Application Note

Debugging AFE7950 for Run Time and Post Bring-Up Failure

Dhruvil Solanki and Nikhil Jain

ABSTRACT

This application note describes the systematic procedure to identify, address and resolve bugs encountered during AFE bring-up and thereby optimize the overall efficiency and reliability of system. In the AFE bring-up process, thorough validation is conducted through read checks and register polling at various stages of bring-up to verify the fulfillment of prerequisites for progression. Additionally, the polling register serve as the acknowledgment bit to indicate successful execution and making sure integrity of AFE functionality during bring-up flow. When encountering bring-up failure, they are stem from failed read checks or poll failures.

Also, the application note describes the debugging strategy for issues seen post bring-up like SERDES/ JESD Link instability, TX tone or RX capture problem.

Table of Contents

Trademarks

All trademarks are the property of their respective owners.

1 Introduction

AFE bring-up involves a systematic and Top to Bottom configuration process. For the ease of splitting the step in the configuration file, the bring-up file is divided into multiple steps. The section configured earlier plays crucial role in the subsequent step of bring-up. Detail for each step mentioned below is available in AFE79xx_ConfigurationGuide under Bring-Up Flow and Log File section.

Bring-up Flow:

rstDevice, fuseChain, mcuWakeUp, pllEfuse, pllConfig, serdesConfig, topConfig, sysConfig, configTune, analogWrites, jesdConfig, agcConfig, miscConfig, gpioConfig, sysrefJesdLinkup, postLinkUp, dlJesdLinkupCheck

In the AFE bring-up process, thorough validation is conducted through read checks and register polling at various stages of bring-up. The following is the format definition of the SPI command in AFE7950 configuration file:

SPIWrite Addr, valuetoWrite, LSB, MSB: This command is used to do SPI write for Addr in AFE and addr is up to 15bits, value toWrite is Value to be write for mentioned LSB to MSB bits.

SPIRead Addr, LSB, MSB: This command reads the value from mention addrs for the set LSB to MSB bits.

SPIBurstWrite starting Address, [array of the value to written in the incremental address]: This command burst writes for the AFE, starting address is mentioned and the array indicates the value to write for each incremental address.

SPIReadCheck Addr, LSB, MSB, Expectedvalue: Read check command verifies if the readout of the register matches the expected value. It is a onetime check. Error to read the expected value cause failure.

SPIPoll Addr, LSB, MSB, Expectedvalue: Poll check command verifies the readout of the register repeatedly for certain set period of time until it reads the value, timeout or fails in case of readout not expected.

2 SPI Failure During Bring-Up

2.1 Detail Regarding Chip Readouts

There are three different chip identification checks for AFE. chip_type, chip_id, chip_ver. Information about what chip readout to expect in part of bring-up generation libs and the read check command is embedded with information about Address, bit size and readout expected.

chip_type = Indicates type of part # 0xa = AFE

chip_id = Indicates chip Id # 0x78 = AFE79xx

chip_ver = Indicates chip version

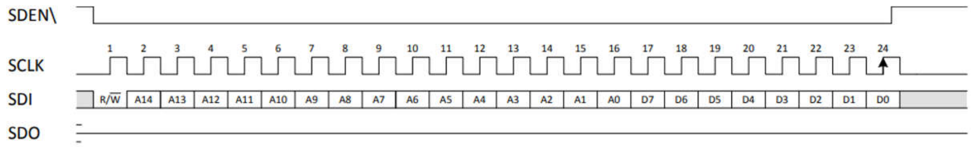
```
SPIReadCheck 0003,0,7,0a //Read    chip_type=0xa;     Address(0x3[7:0])
SPIReadCheck 0004,0,7,78
SPIReadCheck 0005,0,7,00 //Read    chip_id=0x78;     Address(0x4[7:0],0x5[7:0])
SPIReadCheck 0006,0,7,20 //Read    chip_ver=0x20;     Address(0x6[7:0],0x7[7:0])
```

2.2 Failure and Fix for Chip Read Check

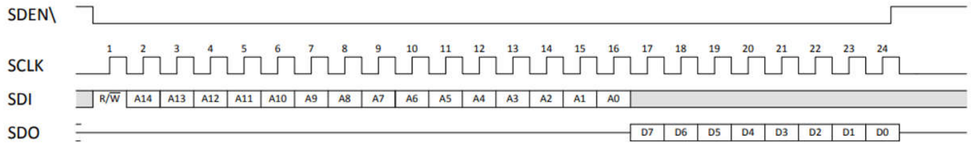
1. Chip readout as 0x0 or 0xff:
   a. We can check if SPI is working correctly. Make sure the Address length is 16, packet length is 24, packet order as Address first, packet type as MSB first, enable state as Active low, Data latch on positive edge for write and negative edge for read also check the physical SPI Driver connection and SPI and GPIO Logic for AFE works at 1.8V.
   b. If all the settings are as expected, next step is to probe the SEN, SCLK, SDI, SDO and check for waveform level and timing data to make sure proper functionality for SPI Driver and AFE.
   c. If Timing and Level of SEN, SCLK and SDI is expected and if AFE is not responding and if SDO is low. Do HW Reset of the AFE and write SPI write for configuring 4 wire or 3 wire mode depend upon your system need.

      SPIWrite 0000,30,0,7 //Bit 4 (1: 4 Pin control; 0: 3 Pin control)

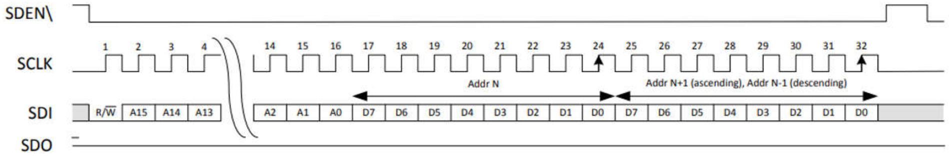
d. Check for device input voltage levels and check for device reset state current level.


SDEN\
SCLK
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24
SDI
R/W A14 A13 A12 A11 A10 A9 A8 A7 A6 A5 A4 A3 A2 A1 A0 D7 D6 D5 D4 D3 D2 D1 D0
SDO

2. Incorrect LSB readout: Check for the above-mentioned SPI setting and check the timing of the SDO, SCLK and SEN. For debug purpose we can call any of the chip readout separately and Probe the SPI Pins. SEN need to be held one more extra clock cycle with the last SCLK edge.



3. Before moving forward with the AFE bring-up process, it is important to verify the implementation of SPI Burst write. This is essential because SPI Burst write is utilized in various aspects of the AFE bring-up, and any issue with the implementation can potentially lead to macro error later on. There is certain macro operation which involves loading and verifying the success of burst write operation. To verify we can use a fix bus write sequence and read the register sequentially. SPIBurstWrite 0010, [01,02,03,04,05,06,07,08,09,0A] then, read SPI address from 0x10 to 0x19 and confirm if readout is as expected. Later set all address back to zero.


SDEN\
SCLK
1 2 3 4 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32
Addr N
Addr N+1 (ascending), Addr N-1 (descending)
SDI
R/W A15 A14 A13 A2 A1 A0 D7 D6 D5 D4 D3 D2 D1 D0 D7 D6 D5 D4 D3 D2 D1 D0
SDO

2.3 Poll Check for SPI Access for PLL Page

When in AFE bring up if we try to write or read in PLL Page, the device needs to internally request SPI access for PLL Page and SPIPoll 0171,0,0,01 is the Poll check to get information about if SPI has got the access to PLL page.

2.4 Failure and Fix for the SPI Poll Check for PLL Page Access

Check for voltage on the AFE power nets (0.925V, 1.2V and 1.8V) are in expected range as per data sheet recommendation.

2.5 Read Check Indicating Status of Fuse Farm Autoload

EFuse auto load is done and checked for any autoload error.

Fuse Farm autoload read check is to verify the fuse are loaded correctly.

```
SPIReadCheck 0150,0,3,0f //Read    obs_func_spi_chain_autoload_done=0xf;
SPIReadCheck 0150,4,7,00 //Read    obs_func_spi_chain_autoload_error=0x0;
SPIReadCheck 0160,0,3,0f //Read    obs_func_spi_chain_autoload_done=0xf;
SPIReadCheck 0160,4,7,00 //Read    obs_func_spi_chain_autoload_error=0x0;
```

2.6 Failure and Fix for Autoload Read Check

The most common reason for failure of autoload read check is if reference clock is not reaching the device. Autoload operation work on clock which is derived after dividing reference clock inside AFE. So, it is important to check if the reference clock is reaching the AFE pin with in expected voltage range also check if 1.2V common mode is present at reference clock AFE side pin (common mode is forced internally).

Check for voltage on the AFE power nets (0.925V, 1.2V and 1.8V) are in expected range as per data sheet recommendation.

3 Macro Failure Breaking the Bring-Up Flow

3.1 Read Check for Macro Error and Poll Check for Macro Done

1. The device is configured through SPI using a combination of direct register reads/writes for simple configurations, and register writes to initiate macros. Macro commands abstract out the internal device configuration sequence to a simple set of configurations and simplify the host interaction. The commands reduce complex configurations into simple writes, avoid computation complexity on the host side and provide simple status information in response. Macro command is used at multiple places in bring-up.
2. Macro Opcode: Operation code which informs AFE the operation to be performed. For differential operations of AFE, different Macro opcode is defined.
   Macro Operand: Operand are values or expressions that is used to perform an operation in AFE.
3. The first SPIPoll, Poll for Bit 0 from Address 0xf0 which indicates if Macro/MCU is ready for a next operation. If the Poll fails, the MCU is still not ready for new operation and if we run a new macro operation the MCU can fail to execute.
4. Later if Macro ready passes, operand and opcode are loaded.
5. After that Macro Done status is polled #SPIPoll 00f0,2,2,04. This is to check if macro operation is complete.
6. After that read check for 0xf0 bit 3 is read to check if macro operation was executed error free or had error.
7. Bring-up can break if any of the this check fail. The following discusses if macro error and failure happen how to figure out the error and resolve the issue.

Sample Code

```
SPIPoll 00f0,0,0,01            //MACRO_READY
SPIWrite 00a3,00,0,7    //MACRO_OPERAND_REG
SPIWrite 00a2,00,0,7    //MACRO_OPERAND_REG
SPIWrite 00a1,00,0,7    //MACRO_OPERAND_REG
SPIWrite 00a0,02,0,7    //MACRO_OPERAND_REG
SPIWrite 0193,01,0,7    //MACRO_OPCODE=0x1;
WAIT 0.001
SPIRead 00f0,2,2    //Read    MACRO_DONE=0x1;
SPIPoll 00f0,2,2,04
SPIReadCheck 00f0,3,3,00    //Read    MACRO_ERROR=0x0;
SPIRead 00f1,0,7    //Read    MACRO_ERROR_OPCODE=0x0;
SPIRead 00f0,4,4    //Read    MACRO_ERROR_IN_OPCODE=0x0;
SPIRead 00f0,5,5    //Read    MACRO_ERROR_OPCODE_NOT_ALLOWED=0x0;
SPIRead 00f0,6,6    //Read    MACRO_ERROR_IN_OPERAND=0x0;
SPIRead 00f0,7,7    //Read    MACRO_ERROR_IN_EXECUTION=0x0;
SPIRead 00f3,0,7    //Read    MACRO_ERROR_EXTENDED_CODE=0x0;
SPIRead 00f2,0,7    //Read    MACRO_ERROR_EXTENDED_CODE=0x0;
```

3.2 Failure and Fix for Macro Error and Poll check for Macro Done

1. Macro operation are sensitive to AFE power nets (0.925V, 1.2V and 1.8V). First make sure all voltages are within range and also the current limit for each rail is sufficient up to 3A. It is observed that if the current sourcing is not sufficient then there could be a voltage dip and could result in Macro failure.
2. For some macro operation to be successful, Sysref functionality needs to beokay. It is important to check if Sysref level is reaching device pin as per data sheet specification range and frequency.
3. When failure is in (SPIPoll 00f0,0,0,01) that is, Macro Ready fails means that the macro is still busy and not able to take up new operation. Once the issue is located to which section of macro ready is failing. Some delay (WAIT 1) can be added before the failing macro ready command and checked. In case if the failure still exists check for **point 1**.
4. When failure is in (SPIPoll 00f0,2,2,04) that is, Macro Done is failing means the macro is still working on current operation. Once the issue is located to which section of macro done is failing. Some delay (WAIT 1) can be added before the failing macro done command and checked, in case if the failure still exists check for **point 1**.
5. Now if the failure is in (SPIReadCheck 00f0,3,3,00) then it means there has happened a macro error during the current macro operation. We have readout register which gives more detail on what type of error has

happen, we can read below writes to get more insight on error and for more detail can check TRM document and parallelly check for **point 1**.

```
SPIWrite 0018,20,0,7
SPIRead 00f1,0,7    //Read    MACRO_ERROR_OPCODE=0x0;
SPIRead 00f0,4,4    //Read    MACRO_ERROR_IN_OPCODE=0x0;
SPIRead 00f0,5,5    //Read    MACRO_ERROR_OPCODE_NOT_ALLOWED=0x0;
SPIRead 00f0,6,6    //Read    MACRO_ERROR_IN_OPERAND=0x0;
SPIRead 00f0,7,7    //Read    MACRO_ERROR_IN_EXECUTION=0x0;
SPIRead 00f3,0,7    //Read    MACRO_ERROR_EXTENDED_CODE=0x0;
SPIRead 00f2,0,7    //Read    MACRO_ERROR_EXTENDED_CODE=0x0;
SPIRead 00f4,0,7    //Read    MACRO_ERROR_EXTENDED_CODE=0x0;
SPIRead 00f5,0,7    //Read    MACRO_ERROR_EXTENDED_CODE=0x0;
SPIWrite 0018,00,0,7
```

6. Effective execution of the for 0x78 macro opcode, hinges on precise implementation of SPI Burst writes.

4 AFE PLL Failure

4.1 Read Check for PLL Lock

PLL Lock status is read to check if device main PLL is locked and is stable.

```
SPIReadCheck 0066,4,4,10    //Lock
SPIReadCheck 0066,6,6,00    //Lock Lost Sticky
```

We monitor two specific bits, Bit[4] signifies current Lock status of PLL, while Bit[6] indicates whether PLL lost lock after being lock for first time. This bit indicates instability in PLL after the first time it has locked.

Table 4-1. Reference Clock Electrical Characteristics

f_{PFD}	PFD frequency		100		500	MHz
F_{REF}	Input Clock frequency		0.1		12	GHz
V_{SS}	Input Clock level		0.6		1.8	V_{PPdiff}
Coupling				AC Coupling Only		
	REFCLK input impedance	Parallel resistance		100		Ω
		Parallel capacitance		0.5		pF

4.2 Failure and Fix for Read Check of PLL

The main source of error for PLL read check failure is if the reference clock is not proper. Check if the reference clock is of correct frequency and power level is as expected range at pin also check for the common mode forced from device is around 1.2V. Also, check for phase noise of reference clock in case of Bit 6 is high along with if PLL 1.8V is stable.

5 AFE Internal Sysref Flag Failure

5.1 Read Check Status of Sysref Flag Bit

We have internal clock and sysref flag status to check if the device has registered the pin sysref.

```
SPIReadCheck 012c,3,3,08    // jesd_clk_rx1
SPIReadCheck 0130,3,3,08    // monitor_jesd_sysref_rx1
```

5.2 Failure and Fix for Read Check Status of Sysref Flag Bit

Failure in read check sysref status points towards improper pin sysref. Make sure sysref frequency and level are in expected range. If using pulse sysref mode is selected, common mode voltage must be supplied through external driver along with swing limits. In case of continuous sysref mode is selected, AC coupling capacitor can be used and device can force common mode internally for sysref pins around 0.7V.

Table 5-1. Sysref Electrical Characteristics

Differential Inputs: ±Mode A				
$F_{SYSREFMAX}$	SYSREF Input Frequency Maximum		1	GHz
$V_{SWINGSRMAX}$	SYSREF Input SWING Maximum		1.8	V_{PPdiff}
$V_{SWINGSRMIN}$	SYSREF Input SWING Minimum	f_{REF} < 500MHz	0.3	V_{PPdiff}
		f_{REF} > 500MHz	0.6	V_{PPdiff}
$V_{COMSRMAX}$	SYSREF Input Common Mode Voltage Maximum		0.8	V
$V_{COMSRMIN}$	SYSREF Input Common Mode Voltage Minimum		0.6	V
Z_T	Input Termination	Differential	100	Ω
C_L	Input Capacitance	Each pin to GD	500	pF

6 JESD Link Check Failure

6.1 Multiple Read Checks Indicating Status of JESD Linkup

1. JESD linkup is done at the end of AFE bring-up. We have multiple check points for JESD linkup status of AFE. During the bring-up flow serdes is already linked during the AFE bring-up. Do make sure the FPGA/ASIC STX is transmitting some data so that the CDR of AFE SRX can adapt and achieve serdes linkup between AFE and FPGA/ASIC.

   During JESD Linkup we check for Serdes and JESD linkup and error status. Below is list read check done during bring-up.

   a. SPIReadCheck 0118,0,7,00 #Below is the definition of error indicated for 0x118

      [3:0] = TIED to 0

      [4] = JESD shorttest alarm

      [5] = TIED to 0

      [6] = serdesab_pll_loss_of_lock

      [7] = serdescd_pll_loss_of_lock

   b. SPIReadCheck 0119,0,7,00

      [0] = SRX1 LOS indicator

      [1] = SRX2 LOS indicator

      [2] = SRX3 LOS indicator

      [3] = SRX4 LOS indicator

      [4] = SRX1 Serdes-FIFO error

      [5] = SRX2 Serdes-FIFO error

[6] = SRX3 Serdes-FIFO error

[7] = SRX4 Serdes-FIFO error

c. SPIReadCheck 011a,0,7,00

TIED to 0

d. SPIReadCheck 011b,0,7,00

[3:0] = TIED to 0

[4] = JESDB: Lane0 Frame-Sync error (Ctrl-K in middle of data) JESDC: Lane0 Fixed ones error

[5] = JESDB: Lane1 Frame-Sync error (Ctrl-K in middle of data) JESDC: Lane1 Fixed ones error

[6] = JESDB: Lane2 Frame-Sync error (Ctrl-K in middle of data) JESDC: Lane2 Fixed ones error

[7] = JESDB: Lane3 Frame-Sync error (Ctrl-K in middle of data) JESDC: Lane3 Fixed ones error

e. SPIReadCheck 011c,0,7,00

Below is lane error for JESD 204B protocol for lane 0:

bit7 = JESDB: multiframe alignment error

bit6 = JESDB: frame alignment error

bit5 = JESDB: link configuration error

bit4 = JESDB: elastic buffer overflow (bad RBD value)

bit3 = JESDB: elastic buffer match error. The first non-/K/ does not match 'match_ctrl' and 'match_data' programmed values

bit2 = JESDB: code synchronization error

bit1 = JESDB: 8b/10b not-in-table code error

bit0 = JESDB: 8b/10b disparity error

If we use JESD 204C protocol, below is the lane error mapped to for lane 0:

bit7 = JESDC: EoEMB alignment error

bit6 = JESDC: EoMB alignment error

bit5 = JESDC: cmd-data in crc mode not matching with spi register bits

bit4 = JESDC: elastic buffer overflow (bad RBD value)

bit3 = JESDC: TIED to 0. bit2 = JESDC: extended multiblock alignment error

bit1 = JESDC: sync-header invalid error ('11' or '00' received in expected sync header location)

bit0 = JESDC: sync-header CRC error

f. SPIReadCheck 011e,0,7,00

Same as above lane error mapping for JESD lane 1

g. SPIReadCheck 011d,0,7,00

Same as above lane error mapping for JESD lane 2

h. SPIReadCheck 011c,0,7,00

Same as above lane error mapping for JESD lane 2

i. SPIReadCheck 00ee,0,3,0f

JESDB: comma_align_lock_lane[0:3]monitor_flag

JESDC: sync_header_align_lock_lane[0:3]monitor_flag

j. SPIReadCheck 00a2,0,7,aa

JESDB: CS_STATE value

JESDC: EMB_STATE value

bits(1:0) = Lane0

bits(3:2) = Lane1

bits(5:4) = Lane2

bits(7:6) = Lane3

For stable link, the bits for each lane enabled can read as "10"

k. SPIReadCheck 00a4,0,7,55

JESDB: FS_STATE value

bits(1:0) = Lane0

bits(3:2) = Lane1

bits(5:4) = Lane2

bits(7:6) = Lane3

For stable link, the bits for each lane enabled can read as "01"

l. SPIReadCheck 00a6,0,7,FF

JESDB/C: ELASTIC_BUFFER_STATE value

bits(1:0) = Lane0

bits(3:2) = Lane1

bits(5:4) = Lane2

bits(7:6) = Lane3

For stable link, the bits for each lane enabled can read as "11".

Same registers are read check for second instance of JESD, that is JESD Lane4, 5, 6, 7

6.2 Failure and Fix for JESD Error

1. If there is error in 0x118 or 0x119, this is serdes related error reg so it is important to check signal integrity of serdes. We have CAPI to check for PRBS test pattern in SRX and read the error counter.

   CAPI for SRX PRBS checker:

   enableSerdesRxPrbsCheck

   clearSerdesRxPrbsErrorCounter

   getSerdesRxPrbsError

   Also, if you want to verify the signal integrity for AFE to FPGA connection for serdes PHY layer, AFE had CAPI to send PRBS pattern.

   CAPI for STX PRBS enable:

   sendserdesTxPrbs
2. Other reg, 0x11b, 0x11c, 0x11d, 0x11e and 0x11f are JESD Lane error indicator. The error bit and the description are self-explanatory to indicate the issue in JESD Link. The following is a list of some common error and design for errors.

JESD204B

1. Using Subclass 1, make sure Sysref is correctly acknowledge by both AFE and FPGA/ASIC in deterministic fashion. Must be source synchronous with Device Clock and FPGA Ref. Clock, rising edge transition determines LMFC alignment.

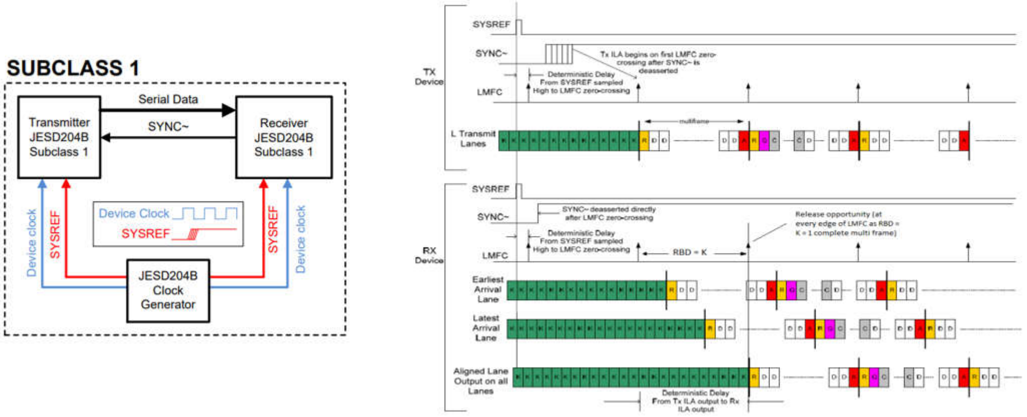

SUBCLASS 1
Transmitter JESD204B Subclass 1
Serial Data
SYNC~
Receiver JESD204B Subclass 1
Device clock
SYSREF
Device Clock
SYSREF
JESD204B Clock Generator
TX Device
SYSREF
SYNC~
Tx ILA begins on first LMFC zero-crossing after SYNC~ is deasserted
Deterministic Delay From SYSREF sampled High to LMFC zero-crossing
LMFC
multiframe
L Transmit Lanes
RX Device
SYNC~ deasserted directly after LMFC zero-crossing
Release opportunity (at every edge of LMFC as RBD = K = 1 complete multi frame)
RBD = K
Earliest Arrival Lane
Latest Arrival Lane
Aligned Lane Output on all Lanes
Deterministic Delay From Tx ILA output to Rx ILA output
K
R
D
A
Q
C

Figure 6-1. Deterministic Latency

> **Note**
>
> For more detail refer to *Understanding JESD204B Subclasses and Deterministic Latency*.

2. Lower serdes eye margin can cause issue, Try to adjust FFE Taps from FPGA/ASIC to improve swing of SRX on AFE.
3. Send K28.5 pattern from FPGA and Check if the AFE SYNC PIN is responding and check if CS state is as expected.
4. If we see any alignment related error there is a need to adjust RBD value. RBD is a release buffer space to buffer data for the time to adjust latency variation of lanes. See the *Determining Optimal Receive Buffer Delay in JESD204B and JESD204C Receivers*, application note.
5. After RBD is set correctly, FS state also becomes correct and link is stable at this point.
6. Check serdes polarity. If serdes polarity is reversed, CS state can come but FS and Buff state can not come.
7. Check if the 204B scrambler status matches for AFE and FPGA/ASIC. Either both can be enabled or both can be disabled.

JESD204C

1. As mentioned previously, sysref can be synchronized and applied in deterministic fashion for AFE and FPGA/ASIC.
2. Lower serdes eye margin can cause issue, try to adjust FFE Taps from FPGA/ASIC to improve swing of SRX on AFE.
3. Major source of alignment error in 204C is due to incorrect RBD size. So refer to how to set RBD application note for same.(*Determining Optimal Receive Buffer Delay in JESD204B and JESD204C Receivers*).
4. Depending on resolution of sample if 16 bits choose Extended Multiblock E = 1, if resolution is 12/24 bits choose E=3.
5. Choose CRC mode same for JESD receiver and transmitter.
6. Check serdes polarity. If serdes polarity is reversed, none of the CS, Buff and FS state comes.

7 Validating Serdes and JESD Link using CAPI

7.1 Useful Serdes Debug CAPIs

Along with PRBS CAPI, below CAPI can be used to debug SERDES Linkup.

getSerdesLinkStatus: This give dynamic information about Serdes Link Status of SRX lane (status of CDR Locked).

getSerdesRxLaneEyeMarginValue: This function gets the eye height of the receiving serder lane after processing.

reAdaptSerDesAllLanes: This can do Logic reset and readapt all lanes.

pollSerdesLinkStatusAllLanes: Bit wise Link Status for each lane. If Bit is 1: Lane Adapted success. If bit is 0, lane recovery does not happen. It returns 1 even for turned off lanes. This needs to be 0xff in good case.

SetSerdesTxCursor: This function can be used to set the FFE Taps for the AFE STX. The detail for Taps is given in configuration guide.

Solving the serdes PHY layer signal integrity problem can remove loss of signal and other serdes related error.

7.2 Useful JESD Debug CAPIs

The following CAPI can be used for JESD debug.

getJesdRxLaneErrors: This is used to check for lane error.

getJesdRxAlarms: This function can log error in complete JESD and Serdes Link.

clearJesdRxAlarms: To clear JESD alarm, as JESD Alarm reg are sticky. To read fresh error status, the recommendation is to clear and read.

getAllLaneReady: This function reads the all lane ready counter which is the offset between the internal LMFC boundary and the multiframe boundary (in JESD204B) or extended multi block boundary (in JESD204C) of the last lane of arrival. This value with some offset is set in RBD.

setManualRbd: To set RBD #detail on how to set RBS is explained in another application note, (*Determining Optimal Receive Buffer Delay in JESD204B and JESD204C Receivers*).

adcDacSync: This is an important function to resync AFE JESD block, there is a requirement here to leak sysref during this function. Which is part of user define function.

The previous information can help to solve for JESD related failure during bring-up.

In previous sections, we have discussed about error or failure possible during AFE bring-up and possible cause for the failure along with guide to resolve the bring-up error. There are few more debugging actions that come handy post bring-up of AFE, when capturing the data for first time or in case of any failure post bring-up. AFE79xx has many functionalities which are dynamically programmable post bring-up and to control or enable those function, functions are made part of CAPI.

8 TX Chain Validation

1. We have CAPI **dacJesdConstantTestPatternValue** which can send fix I and Q value in TX Chain at the input to NCO. A Single tone at NCO value and the tone amplitude can be controlled using the I and Q values.
2. This test is useful in case of checking if TX Chain is configured properly when random or improper data in TX spectrum, In case of improper data if we enable the **dacJesdConstantTestPatternValue** and see tone output is correct at NCO, we can narrow down the problem is not in TX chain but in JESD block of TX chain or the Serdes linkup or JESD of FPGA/ASIC.
3. We can use this test single tone for TX output power check or calibrating RX chain equivalent to a signal generator and also program the frequency by configuring the TX NCO dynamically with in some range, To configure NCO dynamically, we can use CAPI **updateTxNco**.
4. Also, in case of any spur debug. we can easily figure out the coupling point is inside AFE or external by changing the TX DSA using CAPI **setTxDsa**. If the spur level change with DSA in can be coupling somewhere internally in Chain if spur level does not change with TX DSA or scale non-linearly it can be external coupling or two sources with external coupling dominating.
5. AFE CAPI Libs also has option to internally read the dbfs power with in Tx Chain, using CAPI **readTxPower**. This can also be handy during debug test.

9 RX Chain Validation

In RX chain we can use a test pattern to send a ramp data from JESD to capture on FPGA, without need of external signal generator connected to RX using CAPI **adcRampTestPattern**. This test pattern is useful to check Jesd capture functionality .

We have **getRxRmsPower** to measure the rms power in rx chain. It can be used in debug when capture is not working and if need to check if ADC is working fine, this function will confirm the basic functionality.

10 Device Health

CAPI to check device health, **checkDeviceHealth** indicates status for PLL, DAC JESD, ADC JESD, SPI, MCU, and PAP.

11 Summary

Application note serves like a guidebook for resolving various issues seen during AFE7950 bring-up by describing the cause of the issue and step by step approach to resolve the failures. As described, most of the issues are dependent on external factors and can have board dependencies so it is good to inspect board as a starting point. Depending on the nature of issue, key factor to check can be operating voltage, SPI timing, reference clock level, sysref common mode and level, Serdes connection mapping and polarity.

Also, we have mentioned different CAPIs which can be used to analyze TX and RX related issue, post bring-up and can be done dynamically. Implementing CAPI in FPGA or ASIC need a processor onboard. We are working on an application note for simplifying the implementation of CAPI on FPGA platform. As CAPI has potential advantage of giving more accessibility to AFEs operation.

12 References

1. Texas Instruments, *JESD204B Overview* high speed data converter training.
2. JEDEC Standard: JESD204C.1.
3. Texas Instruments, *Determining Optimal Receive Buffer Delay in JESD204B and JESD204C Receivers*, application note.
4. Texas Instruments, *Understanding JESD204B Subclasses and Deterministic Latency*.

IMPORTANT NOTICE AND DISCLAIMER

TI PROVIDES TECHNICAL AND RELIABILITY DATA (INCLUDING DATA SHEETS), DESIGN RESOURCES (INCLUDING REFERENCE DESIGNS), APPLICATION OR OTHER DESIGN ADVICE, WEB TOOLS, SAFETY INFORMATION, AND OTHER RESOURCES “AS IS” AND WITH ALL FAULTS, AND DISCLAIMS ALL WARRANTIES, EXPRESS AND IMPLIED, INCLUDING WITHOUT LIMITATION ANY IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE OR NON-INFRINGEMENT OF THIRD PARTY INTELLECTUAL PROPERTY RIGHTS.

These resources are intended for skilled developers designing with TI products. You are solely responsible for (1) selecting the appropriate TI products for your application, (2) designing, validating and testing your application, and (3) ensuring your application meets applicable standards, and any other safety, security, regulatory or other requirements.

These resources are subject to change without notice. TI grants you permission to use these resources only for development of an application that uses the TI products described in the resource. Other reproduction and display of these resources is prohibited. No license is granted to any other TI intellectual property right or to any third party intellectual property right. TI disclaims responsibility for, and you will fully indemnify TI and its representatives against, any claims, damages, costs, losses, and liabilities arising out of your use of these resources.

TI’s products are provided subject to TI’s Terms of Sale or other applicable terms available either on ti.com or provided in conjunction with such TI products. TI’s provision of these resources does not expand or otherwise alter TI’s applicable warranties or warranty disclaimers for TI products.

TI objects to and rejects any additional or different terms you may have proposed.

Mailing Address: Texas Instruments, Post Office Box 655303, Dallas, Texas 75265