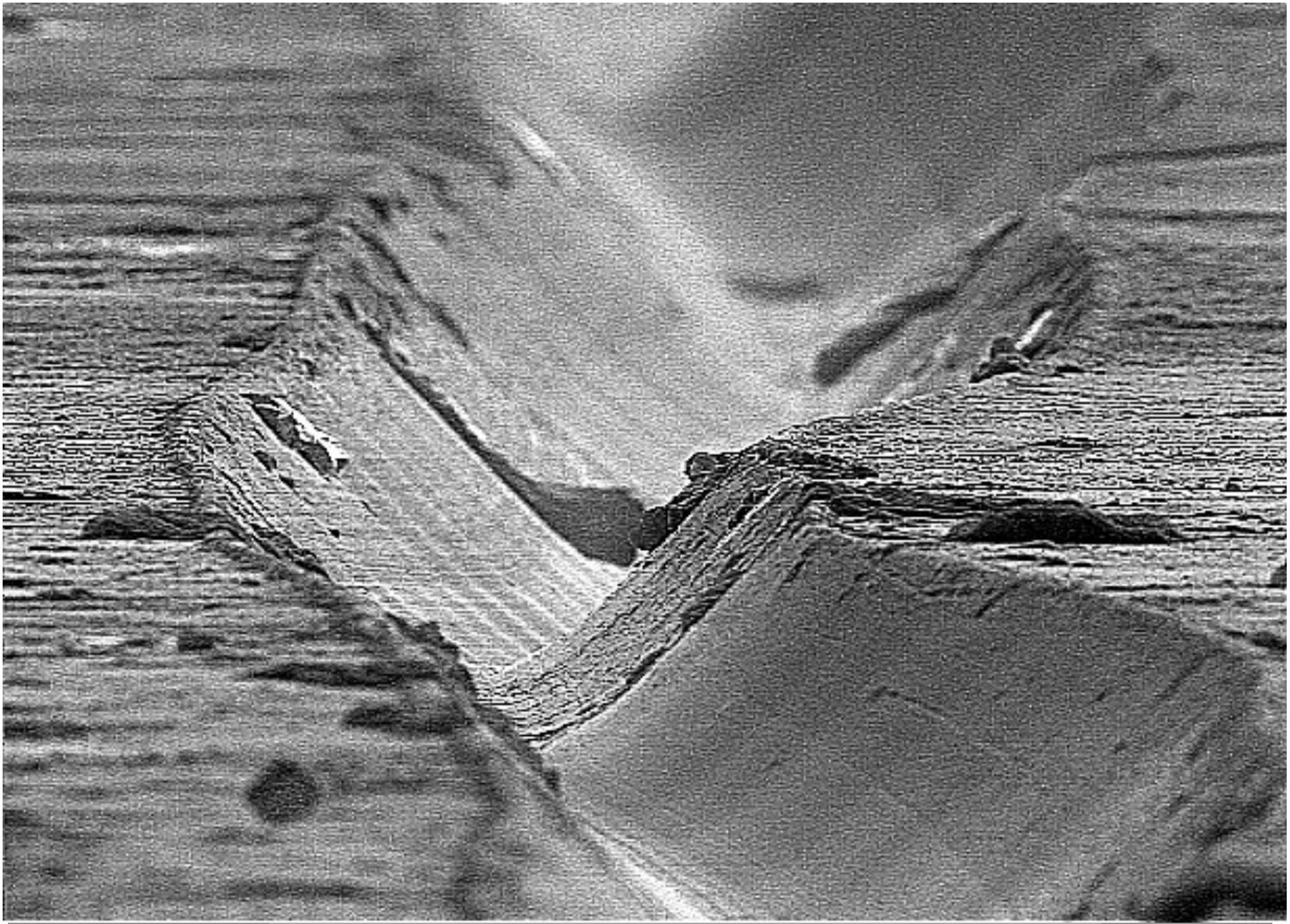
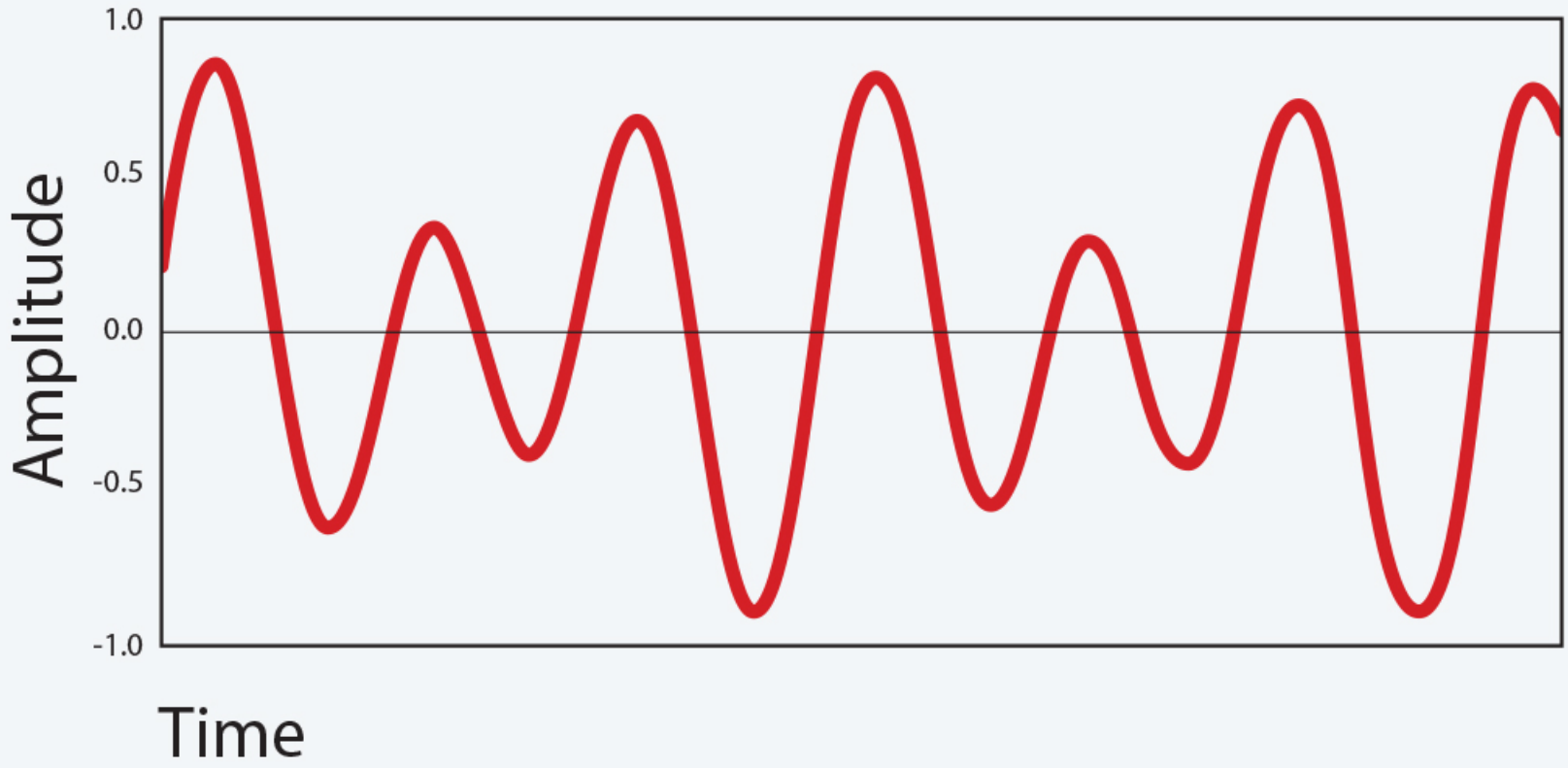
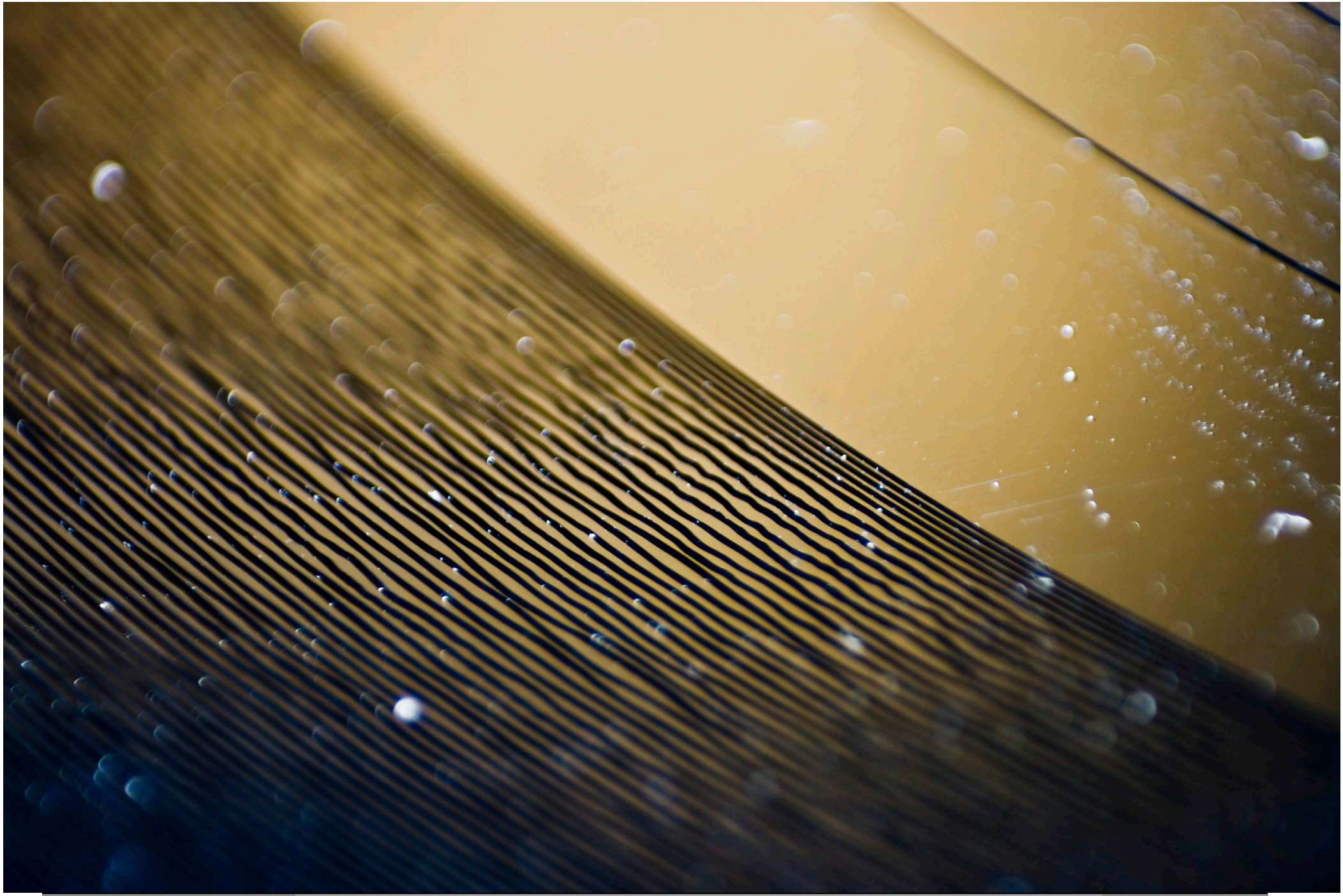
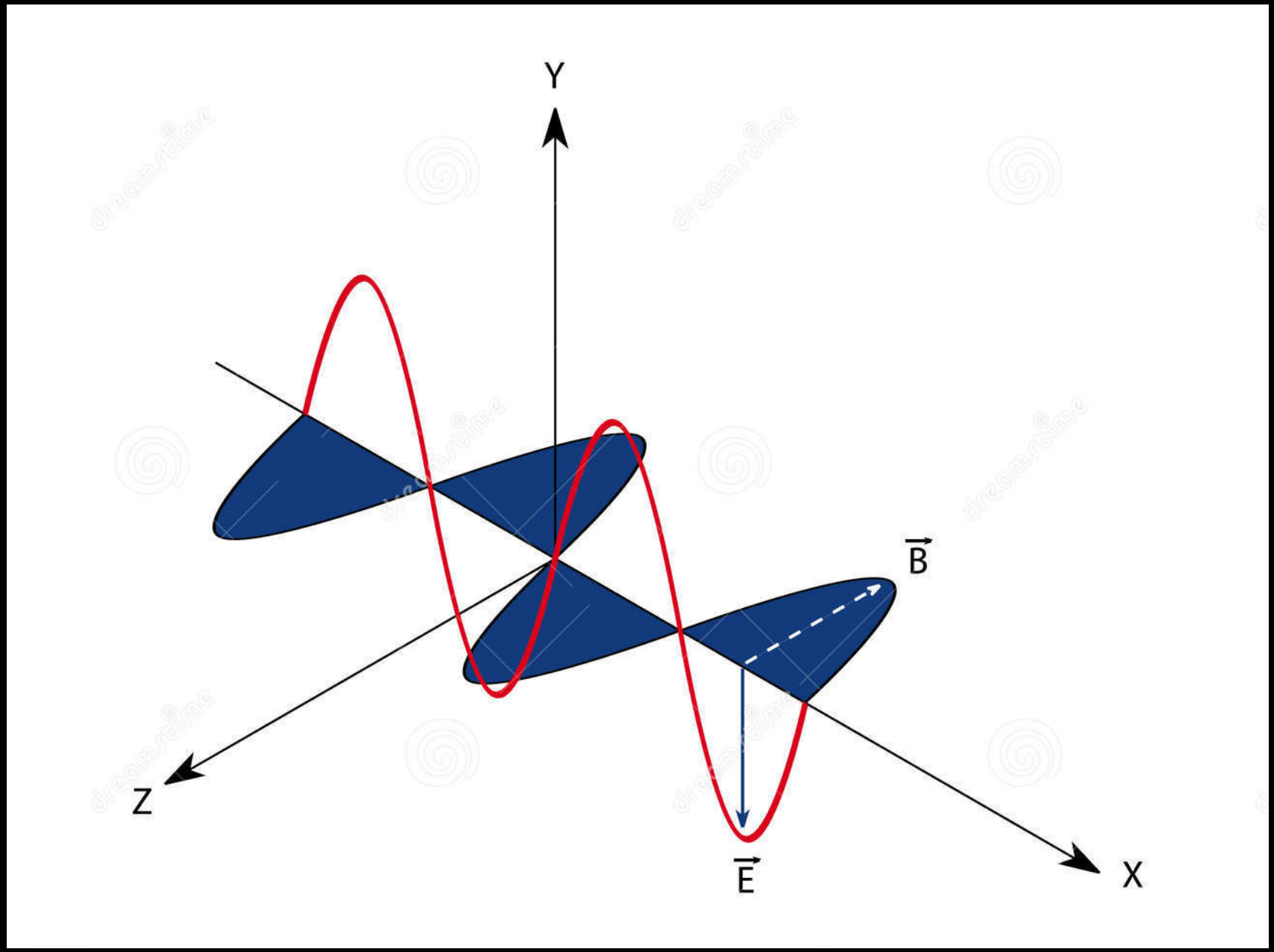


Analog Waveform

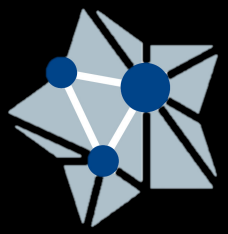





Y
B
Z
E
X

L04: VLBI data

acquisition, formats and transfer



JIVE
Joint Institute for VLBI
ERIC

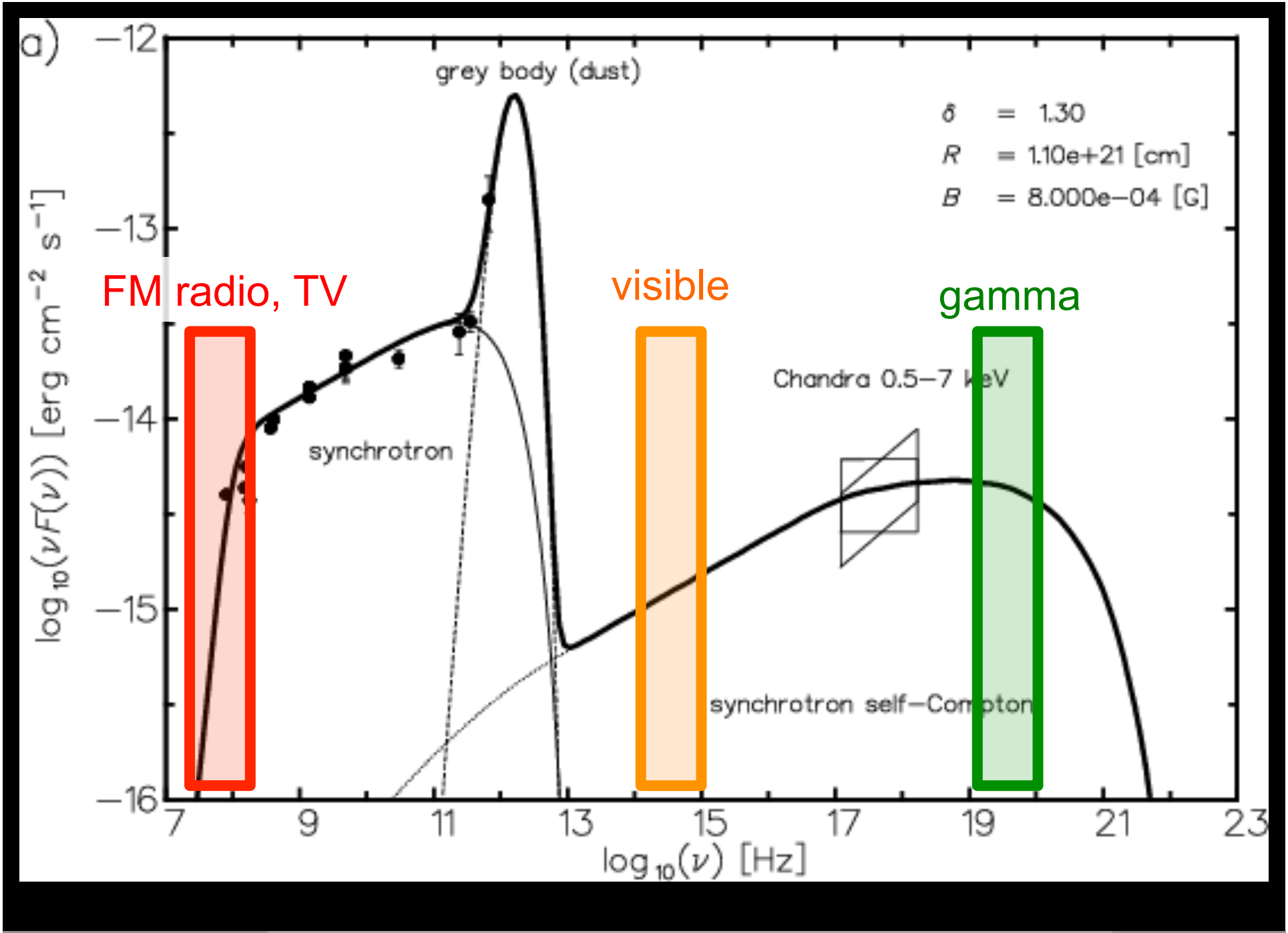
Harro Verkouter

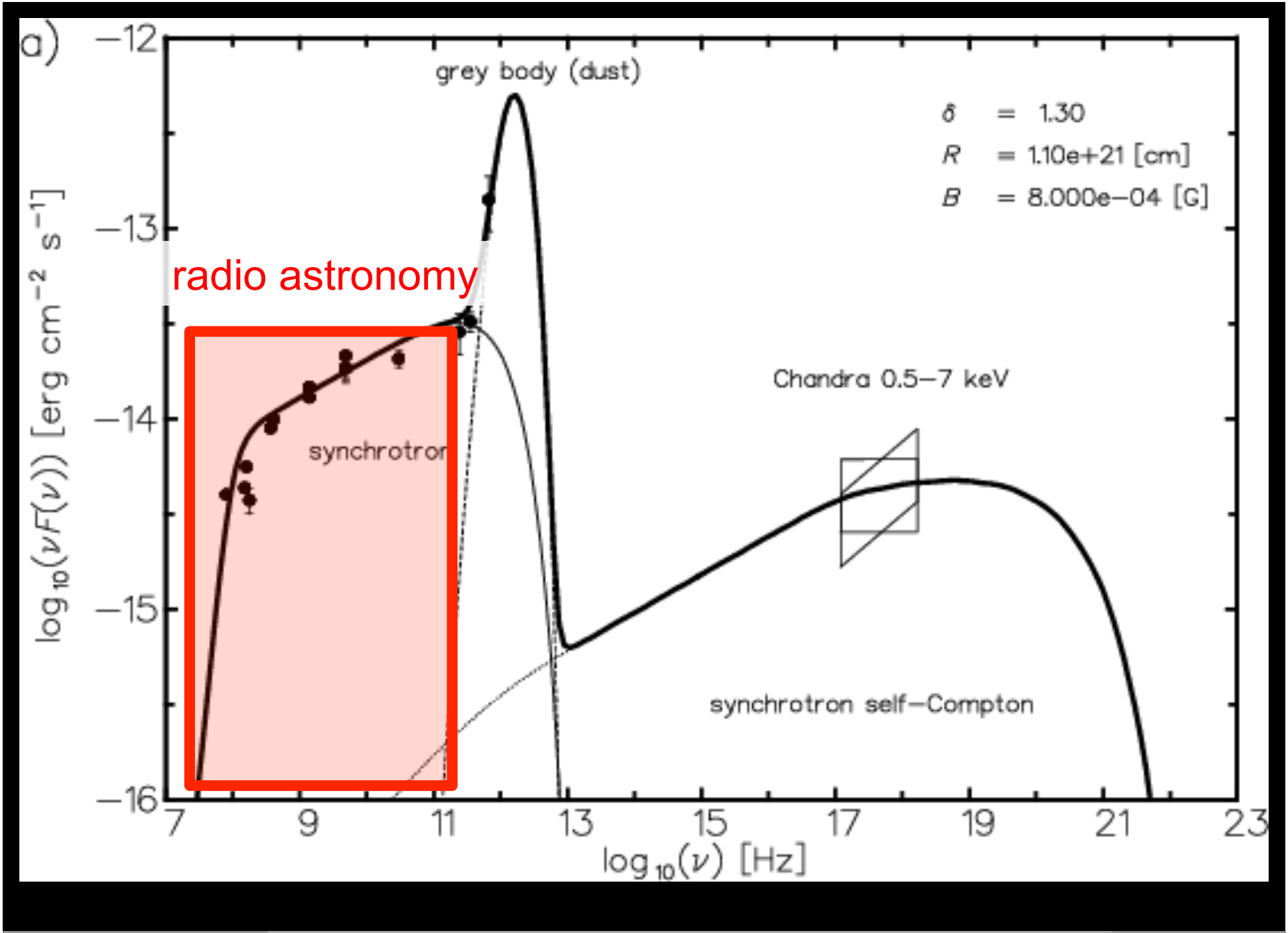
?


WD740

Analog

Digital

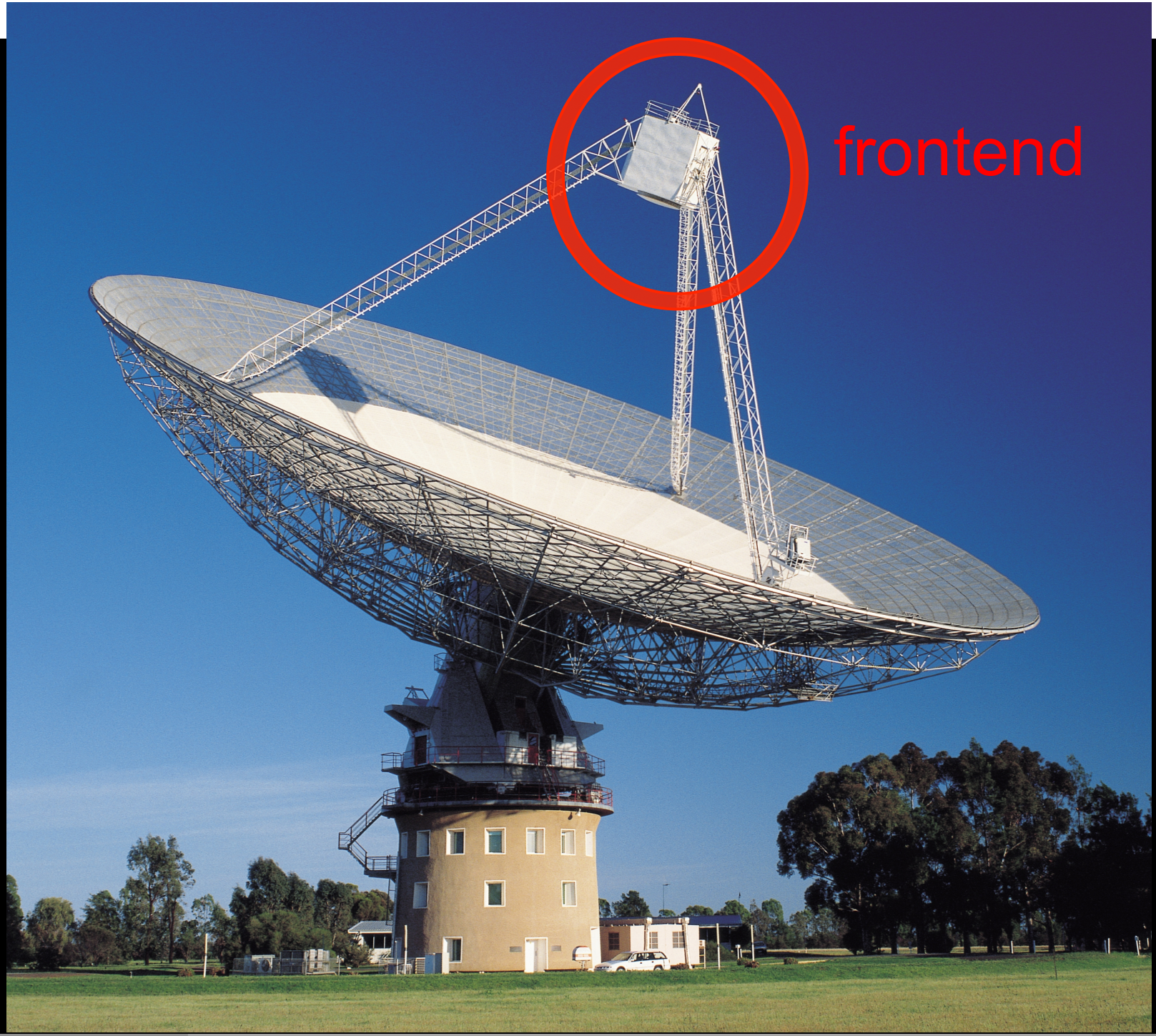

a)
grey body (dust)
δ = 1.30
R = 1.10e+21 [cm]
B = 8.000e−04 [G]
FM radio, TV
visible
gamma
Chandra 0.5−7 keV
synchrotron
synchrotron self−Compton
log10(νF(ν)) [erg cm−2 s−1]
log10(ν) [Hz]
−12
−13
−14
−15
−16
7
9
11
13
15
17
19
21
23

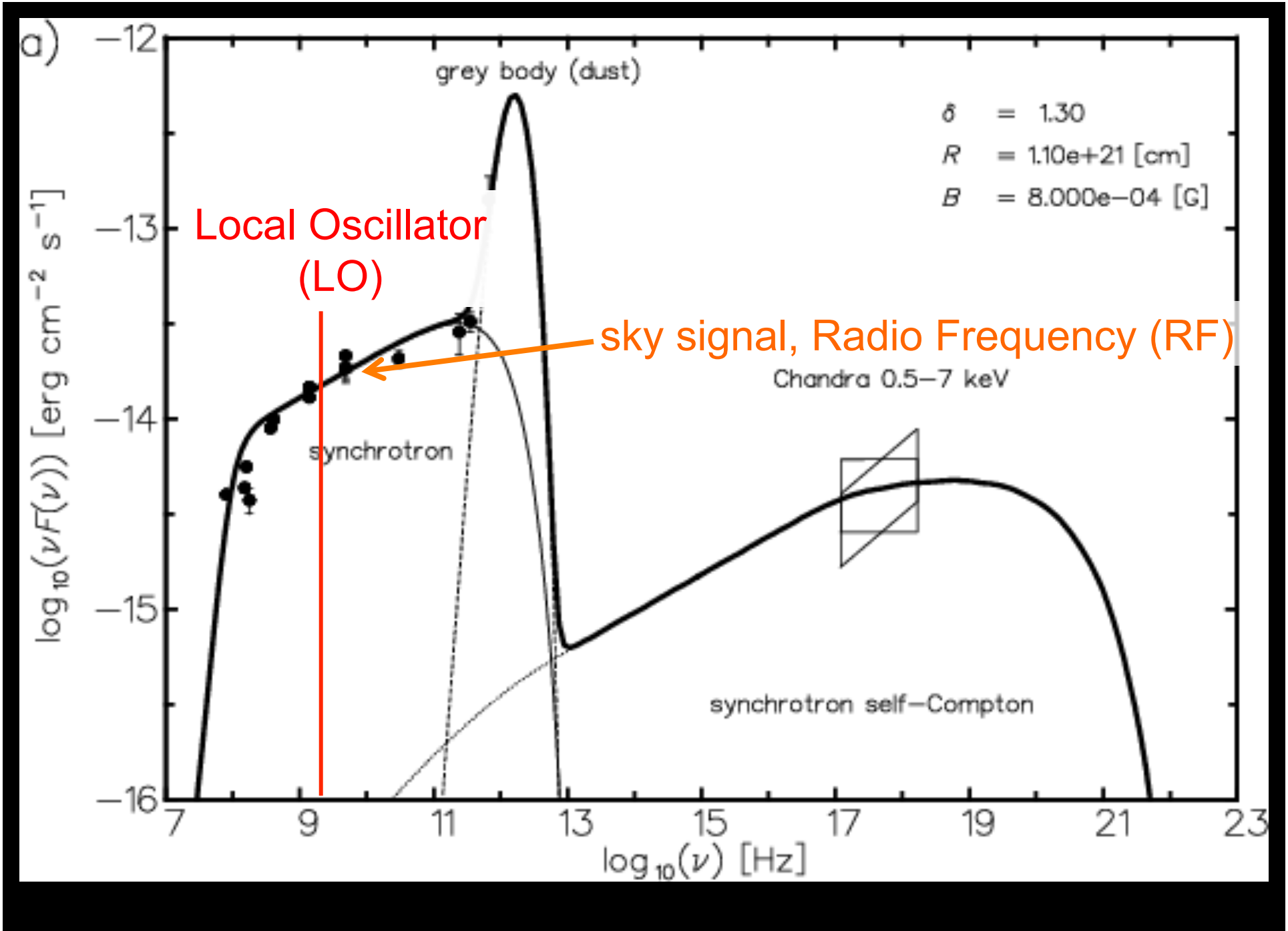

a)
grey body (dust)
δ = 1.30
R = 1.10e+21 [cm]
B = 8.000e−04 [G]
radio astronomy
synchrotron
Chandra 0.5–7 keV
synchrotron self–Compton
log10(νF(ν)) [erg cm−2 s−1]
log10(ν) [Hz]
−12
−13
−14
−15
−16
7
9
11
13
15
17
19
21
23

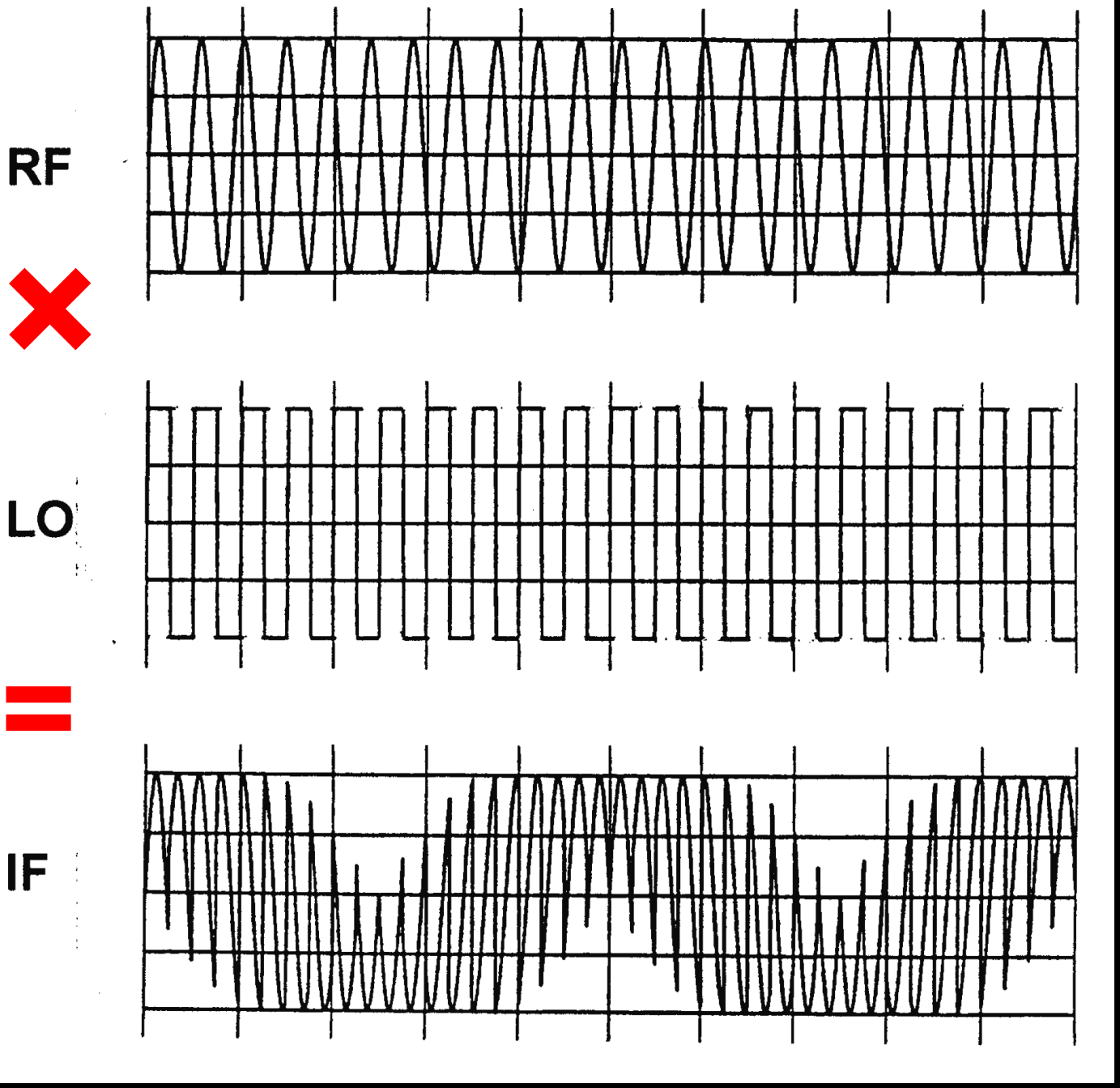
1-100GHz s s!

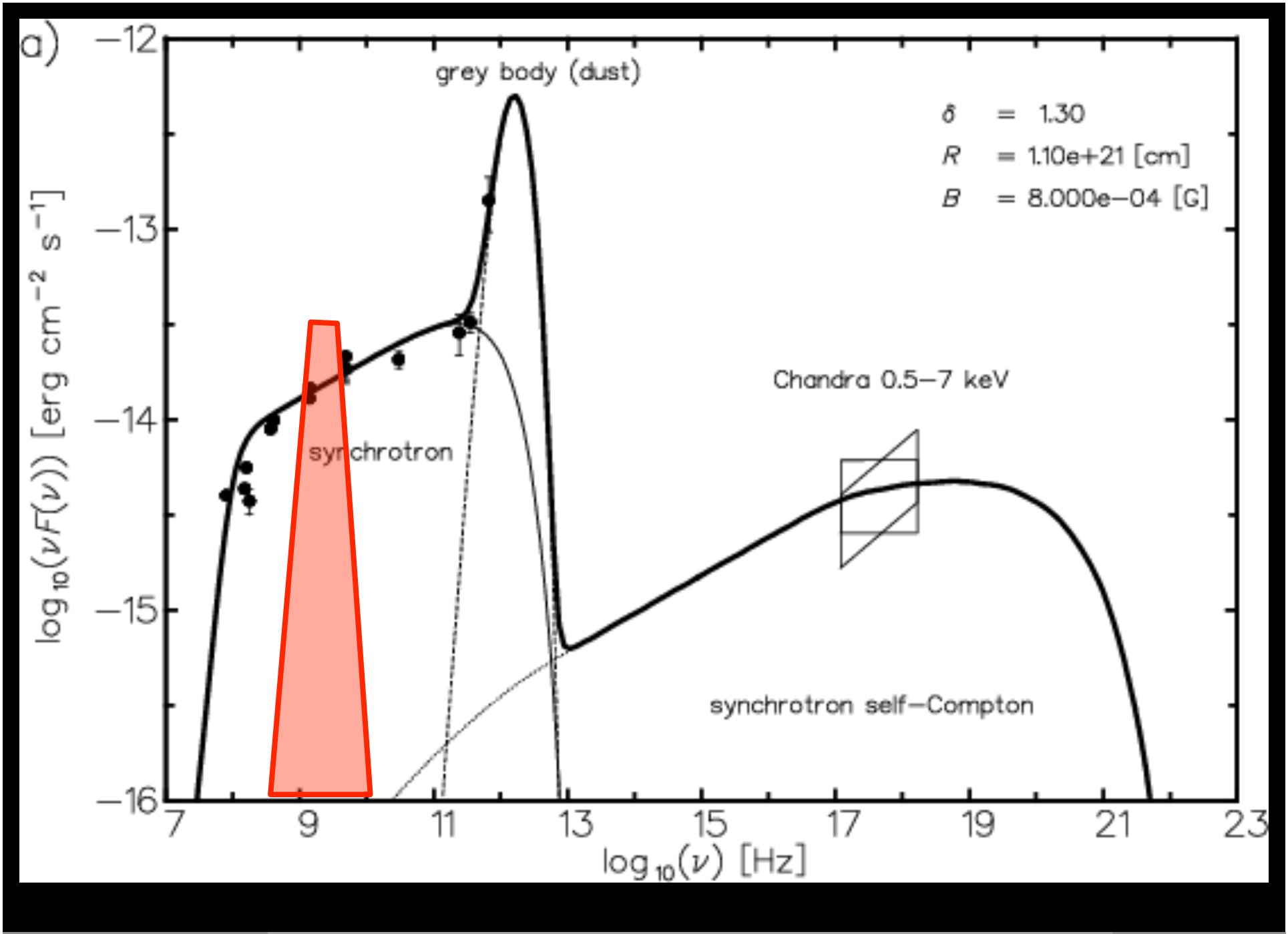
Nyqu = 2-200 Gs/s

A/D ve rs: Ms/s

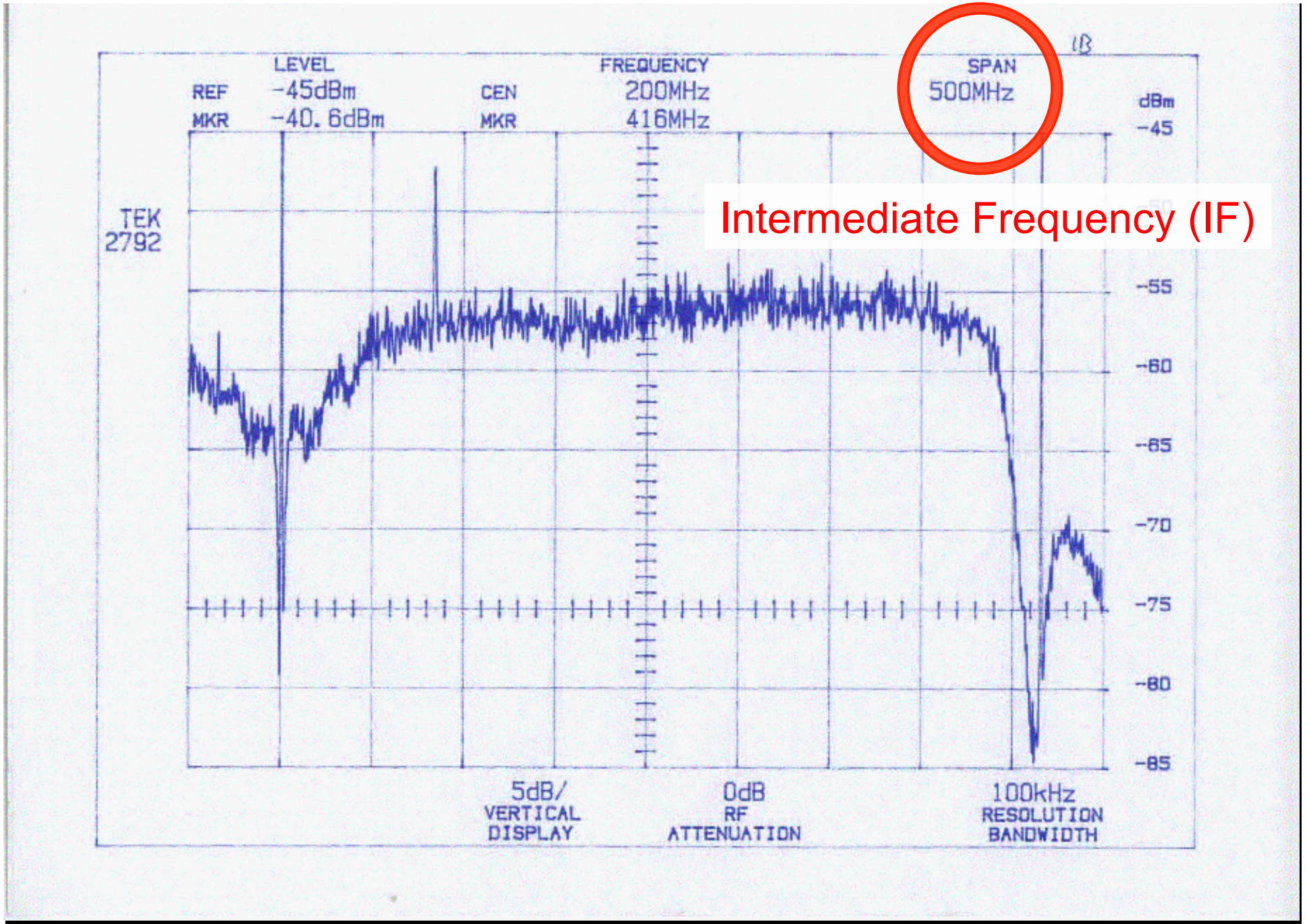

frontend

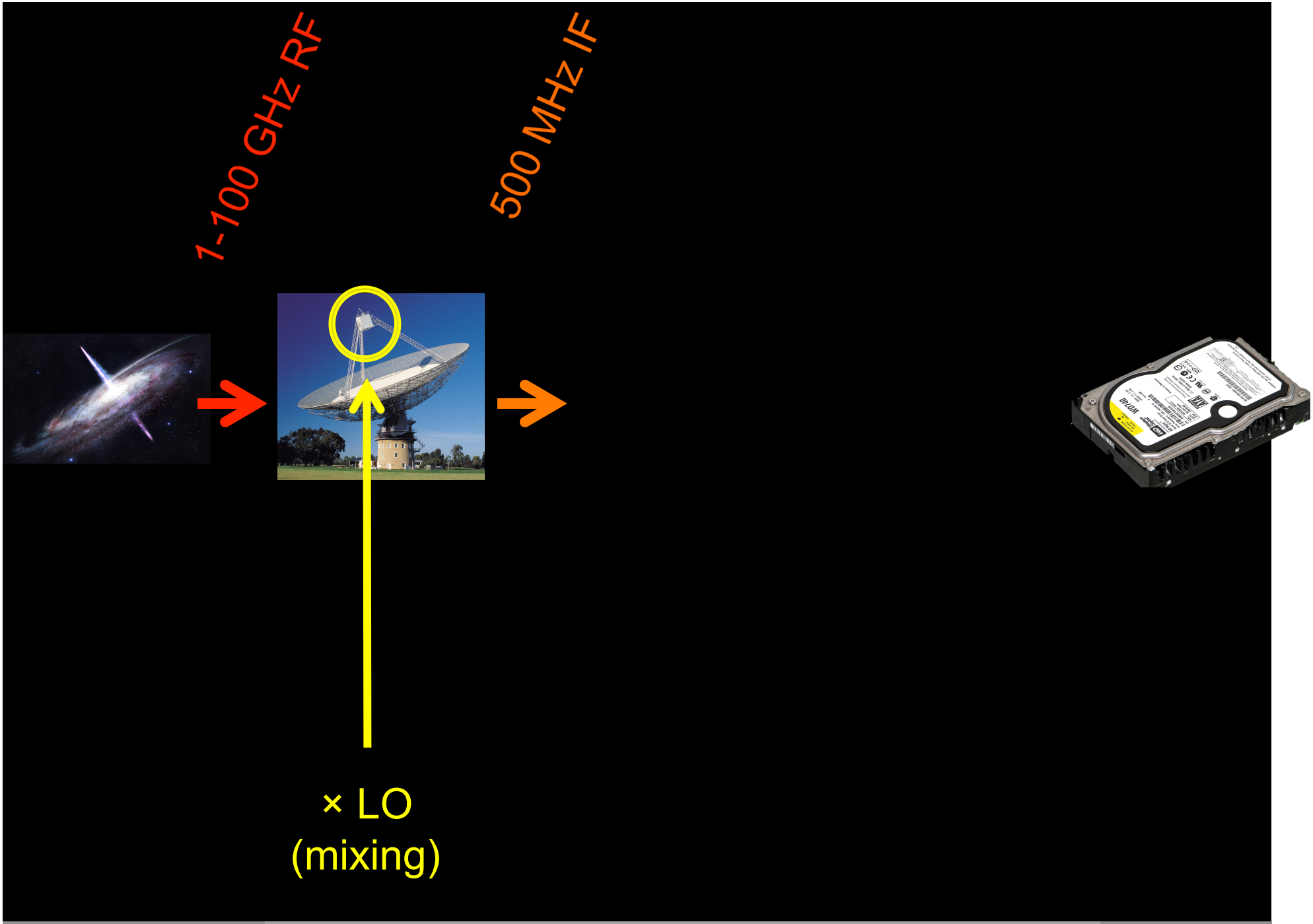

a)
grey body (dust)
δ = 1.30
R = 1.10e+21 [cm]
B = 8.000e−04 [G]
Local Oscillator
(LO)
sky signal, Radio Frequency (RF)
Chandra 0.5–7 keV
synchrotron
synchrotron self–Compton
log10(νF(ν)) [erg cm−2 s−1]
log10(ν) [Hz]
−12
−13
−14
−15
−16
7
9
11
13
15
17
19
21
23

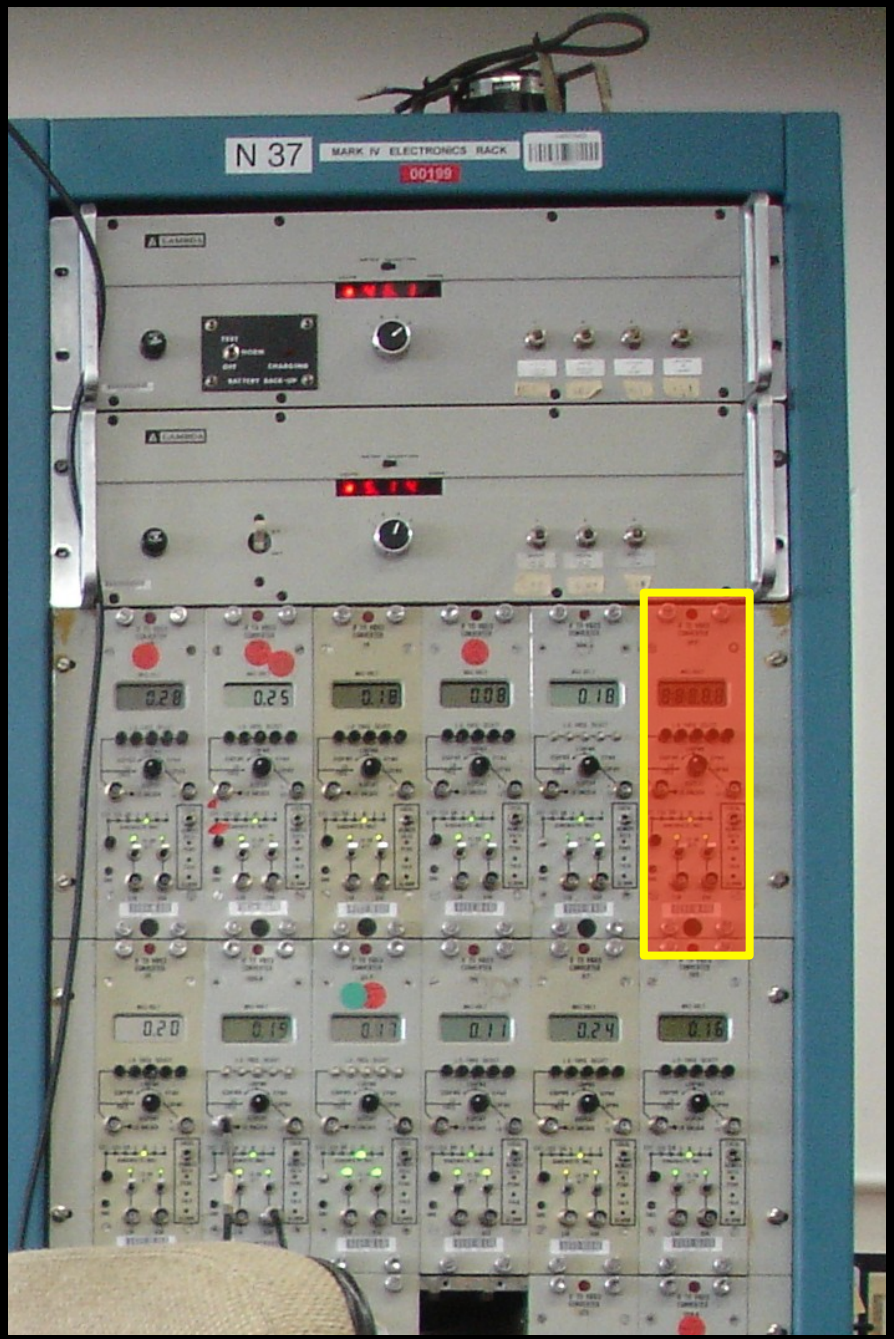

RF
LO
IF


a)
grey body (dust)
δ = 1.30
R = 1.10e+21 [cm]
B = 8.000e−04 [G]
Chandra 0.5−7 keV
synchrotron
synchrotron self−Compton
log10(νF(ν)) [erg cm−2 s−1]
log10(ν) [Hz]
−12
−13
−14
−15
−16
7
9
11
13
15
17
19
21
23


IF wave guide/transmissionline
~500MHz bandwidth

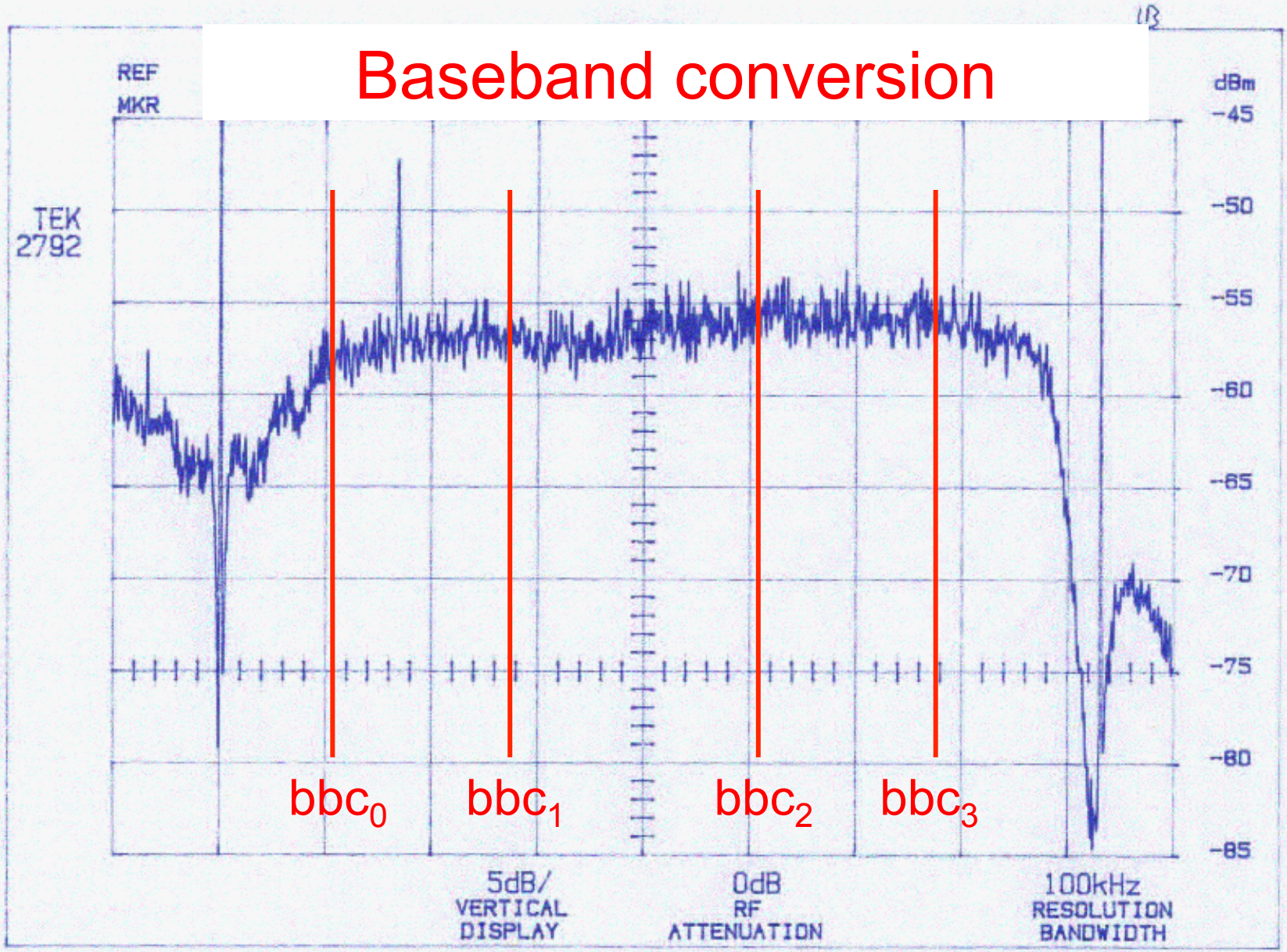

LEVEL
REF -45dBm
MKR -40.6dBm
FREQUENCY
CEN 200MHz
MKR 416MHz
SPAN
500MHz
dBm
-45
-55
-60
-65
-70
-75
-80
-85
TEK
2792
Intermediate Frequency (IF)
5dB/
VERTICAL
DISPLAY
0dB
RF
ATTENUATION
100kHz
RESOLUTION
BANDWIDTH


1-100 GHz RF
500 MHz IF
× LO
(mixing)

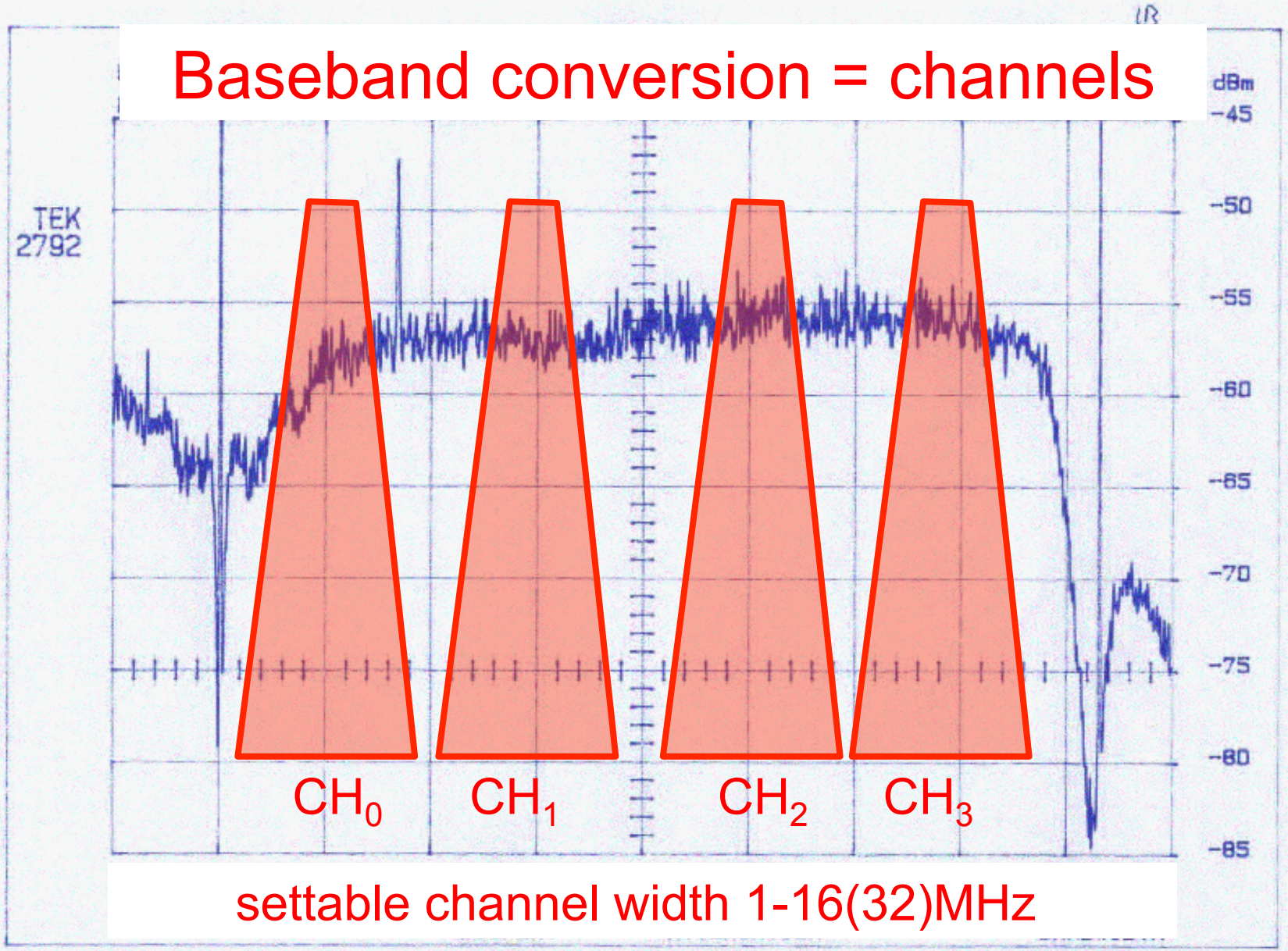


1 tuneable BBC

Baseband conversion

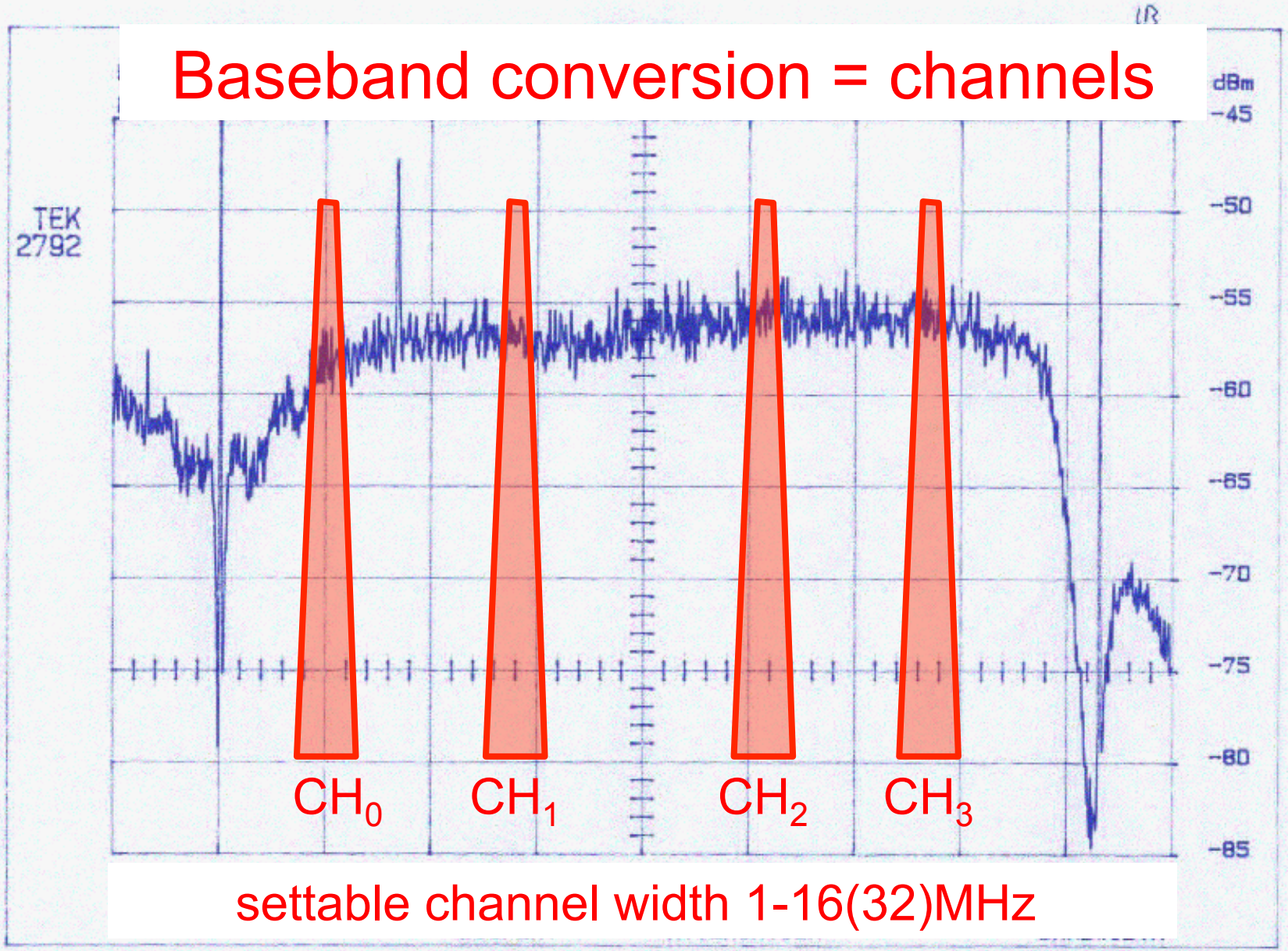


Baseband conversion = channels

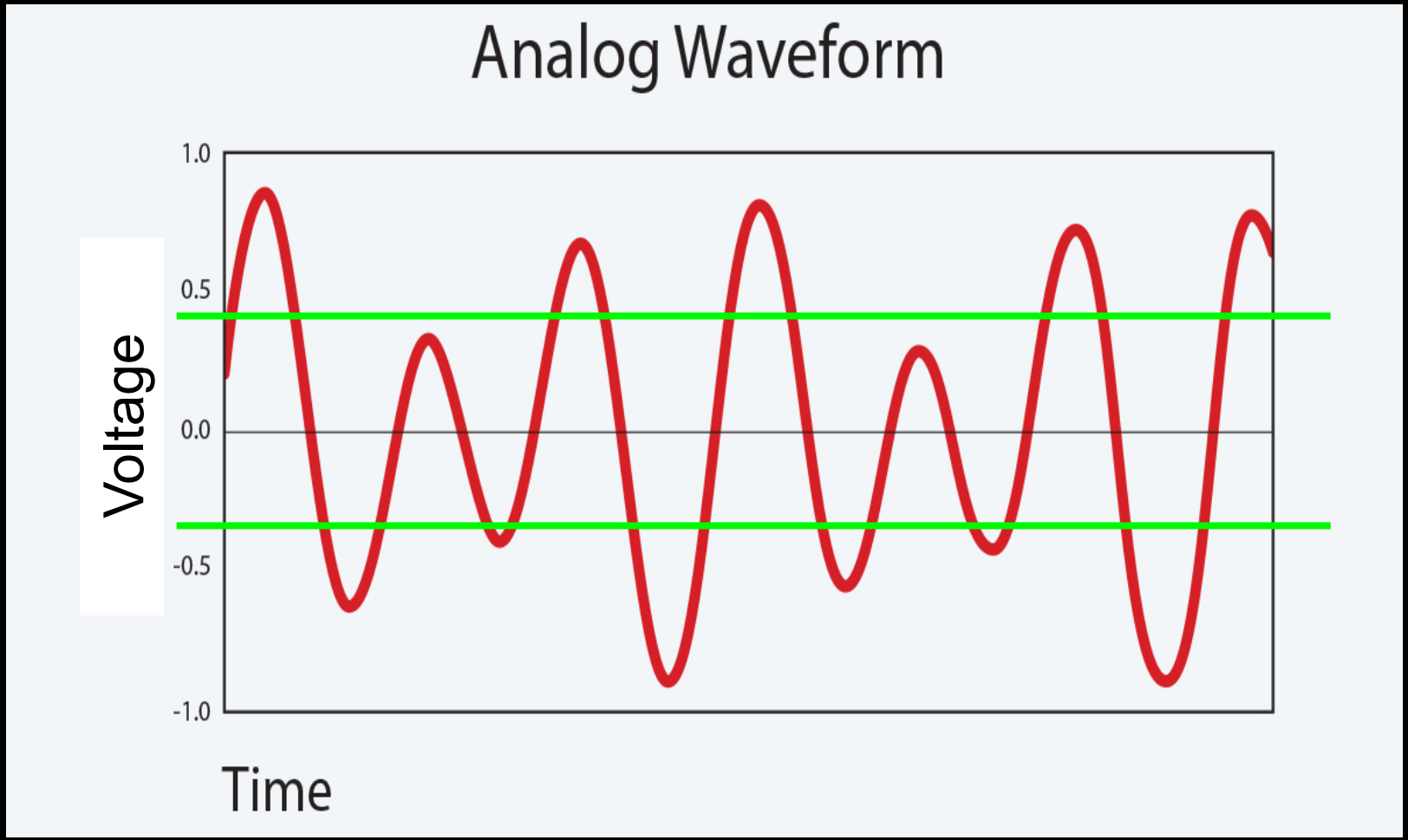


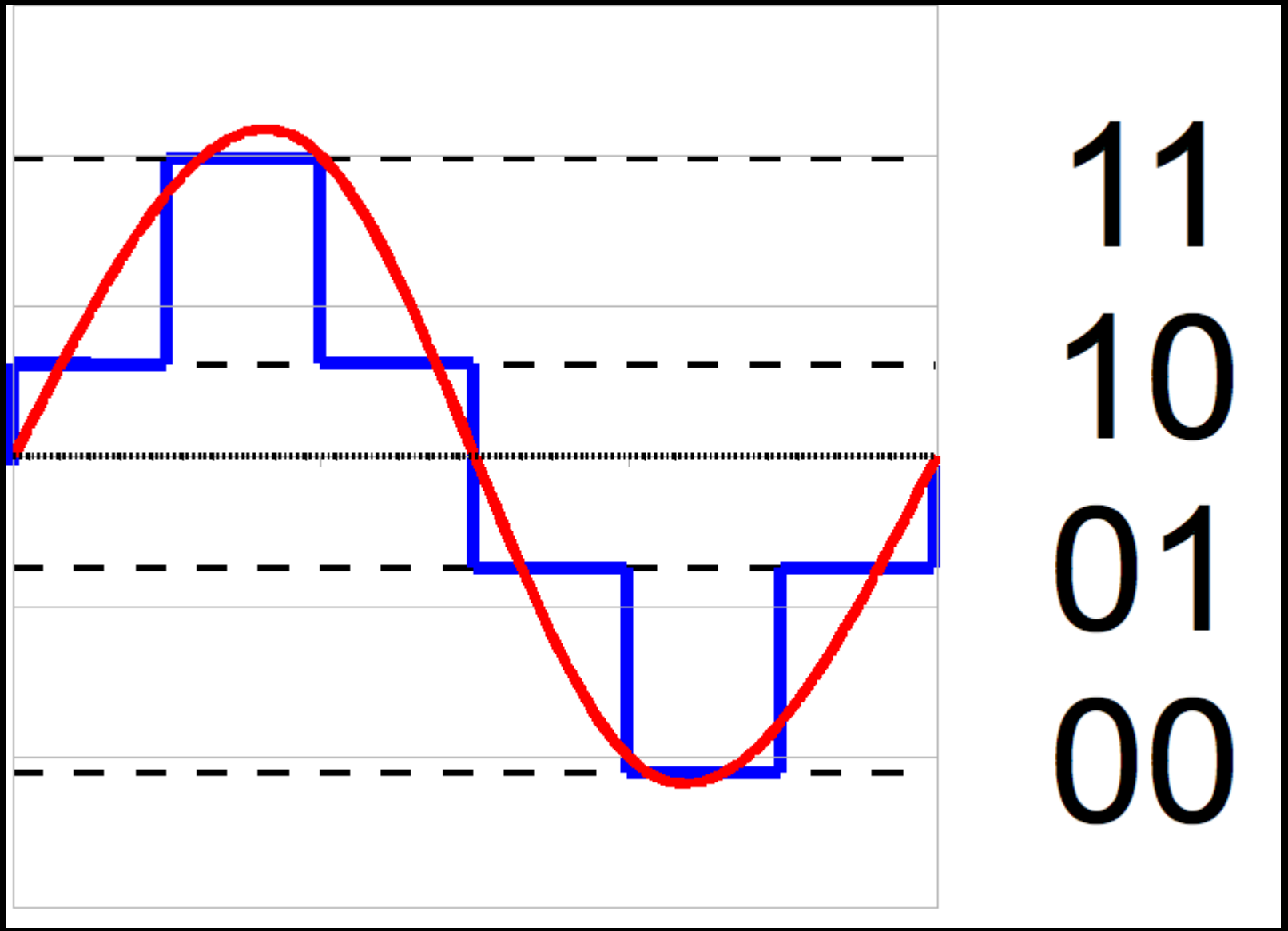
settable channel width 1-16(32)MHz

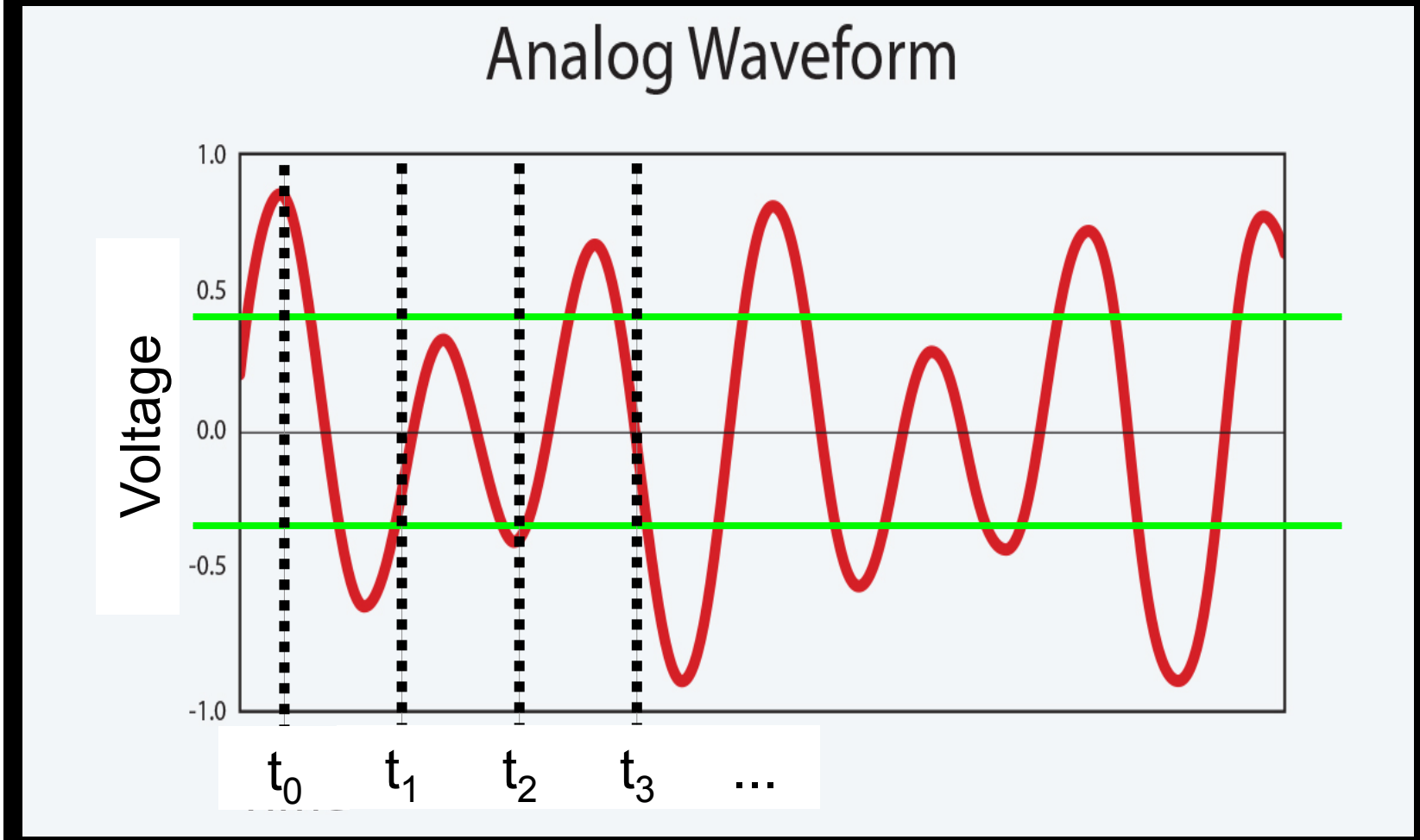
Baseband conversion = channels



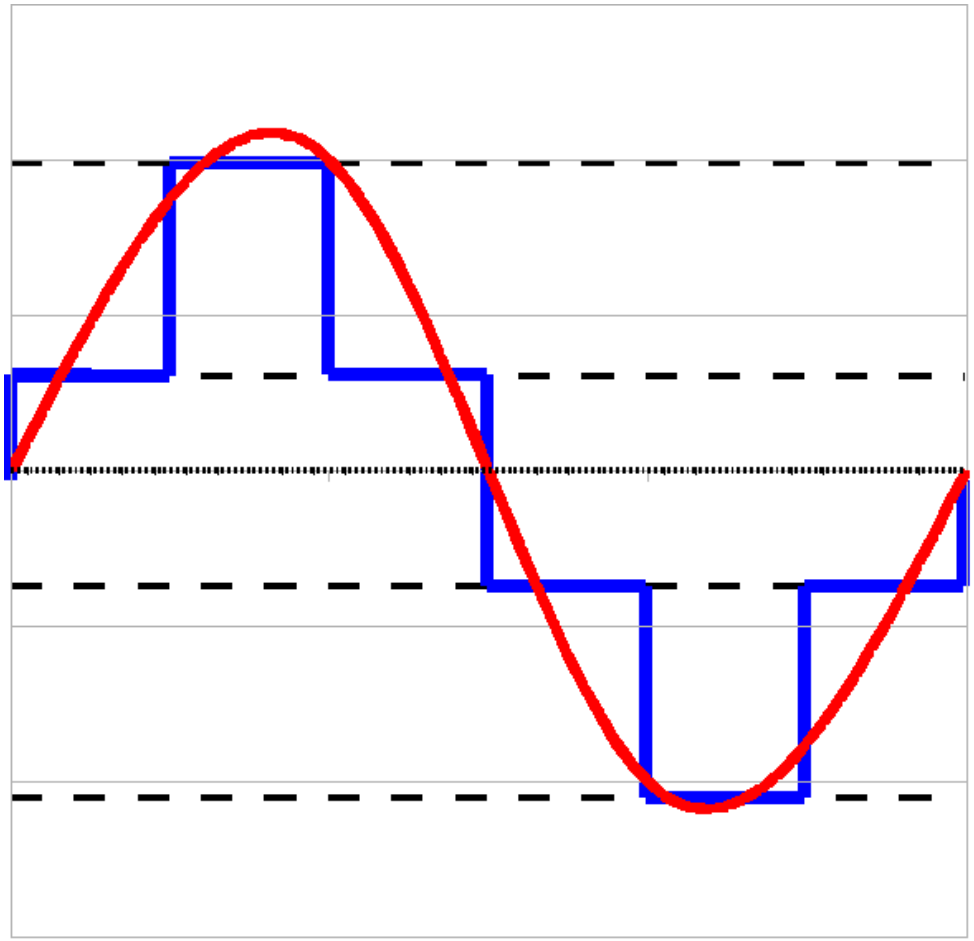
settable channel width 1-16(32)MHz


Analog Waveform
Voltage
1.0
0.5
0.0
-0.5
-1.0
Time
+Vthr
-Vthr


11
10
01
00



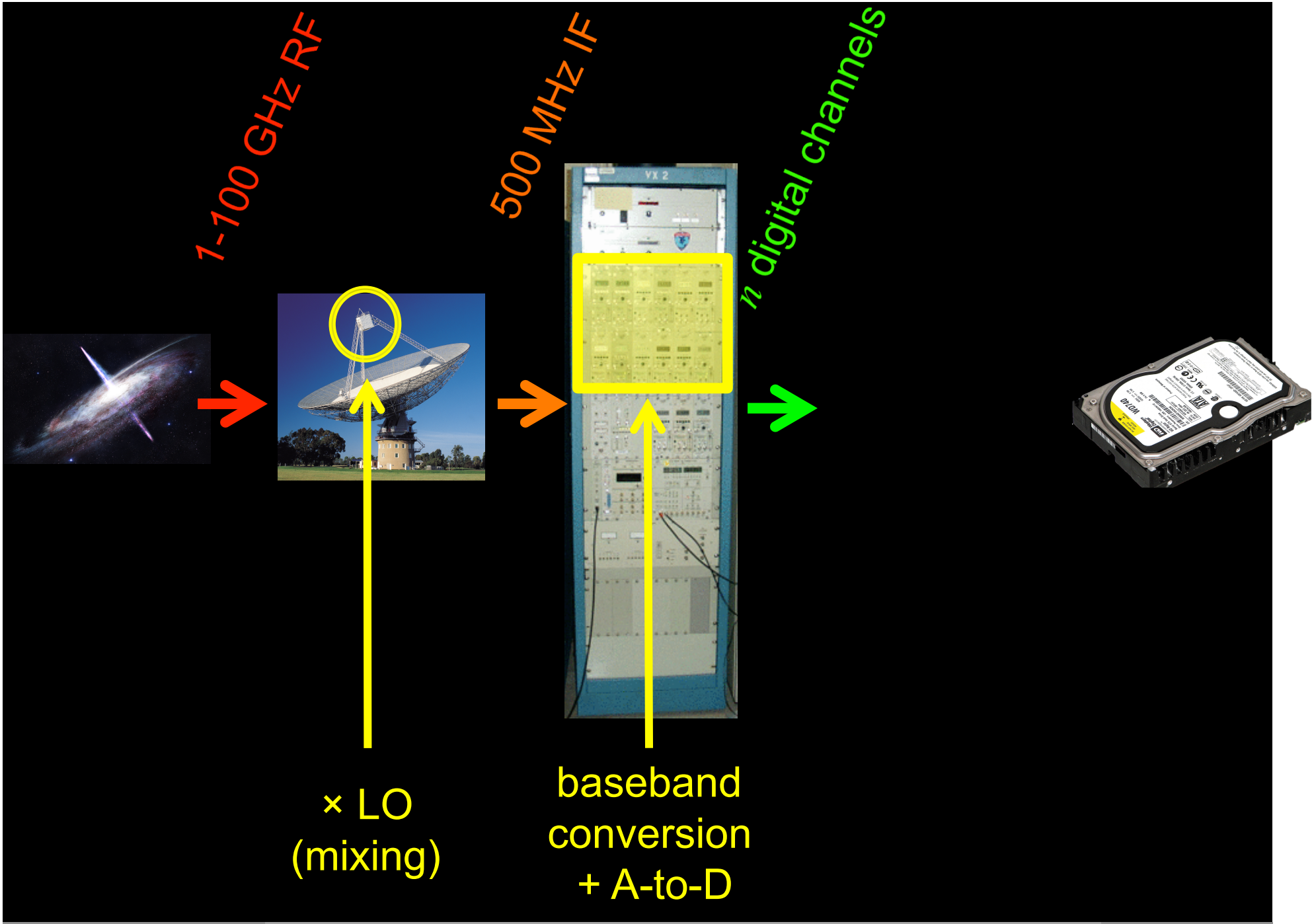
11 01 00 10 ... ← 2-bit numbers



1	1
1	0
0	1
0	0

sign bit

magnitude bit


1-100 GHz RF
500 MHz IF
n digital channels
× LO
(mixing)
baseband
conversion
+ A-to-D

Can't just write to disk

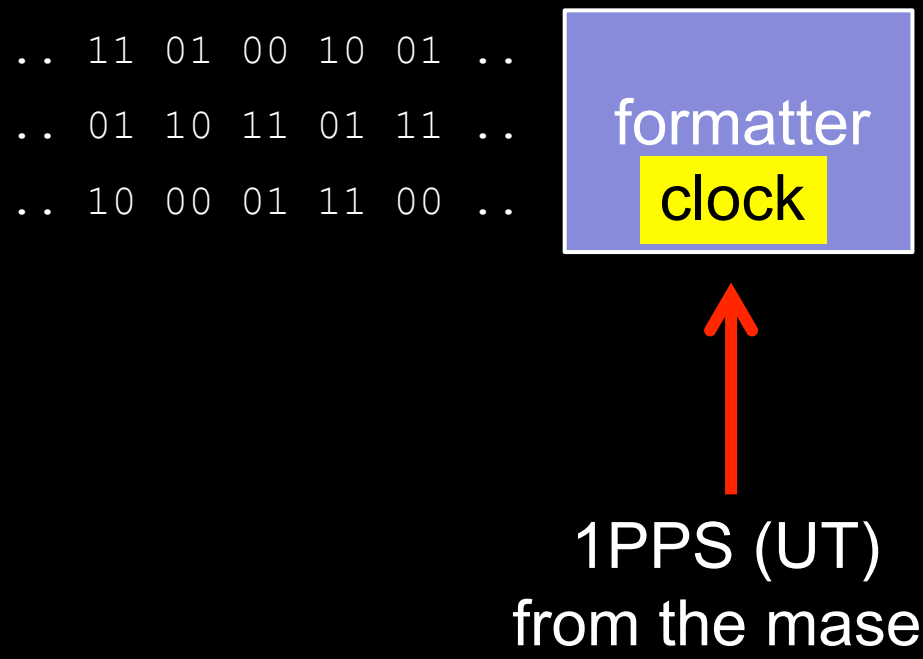
CH A: ... 11 01 00 10 01 ...

CH B: ... 01 10 11 01 11 ...

CH *z*: ... 10 00 01 11 00 ...

Enter the/a* formatter

Function 1: time tagging the samples

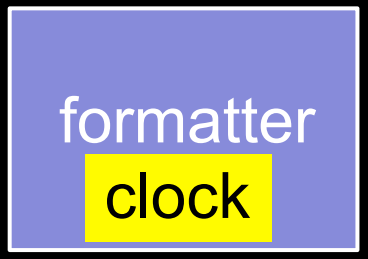


(*) there are multiple types of formatters

Enter the/a* formatter

Function 2: channel selection + decimation

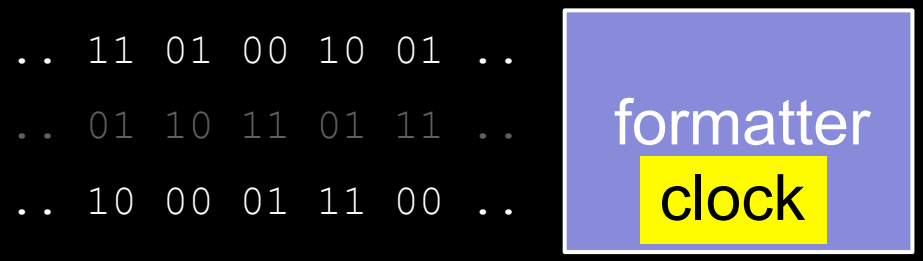
```
.. 11 01 00 10 01 ..
.. 01 10 11 01 11 ..
.. 10 00 01 11 00 ..
```



(*) there are multiple types of formatters

Enter the/a* formatter

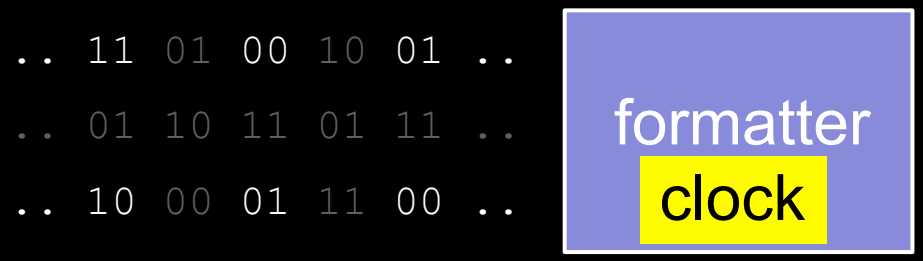
Function 2: channel selection + decimation



(*) there are multiple types of formatters

Enter the/a* formatter

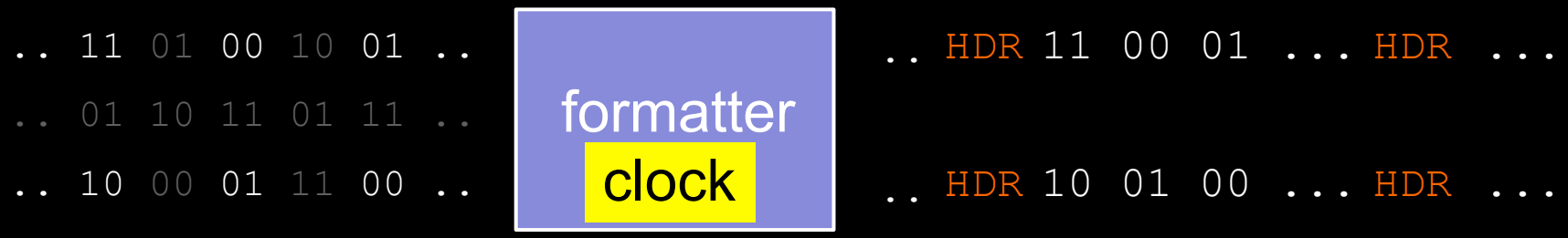
Function 2: channel selection + decimation



(*) there are multiple types of formatters

Enter the/a* formatter

Function 3: formatting into frames

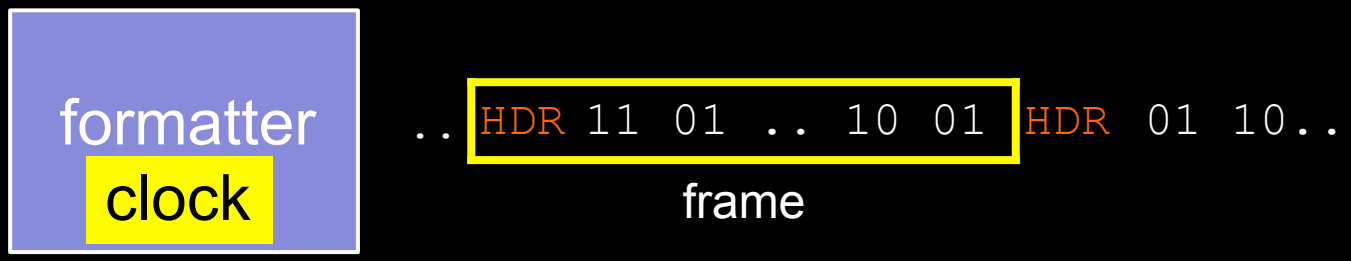


(*) there are multiple types of formatters

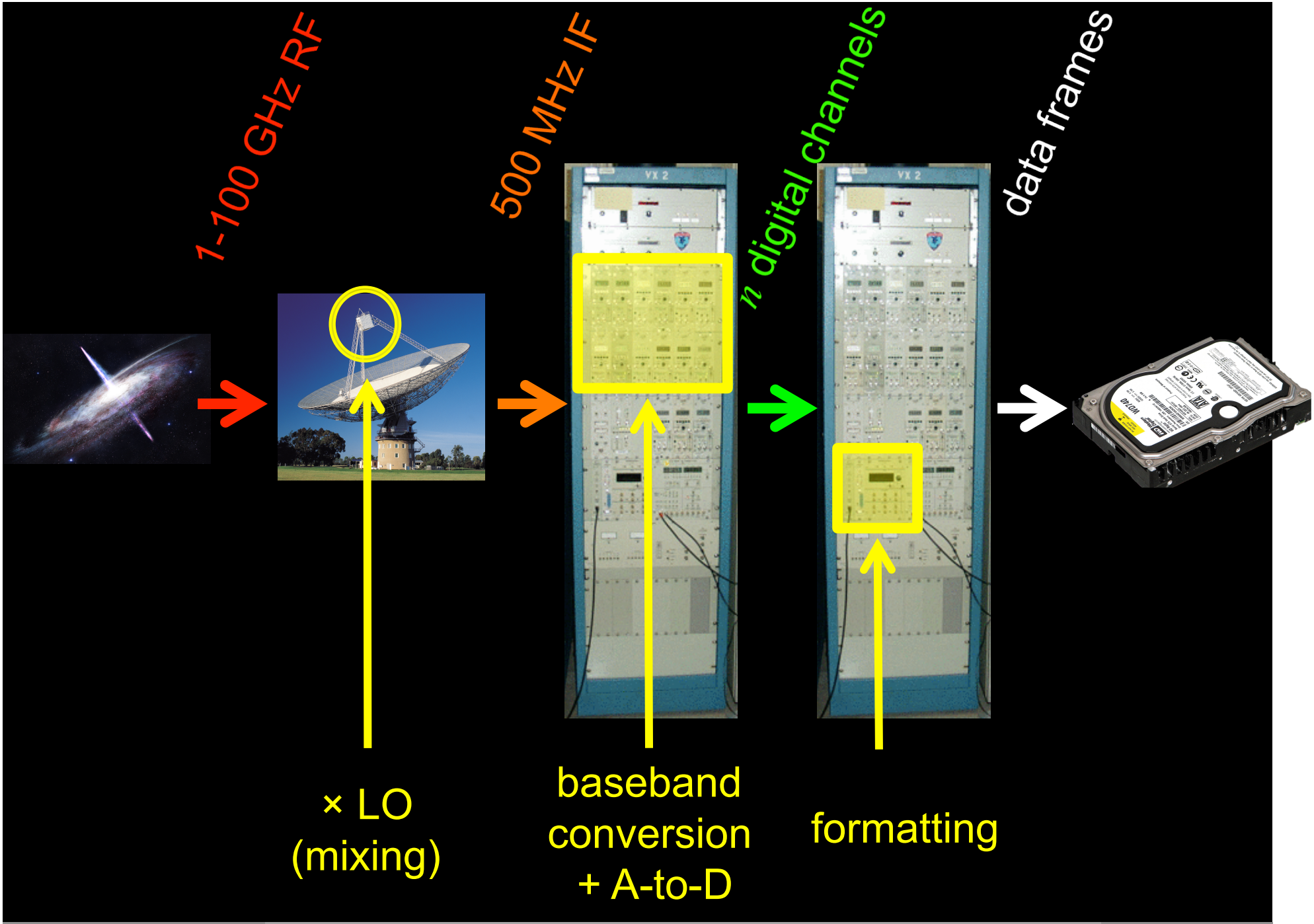
Enter the/a* formatter

Turn bit streams into sequence of frames

the type of formatter determines the frame/data format

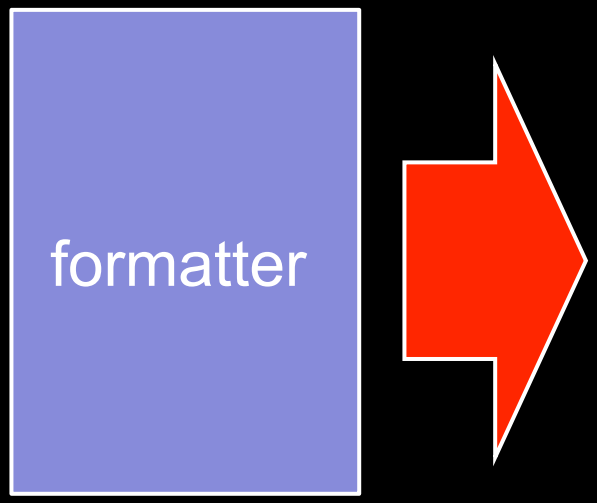


(*) there are multiple types of formatters


1-100 GHz RF
500 MHz IF
n digital channels
data frames
× LO
(mixing)
baseband
conversion
+ A-to-D
formatting

Can't just write to disk

Each type of BBC/formatter has its own connector type



- MarkIV/VLBA
  - RS422: 32 parallel serial wires, 1 bit / each
- Mark5B
  - V(LBI) S(tandard) I(nterface) / H(ardware) flatcable w/ 32 parallel lines, 1 bit / each
- FiLa10G/RDBE
  - 10Gbps ethernet

Can't just write to disk

Nyquist:

f_s = 2 x BW

Thus:

BW = 16 MHz

f_s = 32 Msamples / s

At 2 bits / sample:

64 Mb / s (per channel)

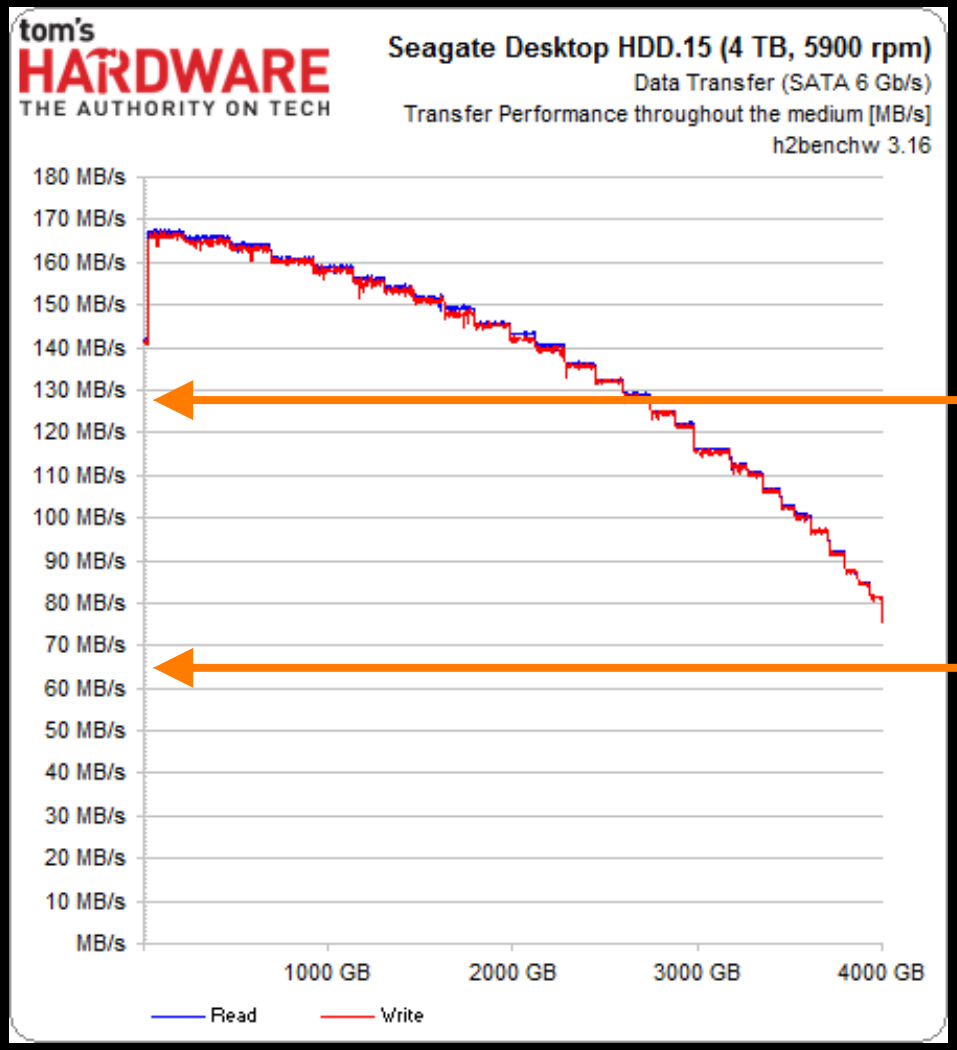
Can't just write to disk

Typical VLBI observation:

16 channels ×16 MHz

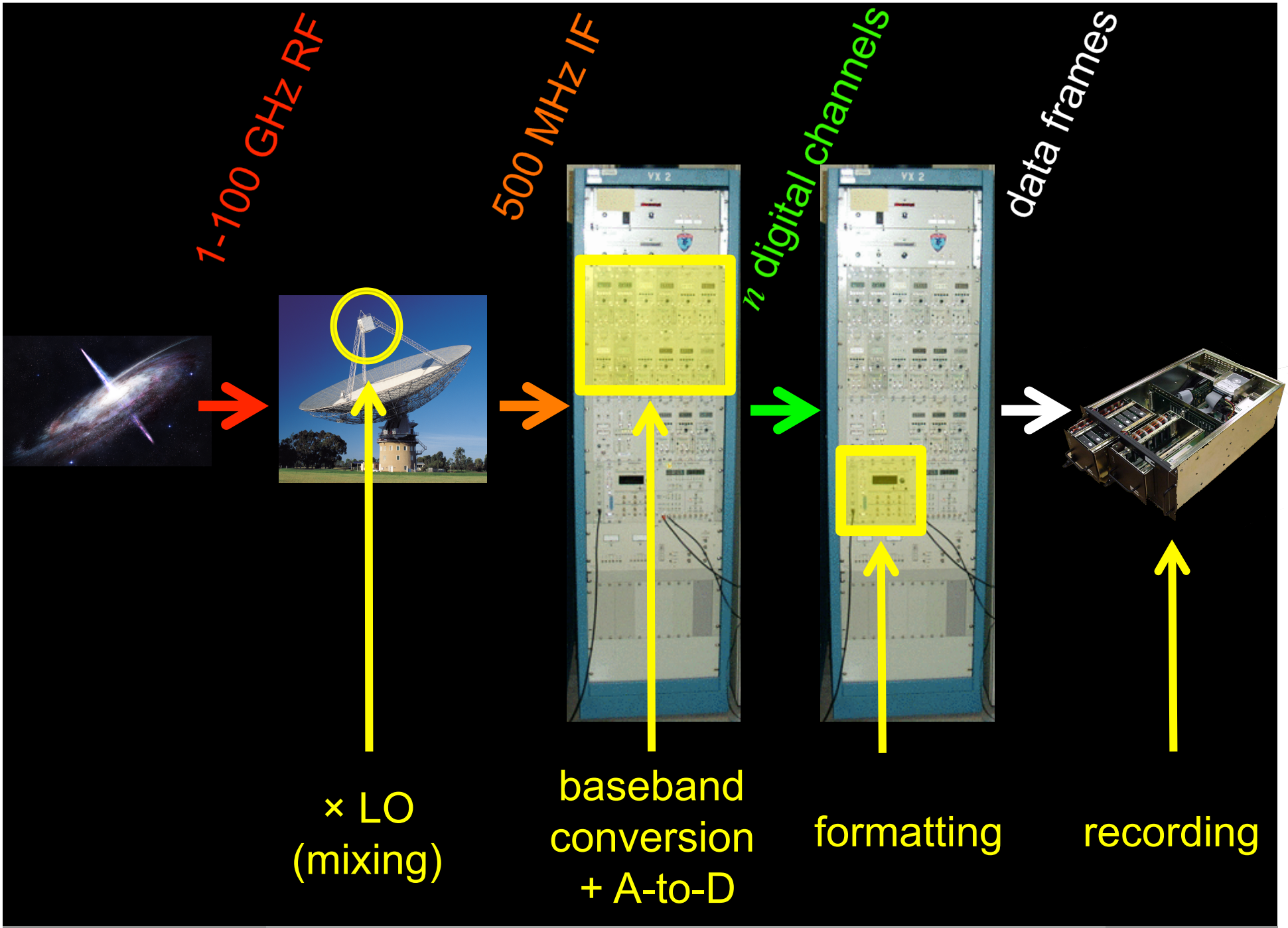
Thus:

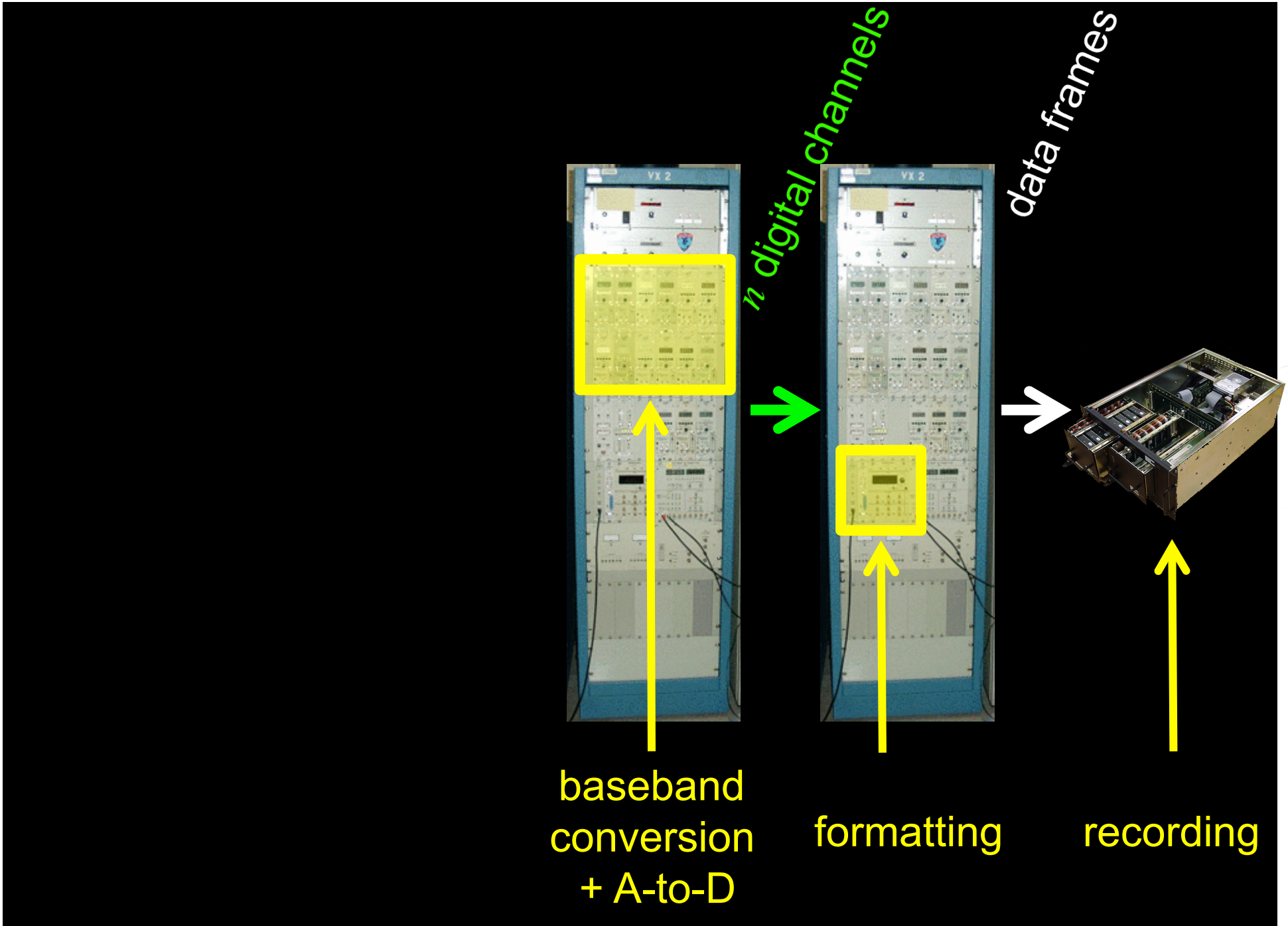
16 × 64 Mbps = 1024 Mbps (128 MB/s)



1024Mbps

512Mbps


1-100 GHz RF
500 MHz IF
n digital channels
data frames
× LO
(mixing)
baseband
conversion
+ A-to-D
formatting
recording


n digital channels
data frames
baseband conversion + A-to-D
formatting
recording


BBC
formatter
recorder

Hardware zoo!

Expensive equipment:

- speed requirements
- long develoment time
- specialistic equipment

Keep it operating until it breaks:

- life time cycle ≥ 10 years
- very rapid development last decade:
  - ‘everyone can do it’

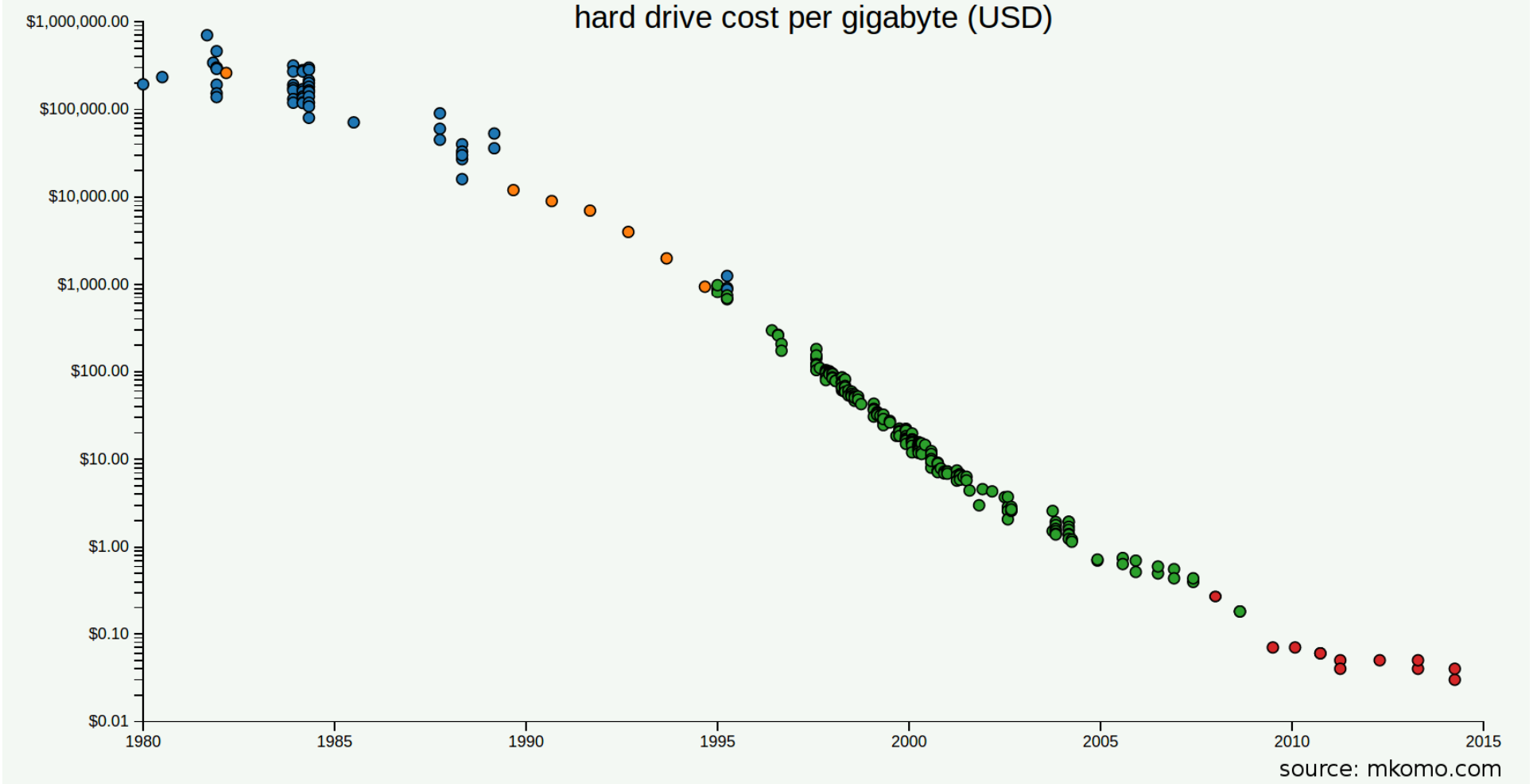
Hardware zoo!

Combinations of hardware:

- depend on connection types
- determine the data format on disk

Decommisic

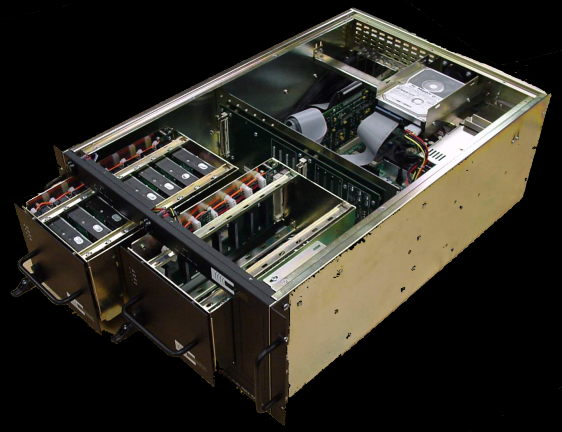
MarkIV/VLBA terminal


hard drive cost per gigabyte (USD)
$1,000,000.00
$100,000.00
$10,000.00
$1,000.00
$100.00
$10.00
$1.00
$0.10
$0.01
1980
1985
1990
1995
2000
2005
2010
2015
source: mkomo.com

Decomissioned equipment

MarkIV/VLBA terminal

RS422



Mark5A hard disk recorder
(around 2003)

Old/current e

DBBC[12]

ADS[123]000

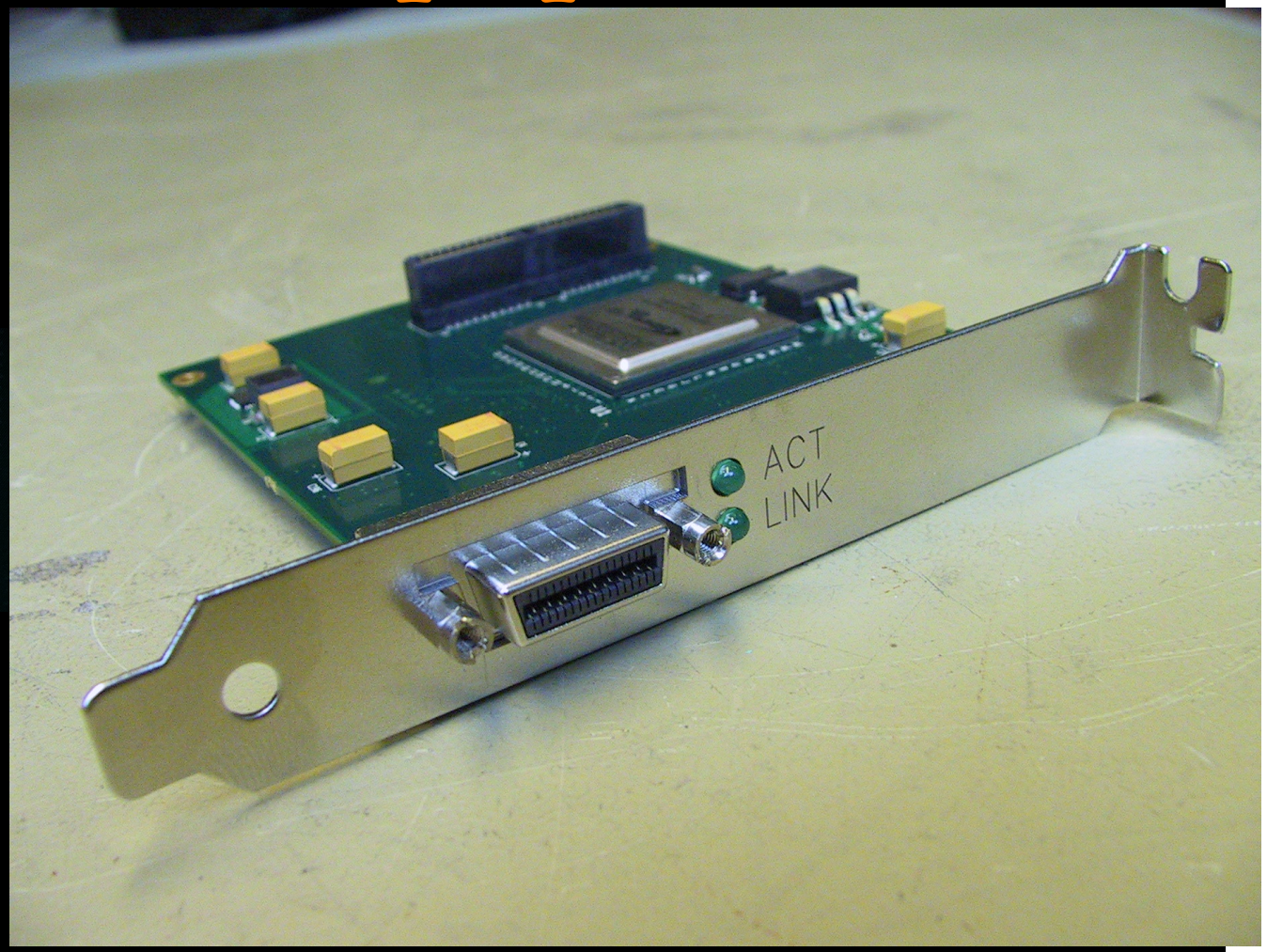
CDAS

Current/new equipment

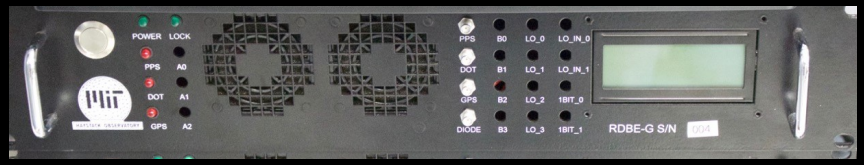
DBBC[12]

⋮

VSI/H



Current/new equipment



Roach Digital Backend (RDBE)

Mark5C

Mark6

FlexBuff

ethernet

MarkIV/VLBA terminal		Mark5A
DBBC[12] (Europe) R1000 (Russia) CDAS[12] (China) ADS2000 (Japan)	Mark5B	
	FiLa10G	Mark5C FlexBuff Mark6
RDBE		Mark5C FlexBuff Mark6

RS422	MarkIV VLBA	**512 Mbps**
VSI/H	DBBC[2] R1002 CDAS ADS[123]000	**2048 Mbps**
Ethernet	DBBC[2]+FiLa10G R[2]DBE CDAS2 BRAS	**8192 Mbps**

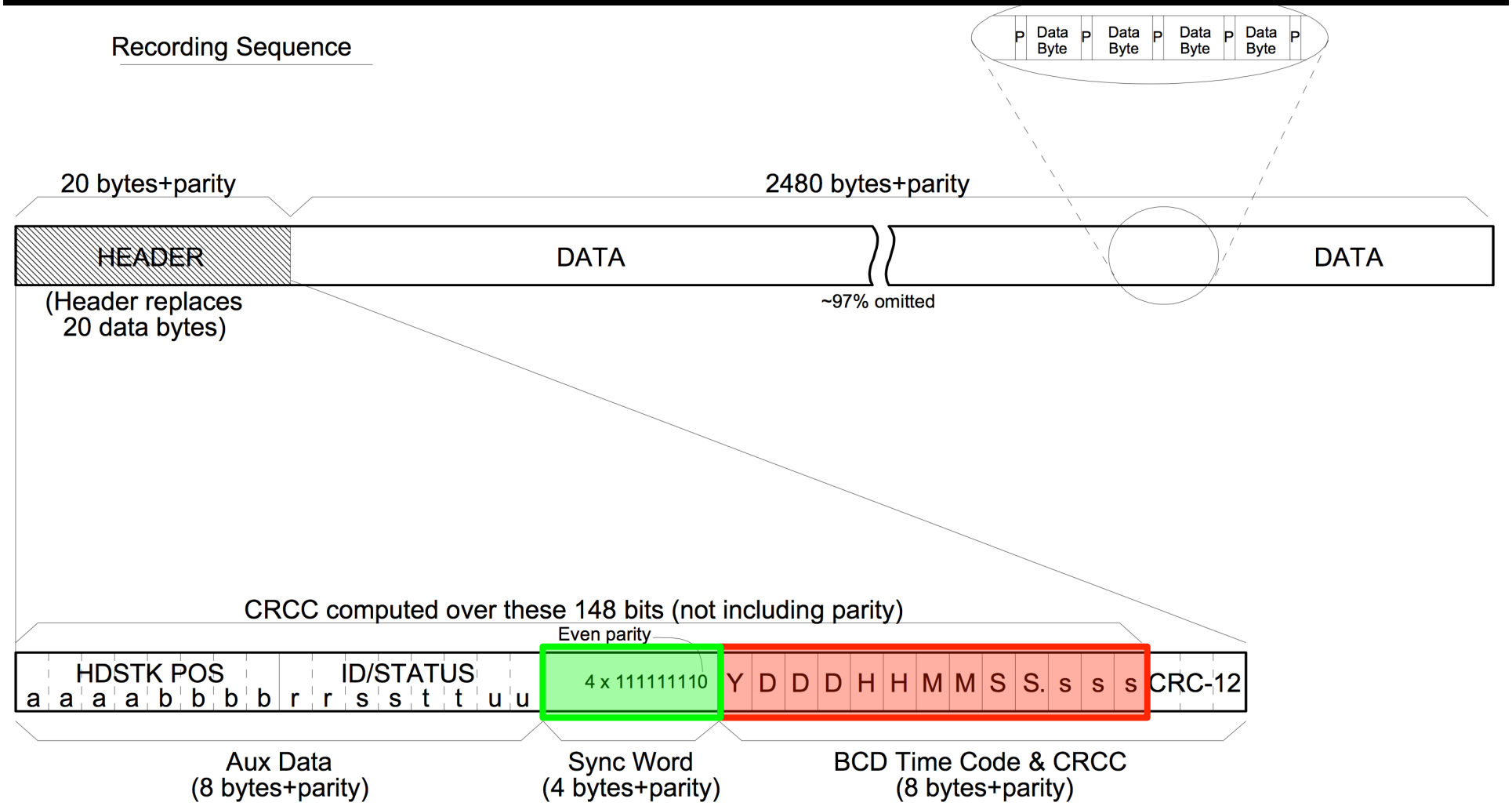
VLBI Data Formats

Different formatters

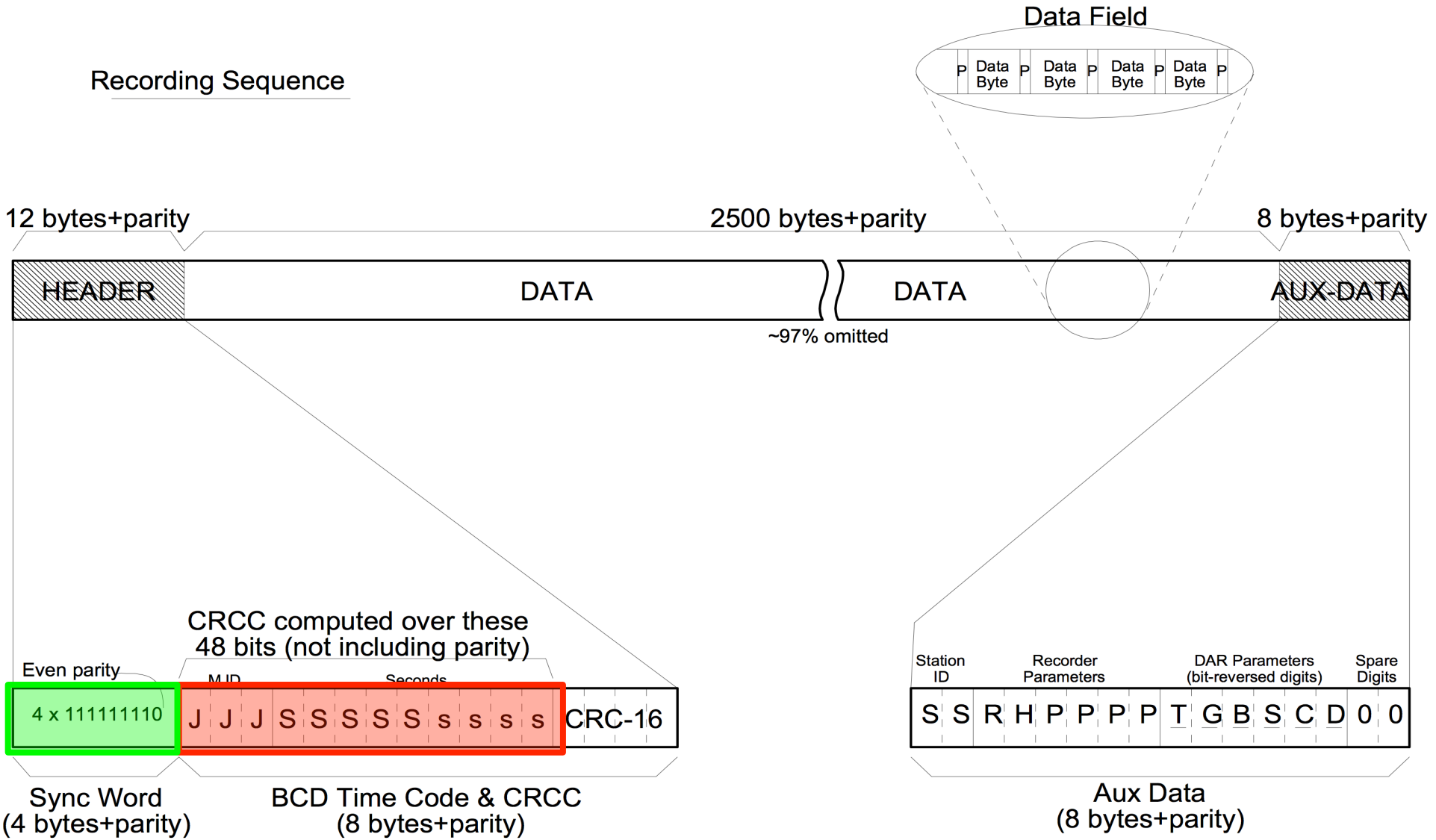
produce different data formats

formatter	format
MarkIV, VLBA	MarkIV, VLBA
Mark5B	Mark5B
FiLa10G	Mark5B, VDIF
RDBE	Mark5B, VDIF

MarkIV format (old, tape)



VLBA format (old, tape)



Differences:

- different time stamp encoding
- MarkIV: header overwrites data
  - 2500 bytes / frame
- VLBA: header inserted into data
  - 2520 bytes / frame

Similar:

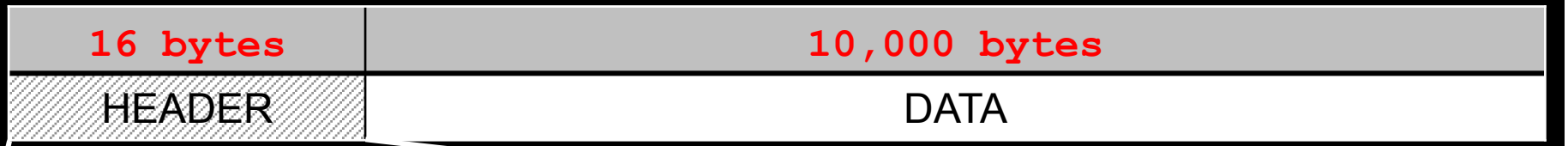
- synchronization word – easy to find in binary data
  - `0xffffffff 0xffffffff 0xffffffff 0xffffffff`
- layout of the data payload section
  - how sign, mag bits are ordered
  - non trivial ordering (see below)
- full header per bit stream
  - n recorded streams = n headers
- frame length depends on number of recorded streams:
  - n x 2500(2520) bytes [160,000 bytes / 64 tracks]

Bit stream	0	1	2	3	4	5	6	7	...
Channel	A	I	A	I	B	J	B	J	...
Type	Sign	Sign	Mag	Mag	Sign	Sign	Mag	Mag	...

Mark5B format (2006, disk)

Dedicated hard disk recorder:

- one header for all bit streams
- fixed frame length of 10,016 bytes:

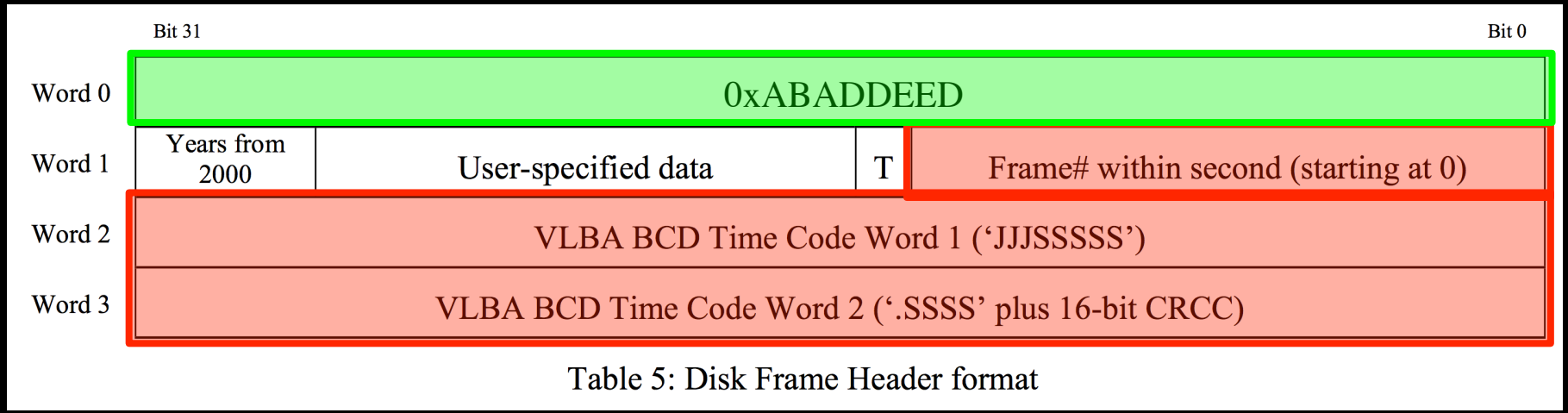


	Bit 31			Bit 0
Word 0	0xABADDEED			
Word 1	Years from 2000	User-specified data	T	Frame# within second (starting at 0)
Word 2	VLBA BCD Time Code Word 1 ('JJJSSSSS')			
Word 3	VLBA BCD Time Code Word 2 ('.SSSS' plus 16-bit CRCC)			

Table 5: Disk Frame Header format

Mark5B format (2006, disk)

Header written by Mark5B formatter



	Bit 31			Bit 0
Word 0	0xABADDEED			
Word 1	Years from 2000	User-specified data	T	Frame# within second (starting at 0)
Word 2	VLBA BCD Time Code Word 1 (‘JJJSSSSS’)			
Word 3	VLBA BCD Time Code Word 2 (‘.SSSS’ plus 16-bit CRCC)			

Table 5: Disk Frame Header format

Mark5B format (2006, disk)

Header written by FiLa10G and RDBE formatters:
requires data reader to *know* #-of-frames-per-second

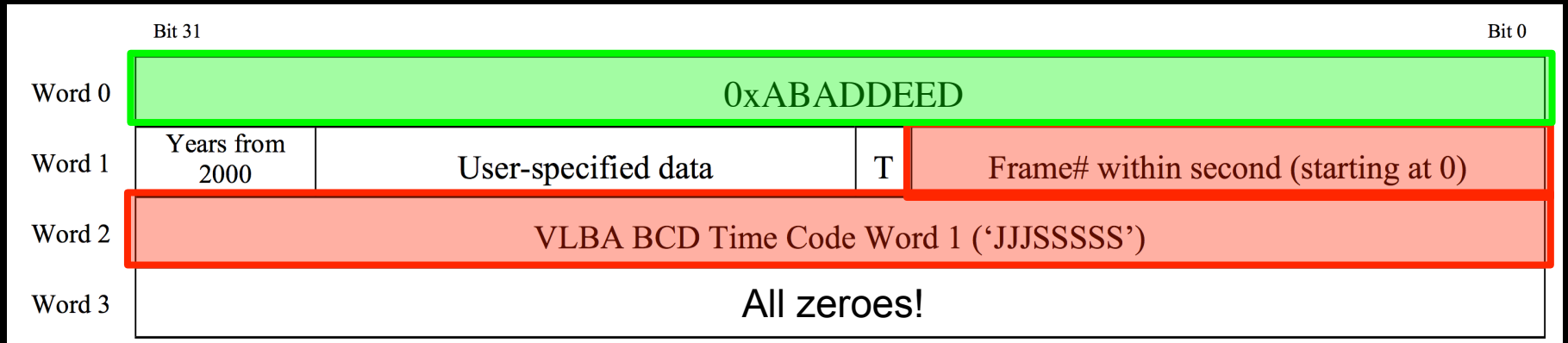
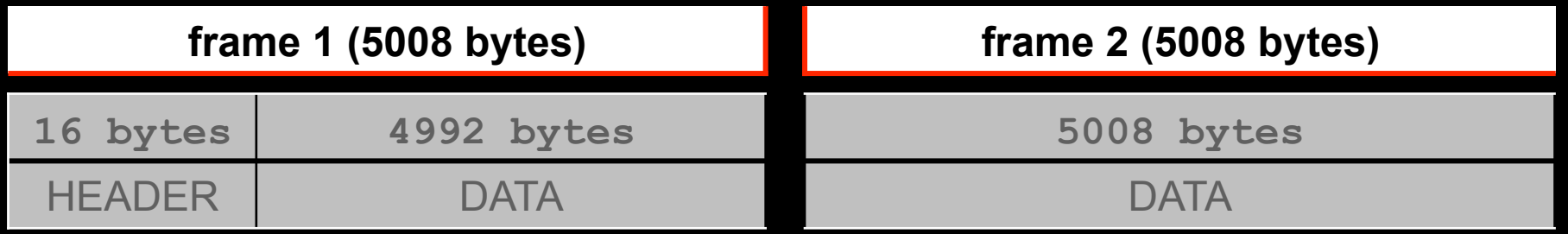


Table 5: Disk Frame Header format

Mark5B format (2006, disk)

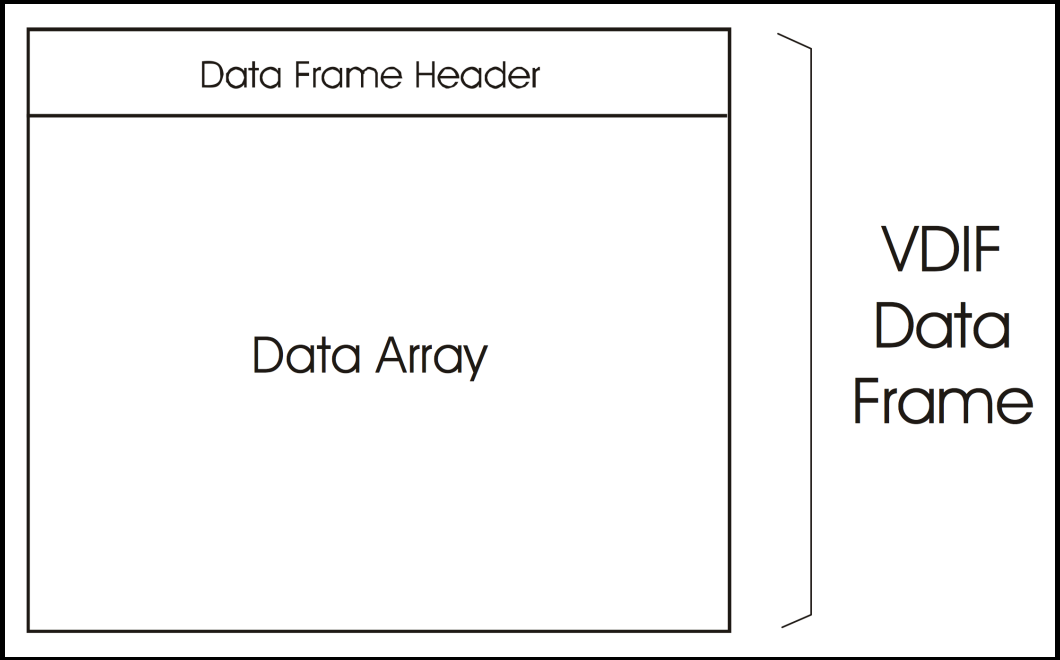
Drawbacks of Mark5B format

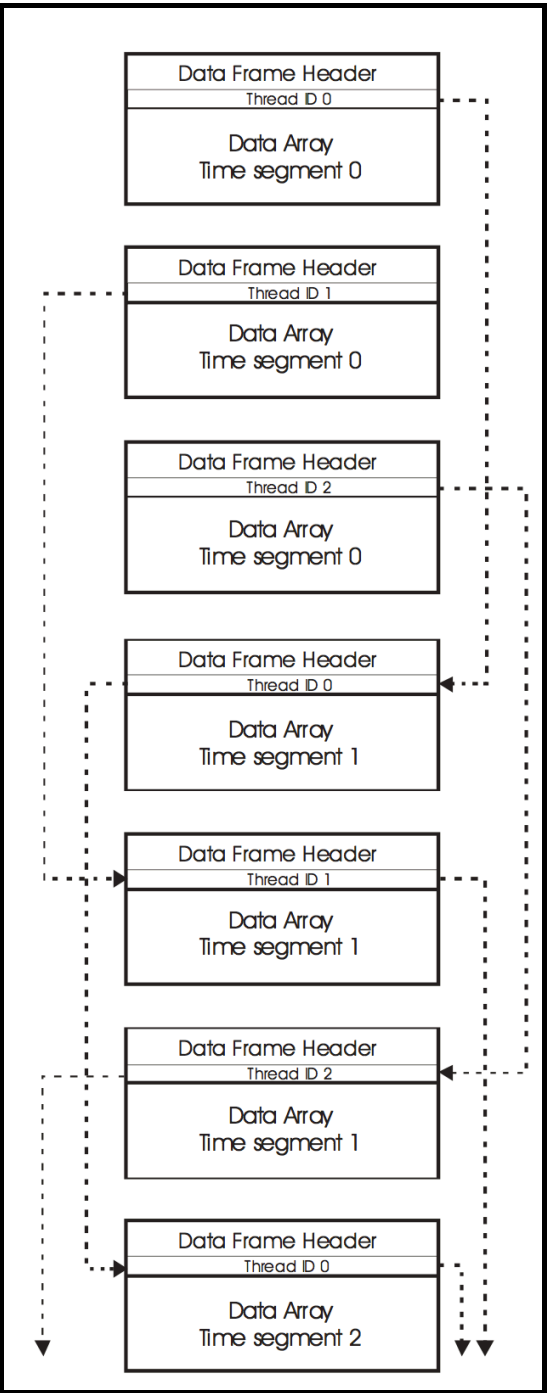
- only one stream / device
- number of channels / stream *must* be 2^n
- placement of sign, magnitude bits *not* fixed!
- maximum 2048 Mbps
- time stamp is only valid for 1000 days (< 3 years!)
- frame size fixed at 10,016 bytes
  - ethernet transport more important / new recorders
  - maximum ethernet frame 9000 bytes!



VDIF (2009)

V(LBI) D(ata) I(nterchange) F(ormat)





VDIF (2009)

32 byte header:

The standard 32-byte VDIF Data Frame Header is shown in Figure 3.

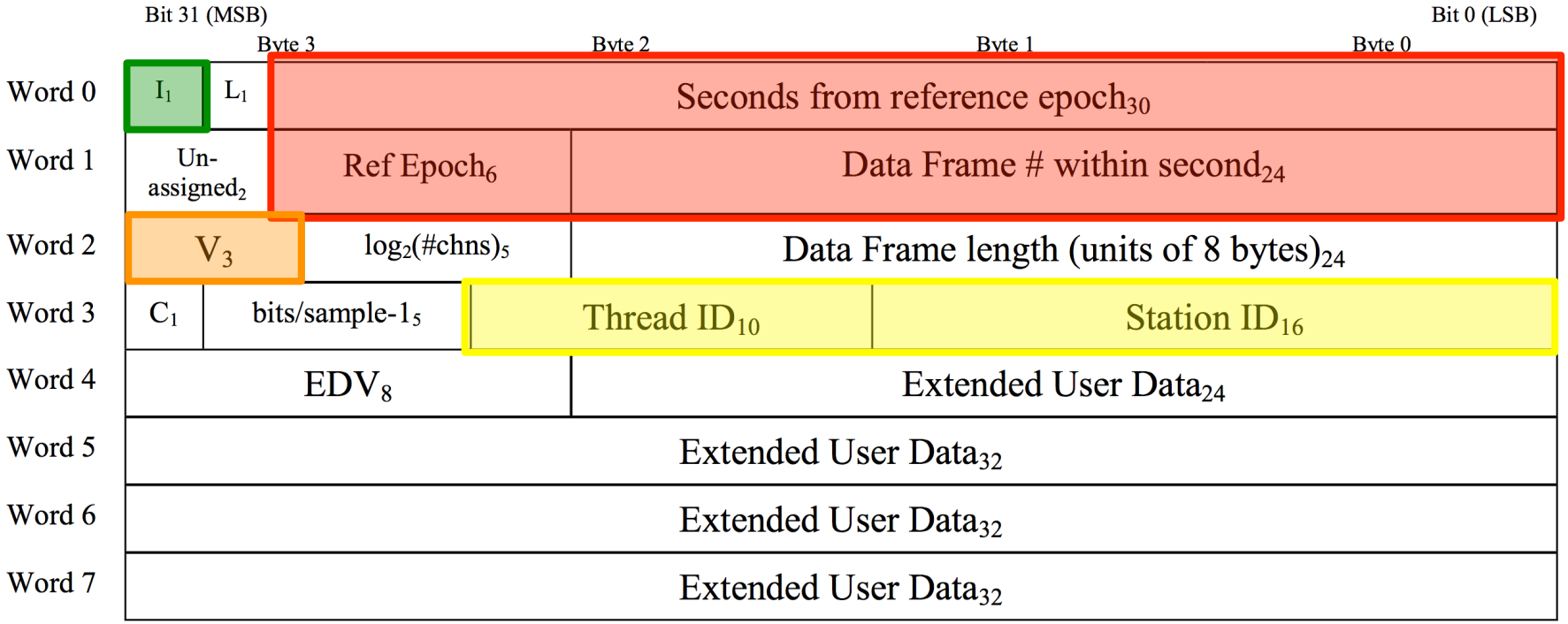
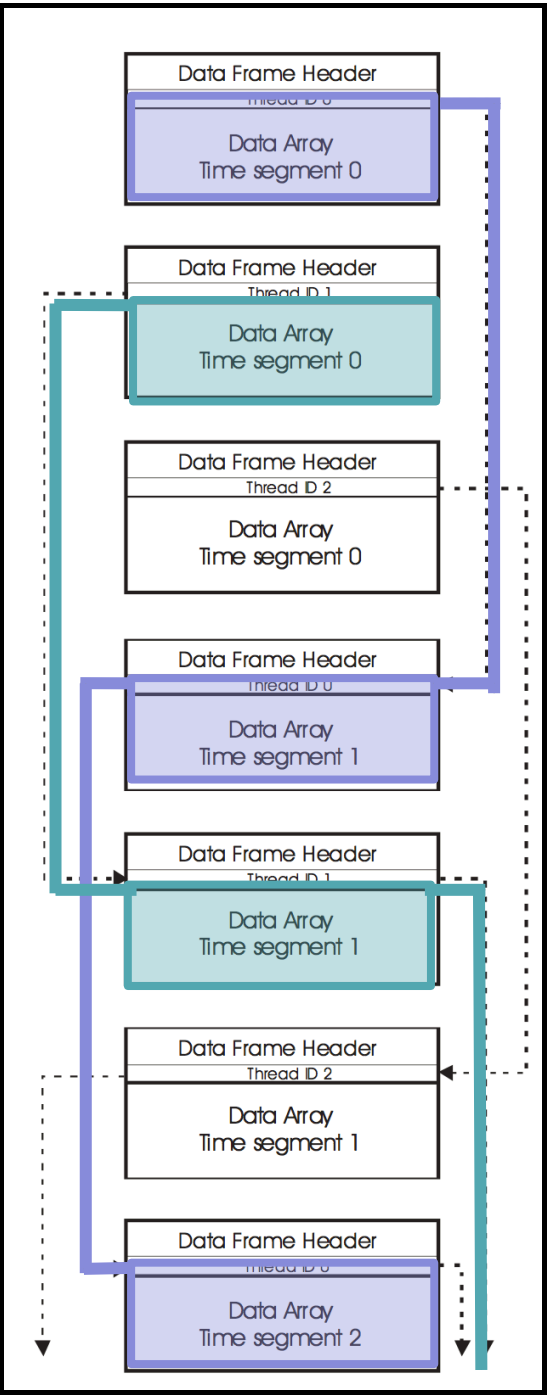


Figure 3: VDIF Data Frame Header format; subscripts are field lengths in bits; byte #s indicate relative byte address within 32-bit word (little endian format)

VDIF (2009)

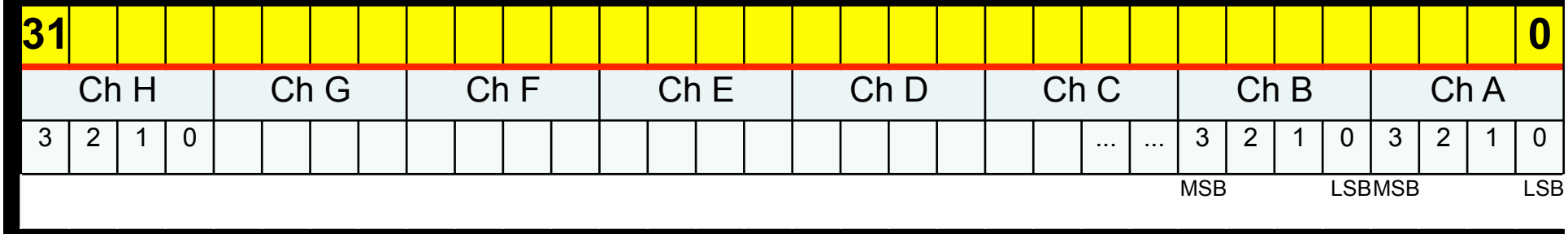
V(LBI) D(ata) I(nterchange) F(ormat)



VDIF (2009)

Data array layout fixed:

- order of channels is 0 .. (n-1)
- order of bits within sample is little endian



8 channels @ 4 bits / sample

VDIF (2009)

Lots of improvements over old(er) formats!

- layout of data array + order of bits is **standardized**
- time stamp accuracy + construction:
  - can observe across leap seconds
  - accomodate very high data rates (128 Gbps easily)
  - can observe for ~33 years with one reference epoch
- data array size encoded in header

But ..... THERE IS NO SYNCWORD!

cannot recogniz[illegible] blurb

This is a serious defect

A canonical format

Single string format description

- human + machine readable
- captures most vital parts of the recording

<fmt>-<rate>-<#ch>-<#bps>

<fmt>	format	MKIV, VLBA MARK5B VDIF_<size>, VDIFL_<size> <size> is VDIF data array size
<rate>	total *data* rate	in Mbps (×10^6 bps)
<#ch>	number of channels	typically 2^n, n=0, 1, 2, ...
<#bps>	bits per sample	1, 2, 4, ...

<fmt>-<rate>-<#ch>-<#bps>

vlba-256-16-1

256Mbps VLBA, 16 channels of 8MHz

mark5b-2048-16-2

2Gbps Mark5B, 16 channels of 32MHz

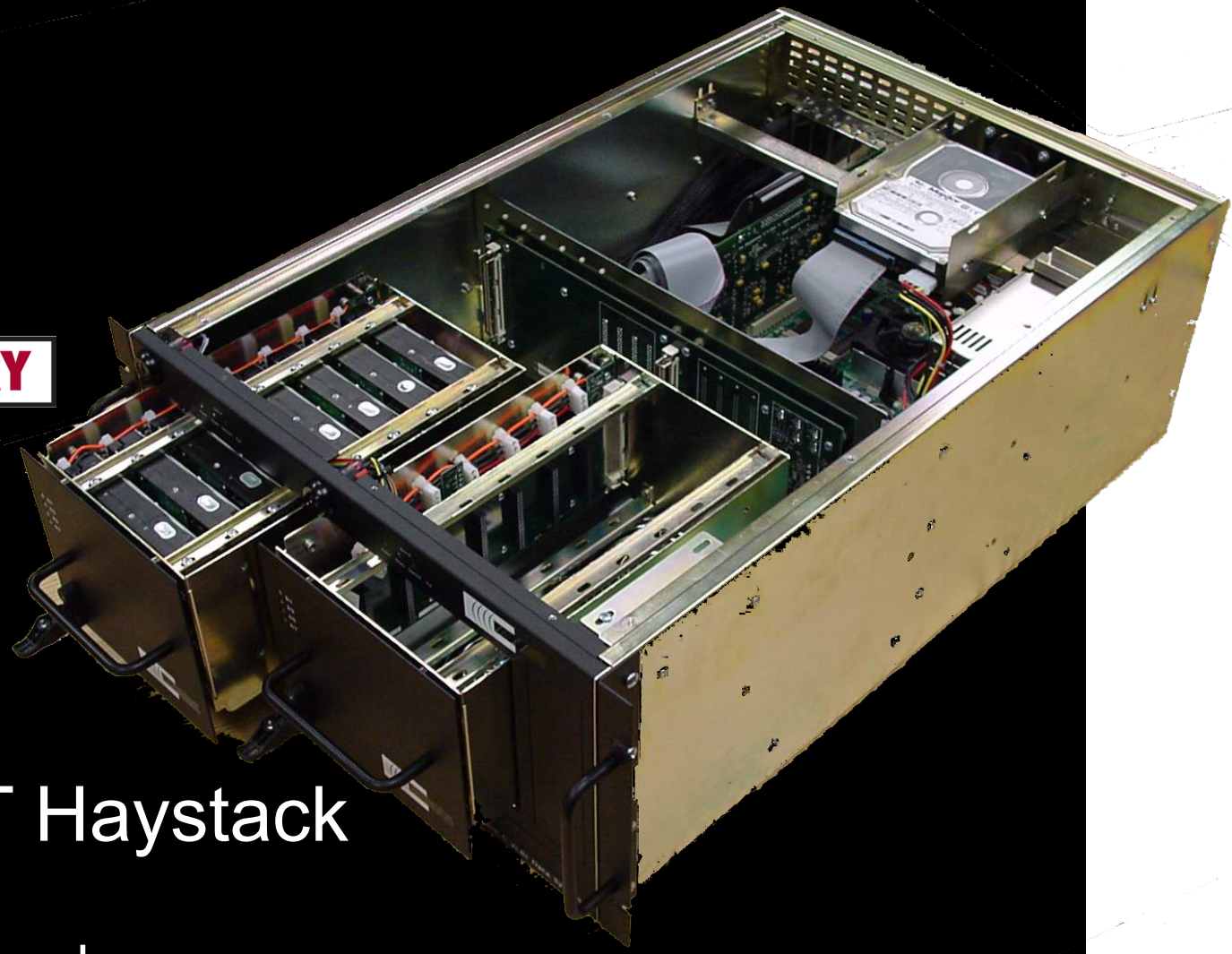
vdif_8000-16384-4-2

16Gbps VDIF, 8000 byte frames, 4 channels of 1GHz

Recording hardware
Recording software
Recording
Transfer

The Mark5 VLBI Data Recorders

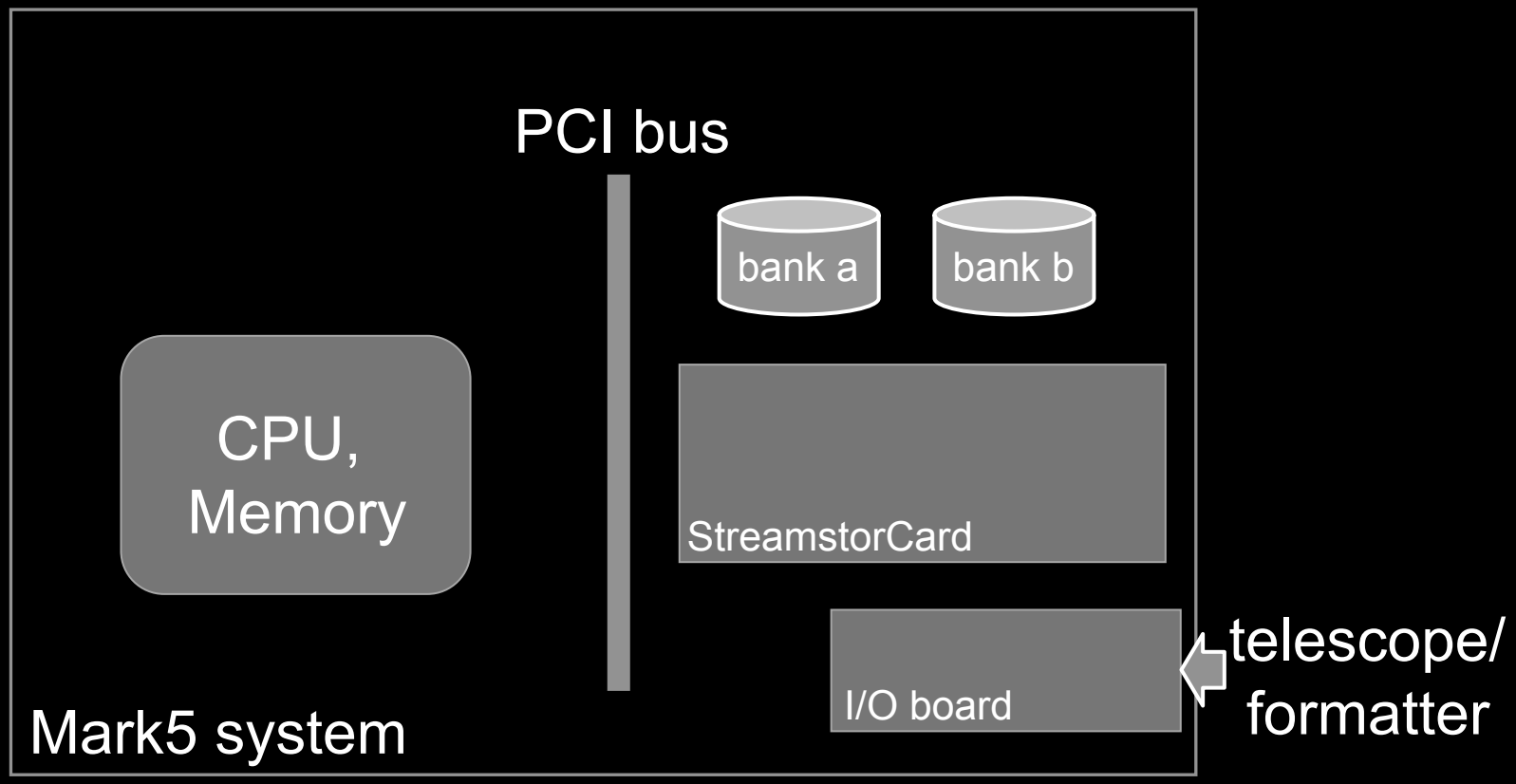
HAYSTACK OBSERVATORY



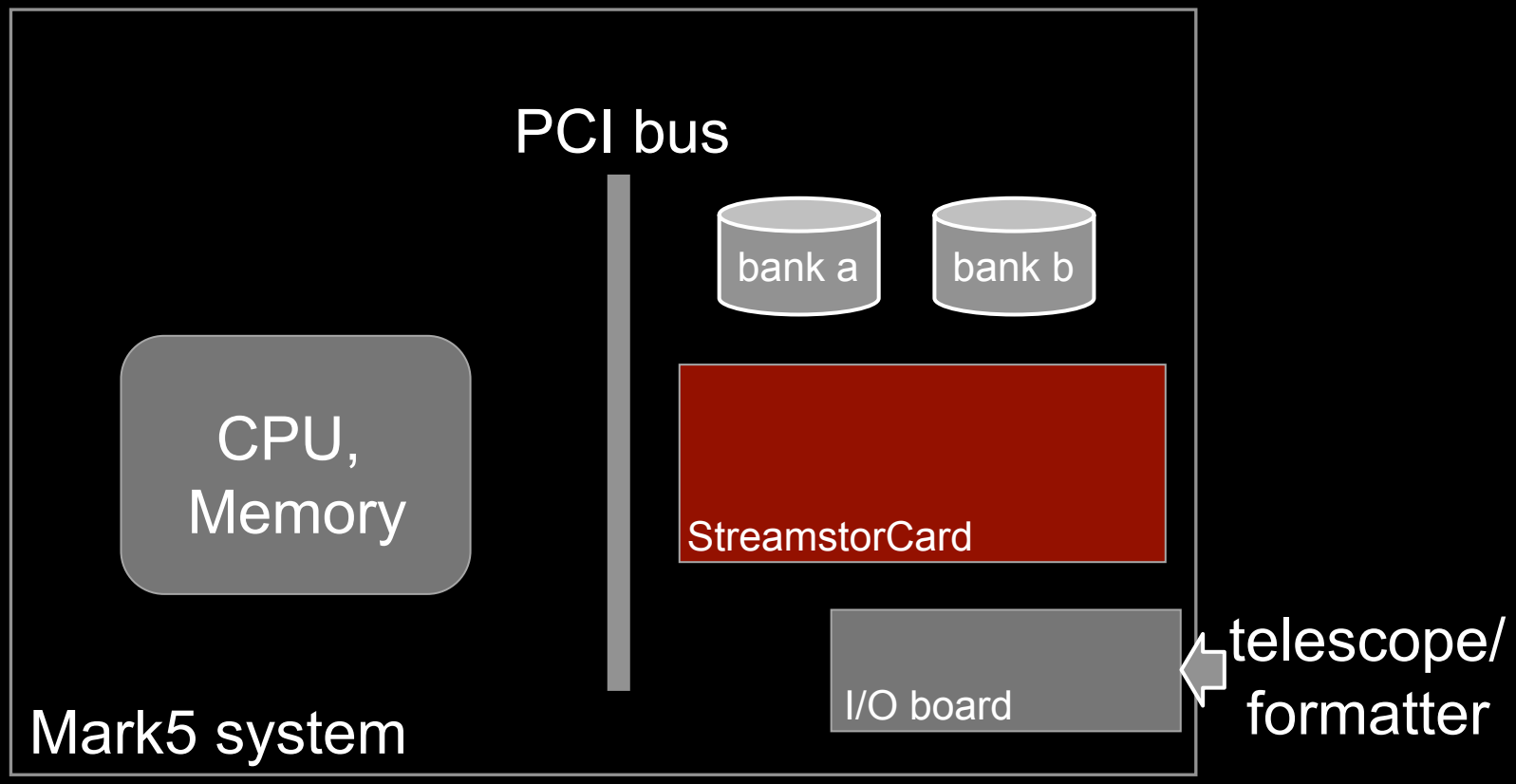
Developed by MIT Haystack

- since 2001
- use removable disk packs
- from 1024 Mbps to 4096 Mbps

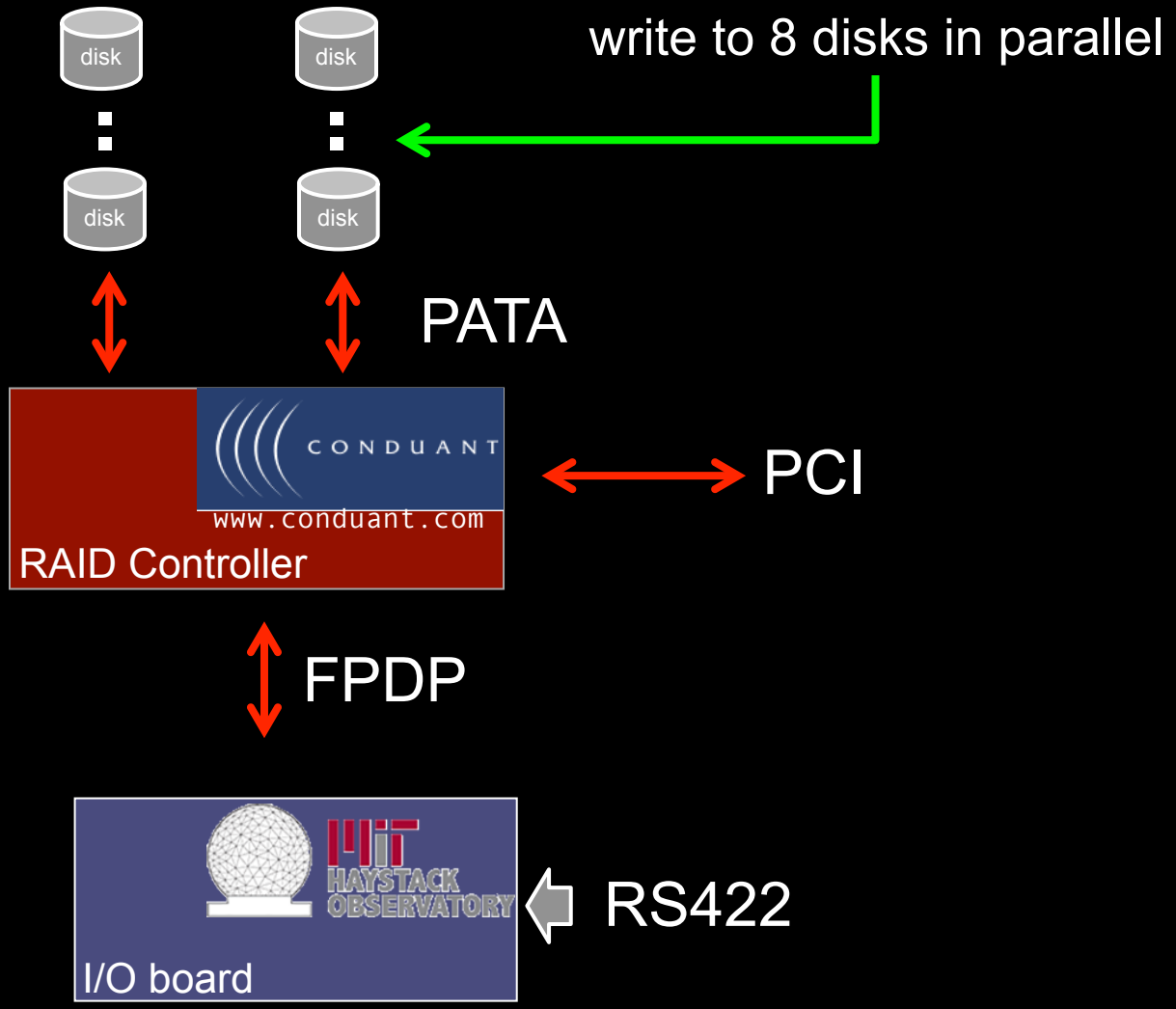
Inside the Mark5



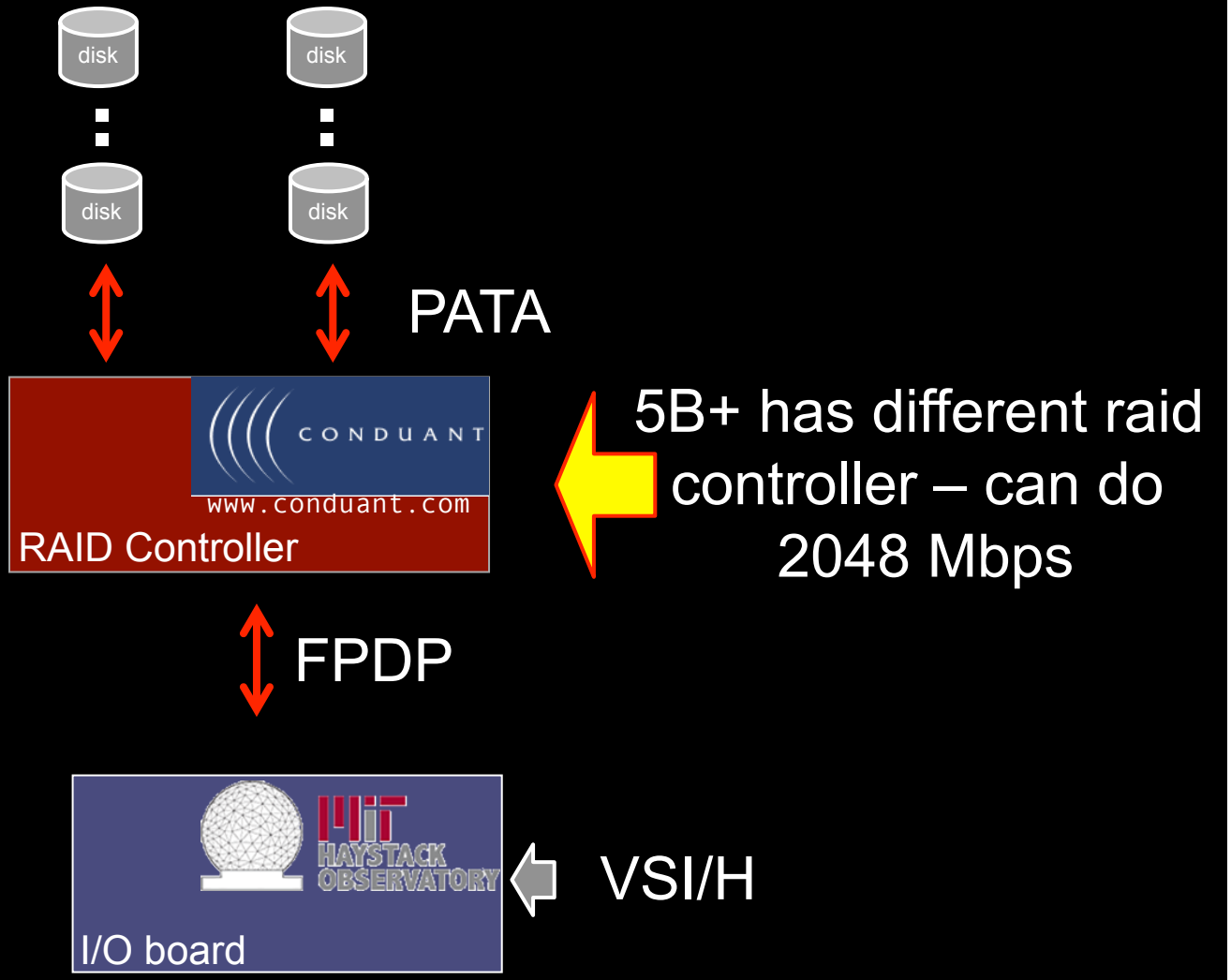
Inside the Mark5



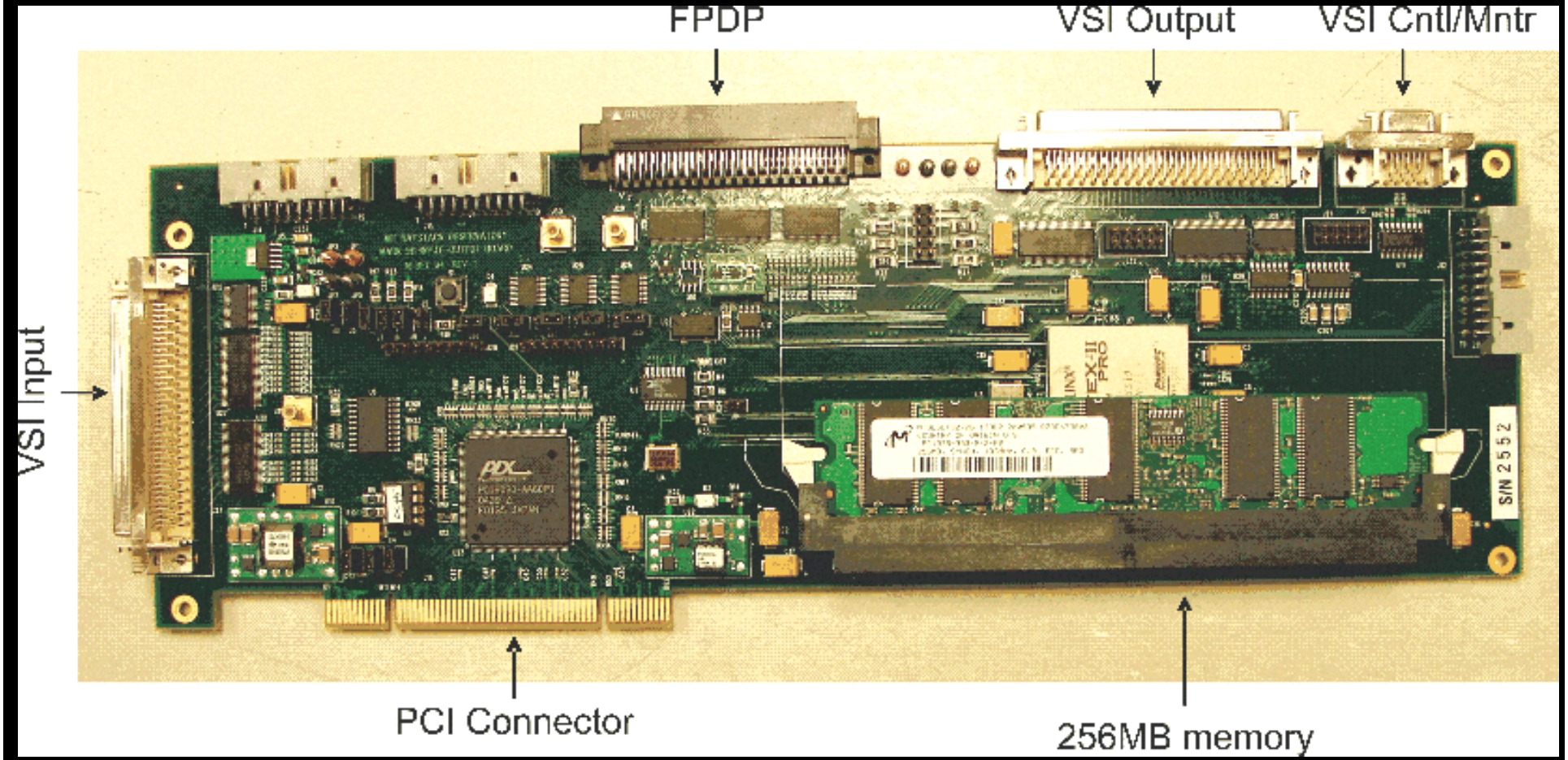
Inside the Mark5 (5A)



Inside the Mark5 (5B)



Inside the Mark5 (5B)



The MIT Haystack Mark5B I/O board

Mark6
and
FlexBuff

The ethernet packet recorders for 8- and 16 Gbps

Mark6

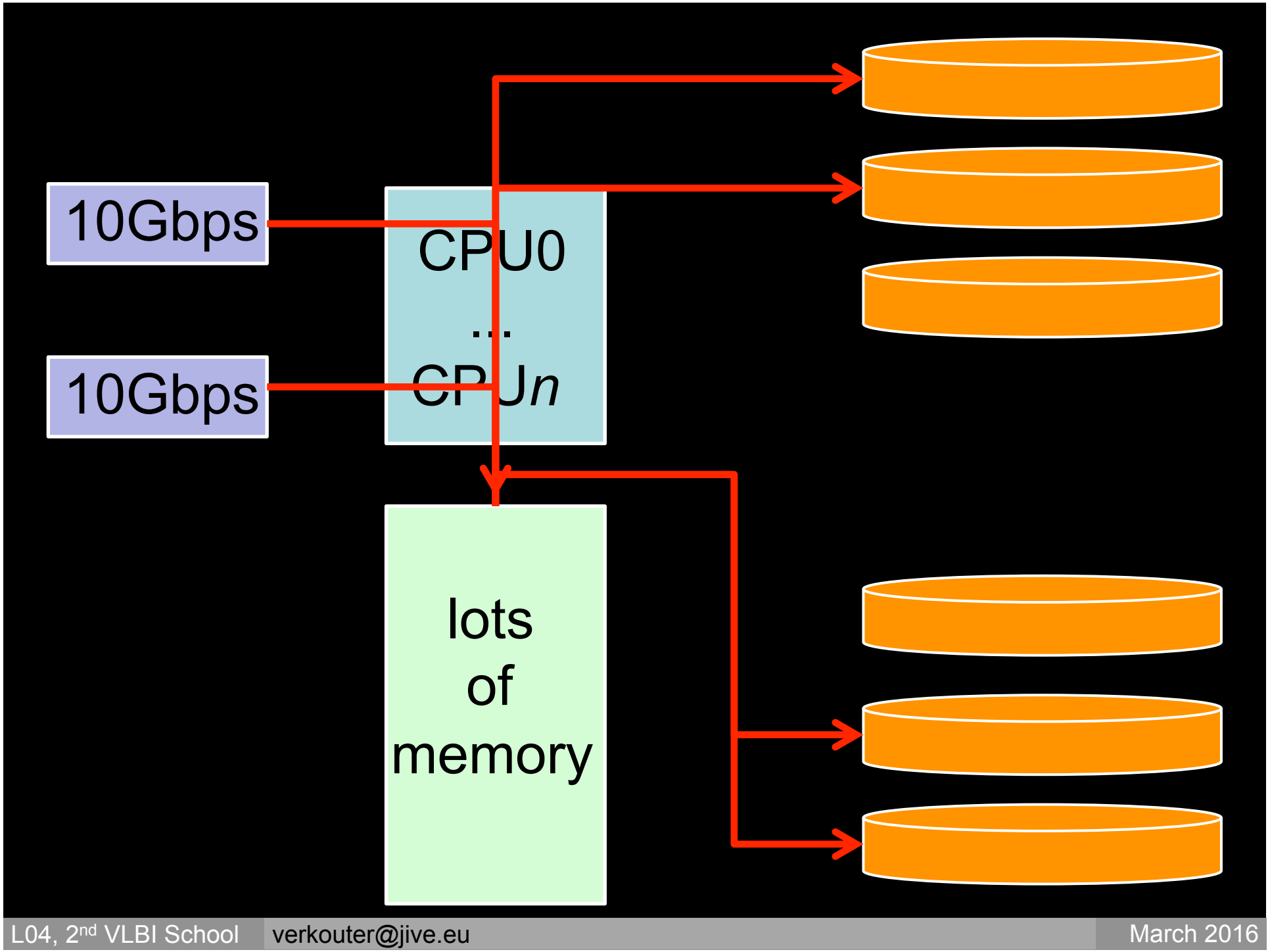
MIT Haystack – Chester Ruszczyk

FlexBuff

European VLBI Network – Metsähovi Observatory (FI)


1
2
Module Status
Present
Power
Ready
Fault
CONDUANT
Power
Mark6

FlexBuff


10Gbps
10Gbps
CPU0
...
CPU*n*
lots
of
memory
/mnt

FlexBuff

Concept developed in Finland, Metsahövi Observatory
Ari Mujunen, implemented by Tomi Salminen

- stripe data over lots of disks
- in regular files

FlexBuff

10Gbps

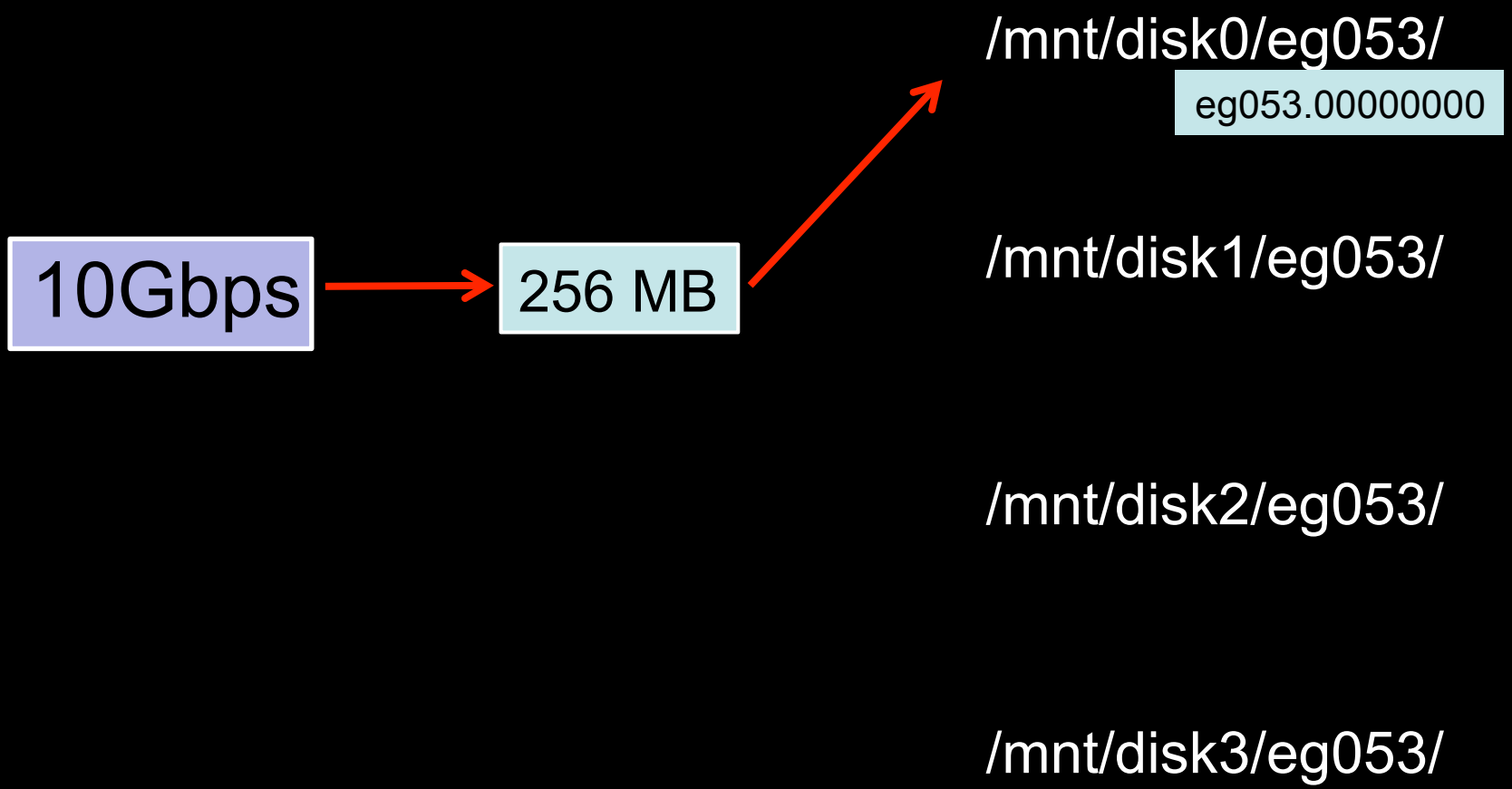
/mnt/disk0/eg053/

/mnt/disk1/eg053/

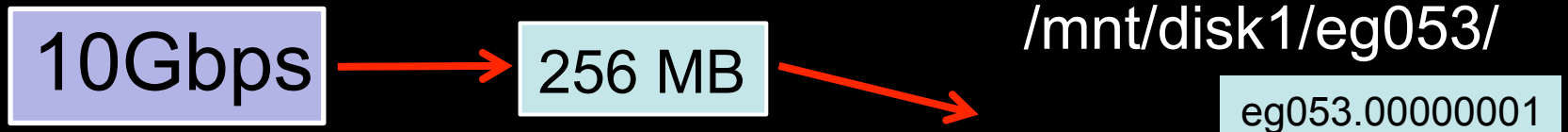
/mnt/disk2/eg053/

/mnt/disk3/eg053/

FlexBuff


/mnt/disk0/eg053/
eg053.00000000
10Gbps
256 MB
/mnt/disk1/eg053/
/mnt/disk2/eg053/
/mnt/disk3/eg053/
recording application header
VLBI data

FlexBuff



/mnt/disk0/eg053/

eg053.00000000

/mnt/disk1/eg053/

eg053.00000001

/mnt/disk2/eg053/

/mnt/disk3/eg053/

recording application header

VLBI data

FlexBuff

10Gbps

/mnt/disk0/eg053/

eg053.00000000

eg053.00000006

/mnt/disk1/eg053/

eg053.00000001

eg053.00000007

/mnt/disk2/eg053/

eg053.00000002

eg053.00000005

/mnt/disk3/eg053/

eg053.00000003

eg053.00000004

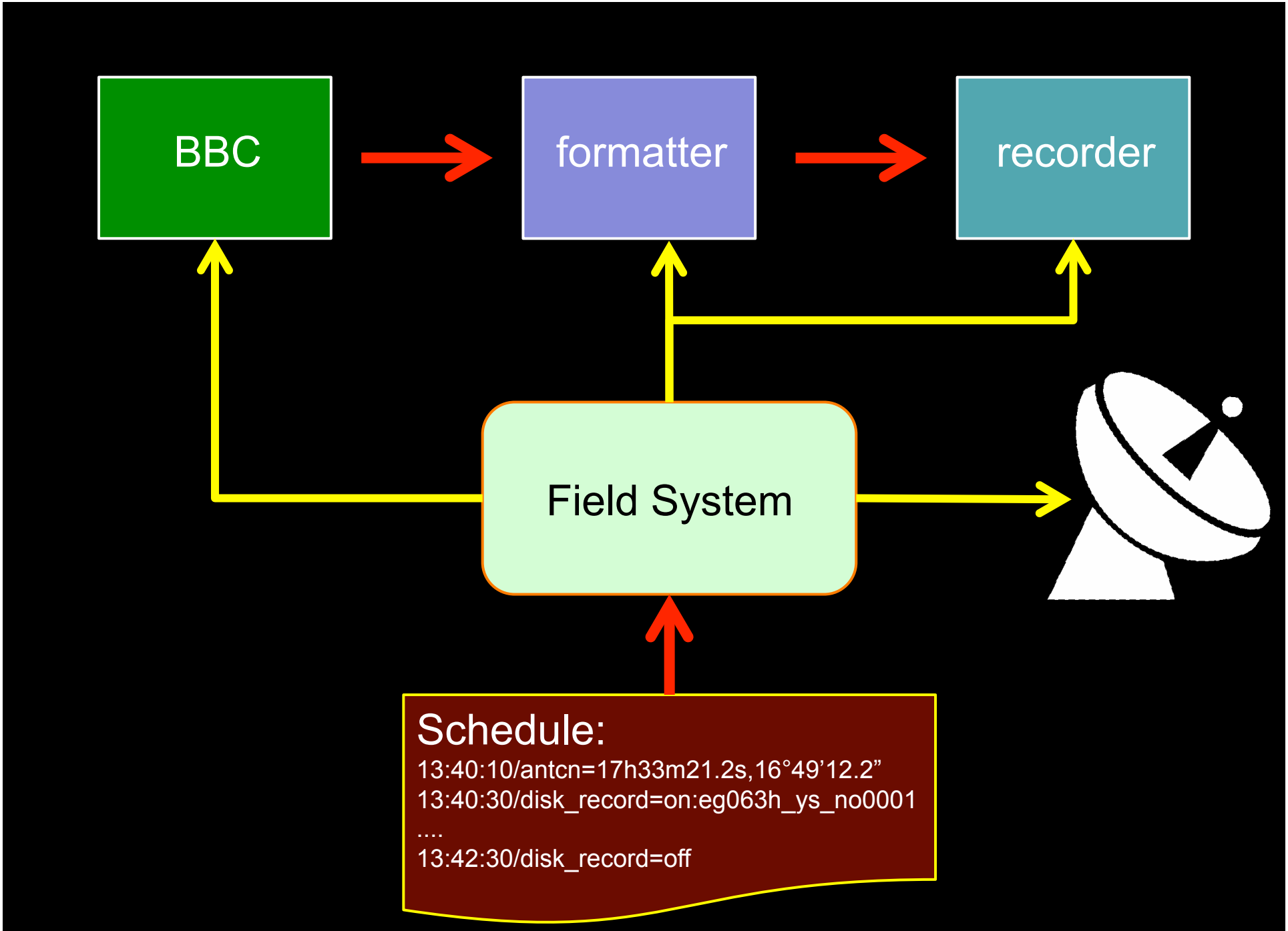
recording application header

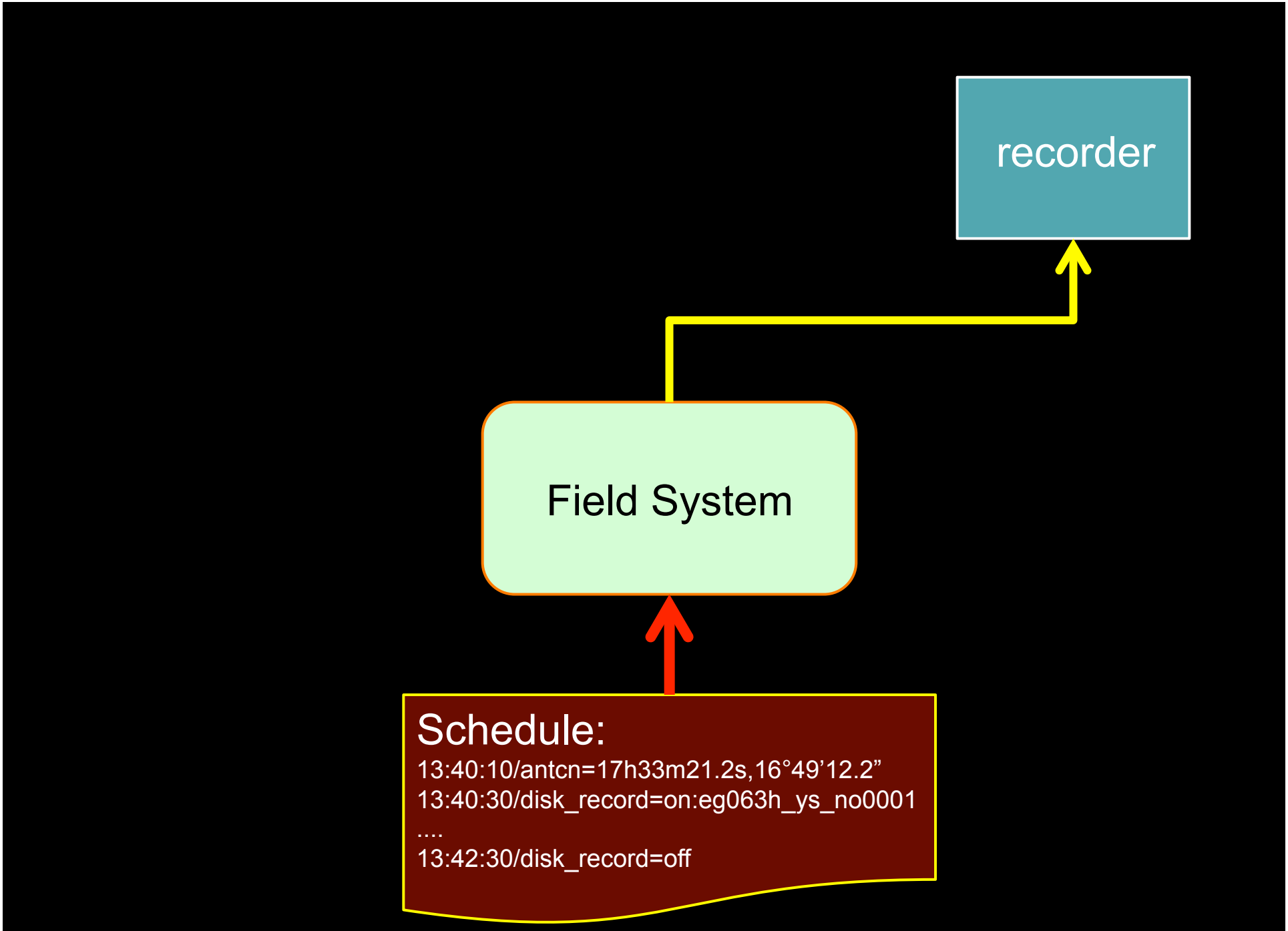
VLBI data

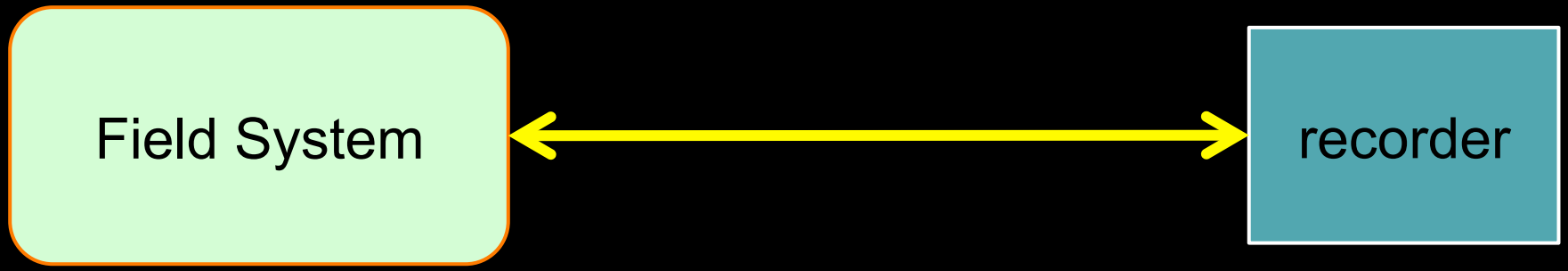
Recording hardware ✓
Recording software
Recording
Transfer

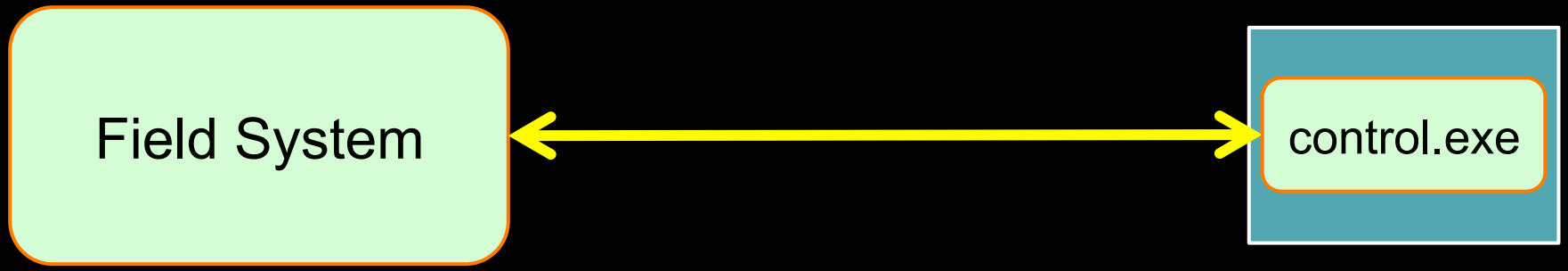
Controlling your VLBI recorder

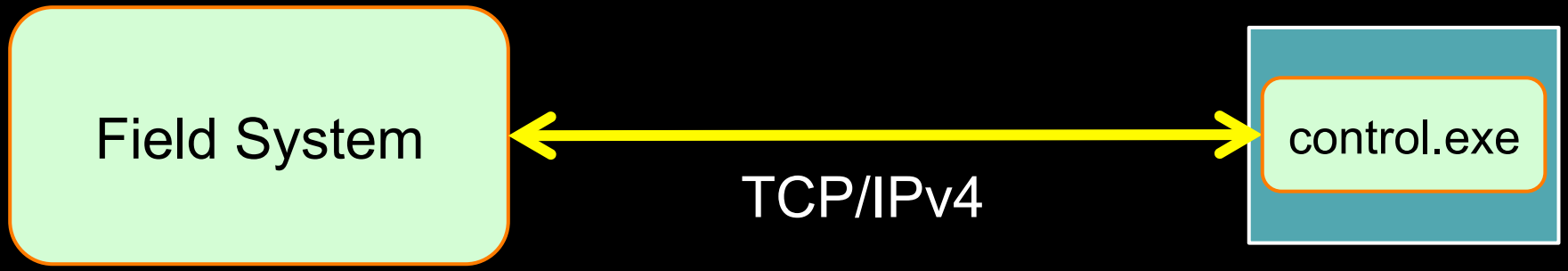

BBC
formatter
recorder


BBC
formatter
recorder
Field System
Schedule:
13:40:10/antcn=17h33m21.2s,16°49'12.2"
13:40:30/disk_record=on:eg063h_ys_no0001
....
13:42:30/disk_record=off


recorder
Field System
Schedule:
13:40:10/antcn=17h33m21.2s,16°49'12.2"
13:40:30/disk_record=on:eg063h_ys_no0001
....
13:42:30/disk_record=off


Field System
recorder


Field System
control.exe

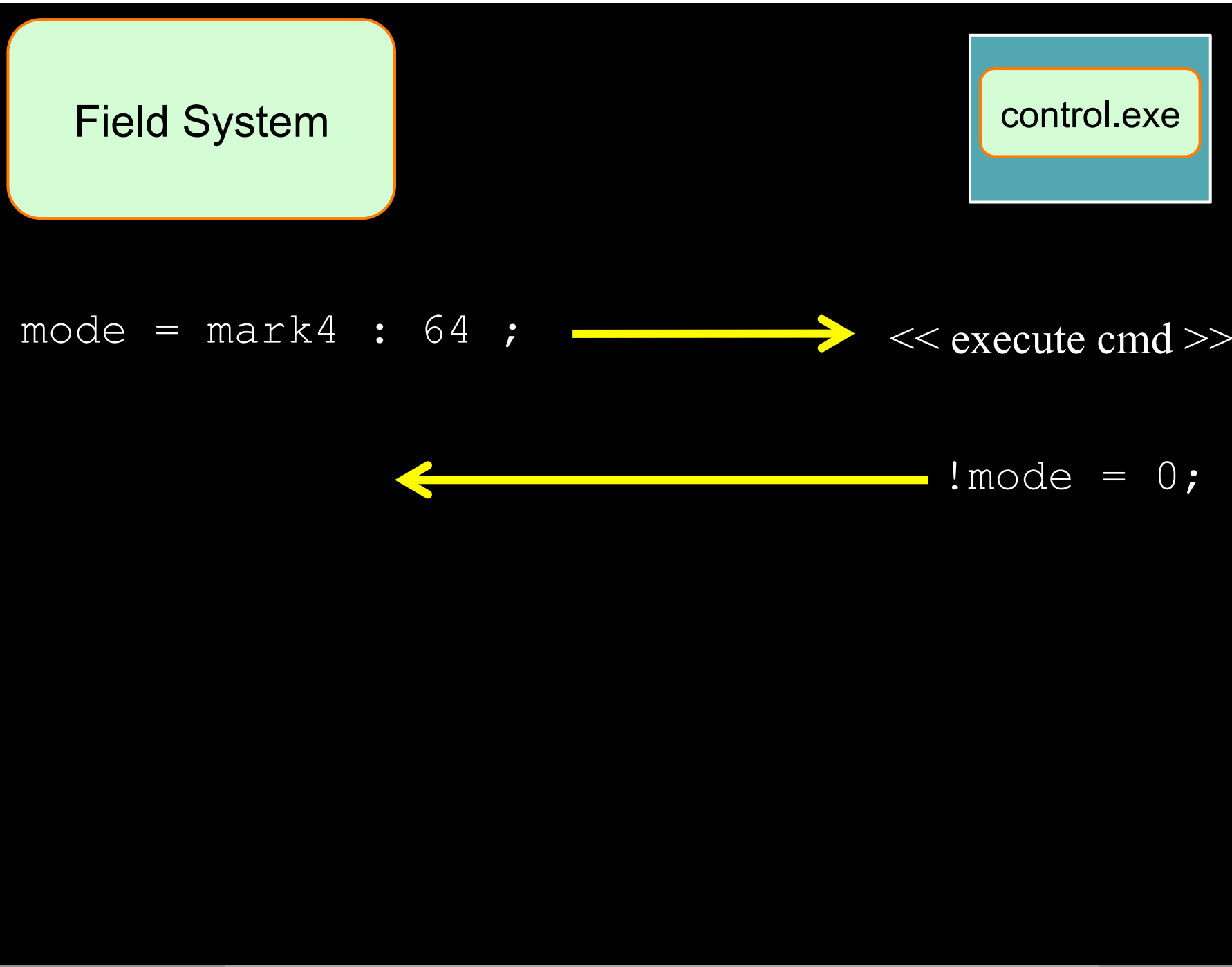


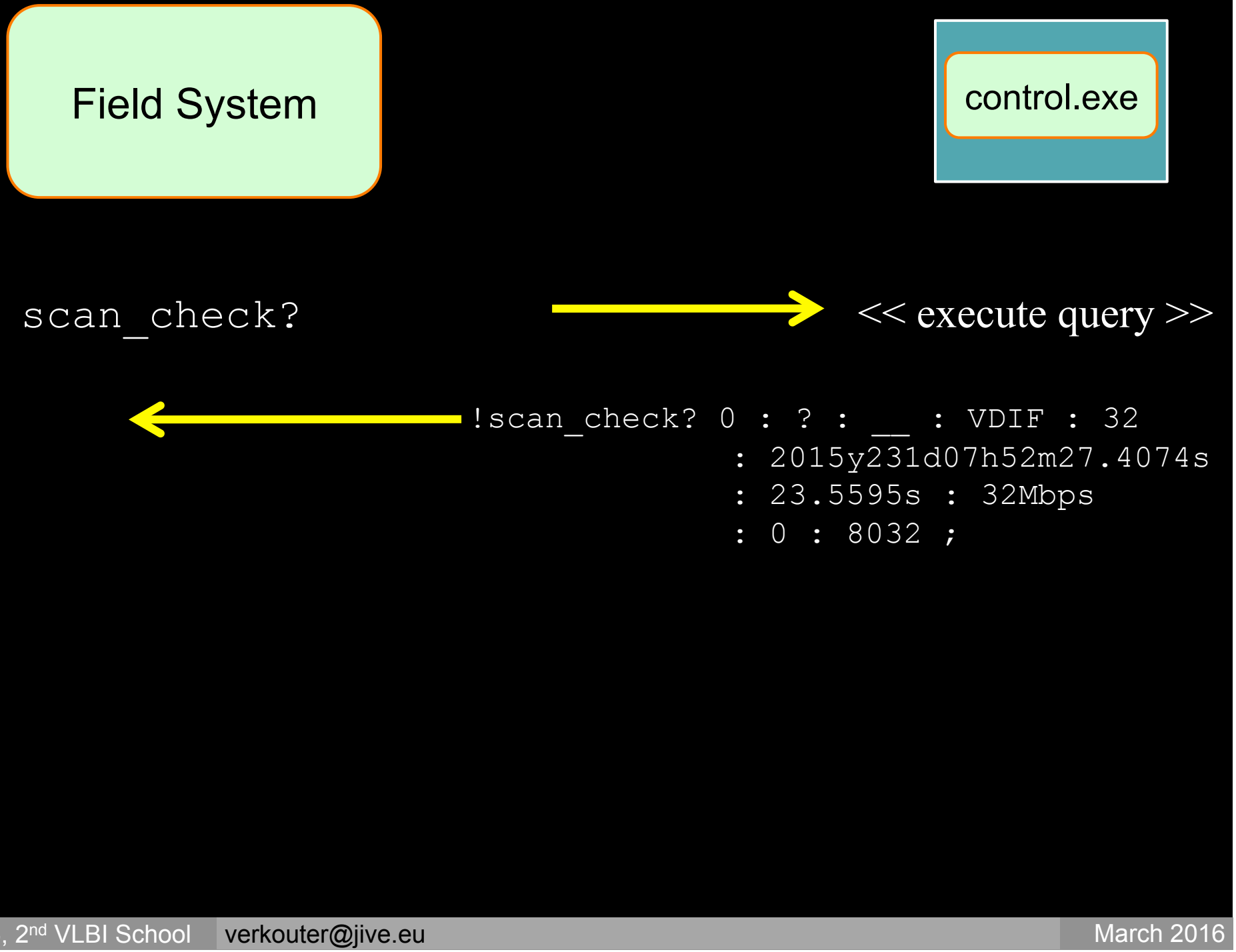
VSI/S – V(LBI) S(tandard) I(nterface) / S(oftware)

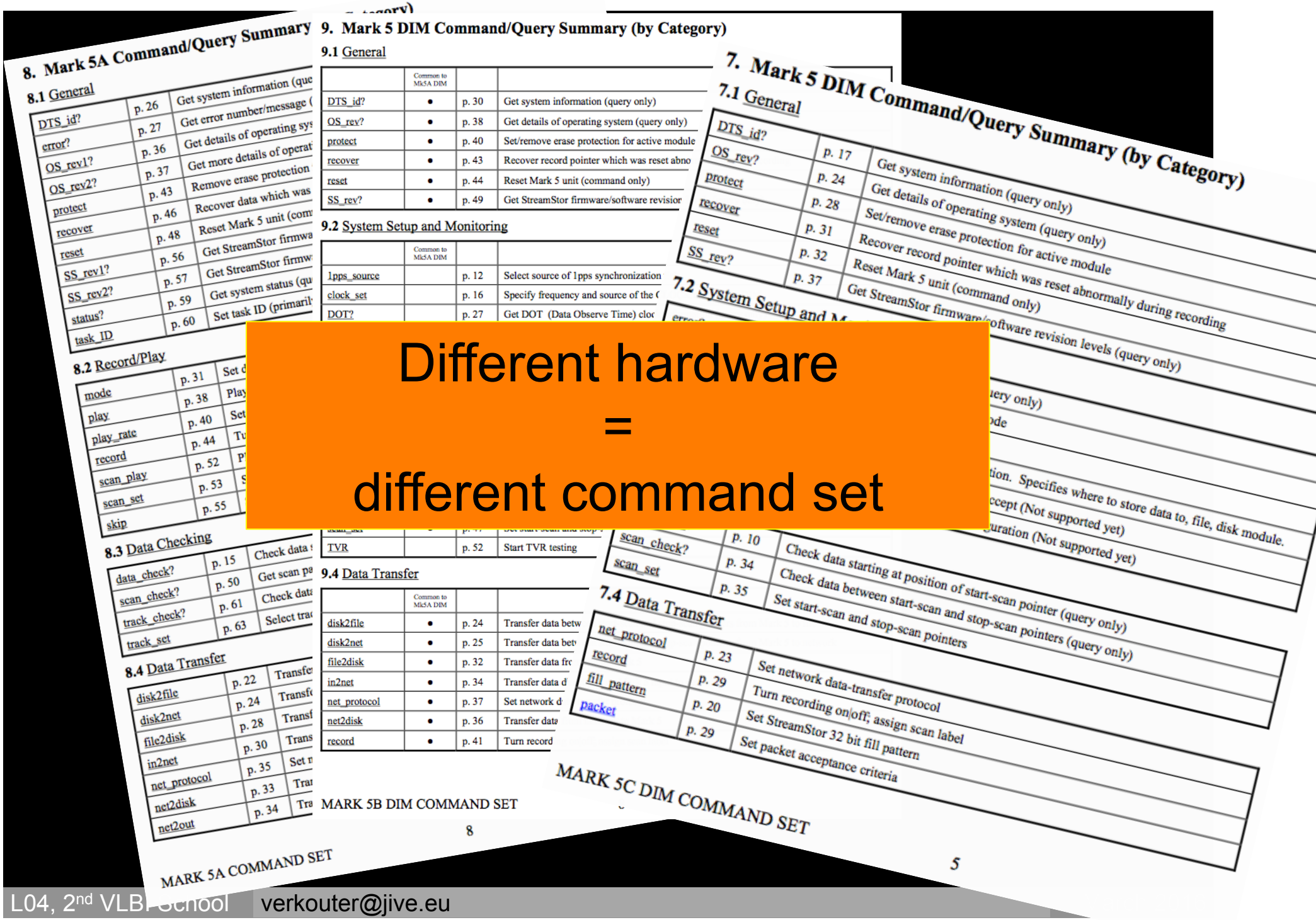
- ASCII based
- `telnet`, raw socket (Python, C, C++, ...)
- query or command yields reply

```
<command> = <param1> [ : <param2> : ... ]
      !<command> = <result code> [ : <value1> : ... ]

<query> ? [ <param1> : <param2> : ... ]
      !<query> ? <result code> [ : <value1> : ... ]
```


Field System
control.exe
mode = mark4 : 64 ;
<< execute cmd >>
!mode = 0;


Field System
control.exe
scan_check?
<< execute query >>
!scan_check? 0 : ? : __ : VDIF : 32
: 2015y231d07h52m27.4074s
: 23.5595s : 32Mbps
: 0 : 8032 ;



Different hardware

=

different command set

	MIT Haystack	EVN
Mark5A	Mark5A	jive5ab
Mark5B	DIMino	jive5ab
Mark5C	drs	jive5ab
Mark6	cplane/ dplane	jive5ab
FlexBuff	-	jive5ab

jive5ab

Replacement for original recorder software

- supports all recorder types
- stable and strict
- under active development
- enables advanced functionality of recorders
- supports multiple network protocols
- allows data transport from any system to another

Recording hardware ✓

Recording software ✓

Recording

Transfer

The VEX file

Describes a V(LBI) EX(periment)

```
VEX_rev = 1.5;
*    SCHED vers: Release 11.4.  March 14, 2015
*    VEX/SCHED:  1.5.87
*    Other versions: Sched:  11.4  Plot: 1.06  JPL-ephem: 0.00
*    log2vex vers: 4.0.2, Release 20 nov 2014
*    Run by jops at 11:49:37  04 Dec 2015
*-----------------------------------------------------------------------------
$GLOBAL;
     ref $EXPER = N15X2;
     ref $EOP = EOP296;

....

$FREQ;
*
def 8391.49MHz16x8MHz;
* mode =  1    stations =Sh:Ur
     sample_rate =  16.000 Ms/sec;  * (2bits/sample)
     chan_def = :  8391.49 MHz : L :   8.00 MHz : &CH01 : &BBC01 : &NoCal; *Rcp
     chan_def = :  8391.49 MHz : L :   8.00 MHz : &CH02 : &BBC02 : &NoCal; *Lcp
     chan_def = :  8391.49 MHz : U :   8.00 MHz : &CH03 : &BBC01 : &NoCal; *Rcp
     chan_def = :  8391.49 MHz : U :   8.00 MHz : &CH04 : &BBC02 : &NoCal; *Lcp
     chan_def = :  8407.49 MHz : L :   8.00 MHz : &CH05 : &BBC03 : &NoCal; *Rcp
     chan_def = :  8407.49 MHz : L :   8.00 MHz : &CH06 : &BBC04 : &NoCal; *Lcp
```

The VEX file

Describes a V(LBI) EX(periment)

```
*-------------------------------------------------------------------------------
$SCHED;                         * Experiment: N15X2
*-------------------------------------------------------------------------------
*
scan No0001;
      start=2015y297d09h00m00s; mode=sess315.X512; source=J1642+3948;
*                 : data_good:   data_stop:  goto_startpos      :pass:wrp:drv;
      station=Ef:     3 sec:     240 sec:  2853.245900800 GB :     :   : 1 ;
      station=Wb:     1 sec:     240 sec: 14376.565760000 GB :     :   : 1 ;
      station=O6:     0 sec:     240 sec: 17216.454323200 GB :     :   : 1 ;
      station=Mc:     1 sec:     240 sec: 15082.992145792 GB :     :   : 1 ;
      station=Nt:     0 sec:     240 sec:     0.000000000 GB :     :   : 1 ;
      station=Sh:     0 sec:     240 sec:  3410.808576000 GB :     :   : 1 ;
      station=Sv:     0 sec:     240 sec: 14488.905216000 GB :     :   : 1 ;
      station=Zc:     3 sec:     240 sec: 11556.188364800 GB :     :   : 1 ;
      station=Bd:     1 sec:     240 sec:  8306.645401600 GB :     :   : 1 ;
      station=Ys:     1 sec:     240 sec:   906.604249600 GB :     :   : 1 ;
      station=Ur:     0 sec:     240 sec:  2820.489574400 GB :     :   : 1 ;
endscan;
```

The VEX file

```
*-----------------------------------------------------------------------------
$SCHED;                        * Experiment: N15X2
*-----------------------------------------------------------------------------
scan No0001;
      start=2015y297d09h00m00s; mode=sess315.X512; source=J1642+3948;
*                 : data_good:   data_stop:  goto_startpos     :pass:wrp:drv;
      station=Ef:     3 sec:     240 sec:  2853.245900800 GB :    :   : 1 ;
endscan;
```

FieldSystem log

```
2015.297.08:20:40.55;Log Opened: Mark IV Field System Version 9.11.8
2015.297.08:20:40.55;location,EFLSBERG,-7.00,50.53,310.0
2015.297.08:20:40.55;horizon1,0.,10.,360.
2015.297.08:20:40.55;antenna,100.0,40.0,30.0,-0.1,365.0,10.0,90.0,azel
...

2015.297.08:20:59.53/mk5/!rtime? 0 : 205108s : 13147.9GB : 82.1685% : ext : 0xfff
2015.297.08:20:59.54/mk5/!dts_id? 0 : mark5b : 21-Sep-2015 15h09m21s : 1 : mark5-
2015.297.08:20:44.07&setup01/mk5b_mode=ext,0xffffffff,,16.000
2015.297.08:20:54.06/mk5/!dot? 0 : 2015y297d8h20m54.0640s : syncerr_eq_0 : FHG_of
...

2015.297.09:00:00.00:disk_record=on
2015.297.09:00:03.06:disk_record
2015.297.09:00:03.06/disk_record/on,n15x2_ef_no0001,305
...

2015.297.09:04:00.00:disk_record=off
```

FieldSystem log

```
2015.297.08:20:40.55;Log Opened: Mark IV Field System Version 9.11.8
2015.297.08:20:40.55;location,EFLSBERG,-7.00,50.53,310.0
2015.297.08:20:40.55;horizon1,0.,10.,360.
2015.297.08:20:40.55;antenna,100.0,40.0,30.0,-0.1,365.0,10.0,90.0,azel
...

2015.297.08:20:59.53/mk5/!rtime? 0 : 205108s : 13147.9GB : 82.1685% : ext : 0xfff
2015.297.08:20:59.54/mk5/!dts_id? 0 : mark5b : 21-Sep-2015 15h09m21s : 1 : mark5-
2015.297.08:20:44.07&setup01/mk5b_mode=ext,0xffffffff,,16.000
2015.297.08:20:54.06/mk5/!dot? 0 : 2015y297d8h20m54.0640s : syncerr_eq_0 : FHG_of
...

2015.297.09:00:00.00:disk_record=on
2015.297.09:00:03.06:disk_record
2015.297.09:00:03.06/disk_record/on,n15x2_ef_no0001,305
...

2015.297.09:04:00.00:disk_record=off
```

Was
the
recording
succesful?

The scan_check? query

- reads a bit of data from the recording
- detect frame format (MarkIV, VLBA, Mark5B, VDIF)
  - mostly by looking for the magic bit pattern (syncword)
  - VDIF done heuristically so may #FAIL
- detect recorded data rate
- detect if data is missing
  - decode time stamp at begin and end of scan
  - $((t_{end} - t_{begin}) \times \text{data\_rate}) - \text{recording\_length}$

The scan_check? query

```
2015.297.11:15:11.10;!scan_check? 0 : ? :
                                  n15x2_ef_no0014 : VDIF : 32 :
                                  2015y297d11h02m01.1389s : 780.786s :
                                  16Mbps : 0 : 8032 ;
```

- scan name
- detected data type
- start time
- length
- data rate
- format specific information

file_check?

Data often found in regular file(s) on disk

- the `scan_check?` equivalent for files
- order of arguments odd but consistent with `scan_check?`

```
file_check? [<strict>] : [<# bytes to read>] : <file name> ;
```

`<strict>`	optional, “0” or “1”
`<# bytes to read>`	optional – for high data rates 1MB of data may not be enough
`<file name>`	the file to check

How do these work?

`scan_check` and `file_check` assume nothing

- search for series of `0xffffffff` in data
  - MarkIV or VLBA sync word
  - amount indicates number of tracks
  - data rate from decoding successive time stamps d_{bytes} / d_t
- search for `0xABADDEED`
  - Mark5B sync word
  - data rate from decoding successive time stamps, or
  - find maximum frame number:
    - data rate = frames/second * 100,00 bytes / frame
- try do decode as if it is a VDIF stream:
  - some properties remain constant between successive frames

How do these work?

In theory can Do It Yourself ...

Use a hexadecimal view of the data file:

```
$> od -t x4 -v -Ax <file> | less
```

From `od(2)`:

`-t x4`	interpret data as 32 bit, print hex: syncwords show up, easy to search
`-v`	verbose, print all lines, otherwise output is compressed
`-Ax`	print addresses in hexadecimal (check correct frame size)

Recording hardware ✓

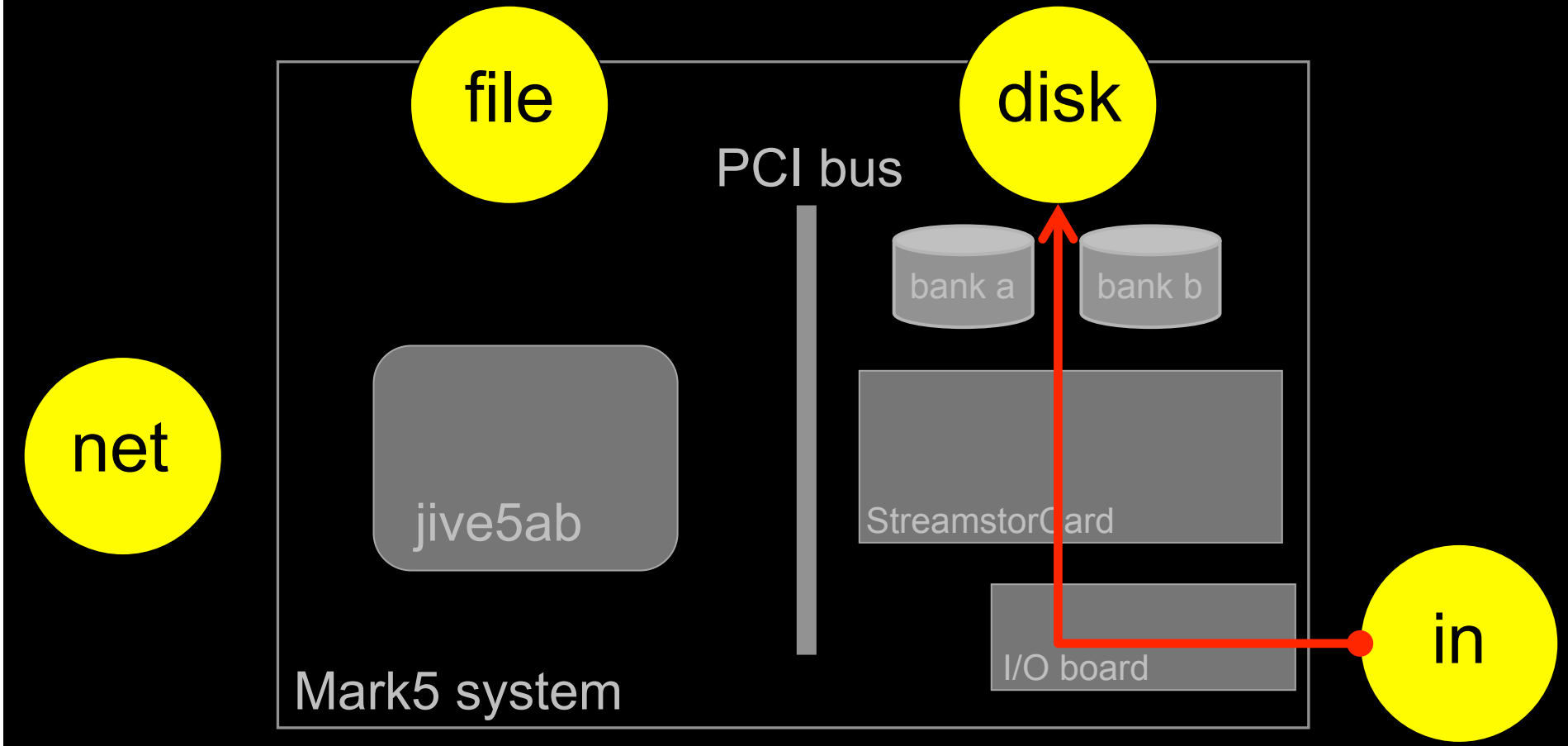
Recording software ✓

Recording ✓

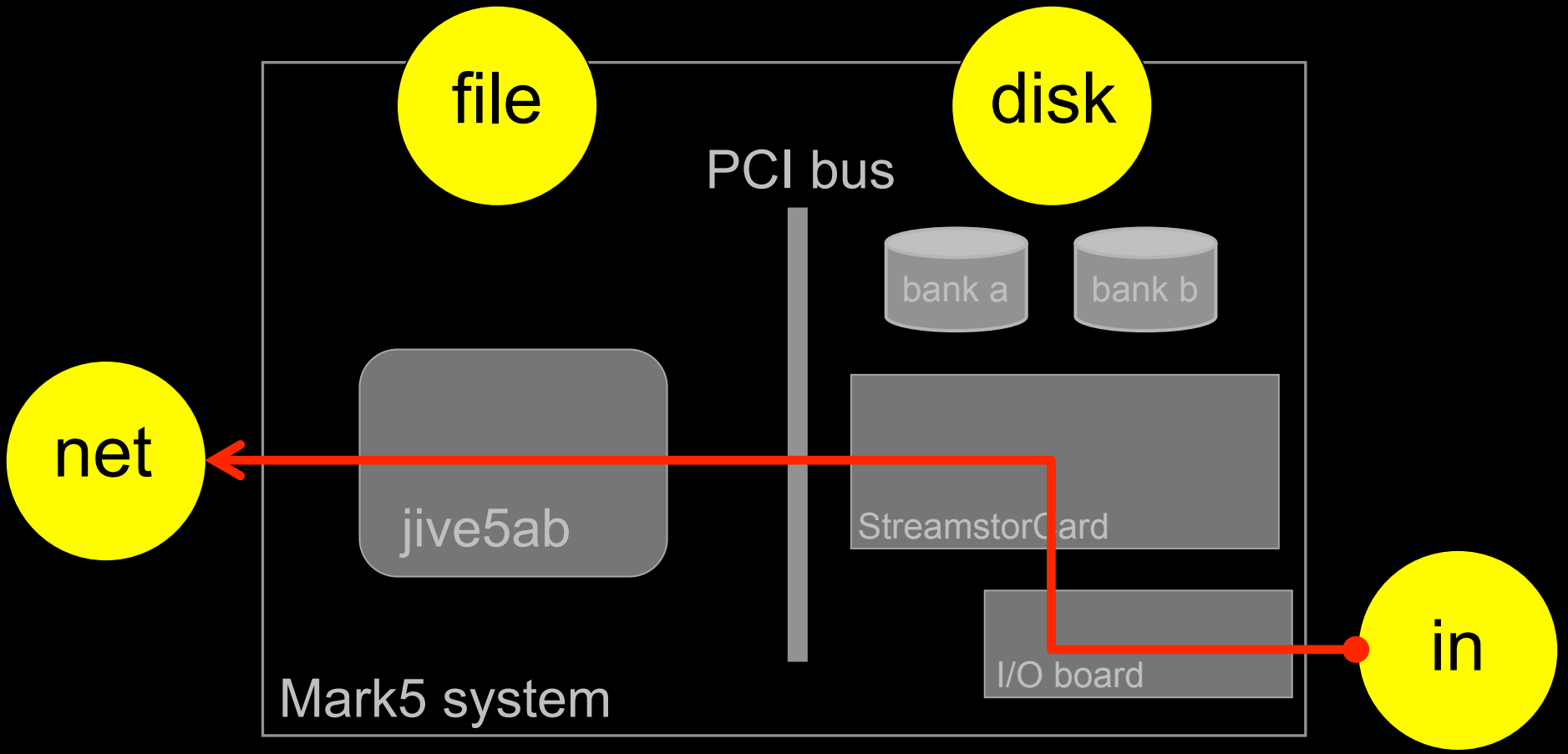
Transfer

Features enabled by jive5ab

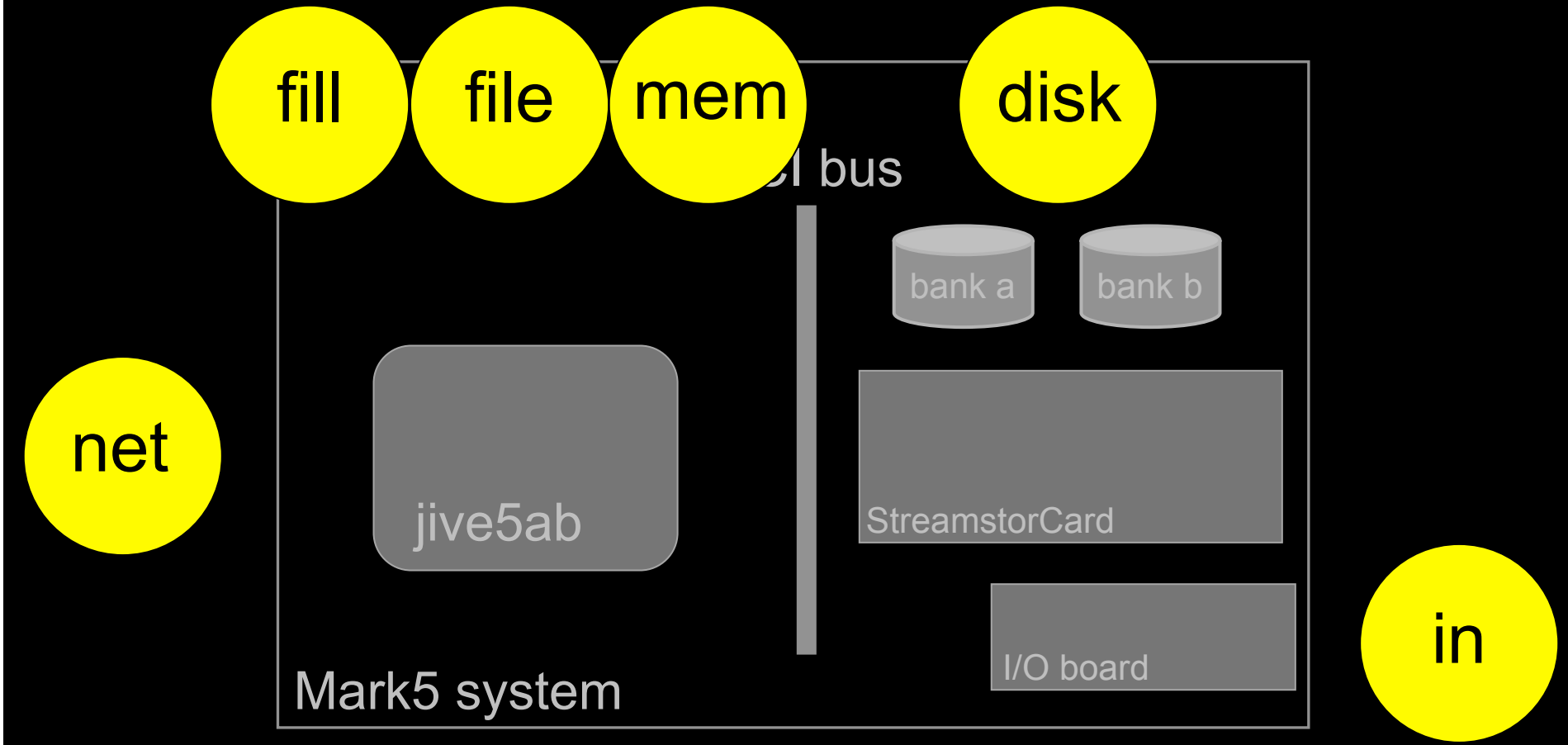
Inside jive5ab



Inside jive5ab

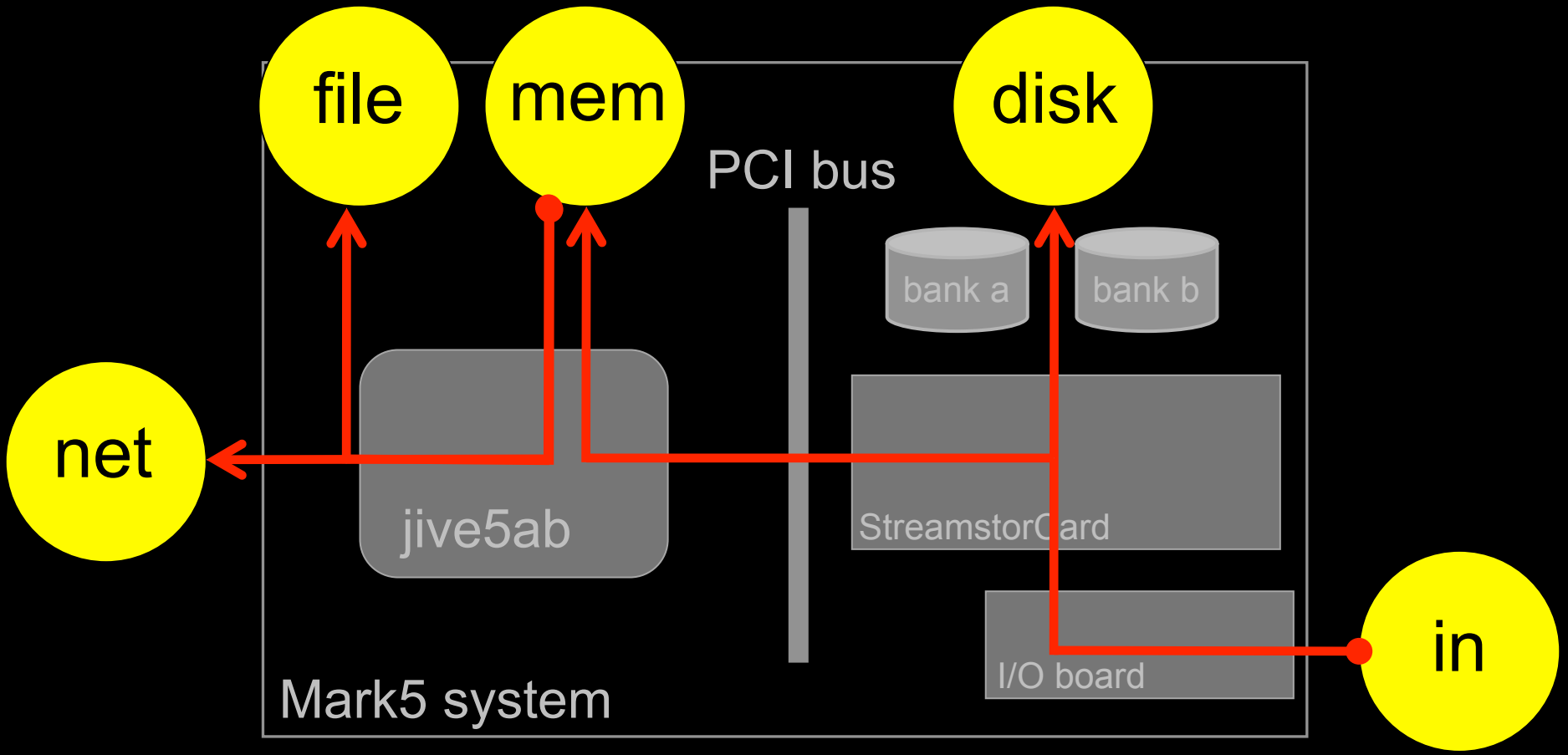


Inside jive5ab



- disk2net, disk2out, disk2file,
- in2net, in2disk, in2fork, in2file
- net2out, net2disk, net2file, net2check, net2sfxc,
- fill2net, fill2file, fill2out
- spill2net, spid2net, spin2net, spin2file, splet2net, splet2file
- spill2file, spid2file, spif2file, spif2net
- file2check, file2mem
- in2mem, in2memfork, mem2net

jive5ab special



Why connectivity is important

V ery

L ong

B aseline

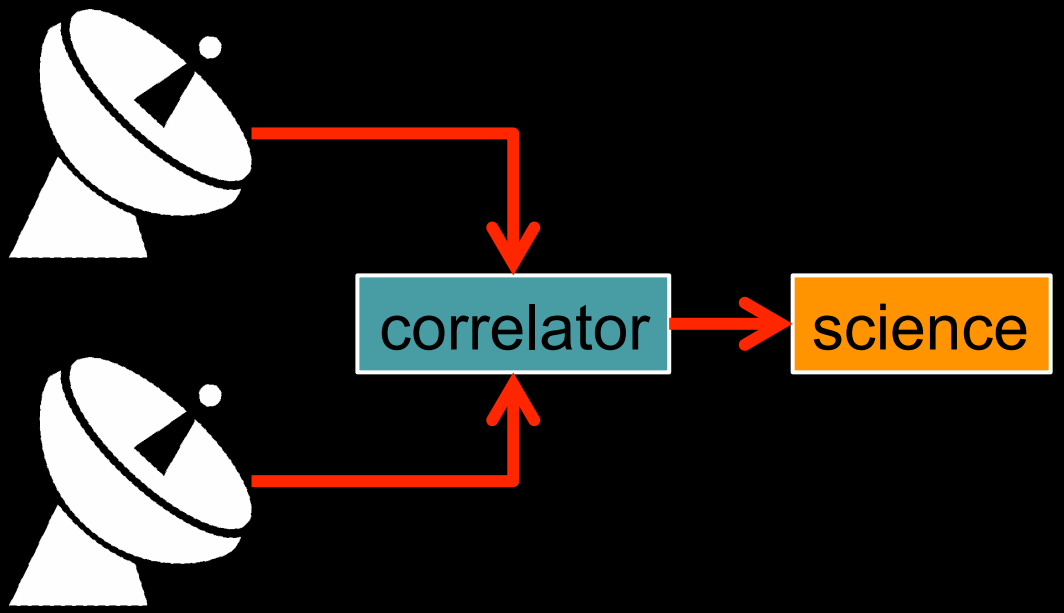
I nterferometry

Very

Long

Baseline

Interferometry



Very

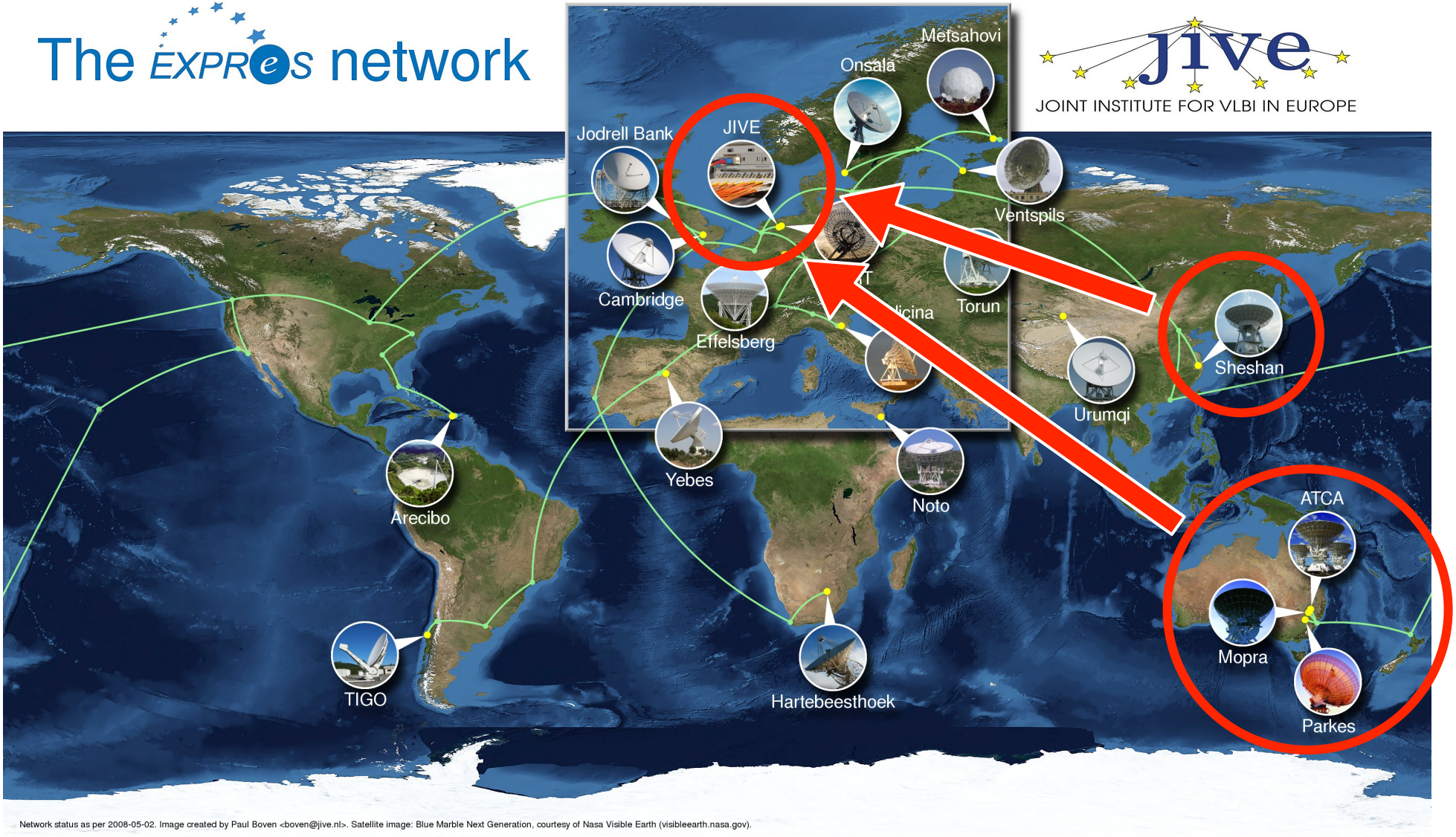
Long

Baseline

Interferometry

The EXPReS network

JIVE
JOINT INSTITUTE FOR VLBI IN EUROPE



Network status as per 2008-05-02. Image created by Paul Boven <boven@jive.nl>. Satellite image: Blue Marble Next Generation, courtesy of Nasa Visible Earth (visibleearth.nasa.gov).

Shipping lumps of metal across the globe suboptimal:

- expensive
- damages disks (= science data loss!)
- introduces long delay
- for *very* large data sets maybe still the only option
- many stations now well connected to The Internet
    - *if* connected, then usually > 1Gbps!

m5copy (a.k.a. e-shipping)

```
$> m5copy [-h] [-udt] [-m MTU] SRC DST
```

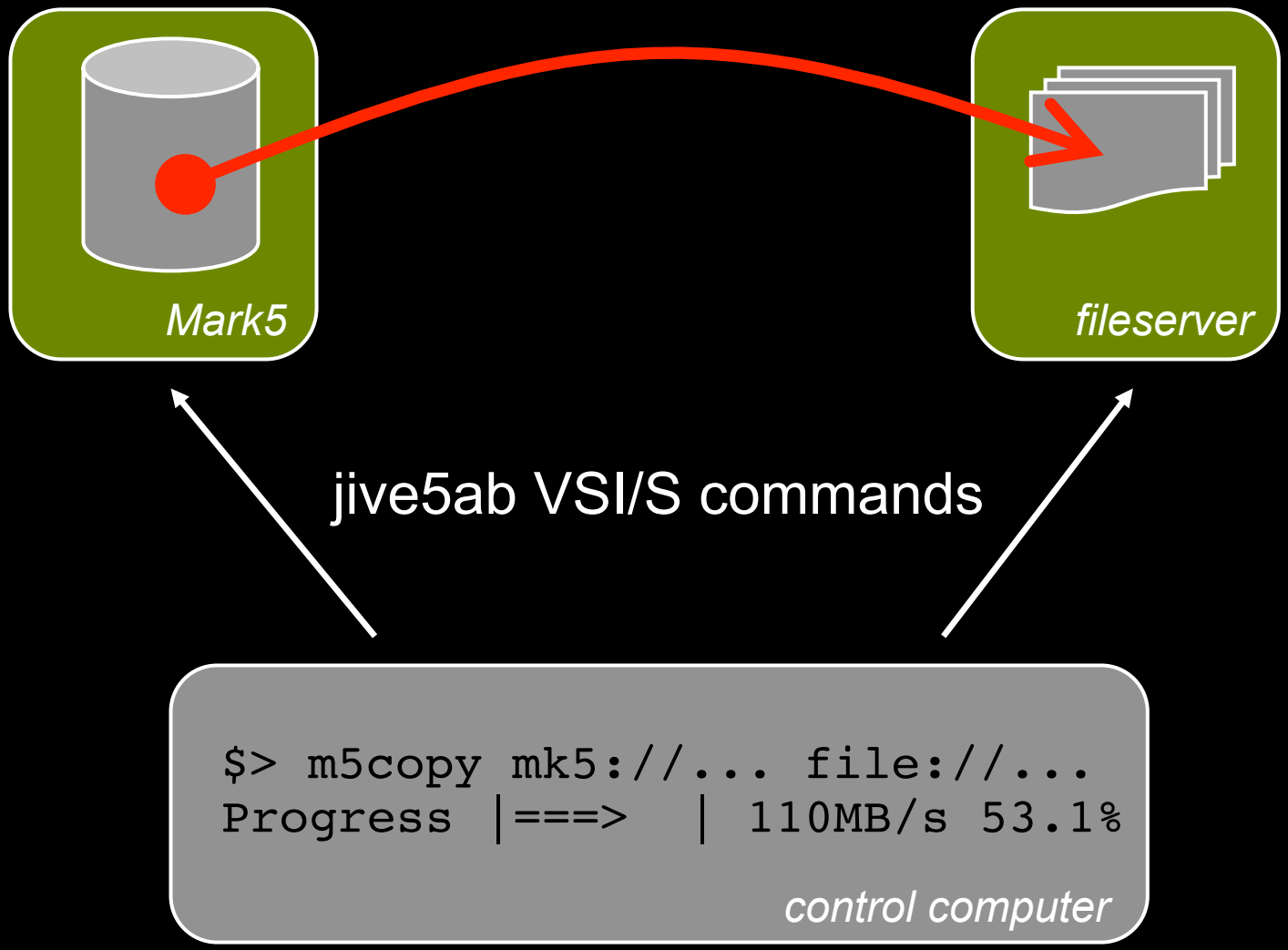
- *"copy VLBI data from somewhere to elsewhere"*

`SRC`, `DST` are url-like data locations

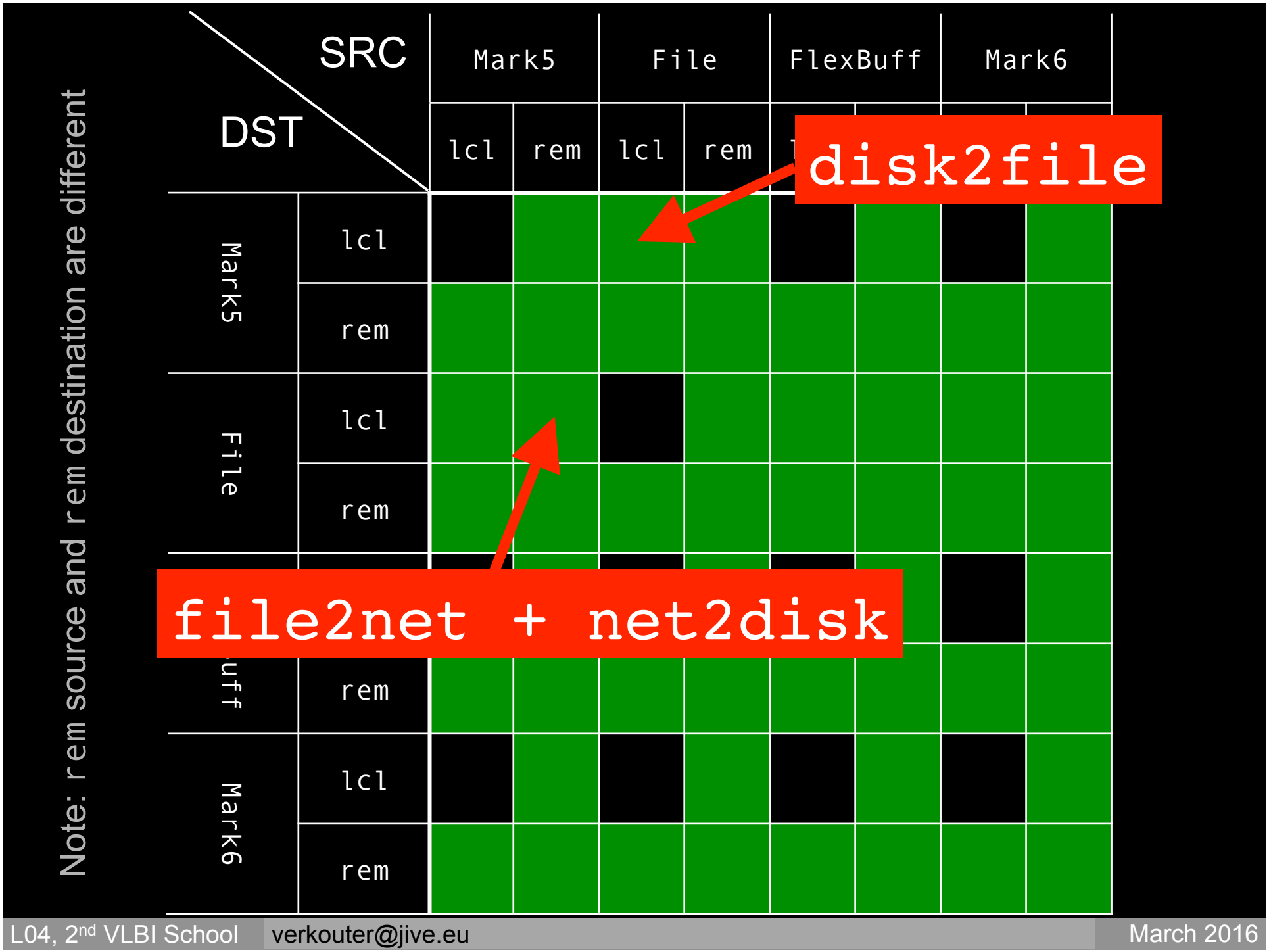
- mk5://.../rk052_ys_no0001
- file://.../path/to/file
- mk6://.../eht_ef_no0230

Supports

- TCP (local network), UDT (international, >> TCP!)
  - TCP collapses on very long distances
- wildcards on scan names, local file names, recordings
- list of scan numbers: `1-10,16,17`
- FlexBuff recording syncing



m5copy (a.k.a. e-shipping)


SRC
DST
Mark5
File
FlexBuff
Mark6
lcl
rem
lcl
rem
lcl
rem
lcl
rem
lcl
rem
lcl
rem
Note: rem source and rem destination are different
disk2file
file2net + net2disk

m5copy capabilities

Copy FlexBuff or Mark6 recordings anywhere:

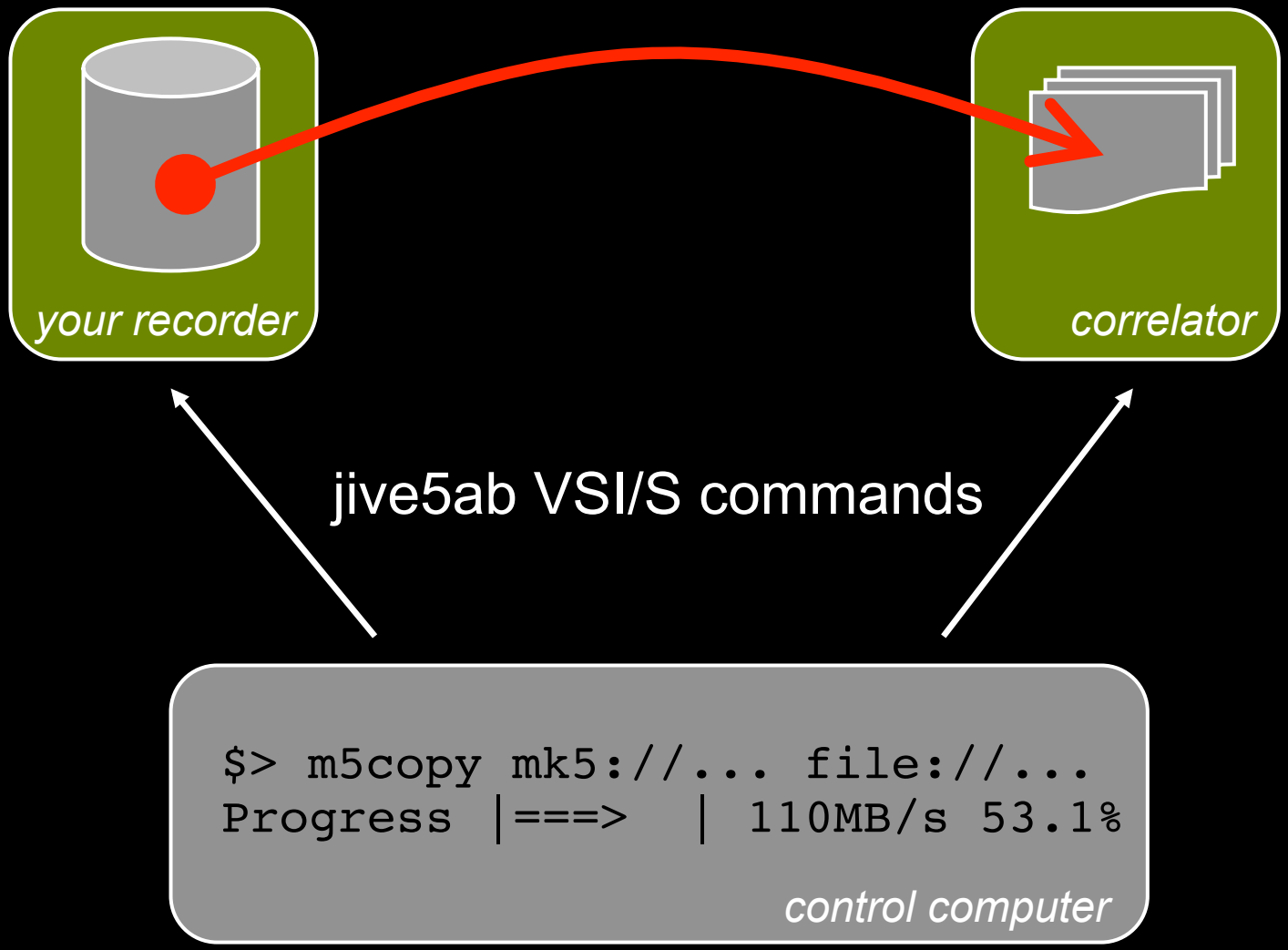
To local or remote file:

```
$> m5copy vbs://.../  file://[host.ip]/path/
$> m5copy mk6://.../  file://[host.ip]/path/
```

To remote Mark5:

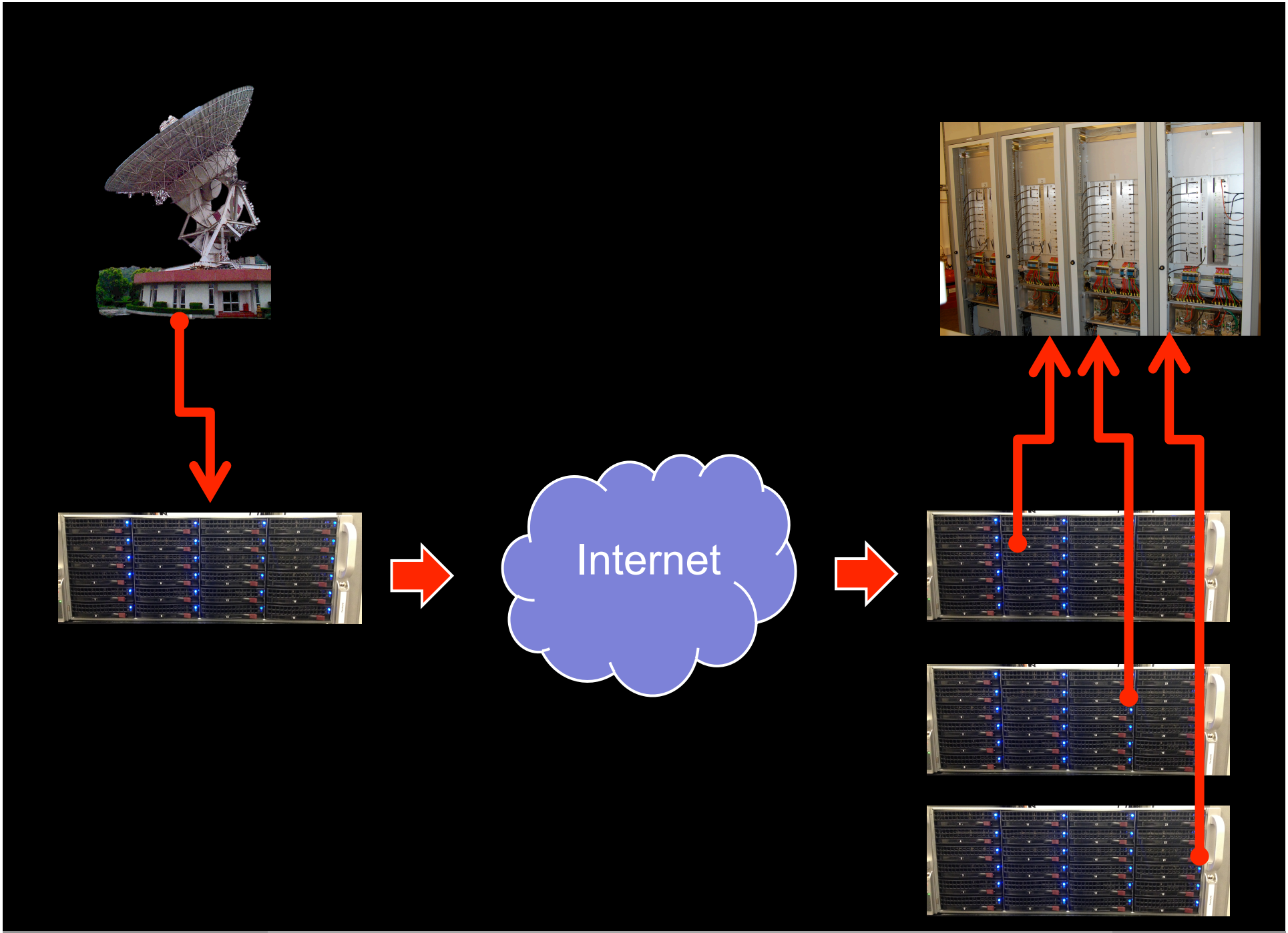
```
$> m5copy vbs://.../  mk5://host.ip/path/
$> m5copy mk6://.../  mk5://host.ip/path/
```

.... etc ...



m5copy (a.k.a. e-shipping)

Disk shipping less operation


Internet

FlexBuff advantages

Independent of FlexBuff layout at station or correlator

vbs → vbs transfers are a sync operation

- only missing bits are transferred

m5copy (a.k.a. e-shipping)

Station

/mnt/disk0/eg053/

- eg053.00000000
- eg053.00000006

/mnt/disk1/eg053/

- eg053.00000001
- eg053.00000007

/mnt/disk2/eg053/

- eg053.00000002
- eg053.00000005

/mnt/disk3/eg053/

- eg053.00000003
- eg053.00000004

Correlator

/mnt/disk0/eg053/

- eg053.00000000
- eg053.00000006
- eg053.00000001
- eg053.00000007

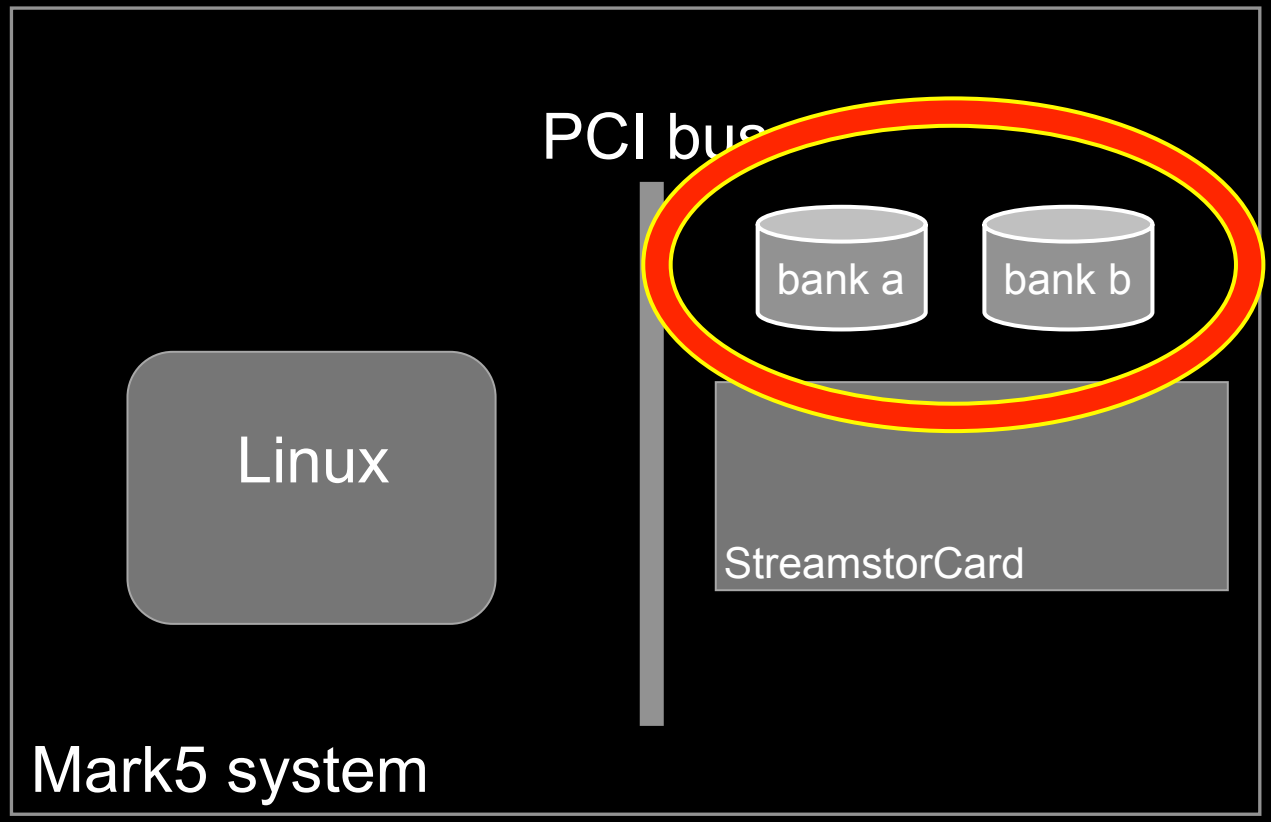
/mnt/disk2/eg053/

- eg053.00000002
- eg053.00000005
- eg053.00000003
- eg053.00000004

m5copy (a.k.a. e-shipping)

Easy access to your VLBI data

Mark5



Mark5

Data access problems on Mark5:

- format on disk is proprietary*
- no vendor support* for accessing as regular files
  - there is no such notion as ‘files’
  - disks are seen as a sequence of bytes
  - user application tracks recording boundaries
- OH NOES!

(*) The Mark5 system is built + sold by Conduant Corporation, Boulder CO, USA

Mark5

DirList.py program

list directory of scans on Mark5 disk pack

```
jops@ccsbeta:~$ DirList.py -a 10.88.0.50
nscans 910, recpnt 38448131885744, VSN <OSOD+135/48000/1024>
  n' scan name                        start byte         end byte
---- ---------                    -------------   -------------
   1 f15m1_o8_no0001                          0    19267804160
   2 f15m1_o8_no0002              19267804160    34680392896
   3 f15m1_o8_no0003              34680392896    57805468672
   4 f15m1_o8_no0004              57805468672    88639288576
   5 f15m1_o8_no0005              88639288576   119475084352
   6 f15m1_o8_no0006             119475084352   150311354016
   7 f15m1_o8_no0007             150311354016   181134748384
```

FlexBuff

/mnt/disk0/eg053/

- eg053.00000000
- eg053.00000006

/mnt/disk1/eg053/

- eg053.00000001
- eg053.00000007

/mnt/disk2/eg053/

- eg053.00000002
- eg053.00000005

/mnt/disk3/eg053/

- eg053.00000003
- eg053.00000004

FlexBuff


/mnt/disk0/eg053/
eg053.00000000
eg053.00000006
/mnt/disk1/eg053/
eg053.00000001
eg053.00000007
/mnt/disk2/eg053/
eg053.00000002
eg053.00000005
/mnt/disk3/eg053/
eg053.00000003
eg053.00000004
Data scattered across disks!

File System in User Space

FUSE project

`https://github.com/libfuse/libfuse`

- allows implementation of *virtual* file systems
  - `sshfs`: remote file system over ssh
  - `ntfs`: access Windows file system from Linux
- can be overlaid over real file system

File System in User Space

Specifically for VLBI:

- `fuseMk5A`: present Mark5 recording as single file
- `vbs_fs`: present FlexBuff scattered data as single file

Typical usage:

```
$> mkdir /path/to/mountpoint
$> fuseMk5A [options] /path/to/mountpoint
$> vbs_fs [options] /path/to/mountpoint
```

```
$> ls -l /path/to/mountpoint
total 0
-rwxrwxrwx 0 root root   2692300800 Aug 31 09:05 EXP_STN_tvg1024
-rwxrwxrwx 0 root root   2820505600 Aug 31 09:06 EXP_STN_tvg1024a
-rwxrwxrwx 0 root root   2307686400 Aug 31 09:06 EXP_STN_tvg1024b
-rwxrwxrwx 0 root root 107888641544 Sep 14 21:31 fr024_hh_no0001
-rwxrwxrwx 0 root root 107893157216 Sep 17 15:18 fr024_hh_no0007
-rwxrwxrwx 0 root root  37435801600 Oct  6 12:17 rg10b_hh_no0013
-rwxrwxrwx 0 root root  37467852800 Oct  6 12:21 rg10b_hh_no0014
```

Recording hardware ✓

Recording software ✓

Recording ✓

Transfer ✓

Lecture 04 in one slide:

VLBI data is:

- filtered from the sky
- captured in channels
- formatted in frames
- recorded on hard disks
- transported to the correlator
  - sending disk packs
  - e-shipping via the internet
- accessed using virtual file systems

Thank you for your attention

Availability

`http://www.jive.eu/~verkout/evlbi/`

`jive5ab` ".deb" installation, source code

`http://www.jive.eu/~verkout/flexbuff/`

Flexbuff scripts and documentation

`http://www.jive.eu/~verkout/evlbi/m5copy`
`http://www.jive.eu/~verkout/evlbi/DirList.py`
`http://www.jive.eu/~verkout/evlbi/SSErase.py`

direct download links, always latest version

`http://www.jive.eu/~verkout/evlbi/m5copy.html`
`http://www.jive.eu/~verkout/evlbi/changelog`

changelogs of `m5copy` and `jive5ab` for inspection