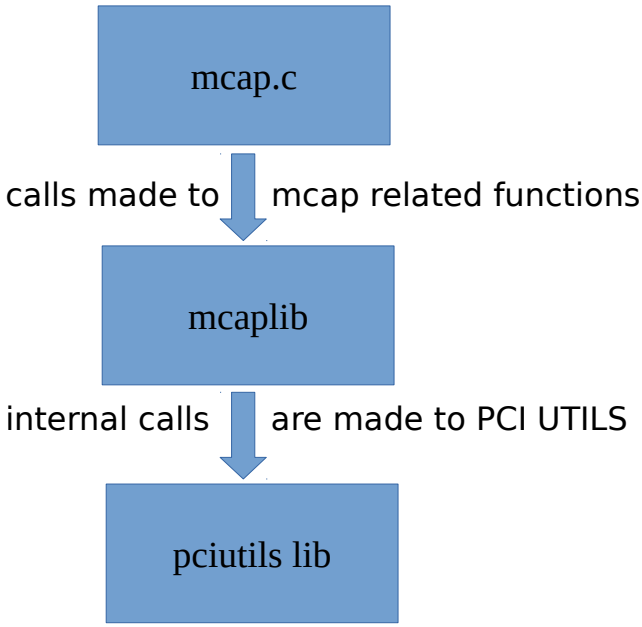
MCAP (Media Configuration Access Port) is an extended capability provided within Xilinx FPGAs to perform programmatical PR over PCIe or Tandem PCIe bitstream loading.

In the following documentation, we are going to provide a detailed explanation of the code flow of MCAP.

Overview of execution flow for mcap:



Explanation for relevant calls open from MCAP Lib:

MCapLibInit(int device_id):

This function instantiates a structure object consistent with a mcap device object with provided device id(Device id is a hexadecimal 16bit ID which can obtained from lspci for the given device).

This function expects a device id for generating an object of ***struct mcap_dev*** and returns a pointer to the object of ***struct mcap_dev.***

The function internally allocates the struct object for ***struct pci_dev.*** The allocation for pci_dev is done using pci_init(). pci_scan_bus() is another function called for getting a list of devices.
After getting the list of devices we run a pci_fill_info() over all the pci devices to find their device ids and vendor ids. This list of ids are looked up to find the right device with device_id provided from the program call.

McapDoBusWalk() is called to get the register base allocated for the device.

MCapLibFree(struct mcap_dev *mdev):

This function deallocates the mdev object created for a given mcap capable device.

MCapConfigureFPGA(struct mcap_dev *mdev, char *bitstream_file_name, EMCAP_CONFIG_FILE):

This function is essential if you intend to configure the fpga with the bitstream provided.

The function internally calls McapProcessBIT() to verify the bitstream.
Based on the last option provided to this function it would be either a programming of a Partial reconfiguration or a complete configuration.
The options are EMCAP_PARTIALCONFIG_FILE and EMCAP_CONFIG_FILE respectively.

(Note: Static configs are only possible for small bitstreams. Refer the following link page 13:
https://www.xilinx.com/support/documentation/user_guides/ug570-ultrascale-configuration.pdf)

Depending on the option provided, internal calls are made to functions McapWritePartialBitStream() and MCapWriteBitStream()

Note: This function internally makes a call to pci_write_long() function.

pdev pointer defined within mdev is send as the first parameter to pci_write_long(). The second and third parameters are regbase+offset and value. The values for offset and value is obtained from calls to MCAPWrite~() functions.

Explanation for relevant calls open from PCIUTILS Lib:

pci_write_long()

This function internally calls pci_write_data() where the call to memcpy is made with params as pci_device→cache + regbase+offset, buf, 4

pci_device is an object of struct pci_dev.

There are MCap functions such as MCapDumpReg, MCapModuleReset and other functions where you can play with register base of MCap. For more information kindly check the source page for MCAP.