SpaceWire CODEC IP

VHDL Verification

Ref:	Uod_Link_Verif
Document Revision:	2.4
Date:	1 April 2009

Document Ref: UoD_Link_Verif

Issue No: 2.4

Issue Date: 1 April 2009

Author: Chris McClements

VHDL Code: cvs tag "uodcodec_cvsrel_2_3"

Number of pages in document: 79

Document History:

Issue	Date	Description	Ref
28/11/02	draft A	Initial.	cmc
01/08/03	1.0	New revision number. References to other link documents updated.	cmc
08/11/04	1.1	Updated with verification reports of CIDL information.	cmc
27/10/05	1.2	SpaceWire CODEC VHDL source code uodcodec_cvsrel_2_0 update.	cmc
20/12/05	1.3	Updates required to Test Case Summary section	cmc
2/07/07	2.1	Updated test-bench and additional requirements conformance tests	cmc
-	2.2	Not released	cmc
22/01/08	2.3	Added tests for CFG_SLOWCLK_10MHZ and specification EXC.ERR.1	cmc
01/04/09	2.4	Corrections after review by ESTEC	cmc

I CONTENTS

II LIST OF FIGURES

III LIST OF TABLES

1 INTRODUCTION

This document defines the VHDL test-bench, the verification test cases matrix, the SpaceWire conformance summary and the of the SpaceWire CODEC IP.

1.1 Summary and main results

The main part of this document defines the functions to which the SpaceWire CODEC IP will be designed and tested.

1.2 Limitations

None

1.3 Aims and Objectives

The aim of this report is to allow the reader to understand the verification structure, design and procedure. The document is also a guide to the results of the verification procedure.

1.4 Documents

In this section the documents referenced in this document are listed.

ECSS-E-ST-50-12C	[AD1]	Space Engineering: SpaceWire – Links, nodes, routers and networks.
ESA VHDL Modelling Guidelines, issue 1, September 1994, ESA/ESTEC	[AD2]	ESA VHDL guidelines for IP and system development. ftp://ftp.estec.esa.nl/pub/vhdl/doc/ModelGuide.pdf
SpacWire CODEC, Functional Specification, 2.3	[AD3]	University of Dundee SpaceWire CODEC Functional Specification
SpaceWire CODEC User Manual, version 2.3	[AD4]	University of Dundee SpaceWire CODEC User manual

Table 1-1 Applicable Documents

1.5 Guide to Report

Section 1 introduces the aims and purpose of this document.

Section 2 defines the VHDL test-bench architecture.

Section 3 defines the verification matrix used to verify the CODEC.

Section 4 defines the verification procedure and the configurations which are run on the test-bench.

Section 5 defines the verification results.

1.6 Definitions

1.6.1 Bit numbering

The following conventions are used in this document

- The most significant bit in a vector is assigned to the highest bit number.
- The least significant bit in a vector is assigned to the lowest numbered bit.

2 TEST-BENCH ARCHITECUTRE AND FUNCTIONAL DESCRIPTION

The architectural and functional design of the test-bench are discussed in this section. The verification methods are also defined.

2.1 Verification Methods

The verification methods employed in the SpaceWire link interface verification are outlined in the following sections.

2.1.1 VHDL code analysis

VHDL code analysis is performed for test cases that cannot be verified automatically by the VHDL test-bench.

2.1.2 VHDL code coverage

VHDL code coverage will be performed by the Modelsim VHDL simulator code coverage option. The code coverage output is a code coverage report which indicates the following.

- Percentage of VHDL code executed in VHDL architectures.
- Line numbers which were not covered by the test run.

The expected code coverage for the verification is 100% of all VHDL statements. Exceptions include finite state machine descriptions which include the VHDL others statement. Any VHDL statements which are not covered by the test-bench are documented in the

2.1.3 Test-bench verification

Automatic test-bench verification shall be employed to determine if the status of output signals matches the expected result. The status of output signals can include the status of received data and time-code transfers. Expected results are documented in the verification log output file and the verification matrix.

2.2 Test-bench environment

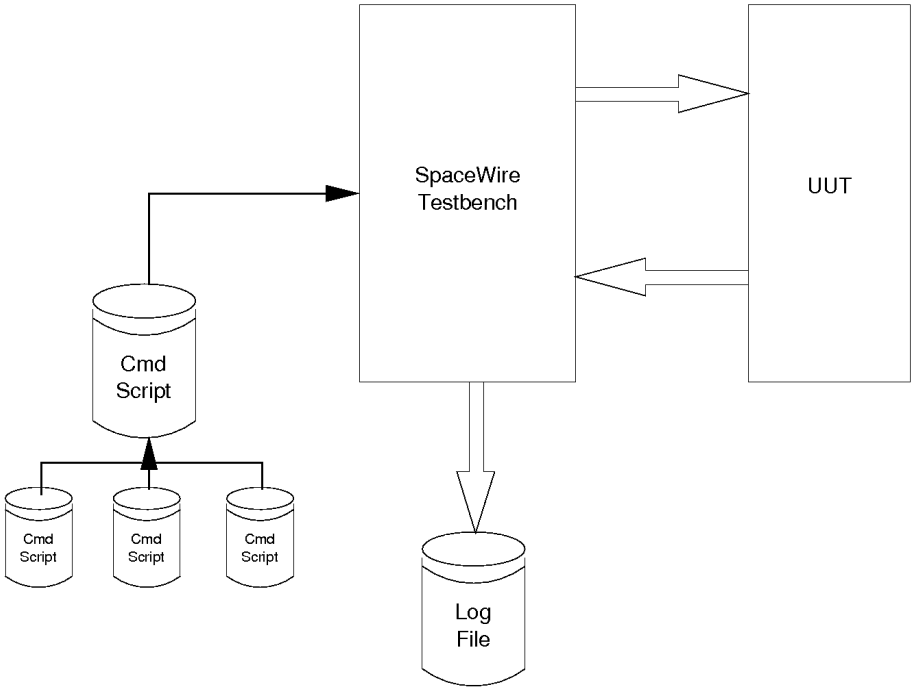

Figure 2-1 Test-bench Environment

The test-bench consists of a command script file, a unit under test which is the SpaceWire link interface, a VHDL test-bench, an output log file and terminal screen output. The following sections outline the function of each test-bench element.

2.2.1 Command Script.

The command script inputs the tests to be performed by the test-bench environment in a sequential manner. It is the purpose of this document via the verification matrix to set the contents of the input command script file. Command script files can reference other command script files. This allows test runs to be repeated for different configurations.

2.2.2 SpaceWire Link Interface Test-bench

The SpaceWire link interface test-bench is based on the Austrian Aerospace SSEPPL test-bench with substantial enhancements. The test-bench consists of an ideal behavioural SpaceWire link interface which connects to the serial bit stream input and output from the UUT. The test-bench environment drives the inputs of the UUT and checks the output from the UUT dependant on the commands read from the command script file. Where possible automatic checks are used to determine if the tests are successful. Tests which are automatic are indicated in the verification matrix.

Note: The document ASIC-TNT-0003-AAE describes the test-bench environment in detail.

2.2.3 UUT

The UUT is the SpaceWire link interface VHDL RTL description.

2.2.4 Log File

The log file allows the user to determine if the test run was successful. The test-bench working output indicates the normal running output of the test-bench. This includes information about state changes or data transfers which are currently taking place.

2.3 Test-bench architecture

The VHDL test-bench architecture (SpaceWire test-bench in Figure 2-1) is shown in Figure 2-2.

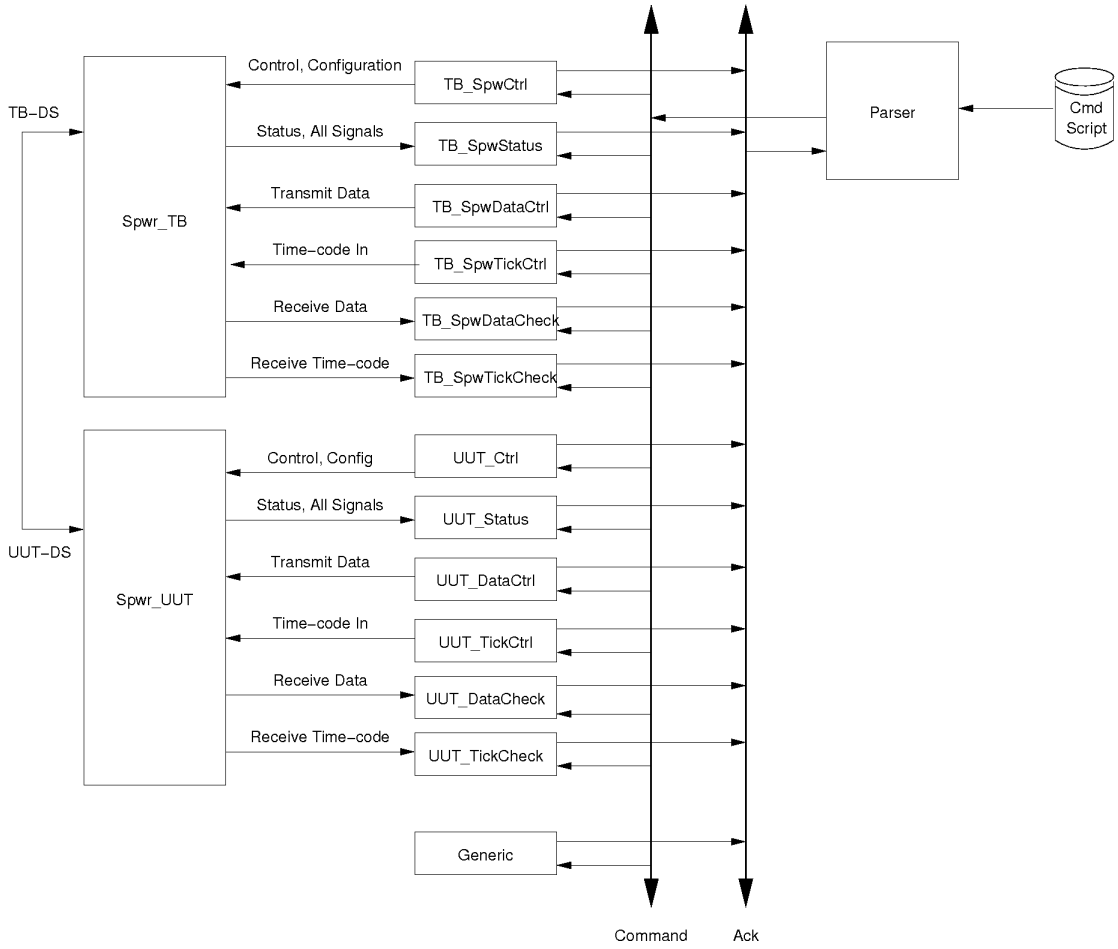

Figure 2-2 Test-bench Architecture

The test-bench components and the commands which can be performed by each component are defined in the following sections.

2.3.1 Spwr_TB

Spwr_TB is a behavioural model of the SpaceWire link interface. It can be commanded to perform functions by the TB_SpwrCtrl test-bench unit and its output can be checked by the TB_SpwrStatusSpwr-Check unit.

2.3.2 Spwr_UUT

Spwr_UUT is a wrapper around the SpaceWire CODEC IP RTL model. The Spwr_UUT component resides in "codec/src/verif/uut-tb/spwrlinkwrap_verif.vhd". The wrapper includes the SpaceWire CODEC IP model, a transmit FIFO to hold transmit data characters, a receive buffer to hold received data characters and a double data rate model to implement the double data rate registers. The files used in the Spwr-UUT module are as follows

```
codec/src/verif/uut-tb/spwrlinkwrap_verif.vhd
codec/src/verif/uut-tb/uut_mem/ asyncfifologic.vhd
codec/src/verif/uut-tb/uut_mem/dpfifo.vhd
codec/src/verif/uut-tb/uut_mem/fifo_out_valid.vhd
codec/src/verif/uut-tb/uut_mem/memblock.vhd
codec/src/verif/uut-tb/uut_mem/readptr.vhd
codec/src/verif/uut-tb/uut_mem/writeptr.vhd
```

2.3.3 Parser

The parser reads from the command files and builds command lists for each of the test-bench components. A command is an array of strings which have meaning at the destination component. The command format is shown below:

```
<COMPONENT> <COMMAND> <ARGUMENTS LIST...>
```

Where

COMPONENT:	The test-bench component to run the command
COMMAND:	The command to perform
ARGUMENTS LIST:	The list of command arguments separated by spaces. The maximum number of arguments is determined by the constant MAX_TOKEN_LEN in the file "src/verif/package/tb_pkg.vhd" (the default value is 50).

The component names which can be addressed in the test-bench listed in Table 2-1.

Component	Test-bench name	Description
Generic	TB_GENERIC	Generic test-bench commands such as command reference number and print commands
TB_SpwCtrl	TB_SPWR_CTRL	Spwr_TB control and configuration inputs
TB_SpwStatus	TB_SPWR_STATUS	Spwr_TB status checking and signal analysis
TB_SpwDataCtrl	TB_SPWR_DATA_CTRL	Write data to Spwr_TB transmit FIFO
TB_SpwTickCtrl	TB_SPWR_TICK_CTRL	Write data to Spwr_TB time-code input
TB_SpwDataCheck	TB_SPWR_DATA_CHECK	Read and check data from Spwr_TB receive FIFO.
TB_SpwTickCheck	TB_SPWR_TICK_CHECK	Receive and check time-codes from Spwr_TB.
UUT_Ctrl	UUT_CTRL	Control (Clock, reset, etc.) and configuration of UUT.
UUT_Status	UUT_STATUS	Status checking and signal analysis from UUT
UUT_DataCtrl	UUT_DATA_CTRL	Write data to the Spwr_UUT transmit FIFO.
UUT_TickCtrl	UUT_TICK_CTRL	Write time-codes to the Spwr_UUT.
UUT_DataCheck	UUT_DATA_CHECK	Receive and check data from the Spwr_UUT receive FIFO.
UUT_TickCheck	UUT_TICK_CHECK	Receive and check time-codes from the Spwr-UUT.

Table 2-1 Test-bench Component Names

2.3.4 TB Generic (TB_GENERIC)

The generic component is used to log text to the simulator log file and record the reference number of the tests being performed.

2.3.4.1 Wait

```
TB WAIT <time>
  <time>      Time to wait
```

Wait for a period of time. During the wait period no commands are executed.

2.3.4.2 Reference

```
TB REF <ref...>
  <ref...>    List of reference designators
```

Assign a reference to a signal so the reference can be traced in the simulator wave window. Each reference designator can be a text string or number. References are also echoed to the standard output.

2.3.4.3 Halt

```
TB HALT
```

Cause an assert false failure VHDL statement to be executed.

2.3.4.4 Echo

```
TB ECHO <text...>
```

Echo text to the simulator standard output.

2.3.5 TB_SpwrCtrl (TB_SPWR_CTRL)

TB_SpwrCtrl interfaces with the Spwr-TB behavioural VHDL model as follows:

Signal	Description
LnkDisable	Link disable input into Spwr-TB.
LnkReset	Reset of Spwr-TB.
LnkStart	Link start input into Spwr-TB.
LnkAuto	Auto start input into Spwr-TB.
WireOutDCtrl	Set DOUT to value else use Spwr-TB DOUT value.
WireOutSCtrl	Set SOUT to value else use Spwr-TB SOUT value.
TxParityErr	Cause Spwr-TB parity error.
TxEnSlowClk	Enable slow clock TxClock.
TxDataPriority	Data priority over FCT characters.
TxClkPeriod	Clock period for initialisation, fast and slow clocks.
TxResetValueD	Reset value of D.
TxResetValueS	Reset value of S.
Timings	Timings of interface state machine.
ClSys	Input clock.
ForceTx10Mbit	Force 10 Mbits input.

Table 2-2 TB Spwr Ctrl signals

The commands which can be performed are listed in the following sections.

2.3.5.1 Link Disable

```
TB_SPWR_CTRL INTERFACE <ON|OFF>
  <ON|OFF>    Set link disable ON or OFF
```

Set the disable bit of the Spwr-TB interface to ON or OFF.

2.3.5.2 Parity Error

```
TB_SPWR_CTRL PARITYERROR <ON|OFF>
  <ON|OFF>    Set ParityError output ON or OFF.
```

Enable parity errors out of transmit link.

2.3.5.3 Serial control

```
TB_SPWR_CTRL SET <WireOutD|WireOutS> <LOW|HIGH|Z>
  <WireOutD|WireOutS>   Data or strobe
  <LOW|HIGH|Z>    Set to '0', '1' or 'Z'
```

Set the Spwr-TB serial output value to low, high or to follow the output from the Spwr-TB transmitter.

2.3.5.4 Reset

```
TB_SPWR_CTRL LINKRESET <time>
  <time>      Time to perform reset
```

Set reset active then inactive after <time>

2.3.5.5 Link Start

```
TB_SPWR_CTRL LINKSTART <ON|OFF>
  <ON|OFF>    Set link start ON or OFF
```

Set the LinkStart bit of the Spwr-TB ON or OFF.

2.3.5.6 Link Auto Start

```
TB_SPWR_CTRL LINKAUTO <ON|OFF>
  <ON|OFF>    Set link auto-start ON or OFF
```

Set the LinkAuto bit of the Spwr-TB ON or OFF.

2.3.5.7 Link Timings

```
TB_SPWR_CTRL LINKCLOCK <INIT|FAST|SLOW|ALL> <time>
  <INIT|FAST|SLOW|ALL>  Timing parameter to set
  <time>      Value to set timing parameter to
```

Set one or all of the transmit bit clock parameters. The parameters are defined as follows:

INIT	Transmitter start-up rate.
FAST	Transmitter rate in Run state.
SLOW	Transmitter rate when sending NULLs.
ALL	Set all transmit rates.

2.3.5.8 Data Priority

```
TB_SPWR_CTRL DATAPRIORITY <ON|OFF>
  <ON|OFF>    Set data priority over FCTs ON or OFF.
```

Set the data priority over FCTs to ON or OFF (TxDataPriority)

2.3.5.9 Strobe Reset

```
TB_SPWR_CTRL STROBERESVAL <1|0>
  <1|0>       Set serial strobe reset value.
```

Set the strobe reset value to 1 or 0.

2.3.5.10 Data Reset

```
TB_SPWR_CTRL DATARESVAL <1|0>
  <1|0>       Set serial data reset value.
```

Set the data reset value to 1 or 0.

2.3.5.11 Link Timing

```
TB_SPWR_CTRL LINKTIMING <INIT|RESET2WAIT|WAIT2READY|STARTED2RESET|
CONNECT2RESET|TIMEOUT|STROBERES> <time>
  <INIT|RESET2WAIT|WAIT2READY|STARTED2RESET|CONNECT2RESET|
  TIMEOUT|STROBERES>    Timing parameter to set, see below
  <time>      Time to set parameter to.
```

Set the timing parameters of the interface state machine.

INIT	Initialise timing parameters to default values
RESET2WAIT	Delay between ErrorReset and ErrorWait states
WAIT2READY	Delay between ErrorWait and Ready states
STARTED2RESET	Delay between Started and ErrorReset states
CONNECT2RESET	Delay between Connecting and ErrorReset states
TIMEOUT	Disconnect detection timeout
STROBERES	Time between strobe and data reset

2.3.5.12 Synchronise

```
TB_SPWR_CTRL SYNCTB
```

Wait for all commands to complete.

2.3.6 TB_SpwrStatus (TB_SPWR_STATUS)

TB_SpwrCtrl interfaces with the Spwr-TB behavioural VHDL model as follows:

Signal	Description
WireOutD	Serial data
WireOutS	Serial strobe
TxFifoRdyIn	Tx FIFO ready flag.
RxFifoDataOut	Rx data output
RxFifoRdyOut	Rx FIFO data ready.
TickOut	Time-code available output
TimeOut	Time-code output
LnkStatus	Link Status output 2:0 Interface state 3 Connection error 4 Parity error 5 Credit error 6 Escape error 7 Transmit data error 8 Tx FIFO empty 9 Rx FIFO full 10 Transmitter sending a NULL 11 Receive buffer request to send FCT 12 Transmitter acknowledge send FCT 13 Receiver got NULL 14 Receiver got FCT 15 Receiver got NChar 16 Transmitter is idle 17 Receiver is idle

Table 2-3 TB Spwr Status signals

Spwr-TB status checking commands are of the form

```
TB_SPWR_STATUS <SIGNAL> VAL_NOW_EQUALS <value>
TB_SPWR_STATUS <SIGNAL> VAL_ALWAYS_EQUALS <timeout> <inc> <value>
TB_SPWR_STATUS <SIGNAL> VAL_BECOMES <timeout> <inc> <value>
TB_SPWR_STATUS <SIGNAL> VAL_NOT_EQUALS <value>
TB_SPWR_STATUS <SIGNAL> VAL_ALWAYS_NOT_EQUALS <timeout> <inc>
<value>
```

Where

SIGNAL: The signal to check

VAL_NOW_EQUALS: Check that the signal is equal to <value>

VAL_ALWAYS_EQUALS: Check the signal is equal to <value> for the timeout period. The signal is checked every <inc> period.

VAL_BECOMES: Check that the signal is always equal to <value> over the timeout period <timeout>. The signal is checked every <inc> period.

VAL_NOT_EQUALS: Check that the signal is not equal to <value>

VAL_ALWAYS_NOT_EQUALS: Check that the signal is not equal to <value> for the complete timeout period <timeout>. The signal is checked every <inc> period.

The signal argument can have one of the following values:

SERIAL_D	SERIAL_S
SERIAL_DS	TXFIFO_RDY
RXFIFO_DATA	RXFIFO_RDY
TICKOUT	TIMEOUT
STATE	CONN_ERR
PARITY_ERR	CREDIT_ERR
ESCAPE_ERR	TXDATA_ERR
TXFIFO_EMPTY	RXFIFO_FULL
TX_NULL	TX_REQ_FCT
TX_ACK_FCT	GOT_NULL
GOT_FCT	GOT_NCHAR
TXIDLE	RXIDLE

Table 2-4 TB_SpwrStatus signal argument values

2.3.7 TB_SpwrDataCtrl (TB_SPWR_DATA_CTRL)

TB_SpwrDataCtrl interfaces with the Spwr-TB behavioural VHDL model as follows:

Signal	Description
TxFifoRdyIn	Ready flag from Spwr-TB.
TxFifoShiftIn	Write a character to the Spwr-TB.
TxFifoDataIn	Data input to Spwr-TB. Bit 0 is the control flag and bit 8:1 are the data bits.
ClSys	Input clock.

Table 2-5 TB Spwr Data Ctrl signals

The commands which can be performed are listed in the following sections.

2.3.7.1 Transmit Data

```
TB_SPWR_DATA_CTRL TRANSMIT_DATA <timeout> <init> <num>
  <timeout>   Time for command to complete
  <init>      Initialisation of PN generator in hex
  <num>       Number of bytes to write to Spwr-TB as an integer
```

Write a number of bytes to the transmit FIFO starting with <init>. A pseudo random noise generator (LFSR) is used to generate the data patterns. The command will continue to execute until it completes or the timeout is reached.

2.3.7.2 Transmit Packet

```
TB_SPWR_DATA_CTRL TRANSMIT_PACKET <timeout> <init> <num> <addr...>
  <timeout>   Time for command to complete
  <init>      Initialisation of PN generator in hex
  <num>       Number of bytes to write to Spwr-TB as an integer
  <addr...>   List of packet addresses to send
```

Write a packet of size num to the transmit FIFO of Spwr-TB starting with an address list <addr> followed by <init>. A pseudo random noise generator (LFSR) is used to generate the data patterns. The packet is terminated with an EOP. The command will continue to execute until it completes or the timeout occurs.

2.3.7.3 Transmit Empty Packets

```
TB_SPWR_DATA_CTRL TRANSMIT_EMPTY <timeout> <type> <num>
  <timeout>   Time for command to complete
  <type>      Type of empty packet, see below
  <num>       Number of empty packets to write
```

Write a number of empty packets into the transmit FIFO. The command will continue to execute until it completes or the timeout occurs. The argument <type> sets the type of empty packet to write according to the following table.

<type>	Type of empty packet to write
0	EOP
1	EEP
2	Mixed, EOP first
3	Mixed, EOP second

Table 2-6 Empty packet <type> argument setting

2.3.7.4 Transmit Bytes

```
TB_SPWR_DATA_CTRL TRANSMIT_BYTE <bytes...>
  <timeout>   Time for command to complete
  <bytes...>  List of bytes to write to transmit FIFO in hex
```

Write a list of bytes to the transmit FIFO. If byte(2:0) id “111” then the byte is interpreted as an FCT. If byte(2:0) is “001” then byte is interpreted as an escape character. The command will continue to execute until it completes or the timeout period is reached.

2.3.7.5 Synchronise

```
TB_SPWR_DATA_CTRL SYNCTB
```

Wait for all commands to complete.

2.3.8 TB_SpwrTickCtrl (TB_SPWR_TICK_CTRL)

TB_SpwrTickCtrl interfaces with the Spwr-TB behavioural VHDL model as follows:

Signal	Description
TickIn	Perform tick in to Spwr-TB
TimeIn	Time-code into Spwr-TB on TickIn
ClSys	Input clock.

Table 2-7 TB Spwr Tick Ctrl signals

The commands which can be performed are listed in the following sections.

2.3.8.1 Transmit Time-code

```
TB_SPWR_TICK_CTRL TIMECODEIN <timecode>
  <timecode>  Time-code to write to the Spwr-TB
```

Insert a time-code into the Spwr-TB.

2.3.8.2 Synchronise

```
TB_SPWR_TICK_CTRL SYNCTB
```

Wait for all commands to complete.

2.3.9 TB_SpwrDataCheck (TB_SPWR_DATA_CHECK)

TB_SpwrDataCheck interfaces with the Spwr-TB behavioural VHDL model as follows:

Signal	Description
RxFifoShiftOut	Read a byte out of the receive FIFO
RxFifoRdyOut	Data is ready to be read from the receive FIFO
RxFifoDataOut	Data output from receive FIFO
ClSys	Interface clock

Table 2-8 TB Spwr Data Check signals

The commands which can be performed are listed in the following sections.

2.3.9.1 Receive Data

```
TB_SPWR_DATA_CHECK RECEIVE_DATA <timeout> <init> <num>
  <timeout>   Time for command to complete
  <init>      Initialisation of PN generator in hex
  <num>       Number of bytes to write to Spwr-TB as an integer
```

Receive a number of bytes from the receive FIFO starting with <init>. Subsequent bytes are generated by a pseudo random noise generator (LFSR). The command will execute to completion or until the timeout period expires.

2.3.9.2 Receive Packet

```
TB_SPWR_DATA_CHECK RECEIVE_PACKET <timeout> <init> <num> <addr list>
  <timeout>   Time for command to complete
  <init>      Initialisation of PN generator in hex
  <num>       Number of bytes to write to Spwr-TB as an integer
```

Receive a packet from the receive FIFO starting with a list of addresses <addr> and then the <init> value. Subsequent bytes are generated by a pseudo random noise generator (LFSR). The command will execute to completion or until the timeout period expires.

2.3.9.3 Receive Empty Packets

```
TB_SPWR_DATA_CHECK RECEIVE_EMPTY <timeout> <type> <num>
  <timeout>   Time for command to complete
  <init>      Type of empty packets to receive
  <num>       Number of empty packets to receive
```

Receive a number of empty packets. The command will execute until completion or until the timeout period is reached. The type of empty packets which can be received is set by the argument <type> as listed in Table 2-9.

<type>	Type of empty packet to write
0	EOP
1	EEP
2	Mixed, EOP first
3	Mixed, EOP second

Table 2-9 Empty packet <type> argument setting

2.3.9.4 Receive Bytes

```
TB_SPWR_DATA_CHECK RECEIVE_BYTES <timeout> <bytes...>
  <timeout>   Time for command to complete
  <bytes...>  List of bytes to receive in hex
```

Receive an explicit list of bytes supplied in the command. The command will either complete or timeout if the timeout period is reached.

2.3.9.5 Synchronise

```
TB_SPWR_DATA_CHECK SYNCTB
```

Wait for all commands to complete.

2.3.10 TB_SpwrTickCheck (TB_SPWR_TICK_CHECK)

TB_SpwrTickCheck interfaces with the Spwr-TB behavioural VHDL model as follows:

Signal	Description
RxTickOut	Tick from Spwr-TB
RxTimeOut	Time-code out from Spwr-TB
ClSys	Interface clock

Table 2-10 TB Spwr Tick Check signals

The commands which can be performed are listed in the following sections.

2.3.10.1 Receive Time-code

```
TB_SPWR_TICK_CHECK RECEIVE_TIMECODE <timeout> <tick>
  <timeout>   Time for command to complete
  <tick>      Time-code to receive
```

Receive a time-code from the Spwr-TB component. The command will execute until the time-code is received or the time-out period is reached.

2.3.11 UUT_Ctrl (UUT_CTRL)

TB_SpwrTickCheck interfaces with the Spwr-TB behavioural VHDL model as follows:

Signal	Description
SYSCLK	UUT system clock
TXCLK	UUT transmit clock
SLOWCLK	UUT default initialisation clock and timings clock
RDCLK	Receive buffer clock.
RST_N	Asynchronous reset (Active low)
CFG_MAXCREDIT	Receive buffer maximum credit value
CFG_SLOWRATE_TXCLK	Configuration of initialisation rate for transmit clock dividers
CFG_SLOWRATE_SYSCLK	Configuration of initialisation rate for system clock 10MHz clock divider
CFG_SLOW_CE	Clock enable for system clock (10MHz)
TXRATE	Divider for variable transmit rate
RX_PROGVAL	Programmable flag value
LINK_START	Start link for interface state machine.
LINK_DISABLE	Disable the interface state machine
AUTO_START	Auto start the interface state machine
FLUSH_TX	Spill packets from the transmitter buffer

Table 2-11 UUT_Ctrl test-bench interface signals

The commands which can be performed are listed in the following sections.

2.3.11.1 System clock

```
UUT_CTRL SYSCLK <period>
  <period>    Period of the system clock
```

Set the period of the system clock to <period>

2.3.11.2 Transmit clock

```
UUT_CTRL TXCLK <ON|OFF> <period>
  <ON|OFF>    Set TXCLK input ON or OFF
  <period>    Period of the transmit clock
```

Set the period of the transmit clock to <period>. When set to OFF then the input is set to zero.

2.3.11.3 Default initialisation clock

```
UUT_CTRL SLOWCLK <ON|OFF> <period>
  <ON|OFF>    Set SLOWCLK clock input ON or OFF
  <period>    Period of the initialisation clock
```

Set the period of the default initialisation clock to <period>. When set to OFF then the input is set to zero.

2.3.11.4 Receiver buffer clock

```
UUT_CTRL RDCLK <ON|OFF> <period>
  <ON|OFF>    Set RDCLK clock input ON or OFF
  <period>    Period of the read buffer clock
```

Set the period of the receive buffer clock to <period>. When set to OFF then the input is set to zero.

2.3.11.5 Reset

```
UUT_CTRL RST <period>
  <period>    Reset active period
```

Set the RST input of Spwr-UUT to active and then inactive after <period>.

2.3.11.6 Maximum credit

```
UUT_CTRL MAX_CREDIT <value>
  <value>     Value of CFG_MAXCREDIT in hex
```

Set the CFG_MAXCREDIT input to <value>

2.3.11.7 Transmit clock default rate divider

```
UUT_CTRL SLOWRATE_TXCLK <value>
  <value>     Value of CFG_SLOWRATE_TXCLK in hex
```

Set the CFG_SLOWRATE_TXCLK input to <value>

2.3.11.8 System clock 10MHz rate divider

```
UUT_CTRL SLOWRATE_SYSCLK <value>
  <value>     Value of CFG_SLOWRATE_SYSCLK in hex
```

Set the CFG_SLOWRATE_SYSCLK input to <value>

2.3.11.9 External 10MHz clock enable for SYSCLK

```
UUT_CTRL SLOW_CE <ON|OFF> <period> <width>
  <ON|OFF>    Set the SLOW_CE input ON or OFF
  <period>    Set the period of SLOW_CE
  <width>     Set the width of the SLOW_CE pulse.
```

Set the CFG_SLOW_CE input. When OFF the CFG_SLOW_CE is set low. The input pulse is shaped as follows:

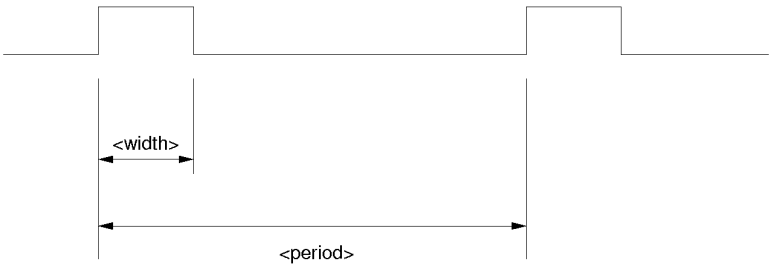

Figure 2-3 CFG_SLOW_CE configuration and control

2.3.11.10 Transmit rate divider

```
UUT_CTRL TXRATE <value>
  <value>     Set TXRATE to value
```

Set the input TXRATE to <value>.

2.3.11.11 Link start

```
UUT_CTRL LINK_START <ON|OFF>
  <ON|OFF>    Set link start ON or OFF
```

Set the input LINK_START to ON or OFF.

2.3.11.12 Link disable

```
UUT_CTRL LINK_DISABLE <ON|OFF>
  <ON|OFF>    Set link disable ON or OFF
```

Set the input LINK_DISABLE to ON or OFF.

2.3.11.13 Auto start

```
UUT_CTRL AUTO_START <ON|OFF>
  <ON|OFF>    Set auto start ON or OFF
```

Set the input AUTO_START to ON or OFF.

2.3.11.14 Flush TX

```
UUT_CTRL FLUSH_TX <ON|OFF>
  <ON|OFF>    Set flush TX ON or OFF
```

Set the input FLUSH_TX to ON or OFF.

2.3.12 UUT_status

TB_SpwrCtrl interfaces with the Spwr-TB behavioural VHDL model as follows:

Signal	Description
DOUT	Serial data
SOUT	Serial strobe
TX_FULL	Transmit FIFO full flag
RXBUF_CLK	Receive buffer clock
RX_DATA	Receive data
RX_EMPTY	Receive empty flag
RX_PROGFLAG	Receive programmable flag
TICK_OUT	Tick out from Spwr-UUT
TIME_OUT	Time out from Spwr-UUT
DISC_RUN_ERR	Disconnect error in run state
PARITY_RUN_ERR	Parity error in run state
ESCAPE_RUN_ERR	Escape error in run state
CREDIT_RUN_ERR	Credit error in run state
STATUS	Link status output 0 Disconnect error 1 Parity error 2 Escape error 3 Receive credit error 4 Transmit credit error 7:5 Interface state 8 Link running 9 Receiver got NULL 10 Receiver got FCT 11 Receiver got Nchar 12 Receiver got time-code 13 Transmitter has credit 14 Nchar sequence error 15 Time-code sequence error

Table 2-12 TB Status signals

Spwr-UUT status checking commands are of the form

```
UUT_STATUS <SIGNAL> VAL_NOW_EQUALS <value>
UUT_STATUS <SIGNAL> VAL_ALWAYS_EQUALS <timeout> <inc> <value>
UUT_STATUS <SIGNAL> VAL_BECOMES <timeout> <inc> <value>
UUT_STATUS <SIGNAL> VAL_NOT_EQUALS <value>
UUT_STATUS <SIGNAL> VAL_ALWAYS_NOT_EQUALS <timeout> <inc> <value>
```

Where

SIGNAL: The signal to check

VAL_NOW_EQUALS: Check that the signal is equal to <value>

VAL_ALWAYS_EQUALS: Check the signal is equal to <value> for the timeout period. The signal is checked every <inc> period.

VAL_BECOMES: Check that the signal is always equal to <value> over the timeout period <timeout>. The signal is checked every <inc> period.

VAL_NOT_EQUALS: Check that the signal is not equal to <value>

VAL_ALWAYS_NOT_EQUALS: Check that the signal is not equal to <value> for the complete timeout period <timeout>. The signal is checked every <inc> period.

The signal argument can have one of the following values:

SERIAL_D	SERIAL_S
SERIAL_DS	TX_FULL
RXBUF_CLK	RX_DATA
RX_EMPTY	RX_PROGFLAG
TICK_OUT	TIME_OUT
DISC_RUN_ERR	PARITY_RUN_ERR
ESCAPE_RUN_ERR	CREDIT_RUN_ERR
STAT_DISC_ERR	STAT_PARITY_ERR
STAT_ESCAPE_ERR	STAT_RXCREDIT_ERR
STAT_TXCREDIT_ERR	STAT_STATE
STAT_RUNNING	STAT_GOT_NULL
STAT_GOT_FCT	STAT_GOT_NCHAR
STAT_GOT_TIMECODE	STAT_HAS_CREDIT
STAT_NCHAR_SEQ_ERR	STAT_TCODE_SEQ_ERR

Table 2-13 Signal argument for UUT_Status command

2.3.13 UUT_Data_Ctrl

UUT_Data_Ctrl interfaces with the Spwr-UUT model as shown in Table 2-14.:

Signal	Description
TX_WRCLK	Write clock into transmitter FIFO
TX_DATA	Data input to transmitter FIFO
TX_WRITE	Write data to transmitter FIFO on rising edge of TX_WRCLK
TX_FULL	Transmitter FIFO is full

Table 2-14 UUT_Data_Ctrl signals

The commands which can be performed are listed in the following sections.

2.3.13.1 Transmit Data

```
UUT_DATA_CTRL TX_WRCLK <period>
  <period>    Period of the transmit FIFO write clock
```

Set the period of the transmit FIFO write clock to <period>.

2.3.13.2 Transmit Data

```
UUT_DATA_CTRL TRANSMIT_DATA <timeout> <init> <num>
  <timeout>   Time for command to complete
  <init>      Initialisation of PN generator in hex
  <num>       Number of bytes to write to Spwr-TB as an integer
```

Write a number of bytes to the transmit FIFO starting with <init>. A pseudo random noise generator (LFSR) is used to generate the data patterns. The command will continue to execute until it completes or the timeout is reached.

2.3.13.3 Transmit Packet

```
UUT_DATA_CTRL TRANSMIT_PACKET <timeout> <init> <num> <addr>
  <timeout>   Time for command to complete
  <init>      Initialisation of PN generator in hex
  <num>       Number of bytes to write to Spwr-TB as an integer
  <addr>      List of packet addresses to send
```

Write a packet of size num to the transmit FIFO of Spwr-TB starting with an address list <addr> followed by <init>. A pseudo random noise generator (LFSR) is used to generate the data patterns. The packet is terminated with an EOP. The command will continue to execute until it completes or the timeout occurs.

2.3.13.4 Transmit Empty Packets

```
UUT_DATA_CTRL TRANSMIT_EMPTY <timeout> <type> <num>
```

```
<timeout>   Time for command to complete
<type>      Type of empty packet, see below
<num>       Number of empty packets to write
```

Write a number of empty packets into the transmit FIFO. The command will continue to execute until it completes or the timeout occurs. The argument <type> sets the type of empty packet to write according to the following table.

<type>	Type of empty packet to write
0	EOP
1	EEP
2	Mixed, EOP first
3	Mixed, EOP second

Table 2-15 Empty packet <type> argument setting

2.3.13.5 Transmit Bytes

```
UUT_DATA_CTRL TRANSMIT_BYTE <bytes...>
  <timeout>   Time for command to complete
  <bytes...>  List of bytes to write to transmit FIFO in hex
```

Write a list of bytes to the transmit FIFO. If byte(2:0) id “111” then the byte is interpreted as an FCT. If byte(2:0) is “001” then byte is interpreted as an escape character. The command will continue to execute until it completes or the timeout period is reached.

2.3.13.6 Synchronise

```
UUT_DATA_CTRL SYNCTB
```

Wait for all commands to complete.

2.3.14 UUT_Tick_Ctrl (UUT_TICK_CTRL)

TB_SpwrTickCtrl interfaces with the Spwr-UUT model as follows:

Signal	Description
TICK_IN	Perform tick in to Spwr-UUT
TIME_IN	Time-code into Spwr-UUT on TICK_IN
SYSCLK	Input clock.

Table 2-16 UUT Tick Ctrl signals

The commands which can be performed are listed in the following sections.

2.3.14.1 Transmit Time-code

```
UUT_TICK_CTRL TIMECODEIN <timecode>
  <timecode>  Time-code to write to the Spwr-TB
```

Insert a time-code into the Spwr-TB.

2.3.14.2 Synchronise

```
UUT_TICK_CTRL SYNCTB
```

Wait for all commands to complete.

2.3.15 UUT_Data_Check

TB_SpwrDataCheck interfaces with the Spwr-UUT model as follows:

Signal	Description
RXBUF_CLK	Read buffer clock input
RX_DATA	Receive FIFO data
RX_READ	Receive FIFO read
RX_EMPTY	Read from receive FIFO

Table 2-17 UUT data check signals

The commands which can be performed are listed in the following sections.

2.3.15.1 Receive Data

```
UUT_DATA_CHECK RECEIVE_DATA <timeout> <init> <num>
  <timeout>   Time for command to complete
  <init>      Initialisation of PN generator in hex
  <num>       Number of bytes to write to Spwr-TB as an integer
```

Receive a number of bytes from the receive FIFO starting with <init>. Subsequent bytes are generated by a pseudo random noise generator (LFSR). The command will execute to completion or until the timeout period expires.

2.3.15.2 Receive Packet

```
UUT_DATA_CHECK RECEIVE_PACKET <timeout> <init> <num> <addr list>
  <timeout>   Time for command to complete
  <init>      Initialisation of PN generator in hex
  <num>       Number of bytes to write to Spwr-TB as an integer
```

Receive a packet from the receive FIFO starting with a list of addresses <addr> and then the <init> value. Subsequent bytes are generated by a pseudo random noise generator (LFSR). The command will execute to completion or until the timeout period expires.

2.3.15.3 Receive Empty Packets

```
UUT_DATA_CHECK RECEIVE_EMPTY <timeout> <type> <num>
  <timeout>   Time for command to complete
  <init>      Type of empty packets to receive
  <num>       Number of empty packets to receive
```

Receive a number of empty packets. The command will execute until completion or until the timeout period is reached. The type of empty packets which can be received is set by the argument <type> as listed in Table 2-18.

<type>	Type of empty packet to write
0	EOP
1	EEP
2	Mixed, EOP first
3	Mixed, EOP second

Table 2-18 Empty packet <type> argument setting

Receive Bytes

```
UUT_DATA_CHECK RECEIVE_BYTES <timeout> <bytes...>
  <timeout>   Time for command to complete
  <bytes...>  List of bytes to receive in hex
```

Receive an explicit list of bytes supplied in the command. The command will either complete or timeout if the timeout period is reached.

2.3.15.4 Synchronise

```
UUT_DATA_CHECK SYNCTB
```

Wait for all commands to complete.

2.3.16 UUT_Tick_Check

UUT_TickCheck interfaces with the Spwr-UUT model as follows:

Signal	Description
TICK_OUT	Tick out from Spwr-UUT
TIME_OUT	Time output from Spwr-UUT
SYSCLK	System clock

Table 2-19 UUT Tick Check signals

The commands which can be performed are listed in the following sections.

2.3.16.1 Receive Time-code

```
UUT_TICK_CHECK RECEIVE_TIMECODE <timeout> <time>
  <timeout>   Time for command to complete
  <time>      Time-code to receive
```

Receive a time-code from the Spwr-UUT component. The command will execute until the time-code is received or the time-out period is reached.

2.3.16.2 Synchronise

```
UUT_TICK_CHECK SYNCTB
```

Wait for all commands to complete.

3 VERIFICATION MATRIX

The verification matrix details the test cases which are performed on the SpaceWire link interface. This section includes the following:

- A section titled test cases which lists the test cases performed in hierarchical order. This section cross references the test cases with the functional specification [AD3]. A verification strategy is included with each test cases and the verification method is defined.
- A summary of the test cases to be performed.
- A SpaceWire link interface conformance summary table which lists all the clauses from the SpaceWire standard [AD1] which are applicable to the SpaceWire link interface (See [AD1] section 12.2.4 table 19).

3.1 Test Cases

The test cases verification matrix below defines the test procedures which are performed on the UoD Link Interface. The purpose of the test cases is to verify the function of the SpaceWire link interface. The following points describe the columns in the verification matrix table:

- Test number which is the reference used when referring to the test
- Cross reference number with the functional specification
- Description of the verification method used to test the requirement
- T/A, T – Test is performed by the test-bench, A – Test is performed by analysis
- Test is automatically checked by VHDL test-bench

3.2 Configuration analysis

The UoD SpaceWire link interface can be configured to suit the users application. The configuration options and the verification method used is outlined below. The test-bench has the ability to concatenate test-bench files together therefore different configurations and clock speeds can be selected to run the basic set of test cases.

Configuration option	Effects	Verification
CFG_PIPELINE	Top level global configuration	All test cases are run with CFG_PIPELINE='0' and CFG_PIPELINE='1'

CFG_DDROUT	Top level transmit encoding method	All test cases are run with CFG_DDROUT='0' and CFG_DDROUT = '1'.
CFG_BITCLK	Transmit bit clock configuration.	All test cases are run for each CFG_BITCLK configuration
CFG_SLOWCLK_10MHZ	Slowclk 10MHz only configuration	Test cases for SYS_SLOWCLK, SYS_SLOWCLK_DIV, TXCLK_SLOWCLK and TXCLK_SLOWCLK_DIV are run with CFG_SLOWCLK_10MHZ set. Both in pipelined and non pipelined mode
CFG_SYNCRDCLK	Receive buffer clock	CFG_SYNCRDCLK is used in T.IF.RBUF.7
CFG_DISCARD_EMPTY_PKT	Empty packet handling by receiver	Test case T.EXC.ERR.5 covers CFG_DISCARD_EMPTY_PKT.
CFG_MAXCREDIT	Maximum outstanding N-chars	Covered in test cases T.IF.RBUF.1-4
CFG_SLOW_CE_SEL	External clock enable for 10MHz	Test cases checked with CFG_SLOW_CE_SEL=1
CFG_TICK_IN_KEEP	Discard time-codes when	Normal test cases are run with CFG_TICK_IN_KEEP=0. Special configuration is run with CFG_TICK_IN_KEEP=1

Table 3-1 Configuration analysis

3.3 CODEC configuration setup

A number of configuration command files setup the test-bench data rates. The final test-bench command file is a concatenation of a configuration set-up file and a command file.

The configuration setup files set the operating speeds of the interface clocks and the dividers for default initialisation and internal 10MHz reference. For example a test command file can be run on single data rate and double data rate implementation which require a 100MHz clock for double data rate and a 200MHz clock for single data rate.

The configuration files which are supported are shown below.

3.3.1 sysclk_30mhz_txclk_100mhz_slowclk_5mhz_rdclk_50mhz.cmd

System clock	= 30 MHz
Transmit clock	= 100 MHz
Slow clock	= 5 MHz
Read clock	= 50 MHz
CFG_SLOWCLK_SYSCLK	= 0x2
CFG_SLOWCLK_TXCLK	= 0x13

3.3.2 sysclk_30mhz_txclk_100mhz_slowclk_10mhz_rdclk_50mhz.cmd

System clock	= 30 MHz
Transmit clock	= 100 MHz
Slow clock	= 10 MHz
Read clock	= 50 MHz
CFG_SLOWCLK_SYSCLK	= 0x2
CFG_SLOWCLK_TXCLK	= 0x13

3.3.3 sysclk_30mhz_txclk_100mhz_slowce_rdclk_50mhz.cmd

System clock = 30 MHz

Transmit clock = 100 MHz

Slow clock = OFF

Read clock = 50 MHz

CFG_SLOWCLK_SYSCLK = 0x2

CFG_SLOWCLK_TXCLK = 0x13

CFG_SLOW_CE = 10 MHz

3.3.4 sysclk_30mhz_txclk_200mhz_slowclk_10mhz_rdclk_50mhz.cmd

System clock = 30 MHz

Transmit clock = 200 MHz

Slow clock = 10 MHz

Read clock = 50 MHz

CFG_SLOWCLK_SYSCLK = 0x2

CFG_SLOWCLK_TXCLK = 0x13

3.3.5 sysclk_30mhz_txclk_10mhz_rdclk_50mhz.cmd

System clock = 30 MHz

Transmit clock = 10 MHz

Slow clock = OFF

Read clock = 50 MHz

CFG_SLOWCLK_SYSCLK = 0x2

3.3.6 sysclk_30mhz_txclk_5mhz_rdclk_50mhz.cmd

System clock	= 30 MHz
Transmit clock	= 5 MHz
Slow clock	= OFF
Read clock	= 50 MHz
CFG_SLOWCLK_SYSCLK	= 0x2

3.3.7 sysclk_100mhz_slowclk_5mhz_rdclk_50mhz.cmd

System clock	= 100 MHz
Transmit clock	= OFF
Slow clock	= 5 MHz
Read clock	= 50 MHz
CFG_SLOWCLK_SYSCLK	= 0x9
CFG_SLOWCLK_TXCLK	= 0x13

3.3.8 sysclk_100mhz_slowclk_10mhz_rdclk_50mhz.cmd

System clock	= 100 MHz
Transmit clock	= OFF
Slow clock	= 10 MHz
Read clock	= 50 MHz
CFG_SLOWCLK_SYSCLK	= 0x9
CFG_SLOWCLK_TXCLK	= 0x13

3.3.9 sysclk_100mhz

System clock	= 100 MHz
Transmit clock	= OFF
Slow clock	= OFF
Read clock	= OFF
CFG_SLOWCLK_SYSCLK	= 0x9
CFG_SLOWCLK_TXCLK	= 0x13

3.3.10 sysclk_200mhz_slowclk_10mhz_rdclk_50mhz.cmd

System clock	= 200 MHz
Transmit clock	= OFF
Slow clock	= 10 MHz
Read clock	= 50 MHz
CFG_SLOWCLK_SYSCLK	= 0x13
CFG_SLOWCLK_TXCLK	= 0x13

3.3.11 sysclk_200mhz

System clock	= 100 MHz
Transmit clock	= OFF
Slow clock	= OFF
Read clock	= OFF
CFG_SLOWCLK_SYSCLK	= 0x13
CFG_SLOWCLK_TXCLK	= 0x13

3.3.12 sysclk_10mhz

System clock	= 10 MHz
Transmit clock	= OFF
Slow clock	= OFF
Read clock	= OFF

3.3.13 sysclk_5mhz

System clock	= 5 MHz
Transmit clock	= OFF
Slow clock	= OFF
Read clock	= OFF

3.4 Test cases

3.4.1 Configuration and interface test cases

The following sections define the configuration and interface test cases

3.4.1.1 Clocking and configuration test cases

Transmit interface test cases are defined in

Single data rate checking

```
codec/src/verif/cmd/if_clocking_check_sdr_rates_en.cmd
codec/src/verif/cmd/if_clocking_check_sdr_rates_div.cmd
```

Double data rate checking

```
codec/src/verif/cmd/if_clocking_check_ddr_rates_en.cmd
codec/src/verif/cmd/if_clocking_check_ddr_rates_div.cmd
```

10mbits only rate checking

```
codec/src/verif/cmd/if_clocking_check_10mbits_only.cmd
codec/src/verif/cmd/if_clocking_check_slowclk_only.cmd
```

State machine timeout and disconnect timeout period

```
codec/src/verif/cmd/if_clocking_check_timeouts.cmd
codec/src/verif/cmd/if_clocking_check_timeouts_10mbits_only.cmd
```

Test No.	Ref No.	Verification Strategy	T/A	Auto
T.IF.SYSCLK.1.a	IF.SYSCLK.1	Check data signalling rate of the signal data rate transmitter ties with SIG.1 for configurations; SYS_SLOWCLK_DIV, SYS_DIV and SYS_EN.	T	Yes
T.IF.SYSCLK.1.b	IF.SYSCLK.1	Check data signalling rate of the double data rate transmitter ties with SIG.1 for configurations: SYS_SLOWCLK_DIV, SYS_DIV and SYS_EN.	T	Yes
T.IF.SYSCLK.2.a	IF.SYSCLK.2	Check data signalling rate of the single data rate transmitter for configurations; SYS_DIV and SYS_EN.	T	Yes
T.IF.SYSCLK.2.b	IF.SYSCLK.2	Check data signalling rate of the double data rate transmitter for configurations; SYS_DIV and SYS_EN.	T	Yes
T.IF.SYSCLK.3	IF.SYSCLK.3	Check disconnect and state machine	T	Yes

		timeout periods when CFG_SLOW_CE is '0'.		
T.IF.SYSCLK.4	IF.SYSCLK.4	Check disconnect and state machine timeout periods when CFG_SLOW_CE is used.	T	Yes
T.IF.TXCLK.1.a	IF.TXCLK.1	Check data signalling rate of the signal data rate transmitter ties with SIG.1 for configurations; TXCLK_SLOWCLK_DIV, TXCLK_DIV and TXCLK_EN.	T	Yes
T.IF.TXCLK.1.b	IF.TXCLK.1	Check data signalling rate of the double data rate transmitter ties with SIG.1 for configurations: TXCLK_SLOWCLK_DIV, TXCLK_DIV and TXCLK_EN.	T	Yes
T.IF.TXCLK.2.a	IF.TXCLK.2	Check data signalling rate of the single data rate transmitter for configurations; TXCLK_DIV and TXCLK_EN.	T	Yes
T.IF.TXCLK.2.a	IF.TXCLK.2	Check data signalling rate of the double data rate transmitter for configurations; TXCLK_DIV and TXCLK_EN.	T	Yes
T.IF.SLOWCLK.1.a	IF.SLOWCLK.1	Check the default initialisation signalling rate in single data rate mode in configurations; SYS_SLOWCLK, SYS_SLOWCLK_DIV, TXCLK_SLOWCLK and TXCLK_SLOWCLK_DIV	T	Yes
T.IF.SLOWCLK.1.b	IF.SLOWCLK.1	Check the default initialisation signalling rate in double data rate mode in configurations; SYS_SLOWCLK, SYS_SLOWCLK_DIV, TXCLK_SLOWCLK and TXCLK_SLOWCLK_DIV	T	Yes
T.IF.SLOWCLK.2	IF.SLOWCLK.2	Check the disconnect and state machine timeouts for single and double data rate configurations for configurations; SYS_SLOWCLK, SYS_SLOWCLK_DIV, TXCLK_SLOWCLK and TXCLK_SLOWCLK_DIV	T	Yes

Table 3-2 Clock and timing test cases

3.4.1.2 Transmit interface test cases

Transmit interface test cases are defined in

```
codec/src/verif/cmd/if_transmit.cmd
codec/src/verif/cmd/if_transmit_10mbits_only.cmd
```

Test No.	Ref No.	Verification Strategy	T/A	Auto
T.IF.TBUF.1	IF.TBUF.1	UUT-link FLUSH_TX is asserted. Packets are written to FIFO. TB-link GotNchar is not asserted	A	Yes
T.IF.TBUF.2	IF.TBUF.2	UUT-link can send data	A	Yes
T.IF.TBUF.3.a	IF.TBUF.3	When CFG_TICK_IN_KEEP is '0' time-codes are discarded when the link is not running.	A	Yes
T.IF.TBUF.3.b	IF.TBUF.3	When CFG_TICK_IN_KEEP is '1' time-codes are not discarded when the link is running. New time-codes overwrite previously held values.	A	Yes

Table 3-3 Transmit interface test cases

3.4.1.3 Receive interface test cases

Status interface test cases are defined in

Receive interface

```
codec/src/verif/cmd/if_receive.cmd
codec/src/verif/cmd/if_receive_10mbits_only.cmd
```

Receive large credit space

```
codec/src/verif/cmd/if_receive_t.if.rbuf.4.cmd
```

Receive external read clock

```
codec/src/verif/cmd/if_receive_t.if.rbuf.8.cmd
```

Test No.	Ref No.	Verification Strategy	T/A	Auto
T.IF.RBUF.1	IF.RBUF.1	Check number of FCTs transmitted at start-up is one	T	Yes
T.IF.RBUF.2	IF.RBUF.2	Set CFG_MAXCREDIT value and Check number of FCTs transmitted at start-up is = CFG_MAXCREDIT/8	T	Yes
T.IF.RBUF.3	IF.RBUF.3	When CFG_MAXCREDIT is larger than buffer size then only buffer size credit shall be requested.	T	Yes
T.IF.RBUF.4	IF.RBUF.4	Check 7 FCTs transmitted at start-up. Set buffer size to 1K and transmit 1K data. Check receive success.	T	Yes
T.IF.RBUF.5	IF.RBUF.5	Reception of data characters successfully	T	Yes
T.IF.RBUF.6	IF.RBUF.6	Read data character successfully	T	Yes
T.IF.RBUF.7	IF.RBUF.7	Set buffer clock configuration and check data transfer and frequency	T	Yes
T.IF.RBUF.8	IF.RBUF.8	Keep empty packets transmitted	T	Yes
T.IF.RBUF.9	IF.RBUF.9	Check TICK_OUT is set when time-code is received.	T	Yes

Table 3-4 Transmit interface test cases

3.4.1.4 Status interface test cases

Status interface test cases are defined in

```
codec/src/verif/cmd/if_status.cmd
codec/src/verif/cmd/if_status_10mbits_only.cmd
```

Test No.	Ref No.	Verification Strategy	T/A	Auto
T.IF.STAT.1	IF.STAT.1	Status outputs are asserted correctly on status events	T	Yes

3.4.2 Signal level test cases

Signal level test cases are defined in

```
codec/src/verif/cmd/signal_level.cmd
codec/src/verif/cmd/signal_level_10mbits_only.cmd
```

Test No.	Ref No.	Verification Strategy	T/A	Auto
T.SIG.1	SIG.1	Link operation	T	Yes
T.SIG.2	SIG.2	Link operation	T	Yes
T.SIG.3	SIG.3	Link operation	T	Yes
T.SIG.4	SIG.4	Set WireInD and WireInS to 0 when both 1	T	Yes
T.SIG.5	SIG.5	Perform a reset of the UUT-link core for each DOUT/SOUT state, e.g. (0,0), (0,1) etc. Check for simultaneous transitions	T	Yes
T.SIG.6	SIG.6	Set input bit stream data rate to 2 Mbit/s. and check no disconnect occurs	T	Yes
T.SIG.7	SIG.7	Vary the data signalling rate from the minimum to the maximum while data transfer is performed.	T	Yes
T.SIG.8	SIG.8	Start-up the UUT-link after reset and check the data signalling rate.	T	Yes
T.SIG.9	SIG.9	Perform a disconnection of the UUT-link and check the data signalling rate.	T	Yes
T.SIG.10	SIG.10	Start-up the UUT-link after reset and check the data signalling rate once the run state has been entered.	T	Yes

Table 3-5 Signal level test cases

3.4.3 Character level test cases

Character level test cases are defined in

```
codec/src/verif/cmd/char_level.cmd
codec/src/verif/cmd/char_level_10mbits_only.cmd
```

Test No.	Ref No.	Verification Strategy	T/A	Auto
T.CHA.1.a	CHA.1	UUT-link transmits a series of data characters after reset. The TB-link receives data characters without parity error.	T	Yes
T.CHA.1.b	CHA.1	TB-link sends a series of data characters after reset. The UUT-link receives data characters without parity error	T	Yes
T.CHA.2.a	CHA.2	UUT-link send data characters with the LSB set to one after reset. TB-link receives data character with correct bit alignment.	T	Yes
T.CHA.2.b	CHA.2	TB-link sends data characters with the LSB set to one. UUT-link receives data characters with correct bit alignment.	T	Yes
T.CHA.3.a	CHA.3	UUT-link sends control characters on start-up and normal operation. TB-link starts up and receives control characters.	T	Yes
T.CHA.3.b	CHA.3	TB-link sends control characters on start-up and normal operation. UUT-link starts up and receives control characters.	T	Yes
T.CHA.4.a	CHA.4	UUT-link transmits FCT characters at start-up as part of NULL character and as flow control. TB-link receives NULLs and FCTs and starts up.	T	Yes
T.CHA.4.b	CHA.4	TB-link transmits FCT at start-up as part of NULL character and flow control. UUT-link receives FCT and NULL and starts up	T	Yes
T.CHA.5.a	CHA.5	UUT-link send data characters ending with an EOP character. TB-link receives data characters followed by EOP.	T	Yes
T.CHA.5.a	CHA.5	TB-link send data characters ending with an EOP character. UUT-link receives data characters followed by EOP.	T	Yes
T.CHA.6.a	CHA.6	UUT-link send data characters ending with an EEP character. TB-link receives data characters followed by EEP.	T	Yes
T.CHA.6.b	CHA.6	TB-link send data characters ending with an EEP character. UUT-link receives data characters followed by EEP	T	Yes
T.CHA.7.a	CHA.7	UUT-link transmits NULL characters at start-up which contain the ESC pattern. TB-link detects NULL characters and perform link start-up.	T	Yes
T.CHA.7.b	CHA.7	TB-link transmits NULL characters at start-up. UUT-link detects NULL characters and performs link start-up.	T	Yes
T.CHA.8.a	CHA.8	UUT-link transmits NULL characters at start-up. TB-link receives NULL pattern and performs start-up without parity error.	T	Yes
T.CHA.8.b	CHA.8	TB-link transmits NULL characters at start-up. UUT-link receives NULL pattern and performs start-up without parity error.	T	Yes
T.CHA.8.c	CHA.8	TB-link transmits NULL with incorrect parity bit. Parity Error is detected at UUT-link	T	Yes

T.CHA.9.a	CHA.9	UUT-link transmits series of time-codes. TB-link receives timecode characters without parity error.	T	Yes
T.CHA.9.b	CHA.9	TB-link transmits series of time-codes. UUT-link receives timecode characters without parity error.	T	Yes
T.CHA.9.c	CHA.9	TB-link transmits time-codes with incorrect parity bit. UUT-link detects parity error.	T	Yes
T.CHA.10.a	CHA.10	TB-link transmits ESC followed by EOP. UUT-link detects escape error and disconnects	T	Yes
T.CHA.10.b	CHA.10	TB-link transmits ESC followed by EEP. UUT-link detects escape error and disconnects	T	Yes
T.CHA.10.c	CHA.10	TB-link transmits ESC followed by ESC. UUT-link detects escape error and disconnects	T	Yes
T.CHA.11.a	CHA.11	UUT-link transmits data and control characters. TB-link does not report parity error.	T	Yes
T.CHA.11.b	CHA.11	TB-link transmits data and control characters with correct parity bit. UUT-link does not report parity error.	T	Yes
T.CHA.11.c	CHA.11	TB-link transmits characters with incorrect parity bit set. UUT-link reports parity error and disconnects.	T	Yes
T.CHA.12.a	CHA.12	TB-link starts up initially after reset with NULL pattern. UUT-link detects first NULL pattern and starts up. successfully.	T	Yes
T.CHA.12.b	CHA.12	TB-link starts up initially after reset with incorrect first NULL pattern. UUT-link does not start-up.	T	Yes
T.CHA.12.c	CHA.12	UUT-link starts up initially after reset and transmits correct NULL pattern. TB-link starts up link connection	T	Yes
T.CHA.13.a	CHA.13	UUT-link transmits data, EOP and EEP characters from host using control flag protocol. TB-link receives data , EOP and EEP characters.	T	Yes
T.CHA.13.b	CHA.13	TB-link transmits data, EOP and EEP. UUT-link receives data , EOP and EEP using correct protocol	T	Yes
T.CHA.14	CHA.14	Transmit bit ordering is preserved for flags	T	Yes
T.CHA.15	CHA.15	UUT-link transmits timecode. TB-link receives timecode	T	Yes
T.CHA.16.a	CHA.16	TB-link initiates start-up and transmits timecode. UUT-link receives timecode in Run state	T	Yes
T.CHA.16.b	CHA.16	TB-link transmits timecode before initiating start-up. UUT-link TICK_OUT is not asserted.	T	Yes

3.4.4 Exchange level test cases

Character level test cases are defined in

```
codec/src/verif/cmd/exch_level.cmd
codec/src/verif/cmd/exch_level_10mbits_only.cmd
```

Test No.	Ref No.	Verification Strategy	T/A	Auto
T.EXC.1	EXC.1	TB-link sends data character to UUT-link with	T	Yes

		incorrect parity bit. UUT-link does not write character to receiver buffer.		
T.EXC.2	EXC.2	Check state machine has states ErrorReset, ErrorWait, Ready, Started, Connecting and Run	T	Yes
T.EXC.FCT.1	EXC.FCT.1	Check Spwr-TB FCT interaction	T	Yes
T.EXC.FCT.2	EXC.FCT.2	No data characters transmitted before first FCT is received	T	Yes
T.EXC.FCT.3	EXC.FCT.3	After TB-link link disconnect check transmitter HAS_CREDIT is low.	T	Yes
T.EXC.FCT.4	EXC.FCT.4	TB-link transmit FCTs. UUT-link reports credit error.	T	Yes
T.EXC.FCT.5	EXC.FCT.5	Check Spwr-TB FCT interaction.	T	Yes
T.EXC.FCT.6	EXC.FCT.6	Check Spwr-TB FCT interaction.	T	Yes
T.EXC.FCT.7	EXC.FCT.7	TB-link receives UUT-link FCTs on start-up. FCTs are received during normal data flow. This shall also include empty packet test which must be accounted for by the	T	Yes
T.EXC.FCT.8	EXC.FCT.8	Check ordering	T	Yes
T.EXC.ERS.1	EXC.ERS.1	System reset is performed. UUT-link state is ErrorReset	T	Yes
T.EXC.ERS.2	EXC.ERS.2	UUT-link does not detect NULLs from TB-link. TB-link GotNULL is not asserted	T	Yes
T.EXC.ERS.3	EXC.ERS.3	After 6.4us delay from reset check UUT-link state.	T	Yes
T.EXC.ERW.1	EXC.ERW.1	Code analysis.	A	-
T.EXC.ERW.2	EXC.ERW.2	Check receiver receive null and transmitter no nulls	T	Yes
T.EXC.ERW.3	EXC.ERW.3	TB-link sends NULLs. GotNULL is asserted	T	Yes
T.EXC.ERW.4	EXC.ERW.4	After 6.4us + 12.8us from reset check UUT-link state	T	Yes
T.EXC.ERW.5.a	EXC.ERW.5	TB-link cause disconnect error . Check UUT-link state is ErrorReset.	T	Yes
T.EXC.ERW.5.b	EXC.ERW.5	TB-link parity error.	T	Yes
T.EXC.ERW.5.c	EXC.ERW.5	TB-link escape error.	T	Yes
T.EXC.ERW.5.d	EXC.ERW.5	TB-link send FCT	T	Yes
T.EXC.ERW.5.e	EXC.ERW.5	TB-link send N-Char	T	Yes
T.EXC.ERW.5.f	EXC.ERW.5	TB-link send Timecode	T	Yes
T.EXC.RDY.1	EXC.RDY.1	Code analysis.	A	-
T.EXC.RDY.2	EXC.RDY.2	Check Transmitter disabled and receiver enabled in ready.	T	Yes
T.EXC.RDY.3	EXC.RDY.3	TB-link sends NULLs. GotNULL is asserted	T	Yes
T.EXC.RDY.4.a	EXC.RDY.4	Assert UUT-link **LINK_START** and check UUT-link state is Started.	T	Yes
T.EXC.RDY.4.b	EXC.RDY.4	Assert UUT-link **AUTOSTART** and TB-link send NULL. Check UUT-link state is Started	T	Yes
T.EXC.RDY.4.c	EXC.RDY.4	Asserted UUT-link **LINK_DISABLE**. Check UUT-link does not enter Started on	T	Yes

		`LINK_START` or AUTO_START		
T.EXC.RDY.5.a	EXC.RDY.5	TB-link cause disconnect error and check UUT-link state is ErrorReset	T	Yes
T.EXC.RDY.5.b	EXC.RDY.5	TB-link parity error.	T	Yes
T.EXC.RDY.5.c	EXC.RDY.5	TB-link escape error.	T	Yes
T.EXC.RDY.5.d	EXC.RDY.5	TB-link send FCT	T	Yes
T.EXC.RDY.5.e	EXC.RDY.5	TB-link send N-Char	T	Yes
T.EXC.RDY.5.f	EXC.RDY.5	TB-link send Timecode	T	Yes
T.EXC.STA.1	EXC.STA.1	TB-link receives NULLs	T	Yes
T.EXC.STA.2	EXC.STA.2	TB-link sends NULLs. GotNULL is asserted	T	Yes
T.EXC.STA.3	EXC.STA.3	UUT-link receive NULL and check UUT-link state is Connecting.	T	Yes
T.EXC.STA.4	EXC.STA.4	UUT-link move from ErrorWait -> Ready -> Started. TB-link detects NULL then FCTs	T	Yes
T.EXC.STA.5	EXC.STA.5	TB-link cause disconnect error and check UUT-link state is ErrorReset	T	Yes
T.EXC.STA.6	EXC.STA.6	TB-link does not send NULLs for 12.8 us. Check UUT-link state is ErrorReset	T	Yes
T.EXC.CON.1	EXC.CON.1	TB-link receives NULLs and FCTs	T	Yes
T.EXC.CON.2	EXC.CON.2	TB-link sends FCT. UUT-link check for state Run	T	Yes
T.EXC.CON.3.a	EXC.CON.3	TB-link causes link disconnect error. Check UUT-link state is ErrorReset	T	Yes
T.EXC.CON.3.b	EXC.CON.3	TB-link parity error	T	Yes
T.EXC.CON.3.c	EXC.CON.3	TB-link credit error	T	Yes
T.EXC.CON.3.d	EXC.CON.3	TB-link send incorrect character	T	Yes
T.EXC.CON.3.e	EXC.CON.3	TB-link send incorrect character	T	Yes
T.EXC.CON.4	EXC.CON.4	TB-link does not send FCTs for 12.8 us. Check UUT-link state is ErrorReset	T	Yes
T.EXC.RUN.1.a	EXC.RUN.1	TB-link causes disconnect error. Check UUT-link state is ErrorReset	T	Yes
T.EXC.RUN.1.b	EXC.RUN.1	TB-link parity error	T	Yes
T.EXC.RUN.1.c	EXC.RUN.1	TB-link escape error	T	Yes
T.EXC.RUN.1.d	EXC.RUN.1	TB-link credit error	T	Yes
T.EXC.RUN.1.e	EXC.RUN.1	UUT-link LINK_DISABLE. check UUT-link state is ErrorReset	T	Yes
T.EXC.ERR.1.a	EXC.ERR.1	TB-link inserts parity error, data characters and time-codes are not output from UUT after the parity error.	T	Yes
T.EXC.ERR.1.b	EXC.ERR.1	TB-link inserts escape error, data characters and time-codes are not output from the UUT after the escape error occurs.	T	Yes
T.EXC.ERR.1.c	EXC.ERR.1	TB-link inserts character sequence error, data characters and time-codes are not output from the UUT after the escape error occurs.	T	Yes
T.EXC.ERR.2	EXC.ERR.2	TB-link Disconnect error in run state	T	Yes
T.EXC.ERR.3	EXC.ERR.3	TB-link Parity error in run state	T	Yes
T.EXC.ERR.4	EXC.ERR.4	TB-link Escape error in run state	T	Yes

T.EXC.ERR.5	EXC.ERR.5	TB-link credit error in run state. This test shall also include credit errors generated by empty packets which must be taken into account by the receive credit counter.	T	Yes
T.EXC.ERR.6	EXC.ERR.6	TB-link sends empty packet. UUT-link discards second EOP, EEP	T	Yes
T.EXC.TIME.1	8.11.2	TB-link causes disconnect error. UUT-link disconnect error is not reported between 727ns and 1000ns period.	T	Yes

Table 3-6 Exchange level test cases

3.4.5 Network level test cases

Network level test cases are defined in

```
codec/src/verif/cmd/network_level.cmd
codec/src/verif/cmd/network_level_10mbits_only.cmd
```

Test No.	Reference No.	Verification Strategy	T/A	Auto
T.NET.REC.1	NET.REC.1	TB-link causes link error. UUT-link last packet is terminated with EEP.	T	Yes
T.NET.REC.2	NET.REC.2	TB-link causes link error. UUT-link spills head of transmitter packet	T	Yes

3.4.6 Performance test cases

Performance test cases are defined in

```
codec/src/verif/cmd/perf.cmd
```

Test No.	Reference No.	Verification Strategy	T/A	Auto
T.PERF.1	PERF.1	Check from first byte of packet received to last byte of packet received.	T	Yes

3.5 Test-bench Run Configurations

The testbench test cases check the operation of the SpaceWire CODEC is within specification for a given SpaceWire CODEC. The test-bench run configurations set the configuration options of the SpaceWire CODEC so different configurations can be checked. For each configuration a run of the test-bench is performed using the specified files. Verification is performed using the verification command files concatenated together "on the fly" to achieve specific spwrlink_pkg.vhd configurations

Using the TCL language loop functionality the test-bench run_verif.tcl script can run almost all the CODEC configurations. The main configuration loop checks each configuration of the pipeline, double data rate, slow clock 10MHz option and transmit bit clock confiurations.

```
# run configurations for pipelining, ddrout, bitclock and
slowclk_10mhz
set bitclkcfg { SYS_DEFAULT SYS_SLOWCLK SYS_SLOWCLK_DIV SYS_DIV
SYS_EN TXCLK_DEFAULT TXCLK_SLOWCLK TXCLK_SLOWCLK_DIV TXCLK_DIV
TXCLK_EN }

# pipelined configuration loop
for {set pipeline 0} {$pipeline < 2} {incr pipeline} {
    # ddr configuration loop
    for {set ddrout 0} {$ddrout < 2} {incr ddrout} {
        # slowclk_10mhz configuration loop
        for {set slowclk_10mhz 0} {$slowclk_10mhz < 2} {incr
slowclk_10mhz} {
            # bitclock configuration loop
            foreach bitclk $bitclkcfg {
               # Generate working spwrlink_pkg from config
               # Generate working command file dependent on config
               # Run the testbench
            }
        }
    }
}
```

To run each configuration the script generates the working spwrlink_pkg.vhd file and concatenates the correct command files together to generate the working command file dependent on the current configuration.

Specific configurations of the SpaceWire CODEC which are not generated automatically are listed below.

3.5.1 Configuration 81 – Non synchronous read clock and large receive buffer size

This configuration checks the non synchronous read clock configuration SpaceWire CODEC.

```
CFG_PIPELINE            0
CFG_DDROUT              0
CFG_BITCLK              TXCLK_EN
CFG_SLOWCLK_10MHZ       0
CFG_SYNCRDCLK           0
CFG_DISCARD_EMPTY_PKT   1
CFG_RXBUF_ADDRLEN       10
CFG_RATE_NUMBITS        6
CFG_SLOW_CE_SEL         0
CFG_TICK_IN_KEEP        0
```

The command is run using the concatenation of the following files:

```
codec/src/verif/cmd/sysclk_30mhz_txclk_200mhz_slowclk_10mhz_rdclk_
50mhz.cmd
codec/src/verif/cmd/if_receive_t.if.rbuf.7.cmd
```

3.5.2 Configuration 82 – Keep empty packets

This configuration checks the not discard empty packet configuration SpaceWire CODEC.

```
CFG_PIPELINE            0
CFG_DDROUT              0
CFG_BITCLK              TXCLK_EN
CFG_SLOWCLK_10MHZ       0
CFG_SYNCRDCLK           1
CFG_DISCARD_EMPTY_PKT   0
CFG_RXBUF_ADDRLEN       5
CFG_RATE_NUMBITS        6
CFG_SLOW_CE_SEL         0
CFG_TICK_IN_KEEP        0
```

The command is run using the concatenation of the following files:

```
codec/src/verif/cmd/sysclk_200mhzclk_10mhz_rdclk_50mhz.cmd
codec/src/verif/cmd/if_receive_t.if.rbuf.8.cmd
```

3.5.3 Configuration 83 – External Slow clock enable

This configuration checks the not discard empty packet configuration SpaceWire CODEC.

```
CFG_PIPELINE            0
CFG_DDROUT              1
CFG_BITCLK              TXCLK_EN
CFG_SLOWCLK_10MHZ       0
CFG_SYNCRDCLK           1
CFG_DISCARD_EMPTY_PKT   1
CFG_RXBUF_ADDRLEN       5
CFG_RATE_NUMBITS        6
CFG_SLOW_CE_SEL         1
CFG_TICK_IN_KEEP        0
```

The command is run using the concatenation of the following files:

```
codec/src/verif/cmd/sysclk_30mhz_txclk_200mhz_slowce_rdclk_50mhz.c
md
codec/src/verif/cmd/signal_level_t.sig.5
codec/src/verif/cmd/if_clocking_check_ddr_rates_en.cmd
codec/src/verif/cmd/if_clocking_check_timeouts.cmd
codec/src/verif/cmd/if_transmit.cmd
codec/src/verif/cmd/if_receive.cmd
codec/src/verif/cmd/if_status.cmd
codec/src/verif/cmd/signal_level.cmd
codec/src/verif/cmd/char_level.cmd
codec/src/verif/cmd/exch_level.cmd
codec/src/verif/cmd/network_level.cmd
codec/src/verif/cmd/perf.cmd
```

3.5.4 Configuration 84 – Keep time-codes when link is not running (CFG_TICK_IN_KEEP=1)

This configuration checks the not discard empty packet configuration SpaceWire CODEC.

```
CFG_PIPELINE            0
CFG_DDROUT              0
CFG_BITCLK              TXCLK_EN
CFG_SLOWCLK_10MHZ       0
CFG_SYNCRDCLK           1
CFG_DISCARD_EMPTY_PKT   0
CFG_RXBUF_ADDRLEN       5
CFG_RATE_NUMBITS        6
CFG_SLOW_CE_SEL         0
CFG_TICK_IN_KEEP        1
```

The command is run using the concatenation of the following files:

```
codec/src/verif/cmd/sysclk_30mhz_txclk_200mhz_slowce_rdclk_50mhz.c
md
codec/src/verif/cmd/if_transmit.tbuf.3.b.cmd
```

3.6 Conformance Summary

Section 12.2.4 of the SpaceWire standard [AD1] lists the relevant clauses and sub-clauses which are required for the SpaceWire link interface.

The following table defines each relevant clause and sub clause. Tests cases which are associated with the clause are defined in the table.

Note: Clauses and subclasses which are not relevant or are covered in other test cases are printed in *italics*.

Note: Test Numbers which have prefixes a, b, c etc. are shortened for clarity

3.6.1 Physical Level

Ref No	Test No	Reference Summary	Notes
5.3	-	*SpaceWire Connectors*	*n/a*
5.5	-	*PCB Tracks usage*	*n/a*

Table 3-7 Physical level conformance

3.6.2 Signal Level

Ref No	Test No	Reference Summary	Notes
6.1	-	*Low Voltage Differential signalling shall be used*	*n/a*
6.2	-	*Failsafe operation of LVDS*	*n/a*
6.3.1.a	T.SIG.1	Data Strobe Encoding shall be used.	
6.3.1.b	T.SIG.2, T.SIG.3	Data value shall follow bit stream value. Strobe shall change when Data Changes	
6.3.2.a	T.SIG.4	Receiver shall be tolerant to simultaneous transitions on **DIN** and **SIN**	
6.3.2.b	T.SIG.5	No simultaneous transitions shall occur on **DOUT** and **SOUT**.	
6.4	-	*LVDS shall be used for data and strobe signals*	*n/a*
6.5	-	*A SpaceWire link shall comprise of two pairs of differential signals in both directions*	*n/a*
6.6.1	T.SIG.6	Minimum data signalling rate.	
6.6.2	-	*Maximum data signalling rate is dependant on signal skew and jitter*	*n/a*
6.6.3	T.SIG.7	SpaceWire link shall operate at any data signalling rate from the maximum to the minimum. The link in one direction can operate at a different data signalling rate than the link in the other direction	
6.6.4	-	*Effects of skew and jitter*	*n/a*
6.6.5	T.SIG.8	Initial data signalling rate after reset	
6.6.5	T.SIG.9	Initial data signalling rate after link disconnection	

6.6.6	T.SIG.10	The transmitter data signalling rate shall not be changed until the state machine moves to the Run state	

Table 3-8 Signal level conformance

3.6.3 Character level

Ref No	Test No	Reference Summary	Notes
7.1	-	*General Description of character level*	*n/a*
7.2	T.CHA.1, T.CHA.2	Data character bit definition. Data bits shall be transmitted LSB to MSB	
7.3.a	T.CHA.3, T.CHA.4, T.CHA.5, T.CHA.6, T.CHA.7	Control character bit sequence definition. Control character code fields :- FCT -> “00” EOP -> “01” EEP -> “10” ESC -> “11”	Test cases cover all control character transmission and reception
7.3.b	T.CHA.8	NULL character bit sequence definition. NULL character middle parity bit usage	
7.3.c	T.CHA.9	Timecode character bit sequence definition, Timecode character middle parity bit usage.	
7.3.d	-	*Timecode character shall be least significant six bytes of a timecode character transmitted. The two most significant bits shall be the timecode control-flag*	*UUT-link accepts time-codes as eight bit characters on the input signal* ***TIME_IN****. External host controls timecode control bit fields. Timecode transmission is covered in T.CHA.9*
7.3.e	T.CHA.10	An Escape character followed by escape, EOP or EEP shall be noted as an escape error	
7.4	T.CHA.11	The parity bit shall be set to produce odd parity	
7.5	T.CHA.12	Data-Strobe shall be set to zero after reset. Data-Strobe bit pattern after transmitter enabled.	
7.6	T.CHA.13	N-char host interface control bit usage.	
7.7.a	T.CHA.14	The timecode interface shall comprise two signals **TICK_IN** and **TICK_OUT**, six bit timecode input and output signals and two bit timecode control-flag input and output.	Time-codes are accepted as an eight bit character. Bits 5-0 are timecode and bits 7-6 are the control-flag
7.7.b	T.CHA.15	**TICK_IN** asserted shall cause a timecode to be transmitted in the Run state.	
7.7.c	T.CHA.16	**TICK_OUT** shall be asserted when a	

		timecode is received and the transmitter is in the Run state.	
7.7.d	-	*Only one node in a system shall have an active* ***TICK_IN*** *signal.*	*The time master TICK_IN controller is a system level issue*
7.7.e	-	*All other nodes shall keep TICK_IN de-asserted.*	*As above.*
7.7.f	-	*A six bit time counter shall be provided from the link receiver to the local time counter.*	*Covered in T.CHA.14*
7.7.g	-	*A six bit time input shall be provided to the transmitted from the local time counter*	*Covered in T.CHA.14*

Table 3-9 Character level conformance

3.6.4 Exchange level

Ref No	Test No	Reference Summary	Notes
8.1	-	*General Description of exchange level*	*n/a*
8.2.1	T.EXC.1	N-Chars are the only characters which are passed to the packet level	
8.2.2.a	T.EXC.2	Only N-Chars shall be passed from the host interface to the link.	
8.2.2.b	T.EXC.3	Received characters shall not be acted upon until the parity bit is checked.	

Table 3-10 Exchange level conformance

3.6.4.1 FCT usage

Ref No	Test No	Reference Summary	Notes
8.3.a	-	*Flow control tokens shall be transferred over the link to control data transfers.*	*Overview of FCT usage*
8.3.b	-	*FCT sent to indicate there is room for eight more data N-Chars*	*Covered in 8.3.c*
8.3.c	T.EXC.FCT.1	For each FCT sent space shall be reserved for eight more N-Chars	
8.3.d	T.EXC.FCT.2	Transmitter credit counter increment by eight for each FCT received	
8.3.e	T.EXC.FCT.3	In state ErrorReset the transmitter credit count shall be set to zero.	
8.3.f	T.EXC.FCT.4	Transmitter credit error when credit counter is greater than 56	
8.3.g	-	*Maximum amount of credit is 56*	*Covered in 8.3.f*
8.3.h	-	*On reset the number of FCTs to send shall be set to the size of the receiver credit counter*	*Covered in T.EXC.FCT.2*

8.3.i	T.EXC.FCT.5	Receiver expected characters credit counter shall be incremented by eight when a FCT is transmitted and decremented by one when an N-Char is written to the buffer	
8.3.j	T.EXC.FCT.6	At reset the expected characters count shall be set to zero.	
8.3.k	-	*Expected characters credit counter shall hold a maximum of 56 characters.*	*Covered in T.EXC.FCT.4*
8.3.l	T.EXC.FCT.7	An FCT shall be transmitted when there is room in the outstanding credit counter and in the buffer space count.	
8.3.m	T.EXC.FCT.8	FCTs transmitted when no Time-codes. N-Chars transmitted when no FCTs or Time-codes. NULLs transmitted when no FCTs, Time-codes or N-Chars.	
8.3.n	T.EXC.FCT.8	Transmission priority	

Table 3-11 Exchange level FCT conformance

3.6.4.2 Encoder decoder block diagram

Ref No	Test No	Reference Summary	Notes
8.4.1	-	*Encoder decoder block diagram general informative*	*n/a*
8.4.2	-	*Transmitter functions and host interface*	*Informative*
8.4.3	-	*Transmitter clock description*	*Covered in 6.6*
8.4.4	-	*Receiver functions and states.*	*Informative*
8.4.5	-	*Receiver clock recovery functions*	*Informative*
8.4.6	-	*State Machine functions*	*Informative*
8.4.7	-	*Timer functions*	*Informative*
8.4.8	-	*Receiver buffer management*	*Informative*
8.4.9	-	*Receiver FIFO buffering*	*Informative*

Table 3-12 Encoder decoder block diagram conformance

3.6.4.3 State machine

Ref No	Test No	Reference Summary	Notes
8.5.1	-	*State machine general description*	*n/a*
8.5.2.1	-	*General description of state machine states*	*n/a*

Table 3-13 State machine conformance

3.6.4.3.1 In State ErrorReset

Ref No	Test No	Reference Summary	Notes
8.5.2.2.a	T.EXC.ERS.1	ErrorReset shall be entered on system reset or link operation error	
8.5.2.2.b	T.EXC.ERS.2	Transmitter and receiver are disabled in state ErrorReset	
8.5.2.2.c	T.EXC.ERS.3	After 6.4us ErrorReset -> ErrorWait	
8.5.2.2.d	-	*State machine shall remain in ErrorReset when reset is asserted*	*Covered in 8.5.2.2.a*

Table 3-14 State ErrorReset conformance

3.6.4.3.2 In State ErrorWait

Ref No	Test No	Reference Summary	Notes
8.5.2.3.a	T.EXC.ERW.1	ErrorWait shall only be entered from ErrorReset	
8.5.2.3.b	T.EXC.ERW.2	The transmitter shall be disabled and the Receiver shall be enabled	
8.5.2.3.c	T.EXC.ERW.3	If a NULL is received then GotNULL shall be asserted	
8.5.2.3.d	T.EXC.ERW.4	The ErrorWait state shall be left unconditionally after the 12.8us timeout.	
8.5.2.3.e	T.EXC.ERW.5	On receiver error or incorrect character received then move to state ErrorReset	

Table 3-15 State ErrorWait conformance

3.6.4.3.3 In State Ready

Ref No	Test No	Reference Summary	Notes
8.5.2.4.a	T.EXC.RDY.1	State Ready shall be entered only from the ErrorWait state.	
8.5.2.4.b	T.EXC.RDY.2	The transmitter shall be disabled and the Receiver shall be enabled	
8.5.2.4.c	T.EXC.RDY.3	If a NULL is received then GOT_NULL shall be asserted	
8.5.2.4.d	T.EXC.RDY.4	Remain in state Ready until LinkEnabled is asserted then move to state Started	
8.5.2.4.e	T.EXC.RDY.5	On receiver error or incorrect character received then move to state ErrorReset.	

Table 3-16 State Ready conformance

3.6.4.3.4 In State Started

Ref No	Test No	Reference Summary	Notes
8.5.2.5.a	-	*State Started shall be entered from state Ready when LinkEnabled is asserted.*	*Covered in T.EXC.RDY.2*
8.5.2.5.b	-	*When started is entered the 12.8us timer shall*	*Covered in*

		be started.	*T.EXC.STA.6*
8.5.2.5.c	T.EXC.STA.1	In the started state the transmitter shall send NULLs. The receiver shall be enabled	
8.5.2.5.d	T.EXC.STA.2	If a NULL is received then GOT_NULL shall be asserted.	
8.5.2.5.e	T.EXC.STA.3	If GOT_NULL is asserted then the state machine shall move to state Connecting.	
8.5.2.5.f	T.EXC.STA.4	At least one NULL shall be requested for transmission before moving to state connecting.	
8.5.2.5.g	T.EXC.STA.5	On receiver error the state machine shall move to state ErrorReset	
8.5.2.5.h	T.EXC.STA.6	On 12.8us timeout the state machine shall move to state ErrorReset.	

Table 3-17 State Started conformance

3.6.4.3.5 In State Connecting

Ref No	Test No	Reference Summary	Notes
8.5.2.6.a	-	*The connecting state shall be entered from the started state when GOT_NULL is asserted.*	*Covered in T.EXC.STA.3*
8.5.2.6.b	-	*The 12.8us timer shall be started when entering Connecting.*	*Covered in T.EXC.CON.4*
8.5.2.6.c	T.EXC.CON.1	The transmitter shall be enabled to send NULLs and FCTs.	
8.5.2.6.d	T.EXC.CON.2	If an FCT is received then state machine shall move to state Run.	
8.5.2.6.e	T.EXC.CON.3	On receiver error or incorrect character error the state machine shall move from state Connecting to state Run.	
8.5.2.6.f	T.EXC.CON.4	On 12.8us timeout the state machine shall move to state ErrorReset.	

Table 3-18 State Connecting conformance

3.6.4.3.6 In State Run

Ref No	Test No	Reference Summary	Notes
8.5.2.7.a	-	*Run state shall be entered from connecting when GotFCT is asserted.*	*Covered in T.EXC.CON.2*
8.5.2.7.b	T.EXC.RUN.1	On LinkDisable or receiver error or credit error the state machine shall move to state ErrorReset	

Table 3-19 State Run conformance

3.6.4.4 Others

Ref No	Test No	Reference Summary	Notes
8.5.3.3	-	*State machine transitions.*	*Covered in*

			T.EXC. state test cases.*
8.6	-	*AutoStart definition*	*Covered in T.EXC.RDY.4*
8.7	-	*Link Initialisation*	*Informative*
8.8	-	*Normal Operation*	*Informative*
8.10	-	*Exception conditions*	*Informative*

Table 3-20 Exchange level others conformance

3.6.4.5 Error Detection

Ref No	Test No	Reference Summary	Notes
8.9.1	T.EXC.ERR.1	On a receive error character synchronization and flow-control status shall cease to be valid.	
8.9.2.1	-	*Disconnect Error*	*Covered in T.EXC. states.*
8.9.2.1.f	T.EXC.ERR.2	If disconnect error is detected in the Run state then the error shall be reported to the Network level	
8.9.2.2	-	*Parity Error*	*Covered in T.CHA.11*
8.9.2.2.c	T.EXC.ERR.3	If parity error is detected in the Run state then the error shall be reported to the Network level	
8.9.2.3.	-	*Escape Error*	*Covered in T.CHA.10 states.*
8.9.2.3.c	T.EXC.ERR.4	If escape error is detected in the Run state then the error shall be reported to the Network level	
8.9.2.4	-	*Credit Error*	*Covered in T.EXC.RUN.1*
8.9.2.4.c	T.EXC.ERR.5	If credit errr is detected in the Run state then the error shall be reported to the Network level	
8.9.2.5	-	*Character sequence error. Character sequence error shall not be reported to the network level.*	*Covered in T.EXC. states.*
8.9.3.	T.EXC.ERR.6	Empty packets may be discarded by the receiver.	
8.9.4	-	*Exchange of silence procedure.*	*Covered in T.EXC. states.*
8.9.5	-	*Reporting errors to network levels*	*Covered above.*

Table 3-21 Exchange level error conformance

3.6.4.6 Time-codes

Ref No	Test No	Reference Summary	Notes
8.12	-	*Timecode counter shall be implemented in SpaceWire node or*	*The purpose of the SpaceWire link interface is to accept time-*

		router	*codes for transmission using TICK_IN and TIME_IN and to output time-codes received using TICK_OUT and TIME_OUT. External logic shall be used to implement the Timecode counter. TICK_IN and TICK_OUT usage is covered in T.CHA.14, 15, 16.*

Table 3-22 Time-codes conformance

3.6.4.7 Timings

Ref No	Test No	Reference Summary	Notes
8.11.1	-	*D and S reset timing*	*Covered in T.SIG.5*
8.11.2	T.EXC.TIM.1	The disconnect timeout period of 850ns nominal shall be from 727ns to 1000ns.	-
8.11.3.a	-	*The 6.4us nominal timeout period shall be from 5.82us to 7.22us.*	*Covered in T.EXC.ERS.3*
8.11.3.b	-	*The 12.8us nominal timeout period shall be from 11.64us to 14.33us.*	*Covered in T.EXC.ERW.4*

Table 3-23 Link timings conformance

3.6.5 Packet Level

Ref No	Test No	Reference Summary	Notes
9.	-	*Packet format description*	*n/a*

Table 3-24 Packet level conformance

3.6.6 Network Level

Ref No	Test No	Reference Summary	Notes
10.4	-	*Spacewire node shall comprise one or more SpaceWire link interfaces. SpaceWire nodes shall accept a stream of packets.*	*n/a*
10.6.1	-	*Types of packet level errors which occur.*	*n/a*
10.6.2	*T.NET.REC.1, T.NET.REC.2*	*On Link error the tail of the transmitter packet shall be discarded and an EEP shall be added to the tail of the currently received packet.*	
10.6.3	-	*Reception of packet with EEP.*	*EEP is received as N-Char and passed on to packet level by receiver.*

10.6.4	-	*Invalid destination address*	*n/a*
10.6.4.4	-	*An EOP,EEP received immediately after an EOP, EEP represents an empty packet. The second EOP, EEP shall be discarded*	*Covered in T.EXC.ERR.5*

Table 3-25 Network level conformance

4 VERIFICATION PROCEDURE

The test-bench is designed to run in the Mentor Graphics Modelsim simulator. The simulator commands can be adapted to run on other simulators. The main simulation flow is; create the verification directory and run the simulation script. The simulation script runs through the SpaceWire CODEC configurations one at a time by; generating the spwrlink_pkg.vhd configuration, generating the working command file by concatenating the applicable command files together, compiling the SpaceWire CODEC configuration, RTL and verification files and running the verification.

Verification is performed using the verification command files concatenated together "on the fly" to achieve specific spwrlink_pkg.vhd configurations. A specific SpaceWire link configuration is also generated "on the fly" and compiled with the RTL and verification source code before the verification run is performed. The working command and working spwrlink_pkg.vhd files are generated using simple TCL procedures which can be run within the Modelsim environment.

4.1 Verification Script

The verification script file is a TCL command file which can be run from a Modelsim shell. The TCL script is located in "**<path to codec>/src/verif/scripts/run_verif.tcl**".

The script file is run by the following command (executed from the **<path to codec>/verif/rtl** directory)

```
# do ../../src/verif/scripts/run_verif.tcl <args>
```

Where <args> can be:

```
-help          Print a help message on how to run the script
-logwlf        Log all signals to wlf files using "add log -r /*"
               before run -all.
-cover         Perform code coverage on the UUT files. This
               compiles the files with the command option "vcom -
               cover bcest..." and sets optimisation to "-O0"
-config <id>   Run configuration <id> where <id> can be config1 or
               config2 etc. or <id> can be all.
-config_range <low> <high>  Run a range of configurations from low
               to high inclusive. Note there are 83 configurations
               in total
```

4.2 Procedure and simulator environment

The test-bench can be executed from Modelsim or from the shell command line. The test-bench scripts are designed to run only TCL commands.

4.2.1 Simulator environment

The Modelsim simulator is used to verify the SpaceWire CODEC. The Modelsim version is *Modelsim SE 6.4c*.

The verification directory is "**`<path to codec>/verif/rtl`**".

The full verification is run from the in Modelsim as follows

```
# do ../../src/verif/scripts/run_verif.tcl <args>
```

Note: Some versions of modelsim requite the vsim -novopt option to run the simulation successfully. This is due to a Modelsim bug which is fixed in Modelsim 6.4.

4.2.2 Procedure from Modelsim

1. Start Modelsim either by running "modelsim" from the windows start menu or by running "vsim" from the shell prompt
2. From the Modelsim prompt change to the verification directory "**`<path to codec>/codec/verif/rtl`**"
3. Run the command "**`do ../../src/verif/scripts/run_verif.tcl –cover1 –logwlf2 –config all`**"

[1] The –cover command can be omitted if the Modelsim license does not support code coverage, such as the Modelsim distribution supplied by Actel.

[2] The **`–logwlf`** is not necessary if the waveforms will not be viewed. At least 1.8Gbytes of storage is required to generate the wlf files.

A plain text version of the commands is listed below:

```
do ../../src/verif/scripts/run_verif.tcl –cover –logwlf –config
all
```

4.2.3 Procedure from shell

1. From a Windows command prompt or a Unix terminal change to the verification directory "**`<path to codec>/codec/verif/rtl`**"

2. Start Modelsim by running "`vsim -c`"
3. Run the command "`do ../../src/verif/scripts/run_verif.tcl -cover[1] -logwlf[2] -config all`"

[1] The –cover command can be omitted if the Modelsim license does not support code coverage, such as the Modelsim distribution supplied by Actel.

[2] The `-logwlf` is not necessary if the waveforms will not be viewed. At least 1.8Gbytes of storage is required to generate the wlf files.

A plain text version of the commands is listed below:

```
vsim -c
do ../../src/verif/scripts/run_verif.tcl -cover -logwlf -config all
```

4.3 Output Files

config_[0-9]*.log	Log file for configuration * (01, 02, etc.)

The following files are only output if the –*cover* option is enabled in run_verif.tcl

config_[0-9]*.log_cov_summary.log	Coverage summary for configuration * (01, 02, etc.)
config_[0-9]*.log_cov_zeroes.log	Detailed view of lines with zero coverage for configuration * (01, 02, etc.)
config_[0-9]*.log_cov_zeroes.log	Detailed view of lines with zero coverage for configuration * (01, 02, etc.)
cov_summary.txt	Summary of collated code coverage information over all configurations.
cov_zeroes.txt	Lines which have zero coverage in the test-bench.
cov_lines.txt	Coverage report for all lines in the UUT.

4.4 Completion of test-bench

Upon completion of the test-bench run the TCL script will check the resultant log files for the completion message using the TCL code excerpt below. The code checks that each log file has been generated and then records the failure message from the file. Each configuration should complete with a message:

"Failure: Test-bench finished (all commands completed)"

Using the following TCL code.

```
# check if all was the target
if { $config == "all" || $config == "range" } {
    # check config logs for failure
    puts ""
    puts "   Checking logs for failure messages"
    if { $config == "all" } {
        set j 1
        set k 83
    } else {
        set j $range_low
        set k $range_high
    }
    # check each log file in turn
    for {set i $j} {$i<=$k} {incr i} {
        # log file name is config$i.log
        if { $i < 10 } {
            set fname "config0$i.log"
        } else {
            set fname "config$i.log"
        }
        # check for file
        if { ! [ file exists $fname ] } {
            puts "ERROR: Could not file $fname when checking for
failure!"
        } else {
            # open file and search for failures
            set ihandle [ open $fname r ]
            set found 0
            # do file line by line
            while { [gets $ihandle line] >= 0 } {
                # do regexp on line
                if { [ regexp "^.*Failure:(.*)$" $line trash keep ] }
{
                    puts "      $fname: $keep"
                    set found 1
                }
            }
            # check if found failure
            if { ! $found } {
                puts "ERROR: Error when doing $fname, regexp not
found"
            }
        }
    }
}
```

5 VERIFICATION RESULTS

The verification results are presented in this section.

5.1 Summary

The verification is performed on code revision “uodcodec_cvsrel_2_3”. All verification steps were completed successfully and the verification is run as defined in section 4.2.

5.2 Results

The verification command is completed with the following output indicating success of all test-cases:

```
Checking logs for failure messages
   config01.log:  Testbench finished (all commands completed)
   config02.log:  Testbench finished (all commands completed)
   config03.log:  Testbench finished (all commands completed)
   config04.log:  Testbench finished (all commands completed)
   config05.log:  Testbench finished (all commands completed)
   config06.log:  Testbench finished (all commands completed)
   config07.log:  Testbench finished (all commands completed)
   config08.log:  Testbench finished (all commands completed)
   config09.log:  Testbench finished (all commands completed)
   config10.log:  Testbench finished (all commands completed)
   config11.log:  Testbench finished (all commands completed)
   config12.log:  Testbench finished (all commands completed)
   config13.log:  Testbench finished (all commands completed)
   config14.log:  Testbench finished (all commands completed)
   config15.log:  Testbench finished (all commands completed)
   config16.log:  Testbench finished (all commands completed)
   config17.log:  Testbench finished (all commands completed)
   config18.log:  Testbench finished (all commands completed)
   config19.log:  Testbench finished (all commands completed)
   config20.log:  Testbench finished (all commands completed)
   config21.log:  Testbench finished (all commands completed)
   config22.log:  Testbench finished (all commands completed)
   config23.log:  Testbench finished (all commands completed)
   config24.log:  Testbench finished (all commands completed)
   config25.log:  Testbench finished (all commands completed)
   config26.log:  Testbench finished (all commands completed)
   config27.log:  Testbench finished (all commands completed)
   config28.log:  Testbench finished (all commands completed)
   config29.log:  Testbench finished (all commands completed)
   config30.log:  Testbench finished (all commands completed)
   config31.log:  Testbench finished (all commands completed)
   config32.log:  Testbench finished (all commands completed)
   config33.log:  Testbench finished (all commands completed)
   config34.log:  Testbench finished (all commands completed)
   config35.log:  Testbench finished (all commands completed)
   config36.log:  Testbench finished (all commands completed)
   config37.log:  Testbench finished (all commands completed)
   config38.log:  Testbench finished (all commands completed)
   config39.log:  Testbench finished (all commands completed)
```

```
config40.log:  Testbench finished (all commands completed)
config41.log:  Testbench finished (all commands completed)
config42.log:  Testbench finished (all commands completed)
config43.log:  Testbench finished (all commands completed)
config44.log:  Testbench finished (all commands completed)
config45.log:  Testbench finished (all commands completed)
config46.log:  Testbench finished (all commands completed)
config47.log:  Testbench finished (all commands completed)
config48.log:  Testbench finished (all commands completed)
config49.log:  Testbench finished (all commands completed)
config50.log:  Testbench finished (all commands completed)
config51.log:  Testbench finished (all commands completed)
config52.log:  Testbench finished (all commands completed)
config53.log:  Testbench finished (all commands completed)
config54.log:  Testbench finished (all commands completed)
config55.log:  Testbench finished (all commands completed)
config56.log:  Testbench finished (all commands completed)
config57.log:  Testbench finished (all commands completed)
config58.log:  Testbench finished (all commands completed)
config59.log:  Testbench finished (all commands completed)
config60.log:  Testbench finished (all commands completed)
config61.log:  Testbench finished (all commands completed)
config62.log:  Testbench finished (all commands completed)
config63.log:  Testbench finished (all commands completed)
config64.log:  Testbench finished (all commands completed)
config65.log:  Testbench finished (all commands completed)
config66.log:  Testbench finished (all commands completed)
config67.log:  Testbench finished (all commands completed)
config68.log:  Testbench finished (all commands completed)
config69.log:  Testbench finished (all commands completed)
config70.log:  Testbench finished (all commands completed)
config71.log:  Testbench finished (all commands completed)
config72.log:  Testbench finished (all commands completed)
config73.log:  Testbench finished (all commands completed)
config74.log:  Testbench finished (all commands completed)
config75.log:  Testbench finished (all commands completed)
config76.log:  Testbench finished (all commands completed)
config77.log:  Testbench finished (all commands completed)
config78.log:  Testbench finished (all commands completed)
config79.log:  Testbench finished (all commands completed)
config80.log:  Testbench finished (all commands completed)
config81.log:  Testbench finished (all commands completed)
config82.log:  Testbench finished (all commands completed)
config83.log:  Testbench finished (all commands completed)
config84.log:  Testbench finished (all commands completed)
```

5.3 Code analysis test cases

The following sections define the code analysis results for test cases which cannot be covered in simulation.

5.3.1 T.EXC.ERW.1

T.EXC.ERW.1 test checks that ErrorWait shall only be entered from state ErrorReset. The command below checks for *ErrorWait* state transitions.

```
$ nl -ba ../../src/vhdl/initfsm/init_fsm.vhd | grep -C 5 -e
'^.*NEXT_INTFCE_STATE[ \t]*<=[ \t]*ErrorWait'
   292        -- dependant on current state
```

```
   293        case INTFCE_STATE is
   294           when ErrorReset =>
   295               -- state transition to state ErrorWait after 6.4
us
   296              if (TIMER_EVENT_EN = '1') then
   297                 NEXT_INTFCE_STATE <= ErrorWait;
   298              else
   299                 NEXT_INTFCE_STATE <= ErrorReset;
   300              end if;
   301
   302           when ErrorWait =>
--
   306                      if (RX_ERR = '1' or GOT_FCT = '1' or
CHAR_SEQ_ERROR = '1') then
   307                 NEXT_INTFCE_STATE <= ErrorReset;
   308              elsif (TIMER_EVENT_EN = '1') then
   309                 NEXT_INTFCE_STATE <= Ready;
   310              else
   311                 NEXT_INTFCE_STATE <= ErrorWait;
   312              end if;
   313
   314           when Ready =>
   315                  -- state transition dependant on errors and
LINK_ENABLED
   316               -- if error then move to state ErrorReset, error
is RX_ERR or
```

From the code above the ErrorWait state is only entered when leaving ErrorReset or remaining in ErrorWait.

5.3.2 T.EXC.RDY.1

T.EXC.RDY.1 test checks that Ready shall only be entered from state ErrorWait. The command below checks for *Ready* state transitions.

```
$ nl -ba ../../src/vhdl/initfsm/init_fsm.vhd | grep -C 5 -e
'^.*NEXT_INTFCE_STATE[ \t]*<=[ \t]*Ready'
   304              -- if error then move to state ErrorReset, error
is RX_ERR or
   305              -- GOT_FCT or CHAR_SEQ_ERROR
   306              if (RX_ERR = '1' or GOT_FCT = '1' or
CHAR_SEQ_ERROR = '1') then
   307                 NEXT_INTFCE_STATE <= ErrorReset;
   308              elsif (TIMER_EVENT_EN = '1') then
```

```
   309                 NEXT_INTFCE_STATE <= Ready;
   310              else
   311                 NEXT_INTFCE_STATE <= ErrorWait;
   312              end if;
   313
   314           when Ready =>
--
   318              if (RX_ERR = '1' or GOT_FCT = '1' or
CHAR_SEQ_ERROR = '1') then
   319                 NEXT_INTFCE_STATE <= ErrorReset;
   320              elsif (LINK_ENABLED = '1') then
   321                 NEXT_INTFCE_STATE <= Started;
   322              else
   323                 NEXT_INTFCE_STATE <= Ready;
   324              end if;
   325
   326           when Started =>
   327              -- state transition dependant on errors and
FIRST_NULL
   328              -- if error then move to state ErrorReset, error
is RX_ERR or
```

From the code above the Ready state is only entered when leaving ErrorWait or remaining in Ready.

5.4 Code Coverage

In this section the code coverage verification results are presented. The code coverage is successful for all functional lines in the UUT model. The lines which are not covered are listed in the following paragraphs:

5.4.1 File: codec/src/vhdl/initfsm/init_fsm.vhd

Line: 500

```
$ nl -ba ../../src/vhdl/initfsm/init_fsm.vhd | grep -C 3 -e '^[
\t]*500'
   497                    tempload6_4 := '1';
   498                 end if;
   499
   500              when others => null;
   501           end case;
```

```
502
503            -- assign the outputs to load the state counter
```

The line cannot be covered by simulation as it is used to protect against malfunction of the state registers by a single event effect.

5.4.2 File: codec/src/vhdl/initfsm/initfsm_counter.vhd

Line: 327

```
$ nl -ba ../../src/vhdl/initfsm/initfsm_counter.vhd | grep -C 3 -e
'^[ \t]*327'
324                     end if;
325                 end if;
326
327              when others => null;
328           end case;
329
330        end if;
```

The line cannot be covered by simulation as it is used to protect against malfunction of the state registers by a single event effect.

5.4.3 File: codec/src/vhdl/receive/rxdecode.vhd

Line: 324

```
$ nl -ba ../../src/vhdl/receive/rxdecode.vhd | grep -C 3 -e '^[
\t]*324'
321              DXSTATE_NEXT <= receiveError;
322
323           when others =>
324              DXSTATE_NEXT <= firstNullEscStart;
325        end case;
326
327     end process rxdecode_next
```

The line cannot be covered by simulation as it is used to protect against malfunction of the state registers by a single event effect.

5.4.4 File: codec/src/vhdl/receive/rxnchar_resync_ff.vhd

Line: 155,157

```
$ nl -ba ../../src/vhdl/receive/rxnchar_resync_ff vhd | grep -C 3 -e
'^[ \t]*155'
   152            when "101" => return "100";
   153            when "100" => return "000";
   154
   155            when others => return "000";
   156         end case;
   157      end gray_plus_one;
   158
```

The line cannot be covered by simulation but all normal cases of the std_logic_vector are covered.

5.5 Asynchronous Interfaces

An analysis of the asynchronous signals in the UoD SpaceWire CODEC is performed in this section.

5.5.1 Receiver

The receiver clock is derived from the serial input bit-stream signals **DIN** and **SIN** and is asynchronous to the system clock domain. The following table outlines the signals which are synchronised into the system clock domain and the protocol used to capture the signal.

Synchronisation of receiver signals is performed in the VHDL component initfsm_sync.vhd

Signal Name	Source CLK	No DFF	Source/Destination Module		Destination CLK
FIRST_NULL	**RX_CLK**	2	rxdecode	init_fsm	**SYSCLK**
GOT_FCT	**RX_CLK**	2	rxdecode	init_fsm	**SYSCLK**
NCHAR_SEQ_ERROR	**RX_CLK**	2	rxdecode	init_fsm	**SYSCLK**
TCODE_SEQ_ERROR	**RX_CLK**	2	rxdecode	init_fsm	**SYSCLK**
GOT_NCHAR	**RX_CLK**	2	rxdecode	rxdataformat	**SYSCLK**

GOT_TIMECODE	RX_CLK	2	rxdecode	rxdataformat	SYSCLK
PARITY_ERROR	RX_CLK	2	rxdecode	init_fsm	SYSCLK
ESCAPE_ERROR	RX_CLK	2	rxdecode	init_fsm	SYSCLK
RXFCT_REQ	RX_CLK	2	rxdecode	init_fsm	TXBITCLK

Table 5-1 rxdecode signal synchronisation

5.5.2 Transmitter

The transmitter bit clock is asynchronous to **SYSCLK** and **RXBUF_CLK** when an external TXCLK, internal divider or 10MHz reference clock is used. The transmitter is reset and enabled by init_fsm signals which are registered in **SYSCLK.** Requests to send FCTs are performed by the receive credit counter module which is registered in **RXBUF_CLK**. The transmitter has credit to send N-chars dependent on **RXDCT_REQ** which is registered by **RX_CLK.**

Signal Name	Source CLK	No DFF	Source/Destination Module		Destination CLK	Notes
DISABLE_TX_N	SYSCLK	2	Init_fsm	txencode	TXBITCLK	
EN_FCT	SYSCLK	2	Init_fsm	txencode	TXBITCLK	(1)
EN_NCHAR	SYSCLK	2	Init_fsm	txencode	TXBITCLK	(1)
SENDFCT_REQ	RXBUF_CLK	2	rxcredit	txencode	TXBITCLK	(2)
RXFCT_REQ	RX_CLK	2	rxdecode	txencode	TXBITCLK	
SENDTIME_REQ	SYSCLK	2	txtcode_send	txencode	TXBITCLK	(2)

Table 5-2 txencode synchronisation

*Note (1) – Only when **CFG_BITCLK** is not SYS_DEFAULT or SYS_EN.*

*Note (2) – Only when **CFG_BITCLK** is not SYS_DEFAULT or SYS_EN and **CFG_SYNCRDCLK** = '0'*

5.5.3 Receiver credit

The receiver credit counter and receive buffer pointer logic is asynchronous to **SYSCLK** when **CFG_SYNCRDCLK** = '0'.

Signal Name	Source CLK	No DFF	Source/Destination Module		Destination CLK	Notes
SENDFCT_ACK	TXBITCLK	2	txencode	rxcredit	RXBUF_CLK	(2)
ERROR_RECOVER_ACK	SYSCLK	2	init_fsm	rxrcredit	RXBUF_CLK	(1)
CREDIT_ERROR_ACK	SYSCLK	2	init_fsm	rxcredit	RXBUF_CLK	(1)

Table 5-3 rxcredit signal synchronisation

*Note (1) – Only when **CFG_SYNCRDCLK** = '0'.*

Note (2) – Only when ***CFG_BITCLK*** *is not SYS_DEFAULT or SYS_EN and* ***CFG_SYNCRDCLK*** *= '0'*

5.5.4 Transmit clock generator

The transmit clock generator is asynchronous to **SYSCLK** therefore **SEL_SLOW** (select the default initialisation rate of 10mbits/s) is synchronised to TXCLK.

Signal Name	Source CLK	No DFF	Source/Destination Module		Destination CLK	Notes
DISABLE_TX_N	**SYSCLK**	2	init_fsm	txclkgen	**TXBITCLK**	
SEL_SLOW	**SYSCLK**	2	init_fsm	txclkgen	**TXBITCLK**	

Table 5-4 txclkgen signal synchronisation