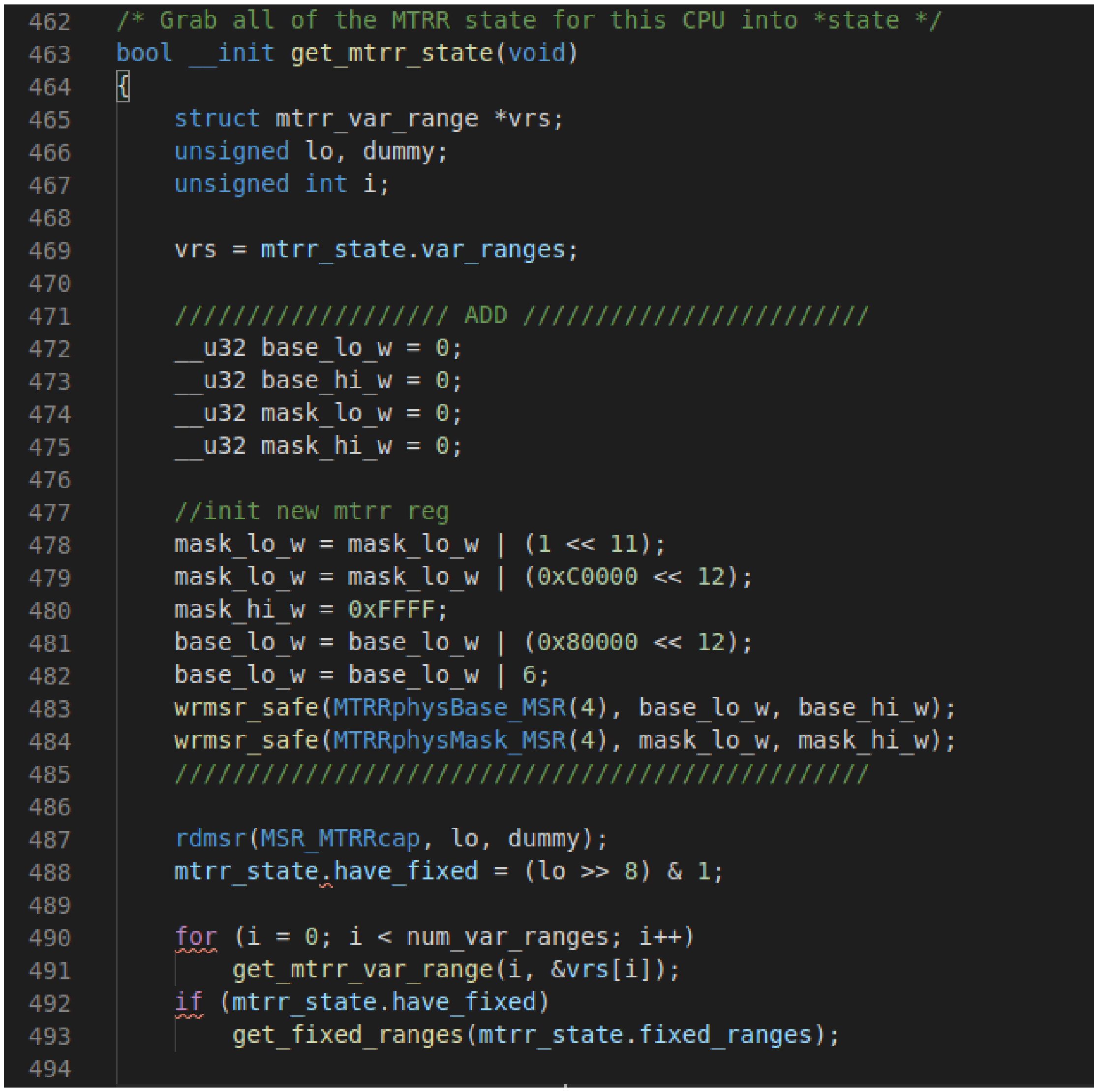
System setup:

CPU: AMD Ryzen 7 3700X

FPGA: Virtex UltraScale+ VCU118 Evaluation Platform

FPGA memory caching for persistent memory emulation

1. Use the following link to download the Ubuntu kernel source cord you prefer (The apt get method was used in our implementation to build the exact kernel that was already running)
   https://wiki.ubuntu.com/Kernel/BuildYourOwnKernel
2. Within the source code find the function get_mtrr_state(void). in linux-5.4.0 kernel this is in linux-5.4.0/arch/x86/kernel/cpu/mtrr/generic.c
3. Find the memory range the FPGAs PCIe bar is mapped to
4. Add the lines highlighted in the following code segment to the function. You will have to replace mask_hi_w and mask_lo_w with the start address of the FPGA address range. What this basically does is that it adds the PCie memory range to the mtrr register and make it cachable.



```
/* Grab all of the MTRR state for this CPU into *state */
bool __init get_mtrr_state(void)
{
    struct mtrr_var_range *vrs;
    unsigned lo, dummy;
    unsigned int i;

    vrs = mtrr_state.var_ranges;

    /////////////////// ADD ///////////////////////
    __u32 base_lo_w = 0;
    __u32 base_hi_w = 0;
    __u32 mask_lo_w = 0;
    __u32 mask_hi_w = 0;

    //init new mtrr reg
    mask_lo_w = mask_lo_w | (1 << 11);
    mask_lo_w = mask_lo_w | (0xC0000 << 12);
    mask_hi_w = 0xFFFF;
    base_lo_w = base_lo_w | (0x80000 << 12);
    base_lo_w = base_lo_w | 6;
    wrmsr_safe(MTRRphysBase_MSR(4), base_lo_w, base_hi_w);
    wrmsr_safe(MTRRphysMask_MSR(4), mask_lo_w, mask_hi_w);
    ////////////////////////////////////////////////

    rdmsr(MSR_MTRRcap, lo, dummy);
    mtrr_state.have_fixed = (lo >> 8) & 1;

    for (i = 0; i < num_var_ranges; i++)
        get_mtrr_var_range(i, &vrs[i]);
    if (mtrr_state.have_fixed)
        get_fixed_ranges(mtrr_state.fixed_ranges);

```

For example:
In line 481 add the most significant 18 bits of your 32-bit physical address replacing 0x80000. If the system uses 64-bit addressing replace 0x80000 with bits [31:12] and use the variable base_hi_w to store the most significant 32 bits of the address.

In line 479 add the most significant 18 bits of the upper-bound of the memory range and replace 0xC0000. If the system uses 64-bit addressing replace 0xC0000 with bits [31:12] and use the variable mask_hi_w to store the most significant bits of the upper-bound. If there are bits that remain after the most significant bit in the upper-bound change them to 1s.

```
Ex:     base:           0x8000_0000_0000
        Upper_bound:    0xC000_0000_0000
Then the variables should be
479     mask_lo_w = mask_lo_w | (0x00000 <<12)
481     base_lo_w = base_lo_w | (0x00000 <<12)
        mask_hi_w = (0xFFF << 18) | 0xC0000
        base_hi_w = 0x80000
```

building and installing the changed kernel

5. Next, change and update the grub file to make the PCIe memory range emulate persistent memory. First open the grub file

```
sudo vi /etc/default/grub
```

Then add the memmap command to the following line

```
GRUB_CMDLINE_LINUX="memmap=nn[KMG]!ss[KMG]"
```

In place of “nn[KMG]!ss[KMG]” add the memory range of the PCIe device. To understand how to use the correct notation take a look at the following link

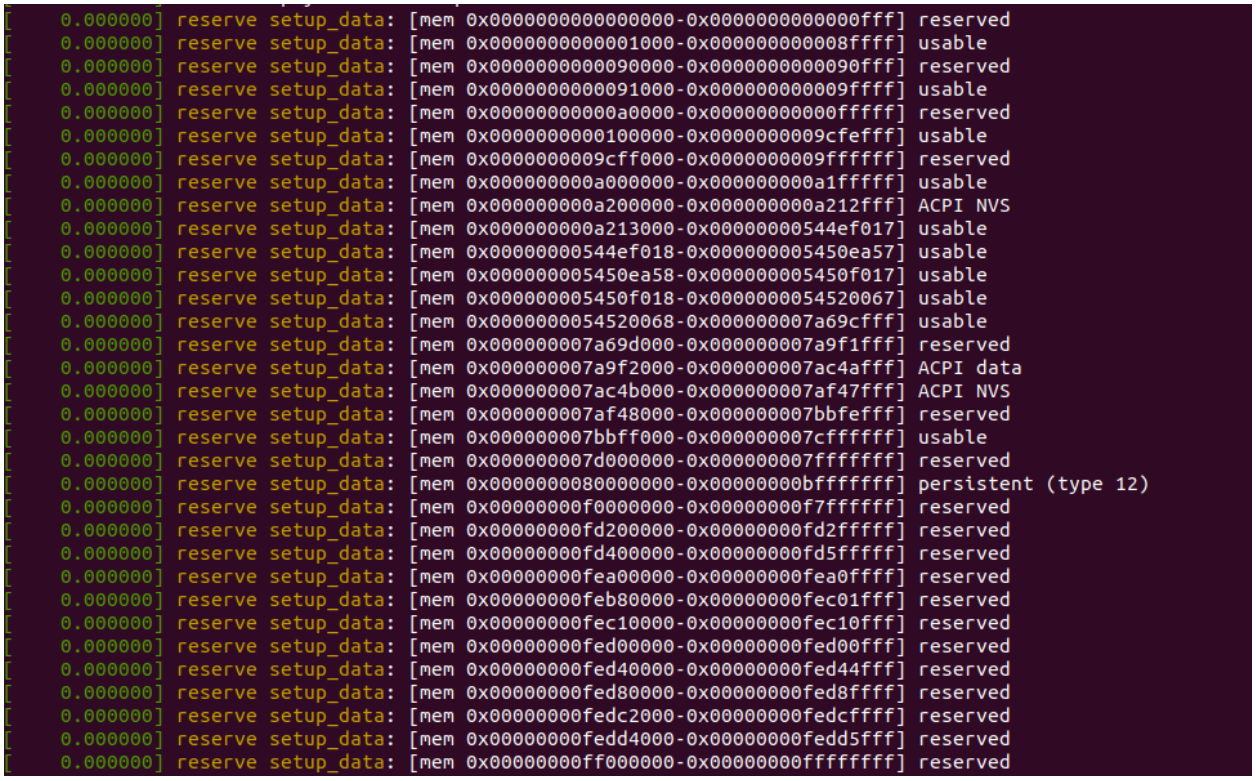
https://nvdimm.wiki.kernel.org/how_to_choose_the_correct_memmap_kernel_parameter_for_pmem_on_your_system

Finally, save changes and update grub

```
sudo update-grub2
```

Warning: Make sure that you correctly add the memory range. If there is a mismatch the system will not bootup

6. After rebooting you should be able to see the emulated persistent device as /dev/pmem0. Also double check the memory type in kernel debug messages



```
[    0.000000] reserve setup_data: [mem 0x0000000000000000-0x0000000000000fff] reserved
[    0.000000] reserve setup_data: [mem 0x0000000000001000-0x000000000008ffff] usable
[    0.000000] reserve setup_data: [mem 0x0000000000090000-0x0000000000090fff] reserved
[    0.000000] reserve setup_data: [mem 0x0000000000091000-0x000000000009ffff] usable
[    0.000000] reserve setup_data: [mem 0x00000000000a0000-0x00000000000fffff] reserved
[    0.000000] reserve setup_data: [mem 0x0000000000100000-0x0000000009cfefff] usable
[    0.000000] reserve setup_data: [mem 0x0000000009cff000-0x0000000009ffffff] reserved
[    0.000000] reserve setup_data: [mem 0x000000000a000000-0x000000000a1fffff] usable
[    0.000000] reserve setup_data: [mem 0x000000000a200000-0x000000000a212fff] ACPI NVS
[    0.000000] reserve setup_data: [mem 0x000000000a213000-0x00000000544ef017] usable
[    0.000000] reserve setup_data: [mem 0x00000000544ef018-0x000000005450ea57] usable
[    0.000000] reserve setup_data: [mem 0x000000005450ea58-0x000000005450f017] usable
[    0.000000] reserve setup_data: [mem 0x000000005450f018-0x0000000054520067] usable
[    0.000000] reserve setup_data: [mem 0x0000000054520068-0x000000007a69cfff] usable
[    0.000000] reserve setup_data: [mem 0x000000007a69d000-0x000000007a9f1fff] reserved
[    0.000000] reserve setup_data: [mem 0x000000007a9f2000-0x000000007ac4afff] ACPI data
[    0.000000] reserve setup_data: [mem 0x000000007ac4b000-0x000000007af47fff] ACPI NVS
[    0.000000] reserve setup_data: [mem 0x000000007af48000-0x000000007bbfefff] reserved
[    0.000000] reserve setup_data: [mem 0x000000007bbff000-0x000000007cffffff] usable
[    0.000000] reserve setup_data: [mem 0x000000007d000000-0x000000007fffffff] reserved
[    0.000000] reserve setup_data: [mem 0x0000000080000000-0x00000000bfffffff] persistent (type 12)
[    0.000000] reserve setup_data: [mem 0x00000000f0000000-0x00000000f7ffffff] reserved
[    0.000000] reserve setup_data: [mem 0x00000000fd200000-0x00000000fd2fffff] reserved
[    0.000000] reserve setup_data: [mem 0x00000000fd400000-0x00000000fd5fffff] reserved
[    0.000000] reserve setup_data: [mem 0x00000000fea00000-0x00000000fea0ffff] reserved
[    0.000000] reserve setup_data: [mem 0x00000000feb80000-0x00000000fec01fff] reserved
[    0.000000] reserve setup_data: [mem 0x00000000fec10000-0x00000000fec10fff] reserved
[    0.000000] reserve setup_data: [mem 0x00000000fed00000-0x00000000fed00fff] reserved
[    0.000000] reserve setup_data: [mem 0x00000000fed40000-0x00000000fed44fff] reserved
[    0.000000] reserve setup_data: [mem 0x00000000fed80000-0x00000000fed8ffff] reserved
[    0.000000] reserve setup_data: [mem 0x00000000fedc2000-0x00000000fedcffff] reserved
[    0.000000] reserve setup_data: [mem 0x00000000fedd4000-0x00000000fedd5fff] reserved
[    0.000000] reserve setup_data: [mem 0x00000000ff000000-0x00000000ffffffff] reserved
```

7. Finally follow "DAX - Direct Access" step in https://pmem.io/2016/02/22/pm-emulation.html to make the memory range directly accessible.
8. Now you can use this file to run workloads on emulated persistent memory on FPGA with caching