The Diamond Libera Daemons

Michael Abbott

Diamond Light Source
michael.abbott@diamond.ac.uk

Libera Workshop 2011

Libera at Diamond

At Diamond we have reimplemented or reworked most of the core components making up Libera.

System	I-Tech	Diamond
Linux	2.6.20.14	2.6.30.9
Rootfs	Debian base	Busybox base
Libera device driver	Original	Original
Health daemon	Simple PI fan controller	PI controller with programmable interface
Event daemon	CSPI server	(None)
Clock daemon	lplld	clockPll
Signal conditioning	ldscd	Integrated into EPICS server
System interface	CSPI	EPICS

Libera Compatibility

- The CSPI API was orginally conceived as the common stable interface to Libera. However:
  - CSPI applications need to be compiled and linked against the CSPI library that they're using.
  - CSPI tightly integrates all the Libera components, so Diamond customisation would have become impractical.
- Diamond Libera components rely on the kernel driver interface. This is stable, well defined, and well behaved.

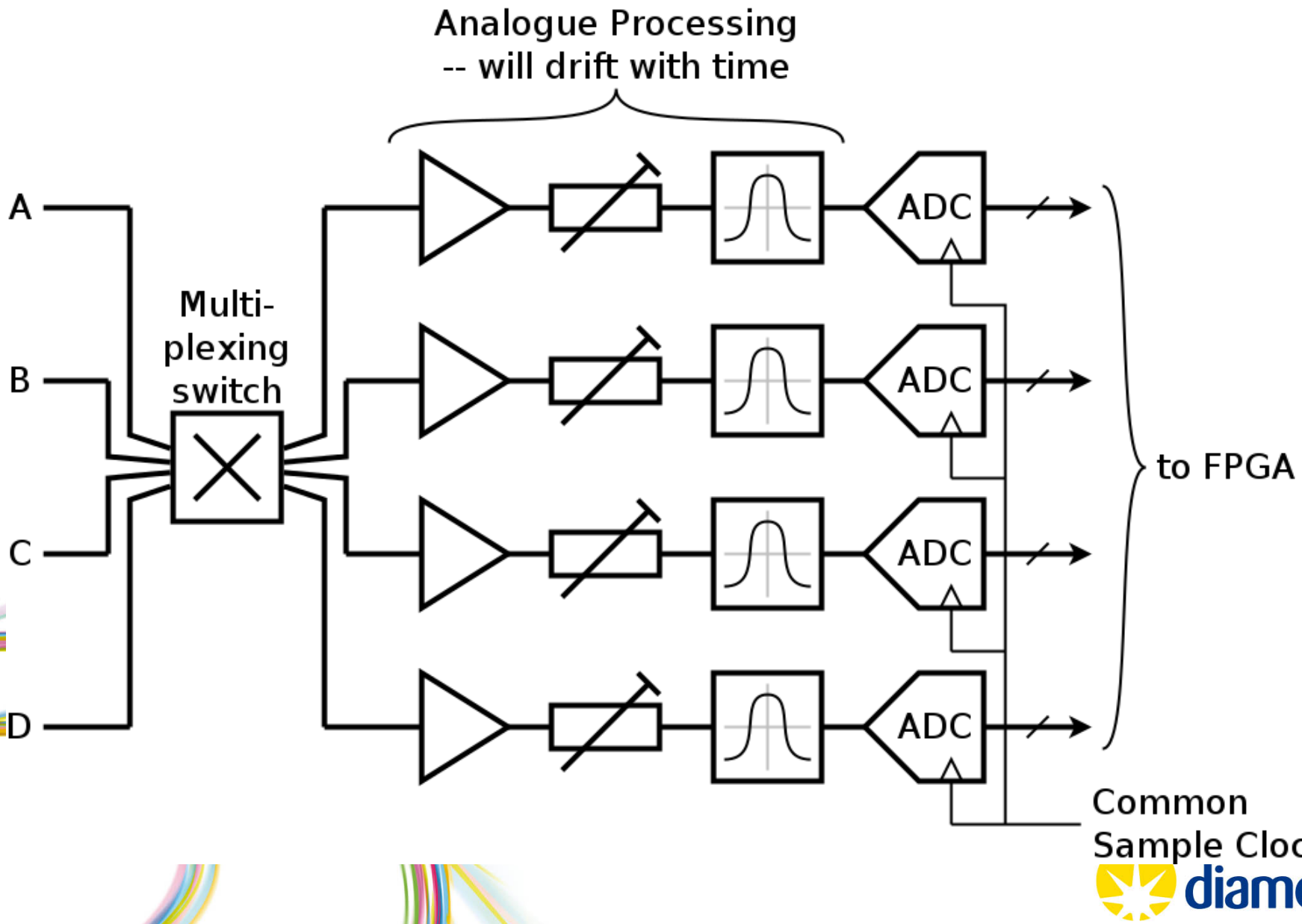
Diamond Clock Daemon

The i-Tech clock daemon was rewritten because the original daemon failed to hold lock and keep stable phase. The Diamond version was written to:

- Keep stable phase for months even when machine RF frequency changes abrubptly
- Support accurate records of global synchronisation status
- Provide detailed operating status information
- Support dynamic changing of phase and frequency offsets

I-Tech subsequently rewrote `lplld` to address many of these issues.

RF Switches and Signal Conditioning



Signal Conditioning

- Signal Conditioning algorithm computes estimates of channel gains using information from the multiplexing switch.
- Channel gain corrections are applied directly to ADC readouts to compensate for gain differences and drifting.
- Gain correction is vital for long term position stability, fundamental to the design of Libera.
- Attenuation control needs to work in cooperation with signal conditioning.
- Thus Automatic Gain Control (AGC) is integrated into signal conditioning.

DLS Signal Conditioning — Differences from i-Tech

- More flexible scheduling of signal conditioning: we normally do a complete compensation calculation every 5 seconds.
- Interval is programmable and computation is fast.
- Amplitude and phase computed together in one calculation.
- Gain compensation settings do not need to be pre-“trained”, no need to maintain tables of coefficients.
- Calculation of signal levels quality is more flexible, safely rejects disturbances from injection transients.
- Detailed reports on computed coefficients available for live inspection and archival.
- Fractional changes to coefficients applied (using IIR) to ensure resistance against sudden changes.

Diamond Automatic Gain Control

- At Diamond normally use global AGC — it's important to avoid attenuator changes during global feedback, so local AGC is normally disabled.
- Local Diamond AGC implemented using FPGA MAXADC register, so very accurate.
- Local AGC algorithm extremely simple: increment or decrement attenuation if MAXADC out of range.
- AGC correction operates at 10 dB per second, easily fast enough to follow machine injection.
- Diamond AGC has no notion of "power", simply works with attenuator settings and MAXADC readings.

Diamond Signal Conditioning — CSPI Compatibility?

- Currently integrated into Diamond EPICS driver, would need to be a separate daemon to work with CSPI.
- Diamond Signal Conditioning has more control parameters and returns more information than the i-Tech ldscd daemon.
- Diamond AGC works in terms of attenuation in dB, no notion of gain or mapping tables.
- Proposal currently on the table to create a separate DSC daemon derived from the Diamond controller and maintained by Diamond.
- Who would maintain the CSPI interface?

Detailed DSC Algorithm

The following slides present some mathematical details of the signal conditioning algorithm implemented by Diamond.

Detailed DSC Algorithm: The Model

The basic model is that each channel c has gain Gc and that at switch position n the input Xb from button b is processed by channel $c_n b$ resulting in a measured signal of

$$Z_n b = G(c_n b) \cdot Xb$$

The purpose of DSC is to compute $Kc \approx G^{-1} c$ so that we in fact measure

$$Y_n b = K(c_n b) \cdot G(c_n b) \cdot Xb \approx Xb$$

In practice Gc also depends on switch position, so we work with

$$Y_n b = K_n(c_n b) \cdot G_n(c_n b) \cdot Xb \approx Xb$$

Detailed DSC Algorithm: Computation

From a sufficiently long DD waveform we can compute $Y_n b$ for all buttons and all relevant switch positions. Note that as this is IQ data we get Y as an array of complex numbers.

Recover $Z_n b = Y_n b / K_n(c_n b)$, assume $\text{average}_c(Gc) = 1$, compute

$$\begin{aligned}\tilde{X}b &= \text{average}_n(Z_n b) = \text{average}_n(G_n(c_n b) \cdot Xb) \\ &= \text{average}_n(G_n(c_n b)) \cdot Xb \\ &\approx \text{average}_c(Gc) \cdot Xb \approx Xb\end{aligned}$$

Then from $G_n(c_n b) = Z_n b / Xb$ can compute new correction K':

$$K'_n(c_n b) = \tilde{X}b / Z_n b$$

Detailed DSC Algorithm: The “average”

Computing the “average” of an ensemble of complex numbers has its complexities. Experiments on Libera show that the best way to compute $\text{average}_n(z_n)$ is as

$$\text{average}_n(z_n) = \text{geo}_n|z_n| \cdot \exp(i \cdot \angle(\text{mean}_n z_n))$$

— a curious blend of arithmetic and geometric means!

$$\text{geo}_n z_n = \sqrt[N]{\prod_n z_n} \qquad \text{mean}_n z_n = \frac{1}{N}\sum_n z_n$$

The calculation of $\angle\overline{z}$ is reliable when the values z_n all have phases with much less than 180° divergence. The phase differences between amplifier channels in Libera is normally less than 60°.