Apollo CMv3 FPGA for Production Testing

Each FPGA needs to be programmed to support specific board testing functions. The specific needs are documented here. It may require multiple “bit” files to provide complete test coverage.

Associated Files

Many files are found on github at https://github.com/apollo-lhc/Cornell_CM_Rev3_HW . Documents related to testing are found in the “BoardTesting” directory.

Standalone Test Board vs. Service Module (SM)

The 6089-129_CM_TESTBOARD has been designed to exercise the CM on the benchtop without using an SM. The schematic is in the “BoardTesting” directory.

The test board can be used for low-power FPGA tests. There is insufficient cooling for tests that involve significant power consumption, such as the heater tests.

1 Overview of the FPGA specs

One or more FPGA programs will support testing the following board functions.

1.1 FPGA JTAG programming

The first firmware loaded into the FPGAs will also confirm the functionality of the FPGA JTAG chain. Both FPGAs will be programmed.

1.2 FPGA FLASH programming and FPGA booting from FLASH

The firmware can be programmed into the FLASH memory to confirm the programming ability. MCU signals can be manipulated to confirm that the FPGAs can boot from FLASH. The MCU can drive “FPGA_CFG_FROM_FLASH”, “F1_CFG_START, and “F2_CFG_START”. The MCU will read “/Fn_CFG_DONE” and “/Fn_INSTALLED”.

1.3 Testing of specific MCU signals

Several signals between the MCU and the FPGAs can be tested by having the FPGA loop specific input signals to output signals. The MCU can toggle the input signals and verify that it sees changes on the output signals. The FPGA inputs are “MCU_to_Fn” (n = 1 or 2). The FPGA outputs are “Fn_TO_MCU” and “Fn_C2C_OK”

1.4 Testing of front panel I/O connections

The signals between the FPGAs and the front panel I/O connector can be tested by using a cable from the front panel to the standalone test board. Specific jumper wires can be installed on the test board to support the testing scheme.

1.5 MCU-to-FPGA I2C Communication

Each FPGA has two I2C links with the MCU. The FPGAs can be configured as slaves. The MCU can confirm that it can communicate over all 4 I2C links.

1.6 Distribution of REFCLK and Logic Clock signals

Test configurations can be loaded into the clock synthesizers. Frequency counters in the FPGAs can report the clock frequency seen on each clock input.

1.7 Eye diagrams for GTY links

Operating configurations can be loaded into the clock synthesizers. Eye diagrams can be collected for each GTY link by using loopback connections.

1.8 High current power supply operation and thermal performance

Heater code can be loaded into the FPGAs and the operation of the power supplies at high current can be tested. The effectiveness of the cooling for the FPGAs, and possibly the FireFlys, can be verified.

2 Distribution of REFCLK and Logic Clock signals

Each FPGA has 28 REFCLK inputs, 6 logic clock inputs, one clock input on “spare_in[2]” from the other FPGA, and one input from a 200 MHz oscillator. Each FPGA will need 35 frequency counters to test the clock distribution. The 200 MHz oscillator will be used as a reference to gate the frequency counters. It is also sent to the other FPGA on “spare_out[2]”. The spare pair cross-connect is used to check the 200 MHz oscillators.

A method to read out the frequency counters needs to be developed (I2C?).

A set of clock synthesizer configuration files have been developed for this test. They generate unique frequencies on every output. A handful of clocks generated from the R0A and R0B synthesizers are fanned out to multiple destinations, so several frequency counters are expected to have the same value.

Tables of expected frequencies are provided below.

2.1 Load the “Step 1” configuration files into each synthesizer

The “Step 1” set of configuration files are: R0Av3X01, R0Bv3X01, R1Av3X01, R1Bv3X01, and R1Cv3X01. All of these are free running, using the attached 48 MHz crystal.

2.2 Load the test code into the FPGA

2.3 Verify the frequency of each clock

Read the frequency counters from each FPGA. Confirm that the values for each FPGA match the expected frequencies for “Step 1” from Table 1 (GTY REFCLKs) and Table 2 (Logic Clocks).

Table 1 Step #1 Frequencies for REFCLK inputs

Quad #	Quad Letter	Ref #	Vivado Constraints Name	FPGA #1 Step #1 Source	FPGA #1 Step #1 Freq	FPGA #2 Step #1 Source	FPGA #2 Step #1 Freq
120	AB	R0	lf_r0_ab	R1A-0A	60	R1A-0	40
120	AB	R1	lf_r1_ab	R1B-0	220	R1B-1	110
122	AD	R0	lf_r0_ad	R0A-2	300	R0A-6	260
122	AD	R1	lf_r1_ad	R1B-2	132	R1B-6	148
124	AF	R0	lf_r0_af	R0A-3	150	R0A-7	130
124	AF	R1	lf_r1_af	R1A-4	168	R1A-6	156
126	R	R0	lf_r0_r	R0A-2	300	R0A-6	260
126	R	R1	lf_r1_r	R1B-3	296	F1B-7	268
129	U	R0	lf_r0_u	R0A-2	300	R0A-6	260
129	U	R1	lf_r1_u	R1B-0A	176	R1B-5	326
131	W	R0	lf_r0_w	R0A-3	150	R0A-7	130
131	W	R1	lf_r1_w	R1A-3	312	R1A-5	336
133	Y	R0	lf_r0_y	R0A-2	300	R0A-6	260
133	Y	R1	lf_r1_y	R1B-4	163	R1B-8	134
220	L	R0	rt_r0_l	OSC	200	OSC	200
220	L	R1	rt_r1_l	R1A-7	272	R1A-8	136
222	N	R0	rt_r0_n	R0A-1	160	R0A-0	320
222	N	R1	rt_r1_n	R1C-4	170	R1C-5	340
224	P	R0	rt_r0_p	R0A-4	280	R0A-5	140
224	P	R1	rt_r1_p	R1C-0A	116	R1C-9	226
226	B	R0	rt_r0_b	R0A-1	160	R0A-0	320
226	B	R1	rt_r1_b	R1C-6	155	R1C-3	310
229	E	R0	rt_r0_e	R0A-1	160	R0A-0	320
229	E	R1	rt_r1_e	R1C-7	286	R1C-2	348
231	G	R0	rt_r0_g	R0A-4	280	R0A-5	140
231	G	R1	rt_r1_g	R1C-8	143	R1C-0	232
233	I	R0	rt_r0_i	R0A-1	160	R0A-0	320
233	I	R1	rt_r1_i	R1C-9A	113	R1C-1	174

Table 2 Step #1 Frequencies for Logic Clock inputs

Logic Clock Name (schematic)	Logic Clock Name (constraint file)	Logic Clock Pins		FPGA #1 Step #1 Source	FPGA #1 Step #1 Freq		FPGA #2 Step #1 Source	FPGA #2 Step #1 Freq
FnL_X12_R0_CLK	lf_x12_r0_clk	P33/P34		R0A-2	300		R0A-6	260
FnL_X4_R0_CLK	lf_x4_r0_clk	N32/M32		R0A-3	150		R0A-7	130
FnR_X12_R0_CLK	rt_x12_r0_clk	R18/R17		R0A-1	160		R0A-0	320
FnR_X4_R0_CLK	rt_x4_r0_clk	N19/N18		R0A-4	280		R0A-5	140
Fn_TCDS40_CLK	tcds40_clk	BF27/BF28		R1B-9	55		R1B-9	55
LHC_CLK	lhc_clk	BE26/BE27		SM	40		SM	40
FnFmSPARE2	in_spare[2]	C29/C30		FPGA#2	200		FPGA#1	200

Additional test steps will be added here. Many will require loading new configuration files in some synthesizers. Briefly:

1) Using control signals from the clock I2C register chips, switch sources on schematic 2.08 from synth R0A to synth R0B, check frequencies. There are 8 switch signals. Maybe do one at a time, or maybe do all 8 at once. If the switch and verify can be fully automated, switching one at a time would be preferable.
2) switch control signals from the clock I2C register chips back to synth R0A
3) load config to change R0A to use input #1 (from R0B) and feedback, check frequencies
4) load original config to change R0A back to free running, load config to change R0B to use input #0 (from R0A) and feedback, check frequencies
5) load config to change R0B to use input #2 (LHC CLOCK) and feedback, check frequencies
6) (optional) load config to change R0B to use input #1 (front panel clock) and feedback, check frequencies
7) Blah blah blah… test various inputs on R1A, R1B, and R1C. Consider using I2C controls to change inputs. It may save a few reconfigurations.