A Digital Backend for the GBT 1cm and 3mm Continuum Radiometers

Martin Shepherd, California Institute of Technology

September 11, 2002

This page intentionally left blank.

Abstract

In Watts et. al. 2002 [1], the preliminary design of a 1cm dual-beam receiver for the GBT was described. The current document is a proposed design for a digital backend for use with both this receiver, and the 3mm receiver. It is proposed that the backend should consist of the combination of three parts. A digitization section, an FPGA based phase demodulation and digital integration package, plus a single board control and monitoring computer, with an Ethernet connection to the GBT monitor and control system.

Contents

1 Introduction 5

2 Green Bank specifics 8

2.1 The inputs of the magic tees 8

2.2 The basic integration time 8

2.2.1 An alternative scheme using multiple gains 10

2.3 Phase switching 11

2.4 Phase switch demodulation 12

2.4.1 The benefits and drawbacks of postponing differencing 13

2.5 Unresolved problems 14

2.5.1 Controlling two receivers 14

2.5.2 Shutting down and restarting the real-time computer remotely 14

3 The logical components of the back end 14

3.1 A phase-switching state machine 14

3.2 An analog integration and digitization unit 15

3.3 A phase-switch demodulator 15

3.4 A serial to parallel conversion unit 16

3.5 A digital integration unit 16

3.6 A real-time clock 16

3.7 A backend control and data acquisition interface 17

4 The physical components of the backend 20

4.1 The A/D converters 21

4.2 The FPGA 21

4.3 The real-time CPU 21

4.4 An enclosure and power supply for the real-time CPU 24
4.5 A hard disk drive 24
4.6 A 1PPS interrupter 24
4.7 Power supplies 25
4.8 The backend software 25
4.8.1 Kernel space drivers 25
4.8.2 User space software 26
4.8.3 Network Time Protocol client software 26
4.8.4 The radiometer server program 26

5 Packaging and screening **27**
5.1 The analog section 27
5.2 The digital section and the real-time computer 27
5.2.1 Cooling the digital backend 28
5.3 External cabling 28

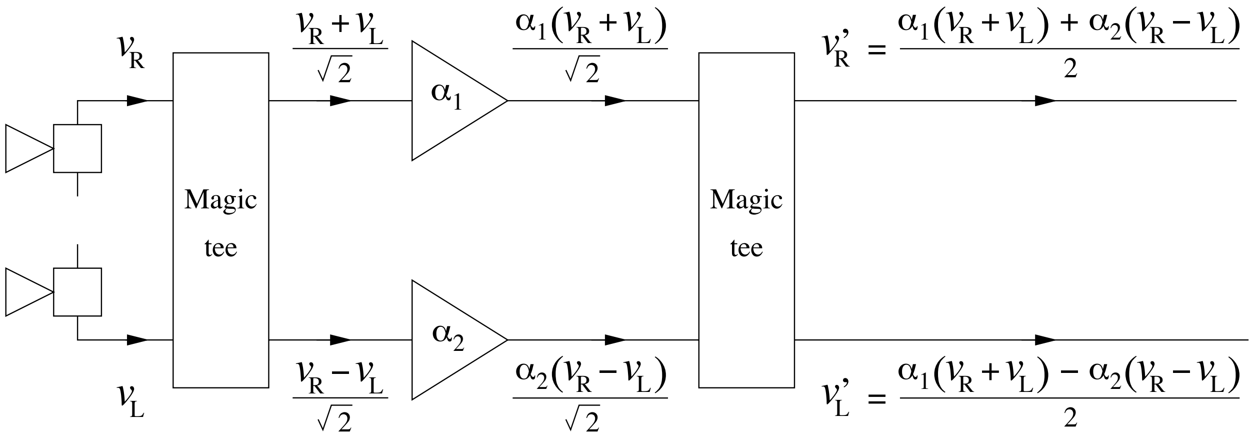
List of Figures

1 The effect of the magic tees 5
2 The control signals of the 180° phase switches in the front-end. 12
3 A rough design of the internals of the backend FPGA and the A/D board 22

1 Introduction

The preliminary design for the 1cm front-end, described in Watts et. al. 2002 [1], closely mimicked the one proposed by Padin [2], which was in turn based on the design of the receivers for the MAP satellite [3]. To provide a concrete model on which to base our discussions of the backend, it will be useful to start by describing the design of the MAP receivers.

The MAP frontend consists of two horns pointing at widely separated parts of the sky, each one measuring both L and R polarizations.



v_R
v_L
Magic tee
$\frac{v_R+v_L}{\sqrt{2}}$
$\frac{v_R-v_L}{\sqrt{2}}$
α_1
α_2
$\frac{\alpha_1(v_R+v_L)}{\sqrt{2}}$
$\frac{\alpha_2(v_R-v_L)}{\sqrt{2}}$
Magic tee
$v_R' = \frac{\alpha_1(v_R+v_L)+\alpha_2(v_R-v_L)}{2}$
$v_L' = \frac{\alpha_1(v_R+v_L)-\alpha_2(v_R-v_L)}{2}$

Figure 1: The effect of the magic tees

As indicated in figure 1, the L polarization, v_L, of one horn is then combined with the R polarization, v_R, of the other horn, in a magic tee, the two outputs of which are proportional to the sum (v_R+v_L) and difference (v_R-v_L) of these inputs. The sum and difference signals are then separately amplified by twin amplifier chains, before being recombined in another magic tee to recover amplified versions, v_L' and v_R', of the original L and R signals. These are then separately detected with square law detectors and subtracted, yielding $v_L'^2 - v_R'^2$. The advantage of using magic tees in this manner is that if the two input signals are of similar strength, the effects of gain variations between the magic tees, on $v_L'^2 - v_R'^2$ are significantly reduced. This can be demonstrated as follows.

If, as shown in figure 1, we use α_1 and α_2 to denote the gains of the two amplifier chains between the magic tees, then the outputs of the second magic tee, v_L' and v_R', are given by,

$$v_R' = \frac{\alpha_1(v_R+v_L)+\alpha_2(v_R-v_L)}{2} \tag{1}$$

$$v_L' = \frac{\alpha_1(v_R+v_L)-\alpha_2(v_R-v_L)}{2} \tag{2}$$

These signals are subsequently square-law detected and differenced, so the resulting output is given by squaring and subtracting these signals.

$$v_{\mathrm{R}}'^2 - v_{\mathrm{L}}'^2 = \alpha_1\alpha_2(v_{\mathrm{R}}^2 - v_{\mathrm{L}}^2) \tag{3}$$

If we assume that gain variations in the two amplifier chains can be represented as Gaussian fluctuations, then the sensitivity of this system to gain fluctuations can be found by propagation of errors.

$$\sigma^2_{(v_{\mathrm{R}}'^2 - v_{\mathrm{L}}'^2)} = (\alpha_2^2\sigma_{\alpha_1}^2 + \alpha_1^2\sigma_{\alpha_2}^2)(v_{\mathrm{R}}^2 - v_{\mathrm{L}}^2)^2 \tag{4}$$

If we do the same calculations for a hypothetical receiver in which the two signals are amplified by completely separate amplifier chains, without any magic tees to share the signals between the two chains, then the equivalent results are as follows.

$$v_{\mathrm{R}}'^2 - v_{\mathrm{L}}'^2 = \alpha_1^2 v_{\mathrm{R}}^2 - \alpha_2^2 v_{\mathrm{L}}^2 \tag{5}$$

$$\sigma^2_{(v_{\mathrm{R}}'^2 - v_{\mathrm{L}}'^2)} = 4(\alpha_1^2\sigma_{\alpha_1}^2 v_{\mathrm{R}}^4 + \alpha_2^2\sigma_{\alpha_2}^2 v_{\mathrm{L}}^4) \tag{6}$$

By comparing these equations, one can see that provided that the difference between the signals v_{R} and v_{L} is much smaller than their individual magnitudes, using magic tees in this way greatly reduces the sensitivity of the receiver to gain variations, compared to a receiver that doesn't. Given that, in practice, the magnitudes of the noise temperatures of the two receivers will be much greater than their differences, and given that astronomical signals where low noise is most needed will be insignificant compared to the receiver noise temperatures, gain variations are heavily suppressed in the MAP receivers.

While the described use of magic tees suppresses gain variations introduced by components between them, it doesn't suppress gain variations introduced by the square-law detectors, the integrators, or the A/D converters which follow the magic tees. To suppress these variations, the MAP receivers include a phase shifter in one of the signal paths between the magic tees. This is rapidly switched by a 2.5kHz square wave between 0° and 180°, which has the effect of making the outputs of the second magic tee alternate between v_{R}' and v_{L}'. These signals are then passed through square-law detectors, integrated to cater to the sampling time limitations of the A/D converters, and finally digitized. If we label the two magic tee outputs as v_a and v_b, then when the phase shifter is at 0°, the two outputs are as follows.

$$\begin{aligned} v_{a(0^\circ)} &= \frac{\alpha_1(v_{\mathrm{R}} + v_{\mathrm{L}}) + \alpha_2(v_{\mathrm{R}} - v_{\mathrm{L}})}{2} && (7) \\ v_{b(0^\circ)} &= \frac{\alpha_1(v_{\mathrm{R}} + v_{\mathrm{L}}) - \alpha_2(v_{\mathrm{R}} - v_{\mathrm{L}})}{2} && (8) \end{aligned}$$

Assuming that the phase shifter is in the upper signal path of figure 1, then when this phase shifter is switched to $180°$, the two outputs change to the following.

$$
\begin{aligned}
v_{a(180°)} &= \frac{-\alpha_1(v_{\mathrm{R}}+v_{\mathrm{L}})+\alpha_2(v_{\mathrm{R}}-v_{\mathrm{L}})}{2} && (9)\\
v_{b(180°)} &= \frac{-\alpha_1(v_{\mathrm{R}}+v_{\mathrm{L}})-\alpha_2(v_{\mathrm{R}}-v_{\mathrm{L}})}{2} && (10)
\end{aligned}
$$

If we denote the gains of the analog electronics that follow these outputs as β_a and β_b, respectively, then after square-law detection and integration over their respective phase-switch states, the outputs become:

$$
\begin{aligned}
\left\langle |v_{a(0°)}|^2 \right\rangle &= \frac{\beta_a}{4}\left\langle |\alpha_1(v_{\mathrm{R}}+v_{\mathrm{L}})+\alpha_2(v_{\mathrm{R}}-v_{\mathrm{L}})|^2 \right\rangle && (11)\\
\left\langle |v_{b(0°)}|^2 \right\rangle &= \frac{\beta_b}{4}\left\langle |\alpha_1(v_{\mathrm{R}}+v_{\mathrm{L}})-\alpha_2(v_{\mathrm{R}}-v_{\mathrm{L}})|^2 \right\rangle && (12)\\
\left\langle |v_{a(180°)}|^2 \right\rangle &= \frac{\beta_a}{4}\left\langle |\alpha_1(v_{\mathrm{R}}+v_{\mathrm{L}})-\alpha_2(v_{\mathrm{R}}-v_{\mathrm{L}})|^2 \right\rangle && (13)\\
\left\langle |v_{b(180°)}|^2 \right\rangle &= \frac{\beta_b}{4}\left\langle |\alpha_1(v_{\mathrm{R}}+v_{\mathrm{L}})+\alpha_2(v_{\mathrm{R}}-v_{\mathrm{L}})|^2 \right\rangle && (14)
\end{aligned}
$$

Phase switch demodulation in the backend then involves forming the following sums.

$$
\begin{array}{lllll}
v_{\mathrm{R}}^{\phi 2} & = & \langle |v_{a(0°)}|^2\rangle + \langle |v_{b(180°)}|^2\rangle & = & \frac{(\beta_a+\beta_b)}{4}\langle |\alpha_1(v_{\mathrm{R}}+v_{\mathrm{L}})+\alpha_2(v_{\mathrm{R}}-v_{\mathrm{L}})|^2\rangle \\
v_{\mathrm{L}}^{\phi 2} & = & \langle |v_{a(180°)}|^2\rangle + \langle |v_{b(0°)}|^2\rangle & = & \frac{(\beta_a+\beta_b)}{4}\langle |\alpha_1(v_{\mathrm{R}}+v_{\mathrm{L}})-\alpha_2(v_{\mathrm{R}}-v_{\mathrm{L}})|^2\rangle
\end{array} \qquad (15)
$$

Finally, on subtraction of these two outputs, the equivalent of equation 3 becomes.

$$
v_{\mathrm{R}}^{\phi 2} - v_{\mathrm{L}}^{\phi 2} = (\beta_a+\beta_b)\alpha_1\alpha_2(\langle |v_{\mathrm{R}}|^2\rangle - \langle |v_{\mathrm{L}}|^2\rangle) \qquad (16)
$$

While the form of this equation implies that phase-switching has the same benefits as those brought by the magic tees, note that this equation is only correct if $v_{\mathrm{R}}^2 - v_{\mathrm{L}}^2$ remains constant during both states of the phase switch. In other words, the only variations in β_a and β_b that are suppressed are those that have frequencies lower than the phase-switch frequency. Because of this, it is desirable to use as high a phase-switch frequency as possible. However in order to be able to do phase-switch demodulation after digitization, the sampling interval of the A/D converter needs to be an integral sub-multiple of one half period of the phase-switch. The A/D converter readout interval thus sets an upper limit to the frequency of the phase switch. Fortunately gain variations tend to have a $1/f$ spectrum, so a relatively low frequency phase-switch is adequate to suppress the worst of the gain variations.

2 Green Bank specifics

2.1 The inputs of the magic tees

Although it doesn't have any impact on the design of the digital backend, it is worth noting that while in the MAP receiver the two inputs to the magic tees are the `R` signal of one horn and the `L` signal of the other horn, the factor which determines whether to use this combination, or whether to pair like polarizations from the two horns, is which combination of polarizations has the lower crosstalk. As such, the GBT receiver may differ from the MAP receiver in this respect.

2.2 The basic integration time

The square-law detectors generate voltages that are proportional to the sum of the equivalent noise temperature of the receiver, and the antenna temperature generated by the target source, so the resolution of the A/D can conveniently be expressed in K/bit. For the weakest signals, we need to ensure that the A/D converter has sufficient resolution that the measurement noise in the no-signal case, as determined by the receiver noise temperature, the measurement bandwidth and the integration time, is sampled by at least 2 to 3 bits. If we parameterize this by saying that the receiver noise should be sampled by at least q bits, that the measurement bandwidth be denoted by Δf, the integration time by τ, and the receiver noise temperature by T_{rx}, then the resolution, r of the A/D converter must be at least

$$r \geq \frac{T_{\text{rx}}}{2^q \sqrt{\Delta f \tau}} \tag{17}$$

K/bit, to adequately sample the weakest signals. For an astronomical source which generates an antenna temperature of T_{ant}, the total number of bits, n, needed in the A/D converter is thus given by

$$n \geq \log_2 \left(\frac{T_{\text{ant}} + T_{\text{rx}}}{r} \right) \tag{18}$$

which, after substitution of equation 17 for r, becomes,

$$n \geq \log_2 \left(\frac{T_{\text{ant}} + T_{\text{rx}}}{T_{\text{rx}}} \sqrt{\Delta f \tau} \right) + q \tag{19}$$

Apart from the Sun and the Moon, the strongest astronomical source at 30GHz is the Crab Nebula, which has a flux of about 385Jy [4]. From the GBT point source sensitivity graphs in the GBT Short User Guide [5], the sensitivity of the GBT at 30GHz is slated to eventually reach 1.9K/Jy.

This means that observing the Crab Nebula with the 1cm receiver could eventually generate an antenna temperature, T_{ant}, of about 730K. In comparison, the 1cm receiver noise temperature T_{rx} will probably only be about 30K. Assuming that the bandwidth Δf of a single channel will be about 3.5GHz, that we select an integration time of 100μs, and settle on $q = 3$, then the above inequality requires that we use an A/D converter with at least 17 bits of effective resolution, preferably with one or two extra bits to give some headroom for stronger signals. For the 3mm receiver, the target receiver noise temperature is 80K, the bandwidth of each channel 8GHz, and the maximum antenna temperature 700K. This translates to a requirement for 16 bits effective resolution in the A/D.

An example of an A/D converter that meets the above integration time and resolution requirements is the 20bit DDC101 made by Burr Brown. The data sheet [6] for this chip implies that at a 10kHz conversion rate, the effective resolution for small signals would be about 18.5 bits. Note that several BurrBrown delta-sigma ADCs, such as the 40kHz ADS1252, appear, at first glance, to be much faster than this chip, but in reality these chips contain a digital filter which performs a running average of the last five samples, so the actual response rate is 5 times slower than advertised. The DDC101 has other advantages than just its speed. Because its inputs are single-ended, and because it operates by integrating the input signal and digitally tracking the resulting ramp, if we use this chip, we won't need either a separate integrator circuit, a differential line driver, or a sample and hold circuit, as we otherwise would with a more traditional chip like the ADS1252. This should greatly simplify the circuit design and the layout of the PCB. There will, however need to be a buffer amplifier preceding each A/D converter to match its input impedance requirements.

Note that the output levels of the detectors need to be adjusted such that the strongest signal won't saturate the inputs of the A/D converters. Assuming that we use a 5V supply, this means that the requirement of at least 17-bits of resolution, translates to the need to suppress other sources of noise to better than $5/2^{17} \approx 40\mu$V. Suppressing 60Hz power-line noise and noise from digital switching transients to this level will be a challenge, especially if the signals from the detectors have to be transmitted over cables between the frontend receiver box and the digital-backend box. To avoid the cable issue, and move all of the low-signal analog electronics away from the digital parts of the backend, it is proposed that the A/D converters be placed on a separate board in the frontend receiver box. Since the DDC101 A/D converter chip delivers its digitized output serially, the serial outputs from the 16 A/D converters can then be connected to the digital backend by way of a simple shielded 16-conductor cable, without any worries about crosstalk or noise.

While an A/D converter with the required resolution and sampling capabilities is available, the large number of bits complicates things further down the line. Assuming that we were to keep just the 19 bits of effective resolution of the above A/D converter, this would leave just 13 bits of headroom for integration within a 32 bit number. This in turn would only allow us to integrate up to 819.2ms seconds in the hardware and the backend CPU. Further integration would require 64-bit data-types.

Note that, contrary to previous discussions, the large dynamic range required of the A/D converter wouldn't be relieved by subtracting the signals before digitization. For example, in Padin [2], it is stated that differencing before digitization "reduces the dynamic range required in the post-detector amplifiers and the A/D". This is indeed the case in the CBI and MAP, where the sensitivity is such that the receiver noise temperature always swamps the antenna temperature, regardless of what one is observing, but it isn't the case for the GBT, since its much larger dish results in antenna temperatures from strong sources that are many times larger than the receiver noise temperature. As such, even if one differences before digitization, when one of the signals to be subtracted is coming from a strong source, and the other from empty sky, the difference signal will have essentially the same magnitude as the unsubtracted signal from the strong source, so the same dynamic range is needed in the A/D converter regardless of whether differencing is done before or after digitization.

One way in which differencing before digitization would help, is that after differencing one has the option of throwing away the least significant bits, when looking at strong signals. Thus by differencing before digitization, one could in principle implement a (complicated if done in hardware) floating point scheme which, by truncating the numbers to a fixed number of significant figures of accuracy, would allow longer hardware integration times, with smaller archiving space requirements. Given the stated requirement that the paired radiometer signals be kept separate for differencing as part of the off-line analysis, this scheme isn't developed further in this proposal.

2.2.1 An alternative scheme using multiple gains

While it is clearly convenient to cover the whole of the dynamic range of the system using a single gain, in practice we may need to adopt separate small and large signal regimes, so that a faster, lower precision A/D converter can be used. The advantages of this are,

- A faster A/D converter would enable the adoption of a higher phase-switching frequency, and thus improve the rejection of higher frequency $1/f$ noise.
- Using a lower precision A/D converter would make the system more tolerant to noise in the preceding analog electronics.
- We would be using a less "cutting-edge" A/D converter, so the potential for unforeseen problems would be reduced.

An example of an A/D converter that would be suitable for this is the ADS7813, which appears to have an effective resolution of 14 bits, and a maximum conversion rate of 40kHz. The dynamic range reachable in a single range using this A/D converter can be calculated from equation 20.

$$\frac{T_{\text{max}} + T_{\text{rx}}}{T_{\text{min}} + T_{\text{rx}}} = \frac{2^{n-q}}{\sqrt{\Delta f \tau}} \tag{20}$$

Where T_{min} and T_{max} denote the minimum and maximum astronomical signals covered by the range. For $n = 14$, $q = 3$, $\Delta f = 3.5$GHz, and $\tau = 1/40$ms, this evaluates to a factor of 6.9. Thus for the highest sensitivity range of the 1cm receiver, where $T_{\text{min}} = 0$K, and $T_{\text{rx}} = 30$K, the

ADS7813 would be able to measure signals up to 180K before saturating. Allowing some overlap between this sensitivity range and the next one, a second range starting at $T_{\text{min}} = 100$K, would be able to measure signals of up to 870K before saturating the A/D converter. For the 3mm receiver, where $\Delta f = 8$GHz and $T_{\text{rx}} = 80$K, the most sensitive range would accommodate signals between 0K and 366K, and a second could be used to extend this with a range from say 150K to 973K.

This looks like a good compromise. It increases the phase switching frequency by a factor of 4 and only requires 14 bits of dynamic range in the analog electronics, instead of 17. The increase in the phase switching frequency, in particular, could be very beneficial, since it isn't clear that the slow, $4 \times 100\mu$s, phase-switching rate imposed by the DDC101 is fast enough. It appears that the serial readout rate is also faster in the ADS7813, with the data-sheet specifying a *minimum* readout clock frequency of 10MHz, compared to a *maximum* of only 8MHz in the DDC101. This would reduce the amount of integration time lost between samples.

On the negative side, separate integrators would be needed and, since the conversion of one phase switch sample would be occurring while the next one was being integrated, the phase switch timing logic would be slightly more complicated. However the ADS7813 chip appears to be a simpler chip to control than the DDC101, so these additions would probably easily be offset by much simpler support electronics.

Of course, we would also need to add an amplification stage with two gains, selected dynamically by the FPGA. However there have been suggestions that this would be desirable even if we were to stick with the DDC101.

The relative merits of using this scheme over the original proposal to use the DDC101, will be considered further during the detailed design phase, so apart from a few comparisons between the two schemes, the rest of this proposal will continue to focus on a design using the DDC101.

2.3 Phase switching

The introduction described a 2-state phase switch cycle, in which a single phase switch in one of the signal paths between the magic tees was alternated between 0 and 180°. Better rejection, however, can be achieved by having 180° phase switches in both signal paths, and adopting a 4-state cycle to go through all of their combinations. Figure 2 shows the proposed sequence of states, corresponding to paired phase-shifts of (0°, 180°), (180°, 180°), (180°, 0°) and (0°, 0°).

Note that the chosen sequence of states, results in each phase switch changing state half as often as the alternate (0°, 0°), (180°, 180°), (0°, 180°), (180°, 0°) ordering, thus halving the speed required of the phase switches. As in the 2-state cycle described earlier, after square-law detection the proposed sequence causes the two outputs of the radiometer to alternate identities at the frequency of the master phase switch clock (ie. 5kHz). The demodulation circuitry is thus the same in both cases.

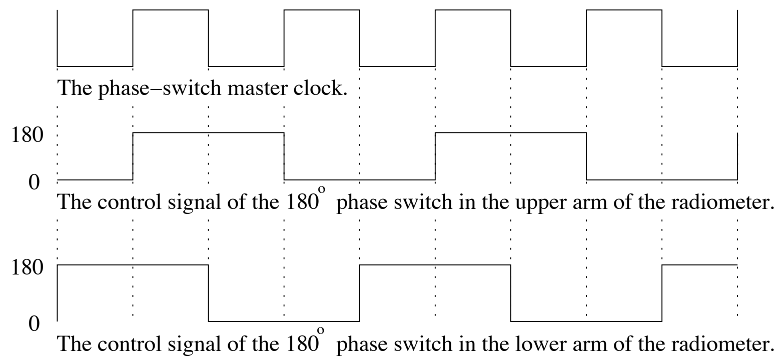

The phase–switch master clock.
180
0
The control signal of the 180° phase switch in the upper arm of the radiometer.
180
0
The control signal of the 180° phase switch in the lower arm of the radiometer.

Figure 2: The control signals of the 180° phase switches in the front-end.

2.4 Phase switch demodulation

Phase switch demodulation involves undoing the alternation of the two A/D outputs introduced by phase switching. Since the job of the phase switch is to suppress gain variations up to the output of the A/D converter, demodulation has to be performed on the digitized signals. This means that each A/D converter cycle must sample an integral sub-multiple of the duration of a single phase-switch state. Since we want the fastest phase switching frequency possible, the A/D conversion interval should simply be the same as the duration of a single phase switch state, minus switchover settling times.

For phase switching to work correctly, we need to be sure that the A/D converters integrate for exactly the same amount of time in each switch state, and that transients during the switchover from one state to the next are not sampled by the A/D. To ensure that the A/D converter samples the same amount of time in each phase-switch state, one either needs a square-wave clock from which a fixed transition period is blanked during integration before the A/D converter, or one can use a state machine which uses precise digital monostables to blank, then integrate for fixed amounts of time. The clock approach has various disadvantages. To accommodate cable and component delays between the various places in which the switching signal is needed, one would need to generate variously delayed copies of the clock signal. Tuning these delays would involve a balancing act that could be upset simply by changing the lengths of the wiring that connect the modulation, blanking and integration gating circuits. Also, if the A/D converter took slightly different times to perform each acquisition, one would have to lengthen the blanking interval to compensate for this jitter. The state-machine approach is much more robust, since it doesn't advance to the next state until the last one has signaled completion. One also doesn't need to delay any clock signals, so there is no balancing act involved. Also, since the blanking interval is performed as a separate step before the A/D integration interval, the actual integration time achieved during a single sample wouldn't be dependent on what blanking interval was chosen.

All in all, the state machine approach is much simpler to implement, more robust, and easier to debug and tune, so it is proposed that this scheme be implemented. Note that this does however preclude the use of external phase switching signals, as originally requested. Supporting arbitrary externally generated phase-switching signals would be difficult, even in the clocked approach. In particular, the whole design of the circuit is based on particular A/D conversion intervals etc, and these intervals would have to change to match the frequency of an external phase switch. It also isn't clear how the driver software would figure out the phase switching interval of an arbitrary external clock, and it would need to know this to be able to convert integration time requests from seconds to samples. It is thus requested that this requirement be dropped, and that the state machine approach be adopted, with its timing characteristics fully controlled by the real-time CPU.

2.4.1 The benefits and drawbacks of postponing differencing

It is hard to think of any astronomical benefits to postponing the differencing of the paired radiometer outputs to off-line processing. The two unsubtracted signals are of little practical use on their own, since if the amplifiers between the magic tees aren't perfectly matched, which they clearly won't be, there will always be some crosstalk between v_{R} and v_{L} in the output v'_{R} and v'_{L} signals, and by not differencing these signals, one loses the suppression of gain variations, otherwise provided by the magic tees and phase switching.

The only obvious advantage of recording both signals is that in the event of either a failure, or the degradation of the performance of one of the receiver front-ends, keeping the channels separate would allow one to pinpoint which receiver was responsible. One would imagine, however that this type of debugging could equally well be done by having the ability to individually switch on and off the front-end amplifiers, plus the ability to inject noise signals into each feed. Alternatively, if it really were necessary to be able to remotely monitor the two power levels before differencing, this could be done separately, using a pair of cheap, low precision A/D converters, used solely for monitoring, while using a single higher precision A/D converter to measure the differenced astronomical signal.

The disadvantages of postponing differencing are numerous.

- We would need twice as many A/D converters and accompanying support components.
- As mentioned earlier, more bits would have to be kept before subtraction, so that in addition to having to archive twice as many data points, the sizes of the data-types used to integrate and record each of these data-points would need to be greater.
- To get rid of detector offsets in the undifferenced radiometer outputs, one needs to subtract the outputs of two detectors in good thermal contact. This comes for free when differencing of the radiometer outputs is done in the hardware, but if one needs the undifferenced signals to be separately usable for astronomical work, then the number of detectors would need to be doubled, with each primary detector being complemented by a twin, attached to a dummy load.

In summary, the extra complexity and expense of keeping the radiometer signal pairs undifferenced seems hard to justify. However the consensus is that this requirement be kept.

2.5 Unresolved problems

2.5.1 Controlling two receivers

Another issue is how to handle both the 1cm and 3mm receivers. This could be done in a number of ways, ranging from duplicating everything, to sharing everything between the two receivers. The simplest approach is to duplicate everything, so since hardware costs aren't the dominant factor in the budget, there is no real reason not to do this.

2.5.2 Shutting down and restarting the real-time computer remotely

Finally there remain questions about whether we need to be able to shutdown the real-time computer when neither the 1cm nor the 3mm receivers are in use.

As far as how to switch the real-time CPU, and possibly the digital backend on and off remotely, switching the CPU off could easily be accomplished via the standard Linux `poweroff` command (assuming a compliant BIOS). Switching it on would either require an Ethernet network interface with wakeup-on-lan capability, or some external way to switch on the power. Note that neither of the CPU's considered below support wakeup-on-lan, and finding a board with this capability might be difficult.

The alternative, which currently seems to be the favored approach, would be to leave the backend switched on at all times, but make sure that it be well enough screened that it not interfere either with itself or with other receivers.

3 The logical components of the back end

Starting from the coarsest level, the backend will consist of the following components.

3.1 A phase-switching state machine

This component will generate the signals that are to be used for phase-switch modulation, switch-transition blanking, and phase-switch demodulation. It will execute the following steps for each state of the phase switch.

1. When all of the A/D converter ready-lines report that the previous sample has been acquired, toggle the state of the phase switch.

2. Wait for 1 transition blanking interval. This interval will be programmable in 0.1 μs steps, between 0.1 and 25.6μs, using a 1-shot implemented with a down counter and a 10MHz clock.
3. Start integrating the next sample.
4. Wait for one A/D integration interval. This interval will be programmable in 1 μs steps, between 1 and 256μs, using a 1-shot implemented with a down counter and a 1MHz clock.
5. Tell the A/D converters to acquire the integrated signals.
6. Wait for the A/D converters to say that the acquisition is complete.
7. Initiate an asynchronous readout of the A/D converters, at the same time as returning to step 1.

A control line for starting and stopping phase switching and acquisition will also be provided. When a stop is requested, this will won't be honored until the current switch cycle has completed all 4 states.

Note that reading out the A/D converters while they are measuring the next sample apparently introduces noise, so it will be necessary to postpone starting the next integration until the previous one has been read out. Reading out the 20-bit DDC101 with an 8MHz serial readout clock will thus result in the loss of 2.5μs from each 100μs integration. Similarly, reading out the 16-bit ADS7813 ADC, with a 10MHz readout clock, would involve a loss of 1.6μs. However, note that since this delay happens at the same time that the phase-switch transition blanking interval-timer is postponing the start of the next integration, and since it has a similar duration, in practice no extra electronics, and possibly no extra delay, will be needed to accommodate this constraint.

3.2 An analog integration and digitization unit

Since there are two feeds, both of which separately sample `L` and `R` polarizations, there will be 4 signal outputs from the receiver. These outputs will be further split into 4 continuum bands, so the backend will require 16 integration and digitization units. As described earlier, all of the functions needed to integrate, acquire and digitize each signal can be found in a single chip, the DDC101. With the alternative ADS7813 ADC, we would have to add integrator stages as well. If multiple gain ranges are needed, then extra amplification stages preceding the integrators, plus associated control electronics, will be needed to accommodate this.

3.3 A phase-switch demodulator

The effect of the phase-switch modulator is to alternate the `L` and `R` signals between the two A/D outputs of each signal pair, immediately after a new sample is acquired by the A/D converters.

The demodulator uses switches to unscramble this change of identity. To reduce the number of switches needed, it makes sense to switch the serial output signals of the A/D converters before they are converted from serial to parallel. Since these are digital signals, the switches will be trivial to implement using logic gates.

3.4 A serial to parallel conversion unit

The phase-switch demodulated serial outputs from the A/D converters will be converted to parallel 20-bit integers using shift registers, clocked by the same clock that is sent to the A/D converters. It is proposed that the serial output be read with an 8MHz clock. This clock should be located on the A/D converter board in the receiver crate, so that it undergoes the same cabling delays as the serial data. This is necessary to ensure that the shift registers operate in sync with the serial signal.

3.5 A digital integration unit

Following serial to parallel conversion, this unit will co-add a configurable number of A/D samples in 32 bit unsigned registers. Every time that the specified number of samples have been co-added, the contents of the integration registers (one per input signal), will be latched into registers that can be read by the real-time CPU, an interrupt will be sent to the real-time CPU to tell it to read them, and the accumulation registers will be zeroed, ready to accumulate the next integration interval.

At the design requirements meeting, it was agreed that a minimum hardware integration time of 1ms would be adequate for any projected needs. The maximum hardware integration, discussed above, is set by the dynamic range and the use of 32-bit integers, to 819.2ms.

3.6 A real-time clock

For precise command-synchronization with the GBT control system and for accurate timestamping of data sent to the GBT monitoring system, the backend CPU needs to obtain time from the GBT master clock. To facilitate this, the GBT distributes time signals via IRIG-B. Unfortunately IRIG-B decoder cards, which usually also include GPS receivers, precision clocks, event timers and other superfluous features, are very expensive, proprietary and hard to find. For example, a search for Compact-PCI based IRIG-B decoder cards only turned up one possibility, which cost $2709. The unnecessary expense, the proprietary interfaces and drivers, and the obvious dangers of only having a single source for an essential part of the system, all point toward finding an alternative. There are 3 major possibilities:

1. At the design review, it was mentioned that the GBT staff have been thinking of feeding IRIG-B signals into the audio ports of future real-time PCs, and writing a Linux driver to

decode this. This would probably be a lot of work to develop, and debug, even if based on the existing Sun-Sparc IRIG-B driver, and would take a significant amount of CPU bandwidth, so it doesn't seem like an attractive approach.

2. The IRIG-B signal (assuming that it has already been demodulated from the 10kHz carrier) is reasonably simple to convert into a frame-start marker and a stream of binary bits, using either discrete digital electronics, or an FPGA. Feeding the stream of bits into a series of serial-to-parallel shift-registers would result in BCD numbers representing the timestamp, which could then be latched into a digital I/O board. The hardware would then send an interrupt to the real-time CPU, which would be expected to read them before the start of the next frame. The real-time CPU would then use these numbers to condition the CPU clock. Since the CPU would only be interrupted once per second, this would place a lot less load on the CPU than the audio driver option, mentioned above, and would be much easier to build and test.

3. Instead of using the IRIG-B signal, simply use NTP over the Ethernet to synchronize the CPU clock to the control and monitoring computer clock, which is presumably locked to the master clock via IRIG-B. This would provide ample precision for timestamping monitor data. For precise command synchronization, it could be used along with a bus interrupt generated by the site 1PPS signal.

The NTP+1PPS-interrupt solution is clearly the simplest one, and given that very mature server and client NTP software is freely available over the Internet for just about any operating system, including Linux, and that the hardware needed to generate an interrupt from the site signal is trivial, it clearly makes much more sense to adopt this than to either spend time developing custom audio IRIG-B decoder drivers, developing custom IRIG-B hardware and clock conditioning software, or spending a lot of money on a commercial IRIG-B decoder, which would also need a special driver and clock conditioning software. It is thus proposed that NTP and a 1PPS interrupt be used for time synchronization.

3.7 A backend control and data acquisition interface

As already discussed, the minimum hardware integration time will be 1ms. Given 4 32-bit outputs per band, and 4 bands, this translates to a maximum data rate of 64kB per second being communicated from the backend hardware to the real-time CPU. A much lower data rate will be needed for sending commands to the hardware from the real-time CPU.

Thus we need to find a way to interface the hardware of the backend to the real-time CPU, with a bandwidth of at least 64kB per second, the ability to generate an interrupt to tell the CPU when a new integration is available to be read, and the ability to write commands as well as read data.

This is theoretically possible with a fast serial port, but would heavily load the CPU down with interrupts, leaving it with little capacity to do anything else. USB would be much better, but the old Linux kernel recommended for use by RTLinux, predates the introduction of USB support in Linux. Using USB would also unnecessarily complicate the hardware of the backend. This leaves 3 obvious possibilities.

1. Use an EPP enabled parallel port [7]. Standard EPP capable parallel ports, which are included in most recent PCs, are capable of data rates of between 500kB and 2MB per second. All EPP parallel ports feature hardware coordinated single-byte read,write and address I/O cycles, without the need for software handshaking. An EPP capable parallel port thus implements an easy to use 8-bit wide I/O bus, without the need for any special hardware. All one does is use standard Intel INB or OUTB assembly-language instructions to read or write bytes to one of 2 PC I/O ports (one for data I/O and one for address I/O), and the interface hardware does the rest. Some EPP interface hardware is also capable of translating 16 and 32-bit reads and writes into 4 fast single-byte I/O operations, which reduces the amount of work that the host CPU has to do, and increases the throughput. Some single-board computers also come with two built-in EPP parallel ports, so if needed, one could separate the backend-control and data-acquisition between two ports. One can also buy PCI boards with extra parallel ports.

2. When choosing the FPGA (aka PLD) that will be used by the backend to do hardware integration and phase-switch demodulation, we could select one which comes with PCI interfacing capability built in. Many of Altera's chips include this, although it isn't clear how much this costs. This would provide a wider bandwidth communication channel than the EPP solution, but it isn't clear yet whether this would increase the price, how much added complication it would add to the FPGA programming, and how much it would reduce the number of gates left over to implement the remainder of the backend logic.

3. We could use an off-the-shelf PCI parallel I/O board. While many boards are available, few of them, apart from a selection of PC/104 ISA bus boards which emulate the old 8255 programmable peripheral interface standard, agree in the way that they are programmed, and would thus lock us to a single vendor. Using the PC/104 boards mentioned above would similarly lock us to one single-board computer architecture and make us reliant on the universally disliked ISA bus.

4. We could use a FPGA development board that would interface directly to the ISA or PCI bus. This would have the advantage that we wouldn't have to design our own PCB for the FPGA, and we would be able to talk to it directly over the bus. This is a new option since the first version of this proposal, and there hasn't been time to evaluate it yet. A potential drawback is that it would lock us to a specific supplier, since the protocol for programming these boards is bound to differ from one vendor to the next.

Since it involves no extra hardware, and easily meets the minimum requirements, it is proposed that the EPP parallel port solution be adopted. If we assume the use of a single parallel port for this, then this would involve a single device driver, responsible for coordinating control, and data acquisition transactions. There would be three types of transaction.

1. Command transactions.

   To ease the programming of the FPGA, commands would be sent using a postfix ordering. In other words, zero or more single-byte command arguments would be sent before the single-byte command identifier. The arguments would be sent using data-bus writes, whereas the command identifier would be sent using an address-bus write. In this way the FPGA would be able to execute each command as soon as its identifier was sent, without having to postpone execution until the associated number of arguments had been received. The arguments would be accumulated in a last-in-first-out (LIFO) buffer, such that the command could pick up just the last N arguments that it expected, ignoring the rest of the contents of the buffer. In addition to simplifying the programming of the FPGA, using postfix ordering and a LIFO buffer, has the advantage that an incomplete command can be abandoned, and a new one started at any time, without having to clear stale information from the argument buffer. This scheme is thus self synchronizing, which will make restarting after a CPU reboot trivial.

   The input buffer would be implemented in the FPGA using 8 rows of shift registers, each shift register having NMAX bits, where NMAX would be the maximum number of arguments needed by any command. Each row would hold one bit of each 8-bit argument. Thus each shift-register clock cycle would move one byte at a time. In this way, each byte read would become stacked in the buffer. On receiving the command identifier byte, the FPGA could then directly readout the arguments from the latches of the shift registers.

2. Data readout transactions.

   Data readout transactions would be used to transfer 64 bytes to the CPU at the end of each hardware integration sample period (1ms – 819.2ms). These bytes would encompass the 32-bit integers coming from each of the 4 outputs of the 4 frequency bands of the radiometer, and would be the only data traveling from the backend FPGA to the real-time CPU. At the end of each integration, the FPGA would check whether the previous frame of 64 bytes had been read out, or canceled, and if so, latch the new integration samples into a 64-byte output buffer, then use the parallel port interrupt line to signal to the CPU that a new frame of 64 bytes was available to be read. The CPU would then read the 64 bytes, one at a time. If the previous 64 bytes hadn't been read out, by the time that a new integration had started, the FPGA would simply discard the latest integration samples, leaving the incompletely read frame in the output buffer, to continue to be read by the CPU. The FPGA would store the data values in an output queue, implemented using 8 rows of 64-bit shift registers (one row per bit of each byte). The shift registers would be jam loaded at the end of each compound integration, then on receiving a data-read request, the FPGA would gather the current last bit of each of the 8 shift register lines to form a data byte, present this on the parallel port, then clock the shift registers to get the next byte, ready to be presented on the next data-read request.

3. Interrupt transactions.

   On receiving an interrupt, the driver would initiate a read of an address byte, and the FPGA would respond by sending an 8-bit bitmask indicating the origins of all interrupts received since the previous time that this byte was read, then clear its internal copy of this bit-mask.

This would allow interrupt sharing, not only between different functions in the FPGA (see below) but also with other arbitrary hardware.

Note that the FPGA would have to allow the interleaving of readout and command transactions. This would allow the FPGA to be reset, or an ongoing readout to be canceled at any time by the driver. An example of where this would be useful is at the start of a new GBT scan. This has to be synchronized with the GBT 1-second tick, so it is essential to be able to tell the FPGA to discard the current integration and immediately start a new integration at any time.

The following is a list of possible commands.

1. Reload the FPGA program from the accompanying EPROM.
2. Stop acquiring.
3. Start acquiring.
4. Specify the number of A/D samples to integrate.
5. Specify whether or not to phase-switch.
6. Specify the phase-switch transition blanking interval.
7. Specify the phase-switching period.
8. Switch on the cal signal at the start of the next integration.
9. Switch off the cal signal at the start of the next integration.
10. Specify a common gain for all of the buffer amplifiers that precede the analog integrators, choosing between 2 or 3 fixed gains.

4 The physical components of the backend

The previous section described the logical functions of various parts of the backend, but only made peripheral mention to what types of hardware would be used to implement this. In order to be able to estimate the prices of the various components, this section gives a list of possible parts, along with sample prices.

4.1 The A/D converters

As already mentioned previously, Burr Brown's 20bit DDC101 [6] fits our requirements. Insight electronics [9] lists the price of the DDC101U as $31.48 each for quantities of 1, so for the 16 that would be needed in the backend, this would come to just over $500. The alternative ADS7813 costs between $37 and $44 each, depending on the type of package selected, so for 16, this would cost between $600 and $700.

4.2 The FPGA

The backend FPGA would be responsible for all of the backend electronics, except for the contents of the separate A/D card.

Apparently for the one or two units that we would need, one can usually get free samples, but for the purposes of having a worst case price estimate, it is proposed that a $400 Altera chip be used.

A rough-out of a possible unclocked design is given in figure 3. This also shows the A/D board.

4.3 The real-time CPU

There are a multitude of choices of PC-style single-board embedded computers available, in a number of standard and non-standard sizes. The requirements for the backend are the following.

- At least 1 serial port for use as a remote console, should the need arise.
- At least 1 EPP-capable parallel port for communication with the backend electronics.
- An Ethernet port. As far as speed is concerned, 10-baseT would be sufficient. A transceiver would be required to connect this to the Green Bank fiber optic Ethernet.
- A reasonably fast CPU, say 200MHz plus. A Pentium class machine should not be needed, but probably isn't too expensive to consider.
- A PCI expansion bus. Currently no expansion cards are envisaged, but it makes sense to plan for this, just in case.
- A small footprint, in a rugged, easily mountable board.
- A way to boot the system and load the backend software. This could be any of the following.
    1. An EIDE interface with a magnetic hard disk.
    2. An EIDE interface with a flash disk (these cost about $1 per MB).

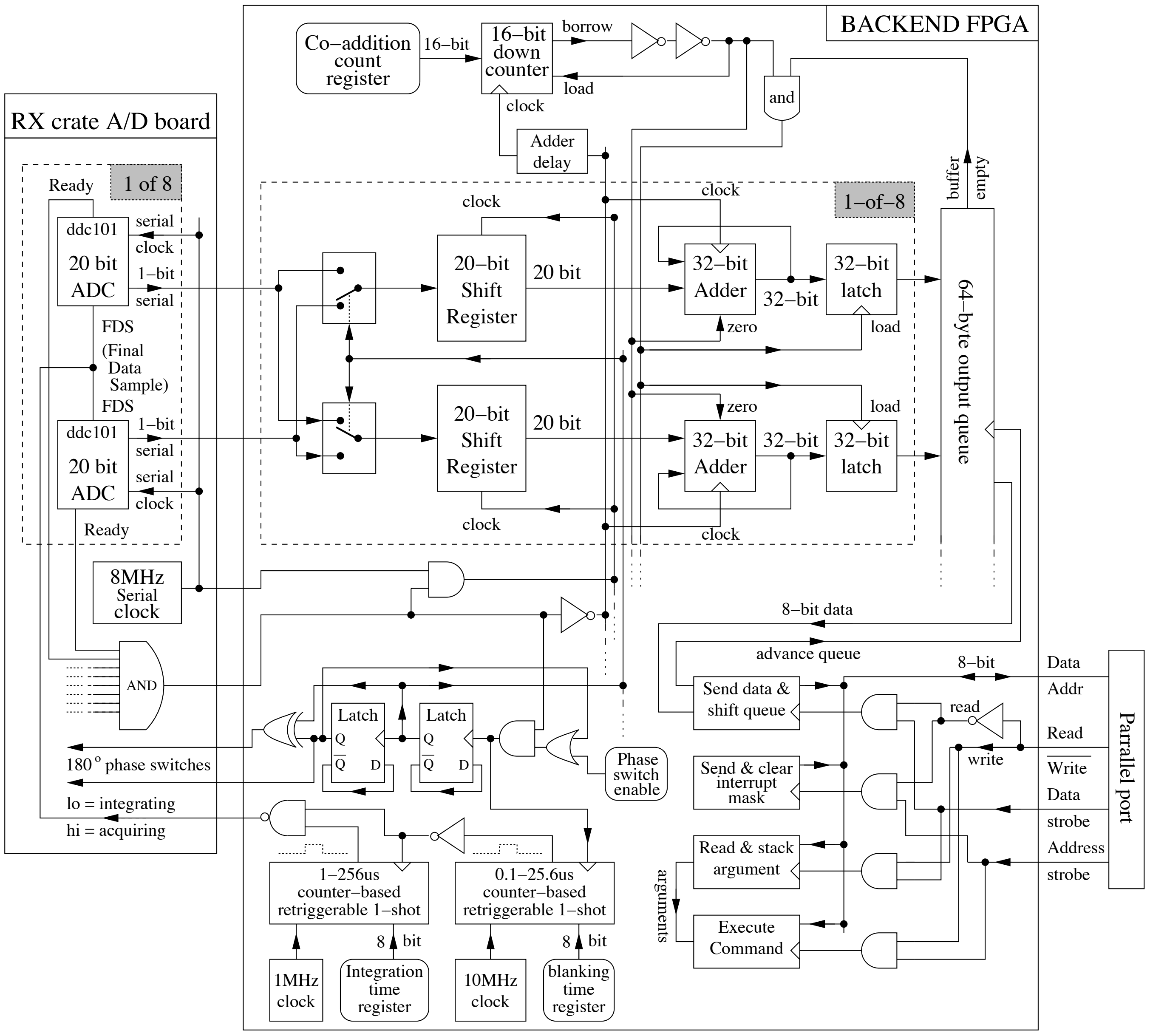


Figure 3: A rough design of the internals of the backend FPGA and the A/D board

3. A floppy disk drive for a floppy containing a heavily cut down version of Linux, plus the backend software.
4. A compact flash socket, plus a compact flash chip containing a network boot loader. The operating system and the backend software would then reside on an external Green Bank workstation.

- Enough RAM to run Linux without swapping. 128MB should be sufficient.
- A real-time clock.
- Linux drivers for all interfaces of interest.
- An industry standard form factor, to avoid dependence on a single manufacturer. The following are the main possibilities.

    1. Compact PCI.

       This requires a card-cage with a backplane. This is quite expensive and bulky.
    2. PC/104+

       PC104+ boards are just 3.55”$\times$3.775” in size, and PCI or ISA expansion cards stack vertically, thus avoiding the need for a backplane. The biggest advantage and drawback of this standard is its tiny size. The disadvantage of the size is that if one wants to build customized expansion cards, it can be difficult to fit much on a card. In our case this shouldn’t be a problem, since the backend electronics won’t be accessed via the PCI or ISA busses, and can thus be mounted separately from the CPU.
    3. EBX

       The EBX standard is compatible with all PC/104+ and PC/104 cards, but also allows the use of larger cards. EBX standard boards are 5.75”$\times$8.00” in size. The larger size makes it possible for manufacturers of single board computers to use cheaper components, while fitting more features on their boards.

An example of a single board computer that satisfies most of the above requirements is the VNS-786[11] from JUMPtec Adastra. This is a standard EBX card, with a PC104+ ISA/PCI expansion bus. It hosts 4 serial ports, 2 EPP parallel ports, 10/100 Base-T Ethernet, either a Tillamook 266MHz processor, or an AMD K6-III 400MHz CPU, up to 3 PCI expansion cards, 2 EIDE ports, a floppy disk controller, a socket for flash or DiskOnChip chips, a battery backed real-time clock, up to 256MB SDRAM, and many other features that we don’t need, such as USB. Choosing the configuration with the Tillamook 266MHz processor and 128MB of RAM, the price comes to about $600 [10]. The pcnet32 Ethernet driver that comes with RedHat, supposedly supports the Ethernet chip on this card. Unfortunately the VGA video interface, and the sound interface aren’t listed as supported. This could be a problem if it was decided that IRIG-B had to be decoded from the audio port. The lack of a driver for the VGA interface probably wouldn’t be a problem, since the system will run headless, and Linux installation can be performed in text mode, which doesn’t need any graphics drivers.

Another possibility is the range of EBX and PC104+ single board computers available from VersaLogic corporation [12]. These are more expensive, costing between $900 and $1200, but there is more choice, they claim full RTLinux compliance, and they sell most of the complementary parts, such as enclosures, adapters and cables for floppy drives and hard drives, all designed to work with their cards. A minor problem with VersaLogic's cards is that they only come with a single EPP parallel port. If it were decided to use a single real-time CPU to control two backends, this would mandate the need to buy an EPP expansion card for this particular CPU.

4.4 An enclosure and power supply for the real-time CPU

In principle the single board computer could be bolted to the inside of the backend enclosure and share a power supply with the backend electronics, but to reduce the amount of work that we have to do, and help prevent interference from the computer from reaching the low signal parts of the backend, it makes more sense to place it inside an off-the-shelf enclosure, complete with its own power supply, and bolt this enclosure inside the backend enclosure.

VersaLogic corporation [12] sells a PC104+ and EBX-compatible enclosure, 200W power-supply, and floppy drive for a total of $300. Note that although the chassis has room for an EBX card, there is only room to stack PC104 and PC104+ cards above this, due to the space taken by the power supply and floppy drive.

4.5 A hard disk drive

The simplest way to boot the system would be from a hard disk. The enclosure and single board computers discussed above all support up to two EIDE 3.5" disk drives. 20GB drives can currently be obtained for about $50 from many sources, and we shouldn't need more than this.

Since the first version of this proposal was submitted, it has been suggested that a solid-state disk would be a better choice. This costs a lot more, but it needs less power, generates less heat, and is more robust. It is proposed that a 1GB drive be bought. This will cost around $1000.

4.6 A 1PPS interrupter

The CPU needs to get an interrupt every second from the Green Bank one second tick. A simple way to do this, would be to share the parallel port interrupt between the 1PPS and the data-readout interrupt. As described earlier, the interrupt mask returned by the FPGA on receiving a request to read an address over the parallel port, would indicate which of the interrupt sources was responsible for each interrupt. The 1-second tick would thus be routed through the FPGA.

4.7 Power supplies

The DDC101 A/D converters mentioned previously, require both positive and negative 5V supplies, and draw up to 100mA from each supply rail. The alternative ADS7813 A/D converters only require a single 5V supply, and have a maximum power dissipation of just 35mW. Since larger supply voltages may be needed for op-amps preceding the A/D converters, it probably makes more sense to buy +12V and -12V supplies, and then use these to supply small +5V and -5V regulators for the individual A/D chips. This would provide the side benefit of isolating the low-signal parts of the board from any interference generated on the power-supply rails by digital transitions in the A/D converters. Thus 3A, +12V and 3A, -12V supplies should be sufficient. Since switch-mode power supplies usually generate a lot of interference, linear supplies should be used for the analog electronics. Apparently, the receiver room already has power supplies for analog electronics, so the backend won't have to provide these.

Given the low frequency requirements of the digital backend, very little current will be drawn by the FPGA. Thus a single +5V 1A power supply should be more than sufficient for the FPGA and its EPROM. If this is housed in the front-end receiver box, a linear supply would probably again be desirable, to help reduce interference leaking into the analog electronics via the case ground. Alternatively, if mounted on a card in the CPU enclosure, it could use the CPU power supply, and a separate supply would not be needed.

CPU enclosures, such as the EBX development enclosure [13] from Versalogic corporation, include an appropriate power supply, so we might not have to provide this ourselves. If such an enclosure is not used, an off-the-shelf 200W ATX PC power supply could be used to power both the CPU board and the FPGA board.

4.8 The backend software

4.8.1 Kernel space drivers

The only custom kernel driver would be the one that interacted with the parallel port. The EPP parallel port interface is easy to use, so writing this driver should not be very difficult.

Since the 1PPS interrupt will be caught by the parallel port driver, it makes sense that this driver also be responsible for synchronizing commands that need to be sent to the backend FPGA on a given 1PPS tick. This will be easiest to do if commands are passed to the driver using ioctl() calls on a file descriptor attached to the driver device file, rather than using write() to send a serial stream of bytes. This will allow commands and their parameters to be passed atomically to the driver. Data readout, on the other hand will be performed by simply reading from the file descriptor. A fixed message size will be used, with each new read always returning the start of a new message, containing the timestamp of the sample, followed by as many of the 16 data words as will fit in the provided buffer.

4.8.2 User space software

4.8.3 Network Time Protocol client software

A mature Linux NTP client program for synchronizing the clock of the real-time CPU with the Green Bank master clock is readily available for free from the Internet. The RedHat Linux distribution includes this program by default.

4.8.4 The radiometer server program

The digital backend software will be developed at Caltech, where the GBT control system won't be available for testing, where experience with "Ygor" is lacking, and where the programming language used and the associated development tools are different than at Green Bank. As such, it is proposed that the task of controlling the backend be partitioned between an interface program running Ygor classes, and a low-level radiometer server part written in C. These two programs will communicate with each other via a TCP/IP connection. Advantages of doing this include:

- This partitioning forces both sides to agree on and document, up-front, the small set of simple operations that the radiometer software must support, thus making it much more likely that the resulting software will work from the start, and making the server program much simpler to maintain.
- Assuming that a portable, language-neutral TCP/IP protocol is adopted for communications between the two programs, both groups can continue to use their favored development environments and languages, without having to learn the other's.
- Assuming that the interface protocol uses TCP/IP, the Ygor manager can run either on the real-time CPU, or on a remote workstation. Since the manager will apparently be responsible for writing FITS files to disk, the latter may be the better choice.
- In order to test the radiometer software, Caltech will need to write test programs that can exercise the radiometer. With a simple protocol defined for talking to the radiometer server, it will be trivial for us to write a simple sockets program with a Tcl/Tk interface for this purpose. Without this partitioning, we would have to write a test program that acted like the Green Bank monitor and control system, which for us would be a much more daunting job.

It has been agreed that Green Bank staff, in consultation with with Caltech, will define the communications protocol. There are a couple of ways in which this could be done.

1. Green Bank could provide a template Ygor master program, with documented stubs for us to fill in. Caltech would then fill in these stubs with the code needed to communicate them to the radiometer server.

2. Green Bank could provide the whole Ygor program, with the server communications already built in, and the TCP/IP protocol that it used, fully documented.

Note that although the interface protocol, either function based, or TCP/IP message based, needs to be defined before the digital backend software can be started, the Ygor master won't actually be needed until Caltech has finished lab-testing the system with its own test software. As such, an added benefit of the partitioning approach is that in the short term it takes some pressure off Green Bank staff.

5 Packaging and screening

There are four logically distinct parts to the backend.

1. The analog part of the backend, consisting of a single circuit board housing the 16 A/D converters, signal conditioning and other support electronics, plus external power supplies.
2. The digital part of the backend, consisting of a single circuit board, housing the FPGA, an EPROM, and an external power supply.
3. The real-time computer, power-supply, hard disk drive etc.
4. External cabling between the above parts.

5.1 The analog section

The analog part of the backend would go in the receiver front-end enclosure, and share its analog power supply. Assuming that each of the 16 radiometer channels requires 1 A/D converter, 2 op-amps, plus associated components, and that these together take 2"$\times$2" of board area, then we would need a total board area of 64 inch^2, corresponding, for example, to a board of 8"$\times$8". Note that this is bigger than the specification in the original version of this proposal. This reflects the need to accommodate a variable gain frontend, and the possibility of needing separate integrators.

5.2 The digital section and the real-time computer

It is proposed that the digital part of the backend, including the FPGA board, the real-time computer, a power supply and a disk drive, be placed in a separate screened metal box. The real-time CPU and associated parts could either be mounted as separate parts within this enclosure, or the previously mentioned EBX development enclosure, containing the CPU board etc, could itself be mounted inside the larger enclosure. Mounting the computer components directly within the larger enclosure would probably be the best strategy, both for cooling reasons, and because if they were mounted separately, we could place the FPGA on an EBX form-factor card, stacked below the CPU card, and still have access to the edge of the card, to allow for a row of connectors. Adding 2" to the dimensions of each part, the space needed for this, would then be about 8"$\times$10"$\times$5" for the CPU and FPGA cards, 8"$\times$5.5"$\times$7.5" for an ATX-standard power supply, and 3"$\times$6"$\times$8" for a standard 3.5" hard disk drive.

5.2.1 Cooling the digital backend

The VNS-786 CPU board mentioned above has a power consumption of between 20 and 40 Watts, depending on which CPU is selected. A solid state disk drive consumes about 5W (note that a normal hard drive draws about 40W). The FPGA card probably wouldn't draw more than about 5W. So given that ATX power supplies are supposedly about 70% efficient, one can estimate that the total heat dissipation would be around 70W. Doubling this for safety, we should probably design to dissipate at least 150W of heat from the digital enclosure.

Within the enclosure, at least one fan should be used to circulate air. A dedicated CPU fan may be needed, depending on what type of CPU is selected. Exchanging heat with the outside of the enclosure is a more complicated problem, since open cooling vents in the enclosure would severely compromise its screening effectiveness. The cheapest solution would be to use vents shielded with EMI vent panels, such Chomeric's "CHO-CELL EMI Vent Panels" [14]. However although these supposedly have shielding effectivenesses of 90db at 10GHz, numbers for higher frequencies aren't shown, so if significant higher frequency harmonics are expected to be emitted by the computer, this filter might not be sufficient to prevent RFI from reaching the 3mm receiver.

Alternatives to consider, that don't need vents, would be liquid cold plates, thermoelectric cold plates or just plain air-cooled heatsinks, attached to the outside of the case. Which, if any of these solutions, would be suitable, and how efficiently hot air circulating inside the case would be conducted through the case, will be evaluated during the detailed design phase. As a worst-case cost estimate, a thermoelectric cooling plate that can dissipate 215W, made by TE Technology Inc [15], complete with matching power-supply and controller, costs about $1600.

5.3 External cabling

All cables entering and exiting this box would have to be screened, with plenty of ground pins connected to the grounded walls of the enclosure, via short, low-resistance shunts. External cables will include the following:

- The 120V power cable. This should be low-pass filtered at the point of entry, using an off-the shelf filter designed for this purpose.
- Optic fiber Ethernet, via an internally mounted, off-the-shelf twisted-pair to optic-fiber transceiver.

- Digital signals between the frontend and digital enclosures, including:
  - 17 wires carrying digital signals from the frontend enclosure to the digital enclosure, these being the 16 A/D serial output lines, plus the serial-line clock, all carrying signals of about 10MHz for a few microseconds at the end of each phase-switch transition. Since data won't be being taken at these moments, these signals themselves shouldn't pose an interference problem to the front-end, and adequate shielding of the cables that carry them should prevent them from interfering with other instruments.
  - Digital status signals going from the frontend to the digital backend. In the present design, this only includes the single ready-to-readout line.
  - Digital control signals going from the digital enclosure to the front-end enclosure. These will include 2 phase-switching signals, 1 noise-diode switching signal, and an acquire-data signal.

  While interference with other instruments would be avoided by transmitting these signals in shielded cable bundles, interference with the continuum receiver frontend is a potential problem, because although none of the intentionally transmitted signals are an issue, extraneous signals in the digital enclosure, especially transients on the digital ground, could easily be transmitted to the front-end box if care isn't taken. To avoid this, it is proposed that the ground lines in the cables only be attached to the analog ground of the front-end box, and that at the receiving end of each cable, an opto-isolator be used to electrically isolate the two systems and only allow signals to go in their intended directions.

References

[1] Watts, G., Norrod, R., Jewell, P. and Shelton, A *"Preliminary Design for a GBT 1 cm Receiver"*, January 2002.

[2] Padin, S. "Design Considerations for the Continuum Section", June 2001.

[3] `http://map.gsfc.nasa.gov/m_mm/ob_techradiometers.html`

[4] Mezger, P.G., Tuffs, R. J., Chini, R., Kreysa, E. and Gemünd, H.P., 1986, Astron Astrophys, **167**, pp.145-150.

[5] `http://www.nrao.edu/GBT/proposals/short_guide/introduction.html`

[6] `http://www-s.ti.com/sc/ds/ddc101.pdf`

[7] `http://www.fapo.com/eppmode.htm`

[8] `http://www-s.ti.com/sc/ds/opa627.pdf`

[9] `http://www.insight-electronics.com`

[10] `http://www.emjembedded.com`

[11] http://www.jumptecadastra.com/VNS-786.PDF

[12] http://www.versalogic.com/products

[13] http://www.versalogic.com/products/DS.asp?ProductID=33

[14] http://www.chomerics.com/products/chocell.htm

[15] http://www.tetech.com/assys/plate.shtml