

RAID With SD Cards

The “block” in block device means you’ll be blocked from doing it...

Aeybel Varghese, Jason Kacines, Brandon Zupan

Why would you make RAID with SD Cards?

We wanted a project that focuses more on the software side than the hardware and verilog side, so our goal was to implement RAID through a Linux Kernel Module.

RAID through SD cards is useful in embedded applications where large amounts of data is collected and redundancy is necessary.

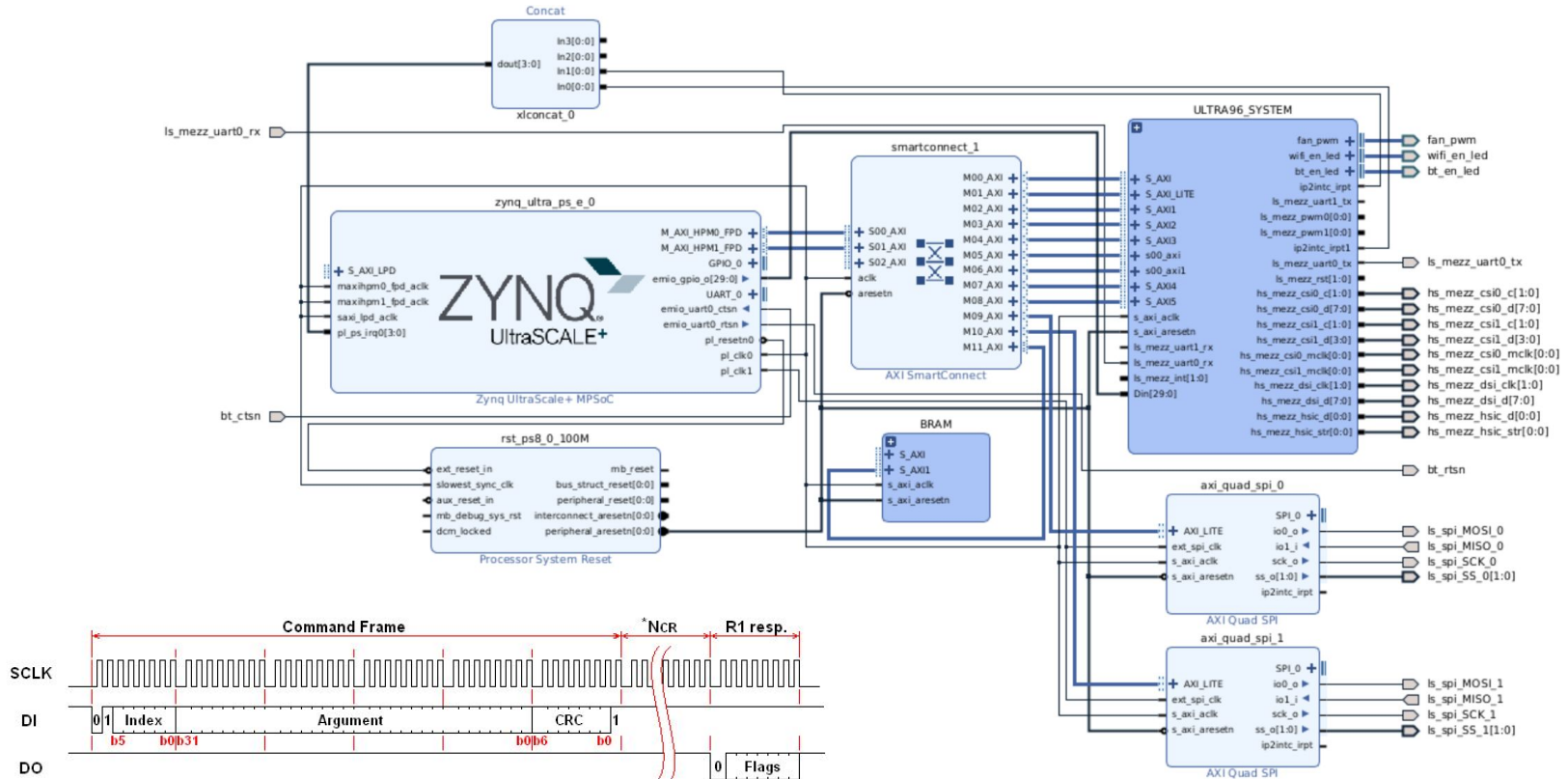
Linux storage devices require block device drivers

Similar to char devices, but has a request function instead of read and write.

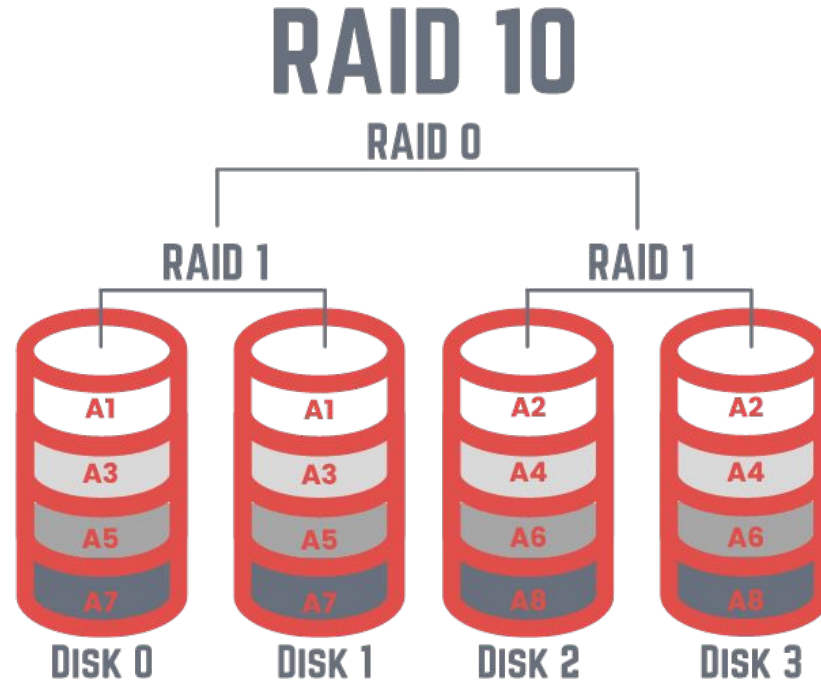
Driver iterates over a linked list of blocks in memory and submits them to the device.

More advanced drivers can re-order requests in the queue to optimize operations on storage devices with moving parts.

SD Card Interface and Drivers



Using RAID 10 for striping and mirroring



Block device drivers don't work on the Ultra96v2

We could not get a block device driver to work on the Xilinx Linux Kernel.

We were able to run sbull, an example block device driver, on a 5.10 linux server.

We think there's a bug causing some kind of fault when iterating over segments of a request on Xilinx versions of the kernel. We tested images with Petalinux 2020.2, 2021.2, and 2022.2, each having similar crashes.

```
/* Do each segment independently */
bio_for_each_segment(bvec, bio, iter) {
    char *buffer = __bio_kmap_atomic(bio, i, KM_USER0);
    char *buf; buf = kmap_atomic(bvec.bv_page) + bvec.bv_offset;
    //sbull_transfer(dev, sector, bio_cur_bytes(bio) >> 9,
    sbull_transfer(dev, sector, (bio_cur_bytes(bio) / KERNEL_SECTOR_SIZE),
        buffer, bio_data_dir(bio) == WRITE);
    //sector += bio_cur_bytes(bio) >> 9;
    sector += (bio_cur_bytes(bio) / KERNEL_SECTOR_SIZE);
    __bio_kunmap_atomic(buffer, KM_USER0);
    kunmap_atomic(buffer);
}
return 0; /* Always "succeed" */
```



We pivoted towards making a program that journals data

```
00000000 48 65 6c 6c 6f 20 74 68 69 73 20 69 73 20 74 68 Hello th is is th
00000010 65 20 6c 69 62 72 61 72 79 00 00 00 00 00 00 e librar y000000
00000020 ff ff ff ff 00 00 00 00 c0 54 02 dc ff ff 00 00 +++++000:~T+xxx00
00000030 66 30 0c a1 ff ff 00 00 00 b0 25 a1 ff ff 00 00 h0 _xxx00 0%+xxx00
00000040 00 00 00 00 00 00 00 00 20 54 02 dc ff ff 00 00 00000000:~T+xxx00
00000050 60 e2 23 a1 ff ff 00 00 60 1a 26 a1 ff ff 00 00 ~#+xxx00:~+B+xxx00
00000060 80 56 02 dc ff ff 00 00 40 55 02 dc ff ff 00 00 ~V+xxx00 @U+xxx00
00000070 40 6b 23 a1 ff ff 00 00 b0 21 26 a1 ff ff 00 00 @k#+xxx00:~lB+xxx00
00000080 01 00 00 00 00 00 00 00 00 00 26 a1 ff ff 00 00 +0000000:00B+xxx00
00000090 00 10 26 a1 ff ff 00 00 01 00 00 00 00 00 00 00 0 ~G+xxx00:~+000000
000000a0 a0 c2 25 a1 ff ff 00 00 b0 21 26 a1 ff ff 00 00 +%+xxx00:~lB+xxx00
000000b0 60 1a 26 a1 ff ff 00 00 01 00 00 00 00 00 00 00 ~+G+xxx00:~+000000
000000c0 00 20 26 a1 ff ff 00 00 00 00 00 4b 49 4e 47 0 G+xxx00:0000KING
000000d0 7c 78 24 a1 ff ff 00 00 c0 54 02 dc 02 04 00 00 |x$+xxx00:~T+xxx00
000000e0 64 78 24 a1 ff ff 00 00 00 00 00 00 00 00 00 00 dx$+xxx00:00000000
000000f0 f8 54 02 dc ff ff 00 00 00 00 00 00 00 00 00 00 ~T+xxx00:00000000
00000100 40 00 00 00 00 00 00 00 28 37 23 a1 ff ff 00 00 @0000000:(?#+xxx00
00000110 02 04 00 00 ff ff 00 00 40 10 26 a1 ff ff 00 00 +~00+xxx00 @~G+xxx00
00000120 08 25 26 a1 ff ff 00 00 b0 21 26 a1 ff ff 00 00 ~%G+xxx00:~lB+xxx00
00000130 00 00 00 00 00 00 00 00 00 00 26 a1 ff ff 00 00 00000000:00B+xxx00
00000140 68 30 0c a1 ff ff 00 00 00 3a 23 a1 ff ff 00 00 h0 _xxx00:0#+xxx00
00000150 70 05 00 00 00 00 00 00 2e 00 00 00 00 00 00 00 p-000000:~.000000
00000160 00 00 00 00 ff ff 00 00 40 55 02 dc ff ff 00 00 0000+xxx00 @U+xxx00
00000170 70 6b 23 a1 ff ff 00 00 78 21 26 a1 ff ff 00 00 pk#+xxx00:~lB+xxx00
00000180 00 00 00 00 ff ff 00 00 90 57 02 dc ff ff 00 00 0000+xxx00 ~W+xxx00
00000190 70 7d 24 a1 ff ff 00 00 b0 07 26 a1 ff ff 00 00 p} $+xxx00:~+G+xxx00
000001a0 b0 07 26 a1 ff ff 00 00 a0 50 23 a1 ff ff 00 00 +~G+xxx00:~#+xxx00
000001b0 20 0e 26 a1 ff ff 00 00 08 00 00 00 00 00 00 00 ~G+xxx00:~+000000
000001c0 40 00 8e c4 aa aa 00 00 00 00 00 00 00 00 00 00 @0+xxx00:00000000
000001d0 00 10 00 00 00 00 00 00 01 00 00 00 00 00 00 00 0 ~0-000000:~+000000
000001e0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00000000:00000000
y
00000400 48 65 6c 6c 6f 20 74 68 69 73 20 69 73 20 74 68 Hello th is is th
00000410 65 20 6c 69 62 72 61 72 79 00 00 00 00 00 00 e librar y000000
00000420 ff ff ff ff 00 00 00 00 c0 54 02 dc ff ff 00 00 +++++000:~T+xxx00
00000430 66 30 0c a1 ff ff 00 00 00 b0 25 a1 ff ff 00 00 h0 _xxx00 0%+xxx00
00000440 00 00 00 00 00 00 00 00 20 54 02 dc ff ff 00 00 00000000:~T+xxx00
00000450 60 e2 23 a1 ff ff 00 00 60 1a 26 a1 ff ff 00 00 ~#+xxx00:~+B+xxx00
00000460 80 56 02 dc ff ff 00 00 40 55 02 dc ff ff 00 00 ~V+xxx00 @U+xxx00
00000470 40 6b 23 a1 ff ff 00 00 b0 21 26 a1 ff ff 00 00 @k#+xxx00:~lB+xxx00
00000480 01 00 00 00 00 00 00 00 00 00 26 a1 ff ff 00 00 +0000000:00B+xxx00
00000490 00 10 26 a1 ff ff 00 00 01 00 00 00 00 00 00 00 0 ~G+xxx00:~+000000
000004a0 a0 c2 25 a1 ff ff 00 00 b0 21 26 a1 ff ff 00 00 +%+xxx00:~lB+xxx00
000004b0 60 1a 26 a1 ff ff 00 00 01 00 00 00 00 00 00 00 ~+G+xxx00:~+000000
000004c0 00 20 26 a1 ff ff 00 00 c0 54 02 dc 02 04 4e 47 0 G+xxx00:0000KING
```

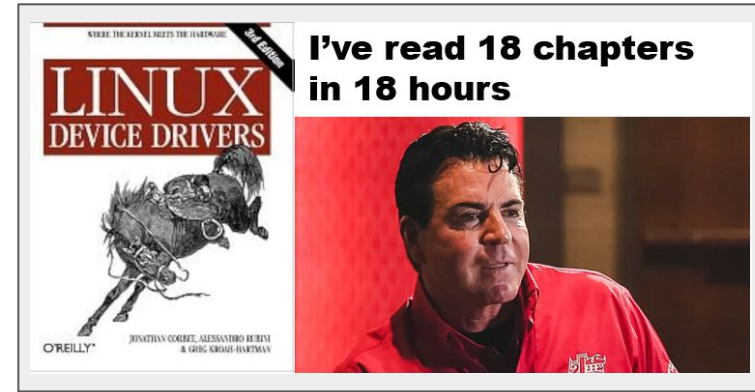
What did we learn?

Deep dive in building Linux Kernels, Petalinux, and cross compilation.

How Linux handles storage devices and uses them to make filesystems.

Driver development for block devices, and how to use IOCTL.

We hope our struggles can be used as a base for future groups that want to make a block device driver.



What would we have done differently?

Start with Linux 5.10 from the beginning, it has more examples and an API that is easier to understand.

Test with more kernels from different distros. We could have found out that the Xilinx one doesn't work and planned around that better.

Implement more in PL to improve performance.

