Open a terminal on the laptop and
/> cd mmfe8/

The GUI runs on Scientific Linux 6.6 running Python 2.6 or 2.7
It needs NumPy and gtk 2.x, which can be loaded with yum. (Python 3.x and gtk 3.x are drastically different.) The LAN that the mmfe8's are connected to is controlled by the same computer. Trying to run the GUI on Windows 7 has been problematic. It is best to ssh into the LAN host and use xming on Windows to display the GUI. These issues will be fixed as time permits.

Start the gui with
/> python mmfe8.py &

Start a python udp server in a different terminal on a different pane on the laptop with
/> python server.py

The server IP address is 127.0.0.1 (loopback) and it is listening on port 50001. This can be changed with vim, gedit or emacs.

If you wish to demonstrate this on a LAN, then the GUI and server have to have appropriate addresses and ports. The server has to have the LAN address of the machine that it is on and the udp port on that machine has to have the permissions set correctly. The server's only other requirement is that its host has python. The GUI has to point to that ip address and udp port from the machine on which it is running.

This version of the GUI is set up for the resolution of the screen of this laptop. It was impossible to reduce the size of the buttons any further. There is a vertical scroll bar on the far right end of the GUI. Use the horizontal scroll bar to get there.

The design on the GUI mimics the Brookhaven LabView GUI. This version was designed to work with Cactus/IPbus, the GLIB board, a Spartan 6 FMC, and the mini1. So, some of the command buttons on the left will be left behind as we move forward to a board with 8 VMMs. The “Quick Set” frame performs exactly as the one on the LabView GUI.

The GUI starts in the loopback mode (127.0.0.1). It sends to UDP port 50001. The IP address can be changed in a pull down box. The addresses and the port can be changed in the python code in the GLOBAL Variables section at the top.

Set the GUI to whatever parameters are necessary to configure the vmm. The “Print Config Regs to Terminal” button does exactly that. You see what will be sent to each of the 51 registers in hexadecimal format. The “Write to the Config Registers” displays the messages sent to the mmfe8. The size of the message is currently limited by the Artix 7 on the mmfe8. There is no room for ddr3 and bram resources are sparse. The commands for writing the configuration stream are limited to about 100 chars. The command starts with a ‘w’ for write and the address of the register in hexadecimal preceded by “0x”. The following space separated tokens are the data to be written to the incremented registers. No further address

are necessary. The last token is the newline char '\n'. This is necessary. Any char after this token causes the command to be seen as wrongly formatted.

The GUI sends messages by instantiating a python socket client for each message. A server is then started on the same socket and waits for a reply. Microblaze sends "OK" or "ERR!". The udp socket should no longer hang if it dfoesn't receive a response.