**Lime Microsystems Limited**

Surrey Technology Centre

Occam Road

The Surrey Research Park

Guildford, Surrey GU2 7YG

United Kingdom

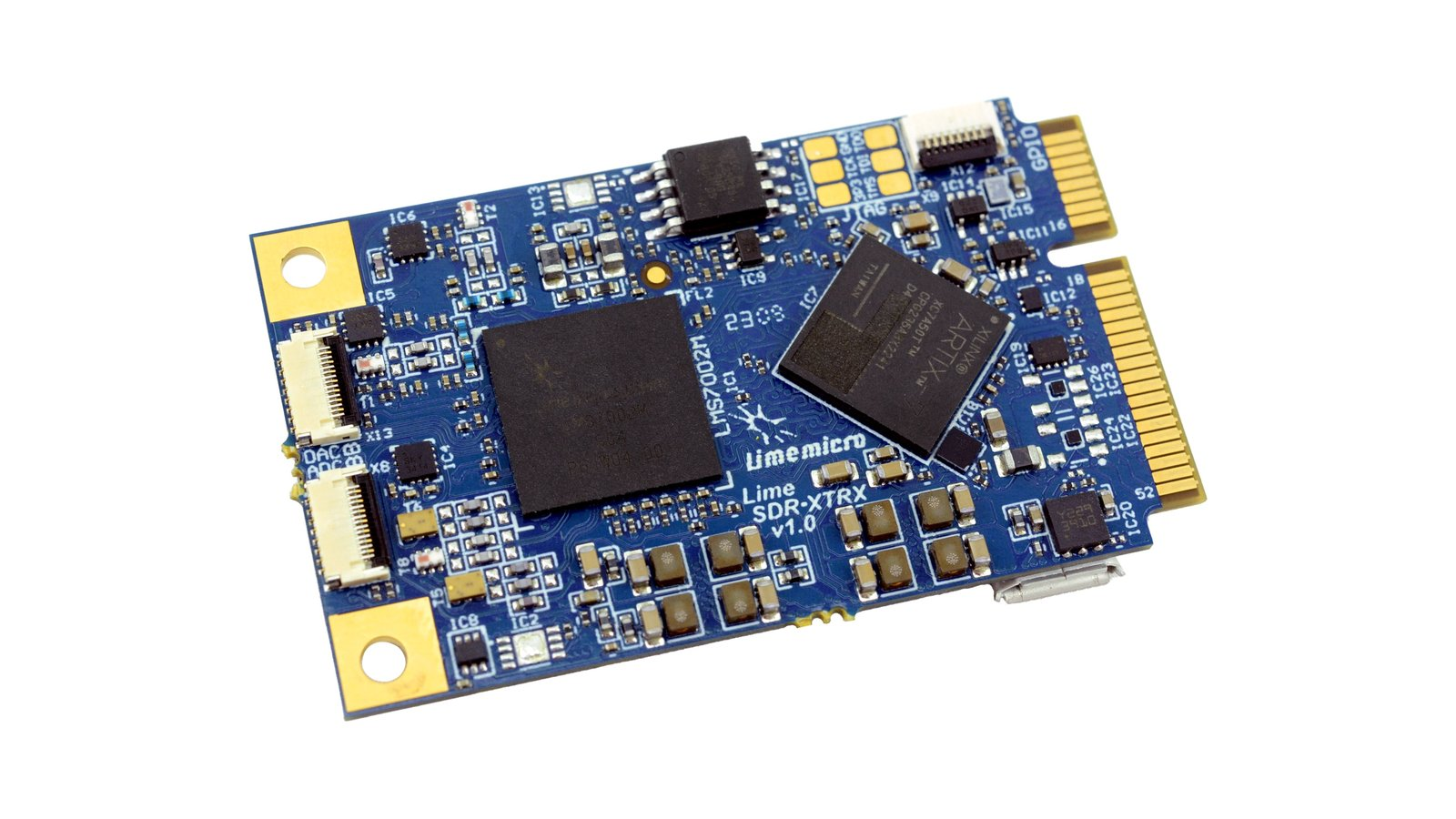
Tel: +44 (0) 1483 685 063

e-mail: [enquiries@limemicro.com](mailto:enquiries@limemicro.com)

**LimeSDR-XTRX v1.2**

**User Manual**

***- Gateware description****-*



REVISION HISTORY

The following table shows the revision history of this document:

|  |  |  |
| --- | --- | --- |
| **Date** | **Version** | **Description of Revisions** |
| 22/01/2024 | 1.00 | Initial version |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

**Table of Contents**

[1 Introduction 4](#__RefHeading___Toc31852_752810035)

[1.1 Software and hardware requirements 5](#__RefHeading___Toc31854_752810035)

[2 Building the gateware 5](#__RefHeading___Toc10904_3865674834)

[3 Board programming 5](#__RefHeading___Toc10906_3865674834)

[4 Gateware description 6](#__RefHeading___Toc2217_636601754)

[4.1 Main block diagram 6](#__RefHeading___Toc2219_636601754)

[4.1.1 PCIe interface with host - *inst0\_pcie\_top* 8](#__RefHeading___Toc2221_636601754)

[4.1.2 Softcore CPU – inst1\_*cpu\_top* 12](#__RefHeading___Toc2225_636601754)

[4.1.2.1 FPGACFG module registers 16](#__RefHeading___Toc9367_3865674834)

[4.1.2.2 PLLCFG module registers 20](#__RefHeading___Toc9369_3865674834)

[4.1.2.3 TSTCFG module registers 22](#__RefHeading___Toc9371_3865674834)

[4.1.2.4 MEMCFG module registers 24](#__RefHeading___Toc9373_3865674834)

[4.1.3 Design clocks – inst2\_*pll\_top* 25](#__RefHeading___Toc2225_636601754_Copy_1)

[4.1.4 Receive and transmit interface – inst3\_*rxtx\_top* 28](#__RefHeading___Toc2225_636601754_Copy_1_)

[4.1.4.1 Receive interface – *rx\_path\_top* 32](#__RefHeading___Toc2225_636601754_Copy_11)

[4.1.4.2 Transmit interface – t*x\_path\_top* 34](#__RefHeading___Toc2225_636601754_Copy_12)

[4.1.5 *LMS7002 digital interface – inst4\_lms7002\_top* 36](#__RefHeading___Toc2225_636601754_Copy_13)

[4.1.6 RF control logic – inst5\_tdd\_control 40](#__RefHeading___Toc10908_3865674834)

[4.1.7 Test modules – inst6\_tst\_top 41](#__RefHeading___Toc10910_3865674834)

[4.2 Clock network 42](#__RefHeading___Toc2227_636601754)

# Introduction

This is the user manual for LimeSDR-XTRX gateware. This document provides build instructions and information on features is implemented in this gateware. It contains the following features:

* PCIe connection to transfer data between host and FPGA
* LMS64 protocol implementation in soft-core CPU
* User accessible configuration registers
* Interface to LMS7002 LimeLightTM digital IQ interface in TRXIQ double data rate mode
* TX samples synchronization with RX samples time stamp
* Reconfigurable PLL blocks for LMS7002 clocking

It is assumed that user is familiar with hardware if not it is suggested to review available user guides and hardware description manuals first.

## Software and hardware requirements

Required hardware:

* [LimeSDR-XTRX v1.2](https://github.com/myriadrf/LimeSDR-XTRX) - Lime Microsystems mini PCIe expansion card SDR board

Required gateware:

* <https://github.com/myriadrf/LimeSDR-XTRX_GW> – gateware for LimeSDR-XTRX v1.2

Required software:

* [Vivado 2022.1 Standard edition](https://www.xilinx.com/support/download/index.html/content/xilinx/en/downloadNav/vivado-design-tools/2022-1.html) – Xilinx FPGA design software

# Building the gateware

Instructions for building the gateware can be found in the [README file](https://github.com/myriadrf/LimeSDR-XTRX_GW/blob/master/README.md) of the gateware repository.

# Board programming

Instructions for programming the board can be found in the [README file](https://github.com/myriadrf/LimeSDR-XTRX_GW/blob/master/README.md)of the gateware repository.

# Gateware description

Description for LimeSDR-XTRX gateware can be found in this chapter.

## Main block diagram

Top level file LimeSDR\_XTRX\_top.vhd has all required external ports and lower level module instantiated. It also has some reset logic and clocks generated for rest of the instances.

LimeSDR-XTRX provides a DMA interface with a PCIe host. There are two channels (F2H\_C0 and H2F\_C0) implemented for control data and two channels for stream data (F2H\_S0 and H2F\_S0). Control endpoints are connected to the MicroBlaze soft-core processor, which provides SPI and I2C communication interfaces for the LMS7002M chip, XO DAC, FLASH, and EEPROM memories. MicroBlaze also provides access to internal SPI configuration registers. Stream channels are dedicated to receiving and sending IQ data from/to the LMS7002M.

High level description for instances can be found in Table 1 and main block diagram in Figure 1. More details for each instance can be found in the following sections.

Figure 1: LimeMM-X8 main block diagram

Table 1: Instances of main block diagram

|  |  |
| --- | --- |
| **Instance name** | **Description** |
| LimeSDR\_XTRX\_top | Top module |
| inst0\_pcie\_top | PCIe bus implementation |
| inst1\_cpu | Microblaze soft-core CPU |
| inst2\_pll\_top | Clock control |
| inst3\_rxtx\_top | Receive and transmit logic between FPGA and external LMS7002 transceiver. |
| inst4\_lms7002\_top | Digital data and control interface for LMS7002 IC. |
| inst5\_tdd\_control | Control logic for TDD RF control |
| inst6\_tst\_top | Clock and GNSS test modules |

### PCIe interface with host - *inst0\_pcie\_top*

Module *pcie\_top* implements PCIe x2 Gen2 endpoint. This block uses two GTP transceivers and AMD 7 Series Integrated Block for PCI Express v3.3 IP core as a physical layer. Module *pcie\_top* handles TLP layer and provides one channel for sending/receiving LMS64 control packets and one DMA channel for sending/receiving stream data packets.

More information about litepcie\_core can be found in Github repository: <https://github.com/enjoy-digital/litepcie/tree/master>.

Table 2: pcie\_top module generics

| **Generic name** | **Type** | **Value** | **Description** |
| --- | --- | --- | --- |
| g\_DEV\_FAMILY | string | "" | Device family |
| g\_S0\_DATA\_WIDTH | integer | 32 | Stream 0 data width |
| g\_C0\_DATA\_WIDTH | integer | 8 | Control 0 data width |
| g\_H2F\_S0\_0\_RDUSEDW\_WIDTH | integer | 11 | Stream 0\_0 FIFO read used words width |
| g\_H2F\_S0\_0\_RWIDTH | integer | 32 | Stream 0\_0 FIFO read data with |
| g\_H2F\_S0\_1\_RDUSEDW\_WIDTH | integer | 11 | Stream 0\_1 FIFO read used words width |
| g\_H2F\_S0\_1\_RWIDTH | integer | 32 | Stream 0\_1 FIFO read data with |
| g\_F2H\_S0\_WRUSEDW\_WIDTH | integer | 10 | Stream 0 FIFO write used words width |
| g\_F2H\_S0\_WWIDTH | integer | 64 | Stream 0 FIFO write width |
| g\_H2F\_C0\_RDUSEDW\_WIDTH | integer | 11 | Control 0 FIFO read used words width |
| g\_H2F\_C0\_RWIDTH | integer | 8 | Control 0 FIFO read width |
| g\_F2H\_C0\_WRUSEDW\_WIDTH | integer | 11 | Control 0 FIFO write used words width |
| g\_F2H\_C0\_WWIDTH | integer | 8 | Control 0 FIFO write width |

*pcie\_top* port descriptions can be found in Table 3.

Table 3 *pcie\_top* port description

| **Port name** | **Direction** | **Type** | **Description** |
| --- | --- | --- | --- |
| clk | out | std\_logic | Internal logic clock (125Mhz) |
| reset\_n | in | std\_logic | Active low reset |
| pcie\_perstn | in | std\_logic | PCIe fundamental reset |
| pcie\_refclk\_p | in | std\_logic | PCIe reference clock |
| pcie\_refclk\_n | in | std\_logic | PCIe reference clock |
| pcie\_rx\_p | in | std\_logic\_vector(1 downto 0) | PCIe receiver |
| pcie\_rx\_n | in | std\_logic\_vector(1 downto 0) | PCIe receiver |
| pcie\_tx\_p | out | std\_logic\_vector(1 downto 0) | PCIe transmitter |
| pcie\_tx\_n | out | std\_logic\_vector(1 downto 0) | PCIe transmitter |
| H2F\_S0\_sel | in | std\_logic | Stream select: 0 - S0\_0, 1 - S0\_1 |
| H2F\_S0\_dma\_en | out | std\_logic | Host->FPGA stream ready |
| S0\_rx\_en | in | std\_logic | Stream 0 enable |
| F2H\_S0\_open | out | std\_logic | FPGA->Host stream 0 ready |
| H2F\_S0\_0 | in | Virtual bus | Stream 0 endpoint FIFO 0 (Host->FPGA) |
| H2F\_S0\_1 | in | Virtual bus | Stream 0 endpoint FIFO 1 (Host->FPGA) |
| F2H\_S0 | out | Virtual bus | Stream 0 endpoint FIFO (FPGA->Host) |
| H2F\_C0 | in | Virtual bus | Control endpoint FIFO (Host->FPGA) |
| F2H\_C0 | out | Virtual bus | Control endpoint FIFO (FPGA->Host) |

Table 4 H2F\_S0\_0 virtual bus description

| **Port name** | **Direction** | **Type** | **Description** |
| --- | --- | --- | --- |
| H2F\_S0\_0\_rdclk | in | std\_logic | Read clock |
| H2F\_S0\_0\_aclrn | in | std\_logic | Asynchronous clear |
| H2F\_S0\_0\_rd | in | std\_logic | Read enable |
| H2F\_S0\_0\_rdata | out | std\_logic\_vector(g\_H2F\_S0\_0\_RWIDTH-1 downto 0) | Read data |
| H2F\_S0\_0\_rempty | out | std\_logic | Read empty |
| H2F\_S0\_0\_rdusedw | out | std\_logic\_vector(g\_H2F\_S0\_0\_RDUSEDW\_WIDTH-1 downto 0) | Read used words |

Table 5 H2F\_S0\_1 virtual bus description

| **Port name** | **Direction** | **Type** | **Description** |
| --- | --- | --- | --- |
| H2F\_S0\_1\_rdclk | in | std\_logic | Read clock |
| H2F\_S0\_1\_aclrn | in | std\_logic | Asynchronous clear |
| H2F\_S0\_1\_rd | in | std\_logic | Read enable |
| H2F\_S0\_1\_rdata | out | std\_logic\_vector(g\_H2F\_S0\_1\_RWIDTH-1 downto 0) | Read data |
| H2F\_S0\_1\_rempty | out | std\_logic | Read empty |
| H2F\_S0\_1\_rdusedw | out | std\_logic\_vector(g\_H2F\_S0\_1\_RDUSEDW\_WIDTH-1 downto 0) | Read used words |

Table 6 F2H\_S0 virtual bus description

| **Port name** | **Direction** | **Type** | **Description** |
| --- | --- | --- | --- |
| F2H\_S0\_wclk | in | std\_logic | Write clock |
| F2H\_S0\_aclrn | in | std\_logic | Asynchronous clear |
| F2H\_S0\_wr | in | std\_logic | Write enable |
| F2H\_S0\_wdata | in | std\_logic\_vector(g\_F2H\_S0\_WWIDTH-1 downto 0) | Write data |
| F2H\_S0\_wfull | out | std\_logic | Write full |
| F2H\_S0\_wrusedw | out | std\_logic\_vector(g\_F2H\_S0\_WRUSEDW\_WIDTH-1 downto 0) | Write used words |

Table 7 H2F\_C0 virtual bus description

| **Port name** | **Direction** | **Type** | **Description** |
| --- | --- | --- | --- |
| H2F\_C0\_rdclk | in | std\_logic | Read clock |
| H2F\_C0\_aclrn | in | std\_logic | Asynchronous clear |
| H2F\_C0\_rd | in | std\_logic | Read enable |
| H2F\_C0\_rdata | out | std\_logic\_vector(g\_H2F\_C0\_RWIDTH-1 downto 0) | Read data |
| H2F\_C0\_rempty | out | std\_logic | Read empty |

Table 8 F2H\_C0 virtual bus description

| **Port name** | **Direction** | **Type** | **Description** |
| --- | --- | --- | --- |
| F2H\_C0\_wclk | in | std\_logic | Write clock |
| F2H\_C0\_aclrn | in | std\_logic | Asynchronous clear |
| F2H\_C0\_wr | in | std\_logic | Write enable |
| F2H\_C0\_wdata | in | std\_logic\_vector(g\_F2H\_C0\_WWIDTH-1 downto 0) | Write data |
| F2H\_C0\_wfull | out | std\_logic | Write full |

### Softcore CPU – inst1\_*cpu\_top*

Module *cpu\_top* is a wrapper for AMD XILINX MicroBlaze Soft Processor Core and user accessible register blocks. Application code of MicroBlaze has following functions:

* LMS64 protocol decoder/encoder
* SPI access to user register blocks
* I2C device control

High level block diagram can be found in Figure 2

Figure 2: cpu\_top block diagram

Module parameter descriptions can be found in Table 9.

Table 9: cpu\_top module parameters

| **Generic name** | **Type** | **Value** | **Description** |
| --- | --- | --- | --- |
| FPGACFG\_START\_ADDR | integer | 0 | FPGACFG register start address |
| PLLCFG\_START\_ADDR | integer | 32 | PLLCFG register start address |
| TSTCFG\_START\_ADDR | integer | 96 | TSTCFG register start address |
| PERIPHCFG\_START\_ADDR | integer | 192 | TSTCFG register start address |
| MEMCFG\_START\_ADDR | integer | 65504 | MEMCFG register start address |

Module port descriptions can be found in tables: Table 3, Table 10, Table 11, Table 12, Table 14, Table 15, Table 16, Table 29.

Table 10: exfifo\_if virtual bus ports

| **Port name** | **Direction** | **Type** | **Description** |
| --- | --- | --- | --- |
| EXFIFO\_IF\_D | in | std\_logic\_vector(31 downto 0) | Read data |
| EXFIFO\_IF\_RD | out | std\_logic | Read enable |
| EXFIFO\_IF\_RDEMPTY | in | std\_logic | Read empty |

Table 11: exfifo\_of virtual bus ports

| **Port name** | **Direction** | **Type** | **Description** |
| --- | --- | --- | --- |
| EXFIFO\_OF\_D | out | std\_logic\_vector(31 downto 0) | Write data |
| EXFIFO\_OF\_WR | out | std\_logic | Write enable |
| EXFIFO\_OF\_WRFULL | in | std\_logic | Write full |
| EXFIFO\_OF\_RST | out | std\_logic | Reset, active high |

Table 12: spi\_0 virtual bus ports

| **Port name** | **Direction** | **Type** | **Description** |
| --- | --- | --- | --- |
| SPI\_0\_MISO | in | std\_logic | Master In Slave Out |
| SPI\_0\_MOSI | out | std\_logic | Master Out Slave In |
| SPI\_0\_SCLK | out | std\_logic | Clock output |
| SPI\_0\_SS\_N | out | std\_logic\_vector(1 downto 0) | Slave Select, active low [0] - internal registers, [1] - LMS7002 |

Table 13: cpu\_top module ports

| **Port name** | **Dir****.** | **Type** | **Description** |
| --- | --- | --- | --- |
| CLK | in | std\_logic | Clock |
| RESET\_N | in | std\_logic | Reset, active low |
| GPI | in | std\_logic\_vector(7 downto 0) | General purpose inputs (UNUSED) |
| GPO | out | std\_logic\_vector(7 downto 0) | General purpose outputs (UNUSED) |
| VCTCXO\_TUNE\_EN | in | std\_logic | Unused |
| VCTCXO\_IRQ | in | std\_logic | Unused |
| PLL\_RST | out | std\_logic\_vector(1 downto 0) | PLL reset |
| PLL\_AXI\_RESETN\_OUT | out | std\_logic\_vector( 0 to 0) | PLL Axi interface reset |
| PLL\_FROM\_AXIM | out | t\_FROM\_AXIM\_32x32 | PLL AXI interface CPU -> PLL |
| PLL\_TO\_AXIM | in | t\_TO\_AXIM\_32x32 | PLL AXI interface PLL -> CPU |
| PLL\_AXI\_SEL | out | std\_logic\_vector(3 downto 0) | PLL AXI slave select |
| FROM\_FPGACFG | out | t\_FROM\_FPGACFG | FPGACFG register bus Registers -> Modules |
| TO\_FPGACFG | in | t\_TO\_FPGACFG | FPGACFG register bus Modules -> Registers |
| FROM\_PLLCFG | out | t\_FROM\_PLLCFG | PLLCFG register bus Registers -> Modules |
| TO\_PLLCFG | in | t\_TO\_PLLCFG | PLLCFG register bus Modules -> Registers |
| FROM\_TSTCFG | out | t\_FROM\_TSTCFG | TSTCFG register bus Registers -> Modules |
| TO\_TSTCFG | in | t\_TO\_TSTCFG | TSTCFG register bus Modules -> Registers |
| FROM\_PERIPHCFG | out | t\_FROM\_PERIPHCFG | PERIPHCFG register bus Registers -> Modules |
| TO\_PERIPHCFG | in | t\_TO\_PERIPHCFG | PERIPHCFG register bus Modules -> Registers |
| TO\_MEMCFG | in | t\_TO\_MEMCFG | MEMCFG register bus Registers -> Modules |
| FROM\_MEMCFG | out | t\_FROM\_MEMCFG | MEMCFG register bus Modules -> Registers |
| SMPL\_CMP\_EN | out | std\_logic\_vector( 0 downto 0) | Sample compare module enable |
| SMPL\_CMP\_STATUS | in | std\_logic\_vector( 1 downto 0) | Sample compare module status |
| EXFIFO\_IF | in | Virtual bus | Control packet fifo Host -> FPGA |
| EXFIFO\_OF | out | Virtual bus | Control packet fifo FPGA -> HOST |
| SPI\_0 | out | Virtual bus | SPI interface 0, used for internal registers and LMS7002 |
| I2C\_1 | out | Virtual bus | I2C interface 1, used for Temperature sensor, XO DAC, Switching voltage regulator 1 (IC22) |
| I2C\_2 | out | Virtual bus | I2C interface 2, used for Switching voltage regulator 2 (IC31) |
| FPGA\_CFG | out | Virtual bus | SPI interface for configuration flash |
| AVMM\_M0 | out | Virtual bus | Avalon master interface (UNUSED) |

Table 14: I2C\_1 virtual bus ports

| **Port name** | **Direction** | **Type** | **Description** |
| --- | --- | --- | --- |
| I2C\_1\_SCL | inout | std\_logic | Clock signal |
| I2C\_1\_SDA | inout | std\_logic | Data signal |

Table 15: I2C\_2 virtual bus ports

| **Port name** | **Direction** | **Type** | **Description** |
| --- | --- | --- | --- |
| I2C\_2\_SCL | inout | std\_logic | Clock signal |
| I2C\_2\_SDA | inout | std\_logic | Data signal |

Table 16: fpga\_cfg virtual bus ports

| **Port name** | **Direction** | **Type** | **Description** |
| --- | --- | --- | --- |
| FPGA\_CFG\_QSPI\_MISO | in | std\_logic | Master In Slave Out |
| FPGA\_CFG\_QSPI\_MOSI | out | std\_logic | Master Out Slave In |
| FPGA\_CFG\_QSPI\_SS\_N | out | std\_logic | Slave Select, active low |

Table 17: avmm\_m0 virtual bus ports

| **Port name** | **Direction** | **Type** | **Description** |
| --- | --- | --- | --- |
| AVMM\_M0\_ADDRESS | out | std\_logic\_vector(7 downto 0) | Adress |
| AVMM\_M0\_READ | out | std\_logic | Read |
| AVMM\_M0\_WAITREQUEST | in | std\_logic | Wait request |
| AVMM\_M0\_READDATA | in | std\_logic\_vector(7 downto 0) | Read data |
| AVMM\_M0\_READDATAVALID | in | std\_logic | Read data valid |
| AVMM\_M0\_WRITE | out | std\_logic | Write |
| AVMM\_M0\_WRITEDATA | out | std\_logic\_vector(7 downto 0) | Write data |
| AVMM\_M0\_CLK\_CLK | out | std\_logic | Clock |
| AVMM\_M0\_RESET\_RESET | out | std\_logic | Reset |

#### FPGACFG module registers

Table 18: FPGACFG module registers (0x0000-0x0009)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Address | Def. Value | Bits | Name | Description |
| 0x0000 | 0x001B | 15-0 | board\_id | Board ID (read only) |
| 0x0001 | - | 15-0 | Major rev | Major revision (read only). |
| 0x0002 | - | 15-0 | compile\_rev | Compile revision (read only). |
| 0x0003 | - | 15-0 | Reserved |  |
| 0x0004 | 0x0000 | 15-0 | Reserved |  |
| 0x0005 | 0x0000 | 15-0 | Reserved |  |
| 0x0006 | 0x0000 | 15-0 | Reserved |  |
| 0x0007 | 0x0003 | 15-2 | Reserved |  |
| 1-0 | ch\_en | Channel enable |
| 01 – Channel A |
| 10 – Channel B |
| 11 – Channels A and B |
| 0x0008 | 0x0102 | 15-10 | Reserved |  |
| 9 | synch\_dis | Packets synchronization using timestamps: |
| 0 - Enabled |
| 1 - Disabled (Default) |
| 8 | mimo\_int\_en | MIMO mode: |
| 0 - Disabled |
| 1 - Enabled (Default) |
| 7 | trxiq\_pulse | TRXIQ\_pulse mode: |
| 0 - OFF (Default) |
| 1 – ON |
| 6 | ddr\_en | DIQ interface mode: |
| 0 - SDR |
| 1 - DDR (Default) |
| 5-2 | Reserved |  |
| 1-0 | smpl\_width | Interface sample width selection: |
| "10" - 12bit (Default) |
| "01" - Do not use |
| "00" - 16bit |
| 0x0009 | 0x0003 | 15-2 | Reserved |  |
| 1 | txpct\_loss\_clr | TX packets dropping flag clear: |
| 0 - Normal operation (Default) |
| 1 - Rising edge clears flag |
| 0 | smpl\_nr\_clr | Reset\_timestamp: |
| 0 - Normal operation (Default) |
| 1 - Timestamp is cleared |

Table 19: FPGACFG module registers (0x000A-0x0010)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Address | Def. Value | Bits | Name | Description |
| 0x000A | 0x0000 | 15-12 | Reserved |  |
| 11 | rf\_sw\_auto\_en | Control of RF switches by internal TDD signal: |
| 0 – Disabled (Default) |
| 1 – Enabled |
| 10 | tx\_cnt\_en | Counter test pattern on TX: |
| 0 – Disabled (Default) |
| 1 – Enabled |
| 9 | tx\_ptrn\_en | Test pattern on TX: |
| 0 – Disabled (Default) |
| 1 – Enabled |
| 8 | rx\_ptrn\_en | Test pattern on RX: |
| 0 – Disabled (Default) |
| 1 – Enabled |
| 7 | tdd\_invert | Invert external TDD signal: |
| 0 – Disabled (Default) |
| 1 – Enabled |
| 6 | tdd\_auto\_en | Control external TDD signal by internal TDD signal: |
| 0 – Disabled |
| 1 – Enabled |
| 5 | tdd\_manual | Manual value of external TDD signal |
| 4 | tx\_rf\_sw | TX RF switch selection: |
| 0 – TX 2 (Default) |
| 1 – TX 1 |
| 3-2 | rx\_rf\_sw | RX RF switch selection: |
| 00 – RX\_W |
| 01 – RX\_L |
| 10 – RX\_H |
| 11 – No connection |
| 1 | Reserved |  |
| 0 | rx\_en | RX/TX unified enable: |
| 0 – Disabled (Default) |
| 1 – Enabled |
| 0x000B | 0x0000 | 15-0 | Reserved |  |
| 0x000C | 0x0003 | 15-0 | Reserved |  |
| 0x000D | 0x0000 | 15-0 | Reserved |  |
| 0x000E | 0x0000 | 15-0 | RX\_PACKET\_SAMPLES | RX packet size in samples |
| 0x000F | 0x03FC | 15-0 | Reserved |  |
| 0x0010 | 0x0001 | 15-0 | txant\_pre | How many samples to delay turning on internal TDD singal |

Table 20: FPGACFG module registers (0x0011-0x0017)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Address | Def. Value | Bits | Name | Description |
| 0x0011 | 0x0001 | 15-0 | txant\_post | How many samples to delay turning off internal TDD singal |
| 0x0012 | 0xFFFF | 15-0 | Reserved |  |
| 0x0013 | 0x6F6B | 15-8 | Reserved |  |
| 7 | LMS\_TXRXEN\_MUX\_SEL | Control LMS TX/RXEN signals by internal TDD signal |
| 0 – Disabled (Default) |
| 1 – Enabled |
| 6 | LMS1\_RXEN | RX hard enable: |
| 0 - Disabled |
| 1 - Enabled (Default) |
| 5 | LMS1\_TXEN | TX hard enable: |
| 0 - Disabled |
| 1 - Enabled (Default) |
| 4 | LMS1\_TXNRX2 | Port 2 mode selection: |
| 0 - TXIQ (Default) |
| 1 - RXIQ |
| 3 | LMS1\_TXNRX1 | Port 1 mode selection: |
| 0 - TXIQ |
| 1 - RXIQ (Default) |
| 2 | LMS1\_CORE\_LDO\_EN | Internal LDO control(UNUSED): |
| 0 - Disabled (Default) |
| 1 – Enabled |
| 1 | LMS1\_RESET | Hardware reset: |
| 0 - Reset activated |
| 1 - Reset inactive (Default) |
| 0 | Reserved |  |
| 0x0014 | 0x0003 | 15-0 | Reserved |  |
| 0x0015 | 0x0000 | 15-0 | Reserved |  |
| 0x0016 | 0x0000 | 15-0 | Reserved |  |
| 0x0017 | 0x2340 | 15-0 | Reserved |  |

Table 21: FPGACFG module registers (0x0018-0x001F)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Address | Def. Value | Bits | Name | Description |
| 0x0018 | 0x0003 | 15-4 | Reserved |  |
| 3 | CORE\_LDO\_EN | LMS internal LDO control |
| 0 – Disabled (Default) |
| 1 – Enabled |
| 2 | EXT\_CLK | Clock source selection |
| 0 – Onboard clock (Default) |
| 1 – External clock |
| 1 | TCXO\_EN | Onboard clock enable |
| 0 – Disabled |
| 1 – Enabled (Default) |
| 0 | LMS\_RST | LMS hardware reset |
| 0 - Reset activated |
| 1 - Reset inactive (Default) |
| 0x0019 | 0x1000 | 15-0 | RX\_PACKET\_SIZE | RX packet size in bytes |
| 0x001A | 0x0000 | 15-0 | Reserved |  |
| 0x001B | 0x0000 | 15-0 | Reserved |  |
| 0x001C | 0x0000 | 15-0 | Reserved |  |
| 0x001D | 0x00FF | 15-0 | Reserved |  |
| 0x001E | 0x0003 | 15-0 | Reserved |  |
| 0x001F | 0xD090 | 15-0 | Reserved |  |

#### PLLCFG module registers

Table 22: PLLCFG module registers (0x0020-0x0025)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Address | Def. Value | Bits | Name | Description |
| 0x0020 | 0x0000 | 0-15 | C1 Phase | Phase value for PLL output clock 1 |
| 0x0021 | 0x0001 | 4-15 | Reserved |  |
| 3 | phcfg\_error | Phase config error (unused) |
| 2 | phcfg\_done | Phase config done (read only) |
| 0 – Not done |
| 1 – Done |
| 1 | pllcfg\_busy | PLL config busy (read only) |
| 0 – Idle |
| 1 – Busy |
| 0 | pllcfg\_done | PLL config done (read only) |
| 0 – Not done |
| 1 – Done |
| 0x0022 | 0x0000 | 15-8 | pllcfg\_err | PLL config error (unused) |
| 7-2 | pll\_lock | Reserved |
| 1 | RX PLL |
| 0 – No Lock |
| 1 – Locked |
| 0 | TX PLL |
| 0 – No Lock |
| 1 – Locked |
| 0x0023 | 0x0000 | 15 | Reserved |  |
| 14 | phcfg\_mode | PLL phase configuration mode: |
| 0 – Manual |
| 1 – Auto |
| 13-8 | Reserved |  |
| 7-3 | pll\_ind | PLL index for reconfiguration: |
| 0000 - TX PLL |
| 0001 - RX PLL |
| Do not use other index values |
| 2 | Reserved |  |
| 1 | phcfg\_start | Start phase configuration |
| 0 – Phase configuration inactive |
| 0 to 1 transition – start configuration |
| 0 | Reserved |  |
| 0x0024 | 0x0000 | 15-0 | Reserved |  |
| 0x0025 | 0x01F0 | 15-0 | Reserved |  |

Table 23: FPGACFG module registers (0x0026-0x003F)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Address | Def. Value | Bits | Name | Description |
| 0x0026 | 0x000A | 15-3 | Reserved |  |
| 2 | m\_byp | PLL multiplier bypass |
| 1 | Reserved |  |
| 0 | n\_byp | PLL divider bypass |
| 0x0027 | 0xAAA | 15-3 | Reserved |  |
| 2 | c1\_byp | PLL output 1 divider bypass |
| 1 | Reserved |  |
| 0 | c0\_byp | PLL output 0 divider bypass |
| 0x0028 | 0xAAAA | 15-0 | Reserved |  |
| 0x0029 | 0xAAAA | 15-0 | Reserved |  |
| 0x002A | 0x0000 | 15-0 | n\_cnt | PLL divider value |
| 0x002B | 0x0000 | 15-0 | m\_cnt | PLL multiplier value |
| 0x002C | 0x0000 | 15-0 | m\_frac(LSB) | PLL multiplier fractional value |
| 0x002D | 0x0000 | 15-0 | m\_frac(MSB) | PLL multiplier fractional value |
| 0x002E | 0x0000 | 15-0 | c0\_cnt | PLL output 0 divider value |
| 0x002F | 0x0000 | 15-0 | c1\_cnt | PLL output 1 divider value |
| 0x0030 | 0xEFFF | 15-0 | auto\_phcfg\_smpls | Number of samples to use during auto phase configuration |
| 0x0031 | 0x0000 | 15-0 | Reserved |  |
| 0x0032 | 0x0000 | 15-0 | Reserved |  |
| 0x0033 | 0x0000 | 15-0 | Reserved |  |
| 0x0034 | 0x0000 | 15-0 | Reserved |  |
| 0x0035 | 0x0000 | 15-0 | Reserved |  |
| 0x0036 | 0x0000 | 15-0 | Reserved |  |
| 0x0037 | 0x0000 | 15-0 | Reserved |  |
| 0x0038 | 0x0000 | 15-0 | Reserved |  |
| 0x0039 | 0x0000 | 15-0 | Reserved |  |
| 0x003A | 0x0000 | 15-0 | Reserved |  |
| 0x003B | 0x0000 | 15-0 | Reserved |  |
| 0x003C | 0x0000 | 15-0 | Reserved |  |
| 0x003D | 0x0000 | 15-0 | Reserved |  |
| 0x003E | 0x0000 | 15-0 | Reserved |  |
| 0x003F | 0x0000 | 15-0 | Reserved |  |

#### TSTCFG module registers

Table 24: TSTCFG module registers (0x0060-0x006D)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Address | Def. Value | Bits | Name | Description |
| 0x0060 | 0x0000 | 15-0 | Reserved |  |
| 0x0061 | 0x0000 | 15-3 | Reserved |  |
| 2 | test\_en | GNSS test: |
| 0 – Idle (Default) |
| 1 – Test started |
| 1 | LMS\_TX\_CLK test: |
| 0 – Idle (Default) |
| 1 – Test started |
| 0 | sys\_clk test: |
| 0 – Idle (Default) |
| 1 – Test started |
| 0x0062 | 0x0000 | 15-0 | Reserved |  |
| 0x0063 | 0x0000 | 15-0 | Reserved |  |
| 0x0064 | 0x0000 | 15-0 | Reserved |  |
| 0x0065 | 0x0000 | 15-3 | Reserved |  |
| 2 | test\_cmplt | GNSS test: |
| 0 – Test not finished |
| 1 – Test finished |
| 1 | LMS\_TX\_CLK test: |
| 0 – Test not finished |
| 1 – Test finished |
| 0 | sys\_clk\_test |
| 0 – Test not finished |
| 1 – Test finished |
| 0x0066 | 0x0000 | 15-0 | Reserved |  |
| 0x0067 | 0x0000 | 15-3 | Reserved |  |
| 2 | test\_rez | GNSS test result. (not implemented) |
| 1 | LMS\_TX\_CLK test result. (not implemented) |
| 0 | sys\_clk test result. (not implemented) |
| 0x0068 | 0x0000 | 15-0 | Reserved |  |
| 0x0069 | 0x0000 | 15-0 | sys\_clk\_cnt | Number of sys\_clk cycles counted by sys\_clk test. Different values on subsequent reads indicate clock is active |
| 0x006A | 0x0000 | 15-0 | Reserved |  |
| 0x006B | 0x0000 | 15-0 | Reserved |  |
| 0x006C | 0x0000 | 15-0 | Reserved |  |
| 0x006D | 0x0000 | 15-0 | Reserved |  |

Table 25: TSTCFG module registers (0x006E-0x007F)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Address | Def. Value | Bits | Name | Description |
| 0x006E | 0x0000 | 15-0 | Reserved |  |
| 0x006F | 0x0000 | 15-0 | Reserved |  |
| 0x0070 | 0x0000 | 15-0 | Reserved |  |
| 0x0071 | 0x0000 | 15-0 | Reserved |  |
| 0x0072 | 0x0000 | 15-0 | lms\_tx\_clk\_cnt | 15-0 bits of cycles counted by lms\_tx\_clk test |
| 0x0073 | 0x0000 | 15-8 | Reserved |  |
| 7-0 | lms\_tx\_clk\_cnt | 23-16 bits of cycles counted by lms\_tx\_clk test. |
| 0x0074-0x007C | 0x0000 | 15-0 | Reserved |  |
| 0x007D | 0xAAAA | 15-0 | tx\_tst\_i | Test value for tx I channel |
| 0x007E | 0x5555 | 15-0 | tx\_tst\_q | Test value for tx Q channel |
| 0x007F | 0x0000 |  | Reserved |  |

#### PERIPHCFG module registers

Table 26: PERIPHCFG module registers (0xFFE0-0xFFFF)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Address | Def. Value | Bits | Name | Description |
| 0x00C0 | 0x0002 | 15-0 | board\_gpio\_ovrd | GPIO control override (each bit controls corresponding GPIO):  0 - Dedicated function  1 - Overridden by user (**Default**) |
| 0x00C1 | 0x0000 | 15-0 | Reserved |  |
| 0x00C2 | 0x0000 | 15-0 | Reserved |  |
| 0x00C3 | 0x0000 | 15-0 | Reserved |  |
| 0x00C4 | 0x0000 | 15-0 | board\_gpio\_dir | Onboard GPIO direction (each bit controls corresponding GPIO):  0 - Input (**Default**)  1 - Output |
| 0x00C5 | 0x0000 | 15-0 | Reserved |  |
| 0x00C6 | 0x0000 | 15-0 | Reserved | GPIO output value (each bit controls corresponding GPIO):  0 - Low level  1 - High level |
| 0x00C7 | 0x0000 | 15-0 | Reserved |  |
| 0x00C8 | 0x0000 | 15-0 | Reserved |  |
| 0x00C9 | 0x0000 | 15-0 | Reserved |  |
| 0x00CA | 0x0000 | 15-2 | Reserved |  |
| 1-0 | periph\_input\_sel | PPS source select:  0 – GNSS\_1PPS (Default)  1 – 1PPSI\_GPIO1  2 – Reserved  3 – Reserved |

#### MEMCFG module registers

Table 27: MEMCFG module registers (0xFFE0-0xFFFF)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Address | Def. Value | Bits | Name | Description |
| 0xFFE0 | 0x0000 | 15-0 | Reserved |  |
| 0xFFE1 | 0x0000 | 15-0 | Reserved |  |
| 0xFFE2 | 0x0000 | 15-0 | Reserved |  |
| 0xFFE3 | 0x0000 | 15-0 | Reserved |  |
| 0xFFE4 | 0x0000 | 15-0 | Reserved |  |
| 0xFFE5 | 0x0000 | 15-0 | Reserved |  |
| 0xFFE6 | 0x0000 | 15-0 | Reserved |  |
| 0xFFE7 | 0x0000 | 15-0 | Reserved |  |
| 0xFFE8 | 0x0000 | 15-0 | Reserved |  |
| 0xFFE9 | 0x0000 | 15-0 | Reserved |  |
| 0xFFEA | 0x0000 | 15-0 | Reserved |  |
| 0xFFEB | 0x0000 | 15-0 | Reserved |  |
| 0xFFEC | 0x0000 | 15-0 | Reserved |  |
| 0xFFED | 0x0000 | 15-0 | Reserved |  |
| 0xFFEE | 0x0000 | 15-0 | Reserved |  |
| 0xFFEF | 0x0000 | 15-0 | Reserved |  |
| 0xFFF0 | 0x0000 | 15-0 | Reserved |  |
| 0xFFF1 | 0x0000 | 15-0 | Reserved |  |
| 0xFFF2 | 0x0000 | 15-0 | Reserved |  |
| 0xFFF3 | 0x0000 | 15-0 | Reserved |  |
| 0xFFF4 | 0x0000 | 15-0 | Reserved |  |
| 0xFFF5 | 0x0000 | 15-0 | Reserved |  |
| 0xFFF6 | 0x0000 | 15-0 | Reserved |  |
| 0xFFF7 | 0x0000 | 15-0 | Reserved |  |
| 0xFFF8 | 0x0000 | 15-0 | Reserved |  |
| 0xFFF9 | 0x0000 | 15-0 | Reserved |  |
| 0xFFFA | 0x0000 | 15-0 | Reserved |  |
| 0xFFFB | 0x0000 | 15-0 | Reserved |  |
| 0xFFFC | 0x0000 | 15-0 | Reserved |  |
| 0xFFFD | 0x0000 | 15-0 | Reserved |  |
| 0xFFFE | 0x0000 | 15-0 | Reserved |  |
| 0xFFFF | 0x0000 | 15-0 | Reserved |  |

### Design clocks – inst2\_*pll\_top*

Module *pll\_top* Figure 3 provides clocks for LMS7002 RX and TX digital interfaces. This module contains two dynamically reconfigurable MMCMs. Clock frequency and phase relationship can be changed while FPGA is in user mode.

OLE objectFigure 3: pll\_top module block diagram

Table 28: pll\_top generics

| **Generic name** | **Type** | **Value** | **Description** |
| --- | --- | --- | --- |
| INTENDED\_DEVICE\_FAMILY | STRING | "" | Device family |
| N\_PLL | integer | 2 | Number of PLLs |
| LMS1\_TXPLL\_DRCT\_C0\_NDLY | integer | 1 | Direct TX clock delay (Obsolete) |
| LMS1\_TXPLL\_DRCT\_C1\_NDLY | integer | 2 | Direct TX clock delay (Obsolete) |
| LMS1\_RXPLL\_DRCT\_C0\_NDLY | integer | 1 | Direct RX clock delay (Obsolete) |
| LMS1\_RXPLL\_DRCT\_C1\_NDLY | integer | 2 | Direct RX clock delay (Obsolete) |

Table 29: pll\_top ports

| **Port name** | **Direction** | **Type** | **Description** |
| --- | --- | --- | --- |
| lms1\_smpl\_cmp\_en | out | std\_logic | Sample compare enable |
| lms1\_smpl\_cmp\_done | in | std\_logic | Sample compare done |
| lms1\_smpl\_cmp\_error | in | std\_logic | Sample compare error |
| lms1\_smpl\_cmp\_cnt | out | std\_logic\_vector(15 downto 0) | Number of samples to compare |
| rcnfg\_axi\_clk | in | std\_logic | AXI bus reconfiguration clock |
| rcnfg\_axi\_reset\_n | in | std\_logic | AXI bus active low reset |
| rcnfg\_from\_axim | in | t\_FROM\_AXIM\_32x32 | AXI bus inputs |
| rcnfg\_to\_axim | out | t\_TO\_AXIM\_32x32 | AXI bus outputs |
| rcnfg\_sel | in | std\_logic\_vector(3 downto 0) | Reconfiguration select |
| to\_pllcfg | out | t\_TO\_PLLCFG | Output signals PLLCFG registers |
| from\_pllcfg | in | t\_FROM\_PLLCFG | Input signals from PLLCFG registers |
| lms1\_txpll | in | Virtual bus | LMS#1 TX PLL ports |
| lms1\_rxpll | in | Virtual bus | LMS#1 RX PLL ports |

Table 30: lms1\_txpll virtual bus

| **Port name** | **Direction** | **Type** | **Description** |
| --- | --- | --- | --- |
| lms1\_txpll\_inclk | in | std\_logic | TXPLL input clock |
| lms1\_txpll\_reconfig\_clk | in | std\_logic | TXPLL reconfiguration clock |
| lms1\_txpll\_logic\_reset\_n | in | std\_logic | TXPLL logic active low reset |
| lms1\_txpll\_clk\_ena | in | std\_logic\_vector(1 downto 0) | TXPLL clock enable |
| lms1\_txpll\_drct\_clk\_en | in | std\_logic\_vector(1 downto 0) | TXPLL direct clock enable (Obsolete) |
| lms1\_txpll\_c0 | out | std\_logic | TXPLL clock output c0 |
| lms1\_txpll\_c1 | out | std\_logic | TXPLL clock output c1 |
| lms1\_txpll\_locked | out | std\_logic | TXPLL locked output |

Table 31: lms1\_rxpll virtual bus

| **Port name** | **Direction** | **Type** | **Description** |
| --- | --- | --- | --- |
| lms1\_rxpll\_inclk | in | std\_logic | RXPLL input clock |
| lms1\_rxpll\_reconfig\_clk | in | std\_logic | RXPLL reconfiguration clock |
| lms1\_rxpll\_logic\_reset\_n | in | std\_logic | RXPLL logic active low reset |
| lms1\_rxpll\_clk\_ena | in | std\_logic\_vector(1 downto 0) | RXPLL clock enable |
| lms1\_rxpll\_drct\_clk\_en | in | std\_logic\_vector(1 downto 0) | RXPLL direct clock enable (Obsolete) |
| lms1\_rxpll\_c0 | out | std\_logic | RXPLL clock output c0 |
| lms1\_rxpll\_c1 | out | std\_logic | RXPLL clock output c1 |
| lms1\_rxpll\_locked | out | std\_logics | RXPLL locked output |

### Receive and transmit interface – inst3\_*rxtx\_top*

Main function of *rxtx\_top* module is to provide timestamp synchronization between receive and transmit packets. Module *tx\_path\_top* receives packets trough *tx\_in\_pct* bus, decodes them and writes decoded IQ samples to external FIFO buffer trough *tx\_smpl\_fifo* bus when decoded sample number is equal to number received from *rx\_path\_top* module.

Module *rx\_path\_top* receives IQ samples trough *rx\_smpl\_fifo* bus, packs them into packets together with sample number counter (64bit wide) and forwards packets to *rx\_pct\_fifo* bus.

Figure 4: rxtx\_top block diagram

Generics of *rxtx\_top* module can be found in Table 32 and port descriptions in Table 33.

Table 32: rxtx\_top generic parameters

| **Generic name** | **Type** | **Value** | **Description** |
| --- | --- | --- | --- |
| index | integer | 1 | Module index, if more than one module exists in design |
| DEV\_FAMILY | string | "" | Device family |
| TX\_EN | boolean | true | TX path enable |
| RX\_EN | boolean | true | RX path enable |
| TX\_IQ\_WIDTH | integer | 12 | TX DIQ sample width |
| TX\_N\_BUFF | integer | 4 | 2,4 valid values |
| TX\_IN\_PCT\_SIZE | integer | 4096 | TX packet size in bytes |
| TX\_IN\_PCT\_HDR\_SIZE | integer | 16 | TX packet header size |
| TX\_IN\_PCT\_DATA\_W | integer | 128 | TX packet data width |
| TX\_IN\_PCT\_RDUSEDW\_W | integer | 11 | TX in packet read used word width |
| TX\_OUT\_PCT\_DATA\_W | integer | 64 | TX out packet data width |
| TX\_SMPL\_FIFO\_WRUSEDW\_W | integer | 9 | TX sample FIFO write used word size |
| TX\_HIGHSPEED\_BUS | boolean | false | Double TX sample FIFO size to |
| RX\_DATABUS\_WIDTH | integer | 64 | RX data width |
| RX\_IQ\_WIDTH | integer | 12 | RX IQ sample width |
| RX\_INVERT\_INPUT\_CLOCKS | string | "OFF" | RX input clock inversion |
| RX\_SMPL\_BUFF\_RDUSEDW\_W | integer | 11 | RX sample bus read used words width in bits |
| RX\_PCT\_BUFF\_WRUSEDW\_W | integer | 12 | RX packet buffer write used words width in bits |
| RX\_DISABLE\_14B\_SAMPLEPACKING | boolean | false | Disable RX sample packing in 14b words |

Table 33: rxtx\_top module ports

| **Port name** | **Direction** | **Type** | **Description** |
| --- | --- | --- | --- |
| sys\_clk | in | std\_logic | System clock, free running |
| from\_fpgacfg | in | t\_FROM\_FPGACFG | Signals from FPGACFG registers |
| to\_fpgacfg | out | t\_TO\_FPGACFG | Signals to FPGACFG register |
| tx\_clk | in | std\_logic | TX path clock |
| tx\_clk\_reset\_n | in | std\_logic | TX path active low reset |
| tx\_pct\_loss\_flg | out | std\_logic | TX packet loss indication flag |
| rx\_clk | in | std\_logic | RX path clock |
| rx\_clk\_reset\_n | in | std\_logic | RX path active low reset |
| rx\_smpl\_nr\_cnt\_en | in | std\_logic | RX path sample number count enable |
| to\_memcfg | out | t\_TO\_MEMCFG | Signals to MEMCFG registers |
| from\_memcfg | in | t\_FROM\_MEMCFG | Signals from MEMCFG registers |
| ext\_rx\_en | in | std\_logic | External RX path enable |
| tx\_dma\_en | in | std\_logic | TX DMA enble flag |
| tx\_smpl\_fifo | out | Virtual bus |  |
| tx\_in\_pct | in | Virtual bus |  |
| rx\_smpl\_fifo | in | Virtual bus |  |
| rx\_pct\_fifo | out | Virtual bus |  |

Table 34: tx\_smpl\_fifo virtual bus

| **Port name** | **Direction** | **Type** | **Description** |
| --- | --- | --- | --- |
| tx\_smpl\_fifo\_wrreq | out | std\_logic | Write request (data valid) |
| tx\_smpl\_fifo\_wrfull | in | std\_logic | Write full |
| tx\_smpl\_fifo\_wrusedw | in | std\_logic\_vector(TX\_SMPL\_FIFO\_WRUSEDW\_W-1 downto 0) | Write used words |
| tx\_smpl\_fifo\_data | out | std\_logic\_vector(127 downto 0) | Write data |

Table 35: tx\_in\_pct virtual bus

| **Port name** | **Direction** | **Type** | **Description** |
| --- | --- | --- | --- |
| tx\_in\_pct\_reset\_n\_req | out | std\_logic | Reset request |
| tx\_in\_pct\_rdreq | out | std\_logic | Read request |
| tx\_in\_pct\_data | in | std\_logic\_vector(TX\_IN\_PCT\_DATA\_W-1 downto 0) | Packet data |
| tx\_in\_pct\_rdempty | in | std\_logic | Read empty |
| tx\_in\_pct\_rdusedw | in | std\_logic\_vector(TX\_IN\_PCT\_RDUSEDW\_W-1 downto 0) | Read used words |

Table 36: rx\_smpl\_fifo virtual bus

| **Port name** | **Direction** | **Type** | **Description** |
| --- | --- | --- | --- |
| rx\_smpl\_fifo\_wrreq | in | std\_logic | Write request |
| rx\_smpl\_fifo\_data | in | std\_logic\_vector(RX\_IQ\_WIDTH\*4-1 downto 0) | Data |
| rx\_smpl\_fifo\_wrfull | out | std\_logic | Write full |

Table 37: rx\_pct\_fifo virtual bus

| **Port name** | **Direction** | **Type** | **Description** |
| --- | --- | --- | --- |
| rx\_pct\_fifo\_aclrn\_req | out | std\_logic | Asynchronous clear request |
| rx\_pct\_fifo\_wusedw | in | std\_logic\_vector(RX\_PCT\_BUFF\_WRUSEDW\_W-1 downto 0) | Write used words |
| rx\_pct\_fifo\_wrreq | out | std\_logic | Write request (data valid) |
| rx\_pct\_fifo\_wdata | out | std\_logic\_vector(RX\_DATABUS\_WIDTH-1 downto 0) | Data |

#### Receive interface – *rx\_path\_top*

Once *rx\_path\_top* Figure 5 is enabled it starts continuously packing IQ samples into 4kB packets. For packet structure see [Stream protocol](https://github.com/myriadrf/LimeSuite/blob/master/docs/StreamProtocol.pdf) document.

Packets are written to 16kB FIFO buffer to maintain continuous data flow in short periods when host cannot accept data. If host halts data transfer for longer time period and four packets are buffered into 16kB buffer, FIFO full condition arises and other packets are dropped. When host starts to receive data after FIFO full condition, host should expect to receive those four buffered packets.

Module *rx\_path\_top* provides two 64bit sample counters. One is for TX logic – *tx\_path\_top*. TX logic uses this counter to synchronize transmitted LMS\_DQ1 samples with received LMS\_DIQ2 samples. Other is used for LMS\_DIQ2 samples packing into 4kB packets.

When *rx\_path\_top* is enabled it starts to collect IQ samples from *rx\_smpl\_fifo* bus, collected samples are written to FIFO buffer and each write enables *smpl\_cnt:inst4* module to increase its counter value. This means that counter value increases in same continuous rate as IQ sample rate.

Module *smpl\_cnt:inst3* is used for LMS\_DIQ2 samples packing into 4kB packets. Module *data2packets* reads IQ samples in bursts from FIFO buffer, each read enables *smpl\_cnt:inst3* module to increase its counter value. One read burst fills one 4kB packet and there are some idle cycles between bursts.

Figure 5: rx\_path\_top block diagram

*Table 38: rx\_path\_top instance description*

| **Instance** | **Description** |
| --- | --- |
| fifo\_inst | FIFO buffer for storing samples. |
| data2packets | Module for packing IQ samples to 4kB packets. |
| smpl\_cnt:inst3 | Sample counter for tx\_path\_top. |
| smpl\_cnt:inst4 | Sample counter for data2packets module. |

#### Transmit interface – t*x\_path\_top*

Transmit module *tx\_path\_top* reads IQ samples from EP03 FIFO buffer packed in 4kB packets. Packet header (see [Stream protocol](https://github.com/myriadrf/LimeSuite/blob/master/docs/StreamProtocol.pdf) document) contains sample number (or so-called time stamp) at which packet should be transmitted.

By using sample numbers from *rx\_path\_top* and received sample numbers in packet header transmitted IQ samples can be synchronized with received IQ samples.

Module *p2d\_wr\_fsm* separates packet header and payload. Packet payload is written into one of four 4kB FIFO buffers located in *packets2data* module and packet header is stored in *p2d\_rd* module. This module can work in two modes:

* **Synchronization enabled** - module compares received sample number from packet header and sample number from *rx\_path\_top*. When sample number from received packet is equal to sample number of *rx\_path\_top* module (this means that it is time to send TX packet), read process begins and IQ samples are transmitted to LMS\_DIQ1 interface. When sample number from received packet is greater than sample number of *rx\_path\_top* module (this means that received packet should be sent after some time) *p2d\_rd* waits until those sample numbers will be equal. When sample number from received packet is less than sample number of *rx\_path\_top* module (this means that packet arrived too late) corresponding FIFO buffer is cleared.
* ***Synchronization disabled*** *–* module does not compare sample numbers and every received packet is transmitted to LMS\_DIQ1 interface*.*

Block diagram can be found in **Figure 6** and instance description in **Table 39.**

Figure 6: tx\_path\_top block diagram

Table 39: tx\_path\_top instance description

| **Instance** | **Description** |
| --- | --- |
| packets2data\_top | Wrapper file |
| packets2data | Wrapper file |
| p2d\_wr\_fsm | Module reads packets from EP03 buffer and places to one of the 4kB FIFO buffers in increasing order and stores corresponding sample number from packet header. |
| p2d\_rd | Module checks one of the FIFO buffers if it is filled with samples in increasing order. When buffer is ready depending on received sample number from packet header and sample number from rx\_path\_top module buffer can be cleared or IQ sample reading begins. |
| fifo\_inst | FIFO buffer |
| sync\_fifo\_rw | Dual clock FIFO buffer for clock domain crossing. |
| bit\_unpack\_64 | Depending on mode selection samples are unpacked (see [Stream protocol](https://github.com/myriadrf/LimeSuite/blob/master/docs/StreamProtocol.pdf) document). |

### *LMS7002 digital interface – inst4\_lms7002\_top*

Module lms7002\_top implements various digital LimeLigth interface modes that LMS7002 IC can operate. Supported modes:

* TRXIQ MIMO DDR
* TRXIQ SISO DDR
* TRXIQ Pulse
* TRXIQ SDR

Instance *tx\_path\_top* is used for transmitting samples trough LMS\_DIQ1 port and *diq2fifo* instance for receiving samples from LMS\_DIQ2 port.

Figure 7: lms7002\_top block diagram

Generic parameter description can be found in Table 40 and port description in Table 41.

Table 40: lms7002\_top generic parameters

| **Generic name** | **Type** | **Value** | **Description** |
| --- | --- | --- | --- |
| g\_DEV\_FAMILY | string | "" | Device family |
| g\_IQ\_WIDTH | integer | 12 | IQ bus width |
| g\_INV\_INPUT\_CLK | string | "ON" | Input clock inversion |
| g\_TX\_SMPL\_FIFO\_0\_WRUSEDW | integer | 9 | TX sample FIFO 0 write used words width |
| g\_TX\_SMPL\_FIFO\_0\_DATAW | integer | 128 | TX sample FIFO 0 data width |
| g\_TX\_SMPL\_FIFO\_1\_WRUSEDW | integer | 9 | TX sample FIFO 1 write used words width (Not in use) |
| g\_TX\_SMPL\_FIFO\_1\_DATAW | integer | 128 | TX sample FIFO 1 write used words width (Not in use) |

Table 41: lms7002\_top port description

| **Port name** | **Direction** | **Type** | **Description** |
| --- | --- | --- | --- |
| from\_fpgacfg | in | t\_FROM\_FPGACFG | Signals from FPGACFG registers |
| from\_tstcfg | in | t\_FROM\_TSTCFG | Signals from TSTCFG registers |
| from\_memcfg | in | t\_FROM\_MEMCFG | Signals from MEMCFG registers |
| mem\_reset\_n | in | std\_logic | Memory module reset |
| tx\_ant\_en | out | std\_logic | TX antenna enable flag |
| rx\_reset\_n | in | std\_logic | RX interface active low reset |
| rx\_diq\_h | out | std\_logic\_vector(g\_IQ\_WIDTH downto 0) | Output of Direct capture on rising edge of DIQ2 port |
| rx\_diq\_l | out | std\_logic\_vector(g\_IQ\_WIDTH downto 0) | Output of Direct capture on falling edge of DIQ2 port |
| rx\_data\_valid | out | std\_logic | Received data from DIQ2 port valid signal |
| rx\_data | out | std\_logic\_vector(g\_IQ\_WIDTH\*4-1 downto 0) | Received data from DIQ2 port |
| rx\_smpl\_cmp\_start | in | std\_logic | RX sample compare start |
| rx\_smpl\_cmp\_length | in | std\_logic\_vector(15 downto 0) | RX sample compare length |
| rx\_smpl\_cmp\_done | out | std\_logic | RX sample compare done flag |
| rx\_smpl\_cmp\_err | out | std\_logic | RX sample compare error flag |
| rx\_smpl\_cnt\_en | out | std\_logic | RX sample counter enable |
| LMS\_PORT1 | out | Virtual bus | interface |
| LMS\_PORT2 | in | Virtual bus | interface |
| LMS\_MISC | out | Virtual bus | LMS miscellaneous control ports |
| tx\_fifo0 | in | Virtual bus | Internal TX ports |
| tx\_fifo1 | in | Virtual bus | (not in use) |

Table 42: LMS\_PORT1 virtual port

| **Port name** | **Direction** | **Type** | **Description** |
| --- | --- | --- | --- |
| MCLK1 | in | std\_logic | TX interface clock |
| MCLK1\_2x | in | std\_logic | (Not in use) |
| FCLK1 | out | std\_logic | TX interface feedback clock |
| DIQ1 | out | std\_logic\_vector(g\_IQ\_WIDTH-1 downto 0) | DIQ1 data bus |
| ENABLE\_IQSEL1 | out | std\_logic | IQ select flag for DIQ1 data |
| TXNRX1 | out | std\_logic | LMS\_PORT1 direction select |

Table 43: LMS\_PORT2 virtual port

| **Port name** | **Direction** | **Type** | **Description** |
| --- | --- | --- | --- |
| MCLK2 | in | std\_logic | RX interface clock |
| FCLK2 | out | std\_logic | RX interface feedback clock |
| DIQ2 | in | std\_logic\_vector(g\_IQ\_WIDTH-1 downto 0) | DIQ2 data bus |
| ENABLE\_IQSEL2 | in | std\_logic | IQ select flag for DIQ2 data |
| TXNRX2 | out | std\_logic | LMS\_PORT2 direction select |

Table 44: LMS\_MISC virtual bus

| **Port name** | **Direction** | **Type** | **Description** |
| --- | --- | --- | --- |
| RESET | out | std\_logic | LMS hardware reset, active low |
| TXEN | out | std\_logic | TX hard power off |
| RXEN | out | std\_logic | RX hard power off |
| CORE\_LDO\_EN | out | std\_logic | LMS internal LDO enable control |

Table 45: tx\_fifo\_0 virtual bus

| **Port name** | **Direction** | **Type** | **Description** |
| --- | --- | --- | --- |
| tx\_reset\_n | in | std\_logic | TX interface active low reset |
| tx\_fifo\_0\_wrclk | in | std\_logic | TX FIFO write clock |
| tx\_fifo\_0\_reset\_n | in | std\_logic | TX FIFO Reset |
| tx\_fifo\_0\_wrreq | in | std\_logic | TX FIFO write request |
| tx\_fifo\_0\_data | in | std\_logic\_vector(g\_TX\_SMPL\_FIFO\_0\_DATAW-1 downto 0) | TX FIFO data |
| tx\_fifo\_0\_wrfull | out | std\_logic | TX FIFO write full |
| tx\_fifo\_0\_wrusedw | out | std\_logic\_vector(g\_TX\_SMPL\_FIFO\_0\_WRUSEDW-1 downto 0) | TX FIFO write used words |

### *RF control logic – inst5\_tdd\_control*

Module tdd\_control implements some simple logic to control RF switches automatically when TDD operation is required.

If automatic control is disabled, RF switches are set to the mode that is specified in register 0x000A (refer to Table 19 FPGACFG module registers (0x000A-0x0010)). If automatic control is enabled, RX switches are set to “No connection” when data is being transmitted from FPGA to LMS and to the value specified in 0x000A register when no data is being transmitted. TX switches are set to the value in 0x000A register when data is transmitted and to the inverse value when it is not.

This module contains only logic and no lower modules, so no block diagram is provides. Port descriptions are in Table 46.

Table 46: tdd\_control port description

| **Port name** | **Direction** | **Type** | **Description** |
| --- | --- | --- | --- |
| MANUAL\_VALUE | in | std\_logic | Value to be used for TDD\_OUT if AUTO\_ENABLE = '0' |
| AUTO\_ENABLE | in | std\_logic | Enable automatic TDD\_OUT control |
| AUTO\_IN | in | std\_logic | RF status. '1': transmitting, '0': not transmitting |
| AUTO\_INVERT | in | std\_logic | Invert TDD\_OUT signal |
| RX\_RF\_SW\_IN | in | std\_logic\_vector(1 downto 0) | RF RX switch configuration to be used when receiving |
| TX\_RF\_SW\_IN | in | std\_logic | RF TX switch configuration to be used when transmitting |
| RF\_SW\_AUTO\_ENANBLE | in | std\_logic | Enable automatic TDD RF RX/TX switch control |
| TDD\_OUT | out | std\_logic | Output signal for external TDD modules |
| RX\_RF\_SW\_OUT | out | std\_logic\_vector(1 downto 0) | RF RX switch control output |
| TX\_RF\_SW\_OUT | out | std\_logic | RF TX switch control output |

### Test modules – inst6\_tst\_top

Module tst\_top implements basic test modules for clocks FPGA\_CLK (refered to as lms\_tx\_clk in code), SYS\_CLK, as well as for the GNSS module.

Clock tests work in the following way:

SYS\_CLK – tested with no reference clock, the test can only answer if the clock is active or not.

FPGA\_CLK – tested by counting the number of cycles in 0.1 seconds by using SYS\_CLK as a reference.

GNSS is tested by sending a test command and waiting for an appropriate answer.



Figure 8: tst\_top block diagram

Table 47: tst\_top port description

| **Port name** | **Direction** | **Type** | **Description** |
| --- | --- | --- | --- |
| SYS\_CLK | in | std\_logic | System clock |
| RESET\_N | in | std\_logic | Reset, active low |
| LMS\_TX\_CLK | in | std\_logic | LMS7002 tx clock |
| GNSS\_UART\_RX | in | std\_logic | GNSS chip's uart rx signal |
| GNSS\_UART\_TX | out | std\_logic | GNSS chip's uart rx signal |
| TO\_TSTCFG | out | t\_TO\_TSTCFG | Configuration register bus Module -> Registers |
| FROM\_TSTCFG | in | t\_FROM\_TSTCFG | Configuration register bus Registers -> Module |

## Clock network

Figure 6 shows dataflow between main modules and clocking scheme. More details about design clocks can be found in Table 48.

Figure 9: Clock network diagram

Table 48: Design clock details

|  |  |  |
| --- | --- | --- |
| **Clock name** | **Frequency, MHz** | **Description** |
| LMS\_MCLK1 | Configurable | Reference clock for LMS\_DIQ1 data bus |
| LMS\_MCLK2 | Configurable | Reference clock for LMS\_DIQ2 data bus. |
| LMS\_FCLK1 | Configurable | Feedback clock LMS7002 IC captures LMS\_DIQ1 data using this clock |
| LMS\_FCLK2 | Configurable | Not used. |
| FPGA\_CLK | 26 | Onboard XO clock |
| PCIE\_REFCLK | 100 | Reference clock from PCIe host. |
| sys\_clk | 125 | PCIe system clock synchronous to PCIE\_REFCLK |