Mark 6: design and status update
Why Mark6? drivers for ever-increasing data rates

- Sensitivity!
- VLBI2010 – enables smaller antennas & shorter scans
- EHT – enables coherent detection of Sgr A* at mm wavelengths (through a fluctuating atmosphere)
Mark 6 goals

- 16Gbps sustained record capability
  - >=32Gbps burst-mode capability

- Support all common VLBI formats
  - possibly general ethernet packet recorder

- COTS hardware
  - relatively inexpensive
  - upgradeable to follow Moore’s Law progress

- 100% open-source software
  - Linux O/S

- Other considerations
  - playback as standard Linux files
  - e-VLBI support
  - smooth user transition from Mark 5
  - preserve Mk5 hardware investments, where possible
Prototype Mark 6
Mark 5 SATA Drive Module Upgrade to Mark 6

- New Front Panel
- Connectors for two eSATA cables
- 8x LED (1 per drive)
- Re-use Handle from old Module
- New Latch provided (pre-installed on new Panel)
- New PCB and power connector
- Easily removable disks
- Cooling slots

Front

Rear
New Drive Module Backplane (x2):
- Sequences power to disks
- Regulates voltage at disk power pins

New Connector Board:
- Simple disconnect to allow easy removal of Module Tray from chassis

Module guide rails

Cooling fans

Connections to chassis Power Supply

Module Tray
Conduant prices (preliminary)

- Mark 6 system chassis (complete, w/electronics); final motherboard yet to get determined ~$10K
- Mark 6 expansion chassis (complete) $2675
- Cable tray $55
- Data cable (each) $85 each

- Mark 5-to-Mark 6 system-chassis upgrade kit (DIY) (w/o power supply; can re-use 850W PS) ~$8K
- Mark 5-to-Mark 6 expansion-chassis upgrade kit (DIY) $675

- Mark 6 module (empty) $495
- Mark 5-to-Mark 6 module upgrade kit (DIY) $250

Project late 2012 availability for complete Mark 6 system
People come and go, but the project carries on...

- Prototype software v.0 achieved 16 Gb/s to a RAID array
- Author of prototype (David Lapsley) left Haystack for greener pastures
- Roger Cappallo and Chet Ruszczyk writing production version
Demonstration Experiment

- June 2012
- Westford – GGAO
- done with prototype software (v.0)
- 16 Gb/s
  - 4 GHz bandwidth on the sky
  - dual polarization with 2 GHz IF’s
  - processed as four 512 MHz channels
3C84.wixdke, 171-1927, AC
S002_Aw - S004_Cg, fgroup X, pol LL

- Fringe quality: 9
- SNR: 940.9
- Int time: 9.720
- Amp: 55.201
- Phase: 0.0
- PFD: 0.0e+00
- Delays (us):
  - SBD: 0.000007
  - MBD: 0.000000
- Fringe rate (Hz): -0.030985
- Ion TEC: 0.00
- Ref freq (MHz): 8592.0000
- AP (sec): 0.480
- Exp.: y2
- Exper #: 3407
- Yr:day: 2012:171
- Start: 192730.00
- Stop: 192740.08
- FRT: 192735.00
- Corr/FF/build:
  - 2012:235:191323
  - 2012:237:152123
  - 2012:237:083218
- RA & Dec (J2000):
  - 03h19m48.1501s
  - +41°30′42.104″

Amp. and Phase vs. time for each freq., 21 segs, 1 APs / seg (0.48 sec / seg.), time ticks 1 sec

- Amplitude
- Phase
mark6 block diagram
c-plane

- control plane
- written in python
- interface to user (e.g. field system)
  - VSI-S protocol
- responsible for high-level functions
  - aggregation of disks into modules
  - scan-based setup & record
  - error-checking, etc.
dplane

- data plane
- written in C
- implements the high-speed data flow
- input from NIC’s
- output to disks within mk6 modules
- manages:
  - start and stop of data flow via packet inspection
  - organization of data into files
  - addition of metadata to files
dplane block diagram
dplane - Technical Highlights

- *pf_ring* used for high-speed packet buffering
- Efficient use of multiple cores - based on # of available cores
  - *smp affinity* of IRQ's
  - *thread binding* to cores
- Most of physical RAM grabbed for large ring buffers and locked in
  - One large ring buffer per stream
  - 1..4 streams - can be changed dynamically
- ~10 MB blocks scattered to files resident on different disks
  - Prepended block# for ease of reassembly
  - Uses faster disks to keep up with flow, but balances disk usage as much as possible
Additional Features

- capture to ring buffers is kept separate from file writing, to facilitate eVLBI, etc.

- FIFO design decouples writing from capturing (e.g. keep writing during slew)

- all mk6 software is open source for the community

- gather uses asynchronous I/O to read n disks into n ring buffers, and write in correct order to single file
Software versions and strategies

**v.0 prototype (RAID 0)**
- command line one-off control

**v.1 operational RAID-based code**
- continuous operation
- control via messages

**v.2 with single output file per stream:**
- write (ordinary) Linux files on RAID arrays
- can use normal file-based correlation directly

**v.2 with multiple files data need to be reconstructed:**
- gather program – interim solution
  - does so very efficiently
  - requires an extra step
- FUSE/mk6 interface could be written
- in difx: will likely implement native mk6 datastream
Current Status

- control plane and v.2 data plane functional
- dplane v.2 works robustly with 8 Gb/s into
  - SSD module continuously
  - 8 disk module configured as RAID array
  - 8 disk module with scattered disks
- performance testing
- integration of two planes
Timeline

- Dec 2011: v.0 prototype achieved 16 Gb/s to four 8 disk RAID arrays
- July 2012: v.1 operational dataplane code using RAID array
- Sept 2012: integration of v.1 control and data plane codes
- Sept 2012: v.2 dataplane code with scattered filesystem
- Oct 2012: complete integration of control and v.2 dplane
- Oct-Nov 2012: performance testing, assessment, & tuning
- Jan 2013: first operational use for VLBI2010 (8 Gb/s on 1 module)