LWA-OVRO Memo No. 14

Time and Synchronization for OVRO LWA352

Larry D'Addario
California Institute of Technology

2020 May 3

Submitted 2020 June 2

To cite this memo in another document, use:
"LWA-OVRO Memo 14, 2020 May 03, http://tauceti.caltech.edu/LWA/lwamemos.html "

Time and Synchronization for OVRO LWA352

Larry R. D'Addario
California Institute of Technology
2020 May 3

I. Background

For many reasons, it is desirable for the outputs of a telescope to be labeled with the times of the observations; we call this *time tagging*. It requires some sort of clock, along with a mechanism for assigning clock readings to telescope outputs. If generating the outputs involves combining multiple signals, as in a radio telescope array, and especially if that processing is distributed across multiple devices, there is also a need to *synchronize* the timing among the signals. That is, we want to ensure that the time tag assigned to an output applies in the same way to all signals, not just one of them or an average of them.

These two objectives, time tagging and synchronization, usually have very different requirements for accuracy and precision. For example, for a telescope that generates outputs every few seconds, time tagging with a precision of 1 s may be sufficient. But if it is combining signals in a band near 1 GHz, synchronization precision must be << 1 ns. For time tagging, but not for synchronization, there may be an additional requirement for accuracy with respect to an external measure of time, such as UTC; the basic requirement is with respect to time on a local clock ("telescope time"). Some telescope designs have conflated these three requirements – synchronization precision, time tagging precision with respect to telescope time, and time tagging accuracy with respect to external time – and attempted to accomplish all with a common implementation, concentrating on the last item to the detriment of the others.

When multiple devices must know something about time, the timing information must be distributed to them from the telescope clock. The information is distributed using *timing signals*. Any timing signal can be described by two fundamental parameters, its *accuracy* and it *ambiguity*. For example, we could use a 10 MHz tone as a timing signal. If its signal-to-noise ratio is such a receiver can measure its phase to .01 cycle, then it has an accuracy of 1 ns; and, since each cycle is indistinguishable from the next, it has an ambiguity of 100 ns. Or consider transmitting periodic pulses at a rate of 1 per second (1 PPS, which is a traditional choice in radio astronomy, introduced in VLBI practice more than 50 years ago). If its SNR is such that a receiver can measure the rising edge to 10 ns, then that is its accuracy; its ambiguity is 1 s. Or consider transmitting a 64b number that is proportional to time with its LSB representing 1 ns. Its ambiguity is 584 years. In principle its accuracy is 1 ns, but in practice it is difficult to build a digital receiver that can measure the number's arrival time to 1 ns, and to build a digital network whose latency is stable and known to 1 ns. The point is that there are practical limits to the ambiguity-to-accuracy ratio that can be achieved with a single signal. To achieve a higher ratio, multiple timing signals can be used in a hierarchy. A fast signal is used to obtain the required accuracy, but with low ambiguity; a slower signal then has just enough accuracy to resolve the ambiguity of the fast signal, but larger ambiguity. We continue in this way with additional signals until the desired ambiguity is achieved, retaining the first signal's accuracy. The key is that each signal must resolve the ambiguity of its predecessor, and of course all signals must be consistent with each other (all derived from the same clock). Some telescopes have been designed with fast and slow timing signals but without paying attention to these principles[1].

[1] For example, OVRO-LWA phases 1 and 2 used a 196.608 MHz tone and 1 PPS, but the latter could not distinguish one cycle of the former and there were not necessarily 196,608,000 cycles between pulses.

Tying telescope time to an external measure of time usually requires much less accuracy than synchronization. Unless the observations need to be analyzed in real time, there is no requirement at all on the agreement between time tags and external time, provided that the difference is known and can be added to the time tags during non-real-time analysis. It is convenient if the two are reasonably close, but we take the requirement to refer only to how accurately the difference is known, not to the difference itself. This leads to the idea that the telescope clock should not be driven by or connected in real time to any external clock. It is only necessary to provide a mechanism for comparing telescope time with external time. If the telescope clock is well behaved (based on a reasonably stable oscillator), then the difference measurement needs to be done only occasionally to achieve the desired accuracy. This makes it easier to ensure that timing signals derived from the telescope clock are consistent with each other and that they are free of discontinuities. A discontinuity in telescope time occurs when the telescope clock needs to be reset, typically to make it closer to external time. Such events can be exceedingly rare and carefully logged when they occur. Sometimes it can be arranged that they occur when no observations are in progress, such as during scheduled maintenance.

Besides time tagging and synchronization, there is a third purpose for which time must be known in a telescope: *real-time control*. This is very different, so it is discussed briefly here but not otherwise covered in this memo. For example, the telescope may be allocated to different users or different observing modes at different times, so the control system must know when to switch. Also, control must often be tied to the position of a target object in the sky (pointing); this usually determines the accuracy required. It requires knowing time on the same scale for which the ephemeris of the object is known, often local apparent sidereal time. For this purpose, an external measure of time like UTC is needed. The required accuracy depends on the beam size and the speed of apparent movement of the target, but is typically much looser than is needed for synchronization. In that case, it may be convenient and reasonable to drive the control system from a independent measure of UTC (like network time from a time server or a GPS receiver) rather than reading the telescope clock and adjusting for its difference from UTC.

II. Synchronization and Time Tagging for LWA352

A. Context

In LWA352, each of the 704 antenna signals will be delivered by the analog signal processing system to a digitizer. All digitizers will be located in the same rack, but they will be distributed across 176 separate chips on 44 separate printed circuit boards, and the digitized signals will be received and initially processed by 11 separate FPGAs. All digitizer boards will be driven by a common 196.608 MHz sampling clock, distributed so that there is very little phase difference among the boards ($<< 1$ ns). Simultaneous sampling among all 704 signals will be achieved by synchronization at the chip level by the chip manufacturer, at the board level by the board designer, and across the 4 boards connected to one FPGA by control signals generated by the FPGA's code during initialization. Since each FPGA's logic will be driven by the sampling clock that is passed through one of the digitizer chips on one of the digitizer boards, the FPGAs are also synchronized to a small fraction of the sampling clock period. It remains to ensure that all of the 704 simultaneous samples received by the 11 FPGAs are assigned the same time tag. How to do that will be the subject of the current section of this memo.

For many purposes, it is also necessary to account for the delay from each antenna to the corresponding digitizer though the analog signal processing system. Those delays need to be measured (calibrated) by separate processes that will not be discussed here. Compensating delays may be applied in real time by the digital signal processing system (necessary for beam

forming) or in non-real-time by subsequent processing.

We assume that the FPGAs are responsible for assigning time tags to the samples, and that at the FPGA's output these time tags are passed with the data to subsequent processing, which carries them along. Of course, not every sample needs to carry a time tag. Since the sample rate is known, tagging only one set of samples is in principle sufficient, after which the time is always known by counting samples. However, time tags are normally inserted periodically to allow checking that no samples have been missed.

To give all simultaneous samples the same time tag, each FPGA must have a copy of the telescope clock, and those clocks must be synchronized. To do that, timing signals derived from the master telescope clock must be distributed to all FPGAs synchronously.

B. Conceptual Designs

We consider several designs, but they all have the following in common. The master clock is implemented as a counter driven by the sampling clock oscillator; thus, the fastest timing signal has already been distributed in a synchronized way to all FPGAs, so we have only to distribute one or more slower timing signals. Next, the master clock is implemented within one of the FPGAs, so the slow timing signal is distributed from it to the others. Finally, only one slow timing signal is needed. Each FPGA implements either the master clock or its copy in the same way, as a counter of the sampling clock sufficiently long to achieve the desired ambiguity (as discussed further below). The timing signal is conceptually just a pulse that is precise enough to resolve one cycle of the sampling clock. (The actual signal design may be slightly more complex.) Prior to sending a pulse, the master tells all other FPGAs what value the time counter should have when that pulse is received; it does this by a separate route, such as an Ethernet packet, perhaps sent via a host computer. Sending this number could be considered the next-slower timing signal of the hierarchy.

Figure 1 shows a straightforward implementation, with one FPGA selected as the master and its timing signal distributed to the others. The distribution circuitry must be designed to preserve the rise time of the pulse and to make the delays to all destinations equal. Since the delay is not zero, it must be measured and the receiving FPGAs or the master must insert compensating delay. The distribution network has 11 outputs, not just 10, so one of them goes back to the master; thus, in principle, the master could make that delay measurement. A problem with this arrangement is that it has a single point of failure. If the master FPGA or its infrastructure fails, we are no longer able to synchronize the others. Mastership could be re-assigned to another FPGA, but then it would be necessary to move the distribution circuit's input to the new master FPGA, and that could not be done by remote control.

A different arrangement is shown in Figure 2. This is symmetrical, with the FPGAs regarded logically as forming a circle, and with each generating a timing signal that is sent to the next. Any one of them can be designated as the master, and that designation can be changed without the need to move any connections. Each of the others simply copies the timing signal from its input to its output, without registering it. It is still necessary to calibrate the delays of each of the connections, but they can be constructed to be all the same. The compensating delay is then different for each FPGA, but this can be taken into account. After traversing the loop, the signal gets back to the master, so in principle it can again measure the delay. Unfortunately, it seems to suffer the same single-point-of failure problem as Fig. 1; it is arguably worse, since now the failure of any FPGA (not just the master) breaks the loop and prevents synchronization. However, this is not quite true. If any FPGA fails, the master must be re-assigned to the next FPGA in the loop (after the failed one). It is then possible to synchronize all remaining FPGAs;

we lose only the ability to check the timing signal at the master, since it no longer gets back there.

Yet another arrangement is shown in Figure 3. This one is also symmetrical, but with each FPGA generating a timing signal that goes to its two neighbors. One of them is still designated as the master, and each of the others merely copies the received signal to its output. Again it is necessary to calibrate the delays of each of the eleven 1:2 distribution circuits, but they can be fabricated so that they are all the same. Whereas each FPGA receives two copies of the timing signal, one from each neighbor, it can check that they agree (after applying the compensating delays, which are in general different). The master also receives two copies; these have traversed the circle in opposite directions. It can in principle measure each of the two delays (which should be the same if the total number of FPGAs is odd). Now if the master FPGA fails, mastership can be assigned to any of the others by some re-programming, which can be done remotely. It is also necessary to tell each of the neighbors of the failed FPGA to ignore one of its timing signal inputs, but it still has the other one. If a non-master FPGA fails, re-assignment of mastership is not necessary because of the redundancy; we need only tell the neighbors of the failed FPGA to ignore one of their timing signal inputs.

To synchronize, each non-master FPGA should set its clock counter to the specified value on the sample clock following the timing signal's rising edge (after the compensating delay). If it was already synchronized, then this should cause no change in the counter's value. It is thus possible to check whether the FPGA was synchronized, and if not to set an error flag. It is useful to have a commandable "check only" mode in which the out-of-sync detection sets the error flag but does not reset the counter. This allows the timing signal to be sent periodically without causing glitches in case of a transient problem or noise spike. It would be possible to implement logic that resets the counter only if it is found to be out-of-sync on some number of consecutive timing pulses.

The timing signal could be designed as a 1 PPS signal, but that is probably not a good choice. One second is far more time than is needed to transmit the desired counter value over 1 GbE. If the distribution circuit contains either amplifiers or passive power dividers, which are usually AC coupled, it is difficult to have a flat response from near-DC to >1GHz, as is needed to resolve the sampling clock. A better choice would be a square wave at ~100 kHz. It has no DC component and devices with flat response from 0.1 to 1000 MHz are available. A more convenient frequency is 196 kHz, since this is obtained by dividing the sampling clock by 2^{10}. This allows 5.2 μs between rising edges, which is enough for sending the desired counter values. In fact, sending the counter values can be avoided if the system is designed to allow synchronizing only when the new counter value should be zero. It is then only necessary to send a 1-bit signal on the slow (GbE) link saying that the next rising edge occurs at zero. All of the FPGA counters can have an ambiguity interval that is ~1 s or even ~1 ms, with the slower-changing bits of the time stamps added in downstream processing, after the FPGAs. There is thus an opportunity to re-synchronize at each ambiguity interval (say, every few ms). At every timing rising edge (every 5.2 μs) it is possible to check the synchronization by making sure that the 10 LSBs of the counters are zero.

C. Some Details

In LWA352, the FPGAs will be on 11 SNAP2 boards [1]. Each of these actually has 4 SMA connectors wired to I/O pins of the FPGA. One of these (J24) is intended to be a high-speed clock input for driving the FPGA logic, but it will not be used for that purpose by us because we will use the sampling clock that is transmitted through the digitizers and gets to the

SNAP2 via FMC connectors. Another (J26) is intended to be used for a fast-edge timing signal such as 1 PPS. Both of these are input-only. The remaining two (J28 and J29) are more general-purpose and bi-directional. See [2], sheet 20. J28 is wired directly to a general-purpose I/O pin of the FPGA. J29 goes through a bi-directional buffer and then to an FPGA GPIO pin. To implement the scheme of Fig. 3, we can use J28 as the timing signal output and J26 and J29 as the timing signal inputs. The output from J28 should be connected directly to a small box (designed by us) containing a buffer and a 1:2 passive power divider.

It is difficult to use the clock input (J24) for any of the timing signals. It connects to complex circuitry intended to allow various clock frequencies to be synthesized from the input, but it can be configured to route the input signal to the FPGA more-or-less directly. However, it is AC coupled at the SMA connector, and it uses a transformer balun whose response below 4.5 MHz is unspecified.

It will be necessary to test the SNAP2 to ensure that the signals can be sent and received with sufficient precision and that the latencies in getting between the connectors and the FPGA are small and understood.

III. External Time

To measure the difference between telescope time and UTC, we need to get a signal into the master FPGA that accurately represents UTC. Unfortunately, the scheme of Fig. 3 has used up all of the SNAP2 interfaces that are convenient for this (the SMA connectors). Either we must give up some of the redundancy of that scheme or use a less-convenient interface such as one of the ZD connectors.

Figure 4 shows an approach derived from the scheme in Fig. 3. Here an external 1 PPS signal that is tied to UTC by, for example, a GPS receiver, is delivered to the master FPGA, replacing the timing signal input that previously came from one of its neighbors. This retains sufficient redundancy so that if any non-master fails, all others continue to work without change. If the master fails, any of the other FPGAs can still become the master for the purpose of synchronization, but the ability to compare telescope time with UTC is lost. This may be acceptable if the drift rate between the two is small enough that the time difference remains nearly constant until the failed FPGA can be physically replaced or the cables re-configured to supply the external 1 PPS to the new master.

IV. Discussion

It is possible for telescope time to be continuous and free of glitches during the process of replacing a failed FPGA board provided that power and the sampling clock are continuously provided to the remaining FPGAs. [Explain this further...]

[Other discussion points...]

References

[1] SNAP2 document.
[2] SNAP2 schematic

FIGURES: Each figure shows just 5 SNAP2 FPGA boards for simplicity, even though there will be 11 in LWA352. All arrangements work in the same way for any number of SNAP2 boards. Figure 1 shows distribution of the 192.608 MHz sampling clock to all digitizer boards; this is omitted from the other figures for clarity, but it is the same for all of them.

Figure 1

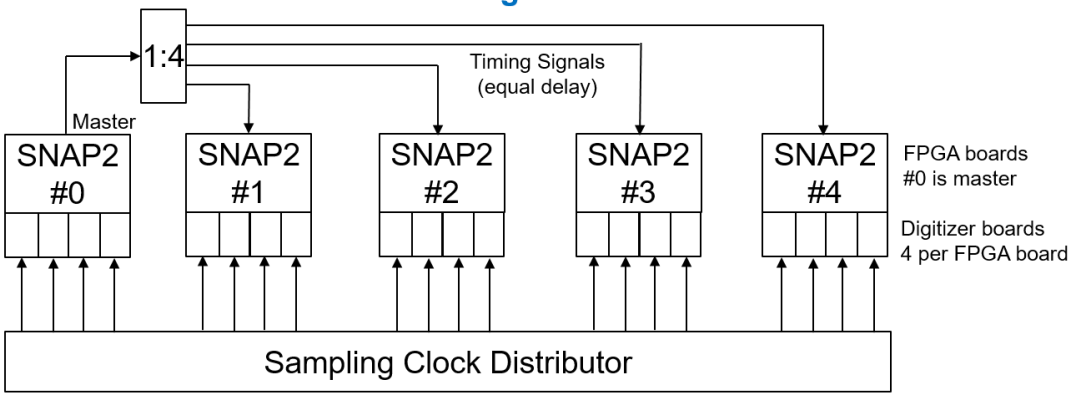


Figure 2

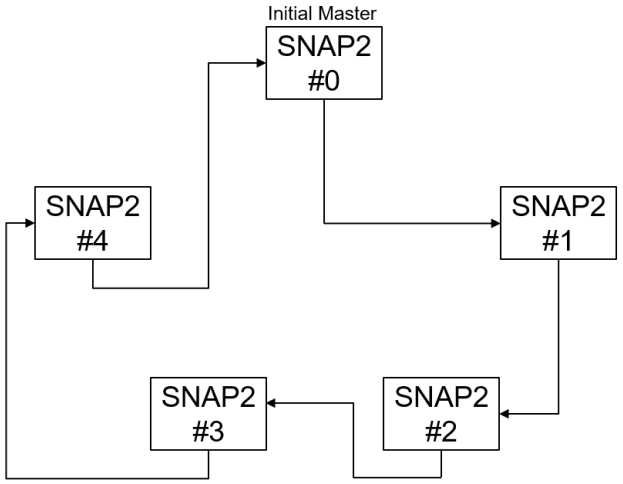


Figure 3

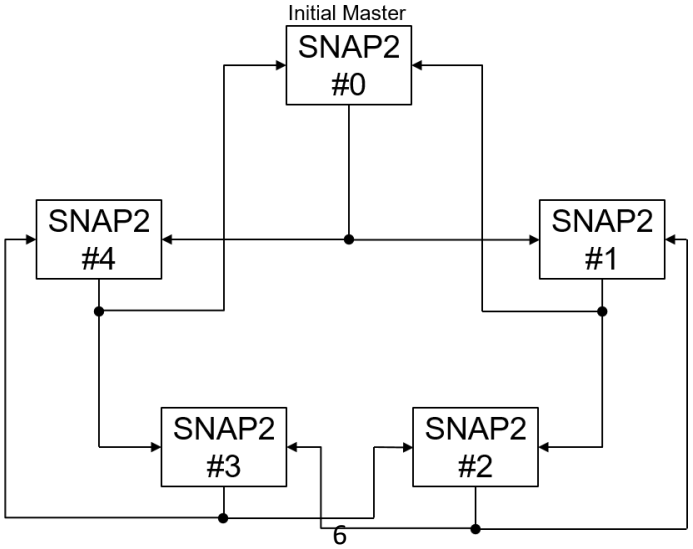
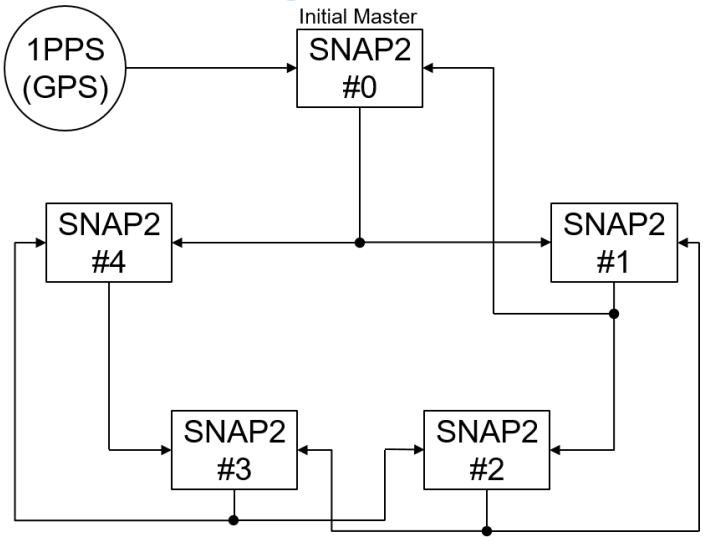


Figure 4


Initial Master
1PPS
(GPS)
SNAP2
#0
SNAP2
#4
SNAP2
#1
SNAP2
#3
SNAP2
#2