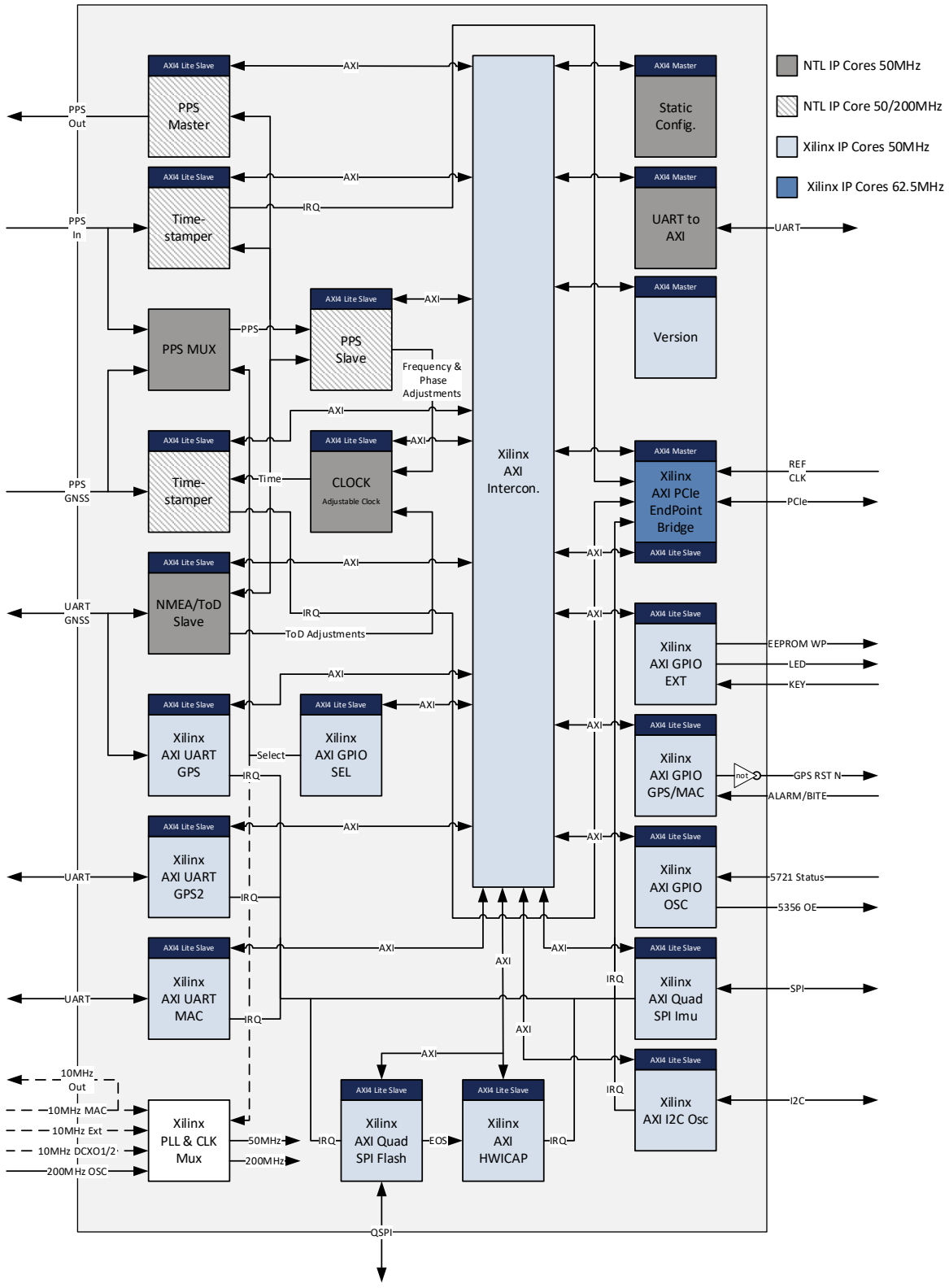
# 1 Design Overview




The TimeCard runs partially from the 200MHz SOM oscillator. The NetTimeLogic cores with all the high precision parts are running on the 10MHz MAC.

## 1.1 AXI Address Mapping

The AXI interconnect has three masters which all have access to the slaves. One is the AXI PCIe interface and the other UART to AXI have all the time access. The Static Configuration only does a basic configuration after reset.
The AXI Slave Interfaces have following addresses:

| AXI Slave interface | Slave | Offset Address | High Address |
|---|---|---|---|
| S_AXI_CTL | AXI PCIe Control | 0x0001_0000 | 0x0001_0FFF |
| axi4l_slave | Version | 0x0002_0000 | 0x0002_0FFF |
| S_AXI | AXI GPIO Ext | 0x0010_0000 | 0x0010_FFFF |
| S_AXI | AXI GPIO GPS/MAC | 0x0011_0000 | 0x0011_FFFF |
| S_AXI | AXI GPIO OSC | 0x0012_0000 | 0x0012_FFFF |
| S_AXI | AXI GPIO SEL | 0x0013_0000 | 0x0013_FFFF |
| AXI_LITE | AXI Quad SPI Imu | 0x0014_0000 | 0x0014_FFFF |
| S_AXI | AXI I2C Osc | 0x0015_0000 | 0x0015_FFFF |
| S_AXI | AXI UART 16550 GPS | 0x0016_0000 | 0x0016_FFFF |
| S_AXI | AXI UART 16550 GPS2 | 0x0017_0000 | 0x0017_FFFF |
| S_AXI | AXI UART 16550 MAC | 0x0018_0000 | 0x0018_FFFF |
| S_AXI_LITE | AXI HWICAP | 0x0030_0000 | 0x0030_FFFF |
| AXI_LITE | AXI Quad SPI Flash | 0x0031_0000 | 0x0031_FFFF |
| axi4l_slave | NTL Adj. Clock | 0x0100_0000 | 0x0100_FFFF |
| axi4l_slave | NTL Signal TS0 | 0x0101_0000 | 0x0101_FFFF |
| axi4l_slave | NTL Signal TS1 | 0x0102_0000 | 0x0102_FFFF |
| axi4l_slave | NTL PPS Master | 0x0103_0000 | 0x0103_FFFF |
| axi4l_slave | NTL PPS Slave | 0x0104_0000 | 0x0104_FFFF |
| axi4l_slave | NTL TOD Slave | 0x0105_0000 | 0x0105_FFFF |

The Version Slave has one single 32-Bit Register. The upper 16 Bits show the version number of the golden image and the lower 16 Bits the version number of the regular image.

E.g.
Register 0x0200_0000 of the Golden image shows: 0x0001_0000
Register 0x0200_0000 of the Regular image shows: 0x0000_0003

## 1.2 Message-Signaled Interrupt Mapping

The interrupts in the design are connected to the MSI Vector of the AXI Memory Mapped to PCI Express Core via a MSI controller. The PCI Express Core needs to set the MSI_enable to '1'. The MSI controller sends INTX_MSI Request with the

MSI_Vector_Num to the PCI Express Core and with the INTX_MSI_Grant the interrupt is acknowledged.

| MSI Number | Interrupt Source |
|---|---|
| 1 | NTL Signal TS0 |
| 2 | NTL Signal TS1 |
| 3 | AXI UART 16550 GPS |
| 4 | AXI UART 16550 GPS2 |
| 5 | AXI UART 16550 MAC |
| 6 | AXI Quad SPI Imu |
| 7 | AXI I2C Osc |
| 8 | AXI HWICAP |
| 9 | AXI Quad SPI Flash |

## 1.3 Connectors (SMA)

ANT1:

10MHz Clock input

ANT2:

PPS Input (must be selected by AXI GPIO SEL GPIO2, Bit4 = 1)

ANT3:

10MHz Clock output (looped 10MHz Clock →MAC RF OUT after clock buffer)

ANT4:

PPS Output from the PPS Master

## 1.4 GPIO Mapping

| | | Bit4 | Bit3 | Bit2 | Bit1 | Bit0 |
|---|---|---|---|---|---|---|
| **AXI GPIO Ext** | GPIO (in) | - | - | - | KEY2 | KEY1 |
| | GPIO2 (out) | EEPROM WP | LED4 (NC atm) | LED3 (NC atm) | LED2 (NC atm) | LED1 (NC atm) |
| **AXI GPIO GPS/MAC** | GPIO (in) | - | | | MAC_BITE | MAC_ALARM |
| | GPIO2 (out) | - | - | - | GPS2_RST (default 0) | GPS_RST (default 0) |
| **AXI GPIO OSC** | GPIO (in) | - | - | - | - | 5721_STATUS |
| | GPIO2 (out) | - | - | - | - | 5356_OE (default 0) |
| **AXI GPIO SEL** | GPIO (in) | - | Clock Dcxo2 selected | Clock Dcxo1 selected | Clock MAC selected | Clock SMA selected |
| | GPIO2 (out) | Select PPS Source (0 GPS PPS, 1 PPS IN SMA) | Select Dcxo2 Clock (default 0) | Select Dcxo1 Clock (default 0) | Select MAC Clock (default 0) | Select SMA Clock (default 0) |

## 1.5 Register Description

The detailed register descriptions of the NetTimeLogic Cores are available in the reference manuals:

../TimeCard/Doc/

The Documentations of the Xilinx Cores are online available.

AXI Memory Mapped to PCI Express:
https://www.xilinx.com/support/documentation/ip_documentation/axi_pcie/v2_8/pg055-axi-bridge-pcie.pdf

AXI GPIO:
https://www.xilinx.com/support/documentation/ip_documentation/axi_gpio/v2_0/pg144-axi-gpio.pdf

AXI I2C:
https://www.xilinx.com/support/documentation/ip_documentation/axi_iic/v2_0/pg090-axi-iic.pdf

AXI UART 16550:
https://www.xilinx.com/support/documentation/ip_documentation/axi_uart16550/v2_0/pg143-axi-uart16550.pdf

AXI Quad SPI:
https://www.xilinx.com/support/documentation/ip_documentation/axi_quad_spi/v3_2/pg153-axi-quad-spi.pdf

AXI HWICAP :
https://www.xilinx.com/support/documentation/ip_documentation/axi_hwicap/v3_0/pg134-axi-hwicap.pdf

## 1.6 Clock Selector

The design can run on different source clocks. It's important that at least the source clock of the SOM module is always available. In the FPGA design an automatic source clock selection is running. The design detects if a clock is available and does then the selection base on following priorities:

1. External 10MHz Clock from SMA connector
2. 10MHz Clock of MAC
3. 10MHz Clock of DCXO1
4. 10MHz Clock of DCXO2

This selection can be overwritten by the AXI GPIO Sel GPIO2. When a clock is selected the FPGA checks if this clock is available. If the selected clock is not detected the automatic selection is applied.

Via the GPIO Sel GPIO it can be checked which one is the selected clock. If all values are 0 the full design is running from the SOM Module Clock.

## 1.7 Static configuration

The FPGA starts with a static configuration with following settings:

- PPS (including TOD) is used as correction input for the clock
- PPS Slave Pulse detection on rising edge

- PPS Slave cable delay 0
- TOD Slave UART Baudrate is 115200
- TOD Slave UART polarity default
- TOD Slave in UBX Mode, all GNSS and no messages disabled
- PPS Master polarity rising edge
- PPS Master cable delay 0
- PPS Master pulse width 100 ms
- Clock, PPS Slave, TOD Slave and PPS Master are enabled
- All Timestampers are disabled

# 2 Program FPGA and SPI Flash

For the initial programming of the FPGA and SPI Flash the JTAG programmer is needed and has to be connected to the USB JTAG.
After a successfully programmed FPGA, the design contains an AXI QUAD SPI Core which allows field updates.

## 2.1 Bitstreams with Fallback Configuration

The FPGA design in split into two different bitstreams/bin-files to allow a failsafe field update. The FPGA configuration starts always at Addr0 where the Golden image with the start address of the Update image is located. It jumps directly to this address and tries to load the Update image. If this load fails it falls back to the Golden image.

Details about this Multiboot/Fallback approach can be found in following Application Note:
https://www.xilinx.com/support/documentation/application_notes/xapp1246-multiboot-bpi.pdf

The Golden/Fallback image contains only a limited functionality which provides access to the SPI Flash. The second image is used for normal operation and it is the one which is replaced in a field update.

*Producation_TimeCard.bin*
This image contains two bitstreams and it shall be used to program the SPI flash for the first time as example in the production. The first bitstream is the Golden/Fallback image and the second the latest version of the regular image.

This combined image has following structure:

```
==================================
Configuration Memory information
==================================
File Format         BIN
Interface           SPIX4
Size                32M
Start Address       0x00000000
End Address         0x01FFFFFF

Addr1         Addr2         File(s)
0x00000000    0x002856FF    Golden_TimeCard.bit
0x00400000    0x0069B5AB    TimeCard.bit
```

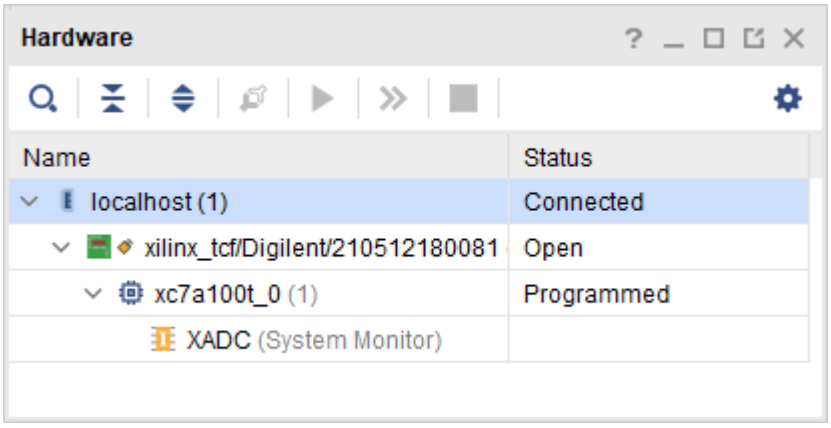
*TimeCard.bin*

This is the update/regular image and it shall be used for the field update via SPI. For the update this bitstream must be placed at 0x00400000 in the SPI flash.

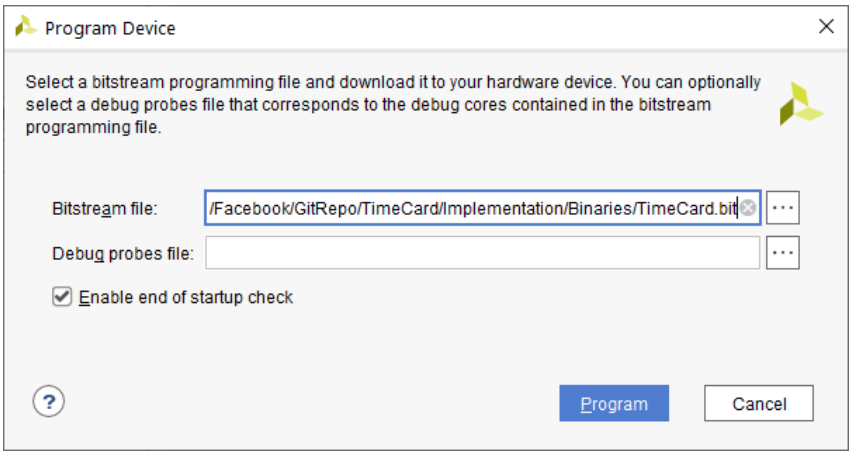
## 2.2 JTAG Programming (volatile)

This will only load the FPGA SRAM, after a power cycle this will be lost.

1. Go to the Hardware Manager in Vivado and Select “Open Target” → “Auto Connect”. After this step following view is available:




2. Right klick on “xc7a100t_0(1)”, a menu will popup
3. Choose “Program Device” from the menu, the following window will pop up

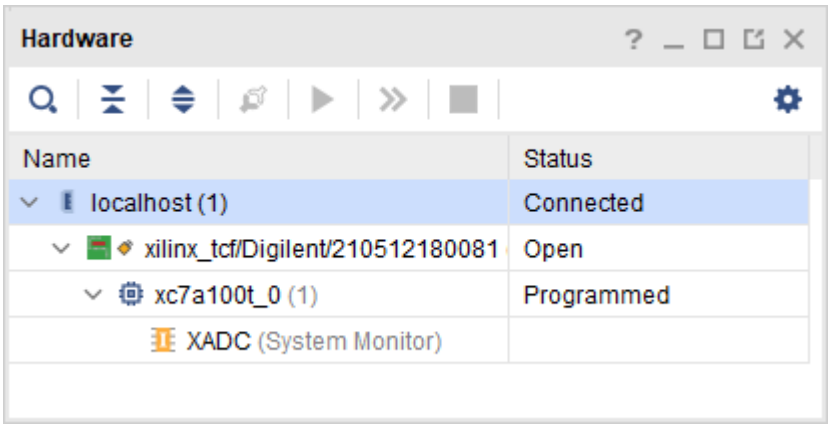



4. Select the bitstream you want to program:
   TimeCard.bit
5. Press Program and wait for completion
6. The RUN LED will blink

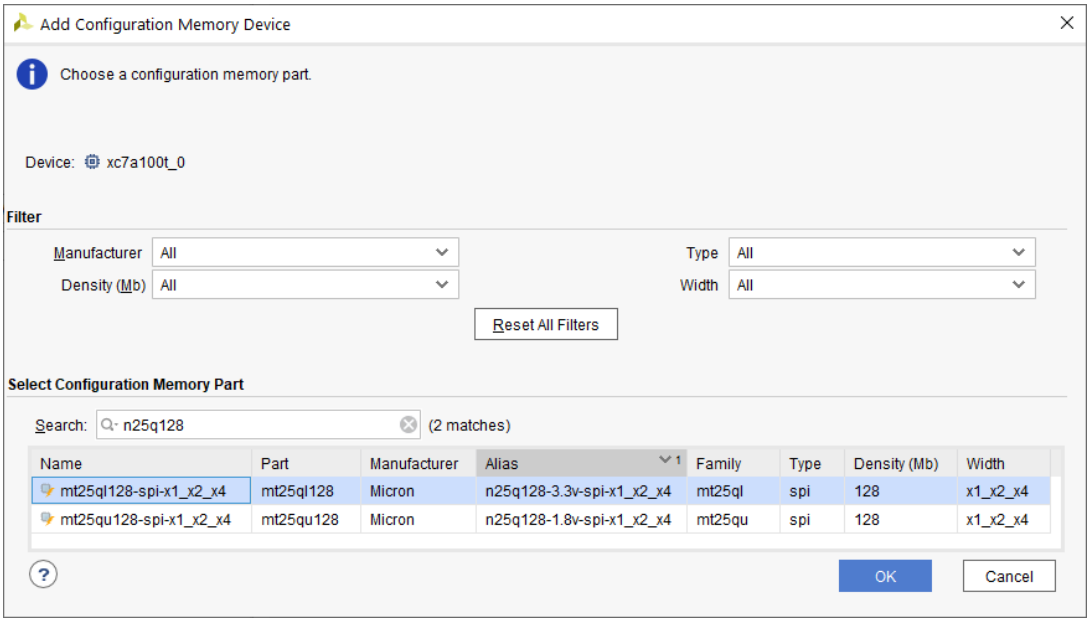
## 2.3 SPI Programming (non-volatile)

If no configuration memory was setup before start with step 1 otherwise start with step 7.

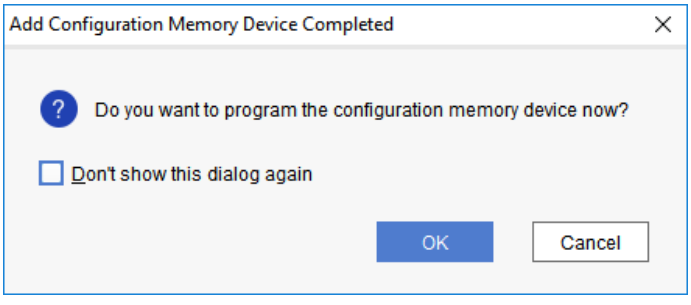
1. Go to the Hardware Manager Menu:




2. Right klick on “xc7a100t_0(1)”, a menu will pop up
3. Choose “Add Configuration Memory Device …” from the menu, the following window will pop up

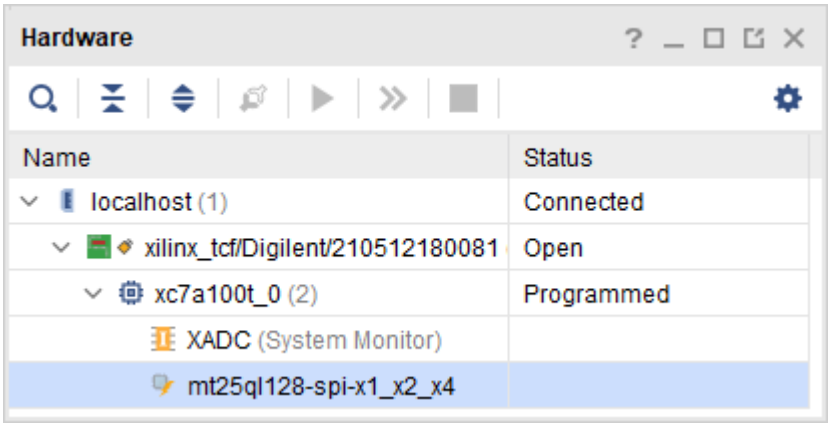



4. Select “mt25ql128-spi-x1_x2_x4” as the SPI Flash type
5. Press Ok, a new window will pop up:

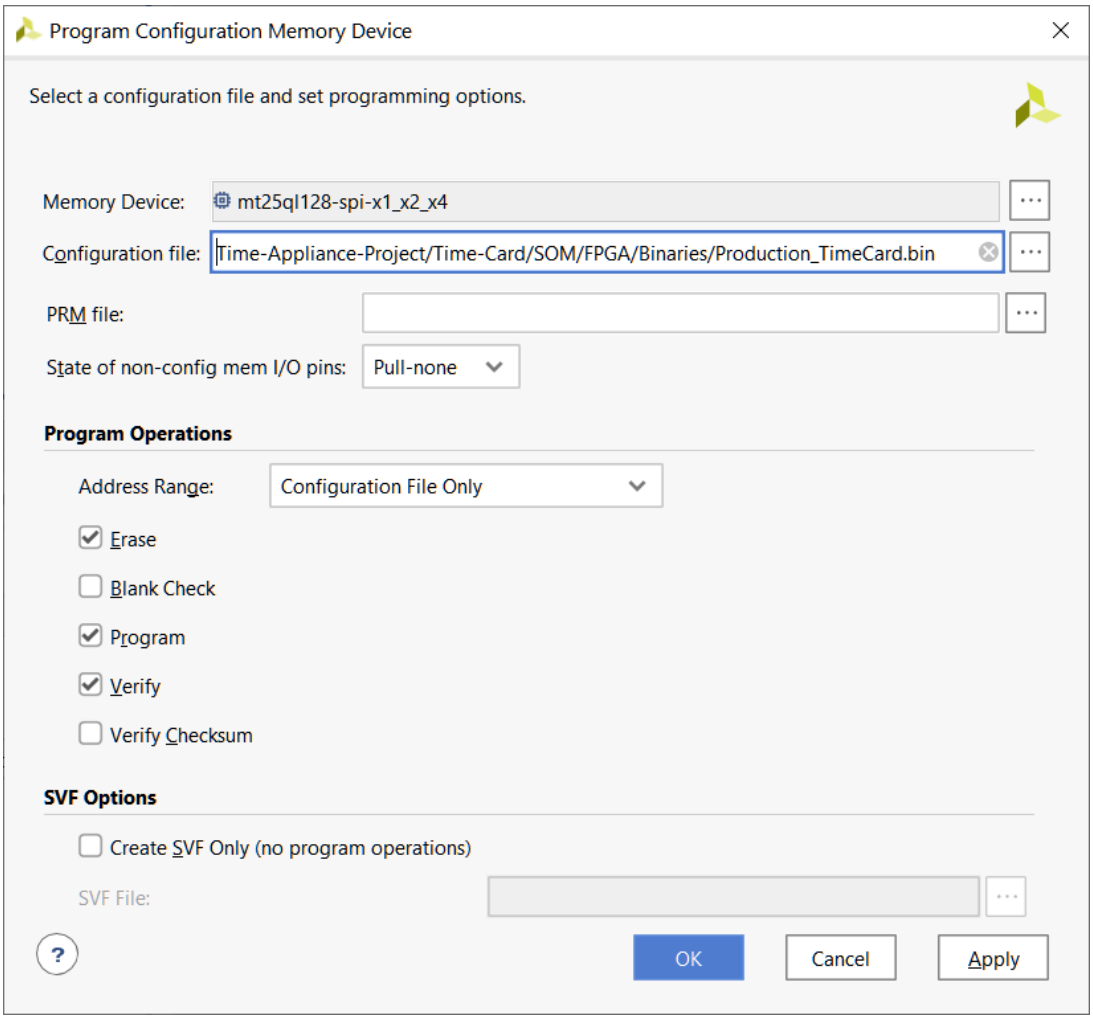



6. Press Cancel

7. Go to the Hardware Manager Menu which will have the flash attached:




8. Right klick on “mt25ql128-spi-x1_x2_x4”, a menu will pop up
9. Choose “Program Configuration Memory Device …” from the menu, the following window will pop up

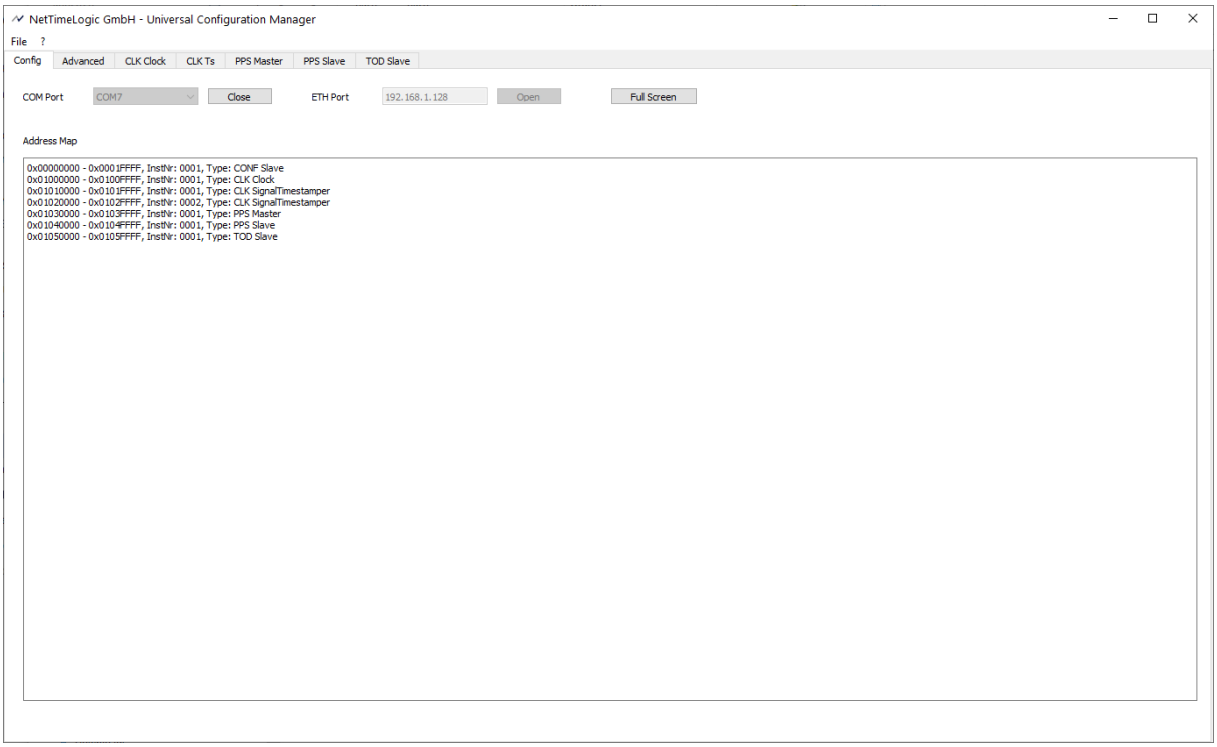



7. Select the bitstream you want to program:
   Production_TimeCard.bin
   IMPORTANT NOTE:
   If in this step the TimeCard.bin is loaded the field update as described above will not work!

8. Press Ok and wait for completion
9. Disconnect the JTAG interface from the board
10. Power cycle or Reset the board / Cold start of the PC
11. The RUN LED will blink

## 2.4 Connect NetTimeLogic Configuration Manager to FPGA

1. Connect the USB/UART to a Host PC running Windows.
2. Start NetTimeLogic's UniversalConfigurationManager Tool
3. Select the COM Port where the Board is connected to and press Open. Then the Address map of the cores instantiated are shown and five new Tabs appear (CLK Clock, CLK TS, PPS Master, PPS Slave and TOD Slave):




4. Change your configuration as you need it and press write. E.g.
   In the Advanced Tab it is possible to load configuration files by selecting a config file and pressing Load Config. Additionally, manual read or write of registers with the Field Address and Value is possible.

# 3 TestApp

The TestApp is used to do some basic testing of the Hardware. It basically uses mmap to the address space of the PCIe mapped address.
Before access two the PCIe end device is possible two steps are required. First the PCIe device must be detected by the system this can be checked with following command:

```
lspci -v
```

```
01:00.0 Memory controller: Xilinx Corporation Device 7011
        Subsystem: Xilinx Corporation Device 0007
        Flags: fast devsel
        Memory at 90000000 (32-bit, non-prefetchable) [disabled] [size=32M]
        Capabilities: <access denied>
```

The PCIe device is detected at address 0x9000_0000 but it is disabled. As a second step the device must be enabled:

```
sudo setpci -s "01:00.0" COMMAND=0x02
```

Now the TestApp is ready to start. The TestApp requires the PCIe base address as an argument:

```
sudo ./TestApp 0x90000000
```

The App reads the version of the NetTimeLogic IP cores and set the system time to the Adjustable Clock. After that it ready every second the time back from the Adjustable Clock:

```
PCIe Base Address is set to 0x90000000
Clock IP Core Version = 0x1020000
Signal TS IP Core Version = 0x1020001
Signal TS IP Core Version = 0x1020001
PPS Master IP Core Version = 0x1020000
PPS Slave IP Core Version = 0x1020000
TOD Slave IP Core Version = 0x2000001
Selected Clk Source is: PPS
Selected Clk Source is: REGS
Set the current local time and date: Fri Oct 23 14:00:35 2020
The time is: 14:00:35 and 3240 ns
The time is: 14:00:36 and 305900 ns
The time is: 14:00:37 and 490780 ns
The time is: 14:00:38 and 671200 ns
The time is: 14:00:39 and 1031680 ns
The time is: 14:00:40 and 1359140 ns
The time is: 14:00:41 and 1542180 ns
The time is: 14:00:42 and 1671980 ns
The time is: 14:00:43 and 1799580 ns
```