Bandwidth Engine® MSR576 SerDes Bringup Guide

Application Note AN-601

Version 0.3, December 2011

MoSys, Inc.

Contents

Figures

Tables

1 Introduction

The Bandwidth Engine MSR576 is a memory-dominated device from MoSys, Inc. It has a high-speed serial interface, known as the GigaChip™ Interface, instead of a parallel interface.

This document describes how to program the device's SerDes to enable its functionality and perform signal integrity tests. It also provides the initialization sequence for the PLL, TX, and RX. The following figure shows a top level diagram of the PCS and SerDes in the Bandwidth Engine device.

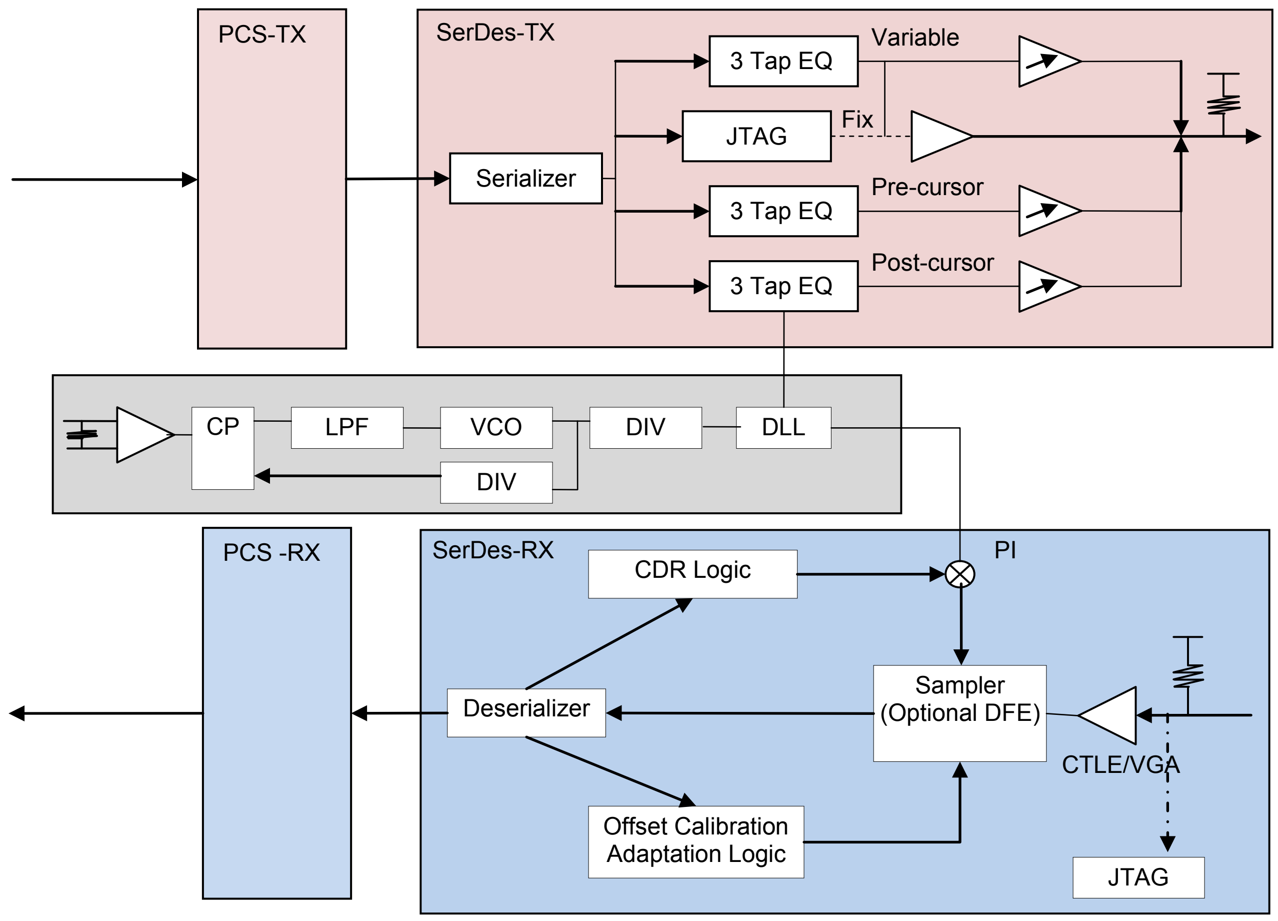


Figure 1 Block diagram of the PCS and SerDes

The SerDes has the following features that can be used for bringup:

- Analog outputs (AMON) to monitor internal supply voltages.
- Test Pattern (TPAT) generators/checkers with error-injection capabilities
  - PRBS31, PRBS23, PRBS11, PRBS9, PRBS7
  - User-defined Programmable Pattern (referred to as PPAT)
- JTAG boundary scan
- SPI port and internal Register Bus for accessing all control and status registers (see Section 4).

2 Board Level Requirements

MoSys provides sample kits for the Bandwidth Engine MSR576 device. Each kit includes a printed circuit board with an MSR576 device, connectors, and voltage regulators. There are two kinds of boards:

- Characterization board: The SerDes balls are brought out to SMA connectors.
- AirMax board: The SerDes balls are brought out to FCI AirMax connectors at the board's edge. The placement of the connectors conforms to the Interlaken Alliance's physical interoperability recommendations.

Many of the procedures in this document can be done with either kind of board. Other procedures apply only to the characterization board, unless you have a way to make a high-quality connection between the MSR576 device and high-speed test equipment.

One key consideration is to perform all measurements as close to the SerDes balls as possible. For the most accurate characterization, choose the RX or TX lane that has the shortest distance to its SMA connectors.

2.1 Power Supplies

Good power supply is critical for SerDes operation and is covered in Section 5.3, "Power Supply." It is best to measure the power supplies close to the SerDes, for both DC level and noise.

Each power supply rail on the characterization or AirMax board may be supplied by an on-board voltage regulator or by an external power supply. For more information, see the *Bandwidth Engine MSR576 Characterization Board User Guide* or the *Bandwidth Engine MSR576 AirMax Board User Guide*.

Bench supplies typically provide cleaner power than the on-board regulators. If you choose to use external power supplies, then follow these guidelines:

- Use supplies whose voltage and current are adjustable.
- Turn on the supply voltages in a sequence that meets the requirements of AN-604, *Bandwidth Engine MSR576 Power-Up and Reset*. If the supplies do not come up in the required sequence, you might need to press the master reset button again after power-up.

2.2 Reference Clock

The reference clock input for each MSR576 device on a board comes from a pair of SMA connectors. The cables may be connected to an external source or to the on-board reference clock.

- On the characterization board, each site is independently connected to a clock source.

Whatever the reference clock source is for the MSR576 device, the device and the external test equipment *must* use the same source. That is, the clocking is mesochronous. The GigaChip Interface cannot introduce skip symbols to compensate for clock rate differences.

External source:

- The reference clock requires LVDS levels. AC coupling is not required if the external source meets the common-mode specifications of the device (see datasheet).
- Use a clean source, such as a pattern generator or a BERT (bit-error-rate tester).

- For line loopback and receiver tolerance testing, the reference clock be must sourced from a BERT.
- The reference clock frequency must be adjustable.

On-board source:

- The on-board crystal oscillator provides four frequencies, from 100 MHz to 312.5 MHz.
- For more information, see the board's *User Guide*.

2.3 SerDes

The SerDes balls (`CMD`*`i`*`RX`, `Q`*`i`*`TX`) require CML levels (see datasheet).

3 Equipment Requirements

Some of the lab equipment required for SerDes bringup is listed below. (Some of this equipment might not apply to the AirMax board.):

- Real-time oscilloscope
- Sampling oscilloscope with CDR module
- High-bandwidth differential probes
- Pulse generator
- Spectrum analyzer
- High performance serial BERT/J-BERT
- High-quality, skew-matched SMA cables
- A means of measuring the temperature of the device:
  - Thermal stream and IR thermometer, or
  - A current source and voltmeter to connect to the on-chip temperature sensing diode (`TEMPDIODEP/N` balls)

The bandwidth of the real-time oscilloscope and probes must be at least 3 times the fundamental frequency, that is, 1.5 times the data rate.

A complete list of equipment is found in Section 10, "Equipment for Characterization."

4 Register Bus, SPI, Racetrack, and the `RESET#` and `CONFIG#` Balls

The MSR576 device has an internal Register Bus for accessing the control and status registers of the device. The SPI port gives external access to the Register Bus. The sample kit includes a USB-to-SPI dongle and a USB cable.

The Linux computer in the sample kit comes with Racetrack software pre-installed. Racetrack provides a Tcl command-line interface to the Register Bus through the SPI port.

The `RESET#` ball is an active-low static input that allows the system to completely reset the device. On both the characterization and AirMax boards, the master reset button (MRESET) provides `RESET#` to the MSR576 devices. When the board asserts and then deasserts `RESET#`, the internal registers are set to their default values. Among other things, all previous changes made to the control and status registers by the Register Bus are lost.

The `CONFIG#` ball is an active-low static input that allows the system to change the configuration of the device before it comes completely out of reset. (On both the characterization and AirMax boards, `CONFIG#` is controlled by jumpers.) If `CONFIG#` is high when the board deasserts `RESET#`, the internal registers are set to their default values, and the device proceeds directly to normal operation. If `CONFIG#` is low when the board deasserts `RESET#`, the internal registers are set to their default values, but the reset process pauses to allow the system to change any addressable register by means of the Register Bus. The reset process resumes when the system writes 1 to the `ext_config_done` bit of the `rstgen_rst` register, and the device proceeds to normal operation. You can use a Racetrack script to change the configuration of the device during reset, then proceed to normal operation. Do not reset the device *after* changing its configuration, because you will lose all your changes.

MoSys recommends that you reset the device before making any change to the SerDes settings.

4.1 Introduction to Racetrack Usage

If you are not familiar with how to use Racetrack with your board, please refer to the “Using the Racetrack Software” section in the board’s *User Guide*. Practice the instructions in the sub-sections entitled “Start Racetrack and Reset the Device” and “Run Racetrack Scripts”.

This section provides a condensed version of the *User Guide* instructions.

4.1.1 Start Racetrack and Reset the Device

- Restore the `CONFIG#` setting to the default (low) value.
- Power up the board with the SPI adapter connected.
- Boot up the Linux machine. Log in with userid `be1`.
- Open a terminal.
- Change directory:

```
$ cd ~
```

- Start Racetrack.

```
$ tools/bin/racetrack
```

Racetrack will print a startup message, followed by the Racetrack prompt (**`racetrack%`**). See your board’s *User Guide* for an example of the full message.

- Press the master reset button to reset the device. Because `CONFIG#` is low, the device should not come out of reset.
- Execute a single command in Racetrack to get the device out of reset:

```
racetrack% poke rstgen_rst.ext_config_done 0x1
```

- The RDY indicator LED should turn on, indicating that device has come out of reset.

4.1.2 Run Racetrack Scripts

Now you can run the Racetrack scripts that MoSys has provided. **Run these scripts after pressing the master reset button.** Each script ends with a "`poke rstgen_rst.ext_config_done 0x1`" statement to bring the device out of reset.

- **`racetrack% cd diags/serdes`**
- Press the master reset button to reset the device. Notice that we reset the device *before* running the script.
- Run `int_loopback_10.3g.tcl`, which is the internal loopback test for 10.3125 Gbps.

```
racetrack% source int_loopback_10.3g.tcl
```

The RDY indicator LED should turn on. The script will run for about a minute, then display the error count for each of the 16 RX-TX lane pairs.

4.2 Racetrack Peek and Poke Commands

When bringing up the SerDes, you need to access various registers in the device. Racetrack provides peek and poke commands with options for accessing the registers associated with all lanes, all lanes of one GigaChip Interface port, or a single lane.

Use the `peek` or `peek_v` command to read a bit field of a register. `peek` returns numeric arguments suitable for use in scripting. `peek_v` returns a verbose, human readable result. Example:

```
racetrack% peek rstgen_rst.ext_config_done
```

Use the `poke` command to write to a bit field of a register. Example:

```
racetrack% poke rstgen_rst.ext_config_done 0x1
```

Here are two important options for the `peek`, `peek_v`, and `poke` commands:

- `-p` *GCI_port* selects one of the GigaChip Interface ports in the device. Specify `0` for port A and `1` for port B.
- `-l` *`lane`* selects an individual lane within a GCI port (0 through 7).

5 Setup and Basic Checks

5.1 Tie `CONFIG#` Low

Tie `CONFIG#` low (asserted), to allow the PLL frequency to be changed before the device comes out of reset.

5.2 External Connections

Make external connections to the board, including the following:

- Connect power, using either the on-board voltage regulators or external power supplies. See the board's *User Guide* for instructions.
  - The MSR576 device needs three power supplies: 1.0V, 1.1V, and 1.5V.

- Connect the USB-to-SPI adapter to a USB port on the Linux machine and to the SPI header on the board.
- Connect a 156.25 MHz reference clock to the device's refclk input SMA connectors, using either the on-board reference clock or an external source. As explained in Section 2.2, the MSR576 device and the external test equipment *must* use the same source.
- If you will do transmitter bringup as described in Section 7, use length-matched cables to connect the TX SMA connectors to test equipment.
- If you will do receiver bringup as described in Section 8, use length-matched cables to connect the RX SMA connectors to test equipment.
- Press the master reset button.

5.3 Power Supply

Check that the power supplies are clean. Noisy power supplies can lead to PLL locking, excessive jitter, or other intermittent issues.

5.4 Register Bus Check

Before bringing-up the SerDes, establish communication between Racetrack and the Register Bus on the device, via USB and SPI.

- Start the Racetrack program on the Linux machine (see Section 4.1).
- If the startup message includes the sentence, "`Chip ID string is BE1.`", then you know that Racetrack has established communication with the Register Bus.

5.5 Reference Clock

Check the reference clock (near the SerDes input) and measure the following:

- Frequency
- Voltages to ensure that the refclk meets the requirements
- refclk jitter

Ensure that all the preceding parameters are within the SerDes datasheet specifications. Too much jitter on the reference clock could result in the PLL not locking or cause too much transmit jitter.

6 PLL Bring-up and Debug

There is one PLL for each GigaChip Interface port, so use the `-p` *`GCI_port`* option with the Racetrack `poke` command.

6.1 PLL Initialization Sequence

In this example, we will operate the SerDes at 10.3125 Gbps.

1. Ensure that the refclk meets the requirements as detailed in Section 2.2. Set refclk frequency to 156.25 MHz.

2. Tie `CONFIG#` low, to allow the PLL frequency to be changed before the device comes out of reset.
3. Press the master reset button (assert `RESET#`) to reset the device. Notice that we reset the device *before* modifying the registers.
4. Ensure that the PLL has the following frequency settings:

Table 1 PLL initialization

Register Name	Field Name	Value
pcssds_rate_ctrl	pll_div	0x21
pcssds_rate_ctrl	pll_band_sel	0x1
pcssds_rate_ctrl	pll_hdr_sel	0x0
pma_misc_cfg	vco_reg_tr	0x1

Use `peek_v` commands to read the settings, for example:

```
racetrack% peek_v pcssds_rate_ctrl.pll_div
```

`pll_div` and `pll_hdr_sel` set the bit rate to 10.3125 Gbps:

- VCO Frequency = 156.25 MHz × 33 × 2 = 10.3125 GHz

`pll_band_sel` chooses the VCO band, and `vco_reg_tr` optimizes the performance of the PLL for the chosen frequency band.

5. Configure the TX lanes to send a clock-like pattern (101010…). See Section 7.2.2, “Configuration for Bringup,” for details.

If you want to change the reference clock frequency or divider values of the PLL, you must reset the device after changing the reference clock, but before changing the PLL settings.

6.2 Proceed to Normal Operation

6. Resume the reset process and allow the device to proceed to normal operation. Do this by writing 1 to the following bit:

Table 2 Assert External Configuration Done

Register Name	Field Name	Value
rstgen_rst	ext_config_done	0x1

Do not press the reset button *after* changing the configuration, because you will lose all your changes.

7. Observe the jitter on the TX balls. If the jitter of the reference clock is within specifications, the random jitter on the TX balls should be low. See Section 7.3, “Random Jitter,” for details.

7 Transmitter Bringup

7.1 Analog Transmitter Settings

The following table shows the Register Bus bit fields that control the analog settings of the transmitter. These two bit fields control the pre-cursor, cursor, and post-cursor weights in the 3-tap pre-emphasis circuit.

Table 3 Analog transmitter settings

Parameter	Register Name	Field Name
Pre-cursor weight	pma_tx_preemph	premptap[3:0]
Cursor weight	pma_tx_cfg	swing
Post-cursor weight	pma_tx_preemph	premptap[8:4]

You can do IBIS-AMI simulations to get a good starting point for optimizing the analog settings. See Section 9 for more information.

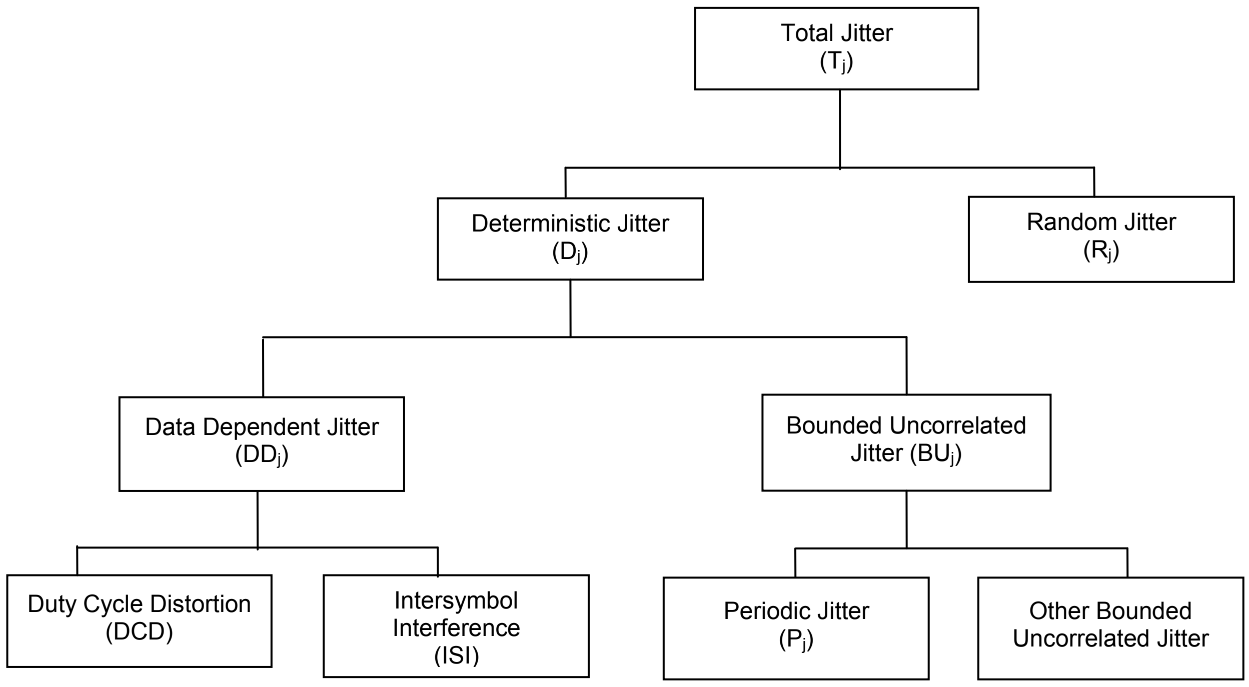
To write the same settings to all lanes of a GigaChip Interface port, use the Racetrack `poke` command with the `-p` *`GCI_port`* option. If you want to adjust a lane individually, use both the `-p` *`GCI_port`* and `-l` *`lane`* options.

7.2 Introduction to Transmitter Jitter

7.2.1 Types of Jitter

Jitter is typically classified as shown in the following figure. With a clock-like pattern there should be no ISI jitter.

Figure 2 Types of jitter


Total Jitter (T_j)
Deterministic Jitter (D_j)
Random Jitter (R_j)
Data Dependent Jitter (DD_j)
Bounded Uncorrelated Jitter (BU_j)
Duty Cycle Distortion (DCD)
Intersymbol Interference (ISI)
Periodic Jitter (P_j)
Other Bounded Uncorrelated Jitter

7.2.2 Configuration for Bringup

TX bringup involves optimizing the transmitter swing and pre-emphasis along with the PLL bandwidth.

1. Send a clock-like pattern (101010…) by configuring the PPAT (programmable pattern) generator and checker. There is one PPAT generator for each GigaChip Interface port, so use the `-p` *`GCI_port`* option with the Racetrack `poke` command.

Table 4 TPAT generator: configure clock-like pattern

Register Name	Field Name	Value
tpat_tx_cfg	mode	0x6
tpat_tx_cfg	ppat_base	0x0
tpat_tx_cfg	ppat_limit	0x0
tpat_ppat_data	tpat_ppat_data[0]	0x02aa

2. Measure the jitter on the transmit output by connecting the transmit output to the high speed scope. Ensure that the cables are skew-matched.

7.3 Random Jitter

Random jitter is a jitter component that is unbounded in amplitude and is Gaussian in nature. Rj is measured as an RMS value. The uncorrelated unbounded Gaussian jitter of the MSR576 device (T_DCD) is no more than 0.15 UIpp, and the total output jitter (T_TJ) is no more than 0.30 UIpp (see datasheet). If you observe jitter greater than these amounts, there is probably a problem with the reference clock source or with the measurement setup.

1. Measure the reference clock jitter and ensure that it meets the requirements.
2. If the on-board clock source is jittery, use a clean external clock source from a pattern generator or a BERT. First measure the jitter characteristics of the clock source by connecting it directly to the scope. Then measure the transmit jitter of the MSR576 device by sending a clock-like pattern.

7.4 Periodic Jitter

Periodic jitter is also referred to as sinusoidal jitter. This is typically caused by signal crosstalk or power supply noise.

- Ensure that AC characteristics (or ripples) in the power supply are within the MoSys® specifications.

7.5 Intersymbol Interference (ISI)

Intersymbol interference (ISI) is caused by the limitation of the physical channel, where the transmitted data lasts longer than the bit period and interferes with the subsequent data bits. ISI is also referred to as Data Dependent Jitter (DDJ) and occurs in non-clocklike waveforms. Higher data rates or a longer PRBS polynomial result in greater ISI. Optimize ISI after you optimize random jitter and periodic jitter.

1. Generate a PRBS-7 pattern (see Table 5) and optimize the transmitter swing and pre-emphasis to get the best possible ISI. To reduce the amount of trial and error, first simulate the system as described in application note AN-600, *IBIS-AMI Modeling of Bandwidth Engine Serial Links*.
2. Measure the width and height of the eye opening. Typically, the transmit jitter and eye measurements are for only half of the link; you will need to adjust the settings to account for the path from the transmit measurement point to the receive pins.
3. Follow the above steps for other lanes/links and data rates and skew parts/PVT.

7.6 Injecting Transmitter Errors

You can inject up to 10 errors at a time into the transmitted test pattern by writing to the `tpat_force_ber` register. When you write a value to this register, the test pattern generator will exclusive-OR the 10 LSBs of the value with 10 bits of the outgoing pattern. For an example of the use of this register, see `diags/serdes/shared/error_count.tcl`.

8 Receiver Bringup

8.1 Verify RX operation

1. Enable PRBS-7 checking mode in the receiver. Apply an external PRBS-7 pattern to the RX pins of the SerDes.

Table 5 TPAT: enable PRBS-7 generator and checker

Register Name	Field Name	Value
tpat_rx_cfg	mode	0x1

(For information on using the other available PRBS sequences, see **`diags/serdes/shared/tpat_prbs31.tcl`**.)

2. Configure the PRBS error counters so they will be cleared to zero whenever you read them. One `tpat_cfg` register controls all lanes of a GCI port.

Table 6 TPAT error counters: enable clear-on-read

Register Name	Field Name	Value
tpat_cfg	ecnt_clr_on_rd	0x1

3. Read the PRBS error counters. There is a separate counter for each lane.

Table 7 TPAT error counters: read

Register Name	Field Name
tpat_ecnt	—

The first time you read an error counter, you will get the number of errors since the last time it was cleared. If you read it again, you will get 0. For more information on using the error counters, see `diags/serdes/shared/error_count.tcl`.

4. Command the external pattern generator to inject errors into the PRBS-7 pattern that you are applying to the SerDes RX balls. Verify that the PRBS error counters report errors. (The counter records 3 errors for an isolated 1-bit error in a PRBS stream, because all of the LFSRs have two taps, and the bad bit will cause an additional error as it shifts past each tap. Predicting the error count is more complicated if errors are near each other in the PRBS stream.) Verify that reading the error counters clears them. When the counters report zero errors, the receiver is operational.
5. Verify all lanes/data rates with different PRBS patterns.

8.2 RX Fine Tuning

This section describes other RX configuration settings that can be adjusted in case of errors. These settings should be left at their default values unless there are errors. To write the same settings to all lanes of a GigaChip Interface port, use the Racetrack `poke` command with the `-p` *`GCI_port`* option. If you want to adjust a lane individually, use both the `-p` *`GCI_port`* and `-l` *`lane`* options.

If you need to adjust the analog settings, you can do IBIS-AMI simulations to get a good starting point. See Section 9 for more information.

8.2.1 Analog Receiver Settings

Table 8 shows the Register Bus bit fields that control the analog settings of the receiver.

Table 8 Analog receiver settings

Register Name	Field	Description
pma_rx_cfg_2	agc_ctl	CTLE gain control
pma_rx_cfg_2	dfe_dac	DFE programmable tap weight
pma_rx_ctle_ctrl	zero	CTLE zero control

8.2.2 CDR Status

The status of CDR (clock and data recovery) can be monitored by reading bit fields of the `pma_rx_cdr_status` register. The expected values are shown below.

Table 9 Monitoring CDR status

Register Name	Field	Expected Value
pma_rx_cdr_status	phase_ctrl	This field should change rapidly while the PLL is locking, then settle down to an arbitrary value and dither around that mean value.
pma_rx_cdr_status	freq_accum	This field should be very stable, with the value close to 0. The value may be positive or negative.

9 IBIS-AMI Simulation

To get a good starting point for optimizing the analog transmitter and receiver settings of the MSR576 device and the host, you can simulate the serial channels (MSR576 chip and package plus board net plus host chip and package). MoSys provides IBIS-AMI models for the MSR576 chip and its package. IBIS-AMI simulations run much faster than SPICE simulations, though with less accuracy. For more information, see AN-600, *IBIS-AMI Modeling of Bandwidth Engine Serial Links*, and the release notes that accompany the MoSys models.

10 Equipment for Characterization

Table 10 gives a recommended list of lab equipment for complete characterization of the SerDes. Some of this equipment might not apply to the AirMax board.

Table 10 Equipment for characterization

Item	Vendor	Type / Number	Description / Specification
Real-time oscilloscope	Agilent	DSA-X 92004A	20 GHz real time oscilloscope

Item	Vendor	Type / Number	Description / Specification
Sampling oscilloscope	Agilent	DCA-J 86100C	20 GHz 2ch digital sampling scope mainframe
CDR mega module	Agilent	86108A	Module for filtering jitter
High-bandwidth differential probes	Agilent	compatible with oscilloscope	20 GHz BW, compatible with oscilloscope
Pulse generator	Agilent	81134A	3.3GHz
Digital multimeters	Fluke	8050A	Measure voltages and currents
DC power supply	Agilent	E3648A, E3649A	Regulated DC power supply
Spectrum analyzer	Agilent	W9068A	Up to 20 GHz
Serial JBERT	Agilent	N4903A/4903B	12.5 Gbps BERT
High-quality SMA cables	Amphenol Connex /Huber Suhner / Astro Lab		Must be able to handle signals >12.5 GHz. Pairs *must* be skew-matched.
Thermal stream	Silicon Thermal/ Thermonics	CH1000/T2420	To force temperature
IR thermometer	Fluke	561	To measure the temperature

The bandwidth of the real-time oscilloscope and probes must be at least 3 times the fundamental frequency, that is, 1.5 times the data rate. For example, if the data rate is 10.3125 Gbps, the bandwidth must be at least 15.5 GHz.

Some simpler equipment is also required:

- Power supply cables
- Crocodile pins
- Potentiometer tweaks
- 50-ohm terminators
- DC blockers
- Torque wrench for SMA cables

11 References

Table 11 References

Vendor	Document type	Document

Vendor	Document type	Document
MoSys	MSR576 documents	Bandwidth Engine MSR576 AirMax Board User Guide Bandwidth Engine MSR576 Characterization Board User Guide Bandwidth Engine MSR576 Datasheet Bandwidth Engine MSR576 Power-Up and Reset (AN-604) Bandwidth Engine MSR576 User Guide IBIS-AMI Modeling of Bandwidth Engine Serial Links. (AN-600)
Agilent	JBERT documents	User Guide Datasheet
Agilent	White papers, application notes	Jitter Analysis: the Dual-Dirac Model, RJ/DJ and Q-Scale Calibrated Jitter, Jitter Tolerance Test and Jitter Laboratory with the Agilent J-BERT
Tektronix	Primer	Understanding and Characterizing Timing Jitter

12 Version History

Date	Version	Changes
Dec. 17, 2010	0.1	First release to MoSys partner companies.
February 2011	0.2	Reclassify document as app note. Incorporate AirMax board. Describe reset and Racetrack more thoroughly. Other minor changes.
Dec. 2011	0.3	Update script usage. Describe error checking and error injection. Mention IBIS-AMI simulation. Revise analog receiver settings. Update equipment list. Delete irrelevant material. Make various clarifications and minor corrections.

11/23/11

MoSys, Inc.
3301 Olcott Street
Santa Clara, CA 95054
USA

Phone	408-418-7500
Fax	408-418-7501
Web	http://www.mosys.com