Verification Continuum™

ZeBu® ZX XTOR Library

Release Notes

Version Q-2021.06-SP2, October 2022

SYNOPSYS®

Copyright Notice and Proprietary Information

Destination Control Statement

All technical data contained in this publication is subject to the export control laws of the United States of America. Disclosure to nationals of other countries contrary to United States law is prohibited. It is the reader's responsibility to determine the applicable regulations and to comply with them.

Disclaimer

SYNOPSYS, INC., AND ITS LICENSORS MAKE NO WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, WITH REGARD TO THIS MATERIAL, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE.

Trademarks

Synopsys and certain Synopsys product names are trademarks of Synopsys, as set forth at http://www.synopsys.com/company/legal/trademarks-brands.html.
All other product or company names may be trademarks of their respective owners.

Free and Open-Source Software Licensing Notices

If applicable, Free and Open-Source Software (FOSS) licensing notices are available in the product installation.

Third-Party Links

Any links to third-party websites included in this document are for your convenience only. Synopsys does not endorse and is not responsible for such websites and their practices, including privacy practices, availability, and content.

www.synopsys.com

Contents

1 Preface

This document lists the enhancements and bug fixes in the Q-2021.06-SP2 release for the ZeBu Transactor (ZX XTOR) Library components.

This document is intended for users who are familiar with the ZeBu product range.

1.1 Related Documentation

Document Name	Description
ZeBu Server Release Notes	Provides information about the ZeBu Server features and limitations. Available in the ZeBu Server documentation package corresponding to your software version.
ZeBu Transactor Documentation	Provides relevant information about the usage of the respective transactor. Available in the respective transactor package.

1.2 Terms and Abbreviations

Abbreviations	Description
ACE Lite	AXI Coherency Extensions Lite
ACE Master	AXI Coherency Extensions Master
AMBA	Advanced Microcontroller Bus Architecture
AHB	Advanced High-performance Bus
APB	Advanced Peripheral Bus
AXI	Advanced Extensible Interface

Abbreviations	Description
CAN_FD	Controller Area Network with Flexible Data Rate
CSI	Camera Serial Interface
DDR	Double Data Rate
DDR DIMM	Double Data Rate Dual Inline Memory Modules
DDR4	Double Data Rate 4
DSI	Display Serial Interface
DP Sink	Display Port Sink Interface
DP Source	Display Port Source Interface
EtherAVB	Audio Video Bridging and Time Sensitive Networking
ENET	Ethernet
GPIO	General Purpose Input-Output
HDMI	High-Definition Multimedia Interface
I2C	Inter–Integrated Circuit Bus
I2S	Inter-Integrated Chip (IC) Sound, Integrated Interchip Sound
JTAG TAP	Joint Test Action Group Test Access Port
SATA	Serial Advanced Technology Attachment

2 Overview

ZeBu transactors are a family of over 30 protocol specific transaction-based verification solutions that allow verification/design engineers to quickly build a complete system-level test environment for their SoC design to be emulated on the ZeBu system. These transactors include synthesizable bus functional models that are loaded into ZeBu reconfigurable testbench (RTB) hardware, providing maximum performance and ensuring that the transactor is always synchronized to the emulated design.

Each transactor also includes C/C++ APIs to quickly create test benches and drivers to generate real-world traffic, and to link to virtual platforms. Off-the-shelf ZeBu transactors are complemented by the ZEMI-3 transactor compiler, which enables you to create your own ZeBu compatible custom synthesizable transactors quickly.

ZeBu transactors support common protocols and standards specifications, such as, PCI Express, USB, MIPI CSI-2, JTAG, Gigabit Ethernet, and so on.

This section lists the transactors released in the ZeBu Transactor (ZX XTOR) Library package for the Q-2021.06-SP1 release.

2.1 Transactor Library Components

The Q-2021.06-SP2 package for the ZX XTOR Library contains the following components:

ZeBu CAN_FD Transactor	ZeBu DP Source Transactor	ZeBu ENET Transactor	ZeBu EtherAVB Transactor
ZeBu GPIO Transactor	ZeBu HDMI2 Source Transactor	ZeBu I2C Transactor	ZeBu KMI Transactor
ZeBu AHB Monitor	ZeBu AMBA Monitor	ZeBu APB Monitor	ZeBu CHI Monitor
ZeBu MIPI I3C Monitor	ZeBu UFS Monitor	ZeBu SATA Device Transactor	ZeBu SATA Host Transactor
ZeBu SPI Master Transactor	ZeBu SPI Slave Transactor	ZeBu Stream Transactor	ZeBu Touchscreen Transactor
ZeBu VideoIn Transactor	ZeBu Virutal USB Bridge	ZeBu Virtual USB Device	ZeBu Virtual USB3 Device

ZeBu Virtual External Display Transactor	ZeBu AHB Master Transactor	ZeBu AHB Slave Transactor	ZeBu AMBA Master Transactor
ZeBu AMBA Slave Transactor	ZeBu APB Master Transactor	ZeBu APB Slave Transactor	ZeBu AVSBus Transactor
ZeBu AXI4 Master Stream Transactor	ZeBu AXI4 Slave Stream Transactor	ZeBu CHIRN Transactor	ZeBu CHISN Transactor
ZeBu DP Sink Transactor	ZeBu DS5 Transactor	ZeBu FlexE Transactor	ZeBu Flash Transactor
ZeBu HDMI2 Sink Transactor	ZeBu I2S Transactor	ZeBu Interlaken Transactor	ZeBu JTAG SWD Transactor
ZeBu JTAG T32 Transactor	ZeBu JTAG TAP Transactor	ZeBU MIPI CSI Transactor	ZeBu MIPI DSI Device Transactor
ZeBu MIPI DSI Host Transactor	ZeBu MIPI I3C Transactor	ZeBu MMC Device Transactor	ZeBu MMC Host Transactor
ZeBu OCP Master Transactor	ZeBu OCP Slave Transactor	ZeBu ONFI Transactor	ZeBu OTU4 Transactor
ZeBu PCIe Transactor	ZeBu PCIe Compliance Test Suite	ZeBu SDIO Device Transactor	ZeBu SDIO Host Transactor
ZeBu SRAMSW Transactor	ZeBu UART Transactor	ZeBu UFS Device Transactor	ZeBu UFS Host Transactor
ZeBu USB Transactor	ZeBu USB2 Transactor	ZeBu VbyOne Transactor	ZeBu Video Transactor
ZeBu ZDDR2 Transactor	ZeBu ZDDR3 Transactor	ZeBu ZDDR4 Transactor	ZeBu ZLPDDR3 Transactor
ZeBu ZLPDDR4 Transactor	ZeBu ZWIDEIO2 Transactor		

The user guides for these transactors are included in the respective transactor .sh package.

2.2 Downloading the Transactor Software

To download the transactor software, perform the following steps:

1. Point your web browser to *http://solvnetplus.synopsys.com*.
2. Enter your Synopsys SolvNet Username and Password.
3. Click **Sign In**.
4. Click **Downloads > ZX XTOR** Library.
5. Click the version number for which you want to download the software.
6. Click **Download Here > Yes, I Agree to the Above Terms**.
   A new web browser opens.
7. Click **Download** for the specific transactor software.

3 Compatibility

This section provides information on the enhancements made to the installation-related tasks and features:

- *Common Infrastructure*
- *ZeBu Server Compatibility*
- *Licensing*
- *Using Xilinx Primitives*

3.1 Common Infrastructure

You must install the Q-2021.06-SP2 packages separately. Do not install these packages over any previous release of the transactor since it may not work.

3.2 ZeBu Server Compatibility

The transactors in the Q-2021.06-SP2 release are compatible with ZeBu Server Q-2020.03-SP1-3 release.

3.3 Licensing

In the 2021.06-SP2 release, the license strings for all titles has been updated to use hardware-agnostic license strings. Only individual title strings are supported and the library license string is no longer supported.

3.4 Using Xilinx Primitives

If you want to use Xilinx primitives, include the path of these primitives in the example transactor, as shown below:

```
-y $ZeBu_XIL/ISE/verilog/src/unisims
```

4 Key Enhancements in Q-2021.06-SP2 Release

This section contains the following topics:

4.1 New Features

No new features have been introduced in this release.

4.2 Enhancements

This section describes the enhancements made to the following transactors in the Q-2021.06-SP2 release:

- *CXL*
- *EtherAVB*
- *xtor_enet_svs*
- *xtor_mipi_dsi_host*
- *xtor_otu4_svs*
- *PCIe*
- *xtor_mipi_i3c*
- *UART*
- *USB*

4.2.1 CXL

For CXL 16x40b scenario, specify the following commands in UTF file.

```
ztopbuild -advanced_command {partitioning auto -slp }
ztopbuild -advanced_command {zcorebuild_command * {partitioning
auto}}
ztopbuild -advanced_command "enable split_xtor_orion"
```

4.2.2 EtherAVB

The following enhancements have been made for security protocols:

- Support for MACSec with GCM-AES-128
- IPSec transport mode frame

- IPSec ESP and AH authentication with HMAC-SHA-256-96
- Extended Sequence Number (ESN) support
- Include initialization vector as part of ESP payload

Note *Decoding APIs for Layer3/4 class are modified to support the above new protocols and that they are not backward compatible. Please check API guide and the examples for the use model updates.*

4.2.3 xtor_enet_svs

The following enhancements have been made:

- To support the ts_width value of 6 and 8, an additional argument has been added to the getFrameTimeStamp(uint8_t ts_width) API. The default value of ts_width argument is 8.
- All non-supported dashboard parameters have been removed from struct dashboardStruct– txGbps/txGbpsAbs/txGbpsMin/txGbpsMax/rxGbps/rxGbpsAbs/ rxGbpsMin/rxGbpsMax/rxTotalPauseTimeQueue0/rxTotalPauseTimeQueue1, and so on. See $ZEBU_IP_ROOT/include/xtor_enet_svs.hh file for more information.

Note *Note that the use of the non-supported dashboard parameters might result in backward compatibility issues. Update your testbench to fix these issues.*

4.2.4 xtor_mipi_dsi_host

Previously, xtor_mipi_dsi_host was inserting SSS bytes, not in compliance with protocol. This was resulting in RxValid/RxData signal misalignment.

To fix the issue, the laneModel file (xtor_mipi_dsi_phy_lm_svs.v) from xtor_mipi_dsi_device_svs package has been updated to align RxValid and RxData signals in the xtor_mipi_dsi_host

4.2.5 xtor_otu4_svs

Added support for the following new features:

- DataIntegrity lock feature to identify transactor generated frames.

- Stop traffic API to switch between PRBS and 100GBASE-R frames of Tx.
- Added support for the following Alarm Insertion methods:
  - SM - BDI Alarm
  - SM - BIAE Alarm
  - SM - IAE Alarm
  - SM - BEI Error
- Clear all status field.
- Continuous container/frame generation.
- Dynamic Error Injection support, which can result in DLOS.
- Dynamic Configuration Support for SM – TTI (SAPI,DAPI, operator specific).
- Frame latency measurement.

Check xtor_otu4_svs.hh file present in include dir. for more information.

4.2.6 PCIe

The following enhancements have been made to the PCIe transactor:

4.2.6.1 Change in API Prototype to Initiate Post-Processing

The struct xtor_pcie_mon_pktprntCfg_flag_t has been added to the doPciePostProcessing API, as shown below.

```
doPciePostProcessing (const char* post_process_txt_file ,FILE*
mon_file, uint8_t LANES_PA , uint8_t PIPE_WIDTH_PA,bool
is_zemi3,svt_c_runtime_cfg
*runtime,xtor_pcie_mon_pktprntCfg_flag_t pkt_flag)
```

This struct customizes data log generated. The following table lists the possible flag values available for the struct:

TABLE 1 xtor_pcie_mon_pktprntCfg_flag_t Values

FLAG	Default value	Description
PCIE_VS_MONITOR_PA_OS_FSDB_DIS	true	when true prints order sets
PCIE_VS_MONITOR_PA_TLP_FSDB_DIS	true	when true prints TLP
PCIE_VS_MONITOR_PA_DLLP_FSDB_DIS	true	when true prints DLLP
PCIE_VS_MONITOR_PA_FSDB_EN	true	when true enable FSDB for PA
PCIE_VS_MONITOR_PA_PKT_LOG_EN	true	when true print traffic field wise with high abstraction

Note *For Backward compatibility The old version api prototype has been kept intact.*

4.2.6.2 Capturing Traffic in Different Frame Of Interval

You can now invoke Start -Stop monitor API multiple time. It allows to capture traffic at user-specifies runtime duration. For each start-stop, separate binary dump file gets generated and you need to post-process them individually to observe traffic in post process log.

```
analyzerStart();   // 1st
           .
       <Traffic exchange>
          .
         analyzerStop();
             .
              .
        analyzerStart();  // 2nd
           .
       <Traffic Exchange>
```

```
            .
        analyzerStop();
```

4.2.7 xtor_mipi_i3c

4.2.7.1 Enhancements for the MIPI I3C Master Transactor

For I3C master, IBI and Payload with PEC features have been added:

To enable these features, use the following commands:

```
uint32_t   registerDevice( bool isI2CDevice, uint32_t
DynamicAddress, uint32_t StaticAddress ,
bool IBI_With_Data = false , bool IBI_PEC_Enable = false )
```

Where:

- bool IBI_With_Data is a new optional parameter to check whether the degistered Device is IBI .Paylaod compatible (BCR bit 2 at one).
- bool IBI_PEC_Enable checks if the registered device sends the IBI Payload followed by a PEC.

As Master is now compatible with the IBI Payload, the SETMRL command now sends the payload side after the CCC SETMRL command, as shown below:

```
void  sendCCC_D_SETMRL  (uint32_t DeviceNumber, uint32_t MRL,
uint8_t IBI_Payload_Size = 1 )
```

The following figure illustrates the Direct SETMRL/GETMRL format and Broadcast SETMRL format, respectively:

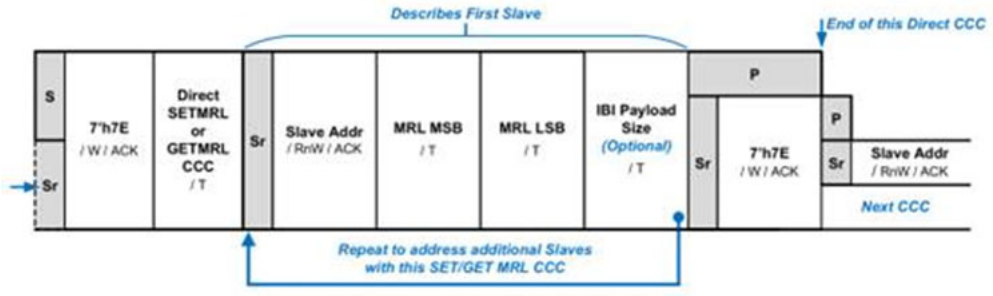


FIGURE 1. Direct SETMRL/GETMRL Format

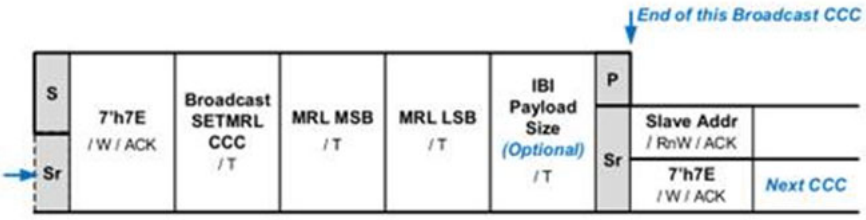


FIGURE 2. Broadcast SETMRL Format

Extracting the IBI Data

To extract the IBI data, call getDataIBI method inside the SIR CallBack, as shown below:

```
void user_SIR_CB (void* ptr1, uint32_t *DeviceAddress, bool *RW){
  xtor_mipi_i3c_svs * I3CMaster = (xtor_mipi_i3c_svs*) ptr1;
  cerr << "#TB MASTER IBI-SIR DETECTED. ADDR=" << *DeviceAddress <<
", RW=" << *RW << endl;

  uint8_t * DataRead_TMP  = new uint8_t [65536];
  uint32_t nbNdata ;
  // XTOR_MIPI_I3C_SVS::I3C_status_t _I3C_status ;
  // _I3C_status = I3CMaster->getStatus() ;
  I3CMaster->getDataIBI  (DataRead_TMP , &nbNdata);
```

```
  uint64_t ReturnData = 0 ;
  for (uint32_t i = 0 ; i < nbNdata ; i ++) {
    ReturnData += (uint64_t)((uint64_t)DataRead_TMP[i] <<
((uint64_t)(nbNdata-1)*8 - i*8)) ; //-- BIG Indian
  }
  cerr << "#TB : IBI Payload  = 0x"<< hex << ReturnData  << dec <<
endl;
  delete [] DataRead_TMP;


  _mst_ibi_detected = true ;
}
```

4.2.7.2 Enhancements for the MIPI I3C Slave Transactor

For MIPI I3C Slave, IBI with payload is added. The Slave transactor can now generate an IBI With Data, as shown below:

```
void       I3Cslave_SIR_IBI_req (uint8_t MDB ,  uint32_t nb_data ,
uint8_t* DataWrite)
```

PEC feature is not supported by slave transactor

Note *The enhancements done for the xtor_mipi_i3c transactor may cause backward compatibility issues. Please update your testbench accordingly to avoid these issues.*

4.2.8 UART

The following enhancements have been made:

- Added support for XTOR_UART_SVS::enXtermEcho(). This enables you to view the data from transactor to DUT on Xterm terminal. Previously, data from DUT to transactor side was visible on Xterm terminal.
- You can now enable/disable prints for traffic from XTOR to DUT on Xterm terminal, in the zRci Xterm testbench, using the EN_ECHO=1/EN_ECHO=0 parameters.

- Introduced the XTOR_UART_SVS::setApproxThreshold(int n) method. This method sets the approximate threshold value when ratio changes from lower to higher value.

  If the Baud rate value changed from lower to higher value, (then depending upon Threshold value) hardware side takes some clock cycles to update Ratio in API getDetectedRatio. Therefore, some packets may be corrupted during that time. The config() method must be called to make the setting effective in the hardware. By default, value is 0xFFFFFF which is maximum value.
- Updated description for the XTOR_UART_SVS::IsConnected() method in the header file to:

  It has to be called in a loop to handle socket communication, so that it can accept the subsequent connections.

4.2.9 USB

The following enhancements have been made:

- Dynamic speed change support: To use this feature, disconnect, reconfigure the speed and reconnect.
- Auto-detection of device disconnect in Host transactor: This is only available in non-blocking configuration. The call to USBUnplug() method from the Host transactor is not required.
- DRD role switch during runtime: ZEBU_IP_ROOT example is updated to demonstrate the same.

5 Key Enhancements in Q-2021.06-SP1 Release

This section contains the following topics:

5.1 New Features

No new features have been introduced in this release.

5.2 Enhancements

The following enhancements have been made to the following titles in the Q-2021.06-SP1 release.

- *AMBA Monitor*
- *SPI*

5.2.1 AMBA Monitor

AMBA Monitor (monitor_amba_svs) now supports read/write outstanding transactions metric in Platform Architect Studio (PAS) mode.

For more information, see ZeBu AMBA Monitor User Guide.

5.2.2 SPI

SPI Slave (xtor_spi_slave_svs) now supports EEPROM transactions in standard mode. For more information, see ZeBu SPI Transactor User Manual

6 Key Enhancements in Q-2021.06-1 Release

No new features or enhancements have been introduced in the Q-2021.06-1 release.

7 Key Enhancements in Q-2021.06 Release

This section contains the following topics:

7.1 New Titles

The following new DPI titles have been introduced in the Q-2021.06 release of ZeBu Transactors :

- ZeBu OCP Monitor
- ZeBu PCIe Monitor
- ZeBu CXL Monitor
- ZeBu CXL Transactor
- ZeBu CXL CTS Transactor
- ZeBu DDR4 Transactor
- ZeBu DDR4 DIMM Transactor
- ZeBu DDR5 Transactor
- ZeBu ENET Transactor
- ZeBu VideoIN Transactor
- ZeBu KMI Transactor
- ZeBu LPDDR5 Transactor
- ZeBu LIN Transactor
- ZeBu Multi-Transactor/Memory Model Examples

7.2 Deprecated Titles

The following Zcei titles have been deprecated in the Q-2021.06 release:

- ZeBu KMI Transactor
- ZeBu Touchscreen Transactor
- ZeBu VideoIn Transactor
- ZeBu ZDDR4 Transactor

7.3 New Features

Enhancements have been done in the following ZeBu Transactors in the Q-2021.06 release:

- *AMBA Monitor*
- *ENET*
- *PCIe*
- *SDIO Device Transactor*
- *UFS Device*

7.3.1 AMBA Monitor

The constructor of the `monitor_amba_svs` API class has changed as follows:

```
static monitor_amba_svs* getInstance (int instance, char* scope,
PROTOCOL_MODE protocol_mode=AXI3, DUMP_MODE dump_mode=FSDB_LOG,
svt_c_runtime_cfg* runtime = NULL);
```

In addition, other enhancements have been made. For details, see *ZeBu AMBA AXI Monitor User Guide* on SolvNetPlus.

7.3.2 CXL

The support for the following new feature has been added in the CXL transactor:

- Deferrable Write: CXL.io deferrable write can be initiated by using the following APIs:
  - np_writeMem32()
  - np_writeMem64()

For details, see *ZeBu CXL Transactor User Guide*.

7.3.3 ENET

ENET Transactor (xtor_enet_svs) supports the following features in contrast to the ENET (Zcei) transactor:

- Manages variable Ethernet frame length, from 64 Bytes to 1,518 Bytes. Does not support Jumbo frame, in contrast to ENET (Zcei) transactor
- Flow control (PAUSE) using software APIs.

Note *For MII interface, it is recommended to use ENET transactor (xtor_enet_svs) transactor as ENET (Zcei) will be deprecated in a future release.*

For details, see *ZeBu ENET Transactor User Guide*.

7.3.4 PCIe

The support for the following new features have been added to the PCIe Transactor:

- New Message types:
  - **INTX**: It's a message interrupt generated by EP device using message TLP with message code for INTA, INTB, INTC, or INTD. The interrupt can be generated through the interrupt pins on the hardware but transactor supports this feature through message API.
  - **MSI**: It's an interrupt generated by EP. It refers to an upstream mem write transactions but with address and data configured in the address and data fields of MSI capability register. Set the MSI enable field before issuing the MSI interrupt.

    To run the MSI feature, use `sendMSI` API. To generate multiple interrupts through MSI, use `PCIE_MSI_MULTIPLE_MSG_CAP`, `addparamconfig` parameters.
  - **MSIX**: Its an interrupt generated by EP. It is an upstream mem write transaction but with address and data configured in the MSIX dedicated BAR. The MSIX dedicated BAR is programmed in the MSIX capability register.

    To run the MSIX feature, use `sendMSIX` API. To program the number of MSIX messages and BAR offset, use PCIE_MSIX_TABLE_SIZE and `PCIE_MSIX_TABLE_OFFSET`, `addparamconfig` parameters.
- Alternate Mode (`TxDaValid` and `RxDataValid`):

- For Normal PIPE, specify the following parameters:

```
PIPE_G1    = 8 ;
PIPE_G2    = 8 ;
PIPE_G3    = 32 ;
PIPE_G4    = 32 ;
PIPE_G5    = 32 ;

FREQ_G1    = 10000 ;
FREQ_G2    = 10000 ;
FREQ_G3    = 10000 ;
FREQ_G4    = 10000 ;
FREQ_G5    = 10000 ;
```

- For SERDES Architecture, specify the following parameters:

```
PIPE_G1    = 1 ;
PIPE_G2    = 1 ;
PIPE_G3    = 4 ;
PIPE_G4    = 4 ;
PIPE_G5    = 4 ;

FREQ_G1    = 10000 ;
FREQ_G2    = 10000 ;
FREQ_G3    = 10000 ;
FREQ_G4    = 10000 ;
```

For details, see *ZeBu PCIe Transactor User Guide*.

7.3.5 SDIO Device Transactor

The following new features have been introduced for the SDIO Device Transactor:

- **Lock/Unlock**: The password protection feature enables the host to lock a card while providing a password, which later is used for unlocking the card.

- **Erase**: This feature helps customer to erase many write blocks simultaneously.
- **Command Queuing**: Command queue separates bus transaction in tasks assignment phase and data transfer phase. Therefore, multiple memory read/ write tasks can be assigned.
- **Function extension specification**: The extension functions/methods in the SD memory card provide suitable way for active control over the new functions. The following new APIs have been added to perform load/dump on extension register space:
  - load_hex_fn_extr: Initializes the function extension register with the content of a text file with hexadecimal data.
  - dump_hex_fn_extr: Reports a block of data from the function extension register content to a text file in a hexadecimal format.

For details, see *ZeBu SDIO Device Transactor User Manual*.

7.3.6 UFS Device

The following modules in UFS Device transactor are now subject to connectivity checks:

- xtor_ufs_device_svs/unipro_top_*.v
- xtor_ufs_device_svs/mphy_serial_model_top*.v
- xtor_ufs_device_svs/MPHY_*.v

This means that any port for these modules, if left unconnected, will report an error. Ensure that all the ports are connected to the respective signals.

For details, see *ZeBu UFS Device Transactor User Manual*.

7.4 Enhancements

Enhancements have been made to the following titles in the Q-2021.06 release.

- *CHI Monitor*
- *EtherAVB*
- *LIN*
- *PCIe*

7.4.1 CHI Monitor

Enabled support for CHI monitor protocol for CHI version B.

7.4.2 EtherAVB

Added support for the following:

- Variable clock frequency for MDIO interface
- MDIO preamble is now configurable to 1 bit or 32 bit

See *ZeBu EtherAVB Transactor User Guide* for details.

7.4.3 LIN

The nomenclature of the LIN transactor has been changed according to new DPI norms.

The title name has been changed from LIN to xtor_lin_svs.

The library names have been changed from `snps_lin` to `lin_svs` and `snps_lin_master` to `xtor_lin_master_svs` and `snps_lin_slave` to `xtor_lin_slave_svs`, respectively.

See *ZeBu LIN Transactor User Guide* for updates.

7.4.4 PCIe

The following enhancements have been made to the PCIe transactor:

- The following updates have been made to the lane model:
  - Added the `rxclk_to_dut` port.
  - Removed the `MON_PIPE` parameter.
- The monitor pointer construction has changed to:

```
pcie_monitor = new monitor_pcie_svs("monitor_pcie_svs_core",<hierarchy of
monitor>.xtor_pcie_svs_monitor,xsched,runtime)
```

- Changed the PCIe Monitor's module name from `xtor_pcie_svs_monitor` to `monitor_pcie_core`. The UTF file must be updated with the same information.

For details, see *ZeBu PCIe Transactor User Manual*.

8 Limitations

This section captures the limitations associated with the Vertical Solutions transactors in the following subtopics:

- *zRCI Flow*
- *Connectivity Checks Limitations*
- *Driver Detection*

8.1 zRCI Flow

In the normal testbench, the zemi3->start() method which is responsible for starting the clocks.

However, in the zRCI-based flow, the start() method is not available and there is no control over start of the clock. In this scenario, the hardware may start much before the Transactor software build is created, resulting in the undefined `hardware platform error` during import calls in hardware.

Therefore, it is important to have a clean clock and reset generation logic in case of the zRCI flow, especially during the starting period. To do so, consider the following suggestions:

- Do not use CRESETN from `zceiClockPort`, which can generate spurious clock edge. Instead, use a register to generate reset logic.
- Remove all the CRESETN-related references from the `designFeatures` file.
- Use the clock delay command in the zRCI TCL file, as shown in the following example:

  ```
  config default_clock posedge top.clk
  ```

- Ensure that the clock generation command in the `designFeatures` file starts from the negedge ("-_") and not the posedge (_-).

8.2 Connectivity Checks Limitations

Port Left Unconnected Error

The port used in the transactor instance should be considered as wire and connected to the respective transactor instance. Otherwise, an error message is displayed as follows:

```
Error-[ZEBUUC-XTOR-PORT-UNCONNECTED] Port left Unconnected
src/env/csi_xtor_PPI.v, 326
"CSI_driver u_CSI_driver( .I_LaneModelVersion
(LaneModelVersion[15:0]), .O_DPHY_Ref_ClkByte (DPHY_Ref_ClkByte),
.O_DPHY_rstn (DPHY_rstn),  .I_TxByteClkHS (TxByteClkHS),
.I_TxReadyHS_Lane0 (TxReadyHS0), .I_TxReadyHS_Lane1 (TxReadyHS1),
.I_TxReadyHS_Lane2 (TxReadyHS2), .I_TxReadyHS_Lane3 (TxReadyHS3),
.O_TxDataHS0 (TxDataHS0[7:0]),  .O_TxRequestHS0 (TxRequestHS0),
.O_TxDataHS1 (TxDataHS1[7:0]),  .O_TxRequestHS1 (TxRequestHS1),
.O_TxDataHS2 (TxDataHS2[7:0]),  .O_TxRequestHS2 (TxRequestHS2),
.O_TxDataHS3 (TxDataHS3[7:0]),  .O_TxRequestHS3 (TxRequestHS3),
.O_TxRequestHS_ClkLane (TxRequestHS_ClkLane),
.O_Enable_Tx_ClkLane (Enable_Tx_ClkLane),  .O_Enable_Tx_Lane0
(Enable_Tx_Lane0),  .O_Enable_Tx_Lane1 (Enable_Tx_Lane1),
.O_Enable_Tx_Lane2 (Enable_T ... "
  The port "I_CSI_Ref_Clk" of Xtor module "CSI_driver", instance
  "u_CSI_driver" has been left unconnected
  Ports of Xtor module cannot be left unconnected.
```

In `ddr2_xtor`, there are four transactor instances, `Dqs_out`, and probe and they are not connected. The transactor instances, `Dqs_out`, and probe must be declared as wire and must be connected to their respective transactor instance.

To resolve this error, it is recommended that you ensure all the connections are correct and complete.

Alternatively, you can use one of the following workarounds to fix this error:

- Add a wire on the unconnected port and ensure that the size of the wire matches the width of the port to ensure that the wire appears as connected at the top-level.

The following figure illustrates how to declare the transactor instances as wires.

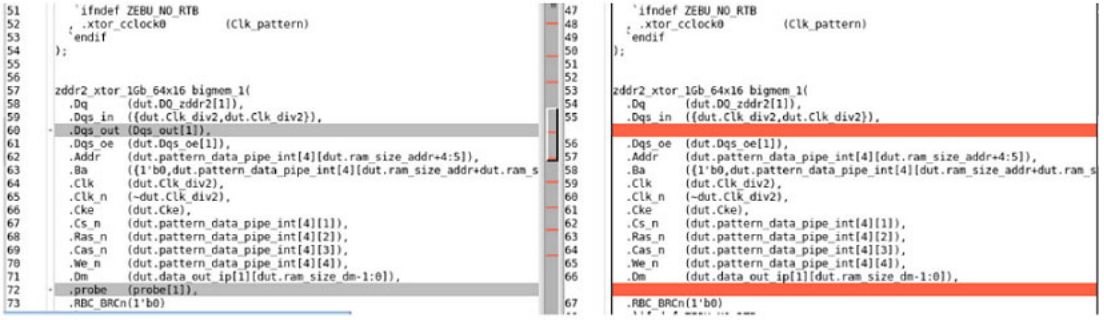


FIGURE 3. Declaring Transactor Instances as Wires

Port-Width Mismatch

The connectivity checks feature displays the following error at the VCS elaboration stage during compilation if there is any port width mismatch:

```
Error-[ZEBUUC-XTOR-PORT-WIDTH-MISMATCH] Port connection width
mismatch
```

To resolve this error, check the port width used in the design with respect to the transactor instance available in your package.

The following figure displays the port width in the design and the transactor instance:

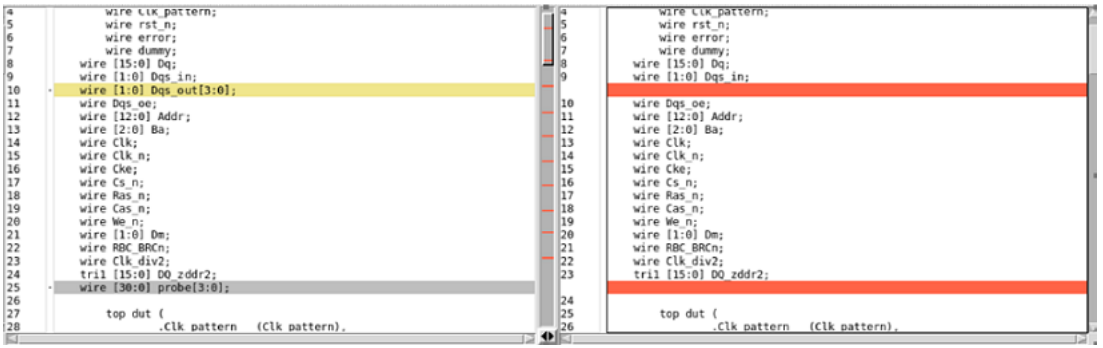


FIGURE 4. Port Width in Design and Transactor Instance

Unconnectivity Checks at IF FPGA

If the respective EDIF file is not sourced, connectivity errors are shown in a popup window as follows:

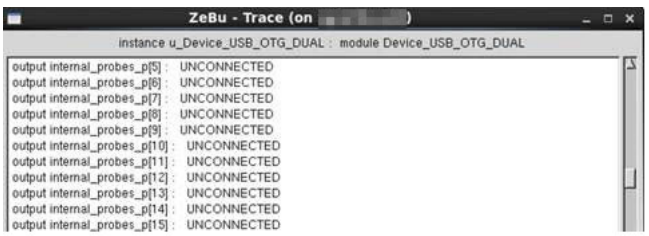
```
Starting DRC Task

INFO: [DRC 23-27] Running DRC with 4 threads

ERROR: [DRC INBB-3] Black Box Instances: Cell 'RTB/
u_usb_driver_Utmi_Device_0/u_usb_otg_bfm_266_Device/
u_Device_USB_OTG_DUAL' of type 'Device_USB_OTG_DUAL' has undefined
contents and is considered a black box.  The contents of this cell
must be defined for opt_design to complete successfully.
```

To resolve this issue, perform the following steps:

1. Source the respective EDIF file as shown in the following figure:



2. Identify the unconnected ports as shown in the following figure:

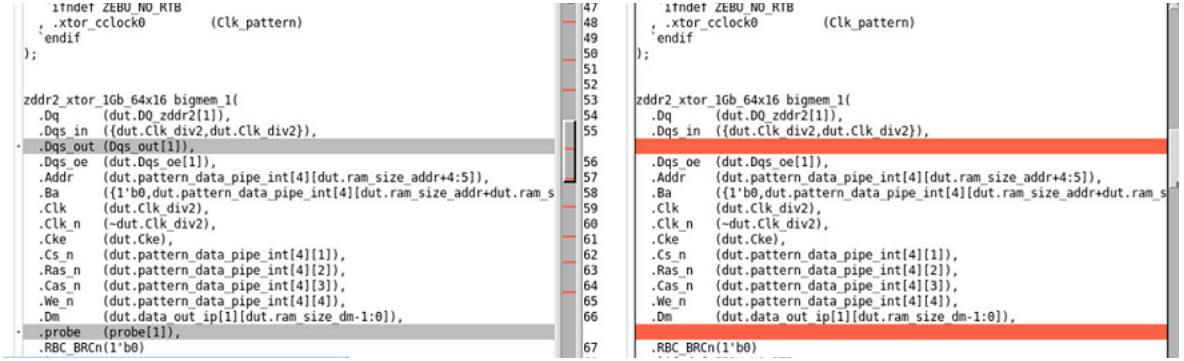


FIGURE 5.

8.3 Driver Detection

If more than 10 instances of the same transactor are used, there might be an error during driver detection.

Contact Synopsys Support for the fix.

9 Fixed STARs in Q-2021.06-SP2 Release

This section lists the STARs fixed for each transactor in Q-2021.06-SP2. Note that the fixed STARs are available in each transactor package. For example, if a STAR is fixed for Video IN transactor, the fix is available in the VideoIn.<version>.sh package.

TABLE 2 Fixed STARs

Transactor	Fixed STARs
CXL	4184113
	4385248
	4161516
DDR5	4248176
ENET	4223329
EtherAVB	4179242
I2C	4316265
MIPI DSI Host	4179102
	4175467
MIPI UFS Device	4389404
	4389199
	4328230
	4388765
	4377883
	4414644
MIPI UFS Host	4271980
MMC Device	4263532
	4263530

TABLE 2 Fixed STARs

Transactor	Fixed STARs
MIPI DSI Host	3984418
	4119394
MMC Device	4132248
PCIe	4242669
	4187601
	4200900
	4292807
PCIe Monitor	4338519
UART	4329037
	4424491
USB2 Host	4177173

10 Fixed STARs in Q-2021.06-SP1 Release

This section lists the STARs fixed for each transactor in Q-2021.06-SP1. Note that the fixed STARs are available in each transactor package. For example, if a STAR is fixed for Video IN transactor, the fix is available in the VideoIn.<version>.sh package.

TABLE 3 Fixed STARs

Transactor	Fixed STARs
AMBA CHIRN	3858026
AMBA AXI Master	4092833
AMBA AXI Monitor	3948034
	4414200
CXL	4079206
	3971696
	3978547
	3945192
	4054598
	3939667
	4016707
	3977068
	3932366
	4020650
	3947565
	3949024
	3947817
EtherAVB	3310891
GDDR6	3649009

TABLE 3 Fixed STARs

Transactor	Fixed STARs
I2C	4132541
	4072526
MIPI CSI	4110400
MIPI DSI Host	3984418
	4119394
MMC Device	4132248
PCIe	4113388
	3984566
	3965626
	4140218
PCIe Monitor	4082885
SDIO Device	4081475
UART	3829692

11 Fixed STARs in Q-2021.06-1 Release

This section lists the STARs fixed in the Q-2021.06-1 release. Note that the fixed STARs are available in each transactor package. For example, if a STAR is fixed for Video IN transactor, the fix is available in the VideoIn.<version>.sh package.

TABLE 4 Fixed STARs

Transactor	Fixed STARs
LPDDR5	3857275
MIPI DSI Device	3833291
	3860709
PCIe	3873371
	3811220
	3761839

12 Fixed STARs in Q-2021.06 Release

This section lists the STARs fixed in Q-2021.06 release. Note that the fixed STARs are available in each transactor package. For example, if a STAR is fixed for Video IN transactor, the fix is available in the VideoIn.<version>.sh package.

TABLE 5 Fixed STARs

Transactor	Fixed STARs
AMBA AHB Slave	9001479042
AMBA AXI3 Master	9001511183
ENET	9001529101
	9001515340
	9001490949
EtherAVB	9001510037
HDMI2 Sink	9001512523
JTAG T32	9001535568
MIPI CSI	9001481899
MIPI DSI	9001535692
PCIeGen4	9001524954
	9001526099
	9001530553
	9001534995
	9001477297
	9001491373
	9001492978

TABLE 5 Fixed STARs

Transactor	Fixed STARs
UART	9001534524
	9001468877
	9001527351
UFS	9001527751
USB2 OTG	9001504831
Video Out	9001536007