Open Compute project Ips Info
Everything is written in VHDL :'(

All Open IP blocks

AdjustableClock
ClockDetector
CommunicationSelector
ConfMaster
CoreList
DummyAxiSlave
FPGA Version
FrequencyCounter
MsiIrq
PpsGenerator
PpsSlave
PpsSourceSelector
SignalGenerator
SignalTimestamper
SmaSelector
TodSlave

AdjustableClock

The Adjustable Clock consists of a timer clock in the Second and Nanosecond format that can be frequency and phase adjusted. It adds the clock period it runs on, in nanoseconds, every clock cycle.The nanosecond counter has an overflow at 1000000000 nanoseconds. At each overflow of the nano-second counter it will add an additional second to the seconds counter. For adjustments, additional nanoseconds can be added or subtracted from the standard period to adjust frequency and phase. When small offset adjustments are not suitable, the time can be overwritten to hard set the clock to a new time e.g. for the startup phase where a jump from the 1.1.1970 to the present is required. An adjustment process takes the offset, drift and time adjustments as inputs and converts them into a combined adjustment which is then applied to the timer clock. The offset and drift is converted into evenly spread adjustments which allows smooth corrections on the clock without time jumps. E.g. a drift of 1ppm is adjusted as 1 ns every 5000000 clock cycles at 50MHz. In parallel to the correction a quality computation done to mark the in-sync state of the clock if corrections are below a certain threshold for at least 4 corrections. The drift and offset adjustments inputs are normaly outputs of PI servo loops (not required though), which are not part of the Adjustable clock. The PI servo loop parameters however are provided by the configuration of the Adjustable Clock to output signals. The drift and offset adjustments of the servos can be chosen individually, since frequency changes might happen quite slowly where offset adjustments probably shall be done much faster. Since the Adjustable Clock is the heart of a synchronization solution it can take several (5) adjustment inputs from different cores as input. Only one adjustment input is taken as source for corrections at the time. From the Registerset, the multiplexer gets the source selection of the adjustments.

The adjustments can be taken by one of the 5 source inputs or by the CPU via the AXI registers (REG mode). In this case, the adjustment calculations (e.g. also via PI servo loops) are executed on a CPU and their outputs are provided to the core as time, offset and drift adjustements. Additionally, the Registerset allows reading the in-sync state and taking a snapshot of the time. (From the Adjustable Clock Readme.md)

ClockDetector

The Clock Detector is a full hardware (FPGA) only implementation that detects the available clock sources and selects the clocks to be used. The selection is done with different clock selector and clock enable outputs. The selection is according to a priority scheme and it can be overwritten with a manual configuration. The configuration and monitoring of the core is done via a AXI4L slave interface.

- This could be used in our system when we have both the TCXO clock and the MAC

CommunicationSelecor

The Communication Selector is a full hardware (FPGA) implementation that selects which communication interface will be used between the FPGA and the module's clock (e.g. MAC, OCXO). The selection can be received by an AXI configuration register (e.g. by using AXI GPIO) and it is set as UART (Selection:'0') or I2C (Selection:'1'). The default configuration is UART communication.

- Could be used for the communicating with the MAC and the TCXO

ConfMaster

I think our system will already have this but it just configures the cores on the FPGA.

CoreList

Provides a list of all the cores to the FPGA?

DummyAxiSlave

Placeholder Core?

FPGA Version

IDK?

FrequencyCounter

The Frequency Counter is a full hardware (FPGA) only implementation that measures the frequency of an input signal. The counter calculates non-fractional frequencies of range [0 Hz -

10'000'000 Hz] and it is aligned to the local clock's new second. The core can be configured by an AXI4Lite-Slave Register interface.

MsiIrq

The MSI IRQ receives single interrupts of the FPGA cores and puts them into a message for theAXI-PCIe bridge. Once a message is ready a request is set and it waits until the grant signal from the Xilinx Core. If there are several interrupts pending the messages are sent with the round-robin principle. It supports up to 32 Interrupt Requests.

PpsGenerator

The PPS Generator is a full hardware (FPGA) only implementation that generates a Pulse Per Second (PPS) aligned to the local clock's new second. The core can be configured by an AXI4Lite-Slave Register interface.

PpsSlave

The PPS Slave is a full hardware (FPGA) only implementation that calculates the offset and drift corrections to be applied to the local clock, in order to synchronize to a PPS input. On the event of a PPS reception, a timestamp is taken and the pulse gets evaluated. If the pulse's width and period are valid, then the offset and the drift corrections of the local clock are calculated, in reference to the PPS input. To smooth the corrections to the local clock, PI servo loops are used. The PPS Slave is configured and monitored by an AXI4L slave interface.

PpsSourceSelector

The PPS Source Selector is a full hardware (FPGA) only implementation that detects the available PPS sources and selects the PPS source according to a priority scheme and a configuration.
The configuration and monitoring of the core is done via an optional external AXI4L slave interface (see the register set of the Clock Detector). If a configuration input is not provided, then the PPS selection is done based on the available PPS inputs and a default priority scheme.

SignalGenerator

The Signal Generator is a full hardware (FPGA) only implementation that allows to generate pulse width modulated (PWM) signals of configurable polarity aligned with the local clock. The Signal Generator takes a start time, a pulse width and period as well as a repeat count as input and generates the signal accordingly. The settings are configurable by an AXI4Lite-Slave Register interface.

SignalTimestamper

The Signal Timestamper is a full hardware (FPGA) only implementation. It allows to timestamp an event signal of configurable polarity. Timestamps are taken on the configured edge of the signal and interrupts are generated. The Signal Timestamper is intended to be connected to a CPU or any other AXI master that can read out the timestamps. The settings are configured by an AXI4Lite-Slave Register interface.

SmaSelector

The SMA Selector is a full hardware (FPGA) only implementation that multiplexes the outputs and demultiplexes the inputs of the 4 SMA connectors of the Timecard. Each connector can be configured as input or output, depending on the configured mapping. The following signals can be sourced by an SMA input:

- 10MHz Clock (only from SMA 1)
- External PPS 1
- External PPS 2
- Source to Signal Timestamper 1
- Source to Signal Timestamper 2
- Source to Signal Timestamper 3
- Source to Signal Timestamper 4
- Source to Frequency Counter 1
- Source to Frequency Counter 2
- Source to Frequency Counter 3
- Source to Frequency Counter 4
- IRIG Slave (unused)
- DCF Slave (unused)
- External UART Rx

The following signals can be mapped to an SMA outputs:

- 10MHz pulse
- PPS of the FPGA
- PPS of the MAC
- PPS of the GNSS 1
- PPS of the GNSS 2
- IRIG Master (unused)
- DCF Master (unused)
- Signal Generator 1
- Signal Generator 2
- Signal Generator 3
- Signal Generator 4
- GNSS 1 UART Messages
- GNSS 2 UART Messages
- External UART Tx

The possible mappings of the directions of the SMA data are:

Connector	Selection 1	Selection 2
SMA 1	Input	Output
SMA 2	Input	Output
SMA 3	Output	Input
SMA 4	Output	Input

The configured mapping is done via 2 AXI4L slave interfaces, named AXI1 and AXI2. Each slave interface controls one mapping option.

TodSlave

TodSlave gathers the ToD from the GNSS receiver and sends the ToD to the rest of the system. The block reads the GNSS data over UART. It is comparable with the u-blox M8, but from reading the data sheets for the u-blox M8 and u-blox MAX-M10S these two chip have the same UART or similar enough to change the block to read the MAX-M10S's messages.