 64 Byte Frame (only 18 byte load)
 - the rate is down to → 25 MByte/s

■ → Frame size should be as large as possible

- using jumboframes when appropriate (take care for switches ..)

What's theoretical possible

■ On PC-Side

- The networkcard
 - Datapath may be limited by PCI Bus (32 Bit 33 MHz → max 132 MByte/s)
 - Interruptfilter → Latency
 - Depending from your OS switch it off or play with the settings
 - Checksum Offloading ...
 - Jumboframe support
- Harddrive performance
 - Try a ramdisc to see the possible datarates
- The OS may run into performance issues when the taskswitching rate gets to high

Revision History

Revision	Date	Description	Versioncode
1.0	Aug 2005	Initial release	0x07055010
1.1			
1.2	Nov 2005	Compatible to 5.0	
1.3	Feb 2006		
1.4	Feb 2006	Lost data when DMA fifo runs empty Issue with arp_request fixed	0x02065114
1.5	Apr 2006	Issue when writing to slow memory fixed	0x03065115
1.6	Aug 2006	Changes in frequency detection (now we use RX_CLK) Compatible to 6.0	0x08066016
1.7	Jan 2007	Enhancements like jumboframes, filters ... Compatible to 6.1	0x01076117
9.0	Apr 2009	New PHY Interfaces, SDC generation, Compatible to 9.0	0x04099090

License Agreement

PLEASE CAREFULLY REVIEW THE FOLLOWING TERMS AND CONDITIONS BEFORE USING THE IFI IP-MODULE. BY USING THIS IFI IP-MODULE AND/OR PAYING A LICENSE FEE, YOU INDICATE YOUR ACCEPTANCE OF SUCH TERMS AND CONDITIONS, WHICH CONSTITUTE THE LICENSE AGREEMENT (the "AGREEMENT") BETWEEN YOU AND IFI. IN THE EVENT THAT YOU DO NOT AGREE WITH ANY OF THESE TERMS AND CONDITIONS, DO NOT USE THE IFI IP-MODULE AND PLEASE PROMPTLY DESTROY ANY COPIES YOU HAVE MADE.

DEFINITIONS:

"Party" means either IFI or YOU.

"Specification" means IFI's technical description for the IFI IP-MODULE covered by this Agreement to the extent such technical description relates to the operation, performance, and other material attributes of the IFI IP-MODULE.

1. License to the IFI IP-MODULE:

- 1.1 Subject to the terms and conditions of this Agreement (including but not limited to YOUR payment of the license fee set forth in Paragraph 4.0), IFI grants to YOU a single-user, non-transferable, non-exclusive, and (except as specified by IFI) perpetual license to use the IFI IP-MODULE as follows. YOU may:
 - (a) design with, parameterize, compile, and route the IFI IP-MODULE;
 - (b) program Altera Devices with the IFI IP-MODULE;
 - (c) use the IFI IP-MODULE on a single computer only; and
 - (d) except as otherwise provided in Paragraph 10.2, YOU may use, distribute, sell, and or otherwise market products containing Licensed Products to any third party in perpetuity. YOU may also sublicense YOUR right to use and distribute products containing Licensed Products as necessary to permit YOUR distributors to distribute and YOUR customers to use products containing Licensed Products. YOU are expressly prohibited from using the IFI IP-MODULE to design, develop or program Non-Altera Devices .
- 1.2 YOU may make only one copy of the IFI IP-MODULE for back-up purposes only. The IFI IP-MODULE may not be copied to, installed on or used with any other computer, or accessed or otherwise used over any network, without prior written approval from IFI.
- 1.3 Any copies of the IFI IP-MODULE made by or for YOU shall include all intellectual property notices, including copyright and proprietary rights notices, appearing on such IFI IP-MODULE. Any copy or portion of the IFI IP-MODULE, including any portion merged into a design and any design or product that incorporates any portion of the IFI IP-MODULE, will continue to be subject to the terms and conditions of this Agreement.
- 1.4 The source code of the IFI IP-MODULE, and algorithms, concepts, techniques, methods and processes embodied therein, constitute trade secrets and confidential and proprietary information of IFI and its licensors and LICENSEE shall not access or use such trade secrets and information in any manner, except to the extent expressly permitted herein. IFI and its licensors retain all rights with respect to the IFI IP-MODULE, including any copyright, patent, trade secret and other proprietary rights, not expressly granted herein.

2. License Restrictions:

YOU MAY NOT USE THE IFI IP-MODULE EXCEPT AS EXPRESSLY PROVIDED FOR IN THIS AGREEMENT OR SUBLICENSE OR TRANSFER THE IFI IP-MODULE OR RIGHTS WITH RESPECT THERETO. YOU MAY NOT DECOMPILE, DISASSEMBLE, OR OTHERWISE REVERSE ENGINEER THE IFI IP-MODULE OR ATTEMPT TO ACCESS OR DERIVE THE SOURCE CODE OF THE IFI IP-MODULE OR ANY ALGORITHMS, CONCEPTS, TECHNIQUES, METHODS OR PROCESSES EMBODIED THEREIN; PROVIDED, HOWEVER, THAT IF YOU ARE LOCATED IN A MEMBER NATION OF THE EUROPEAN UNION OR OTHER NATION THAT PERMITS LIMITED REVERSE ENGINEERING NOTWITHSTANDING A CONTRACTUAL PROHIBITION TO THE CONTRARY, YOU MAY PERFORM LIMITED REVERSE ENGINEERING, BUT ONLY AFTER GIVING NOTICE TO IFI AND ONLY TO THE EXTENT PERMITTED BY THE APPLICABLE LAW IMPLEMENTING THE EU SOFTWARE DIRECTIVE OR OTHER APPLICABLE LAW NOTWITHSTANDING A CONTRACTUAL PROHIBITION TO THE CONTRARY.

3. Term:

This Agreement is effective until terminated. YOU may terminate it at any time by destroying the IFI IP-MODULE together with all copies and portions thereof in any form (except as provided below). It will also terminate immediately if YOU breach any term of this Agreement and upon conditions set forth elsewhere in this Agreement. Upon any termination of this Agreement, YOU shall destroy the IFI IP-MODULE, including all copies and portions thereof in any form (whether or not merged into a design or Licensed Product), and YOUR license and rights under this Agreement shall terminate except that YOU and YOUR customers may continue to sell and use Licensed Products which have been developed in accordance with this Agreement and shipped prior to the termination. In no event may any portions of the IFI IP-MODULE be used in development after termination. In the event of termination for any reason, the rights, obligations, and restrictions under Paragraphs 2, 4, 9, and 10 shall survive termination of this Agreement.

4. Payment:

In consideration of the license granted by IFI under Paragraph 1.1 and other rights granted under this Agreement, YOU shall pay the license fee for the IFI IP-MODULE that has been specified by IFI. Such payment shall, as directed by IFI, be made directly to IFI. YOU shall pay all taxes and duties associated with this Agreement, other than taxes based on IFI's income.

5. Maintenance and Support:

IFI shall, but only until the date, in the format YYYY.MM, provided in the license file for a IFI IP-MODULE ("Maintenance Expiration Date"):

- 5.1 use commercially reasonable efforts to provide YOU with fixes to defects in the IFI IP-MODULE that cause the IFI IP-MODULE not to conform substantially to the Specifications and that are diagnosed as such and replicated by IFI;
- 5.2 provide YOU with fixes and other updates to the IFI IP-MODULE that IFI chooses to make generally available to its customers without a separate charge; and
- 5.3 respond by telephone or email to inquiries from YOU.

6. Limited Warranties and Remedies:

- 6.1 IFI represents and warrants that, until the Maintenance Expiration Date ("Warranty Period"), the IFI IP-MODULE will substantially conform to the Specifications. YOUR sole remedy, and IFI's sole obligation, for a breach of this warranty shall be (a) for IFI to use commercially reasonable efforts to remedy the nonconformance, or (b) if IFI is unable substantially to remedy the nonconformance, for YOU to receive a refund of license fees paid during the previous one (1) year for the defective IFI IP-MODULE. If YOU receive such a refund, YOU agree that YOUR license and rights under this Agreement for the defective IFI IP-MODULE shall immediately terminate and YOU agree to destroy the defective IFI IP-MODULE, including all copies thereof in any form and any portions thereof merged into a design or product, and to certify the same to IFI.
- 6.2 The foregoing warranties apply only to IFI IP-MODULES delivered by IFI. The warranties are provided only to YOU, and may not be transferred or extended to any third party, and apply only during the Warranty Period for claims of breach reported (together with evidence thereof) during the Warranty Period. YOU shall provide IFI with such evidence of alleged non-conformities or defects as IFI may request, and IFI shall have no obligation to remedy any non-conformance or defect it cannot replicate. The warranties do not extend to any IFI IP-MODULE which have been modified by anyone other than IFI.

7. Representation:

Each party represents that it has the right to enter into this Agreement and to perform its obligations hereunder.

8. Indemnification:

- 8.1 Expressly subject to Section 9, IFI shall defend YOU against any proceeding brought by a third party to the extent based on a claim that the IFI IP-MODULE, as delivered by IFI and as used in accordance with this Agreement, infringes a third party's copyright, trade secret, patent, or any other intellectual property right ("IP right"), and pay any damages awarded in the proceeding as a result of the claim (or pay any amount agreed to by IFI as part of a settlement of the claim), provided that IFI shall have no liability hereunder unless YOU notify IFI promptly in writing of any such proceeding or claim, give IFI sole and complete authority to control the defense and settlement of the proceeding or claim, and provide IFI with any information, materials, and other assistance requested by IFI.
- 8.2 In the event of any such claim or proceeding or threat thereof, IFI may (and, in the event any such claim or proceeding results in the issuance of an injunction by a court of competent jurisdiction prohibiting YOU from using the IFI IP-MODULE, IFI shall), at its option and expense and subject to the limitations of Paragraph 9, seek a license to permit the continued use of the affected IFI IP-MODULE or use commercially reasonable efforts to replace or modify the IFI IP-MODULE so that the replacement or modified version is non-infringing or has a reduced likelihood of infringement, provided that the replacement or modified version has functionality comparable to that of the original. If IFI is unable reasonably to obtain such license or provide such replacement or modification, IFI may terminate YOUR license and rights with respect to the affected IFI IP-MODULE, in which event YOU shall return to IFI the affected IFI IP-MODULE, including all copies and portions thereof in any form (including any portions thereof merged into a design or product), and certify the same to IFI, and IFI shall refund the license fee paid by YOU for the affected IFI IP-MODULE.
- 8.3 IFI shall have no liability or obligation to YOU hereunder for any infringement or claim based on or resulting from (a) the combination or use of the IFI IP-MODULE with other products or components; (b) modification of the IFI IP-MODULE by anyone other than IFI, (c) the use of other than the most recent version of the IFI IP-MODULE if the infringement or claim would have been avoided (or the likelihood thereof reduced) by use of the most recent version; (d) requirements specified by YOU; (e) use of the IFI IP-MODULE in any way not contemplated under this Agreement; or (f) any use of the IFI IP-MODULE, to the extent that IFI has indicated in the applicable Specification that third-party licenses 8.3a may be required to use such IFI IP-MODULE if LICENSEE has not obtained the necessary third-party licenses.
- 8.4 The provisions of this Paragraph 8 state the entire liability and obligations of IFI, and YOUR sole and exclusive rights and remedies, with respect to any proceeding or claim relating to infringement of copyright, trade secret, patent, or any other intellectual property right.

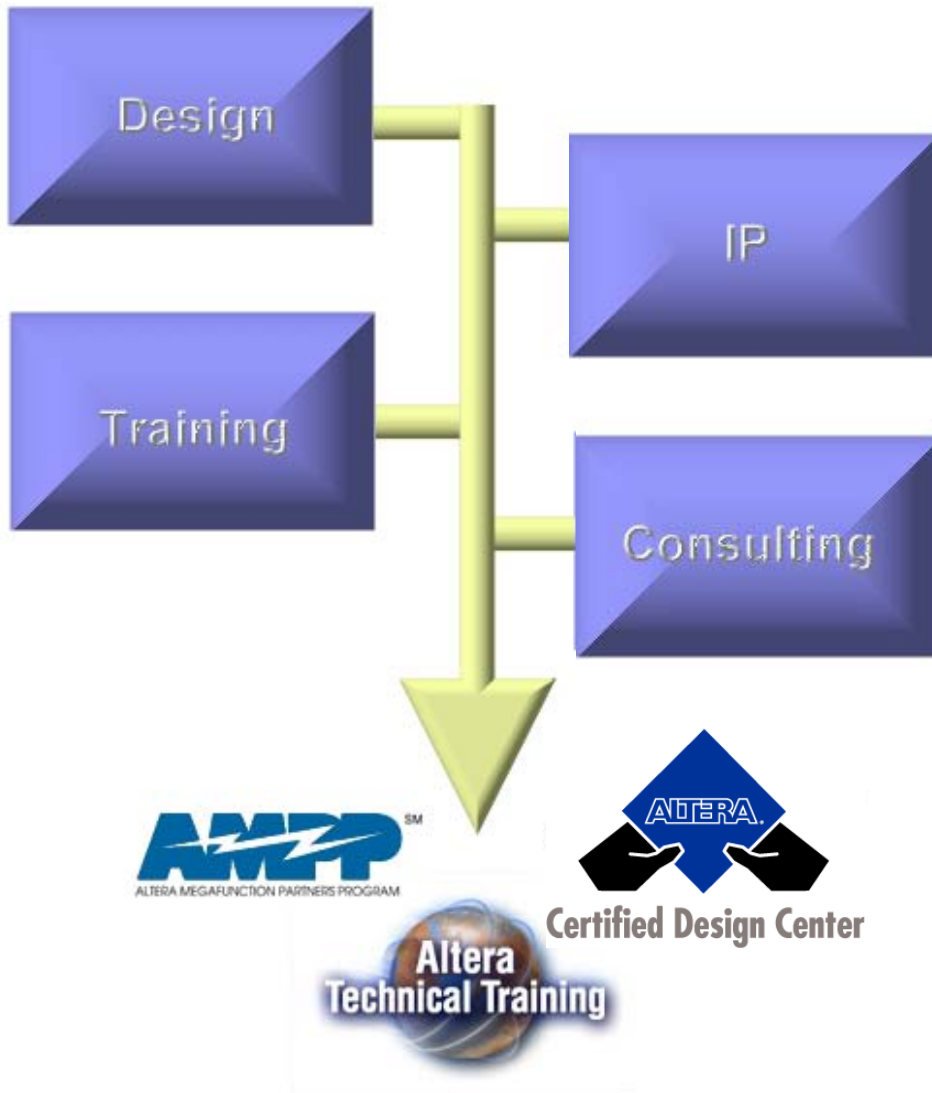
LIMITATIONS OF LIABILITY

- 9.1 In no event shall the aggregate liability of IFI relating to this Agreement or the subject matter hereof under any legal theory (whether in tort, contract or otherwise), including any liability under Paragraph 8 or for any loss or damages directly or indirectly suffered by YOU relating to the IFI IP-MODULE, exceed the aggregate amount of the license fees paid by YOU in the previous one (1) year under this Agreement.
- 9.2 IN NO EVENT SHALL IFI BE LIABLE UNDER ANY LEGAL THEORY, WHETHER IN TORT, CONTRACT OR OTHERWISE (a) FOR ANY LOST PROFITS, LOST REVENUE OR LOST BUSINESS, (b) FOR ANY LOSS OF OR DAMAGES TO OTHER SOFTWARE OR DATA, OR (c) FOR ANY INCIDENTAL, INDIRECT, CONSEQUENTIAL OR SPECIAL DAMAGES RELATING TO THIS AGREEMENT OR THE SUBJECT MATTER HEREOF, INCLUDING BUT NOT LIMITED TO THE DELIVERY, USE, SUPPORT, OPERATION OR FAILURE OF THE MEGACORE LOGIC IFI IP-MODULE, EVEN IF IFI HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH LIABILITY.

10. General:

- 10.1 YOU may not sublicense, assign, or transfer this license, or disclose any trade secrets embodied in the IFI IP-MODULE, except as expressly provided in this Agreement. Any attempt to sublicense, assign, or otherwise transfer without prior written approval of the other party any of the rights, duties, or obligations hereunder is void.
- 10.2 This Agreement is entered into for the benefit of IFI and its licensors and all rights granted to YOU and all obligations owed to IFI shall be enforceable by IFI.
- 10.3 If YOU have any questions concerning this Agreement, including software maintenance or warranty service, YOU should contact IFI Ing.Büro Für Ic-Technologie, P. Riekert & F. Sprenger, Kleiner Weg 3, 97877 Wertheim, Germany.
- 10.4 YOU agree that the validity and construction of this Agreement, and performance hereunder, shall be governed by the laws of German jurisdictions, without reference to conflicts of laws principles. YOU agree to submit to the exclusive jurisdiction of the courts in Germany, for the resolution of any dispute or claim arising out of or relating to this Agreement. The Parties hereby agree that the Party who does not prevail with respect to any dispute, claim, or controversy relating to this Agreement shall pay the costs actually incurred by the prevailing Party, including any attorneys' fees.
- 10.5 In the event that any provision of this Agreement is held by a court of competent jurisdiction to be legally ineffective or unenforceable, such provision shall be reformed only to the extent necessary to make it enforceable and the validity of the remaining provisions shall not be affected.
- 10.6 The article headings throughout this Agreement are for reference purposes only and the words contained therein shall not be construed as a substantial part of this Agreement and shall in no way be held to explain, modify, amplify, or aid in the interpretation, construction or meaning of the provisions of this Agreement.
- 10.7 BY USING THE IFI IP-MODULE, YOU AND IFI ACKNOWLEDGE THAT YOU AND IFI HAVE READ THIS AGREEMENT, UNDERSTAND IT, AND AGREE TO BE BOUND BY ITS TERMS AND CONDITIONS. YOU AND IFI FURTHER AGREE THAT IT IS THE COMPLETE AND EXCLUSIVE STATEMENT OF THE AGREEMENT BETWEEN YOU AND IFI, WHICH SUPERSEDES ANY PROPOSAL OR PRIOR AGREEMENT, ORAL OR WRITTEN, AND ANY OTHER COMMUNICATIONS BETWEEN YOU AND IFI RELATING TO THE SUBJECT MATTER OF THIS AGREEMENT, UNLESS YOU HAVE A SEPARATE LICENSE SIGNED BY AN AUTHORIZED IFI REPRESENTATIVE.

IFI Products and Services



■ Dedicated to Altera since 1985

■ IPs

- GMACII
- CAN2.0B
- USB
- MediaLB
- IEEE 1588

■ Training Classes

- IFI – QUARTUS
- IFI – ALTERA Expert
- IFI – VHDL
- IFI – Nios®II

■ Design services for all ALTERA devices

■ Consulting

INGENIEURBÜRO **F**ÜR **I**C-TECHNOLOGIE

Peter Riekert & Franz Sprenger

Kleiner Weg 3

97877 Wertheim Germany

Tel.: (+49)9342 / 9608-0

Fax: (+49)9342 / 5381

eMail: ifi @ ifi-pld.de

<http://www.ifi-pld.de>