Charge injection with python using helper functions

- Using python
- The charge injection function algorithm
- Algorithm description

Using python

After executing the initial configuration sequence here, you can start the charge injection sequence.

The sequence is as follows

Charge injection sequence

```
# Columns are 384
# Pulser value 10 bits. Min 0, Max 1023
asicIndex = 3
chargeInjectionFirstCol = 50
chargeInjectionLastCol = 100
chargeInjectionPulserValue = 200
APP.prepareChargeInjection(asicIndex, chargeInjectionFirstCol, chargeInjectionLastCol,
chargeInjectionPulserValue)

# If you want a software trigger, uncomment the following command.
# root.Trigger()
```

If you need to run it for all 4 asics, pass the asicIndex as a -1, run it, then send a trigger to acquire images.

The charge injection function algorithm

The charge injection python function is as follows

Charge injection function

```
    def fnChargeInjection(self, dev):
        with self.root.updateGroup(.25):

            """
            One time set reg
            """
            self.AsicTop.RegisterControlDualClock.SyncDelay.set(0)
            if (self.TestChargeInjection.ASIC.get() == -1) :
                startAsic = 0
                endAsic = 4
            else :
                startAsic = self.TestChargeInjection.ASIC.get()
                endAsic = self.TestChargeInjection.ASIC.get() + 1

            for asicIndex in range(startAsic, endAsic, 1) :
                print("Enabling ASIC {}".format(asicIndex))
                self.Mv2Asic[asicIndex].enable.set(True)
                self.Mv2Asic[asicIndex].FE_ACQ2GR_en.set(True)
                self.Mv2Asic[asicIndex].FE_sync2GR_en.set(False)
                self.Mv2Asic[asicIndex].test.set(1) # connecting charge injection

            # Hard coding the first adc column group
            lane_selected = np.zeros(384)
            lane_selected[self.TestChargeInjection.FirstColumn.get() : self.TestChargeInjection.LastColumn.get()
+ 1] = 1
            # lane_selected[0:63] = 1

            for asicIndex in range(startAsic, endAsic, 1) :
                self.Mv2Asic[asicIndex].InjEn_ePixM.set(1)
                self.Mv2Asic[asicIndex].Pulser.set(int(0))

            self.root.runControl.runState.set(0x0)

            for pulse_value in range(1, 1023, 2):
                self.TestChargeInjection.Progress.set(pulse_value/1023)

                for asicIndex in range(startAsic, endAsic, 1) :
                    self.Mv2Asic[asicIndex].Pulser.set(int(pulse_value))

                self.TestChargeInjection.PulserValue.set(self.Mv2Asic[asicIndex].Pulser.get())

                for column in lane_selected:
                    if self.TestChargeInjection._runEn == False :
                        for asicIndex in range(startAsic, endAsic, 1) :
                            self.Mv2Asic[asicIndex].test.set(0)
                        return
                    else :
                        for asicIndex in range(startAsic, endAsic, 1) :
                            self.Mv2Asic[asicIndex].InjEn_ePixM.set(int(column))
                            self.Mv2Asic[asicIndex].ClkInj_ePixM.set(1)
                            # ff chain advances on falling edge of clock signal
                            self.Mv2Asic[asicIndex].ClkInj_ePixM.set(0)

                self.root.Trigger()

            #disabling charge INJECTION
            for asicIndex in range(startAsic, endAsic, 1) :
                self.Mv2Asic[asicIndex].test.set(0)

        return
```

Algorithm description

The sequence of events to enable and execute charge injection are as follows

- Setting charge injection necessary registers in the relevant ASIC
    - FE_ACQ2GR_en = True
    - FE_sync2GR_en = False
    - InjEn_ePixM = 0
- Enable charge injection
    - test = True
- Set the value of the Pulser (Intensity of chosen columns)
- Shift in one element at a time an array of 384 elements (1 element per column) where each value is either 0 or 1 by setting the InjEn_ePixM register to the value then toggling the ClkInj_ePixM register to 1, then back to 0
    - 0 being disable charge injection for the column
    - 1 being enable charge injection for the column
- Once done disable charge injection by setting test register to 0