96B Quad Ethernet Mezzanine

Jeff Johnson

Nov 25, 2020

DATASHEET

CHAPTER

ONE

DESCRIPTION

The 96B Quad Ethernet Mezzanine is an add-on/expansion board for SoC based development platforms designed to the 96Boards specification. The mezzanine card has 4x Texas Instruments DP83867 Gigabit Ethernet PHYs to provide 4 ports of gigabit Ethernet connectivity to the carrier development platform.

Fig. 1.1: 96B Quad Ethernet Mezzanine on the Ultra96

CHAPTER

TWO

FEATURES

- 4x TI DP83867 Gigabit Ethernet PHYs
- Quad Ethernet RJ45 with magnetics
- Power and reset pushbuttons
- Low-speed expansion connector for stacking
- Supports the Avnet Ultra96 v1 and v2 dev platforms
- Example designs for Vivado
- Standalone and PetaLinux example designs

CHAPTER

THREE

WHERE TO BUY

The mezzanine card can be purchased from Opsero's online shop at the link below:

96B Quad Ethernet Mezzanine order page

3.1 Pin Configuration

The 96B Quad Ethernet Mezzanine has both the low-speed and high-speed expansion connectors as defined by the 96Boards Consumer Edition specification. The following tables define the pinout of those connectors for this mezzanine card.

3.1.1 Low-speed expansion header

The low-speed expansion header connects the main power supply (SYS_DCIN) to the mezzanine card and it also provides I/Os that are used for the MDIO bus, the PHY resets and the "power good" signals from the mezzanine card's switching regulators.

Only 9 I/O pins of the low-speed expansion header are used by the 96B Quad Ethernet Mezzanine card. The others are directly passed through to the expansion socket on the top side of the board which can be used for stacking a second mezzanine card. See the following section for the pinout of the low-speed expansion socket.

The mezzanine card does not draw power from the +5V pin (37). This pin is directly passed through to the expansion socket on the top side of the board.

The mezzanine card does not draw power from the 1.8V supply pin (35), but it does use this pin to detect power-up of the carrier board. To supply +1.8V power to the Ethernet PHYs, the mezzanine card generates it's own +1.8VDC supply using an on-board switching regulator that is powered by SYS_DCIN (the main supply). The +1.8VDC that is generated on the mezzanine card is also passed through to the top-side expansion socket to provide power to a stacked mezzanine if required.

96Boards pin name	Pin	Description	96Boards pin name	Pin	Description
GND	1	Ground	GND	2	Ground
UART0_CTS	3	Passed through	PWR_BTN_N	4	Passed through
UART0_TXD	5	Passed through	RST_BTN_N	6	Passed through
UART0_RXD	7	Passed through	SPI0_SCL	8	Passed through
UART0_RTS	9	Passed through	SPI0_DIN	10	Passed through
UART1_TXD	11	Passed through	SPI0_CS	12	Passed through
UART1_RXD	13	Passed through	SPI0_DOUT	14	Passed through
I2C0_SCL	15	Passed through	PCM_FS	16	Passed through
I2C0_SDA	17	Passed through	PCM_CLK	18	Passed through
I2C1_SCL	19	Passed through	PCM_DO	20	Passed through
I2C1_SDA	21	Passed through	PCM_DI	22	Passed through
GPIO-A	23	Passed through	GPIO-B	24	Passed through
GPIO-C	25	Passed through	GPIO-D	26	POWER GOOD 1.0V (1.8V logic)
GPIO-E	27	POWER GOOD 2.5V (1.8V logic)	GPIO-F	28	POWER GOOD 1.8V (1.8V logic)
GPIO-G	29	Port 0 PHY reset (active low)	GPIO-H	30	Port 1 PHY reset (active low)
GPIO-I	31	Port 2 PHY reset (active low)	GPIO-J	32	Port 3 PHY reset (active low)
GPIO-K	33	MDIO data signal	GPIO-L	34	MDC clock signal
+1V8	35	+1.8V supply from dev platform	SYS_DCIN	36	Main power supply
+5V	37	+5.0V supply from dev platform Passed through	SYS_DCIN	38	Main power supply
GND	39	Ground	GND	40	Ground

3.1.2 Low-speed expansion socket

The low-speed expansion socket is on the top-side of the mezzanine card and can be used for stacking a second mezzanine card. It provides access to all of the I/O that the 96B Quad Ethernet mezzanine does not use, and all of the power supplies.

In the table below, all of the pins with the description "Passed through" can be used by the stacked mezzanine card. The specific usage of the pins will depend on the development platform being used.

Note: The +1.8V power supply pin (35) is connected to the +1.8V that is generated by a switching regulator on the 96B Quad Ethernet Mezzanine, it is not passed through from the development platform. This allows the stacked mezzanine to draw 100mA or more from the supply.

96Boards pin name	Pin	Description	96Boards pin name	Pin	Description
GND	1	Ground	GND	2	Ground
UART0_CTS	3	Passed through	PWR_BTN_N	4	Passed through
UART0_TXD	5	Passed through	RST_BTN_N	6	Passed through
UART0_RXD	7	Passed through	SPI0_SCL	8	Passed through
UART0_RTS	9	Passed through	SPI0_DIN	10	Passed through
UART1_TXD	11	Passed through	SPI0_CS	12	Passed through
UART1_RXD	13	Passed through	SPI0_DOUT	14	Passed through
I2C0_SCL	15	Passed through	PCM_FS	16	Passed through
I2C0_SDA	17	Passed through	PCM_CLK	18	Passed through
I2C1_SCL	19	Passed through	PCM_DO	20	Passed through
I2C1_SDA	21	Passed through	PCM_DI	22	Passed through
GPIO-A	23	Passed through	GPIO-B	24	Passed through
GPIO-C	25	Passed through	GPIO-D	26	Not connected
GPIO-E	27	Not connected	GPIO-F	28	Not connected
GPIO-G	29	Not connected	GPIO-H	30	Not connected
GPIO-I	31	Not connected	GPIO-J	32	Not connected
GPIO-K	33	Not connected	GPIO-L	34	Not connected
+1V8	35	+1.8V supply from 96B Eth mezzanine	SYS_DCIN	36	Main power supply Passed through
+5V	37	+5.0V supply from dev platform Passed through	SYS_DCIN	38	Main power supply Passed through
GND	39	Ground	GND	40	Ground

3.1.3 High-speed expansion connector

The high-speed expansion connector routes the SGMII input (Soc-to-PHY) and output (PHY-to-SoC) signals to the development platform. It also routes the SGMII 625MHz clock (input to SoC), which is generated by the PHY connected to port 3, and is typically used by the SGMII receiver.

Also routed through the high-speed connector are 2 configurable outputs of the DP83867 PHYs called “GPIO0” and “GPIO1”. These can be used for start-of-packet detection, loss of sync detection, and receive error detection among other things. Please refer to the datasheet of the DP83867 for more detailed information on these pins and their function.

96Boards pin name	Pin	Description	96Boards pin name	Pin	Description
SD_DAT0/SPI1_DOUT	1	Not used	CSI0_C+	2	Port 0 SGMII output data+
SD_DAT1	3	Not used	CSI0_C-	4	Port 0 SGMII output data-
SD_DAT2	5	Not used	GND	6	Ground
SD_DAT3/SPI1_CS	7	Not used	CSI0_D0+	8	Port 1 SGMII output data+
SD_SCLK/SPI1_SCLK	9	Not used	CSI0_D0-	10	Port 1 SGMII output data-
SD_CMD/SPI1_DIN	11	Not used	GND	12	Ground
GND	13	Ground	CSI0_D1+	14	Port 1 GPIO1 (1.2V output)
CLK0/CSI0_MCLK	15	Not used	CSI0_D1-	16	Port 1 GPIO0 (1.2V output)
CLK1/CSI1_MCLK	17	Not used	GND	18	Ground
GND	19	Ground	CSI0_D2+	20	Port 0 SGMII input data+
DSI_CLK+	21	Port 3 SGMII input data+	CSI0_D2-	22	Port 0 SGMII input data-
DSI_CLK-	23	Port 3 SGMII input data-	GND	24	Ground
GND	25	Ground	CSI0_D3+	26	Port 1 SGMII input data+
DSI_D0+	27	Port 2 SGMII output data+	CSI0_D3-	28	Port 1 SGMII input data-
DSI_D0-	29	Port 2 SGMII output data-	GND	30	Ground
GND	31	Ground	I2C2_SCL	32	Not used
DSI_D1+	33	Port 0 GPIO1 (1.2V output)	I2C2_SDA	34	Not used
DSI_D1-	35	Port 0 GPIO0 (1.2V output)	I2C3_SCL	36	Not used
GND	37	Ground	I2C3_SDA	38	Not used
DSI_D2+	39	Port 3 GPIO0 (1.2V output)	GND	40	Ground
DSI_D2-	41	Port 3 GPIO1 (1.2V output)	CSI1_D0+	42	SGMII 625MHz clock+

Continued on next page

Table 3.1 – continued from previous page

96Boards pin name	Pin	Description	96Boards pin name	Pin	Description
GND	43	Ground	CSI1_D0-	44	SGMII 625MHz clock-
DSI_D3+	45	Port 2 SGMII input data+	GND	46	Ground
DSI_D3-	47	Port 2 SGMII input data-	CSI1_D1+	48	Port 2 GPIO1 (1.2V output)
GND	49	Ground	CSI1_D1-	50	Port 2 GPIO0 (1.2V output)
USB_D+	51	Not used	GND	52	Ground
USB_D-	53	Not used	CSI1_C+	54	Port 3 SGMII output data+
GND	55	Ground	CSI1_C-	56	Port 3 SGMII output data-
HSIC_STR	57	Not used	GND	58	Ground
HSIC_DATA	59	Not used	RESERVED	60	Not used

3.2 Specifications

3.2.1 Recommended Operating Conditions

		MIN	TYP	MAX	UNIT
Supply voltage	SYS_DCIN	+8	+12	+17	V
Output current	+1V8 (pin 35) Low-speed expansion socket	0		200	mA

3.2.2 Power Consumption

The specifications below refer to the total power consumption of the mezzanine card and the carrier board combined. It is important to note that the use of the mezzanine will affect the power consumption of the SoC on the carrier board. This is due to the peripherals and IP that must be enabled on the SoC to interface with the Ethernet PHYs. Also note that the total power consumption is dependent on the ambient temperature and channel utilization.

Ultra96-v1

	SYS_DCIN	UTILIZATION	MIN	TYP	MAX	UNIT
Current draw	16 VDC	100%		510		mA
	12 VDC	100%		645		mA
	8 VDC	100%		935		mA

- Tests performed at ambient temperature of 25 degrees C
- Tests performed using IP in the FPGA to generate the Ethernet packets

Ultra96-v2

	SYS_DCIN	UTILIZATION	MIN	TYP	MAX	UNIT
Current draw	16 VDC	100%		550		mA
	12 VDC	100%		710		mA
	8 VDC	100%		1050		mA

- Tests performed at ambient temperature of 25 degrees C
- Tests performed using IP in the FPGA to generate the Ethernet packets

3.2.3 Reset Timing

When hardware resetting the PHYs, we recommend using this timing:

1. Hold the RESET_N signal LOW for 10ms
2. Release the RESET_N signal (HIGH) and wait for 5ms

3.2.4 MDIO Timing

- The maximum MDC frequency supported by the DP83867 PHY is 25MHz.

3.2.5 DP83867 Electrical and Timing

For electrical specs and timing related to the DP83867 signals listed below, please refer to the DP83867 datasheet:

- Reset
- SGMII
- GPIO0 and GPIO1
- MDIO
- Start-of-Frame detect

3.2.6 Certifications

- RoHS
- CE

3.3 Detailed Description

3.3.1 Hardware Overview

The figure below illustrates the various hardware components that are located on the top-side of the 96B Quad Ethernet Mezzanine card.

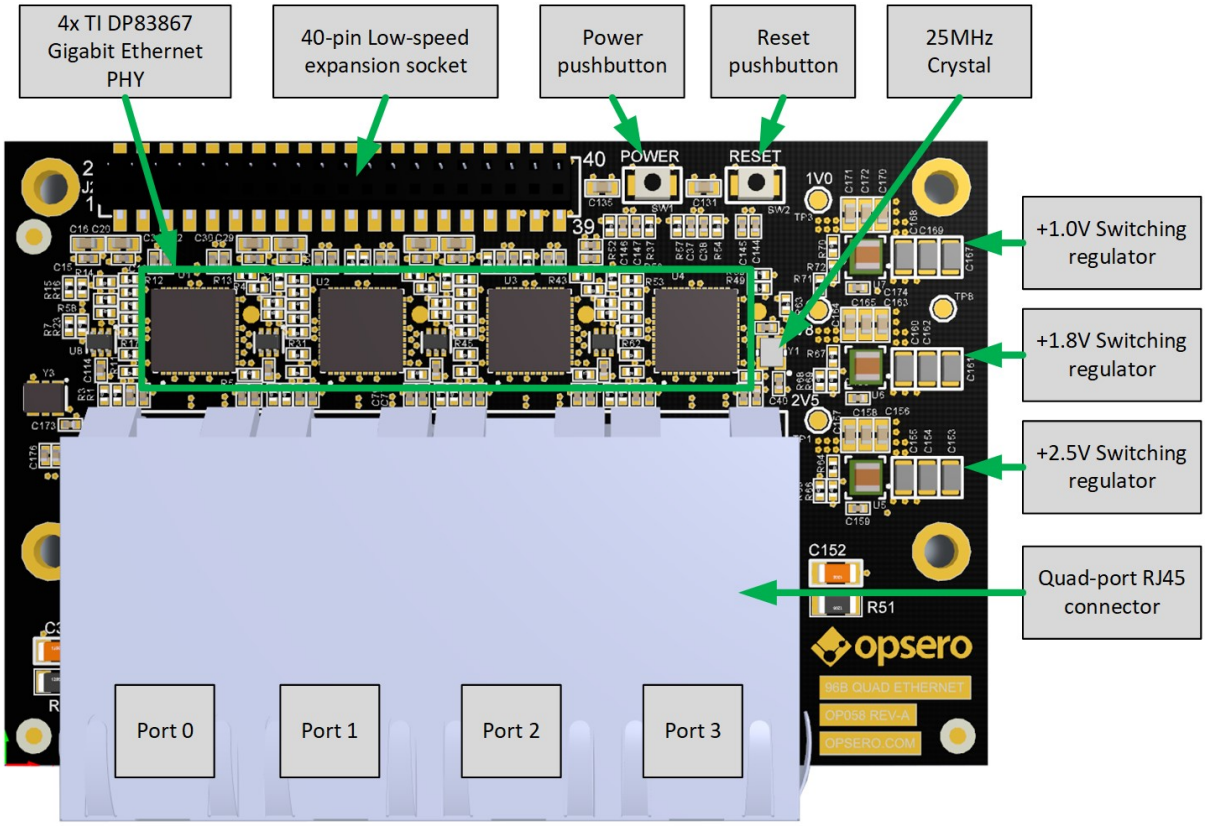

Fig. 3.1: Mezzanine card labelled top-side

The main components on the top-side of the mezzanine card are:

- 4x TI DP83867 Gigabit Ethernet PHYs
- 40-pin low-speed expansion socket for stacking a second mezzanine
- Power and reset pushbuttons
- 25MHz crystal
- Switching regulators for +1.0V, +1.8V and +2.5V
- Quad-port RJ45 connector

The figure below illustrates the various hardware components that are located on the bottom-side of the mezzanine card.

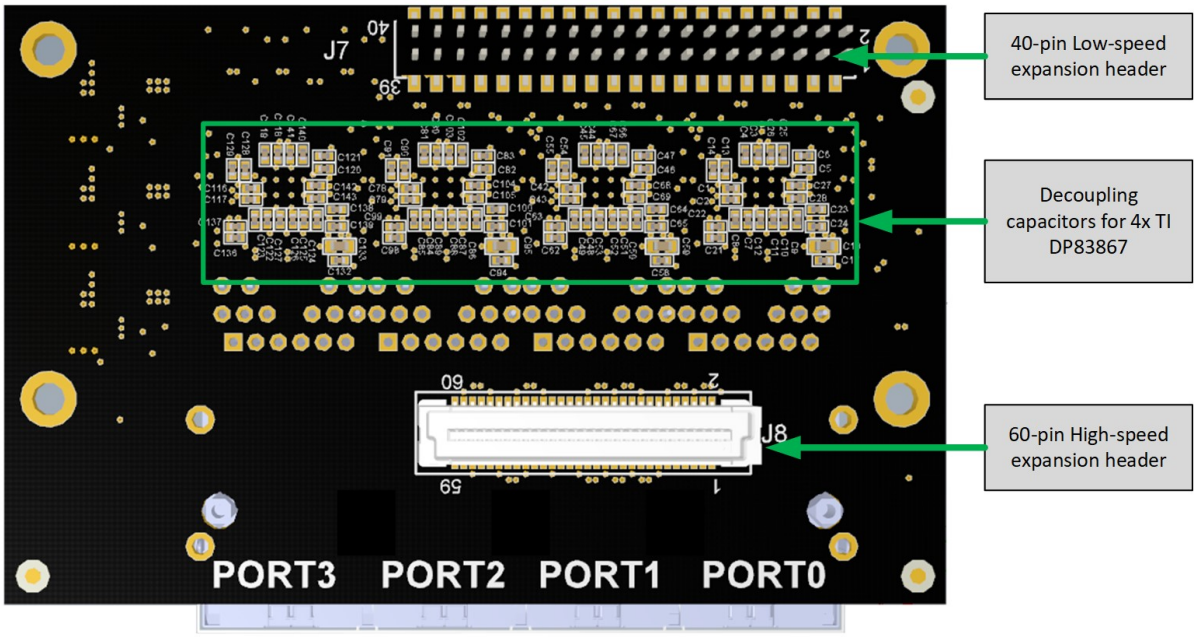

Fig. 3.2: Mezzanine card labelled bottom-side

The main components on the bottom-side of the mezzanine card are:

- 40-pin low-speed expansion header
- Decoupling capacitors for the DP83867 Ethernet PHYs
- 60-pin high-speed expansion header

3.3.2 TI DP83867 Gigabit Ethernet PHY

There are 4x TI DP83867 Gigabit Ethernet PHYs on the mezzanine card, one for each of the four Gigabit Ethernet ports. For interfacing with a MAC, the DP83867 has both RGMII and SGMII interfaces, however the mezzanine card only uses the SGMII interface of each PHY. The DP83867 is designed for low-power, it has low-latency and it provides IEEE 1588 Start of Frame Detection. For more specific information on the DP83867, please refer to the datasheet.

In this documentation, we will refer to the PHYs as PHY0, PHY1, PHY2 and PHY3, corresponding to their placement from left-to-right and as shown in Fig. 3.1.

3.3.3 Low-speed expansion connectors

The 96B Quad Ethernet Mezzanine has two low-speed expansion connectors: a pin header on the bottom-side, and a pin socket on the top-side. The pin header interfaces with the development platform, while the pin socket is used for "stacking" a second mezzanine card on top of the 96B Quad Ethernet Mezzanine. The figure below illustrates the connections to the low-speed expansion pin header (bottom side) and socket (top side).

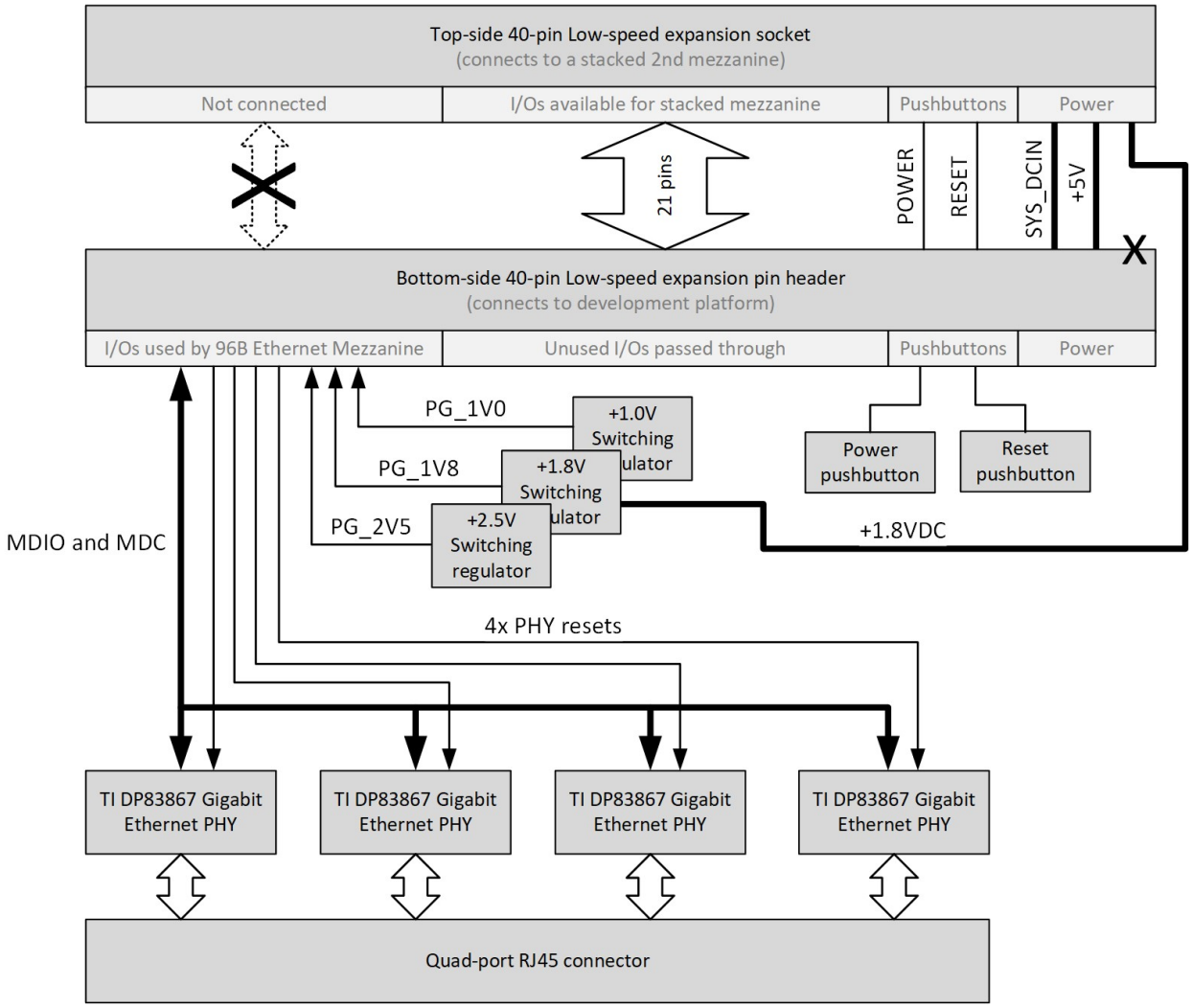

Fig. 3.3: Low-speed expansion and stacking connectors

Bottom-side low-speed expansion pin header

The 96B Quad Ethernet Mezzanine has a 40-pin low-speed expansion pin header, located on the bottom side of the board (see Fig. 3.2). This pin header connects directly to the development platform and it provides the main power supply to the 96B Quad Ethernet Mezzanine card as well as various I/O signals used by the mezzanine:

- SYS_DCIN (the main power supply)

- Power and reset pushbuttons
- MDIO and MDC signals for the Ethernet PHYs
- Reset signals for the Ethernet PHYs
- Power good signals from the switching regulators

Top-side low-speed expansion socket

Not all of the I/Os on the low-speed expansion header are used by the 96B Quad Ethernet Mezzanine card. To make these unused I/Os accessible to another mezzanine card, the 96B Quad Ethernet Mezzanine was designed with a 40-pin low-speed expansion socket, located on the top-side of the board (see Fig. 3.1). This expansion socket connects to all of the unused I/Os of the 96B Quad Ethernet Mezzanine, as well as the power/reset pushbuttons and the power supplies. It is designed to allow a standard 96Boards mezzanine card to be "stacked" on top of the 96B Quad Ethernet Mezzanine.

There are 21 unused I/Os that are passed through from the low-speed header to the low-speed socket and can be used by a "stacked" (2nd) mezzanine:

- UART0_CTS,TXD,RXD,RTS
- UART1_TXD,RXD
- I2C0_SCL,SDA
- I2C1_SCL,SDA
- SPI0_SCL,DIN,CS,DOUT
- PCM_FS,CLK,DO,DI
- GPIO-A,B,C

All power supply pins SYS_DCIN, +5V and +1V8 are connected to the appropriate supplies however only SYS_DCIN and +5V are directly passed through from the bottom-side low-speed expansion header. The +1V8 power supply is instead connected to the +1.8V that is generated by the 96B Quad Ethernet Mezzanine's on-board switching regulator. This allows the "stacked" mezzanine card to draw more than the standard's 100mA limit from the +1.8V supply.

The POWER and RESET pushbuttons are directly passed through.

3.3.4 High-speed expansion connector

The 96B Quad Ethernet Mezzanine has a 60-pin high-speed expansion header for interfacing with high-speed I/Os on the development platform. The mezzanine uses most of these I/Os for interfacing the SGMII links and the GPIO0/1 outputs of the DP83867 PHYs.

Each SGMII link is composed of two differential pairs, one for the transmit signal and one for the receive signal. These links typically operate at 1.25Gbps in each direction. These differential pairs are routed on the 96B Quad Ethernet Mezzanine with a controlled differential impedance of 100 ohms.

The DP83867 PHYs have two outputs, named GPIO0 and GPIO1, that can be used for Start of Frame detection among other things (see datasheet for details). Both of these outputs are routed through the high-speed expansion header so that they can be used by the development platform.

3.3.5 Power Supplies

The 96B Quad Ethernet Mezzanine has three switching regulators that generate the supply voltages required by the TI DP83867 Gigabit Ethernet PHYs (+1.0V, +1.8V and +2.5V). The switching regulators are all fed by the SYS_DCIN

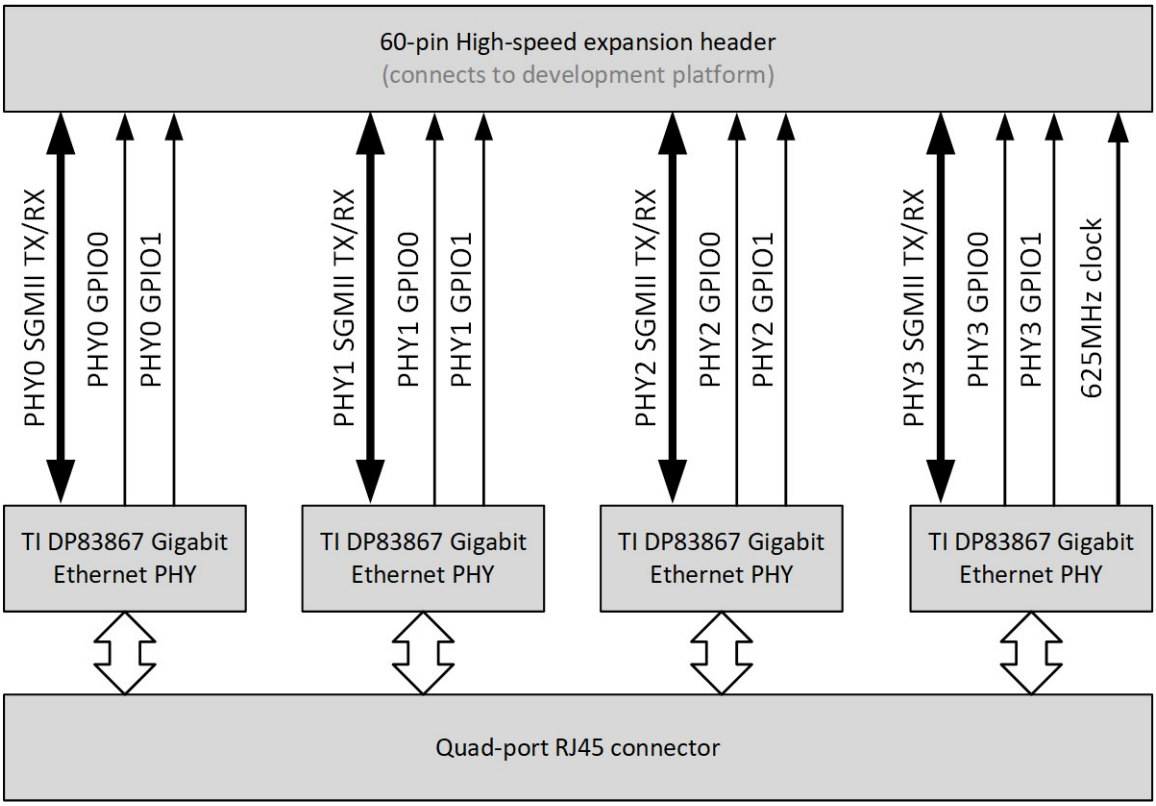

Fig. 3.4: High-speed expansion connector

main supply voltage that is provided by the development platform through the low-speed expansion connector. The 96B Quad Ethernet Mezzanine can accept a SYS_DCIN input supply voltage of +8VDC to +17VDC, although it is recommended that a +12VDC supply be used.

Power Sequencing

The SYS_DCIN voltage is always present as long as the power supply is connected to the development platform, and even when the development platform is turned OFF. To prevent the switching regulators from running when the development platform is turned OFF, the 96B Quad Ethernet Mezzanine uses the ENABLE inputs of the switching regulators. The signal used to ENABLE the switching regulators is the +1V8 supply pin of the low-speed expansion connector.

The power sequencing of the switching regulators was designed to meet the requirements of the DP83867 PHY and is as follows:

1. The power supply is connected to the development platform and the SYS_DCIN voltage rises to +12VDC (+12VDC expected, +8-17VDC accepted).
2. The development platform is turned ON, and the +1V8 pin of the low-speed expansion connector rises to +1.8V.
3. The +1.0V and +1.8V switching regulators are enabled by the +1V8 pin, and their respective POWER GOOD signals are asserted.
4. The +2.5V regulator is enabled by the POWER GOOD signal of the +1.8V switching regulator and it's POWER GOOD signal is asserted.

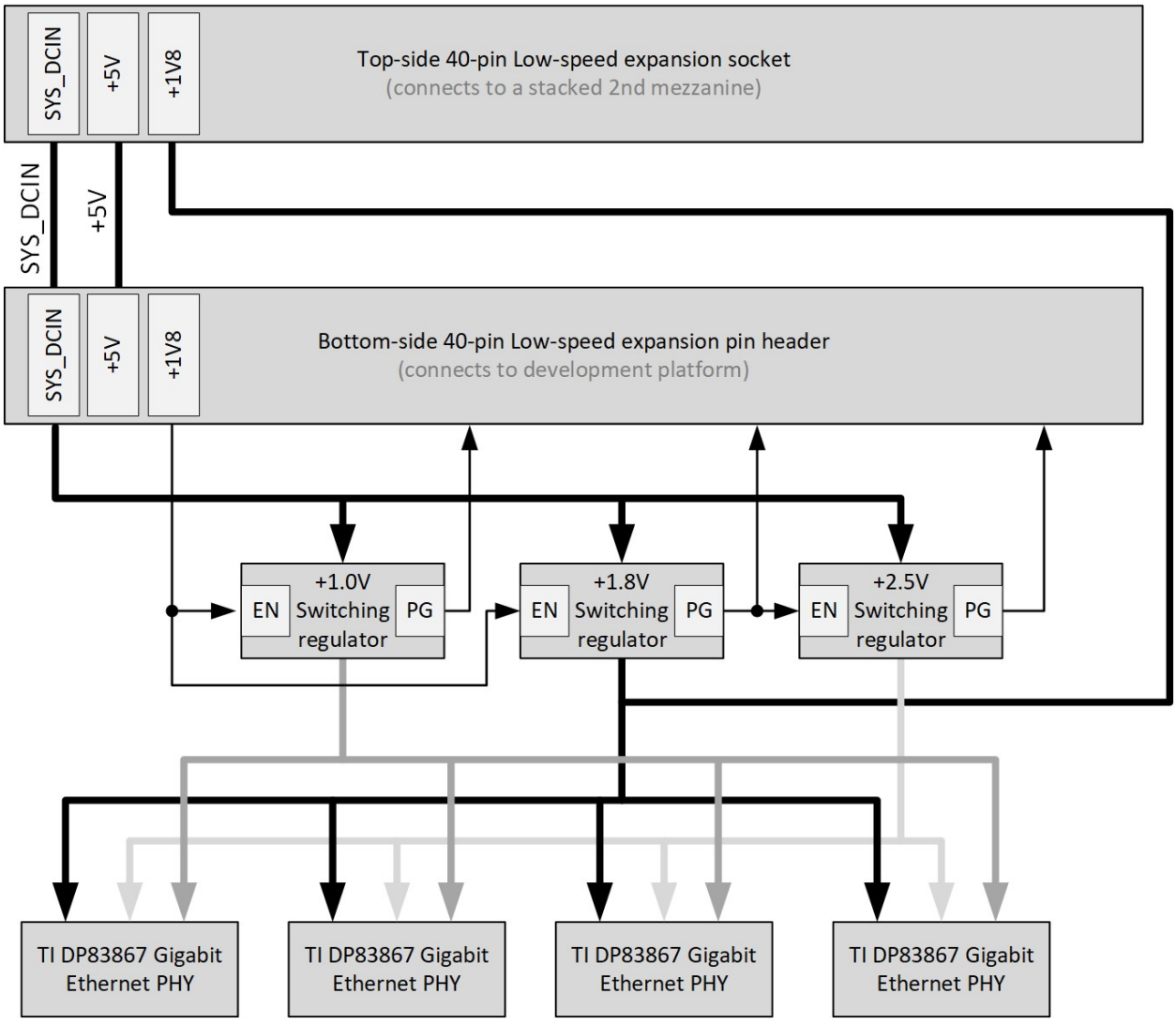

Fig. 3.5: Power supplies

Power good signals

To enable diagnostic checking of the 96B Quad Ethernet Mezzanine power supplies, each of the POWER GOOD signals are connected to the low-speed expansion connector. They are connected to the following pins:

- Power good +1.0V: Pin 26, GPIO-D
- Power good +1.8V: Pin 28, GPIO-F
- Power good +2.5V: Pin 27, GPIO-E

Stacking socket +1V8

The +1V8 supply pin of the low-speed stacking socket (intended for "stacking" a 2nd mezzanine card on top of the 96B Quad Ethernet Mezzanine) is connected to the +1.8V supply that is generated by the on-board switching regulator. The mezzanine was designed this way to allow the "stacked" mezzanine card to draw more than 100mA from the +1.8V supply, the maximum current that many 96Boards development platforms are designed to support.

Note that the SYS_DCIN and +5V supply pins of the low-speed stacking socket are connected directly to the associated pins on the low-speed expansion header on the bottom-side of the board.

3.3.6 Clocks

The figure below illustrates the clock connections on the 96B Quad Ethernet Mezzanine.

Each of the 4x DP83867 PHYs requires an input clock of 25MHz that can either be provided by a crystal, by a clock generator or by the CLK_OUT pin of another DP83867 device. To provide the 25MHz clock to all devices, the 96B Quad Ethernet Mezzanine connects a crystal to PHY3, and the CLK_OUT output of that PHY is used to drive the clock inputs of the 3 other PHYs. For hardware verification, the CLK_OUT output of each PHY is connected to a testpoint that can be probed on the top-side of the mezzanine card. Note that the CLK_OUT output pin of the DP83867 can be configured to output other signals/frequencies, however for the correct operation of the 96B Quad Ethernet Mezzanine, the default configuration of a 25MHz output should not be changed.

The DP83867 PHYs each have the ability to generate a 625MHz output clock that can be used by the SGMII receiver. The 96B Quad Ethernet Mezzanine routes only one of these clock outputs to the high-speed expansion connector, the one generated by PHY3. Note that this clock output is not enabled by default and must be enabled via the MDIO bus if required by the development platform.

3.3.7 Resets

The DP83867 Ethernet PHYs each have a hardware reset pin (RESET_N) that is routed separately to the low-speed expansion connector (see Fig. 3.3 for details). The reset pin must be driven by the development platform with an active-low signal. There are no pull-up resistors connected to the reset signals on the 96B Quad Ethernet Mezzanine card, however the DP83867 devices have pull-up resistors internal to the device. We recommend always driving the reset pins from the development platform in order to ensure reliable reset behavior.

3.3.8 PHY Configuration

Configuration of the PHY by software is performed using the MDIO bus. The MDIO bus consists of two signals: a bidirectional data signal (MDIO) and a clock signal (MDC). The data signal (MDIO) is driven by the master and slaves as an open drain output, and it is connected to a pull-up resistor located on the mezzanine card. The clock signal (MDC) is driven by the master only (the SoC on the development platform) and it does not require a pull-up resistor. For more information on the MDIO serial bus standard, please refer to the Wikipedia page on MDIO.

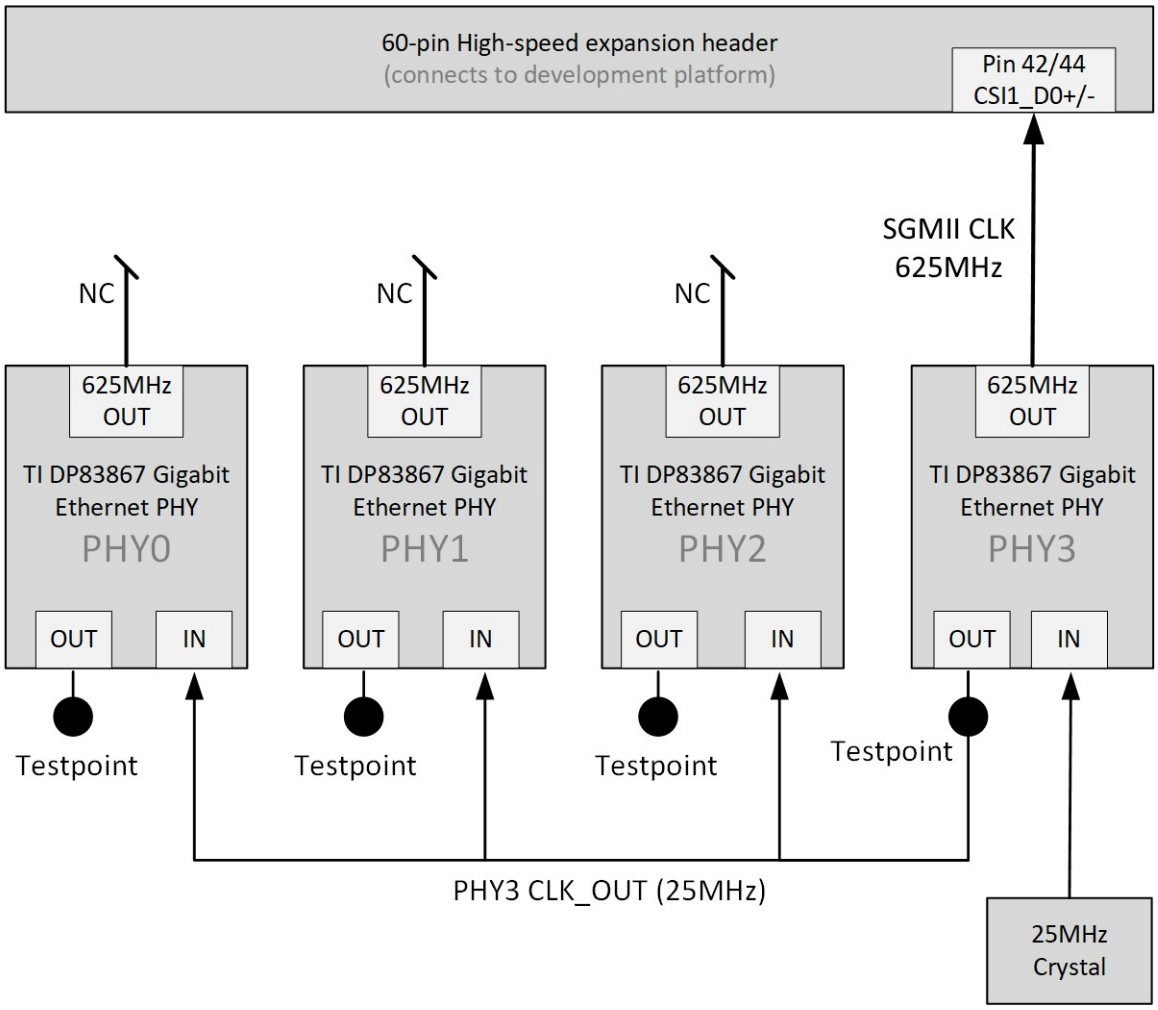

Fig. 3.6: Clocks

All of the 4 Ethernet PHYs are connected in a chain configuration to a single MDIO bus. Each PHY has it's own unique "PHY address" which is used when targetting the PHY on the MDIO bus. The diagram below illustrates the MDIO bus architecture and it's connection between the low-speed expansion connector and the Ethernet PHYs.

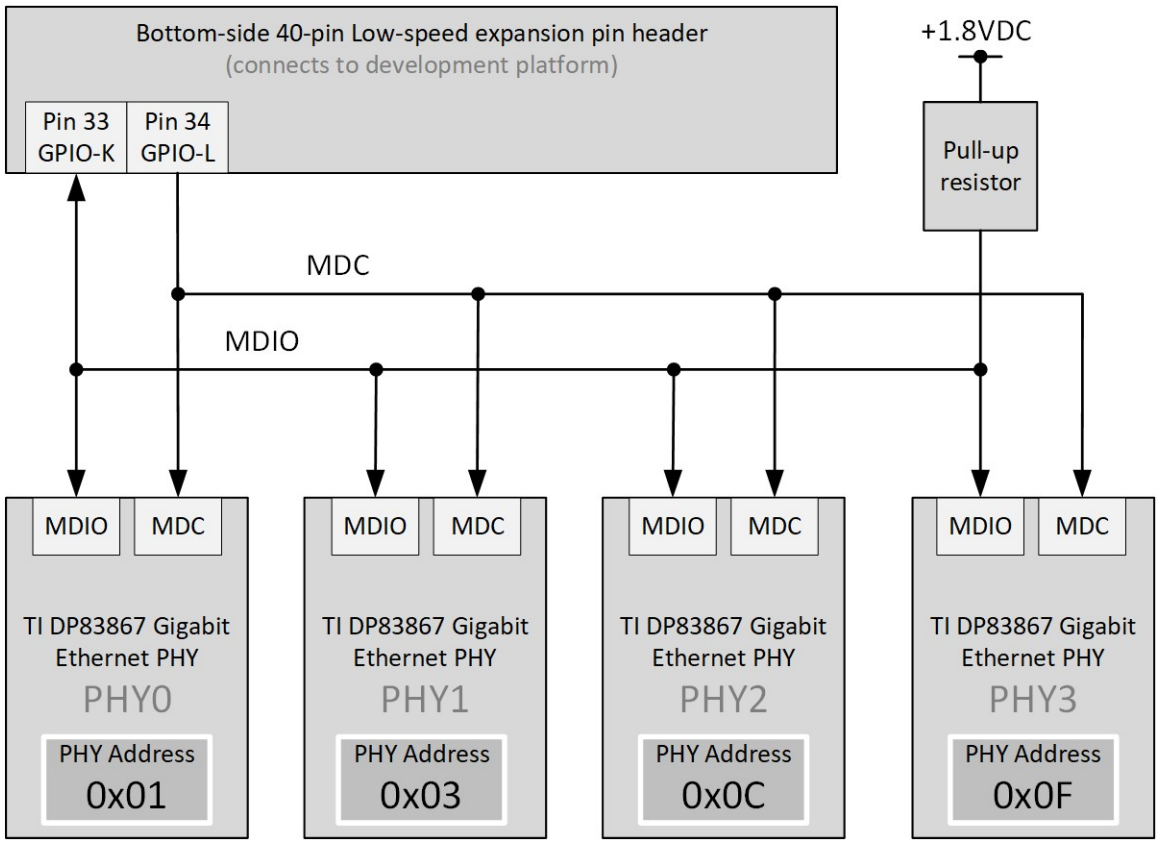

Fig. 3.7: MDIO bus architecture

As illustrated in the diagram, each PHY has a unique address that must be used when communicating with the PHYs over the MDIO bus. The PHY addresses are as follows:

- PHY0 (Port 0): PHY address 0x01
- PHY1 (Port 1): PHY address 0x03
- PHY2 (Port 2): PHY address 0x0C
- PHY3 (Port 3): PHY address 0x0F

3.4 Mechanical Information

3.4.1 Dimensions

The mechanical dimensions of the 96B Quad Ethernet Mezzanine card are illustrated in the figure below. All dimensions are in millimeters (mm).

The assembly drawing above is also available as a PDF at the link below:

96B Quad Ethernet Mezzanine Rev-A Assembly Drawing

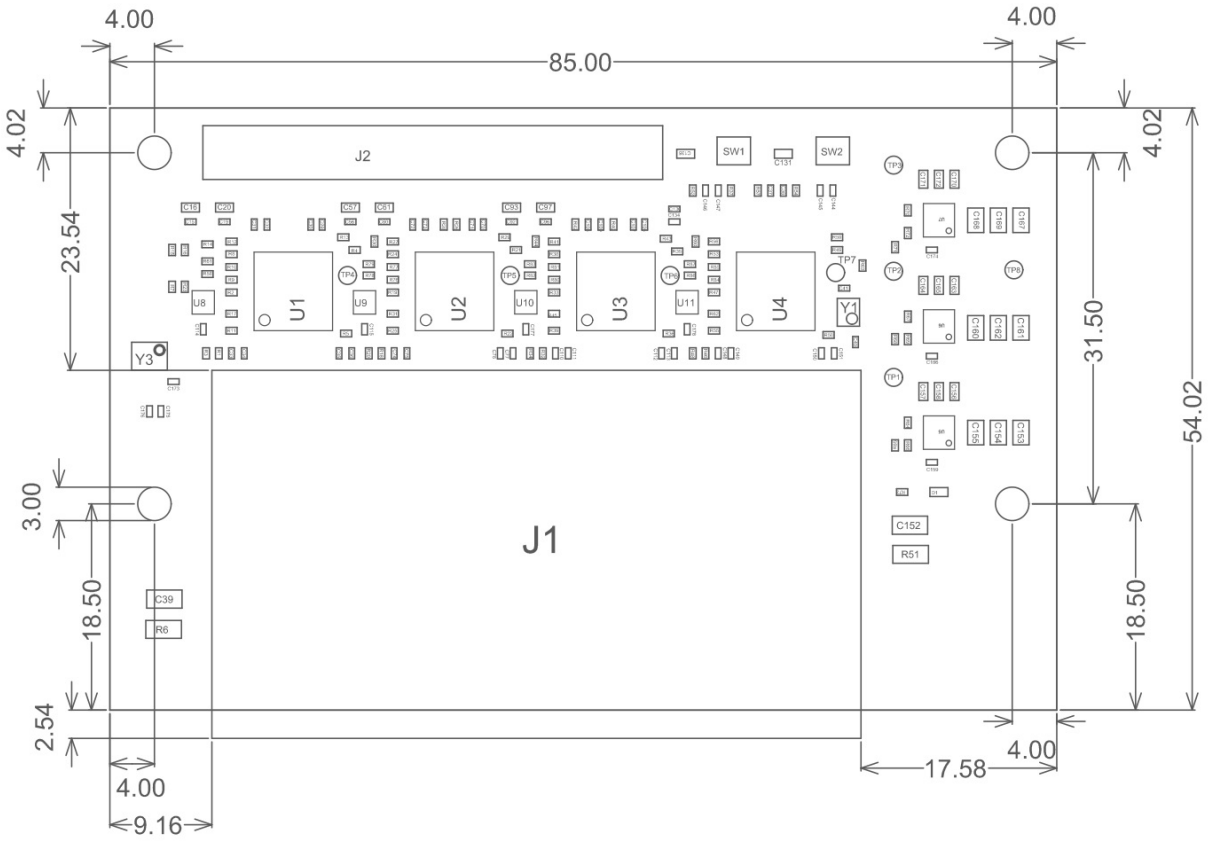

Fig. 3.8: 96B Quad Ethernet Mezzanine mechanical drawing

3.4.2 3D Model

The 3D model of the board is available as a STEP file at the link below:

96B Quad Ethernet Mezzanine Rev-A 3D STEP model

3.5 Getting Started

3.5.1 Example Designs

The example designs for the 96B Quad Ethernet Mezzanine are hosted on Github and are designed for the Avnet Ultra96 development platform. There are currently two examples and they are differentiated by the type of Ethernet MACs used and their location in the system.

AXI Ethernet based example

This example design is based on Xilinx's soft (ie. FPGA implemented) MAC, the AXI Ethernet Subsystem IP, that can be found in the Vivado IP Catalog. As the MAC is implemented in the FPGA fabric, this example is ideal for applications that require some packet processing to be performed in the FPGA.

PS GEM based example

This example design utilizes the 4x Gigabit Ethernet MACs (GEMs) that are embedded into the Processing System (PS) of the Zynq Ultrascale+™ device of the Ultra96. The MACs in this example design do not use up any of the FPGA fabric, which makes it ideal for applications that need to use the FPGA for other purposes.

3.5.2 Requirements

In order to use the example designs, you will need the following:

- Windows or Linux PC
- Xilinx Vivado
- Xilinx Vitis
- 1x Ultra96 development platform
- 1x 96B Quad Ethernet Mezzanine

If you want to build PetaLinux for the example designs, you will also need:

- Linux PC or a virtual Linux machine
- PetaLinux SDK

You will also need a CAT-5e Ethernet cable and a link partner, such as a PC with an Ethernet port or a network router.

Additionally, you may need to install the Ultra96 board definition files to your Vivado installation, and obtain an AXI Ethernet evaluation license if you intend to use that design.

Install Ultra96 board definition files

To use the example projects, you must first install the board definition files for the Ultra96 into your Vivado and Vitis installation. The Ultra96 board definition files are hosted on Avnet's Github repo.

Clone or download that repo, then copy the `ultra96v1` and `ultra96v2` directories from it to the following directories on your machine: * `<path-to-xilinx-vivado>/data/boards/board_files` * `<path-to-xilinx-vitis>/data/boards/board_files`

AXI Ethernet evaluation license

If you intend to build the AXI Ethernet based design, you will need to get an evaluation (or full) license for the Tri-mode Ethernet MAC from Xilinx. You can find instructions for that here: Xilinx Soft TEMAC license

3.5.3 Build instructions

Download the source code

The source code for both example designs can be found on our Github page:

96B Quad Ethernet Mezzanine Github page

The repository contains the following directories:

- **Vivado**: Contains the scripts to build the Vivado projects
- **Vitis**: Contains a script to generate and build the standalone software applications
- **PetaLinux**: Contains a script and configuration files to build the PetaLinux projects
- **EmbeddedSw**: Contains modifications to the lwIP software library

Build the Vivado and Vitis projects

Once you have installed the board definition files, and you have installed the required licenses, then you can use the sources in this repository to build the Vivado, Vitis and PetaLinux projects. Start by cloning the repo or download it as a zip file and extract the files to your hard drive, then follow these steps depending on your OS:

Windows users

1. Open Windows Explorer, browse to the repo files on your hard drive.
2. In the Vivado directory, you will find multiple batch files (*.bat). Double click on the batch file of the example project that you would like to generate - this will generate a Vivado project.
3. You will be asked to select between Ultra96 v1 and v2. It is important to select the correct version of the Ultra96 that you are using. Type 1 or 2 (for v1 or v2) and press ENTER. The script will now generate the Vivado project for your board.
4. Run Vivado and open the project that was just created.
5. Click Generate bitstream.
6. When the bitstream is successfully generated, select "File->Export->Export Hardware". In the window that opens, tick "Include bitstream" and "Local to project".
7. Return to Windows Explorer and browse to the Vitis directory in the repo.

8. Double click the *build-vitis.bat* batch file. The batch file will run the *build-vitis.tcl* script and build the Vitis workspace containing the hardware design and the software application. Please refer to the "README.md" file in the Vitis subdirectory for instructions on running the software application on hardware.
9. If you are interested in building PetaLinux, you will need to use a Linux machine and follow the steps for Linux users below.

Linux users

1. Launch the Vivado GUI.
2. On the welcome page, there is a Tcl console. In the Tcl console, `cd` to the repo files on your hard drive and into the Vivado subdirectory. For example: `cd /media/projects/ethernet96/Vivado`.
3. Specify the version of Ultra96 you want to build the project for (v1 or v2) by using one of the following commands: `set argv {1}` for v1, or `set argv {2}` for v2.
4. In the Vivado subdirectory, you will find multiple Tcl files. To list them, type `exec ls {*}[glob *.tcl]`. Determine the Tcl script for the example project that you would like to generate (for example: `build-ps-gem.tcl`), then `source` the script in the Tcl console: For example: `source build-ps-gem.tcl`
5. Vivado will run the script and generate the project. When it's finished, click Generate bitstream.
6. When the bitstream is successfully generated, select "File->Export->Export Hardware". In the window that opens, tick "Include bitstream" and "Local to project".
7. To build the Vitis workspace, open a Linux command terminal and `cd` to the Vitis directory in the repo.
8. The Vitis directory contains the `build-vitis.tcl` script that will build the Vitis workspace containing the hardware design and the software application. Run the build script by typing the following command: `<path-of-xilinx-vitis>/bin/xsct build-vitis.tcl` Note that you must replace `<path-of-xilinx-vitis>` with the actual path to your Vitis installation.
9. Please refer to the "README.md" file in the Vitis subdirectory for instructions on running the software application on hardware.
10. To build the PetaLinux project, follow the steps in the following section.

Build the PetaLinux projects

Once the Vivado project(s) have been built and exported, you can now build the PetaLinux project(s).

Note: The PetaLinux projects can only be built on a Linux machine (or virtual Linux machine).

Linux users

1. To build the PetaLinux project, first launch PetaLinux by sourcing the "settings.sh" bash script, eg: `source <path-to-installed-petalinux>/settings.sh`.
2. Now `cd` to the PetaLinux directory in the repo and run the `build-petalinux` script. You may have to add execute permission to the script first using `chmod +x build-petalinux`, then run it by typing `./build-petalinux`.

Warning: UNIX line endings: The scripts and files in the PetaLinux directory of this repository must have UNIX line endings when they are executed or used under Linux. The best way to ensure UNIX line endings, is to clone the repo directly onto your Linux machine. If instead you have copied the repo from a Windows machine, the files will have DOS line endings and you must use the `dos2unix` tool to convert the line endings for UNIX.

3.5.4 Launch on hardware

Echo server via JTAG

1. Open Vitis.
2. Power up your hardware platform and ensure that the JTAG is connected properly.
3. Select "Xilinx Tools->Program FPGA". In the "Program FPGA" dialog box that appears, select the "Hardware Platform" that you want to run, this will correspond to name of the Vivado project that you built earlier.
4. Click on the software application that you want to run, it should be the one with the postfix "_echo_system".
5. Open the drop-down menu of the "Run" button (play) on the toolbar. Select "Run Configurations", then in the dialog box that appears, double-click on the option "Single Application Debug". This will create a new run configuration for the application.
6. Select the new run configuration and click "Run".

PetaLinux via JTAG

To launch the PetaLinux project on hardware via JTAG, connect and power up your hardware and then use the following commands in a Linux command terminal:

1. Change current directory to the PetaLinux project directory: `cd <petalinux-project-dir>`
2. Download bitstream to the FPGA: `petalinux-boot --jtag --fpga` Note that you don't have to specify the bitstream because this command will use the one that it finds in the `./images/linux` directory.
3. Download the PetaLinux kernel to the FPGA: `petalinux-boot --jtag --kernel`

PetaLinux via SD card

To boot PetaLinux on hardware via SD card:

1. The SD card must first be prepared with two partitions: one for the boot files and another for the root file system.
    - Plug the SD card into your computer and find it's device name using the `dmesg` command. The SD card should be found at the end of the log, and it's device name should be something like `/dev/sdX`, where `X` is a letter such as a,b,c,d, etc. Note that you should replace the `X` in the following instructions.
    - Run `fdisk` by typing the command `sudo fdisk /dev/sdX`
    - Make the `boot` partition: typing `n` to create a new partition, then type `p` to make it primary, then use the default partition number and first sector. For the last sector, type `+1G` to allocate 1GB to this partition.
    - Make the `boot` partition bootable by typing `a`
    - Make the `root` partition: typing `n` to create a new partition, then type `p` to make it primary, then use the default partition number, first sector and last sector.
    - Save the partition table by typing `w`

- Format the `boot` partition (FAT32) by typing `sudo mkfs.vfat -F 32 -n boot /dev/sdX1`
- Format the `root` partition (ext4) by typing `sudo mkfs.ext4 -L root /dev/sdX2`

2. Copy the following files to the *boot* partition of the SD card: Assuming the `boot` partition was mounted to `/media/user/boot`, follow these instructions:

```
$ cd /media/user/boot/
$ sudo cp /<petalinux-project>/images/linux/BOOT.bin .
$ sudo cp /<petalinux-project>/images/linux/image.ub .
```

3. Create the root file system by extracting the `rootfs.tar.gz` file to the `root` partition. Assuming the `root` partition was mounted to `/media/user/root`, follow these instructions:

```
$ cd /media/user/root/
$ sudo cp /<petalinux-project>/images/linux/rootfs.tar.gz .
$ sudo tar xvf rootfs.tar.gz -C .
$ sync
```

   Once the `sync` command returns, you will be able to eject the SD card from the machine.

4. Connect and power your hardware.

3.5.5 Echo Server Example Usage

Default IP address

The echo server is designed to attempt to obtain an IP address from a DHCP server. This is useful if the echo server is connected to a network. Once the IP address is obtained, it is printed out in the UART console output.

If instead the echo server is connected directly to a PC, the DHCP attempt will fail and the echo server will default to the IP address 192.168.1.10. To be able to communicate with the echo server from the PC, the PC should be configured with a fixed IP address on the same subnet, for example: 192.168.1.20.

Ping the port

The echo server can be "pinged" from a connected PC, or if connected to a network, from another device on the network. The UART console output will tell you what the IP address of the echo server is. To ping the echo server, use the `ping` command from a command console.

For example: `ping 192.168.1.10`

Change the targetted port

The echo server example design currently can only target one Ethernet port at a time. Selection of the Ethernet port can be changed by modifying the defines contained in the `platform_config.h` file in the application sources. Set `PLATFORM_EMAC_BASEADDR` to one of the following values:

For designs using the GEMs:

- Port 0: `XPAR_XEMACPS_0_BASEADDR`
- Port 1: `XPAR_XEMACPS_1_BASEADDR`
- Port 2: `XPAR_XEMACPS_2_BASEADDR`
- Port 3: `XPAR_XEMACPS_3_BASEADDR`

For designs using AXI Ethernet:

- Port 0: `XPAR_AXIETHERNET_0_BASEADDR`
- Port 1: `XPAR_AXIETHERNET_1_BASEADDR`
- Port 2: `XPAR_AXIETHERNET_2_BASEADDR`
- Port 3: `XPAR_AXIETHERNET_2_BASEADDR`

3.5.6 PetaLinux Example Usage

In the PetaLinux projects, the Ethernet ports are assigned to the network interfaces *eth0-eth3* as follows:

- **eth0**: Port 0
- **eth1**: Port 1
- **eth2**: Port 2
- **eth3**: Port 3

The following examples demonstrate how to use these network interfaces to configure the Ethernet ports for use in PetaLinux.

Enable port

In this example we enable port 0 (eth0).

```
root@ps_gem:~# ifconfig eth0 up
[  209.778955] TI DP83867 ff0b0000.mdio-mii:03: attached PHY driver [TI DP83867] (mii_
↪bus:phy_addr=ff0b0000.mdio-mii:03, irq=POLL)
[  209.793249] pps pps1: new PPS source ptp1
[  209.797193] macb ff0b0000.ethernet: gem-ptp-timer ptp clock registered.
[  209.803995] IPv6: ADDRCONF(NETDEV_UP): eth0: link is not ready
[  213.868935] macb ff0b0000.ethernet eth0: link up (1000/Full)
[  213.874547] IPv6: ADDRCONF(NETDEV_CHANGE): eth0: link becomes ready
```

Enable port with fixed IP address

In this example we enable port 1 (eth1) with a fixed IP address.

```
root@ps_gem:~# ifconfig eth1 192.168.2.19 up
[  209.778955] TI DP83867 ff0b0000.mdio-mii:03: attached PHY driver [TI DP83867] (mii_
↪bus:phy_addr=ff0b0000.mdio-mii:03, irq=POLL)
[  209.793249] pps pps1: new PPS source ptp1
[  209.797193] macb ff0c0000.ethernet: gem-ptp-timer ptp clock registered.
[  209.803995] IPv6: ADDRCONF(NETDEV_UP): eth1: link is not ready
[  213.868935] macb ff0c0000.ethernet eth1: link up (1000/Full)
[  213.874547] IPv6: ADDRCONF(NETDEV_CHANGE): eth1: link becomes ready
```

Check status of a port with ethtool

In this example we check the status of port 2 (eth2) with “ethtool”.

```
root@ps_gem:~# ethtool eth2
Settings for eth2:
        Supported ports: [ TP MII ]
        Supported link modes:   10baseT/Half 10baseT/Full
                                100baseT/Half 100baseT/Full
                                1000baseT/Half 1000baseT/Full
        Supported pause frame use: No
        Supports auto-negotiation: Yes
        Advertised link modes:  10baseT/Half 10baseT/Full
                                100baseT/Half 100baseT/Full
                                1000baseT/Half 1000baseT/Full
        Advertised pause frame use: No
        Advertised auto-negotiation: Yes
        Link partner advertised link modes:  10baseT/Half 10baseT/Full
                                             100baseT/Half 100baseT/Full
                                             1000baseT/Full
        Link partner advertised pause frame use: No
        Link partner advertised auto-negotiation: Yes
        Speed: 1000Mb/s
        Duplex: Full
        Port: MII
        PHYAD: 1
        Transceiver: internal
        Auto-negotiation: on
        Link detected: yes
```

Ping link partner using specific port

In this example we ping the link partner from port 1 (eth1).

```
root@ps_gem:~# ping -I eth1 192.168.1.10
PING 192.168.1.10 (192.168.1.10): 56 data bytes
64 bytes from 192.168.1.10: seq=0 ttl=128 time=0.939 ms
64 bytes from 192.168.1.10: seq=1 ttl=128 time=0.496 ms
64 bytes from 192.168.1.10: seq=2 ttl=128 time=0.486 ms
64 bytes from 192.168.1.10: seq=3 ttl=128 time=0.485 ms
64 bytes from 192.168.1.10: seq=4 ttl=128 time=0.501 ms
^C
--- 192.168.1.10 ping statistics ---
5 packets transmitted, 5 packets received, 0% packet loss
round-trip min/avg/max = 0.485/0.581/0.939 ms
```

Check port configuration

In this example we check the configuration of port 1 (eth1).

```
root@ps_gem:~# ifconfig eth1
eth1      Link encap:Ethernet  HWaddr 00:0A:35:00:01:23
          inet addr:192.168.1.11  Bcast:192.168.1.255  Mask:255.255.255.0
          inet6 addr: fe80::20a:35ff:fe00:123%4294741717/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:148 errors:0 dropped:0 overruns:0 frame:0
          TX packets:74 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
```

(continues on next page)

(continued from previous page)

```
          RX bytes:17567 (17.1 KiB)  TX bytes:12943 (12.6 KiB)
          Interrupt:31
```

3.6 Board Setup

3.6.1 Mezzanine fastening hardware

For typical development use, in a lab or on a desk, the mating force of the expansion connectors alone is enough to securly fix the mezzanine card to the carrier board. However, for applications requiring higher mechanical robustness, the mezzanine can be fixed to the carrier board using 7mm standoffs and M2.5 machine screws. We suggest the following part numbers, however equivalent parts can also be used:

- Hex standoff, Thread M2.5 x 0.45, Aluminium, Board-to-board length 7mm
  Part number: M2102-2545-AL
  Manufacturer: RAF Electronic Hardware
- Machine screw, Thread M2.5 x 0.45, Length (below head) 4mm, Stainless steel, Phillips head
  Part number: 425-035
  Supplier: Spaenaur

3.6.2 Stacking a second mezzanine

A second mezzanine card can be stacked on top of the 96B Quad Ethernet Mezzanine as shown in the image below.

The RJ45 connector (0826-1X4T-23-F) has a height of 13.59mm, while the expansion socket has a height of 4.5mm as defined by the 96Boards spec. For this reason, an extender (see image below) is required for stacking most mezzanine cards onto the 96B Quad Ethernet Mezzanine. The extender is a 40-pin pin socket with 8mm long pins that is inserted into the 96B Quad Ethernet Mezzanine's low-speed expansion socket, effectively increasing it's height above that of the RJ45 connector. The stacked mezzanine card is then plugged into the extender socket and sits comfortably above the RJ45 connector. We recommend that the following connector be used as the extender socket, however equivalent parts can also be used:

- 40-pin Pin socket with 8mm long pins
  Part number: F263-1220A0BSYE1
  Manufacturer: Yxcon

When using the extender socket recommended above, the stacked mezzanine sits at a height of 16mm above the 96B Quad Ethernet Mezzanine, and it can be fixed to the mezzanine by using 16mm standoffs and M2.5 machine screws. We suggest the following part numbers, however equivalent parts can also be used:

- Hex standoff, Thread M2.5 x 0.45, Aluminium, Board-to-board length 16mm
  Part number: M2111-2545-AL
  Manufacturer: RAF Electronic Hardware
- Machine screw, Thread M2.5 x 0.45, Length (below head) 4mm, Stainless steel, Phillips head
  Part number: 425-035
  Supplier: Spaenaur

The image below illustrates the use of the extender socket and 16mm standoffs.

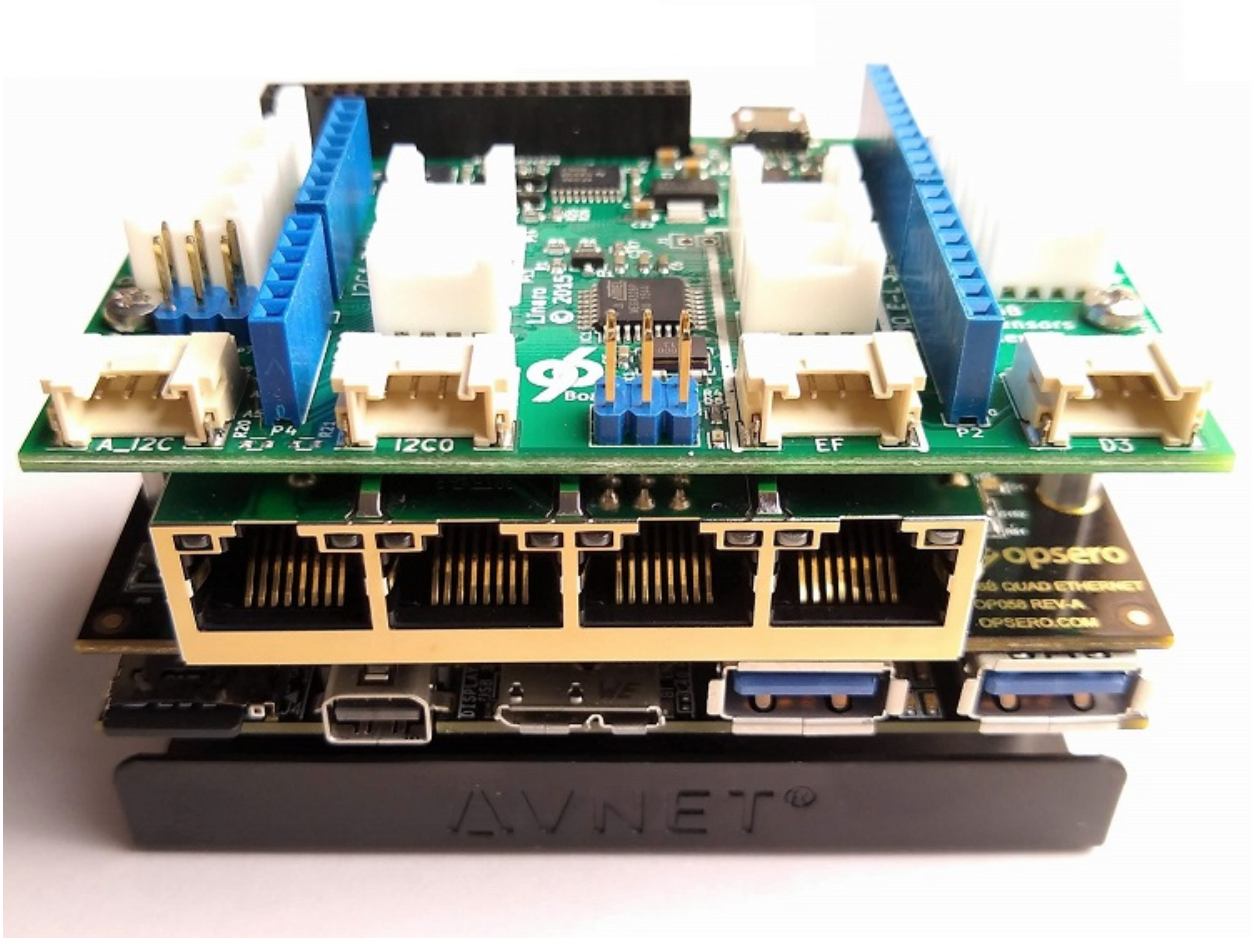

Fig. 3.9: 96B Quad Ethernet Mezzanine with stacked Sensors mezzanine (front)

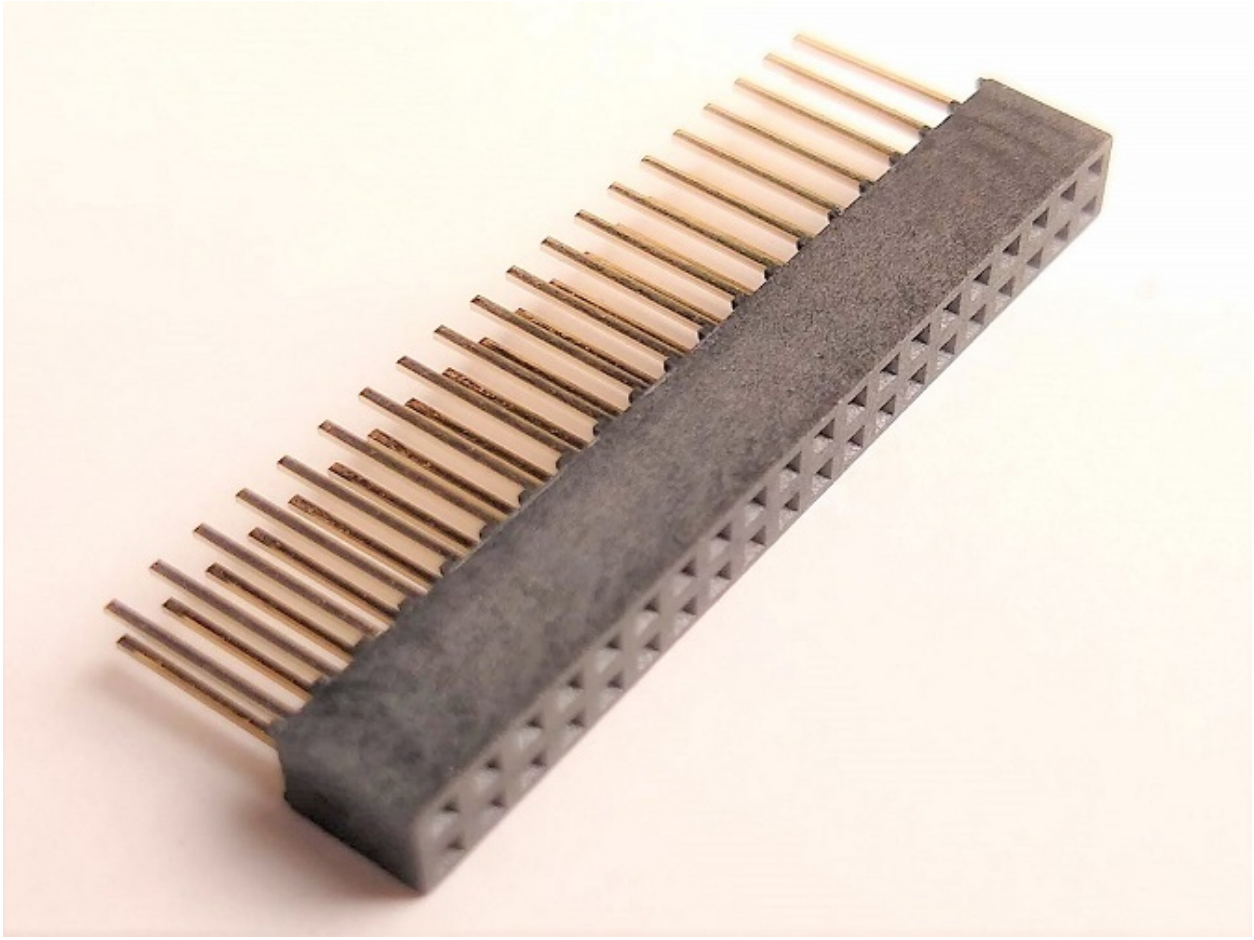

Fig. 3.10: Extender for stacking second mezzanine card

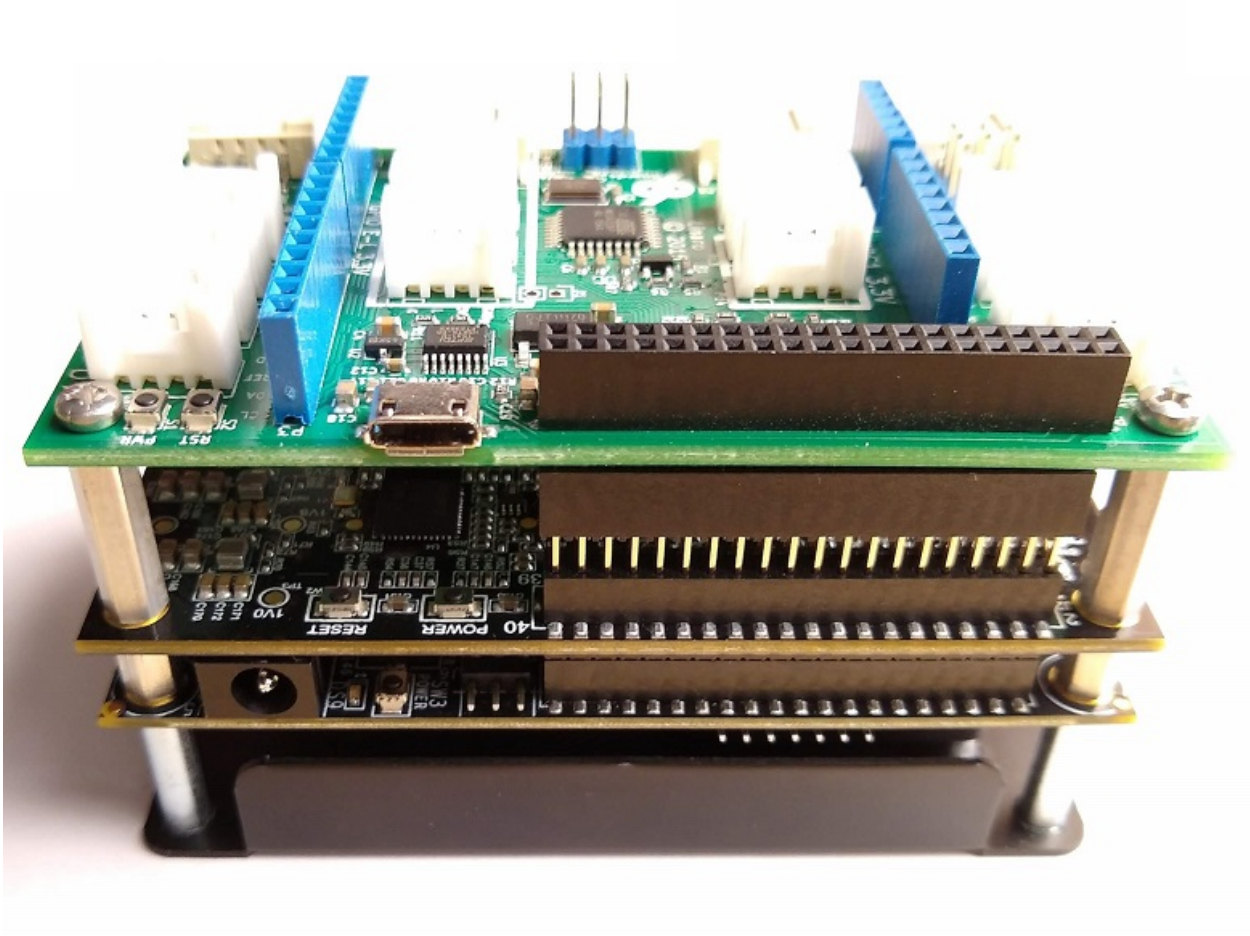

Fig. 3.11: 96B Quad Ethernet Mezzanine with stacked Sensors mezzanine (back)

3.7 Programming Guide

This programming guide is specific to users of the Avnet Ultra96 board. The purpose of the guide is to provide the user with the details of the programming requirements to enable them to operate the hardware and customise functionality.

3.7.1 Vivado design

We recommend that all users start with our example Vivado designs when using the mezzanine card with the Ultra96. For those who wish to better understand the example designs, or develop their own Vivado designs, this section provides more detail on the critical elements.

SGMII Implementation

The 96B Quad Ethernet Mezzanine card was designed to conform with the 96Boards specification for mezzanine cards, however the pinout of the high-speed expansion connector was chosen to maximize its usability when paired with the Ultra96. This section provides an explanation of the pin selection of the SGMII lanes and how to implement the SGMII interfaces in the Vivado design.

The Ultra96 high-speed expansion connector does not provide access to any gigabit transceivers of the Zynq Ultrascale+ device. For this reason, the SGMII interfaces must be implemented using *SGMII over LVDS*. To implement SGMII over LVDS on the Zynq Ultrascale+, the appropriate IP core is the PCS/PMA or SGMII IP. This IP core has several requirements on the I/O pins with which it can be used. Two of the critical requirements are:

1. The I/O pair used for TX, and that used for RX must be in the same BYTE_GROUP
2. The I/O pair used for TX must be in the opposite nibble to that used for RX

For example, the TX and RX pairs of a single SGMII interface could be located in BYTE_GROUP 1, with the TX pair in the lower nibble and the RX pair in the upper nibble.

Given these requirements, the possible pin selections can be determined by looking at the I/O pins that are available to us, and their respective BYTE_GROUPs and nibbles. The high-speed expansion connector of the Ultra96 makes these I/O pins from bank 65 available for use:

BYTE_GROUP	Nibble	Available bits (pairs)
0	Lower	0,1,2
	Upper	Not connected
1	Lower	0,1,2
	Upper	0,1
2	Lower	Not connected
	Upper	0
3	Lower	0,1,2
	Upper	0

Considering the requirements of the SGMII IP and the choice of I/O pins on the high-speed expansion connector of the Ultra96, it is only possible to connect 3 SGMII interfaces:

- Interface 0: BYTE_GROUP 1, lower pair 0, upper pair 0
- Interface 1: BYTE_GROUP 1, lower pair 1, upper pair 1
- Interface 2: BYTE_GROUP 3, lower pair 0, upper pair 0

The 96B Quad Ethernet Mezzanine card has 4 SGMII interfaces. To connect the 4th interface to the Ultra96, we in fact use two SGMII interfaces, where only one direction (TX/RX) of each interface is actually used. This allows us to

satisfy the requirements of the SGMII IP with the remaining pins that are available to us. Here is how the 4th interface is connected:

- Interface 3 RX: BYTE_GROUP 0, lower pair 0 (RX), upper pair 0 (N/C pins)
- Interface 3 TX: BYTE_GROUP 2, upper pair 0 (TX), lower pair 0 (N/C pins)

With this pin selection, the 4th interface requires two SGMII IPs to function - one that handles the RX interface and another that handles the TX interface. The unused TX and RX pins of the SGMII IP cores are assigned to pins that are not externally connected. To connect the GMII interface between the MAC IP and the two SGMII IP cores, we open the interfaces and connect only the RX GMII pins to the RX SGMII core, and the TX GMII pins to the TX SGMII core.

A single SGMII IP core implements all of the SGMII interfaces connected to a single BYTE_GROUP. As interfaces 0 and 1 are connected to BYTE_GROUP 1, they are implemented by a single SGMII IP core. Interface 2 has its own SGMII IP core, as does interface 3's RX lane and interface 3's TX lane.

Note: It is possible to implement SGMII over LVDS using the AXI Ethernet Subsystem IP. However, at the time of this writing, the AXI Ethernet Subsystem IP can only implement a single SGMII over LVDS interface. For this reason, the IP cannot be used to implement both interface 0 and interface 1. Instead, to use both interfaces 0 and 1 with AXI Ethernet Subsystem IP, the SGMII over LVDS interface must be implemented with the PCS/PMA or SGMII IP core, and then connected to the AXI Ethernet Subsystem IP cores through internal GMII interfaces. See the AXI Ethernet example design for the required connections.

MDIO bus

The 96B Quad Ethernet Mezzanine card uses a single MDIO bus to connect the development platform to the Ethernet PHYs. The Vivado design should only contain one MDIO interface that connects to the external MDIO bus. Communication with all Ethernet PHYs must be performed through this single bus.

The mezzanine card Ethernet PHYs for ports 0,1,2 and 3 have addresses 0x1, 0x2, 0xC and 0xF respectively. If your Vivado design uses IP cores that themselves have PHY addresses (such as the PCS/PMA or SGMII IP), and connect to the same MDIO bus as that of the external PHYs, ensure that these IP cores have unique addresses with respect to the external PHYs.

If using the PCS/PMA or SGMII IP core, the MDIO interfaces of these cores can be chained together such that the output of one connects to the input of another. Be sure to also connect the tri-state signal (MDIO_T) from one core to the next, this is essential for correct operation of the MDIO bus.

EMIO GPIOs

In both the PS GEM and AXI Ethernet designs, the EMIO GPIOs are connected to the PHY resets and the PHY GPIOs as shown in the table below:

EMIO GPIO	PHY Connection	GPIO bank	GPIO bit	Pin mapping
0	PHY0 RESET_N	3	0	416
1	PHY1 RESET_N	3	1	417
2	PHY2 RESET_N	3	2	418
3	PHY3 RESET_N	3	3	419
4	PHY0 GPIO_0	3	4	420
5	PHY0 GPIO_1	3	5	421
6	PHY1 GPIO_0	3	6	422
7	PHY1 GPIO_1	3	7	423
8	PHY2 GPIO_0	3	8	424
9	PHY2 GPIO_1	3	9	425
10	PHY3 GPIO_0	3	10	426
11	PHY3 GPIO_1	3	11	427

The first four EMIO GPIOs are connected to the external PHY RESET_N pins. These can be driven LOW to place the respective PHY in hardware reset. The remaining EMIO GPIOs are connected to the external PHY GPIO_x pins. Although named "GPIO_x", these PHY pins are in fact output-only and their purpose can be configured by setting the GPIO Mux Control Register of the PHYs via the MDIO bus. Please refer to the DP83867 datasheet for more information.

Constraints

For more information on the required constraints, please refer to the XDC files used by the example designs, located in the constraints directory of the Github repository.

3.7.2 PetaLinux

This section provides the information required to build PetaLinux or other Linux distributions for use with the 96B Quad Ethernet Mezzanine card.

Device tree for GEM design

The required additions to the device tree include:

- Define the `phy0` to `phy3` nodes within the `gem0` node
- Within each phy node:
  - Define the PHY address (`reg`)
  - Set the TX and RX internal delay
  - Set the FIFO depth
  - Enable SGMII clock for PHY3 (`ti,6-wire-mode`)
  - Disable SGMII auto-negotiation in PHY3 (`ti,dp83867-sgmii-autoneg-dis` see DP83867 patch below)
- Add these properties to each of the `gem0` to `gem3` nodes:

  - Set PHY handle (use labels defined in the `gem0` node)
  - Set PHY mode set to GMII
  - Set PHY reset to connected GPIO
  - Set PHY reset to active-low

For more detail, refer to the device tree for the GEM design in the Github repository.

Device tree for AXI Ethernet design

- Define the `phy0` to `phy3` nodes within the `mdio` node of the `axi_ethernet_0` node
- Within each phy node:
  - Define the PHY address (`reg`)
  - Specify PHY type to SGMII (`xlnx,phy-type = <0x4>;`)
  - Set the TX and RX internal delay
  - Set the FIFO depth
  - Enable SGMII clock for PHY3 (`ti,6-wire-mode`)
  - Disable SGMII auto-negotiation in PHY3 (`ti,dp83867-sgmii-autoneg-dis` see DP83867 patch below)
- Add these properties to each of the `axi_ethernet_0` to `axi_ethernet_3` nodes:
  - Set PHY handle (use labels defined in the `axi_ethernet_0` node)
  - Set PHY mode set to GMII

For more detail, refer to the device tree for the AXI Ethernet design in the Github repository.

Rootfs configuration

In the rootfs configuration, we add the following packages:

- ethtool
- ethtool-dev
- ethtool-dbg
- git
- openamp-fw-echo-testd
- openamp-fw-mat-muld
- openamp-fw-rpc-demo
- packagegroup-petalinux
- packagegroup-petalinux-matchbox
- packagegroup-petalinux-openamp
- packagegroup-petalinux-self-hosted
- packagegroup-petalinux-v4lutils
- packagegroup-petalinux-x11

- libftdi
- cmake
- iperf3
- lmsensors-sensorsdetect
- packagegroup-base-extended
- packagegroup-petalinux-96boards-sensors
- packagegroup-petalinux-ultra96-webapp
- python-pyserial
- python3-pip
- ultra96-ap-setup

In PetaLinux SDK, the rootfs is configured using this command: `petalinux-config -c rootfs`

Kernel configuration

The following options must be set in the Kernel configuration:

- CONFIG_XILINX_DMA_ENGINES=y
- CONFIG_XILINX_DPDMA=y
- CONFIG_XILINX_ZYNQMP_DMA=y
- CONFIG_ETHERNET=y
- CONFIG_NET_VENDOR_XILINX=y
- CONFIG_XILINX_AXI_EMAC=y
- CONFIG_XILINX_PHY=y
- CONFIG_NET_CADENCE=y
- CONFIG_MACB=y
- CONFIG_NETDEVICES=y
- CONFIG_HAS_DMA=y
- CONFIG_CPU_IDLE=n

In PetaLinux SDK, the kernel is configured using this command: `petalinux-config -c kernel`

DP83867 Ethernet PHY driver patch

SGMII autonegotiation is disabled in the PCS/PMA or SGMII core for port 3, therefore we need to modify the driver so that it can also disable SGMII autonegotiation in the PHY.

To allow for this, we patch the DP83867 driver to accept an extra property in the device tree:

- `ti,dp83867-sgmii-autoneg-dis`: When added to the GEM node, this will disable the SGMII autonegotiation feature when the PHY is configured (eg. ipconfig eth0 up)

This property should be included in the `gem3` node or the `axi_ethernet_3` node of the device tree (depending on the Vivado design being used).

The source code for this patch can be found in this path of the Github repo: `PetaLinux/src/common/project-spec/meta-user/recipes-kernel/linux/linux-xlnx`

ZynqMP FSBL hooks patch

This patch modifies the ZynqMP FSBL to add code to the *XFsbl_HookBeforeHandoff* which is executed before the FSBL hands over control to U-Boot. This code is necessary for initialization of the 96B Quad Ethernet Mezzanine and the PCS/PMA or SGMII IP cores, so that U-Boot and Linux can make use of the Ethernet ports. The added code does the following:

1. Initializes GEM0 so that it's MDIO interface can be used (we need it to communicate with the external PHYs and the PCS/PMA or SGMII IP cores)
2. Assert reset of PCS/PMA or SGMII IP core
3. Hardware reset the 4x Ethernet PHYs and release from reset
4. Enable the 625MHz SGMII output clock of the PHY of port 3 of the 96B Quad Ethernet Mezzanine card (PHY address 0xF). This clock is required by the PCS/PMA or SGMII IP core
5. Release the PCS/PMA or SGMII IP core from reset
6. Disable ISOLATE bit on all PCS/PMA or SGMII IP cores, and enable autonegotiation on those cores for ports 0-2. Note that port 3 cannot support SGMII autonegotiation.

The source code for this patch can be found in this path of the Github repo: `PetaLinux/src/common/project-spec/meta-user/recipes-bsp/fsbl/files`

3.8 References

3.8.1 96B Quad Ethernet Mezzanine Board Files

1. 96B Quad Ethernet Mezzanine Rev-A Schematics PDF
2. 96B Quad Ethernet Mezzanine Rev-A Assembly Drawing PDF
3. 96B Quad Ethernet Mezzanine Rev-A 3D STEP model

3.8.2 Ultra96

1. Ultra96 product page
2. Ultra96 documentation page

3.8.3 Part Datasheets

Use the links below to access the datasheets of the significant parts on the mezzanine card:

1. Gigabit Ethernet PHY, DP83867, Texas Instruments
2. Quad RJ45 connector, 0826-1X4T-23-F, Bel Fuse Inc.
3. Switching regulator, TPS82150SILT, Texas Instruments

4. Non-inverting buffer, NC7WV16P6X, ON Semiconductor
5. Crystal 25MHz, 8Z-25.000MAAJ-T, TXC
6. Tactile switch, B3U-1000P, Omron
7. 40-pin Pin Socket 2mm, 55510-140LF, Amphenol FCI
8. 40-pin Pin Header 2mm, 57202-G52-20LF, Amphenol FCI
9. 60-pin Header 0.8mm, 61083-063402LF, Amphenol FCI

3.8.4 Accessories

Extender for stacking a second mezzanine card:

1. 40-pin Pin socket with extended pins, F263-1220A0BSYE1, Yxcon