The ZIO Framework

A modular environment for laboratory I/O

http://www.ohwr.org/projects/zio

Alessandro Rubini, Federico Vaga

Independent consultants in Pavia, Italy.

Working for CERN "hardware and timing" group

The Requirements (hard)

Hardware timestamps (better than 1ns precision)

Big data blocks (stripes of many samples)

Off-line management of data blocks

High data rate

Easy monitoring of a diverse I/O environment

The Requirements (soft)

Sysfs-based configuration

No ioctl(2) thank you

Centralized locks (drivers must ignore the issue)

Modular design (each object should be replaceable)

A documented and stable framework

Device types

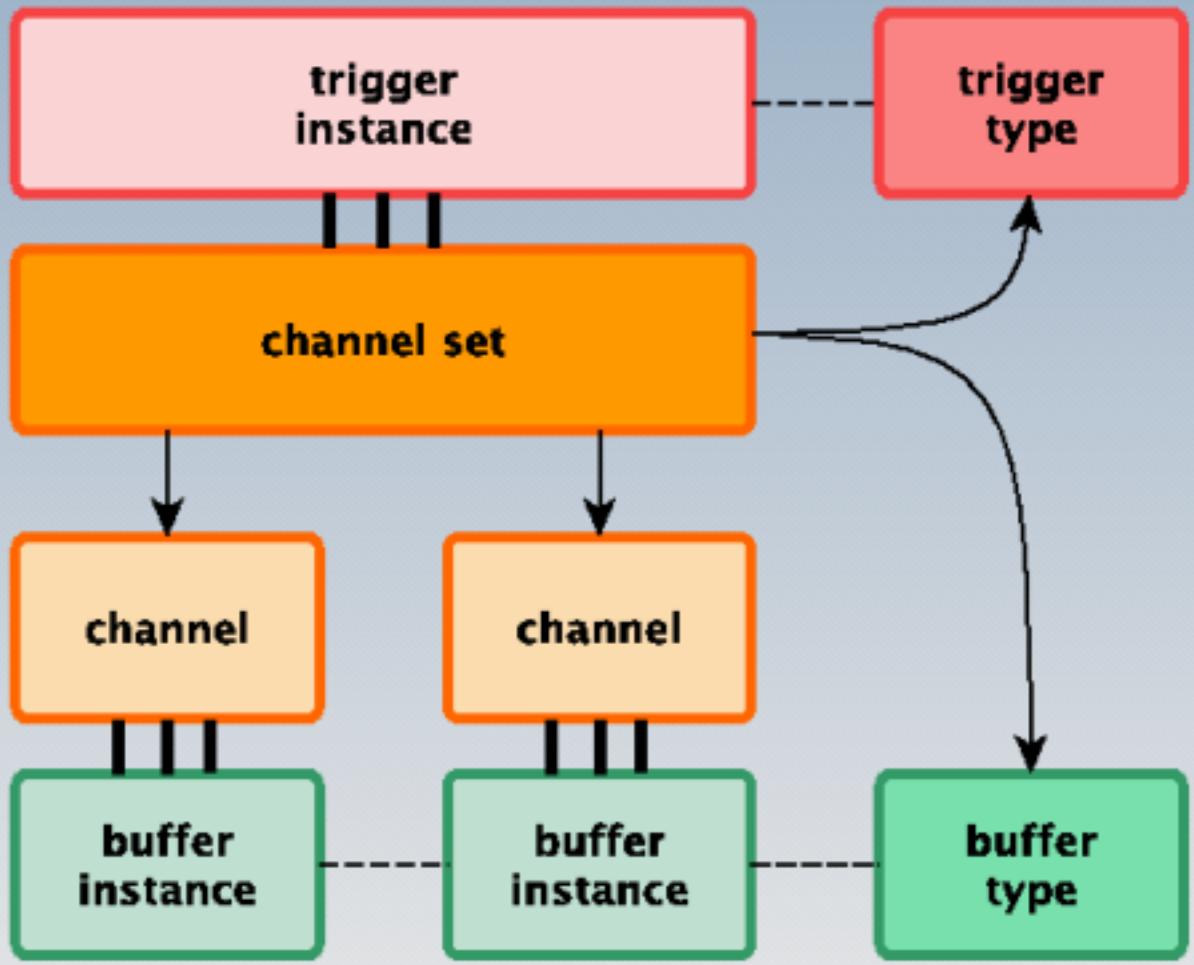
Both input and output from the start

All 3 of digital, analogue, and time

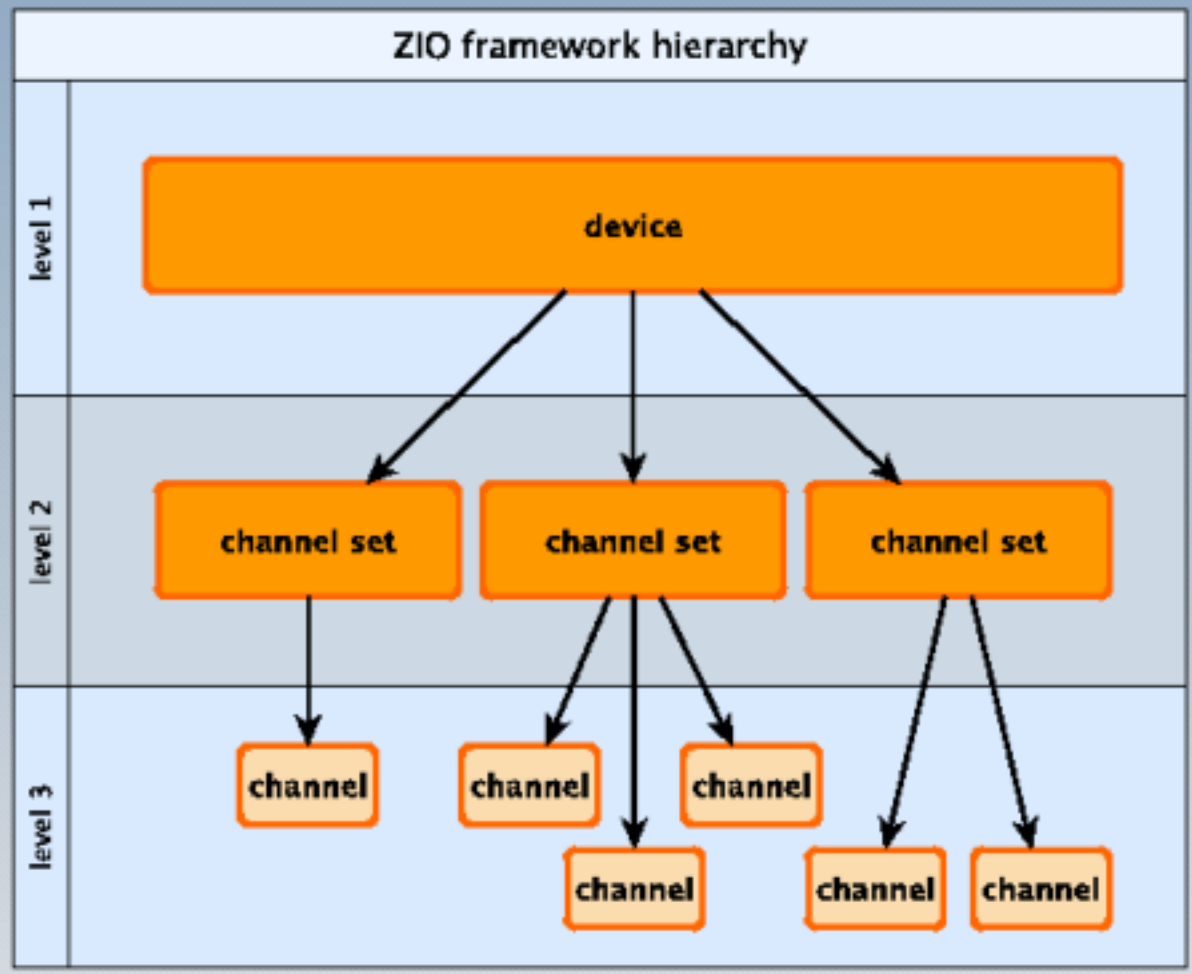
- Input: collect data at a specific time or event
- Output: drive waveforms at a specific time or event
- TDC: returns the timestamp of an input pulse
- DTC: outputs a pulse at a predefined time stamp

The Channel-Set

The basic concept of ZIO is the "cset".
Channels in a cset share data size and trigger



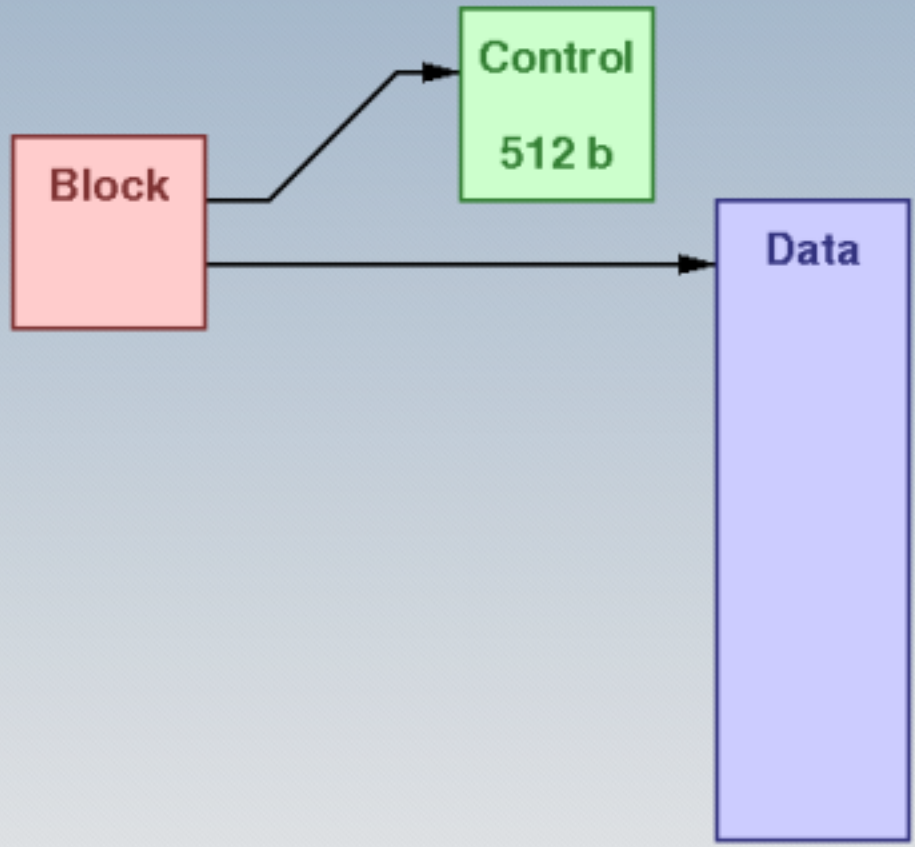
ZIO Layers


ZIO framework hierarchy
level 1
level 2
level 3
device
channel set
channel set
channel set
channel
channel
channel
channel
channel
channel

The Block

The atomic data item in ZIO is a block

- It includes both data and meta-data ("control")
- Data within ZIO never travels without meta-data.



Control Structure

```
/* byte 0 */
uint8_t major_version;
uint8_t minor_version;
uint8_t more_ctrl;      /* number of further ctrl, for interleaved    */
uint8_t alarms;         /* set by channel, persistent, write 1 to clr */

/* byte 4*/
uint32_t seq_num;       /* block sequence number                      */
uint32_t flags;         /* endianness etc, see below                  */
uint32_t nsamples;      /* number of samples in this data block       */

/* byte 16 */
uint16_t ssize;         /* sample-size for each of them, in bytes     */
uint16_t sbits;         /* sample-bits: number of valid bits          */
uint16_t cset_i;        /* index of channel-set within device         */
uint16_t chan_i;        /* index of channel within cset               */

/* byte 24 */
uint8_t hostid[8];      /* Macaddress or whatever unique              */

/* byte 32 */
struct xio_timestamp tstamp;

/* byte 56 */
uint32_t mem_offset;    /* position in mmap buffer of this block      */
uint32_t reserved;      /* possibly another offset, or space for 64b  */

/* byte 64 */
/* The control block includes what device the data belongs to         */
char devname[XIO_OBJ_NAME_LEN];

/* byte 76 */
/* Each data block is associated with a trigger and its features      */
char triggername[XIO_OBJ_NAME_LEN];

/* byte 88 */
struct xio_ctrl_attr attr_channel;
struct xio_ctrl_attr attr_trigger;

/* byte 488 */
uint8_t __fill_end[XIO_CONTROL_SIZE - 488];
```

ZIO Device

Physically, the device is a PCB or a chip

Logically, it is a probe unit and a kernel module.

For ZIO, it is simply a group of csets.

Triggers

A trigger is a software module that requests I/O.

- Time-driven (kernel timer or hardware-internal)
- Event-driven (external interrupt or hardware-internal)
- Data-driven (in-driver monitoring or hardware-internal)
- Transparent (requests I/O when user reads or writes)

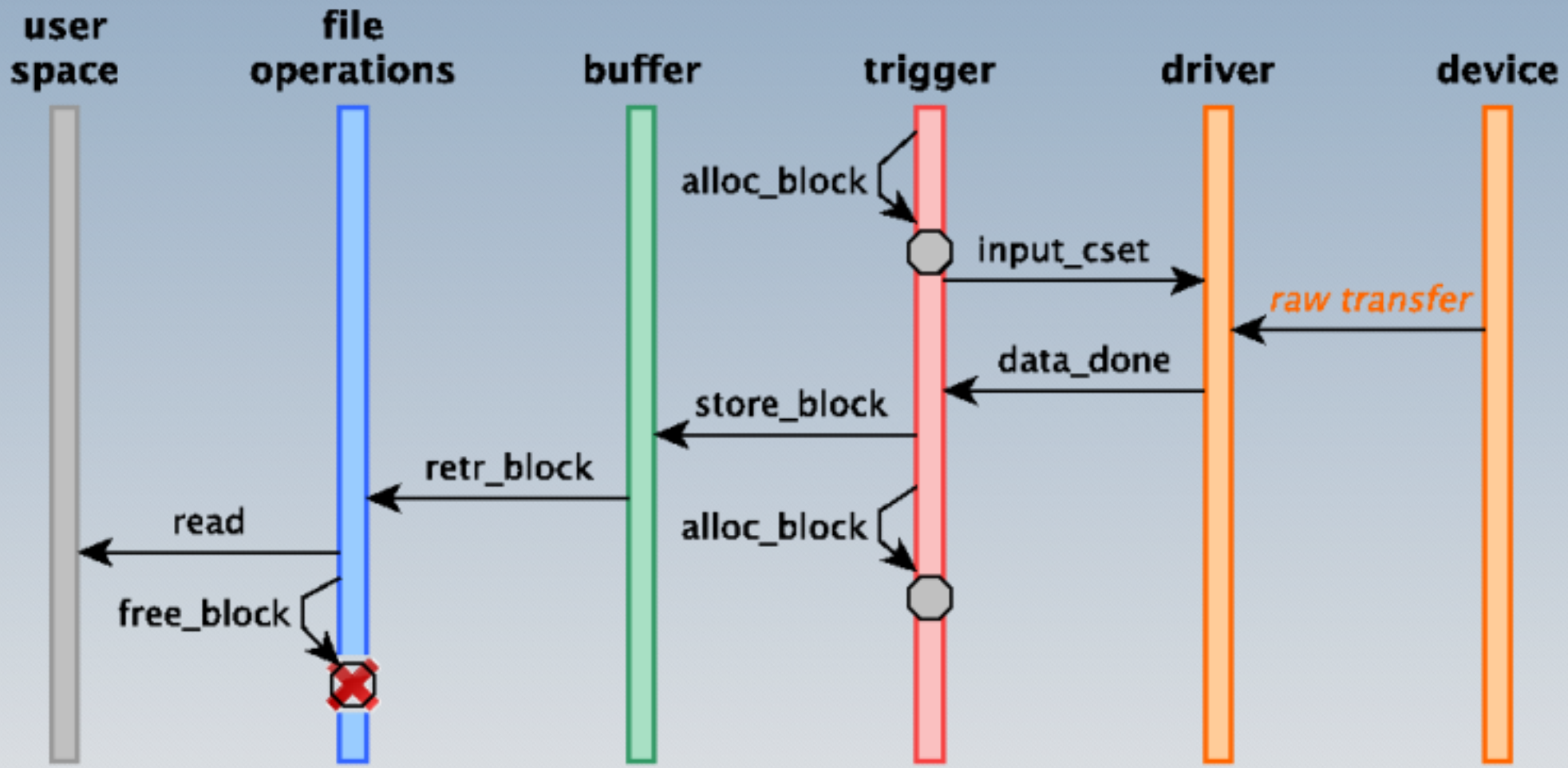
Buffers

A buffer is a software module between trigger and user

- Kmalloc-based (only read/write)
- Vmalloc-based (mmap capable)
- DMA-oriented (maybe device-specific)
- On-board memory (device-specific)
- Software ring buffer (discarding metadata)

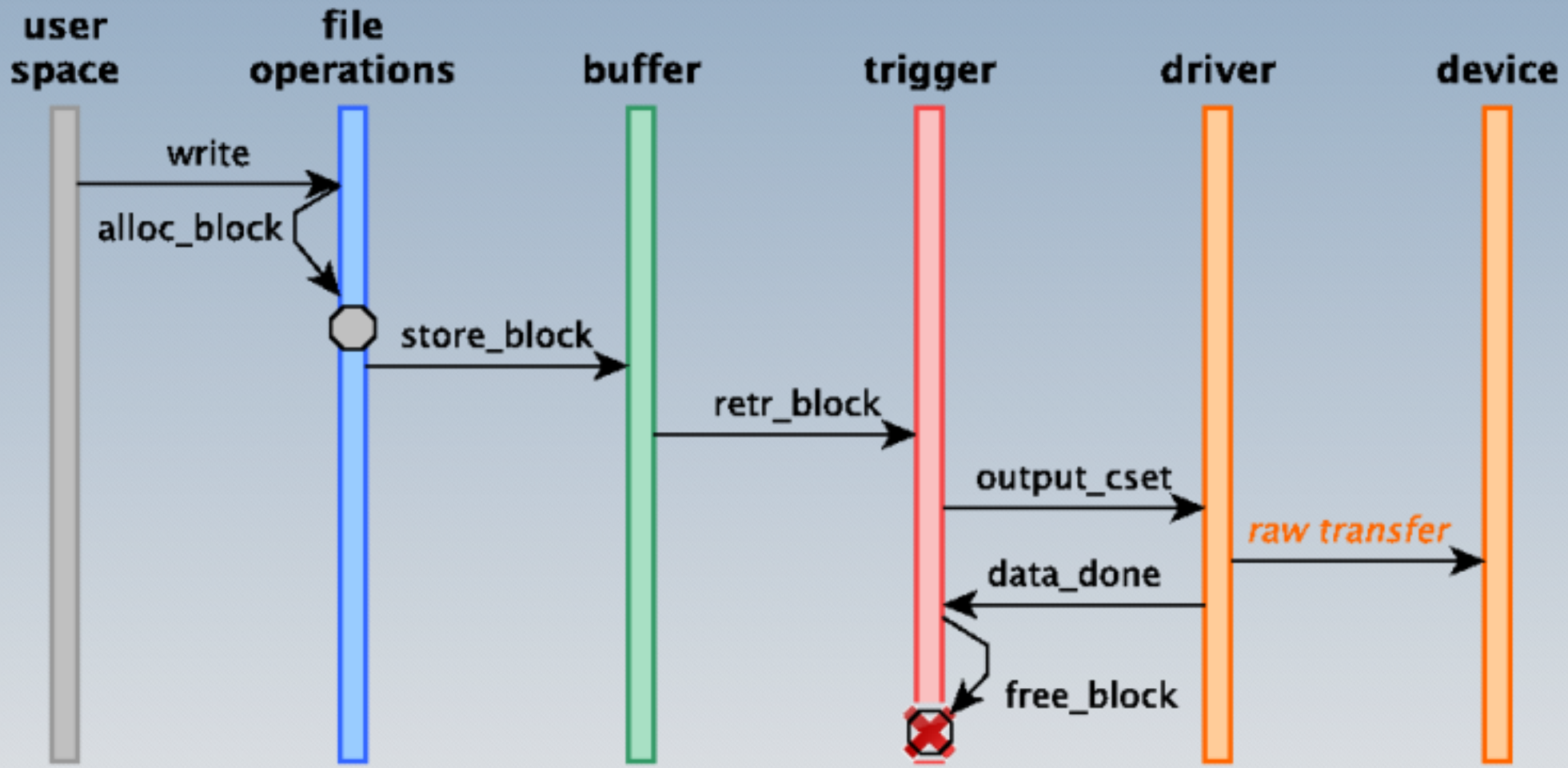
Input Data Flow

This is the input pipeline in ZIO (time flows down)



Output Data Flow

The output pipeline is symmetrical

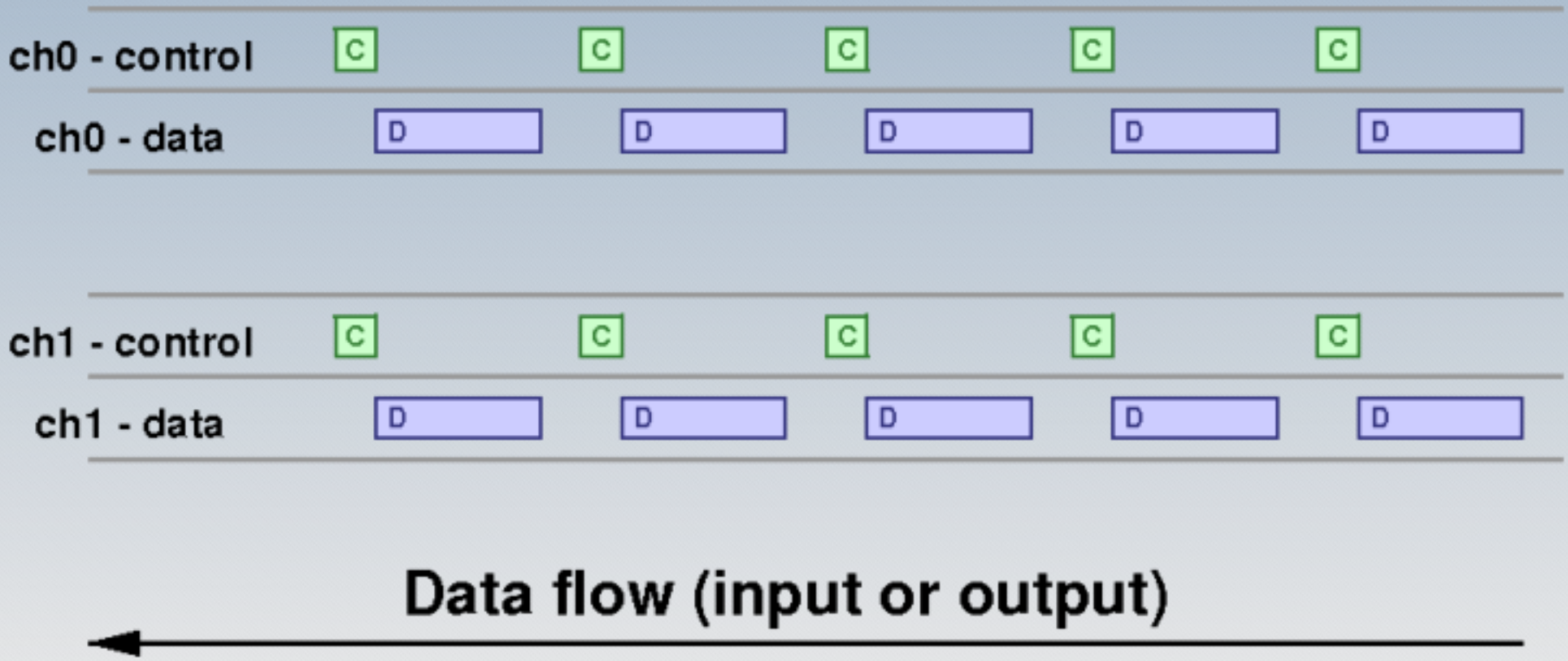


Char Devices

Data and metadata travel in two different devices

Control device: 512-byte fixed-size structures

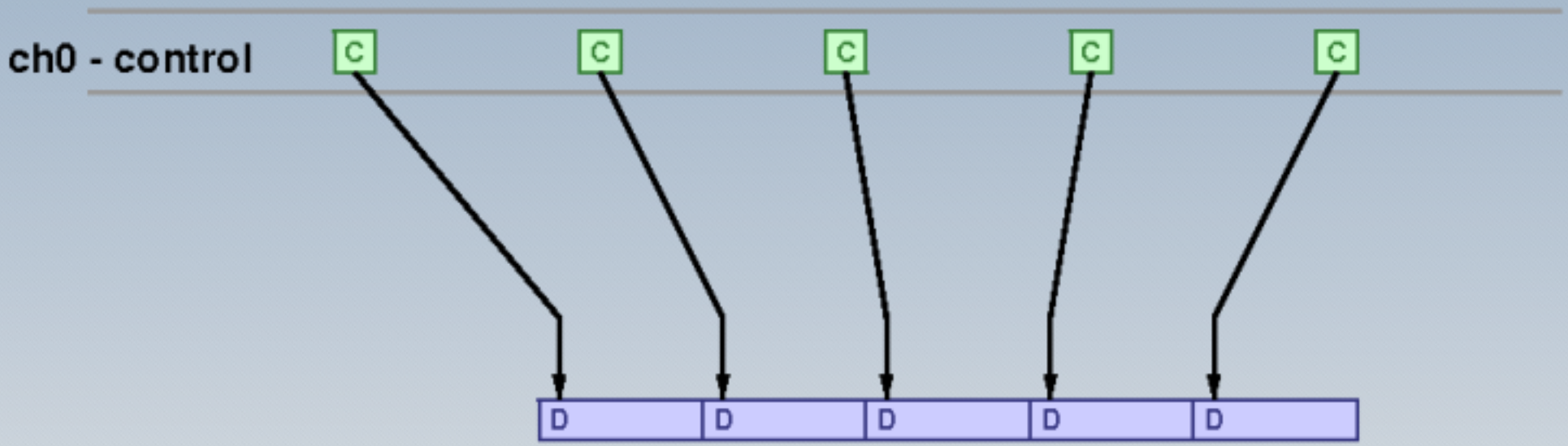
Data device: blobs of samples



Mmap Support

Using mmap (or DMA to user space) is trivial

The control structure includes the data_offset

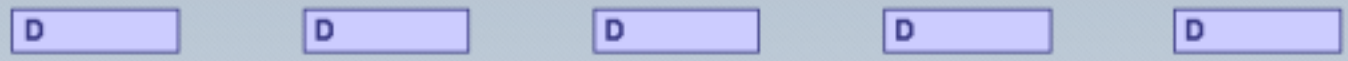


The control channel times I/O and refers to mmap data

The Future: PF_ZIO

The next research idea is PF_ZIO, for I/O networks

SOCK_STREAM



SOCK_DGRAM

SOCK_RAW

PF_ZIO is not ZIO over Ethernet

Applications will perform I/O by exchanging frames

The PF_ZIO address space is I/O channels

A host may drive hundreds of channels over a field bus

Sockets may prove better than hundreds of char devices

Zero-copy networking will help with high data rates

Implementation Status (2012-02-05)

Software-only modules, for stress-test and benchmark
Simple hardware modules
(Hardware for the real use-case is almost ready)

device: zio-zero (input and output, raw or timely)
device: line discipline (input: uart or pty for stress-test)
device: GPIO (input and output)
device: AD7888/AD7887 (SPI input)
device: TDC/DTC

trigger: kernel timer
trigger: transparent trigger (user-driven)
trigger: external irq or external GPIO

buffer: kmalloc
buffer: vmalloc (mmap capable)
buffer: cbuf (SOCK_STREAM alike, coalescing blocks)

Thank you for your attention

http://www.ohwr.org/projects/zio

git://ohwr.org/misc/zio.git

http://www.ohwr.org/projects/zio/repository
http://www.ohwr.org/projects/zio/wiki
http://www.ohwr.org/projects/zio/documents

Alessandro Rubini, Federico Vaga