Application Note

AFE79xx Bringup Using HDL Module Without Microcontroller

Dineej, Srinivas Murthy

ABSTRACT

Device bringup of AFE79xx device is generally done using micro controller through SPI. In some of the applications, micro-controller can not be allowed in the system. For such applications, HDL based device bringup is needed to configure the AFE. Texas Instruments BRINGUP_AFE HDL module integrated into the FPGA along with a memory (to store the bringup instructions) can be used to configure AFE79xx devices.

Table of Contents

Trademarks

1 Introduction

RF Transceivers are used to communicate data, voice, or video information over the wireless communication medium as electromagnetic waves. The transceivers are widely deployed in systems such as telecommunication (base stations), RADAR, aerospace research, ammunition, and so on. Typically, these highly complex devices are configured using a microcontroller in the FPGA or ASIC through SPI. In some of the defense applications, use of microcontroller in the system is not acceptable. For such applications, HDL module implemented inside the FPGA with state machines can be used to configure the device without any microcontroller.

2 Hardware Setup for Using BRINGUP_AFE Module

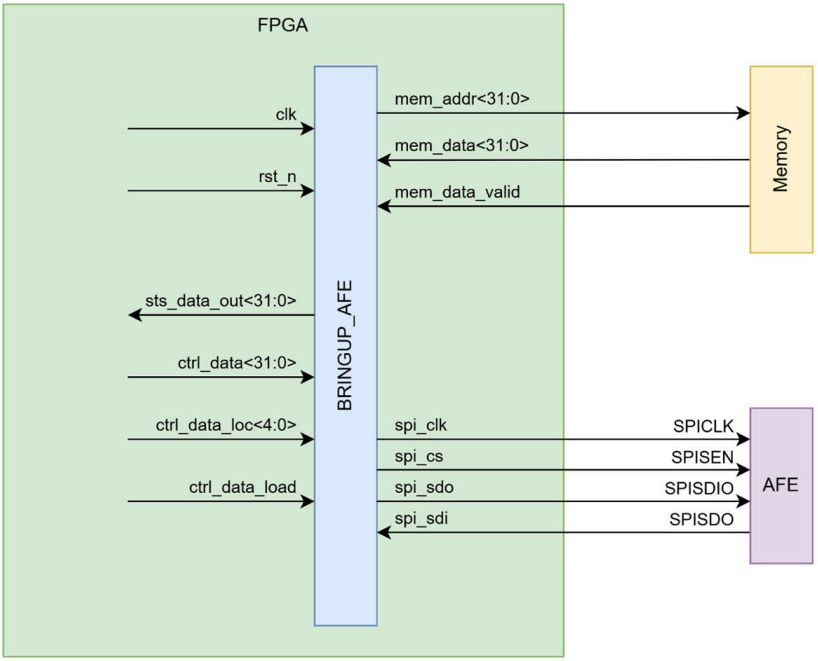

Figure 2-1. Hardware Connections of BRINGUP_AFE Module

Table 2-1. BRINGUP_AFE Module – Port Descriptions

Port	I/O	Bit Width	Description
clk	input	1	Module clock. Same clock is used for SPI clock. Max clock speed is 10MHz
rst_n	input	1	Active low reset
mem_addr	output	32	External memory address from which instructions need to be loaded during bringup. Memory address starts with 0 and increments by 1 after each instruction
mem_data	input	32	Bringup instruction stored at *mem_addr*
mem_data_valid	input	1	Instruction in *mem_data* port is only executed when *mem_data_valid* = 1 Clear *mem_data_valid* when *mem_addr* changes set *mem_data_valid* after *mem_data* reflects the instruction stored at *mem_addr* location
ctrl_data	input	32	*ctrl_data* is used to configure BRINGUP_AFE module registers
ctrl_data_loc	input	5	Address of the register in which *ctrl_data* need to be loaded. *sts_data_out* reflects the data in this register address
sts_data_out	output	32	Status information in register address *ctrl_data_loc*
ctrl_data_load	input	1	When *ctrl_data_load* = 1, *ctrl_data* is loaded to *ctrl_data_loc* register
spi_clk	output	1	SPI serial interface clock. Connect to SPICLK pin of AFE
spi_cs	output	1	SPI active low serial data enable. Connect to SPISEN pin of AFE
spi_sdo	output	1	MOSI. Connect to SPISDIO pin of AFE
spi_sdi	input	1	MISO. Connect to SPISDO pin of AFE

3 Interface Between Memory and BRINGUP_AFE Module

BRINGUP_AFE module sets *mem_addr* port to indicate the address from which memory can populate the data on *mem_data* port. Upon reset (*rst_n* = 0) or when *start_bringup* = 0, *mem_addr* is set to 0. When device is out of reset (*rst_n* = 1) and *start_bringup* = 1, bringup instructions are executed. After executing each instruction, *mem_addr* is incremented by 1. Whenever *mem_addr* value changes, memory can set *mem_data_valid* port to 0 within one BRINGUP_AFE clock cycle. Once memory completed populating the data at *mem_addr* on to *mem_data* port, *mem_data_valid* can be set to 1.

Note

If the data at *mem_addr* in memory is reflected on *mem_data* port within one clock cycle of BRINGUP_AFE module, then *mem_data_valid* port can always be set to 1.

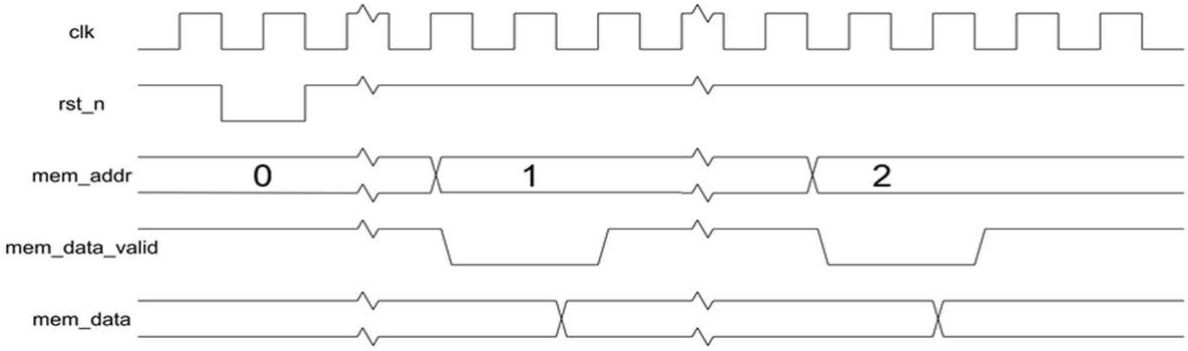

Figure 3-1. Timing of Data Transfer Between BRINGUP_AFE Module and Memory

4 Configuration of Control Registers and Status Flags in BRINGUP_AFE Module

BRINGUP_AFE module uses 32-bit registers to control the module and to store the status of the bringup. Each of these 32-bit registers are accessed using *ctrl_data_loc*, *ctrl_data*, *ctrl_data_load*, and *sts_data_out* ports. *Ctrl_data_loc* is used to select the 32-bit register to be modified or read. After setting *Ctrl_data* followed by a 0-1-0 toggle on *ctrl_data_load* modifies 32-bit data at *ctrl_data_loc* with *ctrl_data*. *Sts_data_out* reflects the 32-bit data at *ctrl_data_loc* in the BRINGUP_AFE module immediately.

Table 4-1. Status and Control Register Mapping

Address	Bits	Field Name	Access	Default	Description
0x0	0:0	bringup_finished	R	0x0	'1' indicates bringup is finished
0x0	1:1	Invalid	R	0x0	Invalid field. Data is irrelevant.
0x0	2:2	bringup_err_sticky	R	0x0	If set, indicates bringup error has occurred. Sticky flag. Toggle *clear_bringup_err* to clear this flag
0x0	3:3	bringup_halt	R	0x0	Indicates if the BRINGUP_AFE is in halt state. Either bringup is finished or bringup error occurred with *stop_at_bringup_fail* = '1'
0x0	4:4	unknown_instruction	R	0x0	Module can not interpret the current instruction. Not expected to happen
0x0	5:5	module_ready_flag	R	0x0	Status flag indicates if the module is ready for bringup
0x0	6:6	bu_cycles_ovr	R	0x0	If the *bu_cycles* are more than 2**32, over flag is set
0x0	7:7	give_sysref_now	R	0x0	when *give_sysref_now* flag is set, send the sysref in case of single shot sysref. Once sysref is given to the AFE, toggle *continue_bringup*. Ignore for continuous sysref
0x0	10:8	Invalid	R	0x0	Invalid field. Data is irrelevant.
0x0	11:11	data_valid	R	0x0	For debug. Value of *mem_data_valid* port
0x0	12:12	Invalid	R	0x0	Invalid field. Data is irrelevant.
0x0	15:13	Invalid	R	0x0	Invalid field. Data is irrelevant.
0x0	23:16	spi_data	R	0x0	For debug. Spi data readback from device for the current instruction
0x0	31:24	Invalid	R	0x0	Invalid field. Data is irrelevant.
0x1	31:0	curr_instr	R	0x0	For debug. Indicates the current instruction
0x2	31:0	Invalid	R	0x0	Invalid field. Data is irrelevant.
0x3	31:0	Invalid	R	0x0	Invalid field. Data is irrelevant.
0x4	31:0	bu_cycles	R	0x0	Number of clock cycles taken for the bringup. When *start_bringup* = 0, reset
0x5	31:0	mem_addr	R	0x0	For debug. Same value as *mem_addr* port
0x6	31:0	mem_data	R	0x0	For debug. Same value as *mem_data* port
0x7	31:0	BRINGUP_AFE_version	R	0x100	Indicates BRINGUP_AFE module version.
0x8	31:0	current_bu_err_code	R	0x0	Error code in case of any bringup error.
0x10	0:0	clear_bringup_err	R/W	0x0	Toggle to clear *bringup_err_sticky*
0x10	1:1	stop_at_bringup_fail	R/W	0x1	If set, module halts at bringup error. Otherwise ignores the bringup error and continues
0x10	2:2	continue_bringup	R/W	0x0	Toggle this to continue the bringup once module is halted on bringup error

Table 4-1. Status and Control Register Mapping (continued)

Address	Bits	Field Name	Access	Default	Description
0x10	4:3	RSVD	R/W	0x0	Reserved. Do not change the default value.
0x10	5:5	ovr_instr	R/W	0x0	Override instruction. Whether to override the current instruction or not
0x10	6:6	ovr_instr_trig	R/W	0x0	0-1-0 toggle starts executing the *ovr_val_instr*. Possible only when *bringup_halt* is 1.
0x10	7:7	RSVD	R/W	0x0	Reserved. Do not change the default value.
0x10	8:8	start_bringup	R/W	0x0	Keep this flag high always during the bringup. Give a 0 --> 1 pulse to start the bringup. 0 can be given for atleast 5 clock cycles before changing to 1
0x10	31:9	RSVD	R/W	0x0	Reserved. Do not change the default value.
0x11	31:0	RSVD	R/W	0x100000	Reserved. Do not change the default value.
0x12	31:1	RSVD	R/W	0x0	Reserved. Do not change the default value.
0x13	31:2	ovr_val_instr	R/W	0x0	Override value for current instruction. Valid only when *ovr_instr* = '1'

Steps to update the control registers

1. Set *ctrl_data_loc* to the address of the register in which *ctrl_data* need to be loaded.
2. Set *ctrl_data*.
3. Set *ctrl_data_load* to 1.
4. Wait for 2 BRINGUP_AFE clock cycles.
5. Set *Ctrl_data_load* to 0.

Steps to read the control or status registers

1. Set ctrl_data_loc to the desired address.
2. Read the data from sts_data_out.

4.1 Instruction Sets

BRINGUP_AFE module uses 32-bit instruction set to configure the device. Bits <31:29> is used as the opcode to identify the type of the instruction.

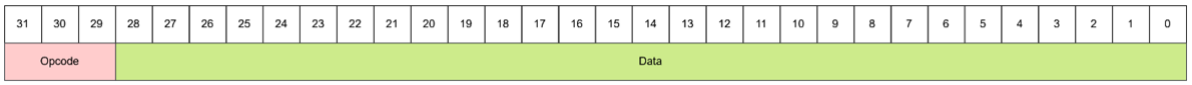

Figure 4-1. 32bit Instruction Field

Table 4-2. Opcode Interpretation in 32-bit Instructions

Opcode<31:29>	Interpretation
0	RSVD
1	AFE register read and register readcheck
2	AFE register write and AFE register read-modify-write
3	AFE register poll
4	Wait
5	Burst write
6	Relay – Used for sub opcode
7	RSVD

Opcode 6 is used as a relay opcode to support additional functionalities.

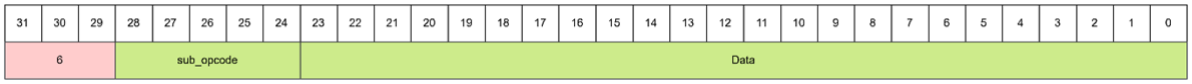

Figure 4-2. Sub Opcode Mapping

Table 4-3. Sub Opcode Interpretation in 32-Bit Instruction

Sub_opcode<28:24>	Interpretation
0	RSVD
1	BRINGUP_AFE module version check
2	Set bringup error code
3	Give sysref now – valid when continuous sysref is false
4	Bringup is finished
5	Set AFE reset pin
6-31	RSVD

Steps to generate the bringup instructions:

1. Set logDumpInst.logFormat=0x4000 in AFE79xx Latte bringup script.
2. Run the bringup file.
3. A text file with bringup instructions is generated in ≈\Texas Instruments\AFE79xxLatte\lib folder.
4. Use the 32-bit instructions in the text file to load in to the memory.

5 Bringup With BRINGUP_AFE Module

After memory is loaded with the device bringup instructions, below steps can be followed to do the bringup with continuous sysref. After the bringup is finished, *bringup_err_sticky* status can be checked for bringup errors.

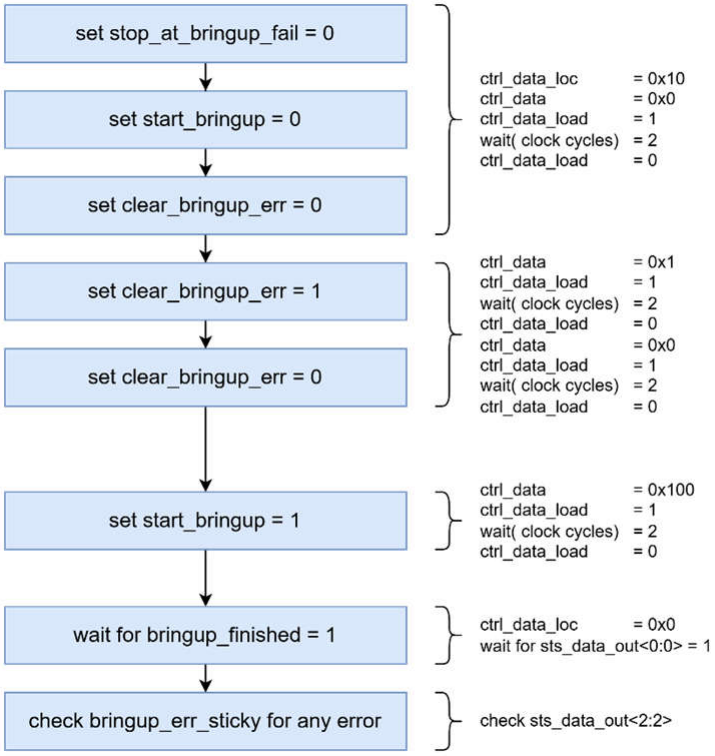

Figure 5-1. Device Bringup Using BRINGUP_AFE Module

6 Bringup With Error Decoding

The following steps can be followed to stop at bringup fail and the error register can be read back to figure out the possible cause of the error.

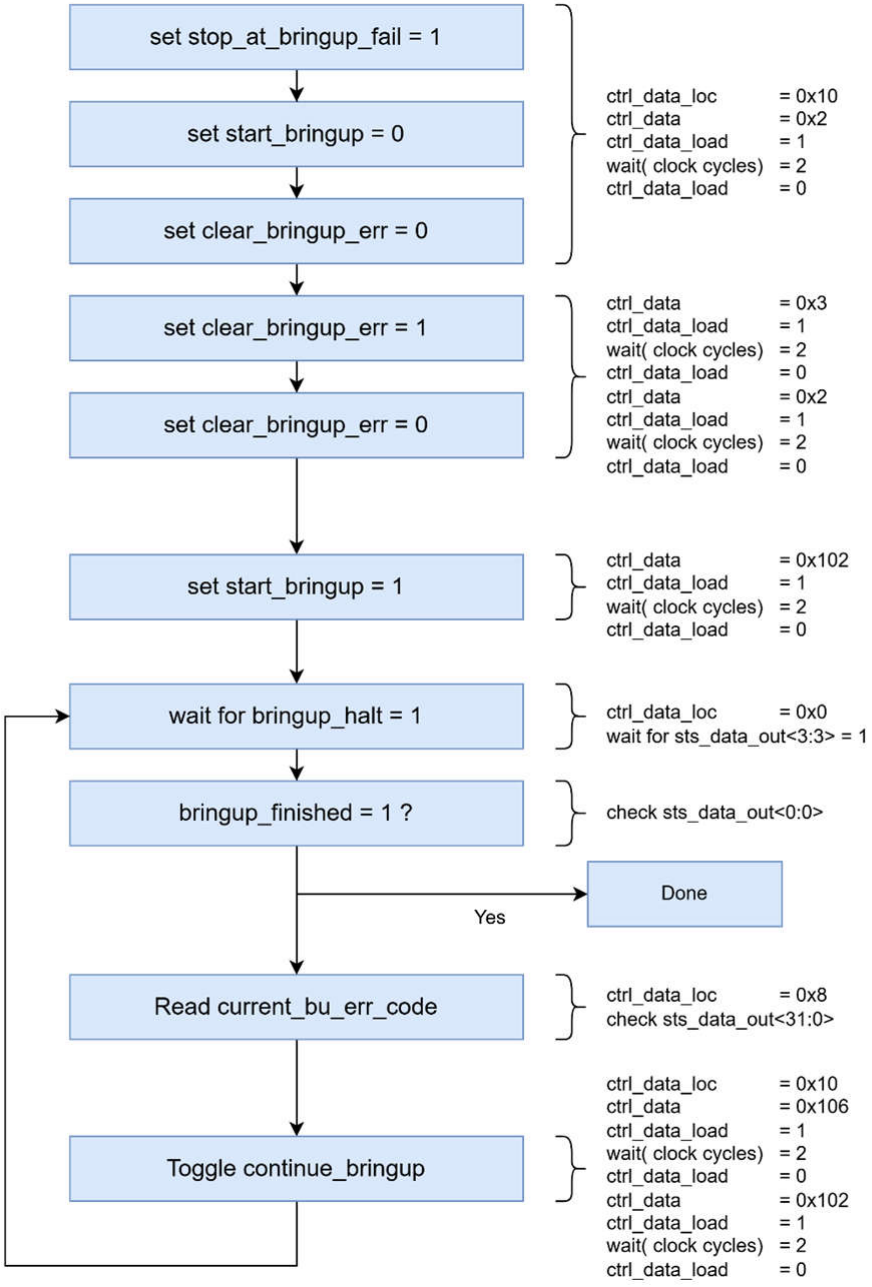

Figure 6-1. Device Bringup With Error Decoding

7 Summary

This application note describes the procedure to configure AFE79xx using HDL module integrated in FPGA without micro controller. BRINGUP_AFE module supports multiple features like bringup with error reporting, bringup with error decoding, debug options using override instructions, bringup with single shot sysref etc. Ultra-fast bringup time of <300ms is supported by running AFE79xx bringup at 10MHz clock.

8 References

- Texas Instruments, *AFE7900 4T6R RF Sampling AFE with 12 GSPS DACs and 3 GSPS ADCs*, data sheet.
- Texas Instruments, *AFE7950 4T6R RF Sampling AFE with 12GSPS DACs and 3GSPS ADCs*, data sheet.

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