LIGO Laboratory / LIGO Scientific Collaboration

LIGO-T070218-00-D	*advanced LIGO*	8/31/2007

Timing Synchronization for Advanced LIGO Timing System

Maxim Factourovich

Distribution of this document:
LIGO Science Collaboration

This is an internal working note
of the LIGO Project.

California Institute of Technology
LIGO Project – MS 18-34
1200 E. California Blvd.
Pasadena, CA 91125
Phone (626) 395-2129
Fax (626) 304-9834
E-mail: info@ligo.caltech.edu

Massachusetts Institute of Technology
LIGO Project – NW22-295
185 Albany St
Cambridge, MA 02139
Phone (617) 253-4824
Fax (617) 253-7014
E-mail: info@ligo.mit.edu

LIGO Hanford Observatory
P.O. Box 159
Richland WA 99352
Phone 509-372-8106
Fax 509-372-8137

LIGO Livingston Observatory
P.O. Box 940
Livingston, LA 70754
Phone 225-686-3100
Fax 225-686-7189

http://www.ligo.caltech.edu/

Timing Synchronization for Advanced LIGO Timing System.

(the final technical paper)

Maxim Factourovich
(mentors: Daniel Sigg, Paul Schwinberg)

Summer Undergraduate Research Fellowship (SURF) program,
California Institute of Technology, Massachusetts Institute of Technology,
LIGO Hanford Observatory, P.O. Box 159, Richland, WA 99352

September 20, 2007

Abstract.

The demands for the advanced LIGO timing system are to achieve timing accuracy and an overall clock synchronization with a precision of better than 1 microsecond. The new prototype of the timing network employs field-programmable gate arrays as its primary logic units, instead of discrete logic implemented within the initial version of the timing system. These gate arrays were programmed to sustain a reliable transmission of synchronization marks every second, with the individual adjustments for the traveling-time delays to each remote location, and to detect and report occurring synchronization errors, if any. Additionally, the same channel is used to distribute the complete timing information provided by the Global Positioning System. Finally, the system was upgraded to transmit, receive and temporary store various data packets which, if necessary, could be read by an external PC via serial interface.

Introduction.

General idea.

The main purpose of the timing system is to create a single timeframe in which the entire network can be synchronized with the precision (at this stage) of no less and preferably better than one microsecond. While this is trivial when one deals with a group of devices localized within a few meters range, the problem arises when the network is spread out over distances of hundreds of meters and above. The size of LIGO interferometers at Hanford, WA and Livingston, LA is on a kilometer scale. The separation of the laser beam endpoints is 4 km, and the effective path traveled by the signals even exceeds that, considering the actual paths defined by the signal-carrying cables. For the 4 km separation, the traveling time of light in vacuum is about 13 microseconds, so one can expect the actual delay in the signal going through solid medium to be as high as 20 microseconds or more. Thus, a system must be built to account for and adjust to that.

The single time frame mentioned above should preferably be synchronized with the most precise clock available, in order to provide a reliable reference. At LIGO, the reference synchronization comes from GPS satellites every second, defining one pulse-per-second (1PPS)

marks. The secondary source is LIGO's internal high-precision cesium clock which has a highly stable period and in theory may alone keep the system within the synchronization requirements.

The timing system consists of two main types of controllers: "Fanouts" and "Slaves" (I capitalize these, to avoid confusion with the literal meanings). The Fanouts work, in a sense, as routers. They acquire precise timing data from the sources mentioned above and distribute it throughout the network, as well as collect all the status information and route it via uplinks to the external data collector (e.g. computer). They also make necessary adjustments for the time delays and allow real-time changing of system configuration, i.e. hot-swapping of remote devices. The Slaves are peripheral devices, the endpoints of the timing network. Every unit of the timing system, whether it is the MFO, a Fanout or a Slave, contains a programmable array of logical elements (FPGA) allowing it to perform elementary processing similar to, although less sophisticated than, that of central processor units commonly used in computers. The entire network design has a star-topology, multi-level structure, that is a single device at every particular level can be connected to any number of devices at the immediate lower level, but only to a single device at upper level. All the connections are symmetric in the sense that every inbound connection from a device is complemented by the outbound connection to the same device, which at this point disallows daisy-chaining of multiple devices on a single channel (i.e. ring topology). The figure below depicts the Master-Fan-Out (MFO) board. It consists mainly of an FPGA core with an additional RAM memory chip, a ~67MHz voltage-controlled oscillator (VCO) used as internal clocking source, an onboard GPS receiver connected to external antenna, various input/outputs and a power supply.

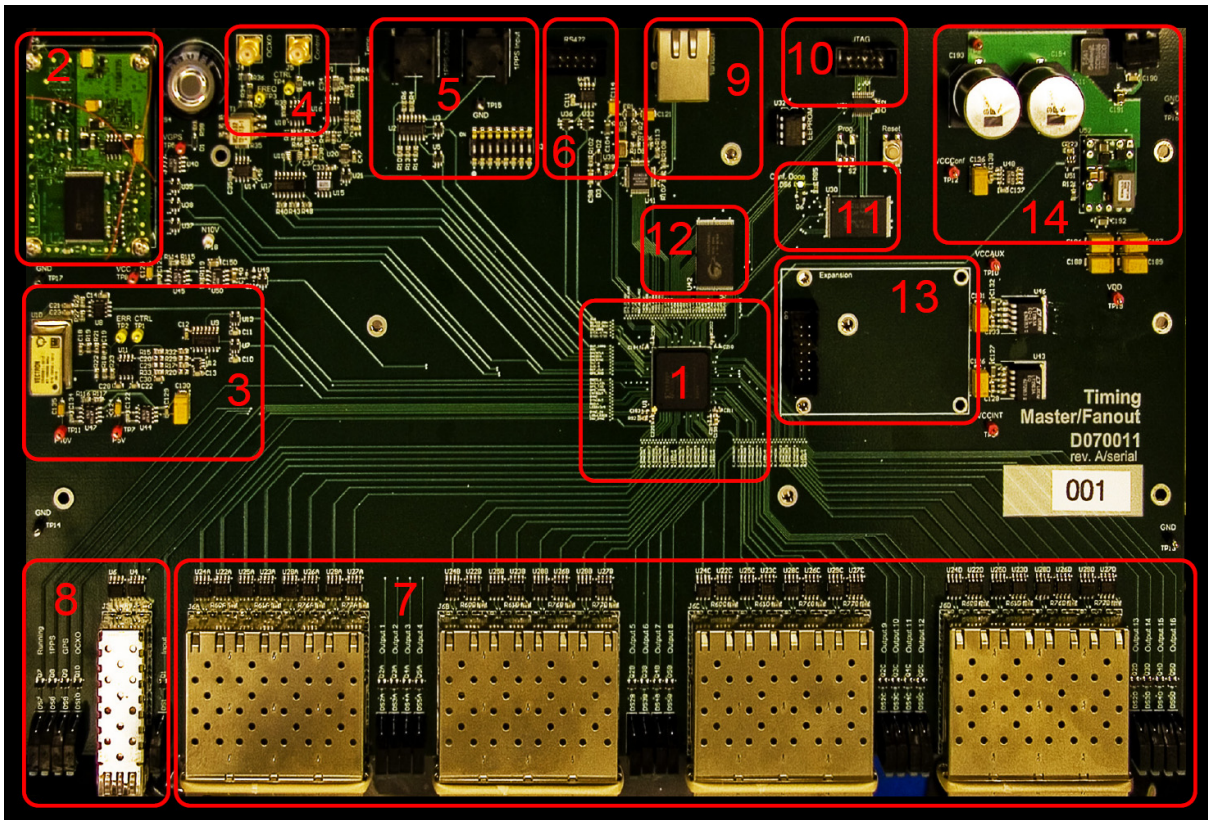

Figure 1. The Master Fanout board (MFO). (**1**) FPGA core chip. (**2**) On-board GPS-receiver. (**3**) **2^{26}Hz** voltage-controlled oscillator, used as internal clock. (**4**) The port for connecting external oven-stabilized **2^{26}Hz** oscillator. (**5**) BNC port for connecting to the external 1PPS source. (**6**) General purpose RS422 port. (**7**) 16 optical I/O fanout channels. (**8**) Uplink optical I/O channel. (**9**) General purpose Ethernet port. (**10**) JTAG port for external programming of the FPGA. (**11**) Flash memory chip (used for programming FPGA). (**12**) RAM block. (**13**) Optional slot (currently used for mounting JTAG programming device). (**14**) The power supply unit.

The LIGO timing system.

Every part of the network, whether it is the Master-Fanout, a Fanout or a Slave, has its own onboard 2^{26}Hz VCO. For each device, there also exists an onboard phase-locking loop (PLL) that locks the VCO frequency to the 2^{23}Hz signal coming from the uplink port (item 8 in the Fig. 1). The MFO gets its oscillator locked to an external master clock via either the uplink or the special port (item 4 in the Fig. 1). The alignment is triggered by the positive edge of the reference signal and its relative phase to the internal oscillator. Whenever the two edges misalign, the voltage applied to the internal clock either increases or decreases, making it either speed up or slow down, respectively. Once locked, the internal oscillator is used to generate synchronous 2^{23}Hz signal that is sent through all connected fanout channels (lower case denotes a port) to the uplinks of the receivers on the other ends. Those in their turn lock the oscillators to this signal, generate corresponding 2^{23}Hz waves, forward them downstream and so forth. In this fashion and after some initial time all the network becomes synchronized with the master clock, with the exception of a phase-shift due to the time intervals (i.e. delays) required for the signal to propagate the distance. This approach gives us an important feature. Whenever the internal oscillators stay locked to the master-clock frequency, we are guaranteed to have precisely 2^{26} clock-cycles within each second. The only missing piece then would be to the magnitudes of the propagation delays in units of the 2^{26}Hz clock-cycles. Each upstream sender will correct for it by sending it ahead of time in such a way as to compensate for the delays, so that at the network will be synchronized with the absolute precision of 2^{-26} second.

Except for the Master-Fanout, no other board "knows" its exact time position other than that of its relative phase shift to the incoming 2^{23}Hz signal (which should normally be near zero). It must somehow get this information in order to appropriately time-stamp the data. Since all its external communications go through a bi-directional optical fiber, a method is needed that will allow the same channel to be used for both, phase-locking the internal VCO and sending the timing information. The nature of the locking signal is oscillatory with strictly defined positive edges, but the negative edges do not affect the PLL and hence can be used to convey messages.

A natural timing message already mentioned in the previous section is the one that is native to the GPS receiver, gets transmitted once every second and defined as one pulse-per-second mark, or 1PPS. In the initial version of LIGO this is implemented by taking the two adjacent 2^{23}Hz cycles that mark the end of one second and the start of the next one, and increase the duty-cycle of the first and decrease that of the last by 50% each. The result is a wide pulse immediately followed by a narrow one, with the positive edge of the last pulse marking the start of the new GPS second.

The method just described allows us to start counting clock-cycles beginning every time a series of the two irregular pulses are encountered, from 0 to 67,108,863 as follows from locking to the 2^{26}Hz frequency. Any event can be time-stamped with the value of the counter at the moment of detection, yielding an overall precision of better than 15ns. A potential problem arises, however, when one considers time interval of exactly 1 second, as well as its whole multiple, after which the cycle-counter will have exactly the same value as it did before. In order to be able to resolve this ambiguity, some additional timing information is needed.

The proposed Advanced LIGO timing system references to GPS as its primary timing source. Every second a GPS receiver sends out a message containing date, time and positioning data. GPS-time standard is a 32-bit double word containing the number of seconds that passed since January 6, 1980, the beginning of the Coordinated Universal Time (also knows as UTC). Using this information along with the clock-cycle counter creates a unique time-stamp for each event, without possibility of confusion.

Thus, our timing synchronization consists of three main components. First, it uses the PLL to lock the internal oscillator frequency to a highly precise master clock, so that during any given second there are exactly 2^{26} clock-cycles. This is used to synchronize the *pace* of all clocks within the network. Second, every device with its own VCO clock employs a logic counter to count the clock-cycles, and the external 1PPS source, to determine the time-stamp of an event within each particular second with a resolution of better than 15ns. Third, the network reads out the complete GPS timing data, i.e. GPS-second count, to uniquely stamp any event in the absolute timescale. The main challenges to overcome here are (a) to create a universal timeframe for devices at all locations, regardless of the propagation delays, and (b) to develop a reliable protocol to convey both, the 1PPS and the GPS-second count messages within a single optical channel while preserving the periodicity and phase of the positive edges.

Since we are already dealing with transferring data, as required by the delivery of GPS-second information, it is logical to think of extending the data transfer capabilities by developing a protocol that will allow devices to actually communicate with one another as well as to report their status information to the main controlling device (e.g. computer). It is natural to use the same principle for encoding bits of information, as the one used to convey the 1PPS mark. The more detailed explanation is given in the next section.

Methods.

As was mentioned earlier, the network design has a star topology (Figure 2). At the root, there is the Master-Fanout board (MFO). The design has several interfaces connecting to it the reference clock, GPS receiver and PC via either serial or Ethernet port. For the downstream connections (i.e. towards the peripherals) it has 16 bidirectional channels equipped with optical transceivers. To these channels there could be connected either Slaves or other Fanouts as intermediate routers, for there is likely to be more than 16 Slaves altogether. A general Fanout has features similar to the MFO with the exception of the external PC, GPS and clock interfaces. The Slaves have neither fanouts nor routing capabilities come always at the network's "endpoints".

The FPGA chips used here belong to Xilinx® Spartan-3E product family, model number XC3S1600E-4FG320C [5]. The onboard GPS receiver is a clone of Motorola® Oncore M12+ series, manufactured by Synergy Systems [3]. The global clock resource for the FPGA is the onboard 2^{26}Hz VCO. Its frequency is used to generate the frequencies supported by the optical channels (2^{23}Hz AC-coupled) and the serial ports. The serial interfaces implemented here have 9600 baud-rate with 10-bit data packets consisting of 8-bit data, '0'-start, '1'-end bits and no parity bit. The signal format complies with the RS422 standard of differential $\pm$3.3V. The programming has been done in the Altium® Designer v.6 environment. Figure 3 shows the signal flow through the Master-Fanout and Slave FPGAs.

Figure 3 shows and compares the flows of data inside the Master-Fanout vs. Slave logic. The flow within a Fanout board is similar to that of MFO except for forwarding the GPS timing information instead of generating it. "Below" any Fanout can be either a Fanout or a Slave, whereas "below" any slave is an actual physical data source. These data sources are managed by the Slaves, who in their turn are controlled by the Fanouts, as seen from Fig. 2. On the top of the whole network is the Master Fanout that controls everything, while being controlled by the external computer that is the ultimate data collector. The common key-blocks of both types of boards are the optical I/O interfaces coupled with the data encoders/decoders, the [synchronous] clocking sources and the modules responsible for packing the data into packets and labeling the

addresses. An additional feature of a Fanout is data buffering and the flow control blocks, as necessary requisites of many-to-one channel transfers.

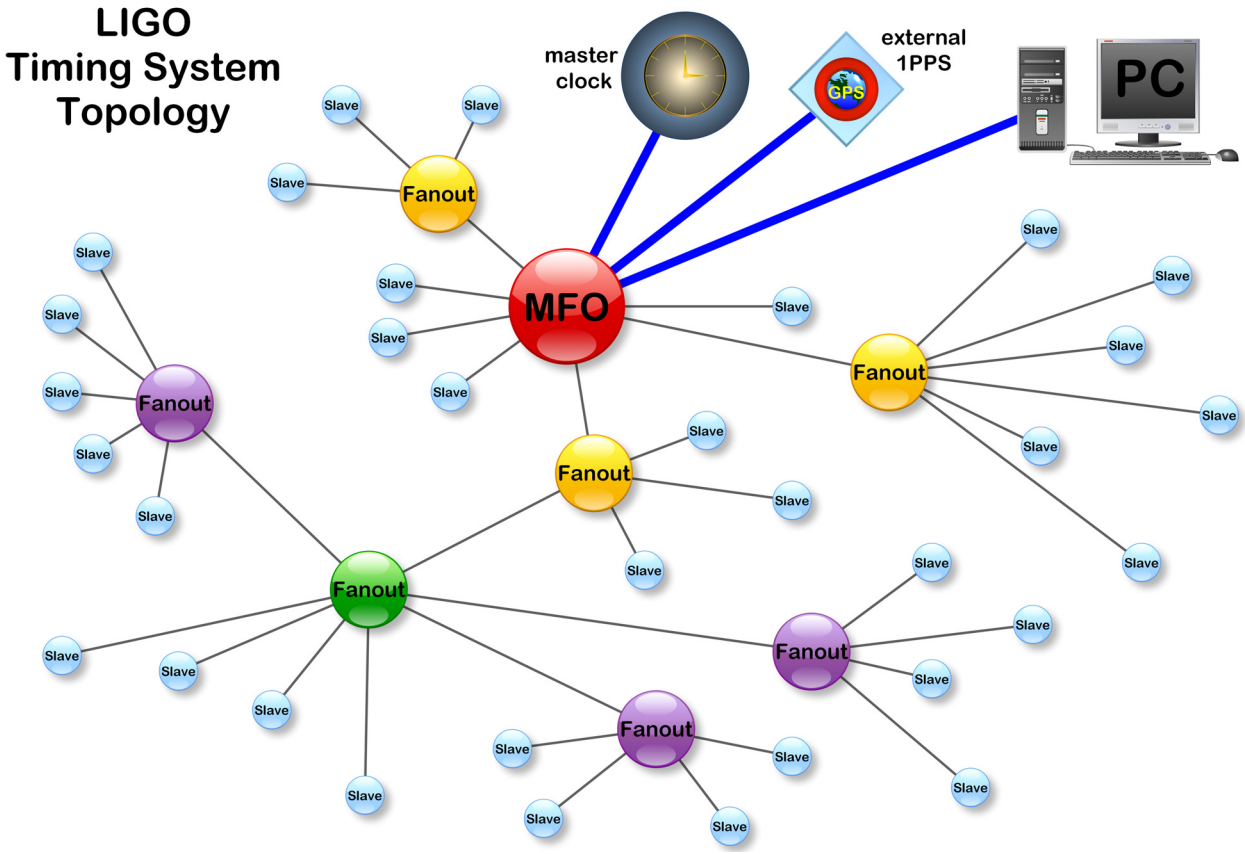

Figure 2. A schematic overview of the LIGO timing system.

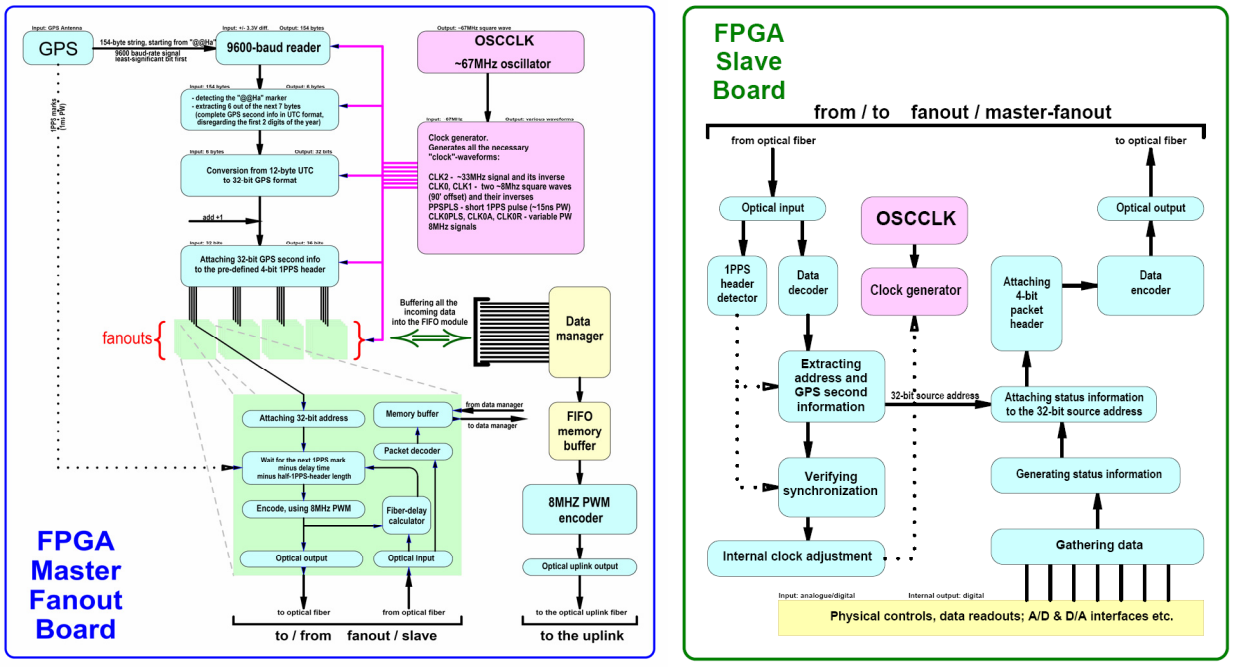

Figure 3. Comparative diagrams of the Master-Fanout vs. Slave signal flows.

Data encoding convention.

The proposed way of encoding data into the 2^{23}Hz square-waves is illustrated in **Figure 4**. The positive edges are not moved, otherwise the locking to the reference signal will be lost, and stay strictly periodic. The negative edges are shifted, producing square pulses with 25%, 50% or 75% duty-cycle. In this convention, we define any symmetric pulse (50% duty-cycle) as "0", and an asymmetric one as "+1" and "–1", depending on whether the duty-cycle is increased or decreased, respectively. If one sampled these pulses at the 2^{26}Hz VCO's frequency that provides the oversampling ratio of 8 (2^{26}Hz vs. 2^{23}Hz), then the [+1]-bit would produce an 8-bit string of "11111100", the [–1]-bit – "11000000", and the [0]-bit would yield "11110000" respectively. Hence the obvious way of modulating the output is therefore to set the output level to the logical "1", then count necessary number of the VCO cycles (2, 4 or 6) and then clear the output to the logical "0". For now, we do not distinguish between the difference in "polarity", "+1" and "–1", and set both of them to mean the binary "1". However, to account for the fiber receiving circuit's AC-coupling and preserve the DC-balance, we let "+1" and "–1" complement each other; we set the rule that no two asymmetric pulses of the same sign can follow one another, whether or not there is any number of zeroes between them. For example, the 16-bit encoding of the year 2007, that in binary format has the expression of "0000111111010111", in our definitions would have a form of

"|0|0|0|0|+|–|+|–|+|0|–|0|+|–|+|"
or "|0|0|0|0|–|+|–|+|–|0|+|0|–|+|–|", but never of something like
"|0|0|0|0|+|+|+|+|–|0|–|0|–|–|+|"

The "1"s were omitted, for brevity, only the signs displayed. Notice the alterations of the signs. There is, however, one exception, when two asymmetric pulses of the same sign go in series. This exception used for uniquely identifying the 1PPS mark.

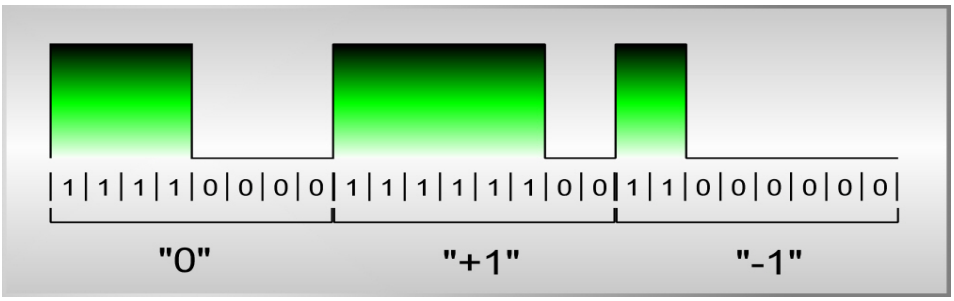

Figure 4. Signal configuration for pulse-width-based binary encoding.

Time slots.

As long as the 2^{26}Hz oscillator remains locked to the 1PPS input, it is virtually guaranteed to have exactly 2^{26} clock-cycles within each second, or 2^{23} bits of information (based on the 2^{23}Hz PWM frequency), or 2^{16} packets of information 128-bit long each. It then becomes convenient not to send packets at random times but to break-down seconds into 128-bit time slots (2^{10} or 1024 clock-cycles) and start every packet transmission at the beginning of the next available time slot. We define the -1^{th} time-slot to be the one that contains the 1PPS packet (explained below), so that the positive edge marking the first bit of the next, the 0^{st} time-slot (or packet), will be used for the 1PPS synchronization. This definition also allows, if necessary, to assign to each time-slot or packet a unique 16-bit time-identifier, e.g. in the case of a time-sensitive transmission.

General data packets and the special 1PPS packet.

Based on the time-slot convention, our "standard" data packet (see **Figure 5(a)**) must have length of 128 bits. For convenience, we also subdivide those into 32-bit, double-word logical segments. Our convention is to transmit the most significant bit first. The first segment (the bits 127..96) of a data packet carries the addressing information: a 1-bit flow-control tag, a 3-bit address "offset" value and the 28-bit address string. The value of the flow-control bit tells the receiver about the availability of the sender to accept the incoming data. The 3-bit offset indicates the topological "depth" to which the packet must go, i.e. how many devices should it pass through on its way to destination. The 28-bit address is a unique label assigned to every FPGA device of the network. The next two segments of the packet contain the actual "payload" information. Following it, the first word of the last segment contains the packet identifier and the last word – the Clock-Redundancy Check (CRC).

The 1PPS data packet, shown in the **Figure 5(b),** is transmitted every second (hence its name), starting precisely at -2^{-16}s minus the travel-delay, with respect to the start of the next second, so that when it arrives at its destination, the positive edge marking the start of the next time-slot must correspond to the beginning of the next second of the reference time-frame (GPS-time) experienced at the target's location. The total length of the 1PPS packet is also 128 bits. The 4-bits in cyan block (bits 31..28) define the 1PPS marker and mark the approaching boundary between the two adjacent seconds that serves as synchronization trigger for remote devices. They comprise the single exception from the rule described in the encoding convention. The following diagram shows the signal configuration of the 4-bit 1PPS header whose unique sequence consists of the bits (+)(+)(–)(–).

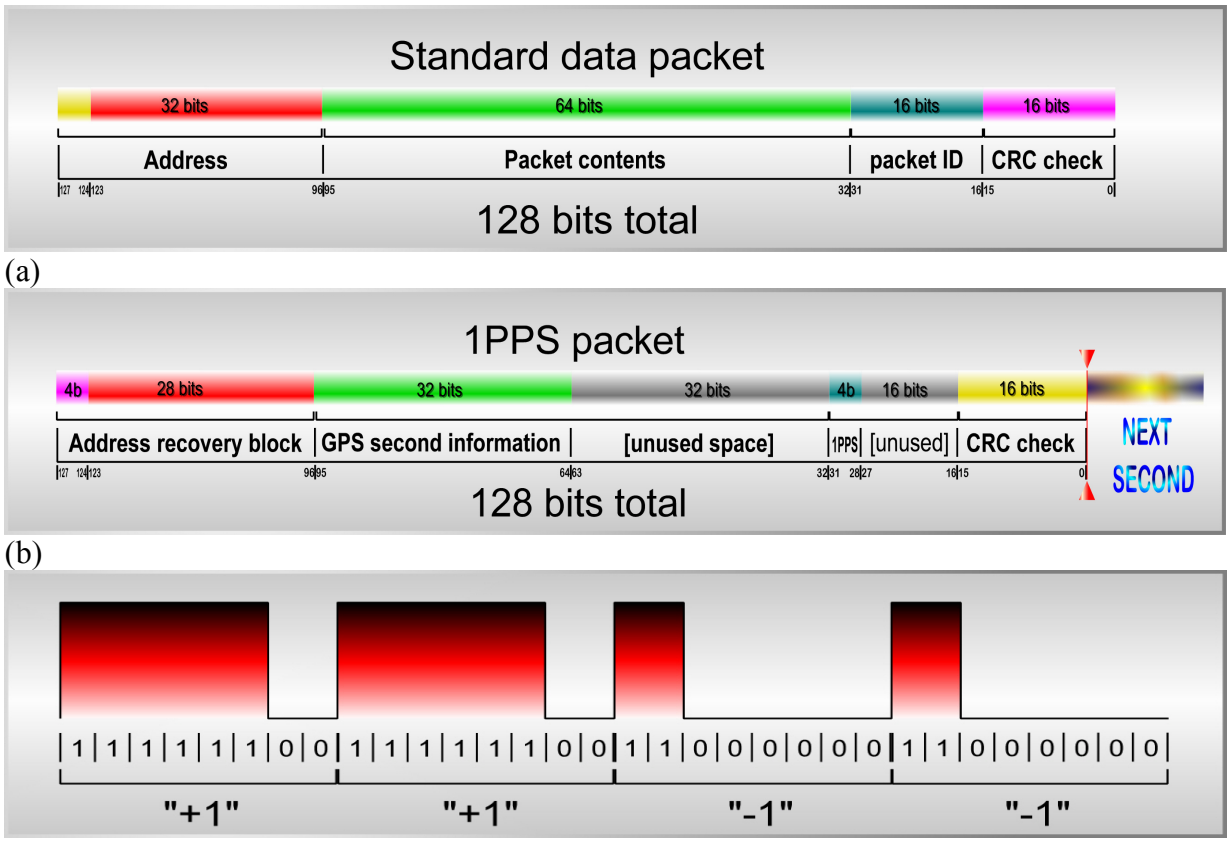

Figure 5. Schematics of (a) a general data packet, (b) a packet carrying 1PPS marker with the address recovery block, and (c) signal configuration for the 1PPS header.

Dynamic address recovery.

Offset, binary	Offset, decimal	Bits modified
000	0	None(MFO) or Invalid!!
001	1	27..24
010	2	23..20
011	3	19..16
100	4	15..12
101	5	11..8
110	6	7..4
111	7	3..0

The purpose of the dynamic address recovery is to assign every element of the network its unique address and to adapt in real-time to eventual changes in the network configuration. Any status packet coming to the MFO must contain its source's address, in order for MFO to "know" whose status is being reported. Similarly, every packet targeting a particular device must be able to find its destination. Hence the addresses of the up-routed and down-routed packets have a principal difference. For the up-coming packet, the address reflects the address of its source, whereas for the down-coming packet has an address of its destination. A Fanout board has 16 downstream I/O connections each of which can be connected either to a Slave or another Fanout board. Addressing each of these 16 channels requires at least 4 bits of addressing information. Serial connection of several Fanouts therefore multiplies that number by the length of the hierarchical chain, producing 8-, 12-, 16-bit and even longer addresses. We define the address length to be 32-bits, which is a manageable power-of-two number and should allow enough flexibility to have at least 5 intermediate Fanouts between the MFO and the end-Slave. The system must also be able to self-adjust to any configuration changes that might take place in real-time, e.g. unplugging a device and then re-plugging it to a different port. It is convenient to make such diagnostics every second and automatically reassign addresses to every "new" devices found.

The address recovery is implemented by sending a fixed-length header within the 1PPS packet, which gets modified on its way downstream, depending on its routing path, and is used to initialize or setup the address register for each receiver of the 1PPS packet. The address recovery is initialized by the Master-Fanout and propagates as following. The value of the 3-bit offset value (bits 126..124 in the **Figure 5b**) is set to "000". Then this value is increased by one and overwritten over the packet's original value. The first 4 bits of the 28-bit address value (red) are set to correspond to the fanout port number, through which the packet is being set. The receiver on the other end reads the 32-bit address and stores it in its address register. Until the arrival of the next 1PPS mark it becomes the device's *internal source address*. The offset has 4-bit unit value, that is, an increment by 1 means offsetting by 4 bits along the 28-bit sequence. If the device is a Fanout itself, then when sending the next 1PPS packet downstream, it takes its own source address and modifies the 4 bits that correspond to its internal offset value (in this case, the bits 23..20, reflecting the offset of 1). When a FO-board receives a packet coming from the uplink, it compares the offset contained in the packet's address to the one in its own register. If the two values are equal, that means that the FO-board is the proper receiver, and the packet doesn't go any further. If the packet's offset is greater than the board's own, then the packet is forwarded to the fanout port that corresponds to the 4 bits of the packet's address whose offset is the same as the internal offset of the board. For instance, the MFO receiving the packet with the offset of "002" would read the first 4 of the 28 bits ("zero-offset" bits 27..24) of the packet's address and forward the packet to the fanout port corresponding to the value of these 4 bits; the next device in the chain will forward the packet to the fanout corresponding to the bits 23..20 (offset "001"); and the next device in the chain will finally absorb the packet.

Every device uses an "internal addressing" flag that reflects the state of whether or not a proper address had been assigned to the device. Whenever this flag is "low", it sets the "transfer allowed" flag (described below) to "high", and no outbound up-the-chain transmission occurs. Unplugging or initialization of the device clears this flag and sets the internal offset to "000". Only a device that has inbound GPS and reference clock connections can recognize itself as MFO. If these conditions are not met but the internal offset is set to "000" nevertheless, an error message is produced, which means that the address is invalid. In this case, the Fanout starts sending downstream 1PPS packets with the "000" offset values as well. Whatever receives it via uplink immediately recognizes "invalid address" as well, resets its own address register to all-zeroes and forwards the packet further down with the same "illegal" offset. This is implemented for the case when an entire branch, rather than a single device, gets unplugged, so that all address values of the disconnected unit and of everything downstream automatically lose their validity. Once the illegal value is written in the internal address register, the internal addressing flag goes Low, and this immediately aborts all upstream communications. This flow control scheme work in such way as to stop receiving data whenever the device loses its uplink connection or the uplink no longer accepts data. Thus, once triggered by the loss-of-signal indicator, this message will quickly propagate through all the disconnected branches, stopping unnecessary communications, putting the line in the idle state and awaiting the re-appearance of the uplink.

An important note about the dynamic address recovery is that it requires a finite amount of time, depending on the network "depth" to complete the full cycle of assigning addresses, since the address validation packets are sent by a unit always after its own address has been validated, which takes up to a second. The unit then waits until the next second to transmit its first 1PPS packet containing the address recovery information.

Data flow control.

Every Fanout board has 16 input channels at fanouts but only one channel of the same bandwidth for all uplink transfers. This has a potential problem of losing valuable data when the rate of the data input exceeds the rate of the output. To some extent, this can be tolerated by using a memory block of the type First-In-First-Out (FIFO) as a temporary storage of the data to be sent. The size of FIFO, however, is limited, so when the input constantly exceeds the output there will be at some point no available space left, causing all the additional input data to be discarded. To avoid this, we set-up internal FIFO flags that reflect its "almost full" and "almost empty" states. We define the "almost full" (AF) flag to go up when the FIFO filling ratio exceeds 75% of its capacity, and the "almost empty" (AE) flag – when the ratio goes below 25%. Whenever either AF or AE flag goes high, a special packet containing such information is sent through all the fanout channels, and in its turn either sets or clears the other ends "transmission clearance" (TC) flag. Like IA flag, the TC can also trigger the state of the "transfer allowed" (TA) flag, which is a highest priority flag for either allowing or restricting outbound up-the-chain transmissions. In general, the TA is triggered by any number of components in a Boolean "NOR" fashion, i.e. raising up any of its components sets TA to "low and automatically suspends the transmission.

The truth table for the "Transfer Allowed" (TA) flag

IA	TC	TA	Transmission occurs
0	0	1	yes
0	1	0	no
1	0	0	no
1	1	0	no

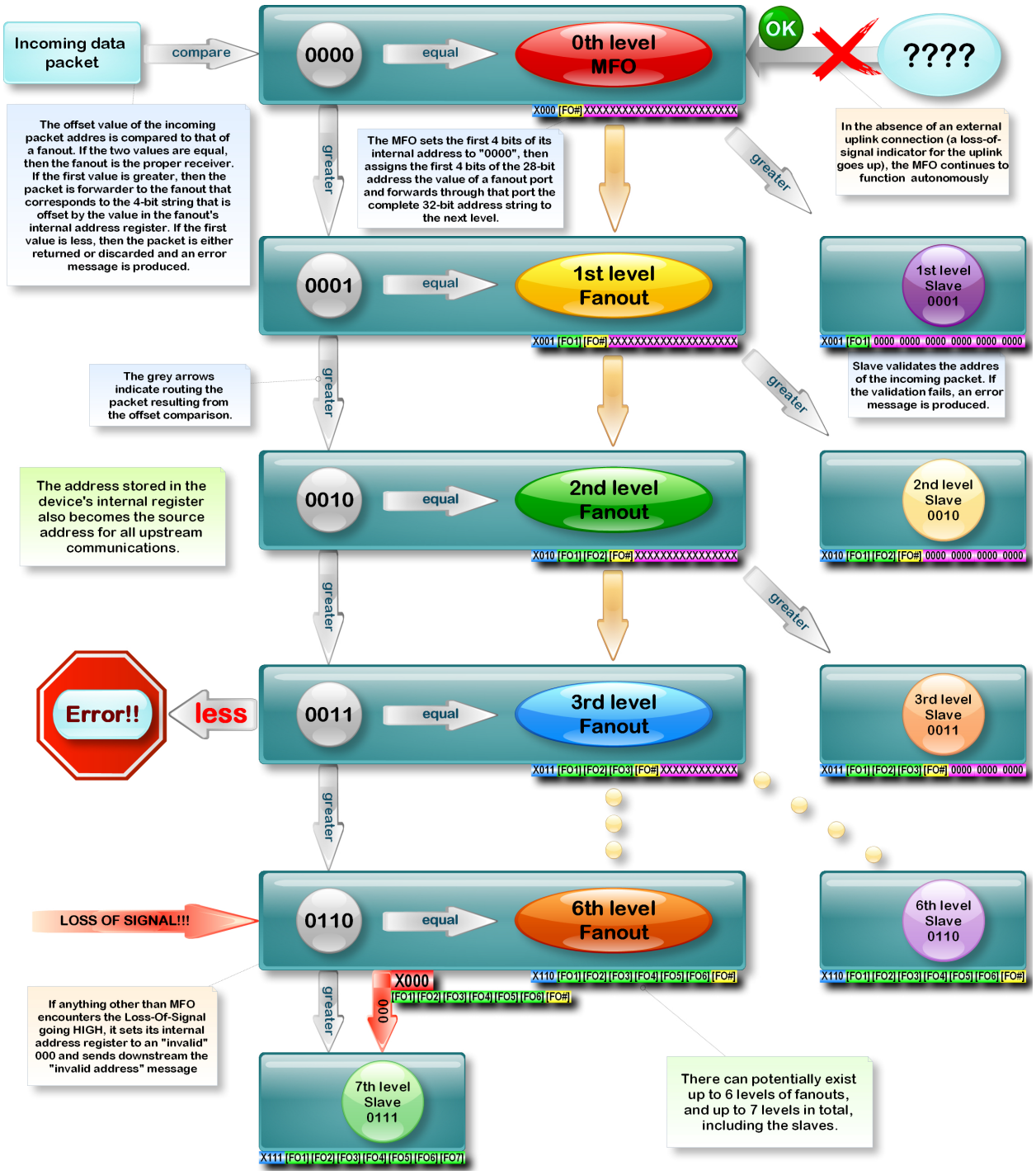

Figure 6. A schematic diagram of the dynamic addressing engine. The yellow arrows indicate the flow of the address recovery string enclosed within the 1PPS packet. The grey arrows show the flow of information packets, once received from an external source. The engine consists of two main cycles. One is used for the address set-up/recovery and performed every second along with the propagation of the 1PPS packet. Every second an additional level is being assigned with the addresses. The second cycle is the data routing. It uses comparison functions at each level, as data packet flows downstream, and either absorbs the packet, re-routes it or produces an error message in case of invalid address. The “loss-of-signal” flag (red arrow) triggers resetting the internal address to an “invalid” 000 and the propagation of this message downstream. When the address is invalid, all upstream communications are aborted.

Discussion.

Testing DC-imbalance tolerance.

Since the drivers of the optical channels are AC-coupled, there may be a limit to which asymmetric pulses will be tolerated. It is important to know, to what extent the shift of the negative edges can be exploited. For that purpose, a special logical circuit was designed that generated series of long pulsed without alteration with the short ones. The signal was then forwarded to one of the fanouts, routed via optical fiber to another fanout of the same board, decoded, buffered, and then sent to the RS422 port for monitoring. Additionally, the decoded string was compared to the initially generated one and, if there would be any difference, an error signal would be produced to trigger one of the onboard LEDs. Since the bandwidth of the RS422 is by far less than that of the 8MHz fanout, the maximum length of the string was limited to 1024bits, which in any case is considerably longer than any asymmetric mark ever to be used in the network. After several tests, the returned signal showed perfect match with the generated one, no error signals were observed, and this suggests that the optical driving circuitry has virtually indefinite tolerance to DC-imbalance.

The 1PPS synchronization and adjustment for the propagation delay.

Delays due to 4km arm length may exceed 20us, while the goal to keep the network synchronized within 1us uncertainty. To provide synchronization of remote devices with such precision, the synchronizing 1PPS marks must therefore be advanced by the proper time interval, so that their arrival times will all fall within 1us period. The implementation of this adjustment requires synchronous two-way communication between every Fanout and each of its immediate neighbors down the chain.

The mechanism of the 1PPS synchronization is somewhat similar to that of the address recovery. It obviously uses the 1PPS packet as primary message conveyor. Note that the default value of the internal addressing flag mentioned above is Low. That is, whether a unit was just powered up or disconnected from the rest of the network, its address is set to illegal value and all the upstream communications are aborted. Whenever this flag is Low and a new uplink connection is encountered, a flag permitting re-synchronization goes High. The unit is re-synchronized with the first 1PPS packet it recognizes from the uplink port, and the "re-sync allow" flag goes low. Generally, this flag goes when any of the following *conditions* is met:

(a) the unit has just been initialized, i.e. boot-up
(b) the uplink connection has been lost
(c) the maximum amount of allowed consecutive sync-errors has been exceeded (this number is set to be greater than 1 to avoid re-synchronizing every time a transmission error occurred or the possible signal jitter gets "quantized" to a wrong value by limited scanning resolution)

After initial resynchronization, the unit sends the 1PPS return packet at the end of the current second, that is simultaneously with the identical packet being received. The fanout on the other end of the uplink also recognizes a new connection and calculates the time interval between the moments when the 1PPS packet was sent and when the return packet was received. This period would normally indicate the propagation delay. In case of initial synchronization, only *half* of that period equals to the true delay, assuming both fibers run in parallel. Recognizing the new connection, the upstream fanout writes half of the delay in its internal "1PPS delay" register and advances the next 1PPS packet by that amount. The downstream unit on the other

end recognizes the shift, waits until it exceeds the maximum allowed error count (condition "c" above) and then resynchronizes again. If everything went well (i.e. no transmission errors or disconnections), then the upstream fanout will get the same delay as the one stored in its internal register. Since the connection is no longer "new", no adjustments will be made from upstream and the loop will go indefinitely, remembering that the internal VCOs are all locked to the master-clock generated 2^{23}Hz signal. The delays are nevertheless constantly monitored and compared to the one that was initially obtained. An additional adjustment is done only in the case when the delay inconsistencies become significant in comparison to the 1us threshold and repeated for several measurements in a row.

References.

1. P.Schwinberg et al., Proposal for a New Timing Distribution System, Caltech, 2004
2. Altium Software Documentation, http://www.altium.com
3. LIGO Master-Fanout Board Schematics, http://www.ligo.caltech.edu/docs/D/D070011-A/
4. Synergy Systems Inc., http://www.synergy-gps.com/
5. Wikipedia, http://www.wikipedia.org
6. Xilinx Spartan-3E Documentation, http://www.xilinx.com