StrataDNX

GDDR6 Interface Configuration and Debugging Guide

Application Note
BCM88480
BCM88280
BCM88290

Table of Contents

1 Introduction

The BCM88480 and BCM88280 each use two external GDDR6 DRAM devices for deep packet buffering. The BCM88290 uses one external GDDR6 DRAM device.

This document provides information about BCM88480, BCM88280, and BCM88290 GDDR6 interface configuration, tuning, and debugging. Use this document with the *Traffic Manager Programming Guide* (88690-PG2xx). If any discrepancies exist between this application note and the *Traffic Manager Programming Guide*, the *Traffic Manager Programming Guide* takes precedence.

NOTE:

- In this document, all references to the BCM88480 device also include the BCM88280 device.
- In this document, most references to the BCM88480 also include the BCM88290 device, unless indicated otherwise. For the main DDR-related differences between the BCM88480 and BCM88290 devices, see Appendix A: BCM88290 DDR Differences.
- This document is aligned with SDK 6.5.31.

2 SOC Properties for GDDR6 Operation

This section describes the GDDR6-related SOC properties and their configuration options. The SOC properties that apply to GDDR6 operation control the following settings:

- The GDDR6 interface status (enabled or disabled)
- The GDDR6 operation frequency (6.8 Gb/s for the BCM88480/BCM88280 and 6.0 Gb/s for the BCM88290)
- The tune mode
- The GDDR6 temperature read (enable or disable)
- The timing parameters for the DDR assembled on board (Micron or Samsung)

2.1 GDDR6 Interface Enable and Disable

The `ext_ram_enabled_bitmap` property defines which DDR interface is enabled.

2.1.1 BCM88480/BCM88280 Settings and Example

Table 1: BCM88840/BCM88820 GDDR6 Interface Enable and Disable

`ext_ram_enabled_bitmap` Property Value	**DDR1**	**DDR0**	**Note**
0 ('b00)	Disable	Disable	SDK default. If using the GDDR6 interface, set the value according to the configuration.
1 ('b01)	Disable	Enable	Use this value when one interface is enabled.
2 ('b10)	Enable	Disable	Do not use during normal operation. Use only for debug purposes.
3 ('b11)	**Enable**	**Enable**	**Use this value when both interfaces are enabled.**

The following example enables both DDR interfaces:

```
ext_ram_enabled_bitmap.BCM8848X=3
```

In the SDK, the property is set in the `config-q2a.bcm` file.

2.1.2 BCM88290 Settings and Example

The BCM88290 uses only the DDR1 interface.

Table 2: BCM88290 GDDR6 Interface Enable and Disable

`ext_ram_enabled_bitmap` Property Value	**DDR1**	**Note**
0 ('b00)	Disable	SDK default. If using the GDDR6 interface, set the value according to the configuration.
2 ('b10)	**Enable**	**Use this value when the interface is enabled.**

The following example enables the DDR1 interface:

```
ext_ram_enabled_bitmap.BCM8848X=2
```

In the SDK, the property is set in the `config-q2a.bcm` file.

2.2 GDDR6 Interface Operation Frequency

The `ext_ram_freq` property defines the GDDR interface operation frequency.

Table 3: GDDR Interface Operation Frequency

Device	**`ext_ram_freq`** **Property Value**	CK Clock Rate	Interface Bandwidth (Per Data Pin)	Notes
BCM88480/ BCM88280	850	850 MHz	6.8 Gb/s (8 × 850)	■ In SDK 6.5.20 and later versions, the default frequency is 6.8 Gb/s, and this SOC property does not need to be set or modified. ■ In SDK 6.5.19 and earlier versions, the default frequency is 8 Gb/s or 7.2 Gb/s. Use this SOC property to set the frequency to 6.8 Gb/s. ■ Property values other than 850 are for debug only.
BCM88290	750	750 MHz	6.0 Gb/s (8 × 750)	—

The following example sets the interface default frequency to 850 MHz/6.8 Gb/s (CK/data):

```
ext_ram_freq.BCM8848X=850
```

The following log output shows a partial dump example from the information that prints to the console during device initialization. The 6800 value (highlighted in green) in the log represents the 6.8 Gb/s (6800 Mb/s) data-bit rate setting.

```
0: BCM Init:    - PLL
PHY PLL Configuration
Fref...........:  100
PDIV...........:    2
NDIV...........:  136
Fvco...........: 6800
Data rate......: 6800
PHY PLL Configuration
Fref...........:  100
PDIV...........:    2
NDIV...........:  136
Fvco...........: 6800
Data rate......: 6800
```

2.3 GDDR6 Tune (Shmoo) Options for Driver Initialization

Prior to any traffic operation on the GDDR6 interface, configure the interface with its tune parameters.

2.3.1 Tune Setting on Driver Initialization

The `dram_phy_tune_mode_on_init` SOC property defines how the tune is performed during driver initialization.

The following table summarizes the different tune setting options.

Table 4: GDDR6 Tune Options

`dram_phy_tune_mode_on_init` Property Value	**Usage**
`RUN_TUNE`	Run tuning at device initialization.
`RESTORE_TUNE_PARAMETERS_OR_RUN_TUNE`	Restore existing tune parameters, if present; otherwise, run tune. Driver initialization using restored parameters is faster than running a new tune using the `RUN_TUNE` option. For more information, see Section 3, Tuning.
`RESTORE_TUNE_PARAMETERS_FROM_SOC_PROPERTIES`	Restore existing tune parameters, if present; otherwise, the tune fails. Driver initialization using restored parameters is faster than running a new tune using the `RUN_TUNE` option. For more information, see Section 3, Tuning.
`SKIP_TUNE`	Skip performing a tune at driver initialization. **NOTE:** This option is for built-in self test (BIST) or debug purposes.

The recommended configuration for operation is as follows:

```
dram_phy_tune_mode_on_init.BCM8848X=RESTORE_TUNE_PARAMETERS_FROM_SOC_PROPERTIES
```

In the SDK, the property is set in the `config-q2a.bcm` file.

2.3.2 Enable Full-Tune Logging on Driver Initialization

Use the `debug soc ddr info` SOC property to display detailed information about the tune process during driver initialization. The information displays before the BCM shell prompt appears.

During normal operation, this option is not required.

In the SDK, the property can be added in the `dnx.soc` file.

2.4 Enable or Disable GDDR6 Temperature Reading

The `ext_ram_temp_read_enable` SOC property enables or disables reading the temperature from the DDR device. The SDK sets the property correctly for Micron and Samsung devices that the SDK supports.

Use the property only if requested for debugging purposes.

To enable a DRAM temperature read, use the following setting:

```
ext_ram_temp_read_enable=1
```

To disable a DRAM temperature read, use the following setting:

```
ext_ram_temp_read_enable=0
```

NOTE: Micron 8G devices do not support temperature reading. For more information, contact the memory vendor.

2.5 GDDR6 SDK Timing and Configuration Parameters

The GDDR6 JEDEC timing parameters the Broadcom® controller uses are part of the SDK for Samsung and Micron GDDR6 devices. For information about the supported GDDR6 devices, refer to the GDDR6 section in the *Hardware Design Guidelines for StrataDNX™ 16-nm Devices* (DNX16-AN1xx).

GDDR6 timing parameters are available in the `/6.5.23/ga/sdk-all-6.5.23/rc/dnx_dram` SDK directory.

It is up to the customer whether to review timing parameters with the memory vendors; however, any change in timing parameters should be reviewed with Broadcom through the CSP.

If using other GDDR6 devices than those listed in this document, it is important to share and review the device parameters with Broadcom as soon as possible. New GDDR6 device integration into the driver may require SDK changes and extended customer validation.

The following table shows an example of timing parameters.

Table 5: GDDR6 SDK Timing Parameters

Parameter (Micron)	Note
`config add ext_ram_t_rc.BCM8848X=45000`	Active-to-active command period, which is required to be a minimum of 45 nanoseconds (ns). The parameter is set to 45,000 picoseconds (ps).
`config add ext_ram_t_rcdrtr.BCM8848X=1c`	Active-to-RDTR command delay, which is required to be a minimum of 1 tCK. The parameter is set to 1c, meaning one clock period (c).

Keep track of timing parameters updates in each SDK release. If an SDK release includes any timing parameters updates, a new version of the timing parameters is published. For example, the name of the timing parameter file changes from `_1V3` to `_1V4`.

NOTE: Be careful when changing the DRAM timing parameters. Additionally, changes to the parameters may require extensive testing.

If the SDK includes updated timing parameters, and tune parameters are saved and used during power up, the following steps are required:

1. Retune with the new parameters at nominal temperature.
2. Resave the tune parameters.

The latest information about GDDR6 device selection is in the GDDR6 section of the *Hardware Design Guidelines for StrataDNX ™ 16-nm Devices* (DNX16-AN1xx).

The following table summarizes the Broadcom SDK GDDR6 device support. For new GDDR6 devices that are not listed in the table, open a CSP case to request additional support.

Table 6: GDDR6 SDK Support and Timing Parameters File

Memory Vendor (Density/Die)	Part Number	SDK Support	Timing Parameters File
Samsung (8 Gb/C-die)	K4Z80325BC-HC14 K4Z80325BC-HC16	Supported	`GDDR6_SAMSUNG_K4Z80325BC_HC14_8GBIT.soc`
Samsung (16 Gb/M-die)	K4ZAF325BM-HC14 K4ZAF325BM-HC16	From SDK 6.5.25	`GDDR6_SAMSUNG_K4ZAF325BM_HC14_16GBIT.soc`
Samsung (16 Gb/C-die)	K4ZAF325BC-SC16	From SDK 6.5.25	`GDDR6_SAMSUNG_K4ZAF325BM_HC14_16GBIT.soc`
Micron (8 Gb)	Micron MT61M256M32JE-10N:A Micron MT61M256M32JE-12N:A	Supported	`GDDR6_MICRON_MT61M256M32_12N_8GBIT.soc`
Micron (16 Gb)	MT61M512M32KPA-12N:C MT61M512M32KPA-14N:C	From SDK 6.5.24	`GDDR6_MICRON_MT61M512M32_14N_16GBIT.soc`
Micron (16 Gb)	MT61M512M32KPA-12AAT:E	From SDK 6.5.31	`GDDR6_MICRON_MT61M512M32_14N_16GBIT.soc` **NOTE:** The parameters in this file have been updated to support the device.

The procedure in the following example shows how to add the Samsung GDDR6 16G M-die timing parameters:

1. Navigate to the SDK directory …`/6.5.31/ga/sdk-all-6.5.31/rc/dnx_dram`.
2. Locate the relevant file for the GDDR6 device on the board. In this example, the file is `GDDR6_SAMSUNG_K4ZAF325BM_HC14_16GBIT.soc` (see Table 6).
3. Add an SOC property to enable the timing parameters for the selected device:
   `config add dram_type_GDDR6_SAMSUNG_K4ZAF325BM_HC14_16GBIT_1V0=1`

NOTE:

- Timing parameters may be updated periodically due to various reasons, such as updates from the memory vendor. If an update occurs, the version (in this example `_1V0`) changes.
- The `GDDR6_SAMSUNG_K4ZAF325BM_HC14_16GBIT.soc` file contains definitions for each set of parameters for the specific device:

```
if $?dram_type_GDDR6_SAMSUNG_K4ZAF325BM_HC14_16GBIT_1V0 "\
    config add ext_ram_t_rfc.BCM8848X=120000;\
    config add ext_ram_t_rc.BCM8848X=51000;\
    config add ext_ram_t_rcd_wr.BCM8848X=11000;\
    config add ext_ram_t_rcd_rd.BCM8848X=17000;\
    …
    …
    …
    config add ext_ram_temp_read_enable.BCM8848X=1;"
```

3 Tuning

A tune must be run on an enabled interface before accessing it. For proper GDDR6 operation, the interface must be tuned on each assembled board. Each board must be individually tuned because the tune calibrates each board according to the specific StrataDNX™ and GDDR6 devices. The tune sets the optimal operation point for different types of DRAM access, for example, read and write transactions.

The following two tuning modes are supported during device initialization:

- Restore tune parameters – On driver initialization, load saved tune parameters from a file and run a partial tune.
- Run tune – Perform a full tune at driver initialization.

In both modes, after the tune is performed, the system dynamically tracks variations in temperature and voltage. The variations should be within device specifications. See Section 5, GDDR6 PVT Testing and refer to the BCM88480 data sheet and the GDDR6 data sheet.

Table 7: Tuning Mode Considerations

Considerations	Restore Tune Parameters	Run Tune
Tune time	Faster than a full tune.	Requires more than a minute per interface.
Preconditions	Generate tune parameters at nominal conditions for each assembled device and save the parameters in a file to use during driver initialization.	Any temperature and voltage conditions that comply with those in the device data sheet.
Limitations	Requires maintenance. Parameters might be changed if any of the following events occur: ■ Broadcom device replacement ■ DRAM device replacement ■ Major change in DDR driver ■ DDR parameter update	—

`dram_phy_tune_mode_on_init` SOC value	`RESTORE_TUNE_PARAMETERS_FROM_SOC_PROPERTIES`
	or
	`RESTORE_TUNE_PARAMETERS_OR_RUN_TUNE`
		`RUN_TUNE`

NOTE: Tune parameters are per device and cannot be copied between devices or boards.

The tuning mode is set using an SOC property (Section 2.3, GDDR6 Tune (Shmoo) Options for Driver Initialization).

For normal operation, use the following setting:

```
dram_phy_tune_mode_on_init.BCM8848X=RESTORE_TUNE_PARAMETERS_FROM_SOC_PROPERTIES
```

For tuning during driver initialization, use the following setting:

```
dram_phy_tune_mode_on_init.BCM8848X=RUN_TUNE
```

3.1 Generating Tune Parameters to Restore

Perform the following steps to generate tune parameters to be restored during driver initialization:

1. Configure the SOC setting to run a tune on driver initialization.
   a. Set the SOC property to run the tune at driver initialization.
      ```
      File config-q2a.bcm: dram_phy_tune_mode_on_init.BCM8848X=RUN_TUNE
      ```
   b. Add the SOC property to view detailed information during driver initialization.
      ```
      File dnx.soc: debug soc ddr info
      ```
2. Set the system to nominal temperature and voltage conditions, and initialize the SDK.
3. Check the extended stages of the tune log, check the CDR status, and look for any errors reported during driver initialization.

   If no errors occur, continue to the next step. If errors occur, stop the sequence and review the information in Section 4.3, BIST Debug (if Errors Occur).
4. Save tune parameters in a file.

   An example of a file name for saved tune parameters is `BCM8848X_dram_tune.soc`.

   The following command shows how to save tune parameters to file.
   ```
   BCM> config save filename=./BCM8848X_dram_tune.soc pattern=g6phy16_tune
   ```
5. Edit the saved results to fit the format for SOC properties.

   Add the text `config add` to every line in the `./BCM8869X_dram_tune.soc` file created in Step 4. Every line should use a format similar to the following:
   ```
   config add g6phy16_tune_<param>.<device>=<value>
   ```
   Example:
   ```
   config add g6phy16_tune_aq_l_max_vdl_addr_dram0.0=0x0000100E
   ```
   The following Linux shell command is an example of how to add `config add` to each line in a file:
   ```
   Linux Shell> sed -i -e "s/^/config add /g" ./BCM8848X_dram_tune.soc
   ```
   The following Linux shell command is an example of how to sort the `./BCM8869X_dram_tune.soc` file:
   ```
   Linux Shell> sort -o ./BCM8848X_dram_tune.soc ./BCM8848X_dram_tune.soc
   ```
   Sorting the file can make it easier to read the file and compare multiple files.
6. Link the file with the saved parameters to enable them to be loaded during driver initialization.

Another option is to use the `import` command instead of SOC properties. In the following procedure, Step 1 through Step 3 are identical to the same steps in the previous procedure.

Perform the following steps to generate tune parameters to be restored during driver initialization (using `import` instead of SOC properties):

1. Configure the SOC setting to run a tune on driver initialization.
   a. Set the SOC property to run the tune at driver initialization.
      ```
      File config-q2a.bcm: dram_phy_tune_mode_on_init.BCM8848X=RUN_TUNE
      ```
   b. Add the SOC property to view detailed information during driver initialization.
      ```
      File dnx.soc: debug soc ddr info
      ```
2. Set the system to nominal temperature and voltage conditions and initialize the SDK.
3. Check the extended stages of the tune log, check the CDR status, and look for any errors reported during driver initialization.

   If no errors occur, continue to the next step. If errors occur, stop the sequence and review the information in Section 4.3, BIST Debug (if Errors Occur).
4. Save tune parameters in a file.

   An example of a file name for saved tune parameters is `BCM8848X_dram_tune.bcm`.

   The following command shows how to save tune parameters to file.
   ```
   BCM> config save filename=./BCM8848X_dram_tune.bcm pattern=g6phy16_tune
   ```
5. Sort the saved results (optional and not required for operation).

   The following Linux shell command is an example of how to sort the `./BCM8869X_dram_tune.bcm file`:
   ```
   Linux Shell> sort -o ./BCM8848X_dram_tune.bcm ./BCM8848X_dram_tune.bcm
   ```
   Sorting the file can make it easier to read the file and compare multiple files.
6. Link the file with the saved parameters to enable them to be loaded during driver initialization.
   ```
   import ./BCM8848X_dram_tune.bcm
   ```

3.2 Manual Tune For Debug

When auto tuning fails or when attempting different debug scenarios, manually tune the DRAM interface to collect tuning logs for further debugging.

The following steps show an example of how to perform a manual tune on the DDR0 and DDR1 interfaces:

1. Set the SOC property to skip tuning on initialization.
   a. Set the SOC property.
      ```
      File config-q2a.bcm: dram_phy_tune_mode_on_init.BCM8848X=SKIP_TUNE
      ```
   b. Add the SOC property to view detailed information during driver initialization.
      ```
      File dnx.soc: debug soc ddr info
      ```
2. Set the temperature and voltage within the device specification and within the specifications for other board devices.
3. Run the tune manually.
   ```
   BCM> debug soc ddr info // enable full print information related to ddr
   BCM> dnx dram debug redirectToOCB enable=1 // disable traffic to dram
   BCM> dnx dram debug eyeScan dram=0 channel=0 // run tune on DDR0 interface channel 0
   BCM> dnx dram debug eyeScan dram=0 channel=1 // run tune on DDR0 interface channel 1
   BCM> dnx dram debug eyeScan dram=1 channel=0 // run tune on DDR1 interface channel 0
   BCM> dnx dram debug eyeScan dram=1 channel=1 // run tune on DDR1 interface channel 1
   ```

4 GDDR6 BIST

The DNX device includes a BIST block to allow different patterns to be generated on the DDR interface. BIST verifies device connectivity and IO functionality. In a limited way, a pseudo-random binary sequence (PRBS) BIST can mimic real traffic; however, BIST activates only the PHY I/Os and DRAM devices. It does not activate all related blocks that are active in an actual traffic use case.

Passing BIST is a gating test before running real, randomized traffic during conditions involving variations of temperature and voltage and different device tests.

DRAM BIST is run by default after each tune at power-up. If DRAM BIST errors occur, error messages print to the screen; otherwise, no messages appear.

For GDDR6 devices larger than 8 Gb, only the lower 8 Gb of the GDDR6 device are used. At power up, BIST tests only the used portion of the GDDR6 device.

In most cases, DRAM BIST is used for debug or for connectivity sanity checks.

NOTE: Before running BIST, disable the ability to read the temperature of the GDDR6 device. If temperature reading is enabled, an error might occur. For information, see Section 2.4, Enable or Disable GDDR6 Temperature Reading. Alternately, disable any process that monitors the temperature.

4.1 Running BIST

NOTE: BIST should run at a specific temperature and voltage. Changing temperature or voltage conditions is not allowed while BIST runs or between runs. If device conditions change, rerun the tune before running BIST. If conditions change during BIST, ignore the BIST results.

The steps in this section that show how to perform the following four types of DRAM BIST runs:

- Short BIST – Use the default BIST run with a short number of random read and write (about 200 transactions).
- Long BIST – In the example, run random read and write transactions for 60 seconds (the duration is configurable).
- all_address BIST – Read and write to all possible address of the DRAM.
- IO_stress BIST – Run a predefined pattern to stress, at most, the I/O.

Table 8: Common DRAM BIST Run Results

Short BIST	Long BIST	all_address BIST	IO_Stress BIST	Interpreting the Results
Pass	—	—	Fail	May indicate power supply issues. Try to run on one channel only or reduced channels to see if BIST still fails. May also indicate on some marginality sourced by signal integrity or power integrity. Check supply rails for related noise and review layout for coupling. Try to run with reduced channels to see if it relates to the number of channels or coupling between channels, and so on.
Fail	Fail	—	—	Basic operation issues that, in most cases, will fail the tune. Check basic connectivity, power, VREF, clocks, reset, JTAG, calibration resistors, and so on.
Pass	Fail	—	—	May indicate power supply issues. Try to run on one channel only or reduced channels to see if BIST still fails. May also indicate on some marginality sourced by signal integrity or power integrity. Check supply rails for related noise and review layout for coupling. Try to run with reduced channels to see if it relates to the number of channels or coupling between channels, and so on.
Pass	Pass	Fail	—	May indicate a DRAM with a singular issue on a specific address. Run the test several times to see if it fails with the same address.

The following steps show an example of how to perform BIST runs. Use the sequence to test several different lengths of times or to run BIST with extended times between start and stop.

1. Set the SOC property to skip tuning on initialization.
   a. Set the SOC property.
      ```
      File config-q2a.bcm: dram_phy_tune_mode_on_init.BCM8848X=SKIP_TUNE
      ```
   b. Set the SOC property to disable DRAM temperature reading.
      ```
      File config-q2a.bcm: ext_ram_temp_read_enable.BCM8848X=0
      ```
   c. Add the SOC property to view detailed information during driver initialization.
      ```
      File dnx.soc: debug soc ddr info
      ```
2. Power cycle the system and set the temperature and voltage within the device specification and within the specifications for other board devices.
3. Disable temperature reading from the GDDR6 devices. In other words, disable any background process or other feature that reads the GDDR6 device temperature.
4. Run the tune manually.
   ```
   BCM> debug soc ddr info                           // enable full print information related to ddr
   BCM> dnx dram debug redirectToOCB enable=1        // disable traffic to dram
   BCM> dnx dram debug dynamicCalibration enable=0   // disable dynamic calibration
   BCM> dnx dram debug eyeScan                       // run tune
   ```
5. Run a short BIST.
   ```
   BCM> dnx dram bist print // ignore first read counters
   BCM> dnx dram bist print // read counters and print results, verify all are zero
   BCM> dnx dram bist start // start short BIST
   ```
6. Run a long BIST.
   ```
   dnx dram bist print         // ignore first read counters
   dnx dram bist print         // read counters and print results, verify all are zero
   dnx dram bist start count=0 // start infinite/long BIST
   sleep 60                    // sleep 60 sec
   dnx dram bist stop          // stop BIST run
   dnx dram bist print         // read counters and print result. Verify all zero
   ```
   The `sleep 60` parameter specifies a 60-second BIST run. This parameter can be extended to any value to add more stress to the interface, or it can run in a loop to run more times. Run the test overnight on several boards (more than three).
7. Write and read all DRAM addresses.
   ```
   BCM.0> dnx dram debug bist type=all_address
   ```
8. Perform an I/O stress test.
   ```
   BCM.0> dnx dram debug bist type=IO_stress
   ```
9. Run Step 5 and Step 6 in a loop or any other combination.

To return to normal operation, do the following:

1. Revert the tune mode at driver initialization to allow the tune to use restored parameters (or fail to tune if the parameters are missing).
   ```
   dram_phy_tune_mode_on_init.BCM8848X=RESTORE_TUNE_PARAMETERS_FROM_SOC_PROPERTIES
   ```
   This step assumes the restore parameters have already been generated and defined.
2. Perform a power cycle.

4.2 BIST Pass and Fail Examples

Both DRAM and all channels pass in the following BIST print example. The data error counter column (highlighted green) shows values equal to zero.

```
=================================================================================================================================
|                                                    Dram BIST results                                                          |
=================================================================================================================================
| Dram index | Channel | Write command counter | Read command counter | Read data counter  | Data error counter | Data error bits |
=================================================================================================================================
| 0          | 0       | 0x0000000289245c00    | 0x0000000289245bfb   | 0x0000000289245bfb | 0x00000000         | 0x00000000      |
| 0          | 1       | 0x0000000289224600    | 0x000000028922452d   | 0x000000028922452d | 0x00000000         | 0x00000000      |
| 1          | 0       | 0x0000000289203b00    | 0x0000000289203a41   | 0x0000000289203a41 | 0x00000000         | 0x00000000      |
| 1          | 1       | 0x00000002891e7b00    | 0x00000002891e7a2d   | 0x00000002891e7a2d | 0x00000000         | 0x00000000      |
=================================================================================================================================
```

Both DRAM and all channels fail in the following example. The data error counter column (highlighted red) shows values that are not equal to zero.

```
=================================================================================================================================
|                                                    Dram BIST results                                                          |
=================================================================================================================================
| Dram index | Channel | Write command counter | Read command counter | Read data counter  | Data error counter | Data error bits |
=================================================================================================================================
| 0          | 0       | 0x0000000000000000    | 0x0000000000000000   | 0x00000000000076fe | 0x0000121a         | 0x0000ffff      |
| 0          | 1       | 0x0000000000000000    | 0x0000000000000000   | 0x00000000000076f8 | 0x00001784         | 0x0000ffff      |
| 1          | 0       | 0x0000000000000000    | 0x0000000000000000   | 0x00000000000076f8 | 0x00001f98         | 0x0000ffff      |
| 1          | 1       | 0x0000000000000000    | 0x0000000000000000   | 0x00000000000076f2 | 0x000010fa         | 0x0000ffff      |
=================================================================================================================================
```

4.3 BIST Debug (if Errors Occur)

The following steps show how to run BIST per-DRAM and per-channel, which reduces the coupling effect between interfaces and channels, if present. Running these types of tests can also help identify connectivity or coupling issues between DDR channels.

1. Set the SOC property to skip tuning on initialization.
   a. Set the SOC property.
      `File config-q2a.bcm: dram_phy_tune_mode_on_init.BCM8848X=SKIP_TUNE`
   b. Set the SOC property to disable DRAM temperature reading.
      `File config-q2a.bcm: ext_ram_temp_read_enable.BCM8848X=0`
   c. Add the SOC property to view detailed information during driver initialization.
      `File dnx.soc: debug soc ddr info`
2. Set the temperature and voltage to nominal conditions.
3. Disable temperature reading from the GDDR6 devices. In other words, disable any background process or other feature that reads the GDDR6 device temperature.
4. Run the tune manually.

```
BCM> debug soc ddr info                          // enable full print information related to ddr
BCM> dnx dram debug redirectToOCB enable=1       // disable traffic to dram
BCM> dnx dram debug dynamicCalibration enable=0  // disable dynamic calibration
BCM> dnx dram debug eyeScan                      // run tune
```

5. Run a short (duration) BIST.

```
BCM> dnx dram bist print
BCM> dnx dram bist print
BCM> Echo "short BIST…"

BCM> echo "short BIST dram 0"
BCM> dnx dram bist start dram=0 channel=0 // run short BIST on DDR0 Channel 0
BCM> dnx dram bist start dram=0 channel=1 // run short BIST on DDR0 Channel 1
BCM> dnx dram bist start dram=0           // run short BIST on DDR0 Channel 0 and Channel 1

BCM> echo "short BIST dram 1"
BCM> dnx dram bist start dram=1 channel=0
BCM> dnx dram bist start dram=1 channel=1
BCM> dnx dram bist start dram=1

BCM> echo "short BIST dram 0 and dram 1"
BCM> dnx dram bist start // run short BIST on DDR0 and DDR1
```

6. Repeat Step 5 two more times.
7. Run a long (duration) BIST.

   The setting in this example runs BIST for 60 seconds. The amount of time can be increased for runs that introduce more stress.

   The commands in this step are similar to the short BIST commands but include the additional setting `count=0`, which represents an infinite run.

```
BCM> dnx dram bist print
BCM> dnx dram bist print
BCM> Echo "Long BIST…"

BCM> Echo "long BIST dram 0"
BCM> Echo "----------------"
```

```
BCM> Echo "long BIST dram 0 Channel 0"
BCM> dnx dram bist start dram=0 channel=0 count=0 // infinite run DDR0 channel 0
BCM> Sleep 60                                     // sleep 60 sec
BCM> dnx dram bist stop                           // stop infinite run
BCM> dnx dram bist print                          // print counters, verify error counters are 0
BCM> dnx dram bist print                          // print cleared counters, verify all 0

BCM> Echo "long BIST dram 0 Channel 1"
BCM> dnx dram bist start dram=0 channel=1 count=0 // infinite run DDR0 channel 1
BCM> Sleep 60
BCM> dnx dram bist stop
BCM> dnx dram bist print
BCM> dnx dram bist print

BCM> dnx dram bist start dram=0 count=0 // infinite run DDR0 Channel 0 and Channel 1
BCM> Sleep 60
BCM> dnx dram bist stop
BCM> dnx dram bist print
BCM> dnx dram bist print

BCM> Echo "long BIST dram 1"
BCM> Echo "----------------"
BCM> Echo "long BIST dram 1 Channel 0"
BCM> dnx dram bist start dram=1 channel=0 count=0
BCM> Sleep 60
BCM> dnx dram bist stop
BCM> dnx dram bist print
BCM> dnx dram bist print

BCM> Echo "long BIST dram 1 Channel 1"
BCM> dnx dram bist start dram=1 channel=1 count=0
BCM> Sleep 60
BCM> dnx dram bist stop
BCM> dnx dram bist print
BCM> dnx dram bist print

BCM> dnx dram bist start dram=1 count=0
BCM> Sleep 60
BCM> dnx dram bist stop
BCM> dnx dram bist print
BCM> dnx dram bist print

BCM> Echo "long BIST dram 0 and dram 1"
BCM> Echo "----------------"
BCM> dnx dram bist start count=0
BCM> Sleep 60
BCM> dnx dram bist stop
BCM> dnx dram bist print
BCM> dnx dram bist print
```

8. Repeat Step 7 two more times.

When opening a CSP case to report a DDR issue, include the BCM shell commands used in the debug as well as the output from the following information dumps:

- Configuration dump
  `BCM> config show`
- MR register dump
  `BCM> dbal table dump table=GDDR6_DRAM_MODE_REGISTERS`
- Full tune log
  See Step 1 in Section 3.1, Generating Tune Parameters to Restore.

5 GDDR6 PVT Testing

After selecting the tune mode of operation during driver initialization for the production devices, run randomized traffic on different devices under different temperature and voltage conditions. For more details on tune options, see Section 2.3, GDDR6 Tune (Shmoo) Options for Driver Initialization.

During testing, do not exceed the following temperature and voltage changes:

- Temperature changes that are larger than 2°C per minute (2°C/min)
- Voltage changes that are larger than 2 mV per minute (2 mV/min)

6 Read GDDR6 Device MR Registers

An MR register dump might be useful when reviewing the GDDR6 settings with a memory vendor or during debug. The dump values are shadow values of the written values to the GDDR6 MR registers.

Do not modify MR registers. If MR registers must be modified, open a CSP case with Broadcom.

The following commands show various methods for dumping MR settings:

- `BCM.0> acc read DCC_HBM_MODE_REGISTERS f`
- `BCM> dbal table dump table=GDDR6_DRAM_MODE_REGISTERS`
- `BCM>g DCC_HBM_MODE_REGISTERS`

7 Broadcom DRAM Support (CSP)

The following steps describe how to enlist help from Broadcom support in debugging DRAM issues:

1. Open a CSP case for support.
2. Add the company project and board names.
3. Add cases that include the following reviews:
   – Schematic review
   – Layout review
   – Connectivity review
   – Length check review
4. Add the following information to the case:
   a. Board pass and fail statistics with as many details as possible.
      - Board information, such as the name and serial number for each board.
      - DDR information, such as the part number.
      - SDK information, such as the version.
   b. Tune log and BIST debug log from at least three boards. See Section 4.3, BIST Debug (if Errors Occur).
   c. Add a log with the `BCM>config show` output.
5. If the DRAM fails during traffic, reproduce the issue and run the `dnx dram debug status` command.

   ```
   BCM.0> dnx dram debug status
   ```

   Provide the command dump to Broadcom support.

NOTE:

- To debug DRAM errors, the error should be reproducible.
- The error should be present on multiple boards. If an error is observed on only one board, the error is most likely an assembly issue and not a system issue. To debug system issues, the issue should be reproduced on several boards.

Appendix A: BCM88290 DDR Differences

The DDR-related differences between the BCM88480/BCM88820 and BCM88290 devices are shown in the following table.

NOTE: For the BCM88290 device, disregard all examples in this document that run on DDR0. Run commands only on DDR1.

Table 9: BCM88480/BCM88280 and BCM88290 DDR Differences

DDR-Related Item	BCM88480/BCM88280	BCM88290
Interface	DDR0 and DDR1	DDR1
`ext_ram_bitmap` SOC setting	3: Both DDR interfaces are enabled	2: The DDR interface is enabled
Operation speed (per DQ bit)	6.8 Gb/s	6.0 Gb/s
`ext_ram_freq` SOC setting	850	750

Related Documents

The references in this section may be used with this document. The documents are available on the Broadcom Customer Support Portal (docSAFE).

For Broadcom documents, replace the “xx” in the document number with the largest number available in the repository to ensure that you have the most current version of the document.

Document Name	Number
BCM88480 data sheet: *800-Gb/s Integrated Packet Processor and Traffic Manager Single-Chip Device*	88480-DS1xx
BCM88280 data sheet: *360-Gb/s Integrated Packet Processor and Traffic Manager Single-Chip Device*	88280-DS1xx
BCM88290 data sheet: *80-Gb/s Integrated Packet Processor and Traffic Manager Single-Chip Device*	88290-DS1xx
Hardware Design Guidelines for StrataDNX™ 16-nm Devices	DNX16-AN1xx
DNX16 Reference Length for GDDR6	DNX16 GDDR6 Length Excel vX.X.zip
Traffic Manager Programming Guide	88690-PG2xx
BCM88480 Device Errata	88480-ER1xx
BCM88280 Device Errata	88280-ER1xx