Apollo CMv3 Production Testing

The goal of production testing is to exercise as many components and connections on the board as possible. Many things cannot be tested with electrical tests. For instance, an open or missing bypass capacitor cannot be detected. However, a shorted capacitor can be flagged. A spreadsheet will specify which test step initially exercises each component. A google spreadsheet is probably needed for this so that multiple people can edit it.

With O(300) boards to be deployed, each hour per board requires 7.5 FTE person-weeks. If it takes a day per board then we would need more than an FTE year. The testing process must be as automated and thorough as possible. This will require hardware testing jigs and cables that standardize the testing process and minimize the amount of setup time. It will also require special software to run on the MCU, several varieties of firmware to run on the FPGAs, and many configuration files for the clock synthesizers. Multiple test setups may be required to allow more than one person at a time to perform tests.

Test programs will be run on Linux. Test results will be stored in a database or in files. Someone needs to set up these files. The default run mode for each program will be to create a Pass/Fail Status. It will be useful, but not necessary, if the programs can also be used to diagnose failures. The people performing the tests will probably not be the ones diagnosing failures. Control of the test programs from a GUI would be nice.

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

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

The test board mates with the three main connectors on the CM. Short cables mate with the front panel connectors of the CM. The test board provides connectors for 12 volt power, the Xilinx JTAG programmer, Linux UART cables, and Linux I2C and digital I/O cables. It also has jumper sites to allow one to connect most CM signal traces in a loop-back mode. Single-ended to differential input buffers allow external sources to drive many CM signals, while differential to single-ended output buffers support connections to scopes or other test equipment.

1 Test Sequence

Testing is done in alternating steps of manual activity followed by computer-driven tests. Details of this summary will be provided later in this document. With the proper software and equipment, most tests can be performed by hired undergrads. After the initial power test, it may be easier to perform further tests by moving the board to the "golden" SM. That moves the programming effort from the MCU to the Zynq on the SM or to the Linux computer. A dividing line between using the test board and using the SM needs to be defined.

1.1 Manual (Undergrad)

Unpack a board. Set FireFly transmit switches to 3.8v position. Install one 4-channel FireFly transceiver on each FPGA, and connect a fiber cable between them. Install copper FireFly loopback cables to other FireFly sites. Connect the board to the test system. Position fans for cooling. Connect a meter to measure 3.3v management power. Ramp the 12v power and note the voltage level when the 3.3v becomes good. Note the current as the voltage is ramped up to 12v. Stop the test if certain voltage or current criteria are not met.

Program the MCU with first-step code and run the program.

1.2 Automatic

1.2.1 Test I2C to each DC-DC converter (schematic 4.02)

Decide what to look for. We want to be able to verify that both reading and writing work as we uniquely access all 7 converters. Detect if I2C switching fails and we talk to the same converter multiple times. Verify that the I2C_RESET_PWR signal to the I2C mux works. Report errors as encountered. Report success after communicating with all seven DC-DC converters.

1.2.2 Configure, enable, and check each DC-DC converter.

Configure and enable converters one at a time, following the power-on sequence of schematic 1.04. Figure out how to uniquely test the paired 0.85 volt converters. They are used in pairs, but we don't know if one is good and one is bad. Check voltages and currents by reading internal DC-DC converter registers and the MCU ADC. Report errors as encountered, and success after enabling all seven DC-DC converters.

1.2.3 Test I2C to clock chips and I2C registers (schematic 4.03)

We want to be able to verify that both reading and writing work as we uniquely access all 5 clock chips. Detect if I2C switching fails and we talk to the same chip multiple times. Verify that we can uniquely access both register chips. Maybe toggle the "RESET" signals that go to the clock chips and verify that something with the clock chips changed? Verify that the I2C_RESET_CLOCKS signal to the I2C mux works. Report errors as encountered. Report success after communicating with all five clock chips and both register chips.

1.2.4 Test I2C to FPGAs (schematic 4.04)

(NOTE: This test requires a board that has FPGAs installed. Development of code may need to be deferred. Also, only the SYSMON ports can be tested before code is loaded into the FPGAs.)

We want to be able to verify that both reading and writing work as we uniquely access each FPGA. Detect if I2C switching fails and we talk to the same FPGA multiple times. Verify that the I2C_RESET_FPGAS signal to the I2C mux works. Report errors as encountered. Report success after communicating with both FPGAs.

1.2.5 Test I2C to each FireFly bank (schematic 4.05 and 4.06)

(NOTE: Complete coverage for this test requires that at least 4 FireFlys are installed in each bank, with two on each I2C MUX. We may choose to do what we can with a single FireFly on each bank.)

We want to be able to verify that both reading and writing work as we uniquely access different FireFly devices. Detect if I2C switching fails and we talk to the same FireFly multiple times. Verify that the I2C_RESET_F1_OPTICS and I2C_RESET_F2_OPTICS signals to the I2C muxes work. Verify that we can uniquely access both register chips. Maybe use the PRESENT signals and the RESET signal to the FireFlys? Report errors as encountered. Report success after communicating with both FPGAs.

1.2.6 Test I2C to EEPROM (schematic 4.01)

(NOTE: We may want to program the configuration block with the board serial number in this step, or that may happen elsewhere.)

We want to verify that we can program and read the EEPROM that is used to hold the board serial number and the synthesizer configuration files. Also, that the write-protect signal works.

1.2.7 Test optics I2C registers related to 3.3v/3.8v options

(haven't given this much thought yet)

1.3 Manual (Undergrad)

Set FireFly transmit switches to the 3.3v position.

1.4 Manual (Undergrad) or Automatic

Load second-step MCU code, which automatically turns on power and monitors conditions. Configure the clock chips for refclk testing. Load first-step FPGA code, which tests reflck inputs and I2C. Verify that oscillator clock is 200 MHz on each FPGA (sent out through front panel connector).

1.5 Automatic

Test optics I2C registers related to 3.3v/3.8v options. Test non-sysmon I2C to each FPGA. Verify that all refclks have the expected frequency (read over I2C). Switch clock chips between different inputs. Test FPGA flash memory.

1.6 Manual (Undergrad) or Automatic

Configure the clock chips for link integrity (IBERT) testing. Load second-step FPGA code, which tests links and some copper connections, such as low speed pairs between the FPGAs and FPGA-to-MCU connections.

1.7 Automatic

Verify that all links and connections are functional. This includes FireFly links, all FPGA-to-FPGA links (there are a few in the TCDS and C2C quads), C2C (loopback on test board), and TCDS (loopback on test board).

Tests on the test stand are finished.

1.8 Manual (Undergrad or technician, lots of board and fiber handling here)

Remove the board from the test stand. Remove the FireFly devices and loopback cables. Install the proper FireFly configuration for the end use. (NOTE: Do we want a step where we put FireFly devices in all sites to check all slow control connections?) Set the FireFly transmit voltage switches to 3.8v for 25Gx12 transmitters. Install the FireFly heatsink. Route FireFly cables to the front panel. Install loopback connectors.

Connect the CM to the golden SM. Install the SM front panel board. Attach a front panel, and connect the handle switch. Install covers. Install the board in an ATCA shelf and apply power.

1.9 Manual (Undergrad) or Automatic

Load operational MCU code. Configure the clock chips for normal operation. Load third-step FPGA code, which tests the operations links and contains heater code.

1.10 Automatic (figure out how to set tab stops so a 2-digit step does not jump so much)

Verify that all links are functional. Run heater tests and verify sufficient cooling and isolation between the FPGAs and the FireFlys.

1.11 Manual (undergrad or technician)

Remove the CM/SM from the ATCA shelf. Remove the FireFly loopback connectors. Separate the CM from the SM. Pack the CM for shipping.

2 Hardware

Two circuit boards will be provided for testing. One is a passive test board (TB) that provides connections to various CM and external signals. It does not contain any CPU. This will be used for

initial power tests and MCU programming. The other is a “golden” known-good service module (SM). This will be used for more sophisticated tests.

FireFly devices and test equipment with cables are also needed.

2.1 Test Board (TB)

The TB will be a passive PCB that connects to the three connectors on the CM -> the low speed connector and the two high speed connectors. It will provide the following:

2.1.1 Power

A high current header to connect a 12-volt power supply.

2.1.2 Serial UART

A header to connect a USB-to-Serial cable. The cable is driven from the linux system. It is used to communicate with the MCU.

2.1.3 I2C

A header to connect a USB-to-I2C cable. The cable is driven from the linux system. It is used to talk to the I2C port on the MCU.

2.1.4 JTAG

A 14-pin header to connect a Xilinx programmer. It is used for programming the FPGAs.

2.1.5 Clocking

The TB will provide both a 40 MHz clock and a 320 MHz clock. If they want to be the exact LHC frequencies, then a synthesizer will probably be needed. Otherwise, a pair of oscillators is sufficient.

2.1.6 High speed loopback

The TB will provide loopback connections for the C2C and TCDS links.

2.2 Service Module (SM)

This is an SM that is known to be fully functional and does not need to be inserted into an ATCA shelf. It will be used to run tests that need more capability than the TB can provide. This includes tests that use the C2C links into the FPGAs.

This SM will be mounted on a plate that allows the CM to be plugged in without the need for splice plates nor the front panel. It will utilize a standalone ATCA backplane from BU to provide power and ethernet.

2.3 Support Equipment

2.3.1 Power supply (12 volt)

This is a variable lab supply that provides zero to 12 volts with a settable current limit. It has meters that display the operating voltage and current. It is used for the standalone CM

2.3.2 Power supply (48 volt)

This is a fixed or variable lab supply that provides 48 volts and at least 6 amps. It has meters that display the operating voltage and current. It is used to power the SM.

2.3.3 Multimeter

This is needed to check voltages that are not connected to the MCU's ADC. Consider a multi-channel meter and a bed-of-nail test board to automate this?

2.3.4 MCU programmer

2.3.5 JTAG Programmer

2.3.6 Linux-to-UART cable

The UART cable is an FTDI "TTL-232R-3V3" (DigiKey 768-1015).

2.3.7 Linux-to-I2C cable

This is needed to check voltages that are not connected to the MCU's ADC. Consider a multi-channel meter and a bed-of-nail test board to automate this?

2.3.8 Frequency Counter (or Oscilloscope) and front-panel test cable

This is used to verify the 200 MHz fixed oscillator.

2.3.9 FireFly cables or devices

The links can be tested with 12-channel Samtec ECUE copper cables that connect transmit to receive, or with 8-channel ECUE cable that connects on 25Gx4 site to another. The advantage of using cables is that it shows the quality of the entire link from the FPGA transmitter to the FPGA receiver. It might not work at 25G.

The links can also be tested with actual 25G FireFly devices. One may want to perform two tests; one with CDR on at 25G and one with CDR off at a slower speed. The FireFly devices are needed to check the slow speed signals on the 10-pin connector. More than one set may be needed if the gold fingers deteriorate before 300 insertions.

2.4 Bed-of-Nails Board

For fuller automation, we could consider a bed-of-nails test board. The CM would sit on top of it and make connections to various test points.

https://fixturfab.com $12k

https://www.youtube.com/watch?v=jEBBgkq-HaY

3 MCU Programs

The first program that is loaded into the MCU will be fully interactive, responding to commands that arrive on the UART. This program will not do anything automatically. Even the basic power-up

sequence for the DC-DC converters will require a series of individual commands for each converter. This allows for close examination of the results after each command. For instance, after enabling a DC-DC converter, the program will allow the examination of the voltage and board current as seen by the ADC inside the MCU, as well as the converter's internal registers.

The granularity of the MCU command set is to be determined. For instance, there could be a sequence of commands for the ADC to set up a channel, do a conversion, and return the result. Or there could be a single command to return a snapshot of all ADC values. The linux program would pull out the pieces it needs at that particular step.

After initial testing is complete, the production MCU code is loaded

4 FPGA Firmware

The heat sinks will not be installed until many basic tests have passed. The FPGA firmware will need to minimize power in the initial tests.

The first firmware loaded into the FPGAs will also confirm the JTAG functionality. It can also be programmed into the FLASH memory to confirm programming and booting from FLASH.

4.1 REFCLK monitor

This firmware will contain frequency counters that can return the frequency of the clock seen on all REFCLK inputs to the GTY transceivers. Each FPGA has clocks connected to 16 REFCLK_0 and 16 REFCLK_1 inputs. One test will involve programming unique frequencies on all 32 clocks (as well as the LOGIC clocks) and verifying their presence.

This firmware will also route the 200 MHz C2C clock to the front panel test connector, where it can be connected to a frequency counter. The 200 MHz clock is the reference for the frequency counters, so it needs a way to independently verify its performance.

4.2 Link performance (eye diagram)

This firmware will generate eye diagrams for all connected links. Links can be connected by way of copper FireFly loopback cables, FireFly optical devices, loopbacks on the TB, connections to the SM, and connections between the FPGAs.

4.3 FPGA power

After the board is fully assembled and installed in an ATCA shelf, the thermal performance needs to be verified. Programmable heaters are used for this.

5 Synthesizer Configuration Files

One set of configuration files will provide unique frequencies to all REFCLK inputs using the on-board oscillators. Other sets will be used to verify the various input sources.

6 Test procedures

Here are the procedures for various subfunctions and the order in which they are run.

6.1 Check 3.3v management power

6.2 Program MCU

6.3 Check LGA80D DC-DC converters

6.4 Check other power supplies

6.5 Configure synths for REFCLK frequency test

6.6 Configure FPGAs for REFCLK frequency test

6.7 Run REFCLK frequency test

6.8 Install testing FireFlys and loopbacks

6.9 Check FireFly slow control

6.10 Configure synths for FireFly tests

6.11 Configure FPGAs and run subsets of FireFly tests (limit FPGA power)

6.12 Final configuration

6.12.1 Install FPGA heatsink

6.12.2 Install DC-DC heatsink

6.12.3 Install FireFlys and heatsink

6.12.4 Check FireFly slow control

6.13 Install front sub-panel

6.14 Mate with SM

6.15 Run final tests in ATCA shelf

6.15.1 Install operational MCU code

6.15.2 Configure FPGA for power test and run test

6.15.3 Configure synths for end use

6.15.4 Configure FPGA for end use

6.15.5