Open Grandmaster

TAP Project

1. Abbreviations

OCP	Open Compute Platform
TAP	Time Appliance Project
PTP	Precision Time Protocol
PTM	Precision Time Measurements
NTP	Network Time Protocol
GM	GrandMaster
GMC	GrandMaster Clock
GTM	Go-To-Market
XO	Oscillator
TCXO	Temperature-compensated Oscillator
OCXO	Oven-Controlled Oscillator
GNSS	Global Navigation Satellite system
ToD	Time of Day
TS	Timestamp

PHC	PTP Hardware Clock
DC	Datacenter
HW	Hardware
SW	Software
OGM	Open GrandMaster
NIC	Network Interface Card
COTS	Commodity off-the-shelf

3. General

OCP TAP is targeting to ease the addition of Time-sync as a service to the datacenter. The Project targets are to define the service requirements, deployment, and design of an open reference design.

The time-sync service is relying on a synchronization technology, for now, we are adopting PTP (IEEE 1588) with some addition to that and NTP.

PTP architecture is scalable and defines the time source from an entity called the **Grandmaster clock** (or stratum 1 in NTP terms). The GM is distributing time to the entire network and usually gets its timing from an external source, (GNSS signal).

The current state-of-the-art grandmaster implementations suffer from a few drawbacks that we wish to accommodate:

- They are HW appliances that usually target different GTM than a DC
- They expose none standard and inconsistent Interfaces and SW feature-sets
- Development cycles and the effort needed to add new features are large and expensive
- It doesn't rely on open-source software
- The accuracy/stability grades aren't in line with DC requirements

This document describes an open architecture of a Grandmaster, that could eventually be deployed either in a DD or in an edge environment.

4. High-Level Architecture

In general, the OGM is divided into 3 HW components:

1. COTS server
2. Commodity NIC
3. Time Card

The philosophy behind this fragmentation is very clear, and each decision, modification that will be made, must look-out to this philosophy:

- COTS servers keep their "value for money" due to huge market requirements. They are usually updated with the latest OS version, security patches, and newer technology, faster than HW appliances.
- Modern Commodity NICs already support HW timestamp, lead the market with Ethernet and PCIe latest Speeds and Feeds. Modern NIC also supports a wide range of OS versions and comes with a great software eco-system. NIC + COTS server will allow the OGM to run a full software (and even open source one) PTP stack.
- Timecard will be the smallest (conceptually) as possible HW board, which will provide the GNSS signal input and stable frequency input. isolating these functions in a timecard will allow OGM to choose the proper timecard for their needs (accuracy, stability, cost, etc) and remain with the same SW, interface, and architecture.

4.1. Responsibilities/ Requirements

4.1.1. OGM Requirements

4.1.2. COTS Server

- run commodity OS
- PCIe as an interconnect
- support PTM

4.1.3. Network Interface Card

- PPS in/out
- PTM
- Hardware timestamps
- optional Time of day tunnel from timecard to SW

4.1.4. Time Card

- Holdover

- GNSS
- Leap
- Time of day

4.2. Block Diagram

<TBD>

4.3. Interfaces

<TBD>

5. Detailed Architecture

5.1. COTS Server

5.1.1. Hardware

<TBD>

5.1.2. Software

<TBD>

5.1.3. Requirements

- Linux/*nix operating system
- Device should expose itself as `/dev/ptpX` (TOD) and `/dev/ppsY`.
  In case of software "bridge" GPSd will do it
- phc2sys copies atomic clock to NIC PHC.
  In case of hardware bridge 1PPS is connected directly from bridge to NIC PHC
- ptp4l serves PTP on NIC
- Chrony/NTPd reading `/dev/ptpX` of a NIC

5.2. NIC

5.2.1. Requirements

5.2.1.1. Form-Factor

- Ø Standard PCIe Stand-up Card
- Ø Half-Height, Half-Length, Tall Bracket (??)
- Ø Single Slot - Passive Cooling Solution
- Ø Support for Standard PCIe Tall and Short brackets

5.2.1.2. Pcie Interface

Ø PCIe Gen3.0/Gen4.0 X n lanes on Gold-fingers
n = at least 8

5.2.1.3. Network Ports

Ø Single or Dual-port Ethernet

5.2.1.4. PPS out

1.	PPS Out Rise/Fall Time	< 5 nano Sec
2.	PPS Out Delay	< 400 pico Sec
3.	PPS Out Jitter	< 250 fento Sec
4.	PPS Out Impedance	= 50 Ohm
5.	PPS Out frequency	1Hz - 10MHz

5.2.1.5. PPS In

1.	PPS In Delay	< 400pSec
2.	PPS In Jitter	< 250fSec
3.	PPS In Impedance	= 50 Ohm
4.	PPS In frequency	1Hz - 10MHz

5.2.1.6. Hardware timestamps

NIC should timestamp all ingress packets.
Non PTP packets can be batch and have a common TS in the SW descriptor, as long as they are not distant more than TBD nanosecond

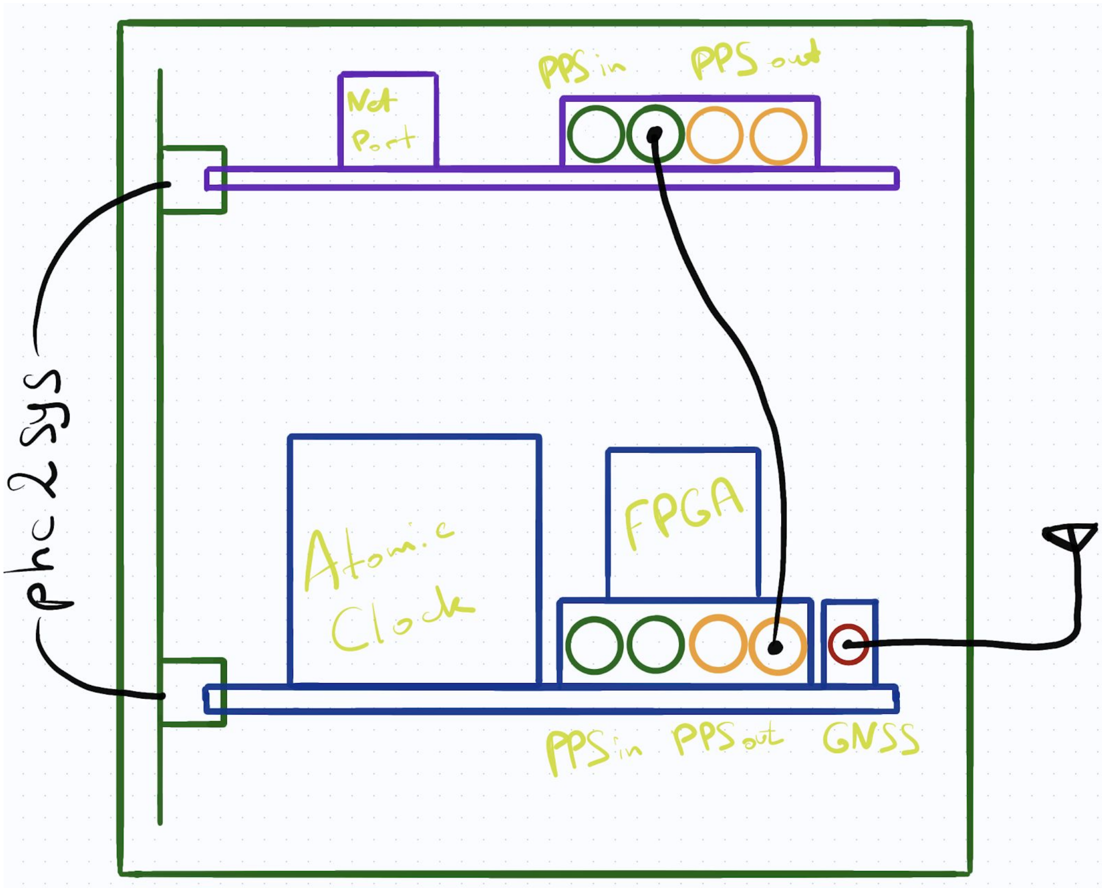
NIC should timestamp all PPP egress packets

- PHC
- PTM
- 1PPS input
- 10MHz input which can be used as frequency input to the TSU unit (optional)
- Network port <TBD speed, form-factor>
- Multi-host support (optional)

Examples:

- NVIDIA ConnectX-6 Dx
  https://www.mellanox.com/products/ethernet-adapters/connectx-6-dx

5.3. Time Card

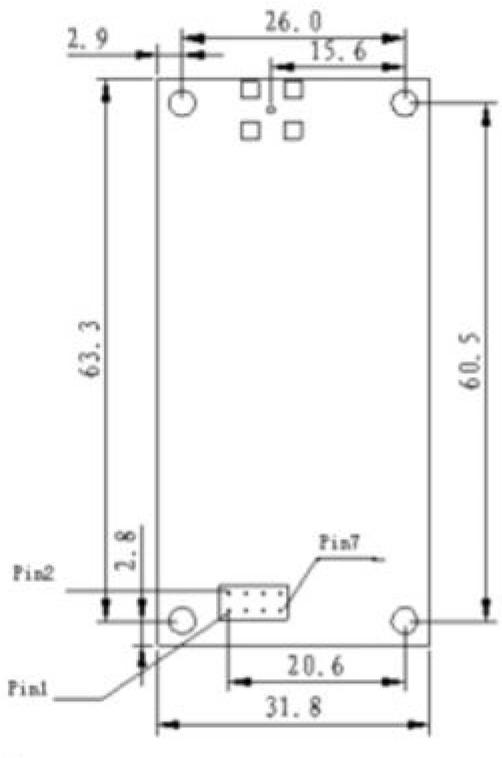


General Idea is this card will be connected via PCIe to the server and provide Time Of Day (TOD) via /dev/ptpX interface. Using this interface ptp4l will continuously synchronize PHC on the network card from the atomic clock on the Time Card. This provides precision < 1us.
For the extremely high precision 1PPS output of the Time Card will be connected to the 1PPS input of the NIC, providing <100ns precision.

5.3.1. GNSS receiver

The GNSS receiver can be a product from ublock or any other vendor as long as it provides PPS output and the TOD using any suitable format.

This is the recommended module: **u-blox LEA-M8T-0-10 concurrent GNSS timing module**



Pin No	Function	Describe
1	Ant Supply	5VDC(±5%)
2	Power In	5VDC (±5%)
3	TXD	TOD output LVTTL
4	NC	Reserved
5	RXD	TOD input LVTTL
6	1PPS	LVTTL
7	NC	Reserved
8	地	GND

5.3.2. Redundancy Protection

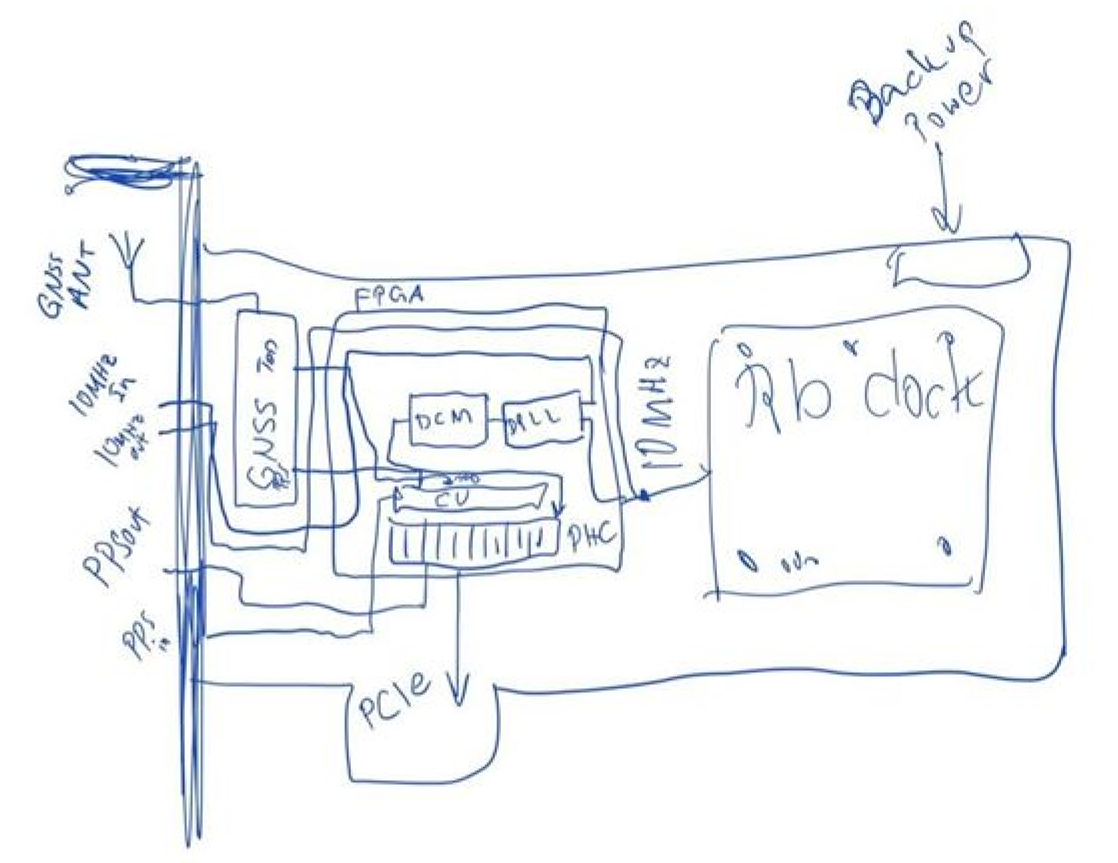
When GNSS signals are lost or interrupted there's a need for redundancy and holdover capacity and covered in the Hardware Holdover section

One approach is to use atomic clocks. Rubidium based atomic clock example: https://www.microsemi.com/product-directory/embedded-clocks-frequency-references/3825-miniature-atomic-clock-mac?fbclid=IwAR26trWBnHtV6ydBpKiViv3qS4jUpHAtQXJumUusIMB_RnCGclg2Qbd6lSc

5.3.3. Bridge

Bridge between GNSS receiver and Atomic clock can be implemented using software or hardware solutions. The hardware implementation is preferred and is our end goal.

5.3.3.1. Hardware Implementation



5.3.4. Interfaces

- PCIe (Generation and width TBD)
  - o to connect to the server and show up as an ordinary PHC device in Linux (/dev/ptpX) as well as PPS (/dev/ppsY). Should be generic, supporting multiple OS versions
- 1PPS / 10MHz SMA output
- 1PPS / 10MHz SMA input
- GNSS Antenna SMA input
- Network port: speed and form factor <TBD>

5.3.5. Form Factor

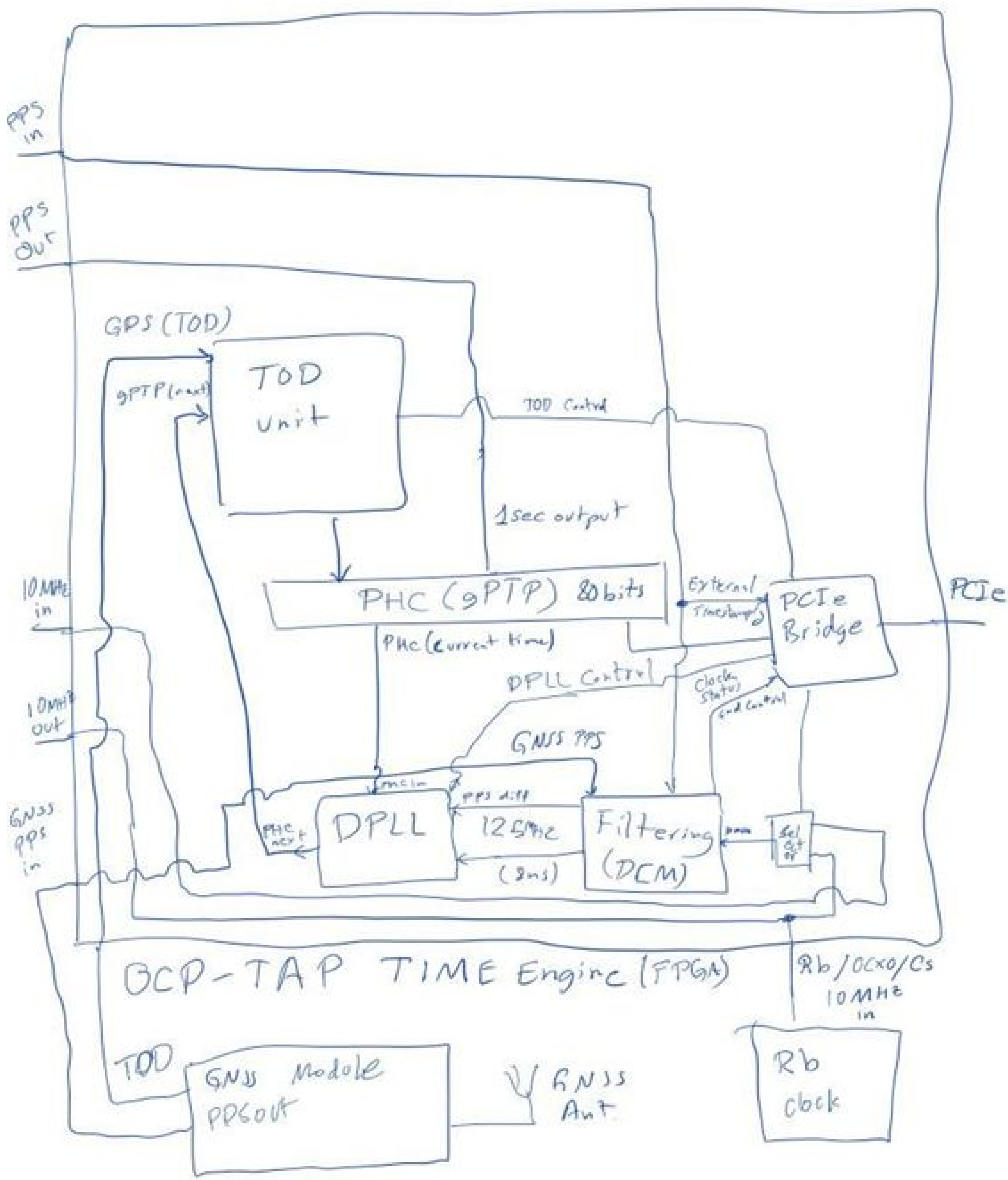
- PCIe to leverage more broader market platforms
- Environmental, cooling

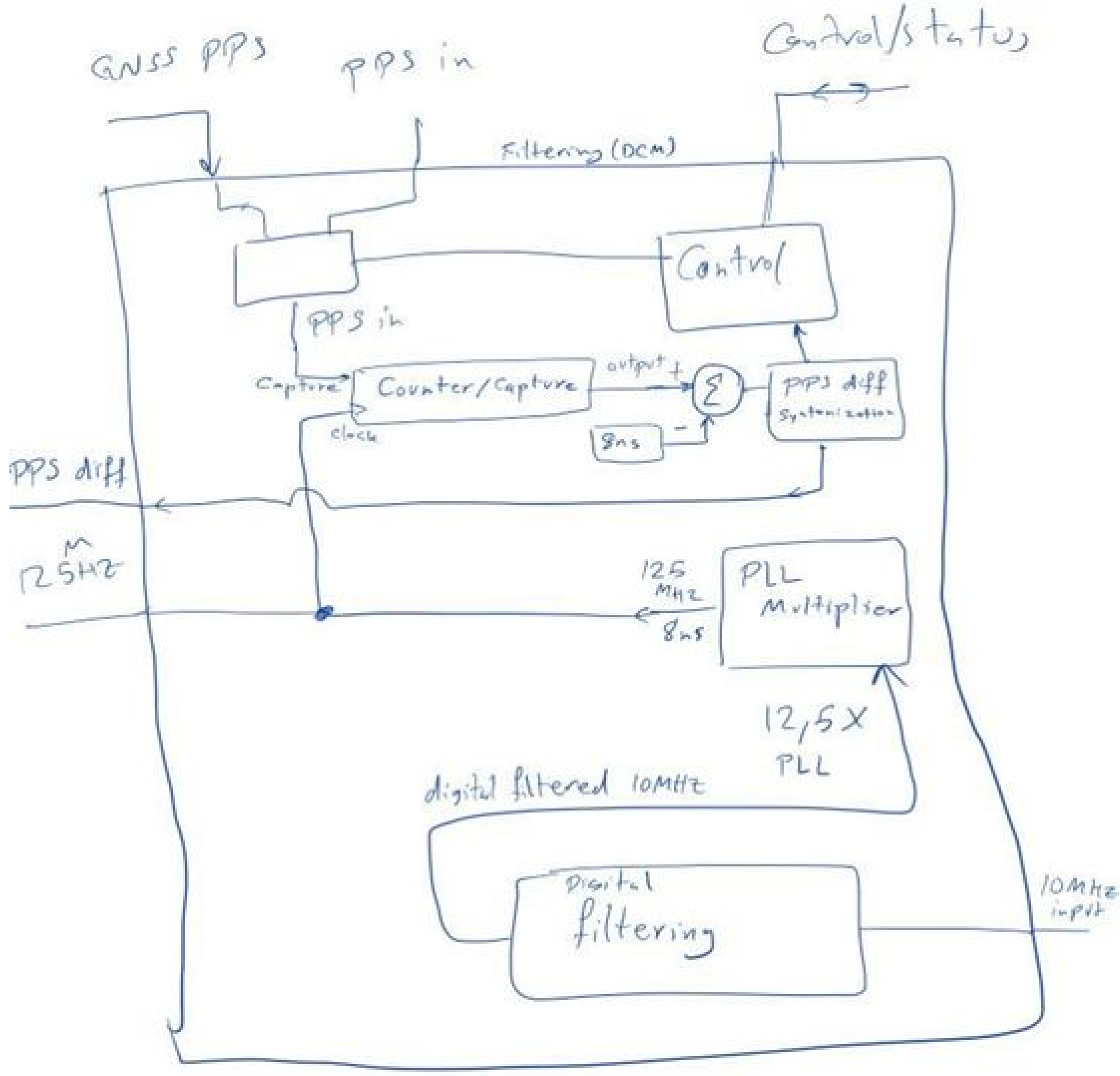
5.3.6. LED

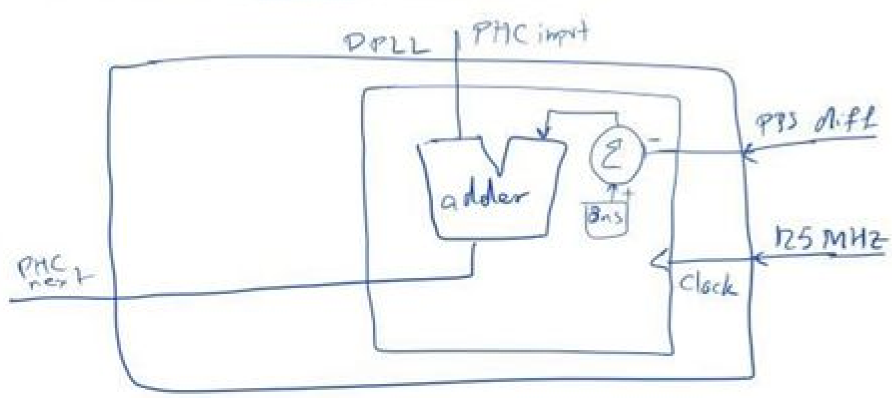
LED should be used to provide externally visible status information of the timing card. For example:

- Off - card is not powered or not properly fitted

- Solid green - card is powered, GNSS ok, 1PPS/10MHz output ok
- Flashing green - card is in warm-up, acquiring satellites
- Solid red - alarm / malfunction




GNSS PPS
PPS in
Control/status
Filtering (DCM)
Control
PPS in
Capture
Counter/Capture
clock
output
8ns
PPS diff
Synchronization
PPS diff
125MHz
125 MHz
8ns
PLL Multiplier
12,5X
PLL
digital filtered 10MHz
Digital
filtering
10MHz input


DPLL
PHC input
PPS diff
adder
8ns
125 MHz
Clock
PHC next

5.4. PTP Flows

5.4.1. Two-Step Sync

5.4.2. One-Step Sync

5.4.3. Delay Response

5.5. Holdover

5.5.1. Definition

5.5.2. Indication

5.5.3. Requirements