

Product guide of the imperix firmware IP

PN116 | Posted on March 26, 2021 | Updated on April 16, 2026



Benoît STEINMANN

Software Team Leader

imperix • in

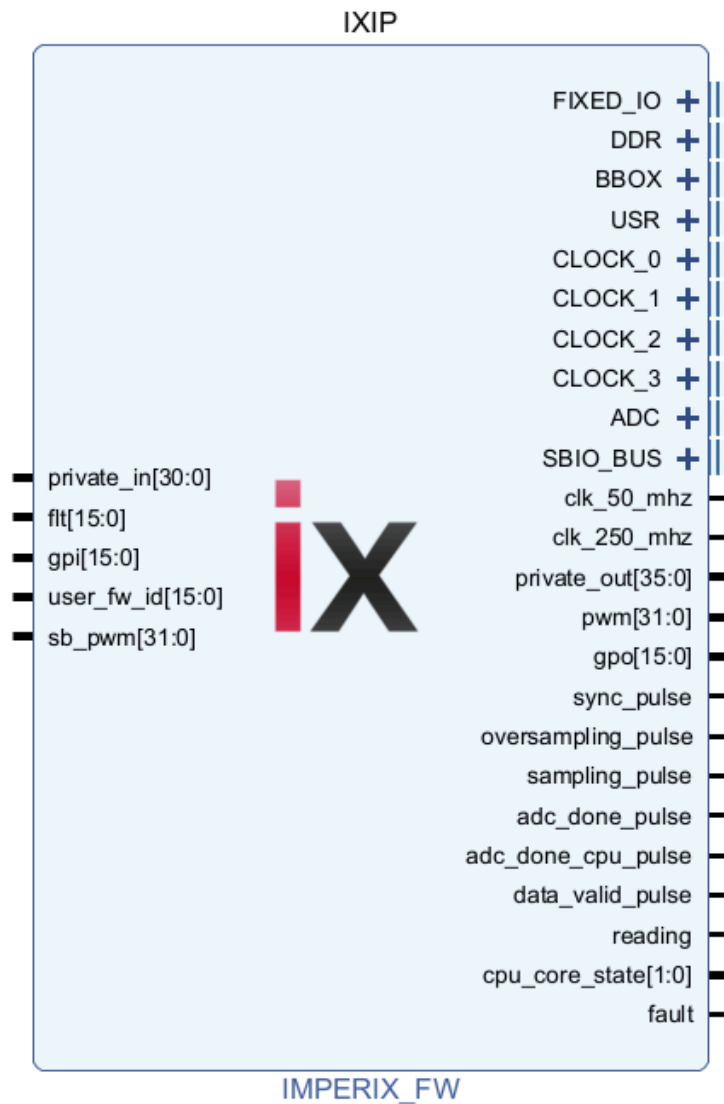
Table of Contents

- [Architecture overview](#)
- [Execution timing & signals](#)
- [Differences between Gen 3 and Gen 4 for FPGA development](#)
- [Imperix firmware IP interfaces description](#)
- [Disabling features to save FPGA resources](#)

This page documents the **imperix firmware IP** for AMD Xilinx Vivado, which is mandatory for programming the FPGA of imperix controllers. The IP encapsulates the imperix proprietary FPGA logic and offers various interfaces allowing users to add their own custom logic around it.

This document details the IP signals and associated timings. For guides on using specific features, the product notes linked in the table below may be consulted.

Interface	Feature	Product note
ADC	Retrieving ADC conversion results as soon as they are available.	PN126
SBIO	Exchanging data between the user code running in the CPU and the FPGA.	PN128
SB-PWM	Driving the PWM output chain, comprised of a dead-time generation system and the hardware protection mechanisms.	PN127
USR	Accessing physical 3V3 I/O pins.	PN179
GT	Accessing Gigabit Transceivers (GT), enabling support of protocols such as Aurora on SFP ports.	PN118
BSCAN	Using ILAs to observe FPGA signals in real-time for debugging purposes.	PN129



imperix firmware IP

The imperix firmware IP can be acquired from the [download](#) page. For developers beginning a new design, the [getting started with FPGA](#) guide provides the instructions for creating the Vivado sandbox template project and starting the development of custom FPGA logic.

Architecture overview

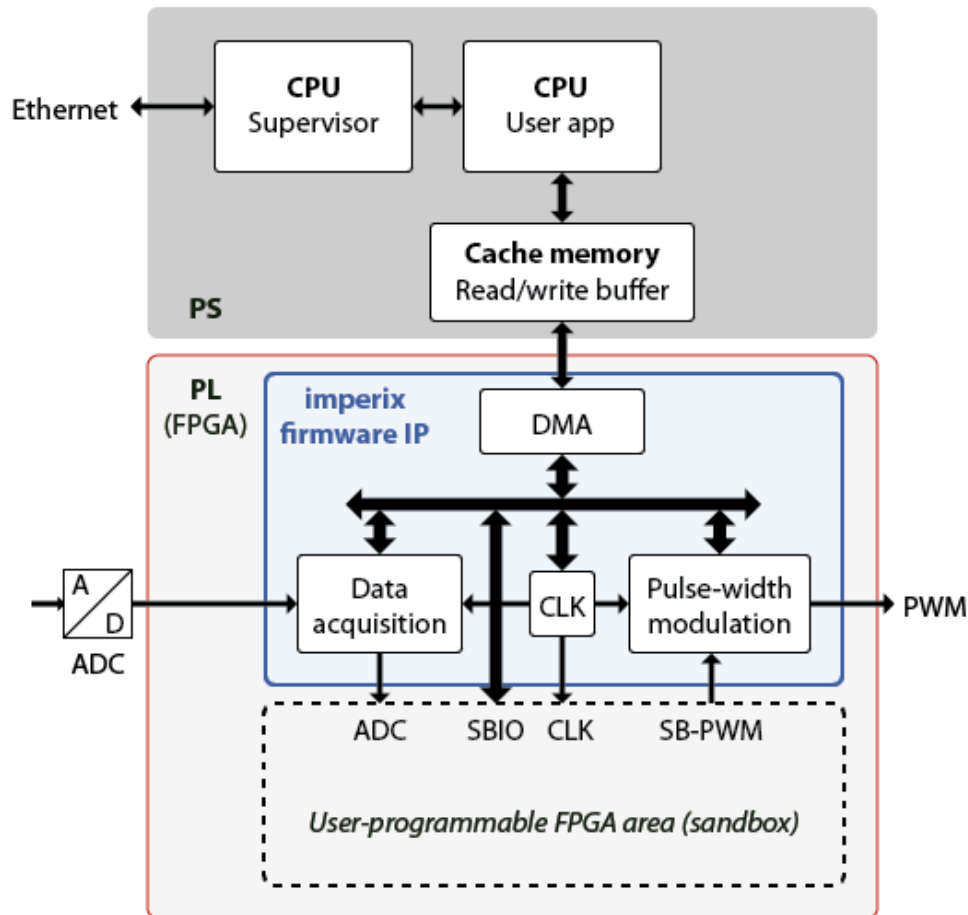
Because FPGA development operates at the hardware level, a proper understanding of the underlying system architecture and real-time execution behavior is essential to successfully integrating custom logic in an imperix controller FPGA.

Imperix is currently supporting two generations of controller, both based on AMD Xilinx System-on-Chips (SoCs):

- **Gen 3** controllers ([B-Box RCP^{3.0}](#), [B-Box Micro](#), [B-Board PRO](#), [TPI8032](#)) are based on a **AMD Xilinx Zynq 7000**
- **Gen 4** controller ([B-Box 4](#)) is based of an **AMD Xilinx Zynq Ultrascale+**

While the specific components (number of CPUs, CPU clock speeds, available FPGA resources, etc.) vary between the two generations, the core working principles and control

workflow remain identical. Specific generational differences regarding FPGA development are detailed later in this guide.



Architecture of imperix controllers

The image above depicts the overall architecture of imperix controllers. The key components are:

- The **supervisor CPU** runs on Linux. It is responsible for loading the application code, supervising system execution and managing data logging.
- The **user app CPU** runs on [BBOS](#). It executes the application-level control code developed by the user using the [ACG SDK](#) (Simulink or PLECS) or the [CPP SDK](#) (C/C++).
- The **imperix firmware IP** encapsulates the mandatory pre-implemented FPGA peripherals.

The illustration above shows a simplified breakdown of the core modules contained within the imperix firmware IP.

- The **data acquisition module** (documented in [PN126](#)) pilots the ADC chips and makes the analog input measurement available to both the CPU user app and the user-programmable FPGA area.
- The **pulse-width modulation** (documented in [PN127](#)) contains the pre-implemented PWM modulators ([CB](#), [SV](#), [DO](#), [SS](#), [PP](#)) and allows driving the PWM outputs from a custom modulator through the SB-PWM input.
- The **CLK** module manages the main clock CLK0 (configured via the [CONFIG](#) block), and the optional CLK1, CLK2 and CLK3 (configured via [CLK](#) block).

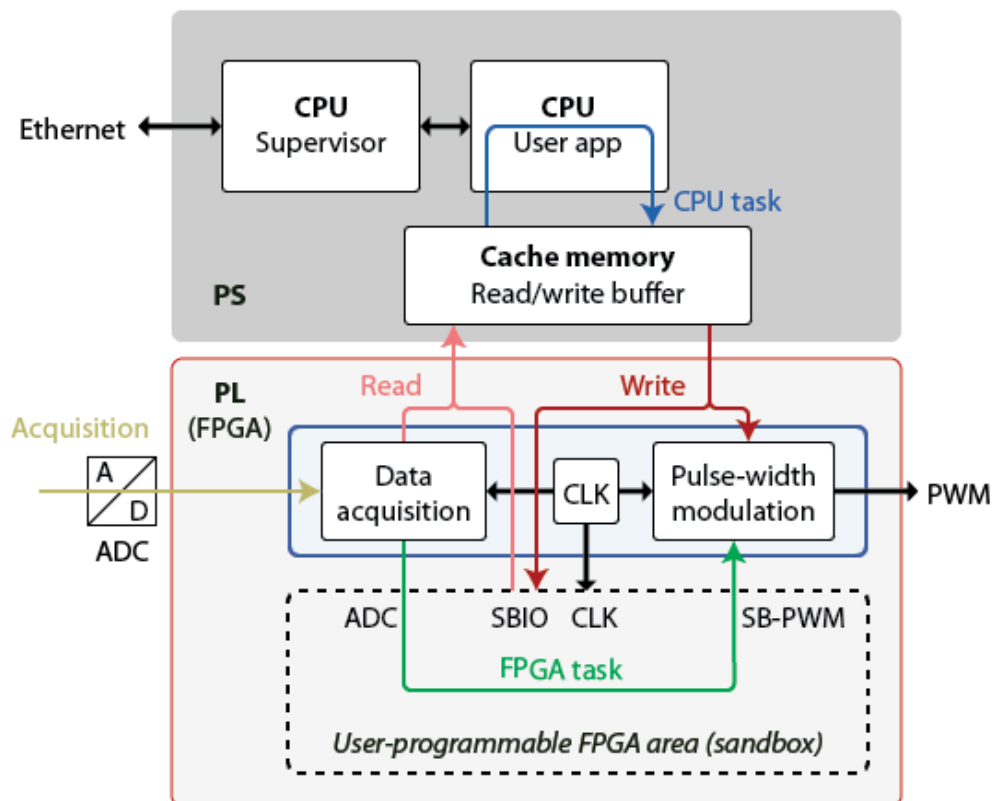
- The **DMA** is in charge of transferring the data between the FPGA and the CPU **read/write buffers**. The *read* and *write* phases are detailed in the next section.

Execution timing & signals

The following figure illustrates the phases involved in the **CPU control path** (acquisition, read, CPU task and write) and the **FPGA control path** (acquisition and FPGA task).

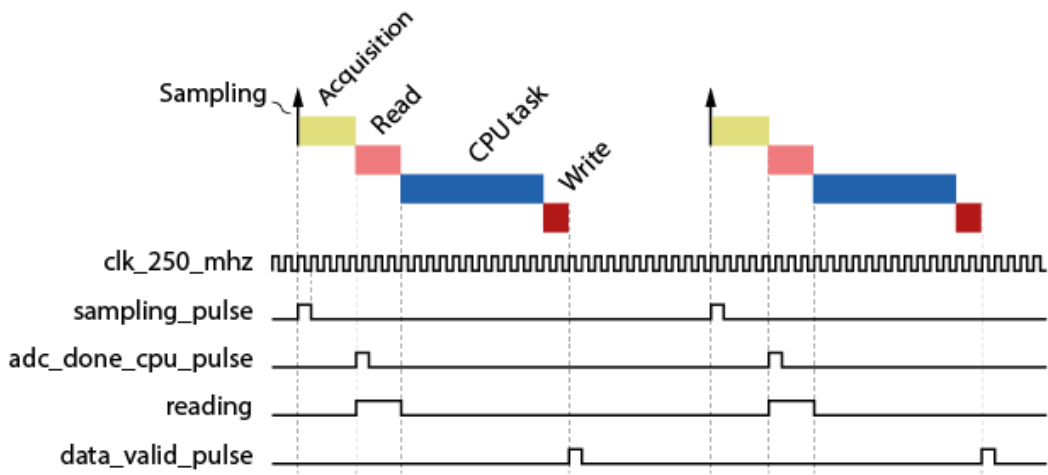
The execution phases are defined as follow:

- **Acquisition:** covers the physical Analog-to-Digital conversion (ADC) and the subsequent acquisition of the results to the FPGA fabric.
- **Read:** data flagged as real-time are transferred from the FPGA to the CPU read buffer.
- **CPU task:** the CPU reads the buffer and executes the main control task (the user's application control code). At the conclusion of this phase, the CPU write buffer is updated with new control values.
- **Write:** data flagged as real-time is transferred from the CPU write buffer back to the FPGA peripherals via DMA.
- **FPGA task:** control algorithm fully executed in the FPGA, such as the FPGA-based control of a grid-tied inverter presented in the [TN147](#) example.



Phases of the CPU and FPGA control paths

As illustrated in the schematic below, the imperix firmware IP provides several signals designed to monitor and synchronize logic with the execution of these different phases.



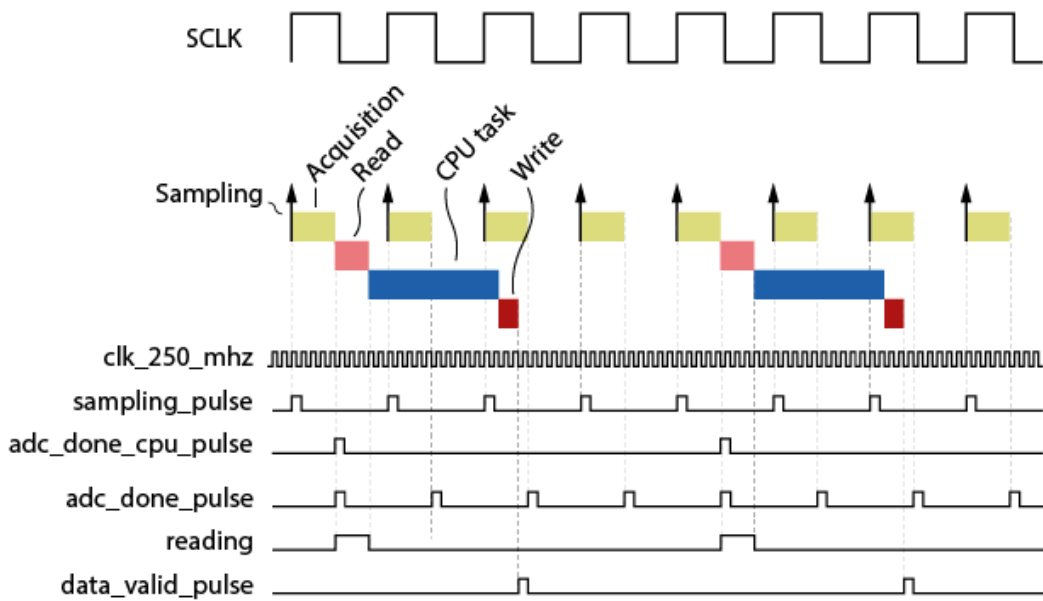
Execution timing signals

Port name	Clk domain	Description
sampling_pulse	clk_250_mhz	Corresponds to the rising edge of the sampling clock SCLK and indicates the ADC sampling instant and the start of the acquisition phase.
adc_done_pulse	clk_250_mhz	Indicates that the acquisition phases finished and new ADC values are available at the ADC interface.
adc_done_cpu_pulse	clk_250_mhz	When the sampling rate differs from the CPU rate, this signal is a decimated version of adc_done_pulse . It is set when the CPU control path stages are scheduled to execute. The following section describes this scenario in more details.
reading	clk_250_mhz	Indicates that the system is in the read phase, in which data flagged as real-time (SBI registers, ADC, GPI, etc.) are sent to the CPU read buffer.
data_valid_pulse	clk_250_mhz	Asserted at the end of the write phase, notifying that new data were written (SBO registers, PWM duty-cycles, GPO, etc.) .

Handling different execution rates

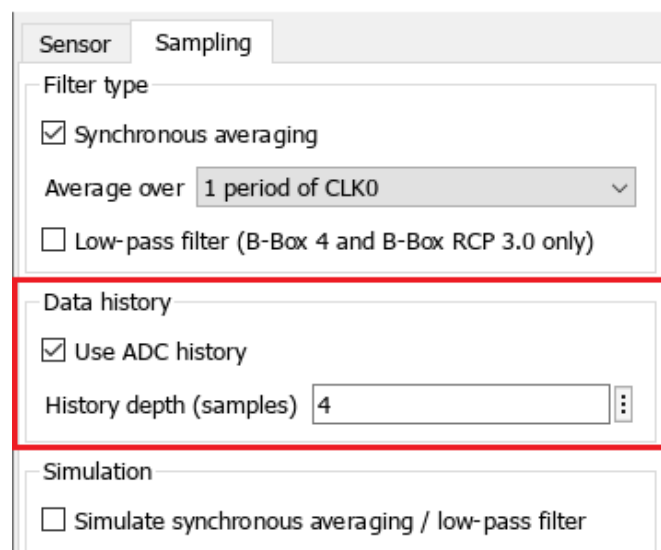
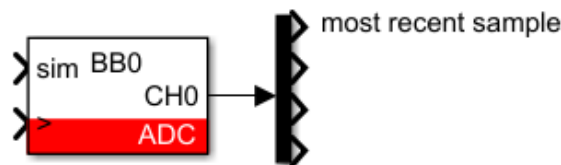
In many applications, FPGA-based control algorithms execute at a higher rate than the CPU task. In such case, a **postscaler** can be configured in the [CONFIG](#) block to reduce the execution rate of the CPU task relative to the sampling frequency.

When a postscaler is used, the `adc_done_cpu_pulse` signal acts as the decimated version of `adc_done_pulse`. It is asserted when the CPU control path stages are scheduled to run.

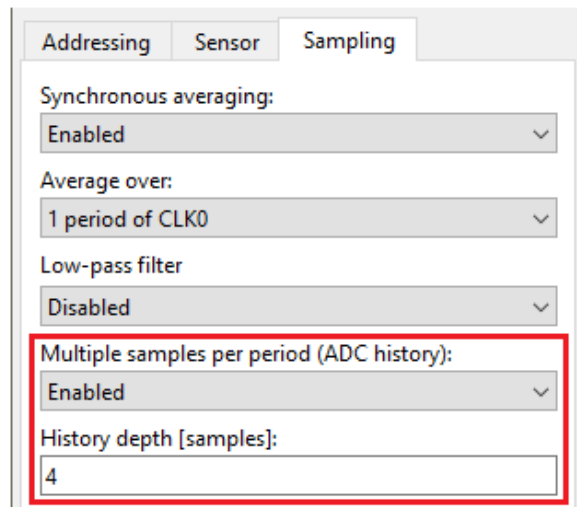
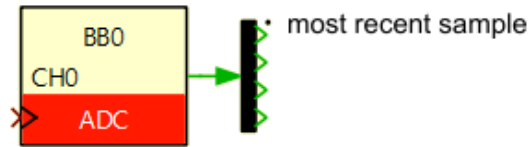


Execution timing with postscaler = 4

By default, the [ADC](#) block or driver only provides only the most recently sampled value. However, it is still possible to retrieve all ADC samples within the CPU using the **ADC history** feature, as illustrated below. This feature allows the retrieval of the N most recent samples (up to a maximum of 64).



Retrieving multiple values per CPU cycle in Simulink



Retrieving multiple values per CPU cycle in PLECS

CPP SDK users can implement this using the `Adc_ConfigureHistory` and `Adc_GetHistory` functions, as illustrated in the code snippet below:

```
tUserSafe UserInit(void) {
    // Sets CLK0 at 50 kHz
    Clock_SetFrequency(CLOCK_0, 50e3);

    // Sets a CPU postscaler of 4
    ConfigureMainInterrupt(UserInterrupt, CLOCK_0, 0.5, 4)

    // Setup a history of 4 samples for ADC0
    Adc_ConfigureHistory(ADC0, 4);

    // some other code...

    return SAFE
}

tUserSafe UserInit(void) {
    float s0, s1, s2, s3;

    s0 = Adc_GetHistory(ADC0, 0); // most recent sample
    s1 = Adc_GetHistory(ADC0, 1);
    s2 = Adc_GetHistory(ADC0, 2);
    s3 = Adc_GetHistory(ADC0, 3);

    return SAFE;
}Code language: C++ (cpp)
```

Differences between Gen 3 and Gen 4 for FPGA development

The Gen 4 introduces several key enhancements, including access to analog measurement in floating point format, faster transceivers, and expanded FPGA resources. The table below summarizes the key differences between the two generations.

Component	Gen 3	Gen 4
FPGA	Kintex 7 125K	Kintex US+ 504K
Logic	LUTs: 78600 Registers: 157200	LUTs: 230400 Registers: 460800
Logic used by imperix IP*	min: LUTs: 25482 (32.4%) Registers: 46374 (29.5%) max: LUTs: 39143 (49.8%) Registers: 67203 (42.7%)	min: LUTs 70733 (30.7%) Registers 108749 (23.6%) max: LUTs: 94233 (40.9%) Registers: 140636 (30.5%)
Usable logic left*	max: LUTs: 53118 (67.6%) Registers: 110826 (70.5%) min: LUTs: 39457 (50.2%) Registers: 89997 (57.3%)	max: LUTs: 159667 (69.3%) Registers: 352051 (76.4%) min: LUTs: 136167 (59.1%) Registers: 320164 (60.5%)
Gigabit Transceivers available from the sandbox	3x GTX (3x SFP connectors) Up to 6.6 Gbps	4x GTH (1x QSFP connector) Up to 16.3 Gbps
Bidirectional direct FPGA I/Os (USR)	36x	36x
PWM outputs	32x	48x
ADC interface	16x (int16)	24x (int16, or float)
RS485/RS422	–	2x
Acquisition delay	2 μ s (B-Box RCP 3.0) 500 ns (B-Box Micro, B-	200 ns / 368 ns**

* The logic used by the imperix IP depends on the enabled features, as explained in the *Disabling Features to Save FPGA Resources* section.

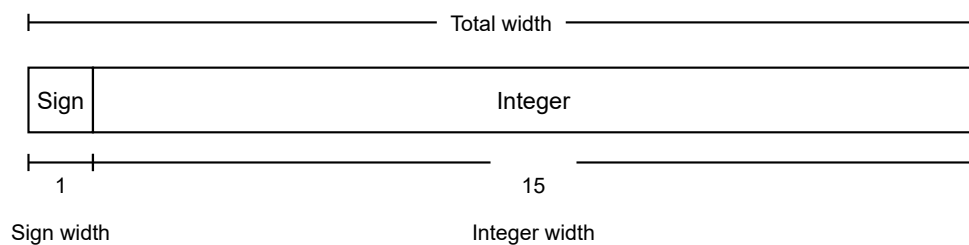
** When channels A12-A23 are used.

Imperix firmware IP interfaces description

ADC interface

The ADC interface returns the raw 16-bit signed integer result from the ADC chips.

[PN126](#) further describe how to retrieving ADC measurements from the FPGA sandbox.



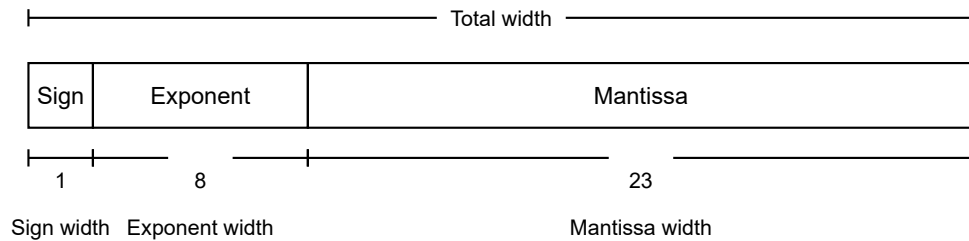
Format: 16-bit signed integer (two's complement, range: -32768 to +32767)

Port Name	Direction	Width	Clk domain	Description
ADC_reg_XX	Output	16bits	clk_250_mhz	ADC 16-bit result in 2's complement format. XX = channel
adc_done_pulse	Output	1 bit	clk_250_mhz	Indicates that new ADC samples are available.
adc_done_cpu_pulse	Output	1 bit	clk_250_mhz	Decimated version of adc_done_pulse when a postscaler is used. Set when the CPU control path stages are scheduled to run.

Description of signals related to the ADC interface

ADC_FLOAT interface

Available exclusively on the B-Box 4 platform, this interface provides the real physical values in single-precision floating-point format. The rescaling from the raw 16-bit ADC value is configured by setting the sensor sensitivity and offset from the [ADC](#) block. Results from this interface are delayed by 192 ns compared to the 16-bit ADC interface.



Format: 32-bit IEEE 754 single-precision floating-point

Port Name	Direction	Width	Clk domain	Description
ADC_FLOAT_reg_XX	Output	32bits	clk_250_mhz	ADC floating-point result post rescaling. XX = channel
adc_done_float_pulse	Output	1 bit	clk_250_mhz	Indicates that new ADC_FLOAT samples are available.
adc_done_cpu_float_pulse	Output	1 bit	clk_250_mhz	Decimated version of adc_done_float_pulse when a postscaler is used. Set when the CPU task is schedule.

Description of signals related to the ADC_FLOAT interface

SBIO interface

The **SBIO_BUS** (**SandBox IO Bus**) is a 16-bit memory-mapped bus allowing the CPU to address up to 1024 registers in the FPGA.

- The [SBI](#) block is used to read on the bus (FPGA → CPU)
- The [SBO](#) block to write (CPU → FPGA).

Direct interaction with the bus signals is usually not required since the imperix FPGA Sandbox template includes a suite of helper modules that abstract the bus complexity.

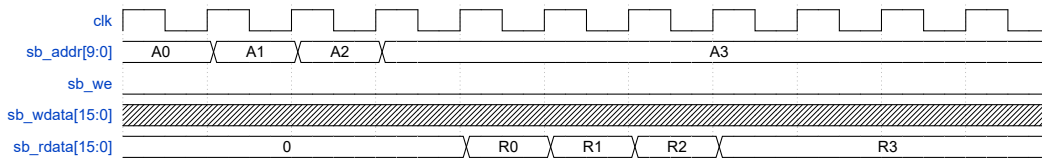
[PN128](#) further describe how to exchange data between the CPU and the FPGA.

Port Name	Direction	Width	Clk domain	Description
SBIO_BUS_sb_addr	Output	10 bits	clk_250_mhz	Address of the accessed register

SBIO_BUS_sb_we	Output	1 bit	clk_250_mhz	Write enable
SBIO_BUS_sb_wdata	Output	16 bits	clk_250_mhz	Write data
SBIO_BUS_sb_rdata	Input	16 bits	clk_250_mhz	Read data

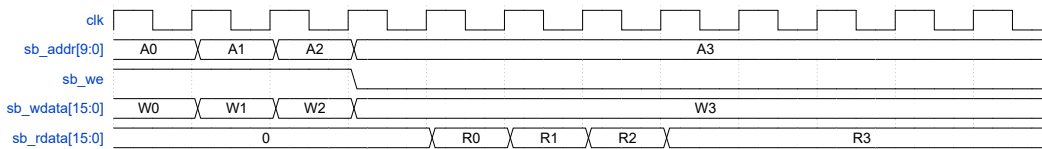
Description of the SBIO interface signals

The bus performs **read** operations by setting the requested address on sb_addr and expects the corresponding data on sb_rdata exactly four cycles later.



Read operation

A **write** occurs on each rising edge of clk when sb_we is asserted. The target location is given by sb_addr and the value written is sb_wdata.



Write operation

The two additional signals listed below are provided to help synchronize custom logic with the SBIO bus operations:

Signal	Direction	Width	Clk domain	Description
reading	Output	1 bit	clk_250_mhz	Asserted high while data (ADC, SBI, GPI) are being read by the CPU.
data_valid_pulse	Output	1 bit	clk_250_mhz	Single-cycle pulse indicating that all SBO registers have been written for the current CPU control task.

SB-PWM interface

The SB-PWM (**SandBox-PWM**) interface allows driving PWM outputs directly from custom FPGA logic.

The sandbox PWM inputs (`sb_pwm` port from the imperix IP) are considered as an additional PWM source, equivalently to the output of all built-in modulators. This means that the dead-time generation, activation and protection mechanisms are also available when driving the PWM from the sandbox and are configured from the [SB-PWM](#) block.

[PN127](#) further describe how to drive PWM outputs from the FPGA.

Port Name	Direction	Width	Clk domain	Description
<code>sb_pwm</code>	Input	48 bits (Gen 4) 32 bits (Gen 3)	<code>clk_250_mhz</code>	PWM signals driven from sandbox logic.

CLOCK interfaces

The CLOCK interface provides access to the 4 clock generator time bases. The main clock `CLOCK_0` is configured via the [CONFIG](#) block, `CLOCK_1`, `CLOCK_2` and `CLOCK_3` are configured via [CLK](#) blocks.

Port Name	Direction	Width	Clk domain	Description
<code>CLOCK_N_period</code>	Output	16 bits	<code>clk_250_mhz</code>	Period of the clock in ticks.
<code>CLOCK_N_prescaler</code>	Output	16 bits	<code>clk_250_mhz</code>	Indicates the <code>CLOCK_N_timer</code> ticking rate. 1 tick = 4 ns * <code>CLOCK_prescaler</code> .
<code>CLOCK_N_timer</code>	Output	16 bits	<code>clk_250_mhz</code>	Timer counting from 0 to <code>CLOCK_period-1</code> at the ticking rate set by the <code>CLOCK_N_prescaler</code> .
<code>CLOCK_N_clk_en</code>	Output	1 bit	<code>clk_250_mhz</code>	Clock enable pulse indicating when <code>CLOCK_N_timer</code> is updated.

CLOCK interfaces signal description, with N going from 0 to 3

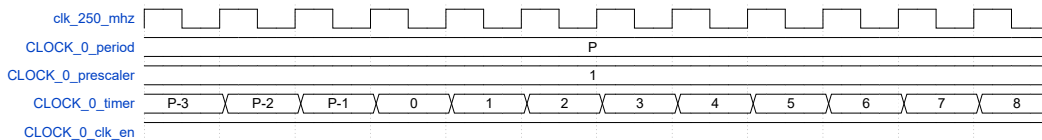
The **prescaler** extends the achievable frequency range beyond what the 16-bit period **timer** counter alone can provide. With `CLOCK_N_prescaler = 1`, the minimum achievable frequency is approximately 3.8 kHz (250 MHz / 65535). For lower frequencies, the `CLOCK_N_prescaler` automatically increases and the firmware tries to closely match the requested frequencies.

Example:

- A frequency of 20 kHz results in `CLOCK_prescaler = 1` and `CLOCK_period = 12500` ticks.
- A frequency of 2 kHz results in `CLOCK_prescaler = 2` and `CLOCK_period = 62500` ticks.

When prescaler = 1:

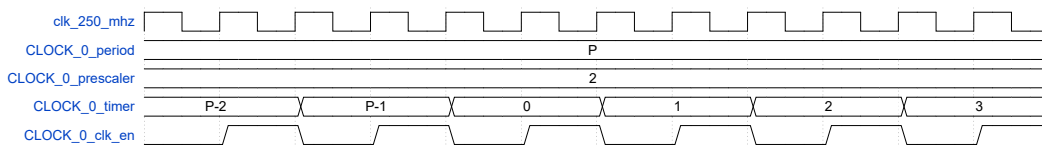
- `clk_en` is asserted every `clk_250_mhz` cycle
- `CLOCK_0_timer` increments on every clock edge



CLOCK_0 with prescaler = 1

When prescaler > 1:

- `clk_en` is asserted once every X cycles (where X = prescaler value)
- `CLOCK_0_timer` increments only when `clk_en` is high



CLOCK_0 with prescaler = 2

GT interfaces

The Gigabit Transceiver (GT) interface provides access to high-speed serial transceivers connected to the **SFP socked** of the controller. This enables custom point-to-point communication with external devices such as other controllers, custom hardware etc.

The [Aurora communication](#) examples details how to use the Aurora 8B/10B or Aurora 64B/66B protocol on imperix controllers.

Examples of SFP communication with third-party hardware-in-the-loop (HIL) simulators (OPAL-RT, Plexim and RTDS) are provided in [SFP communication with third-party devices](#).

Feature	Gen 4	Gen 3
Available transceivers	4x GTH (1x QSFP connector, lanes 4–7)	3x GTX (3x SFP connectors)
Maximum line rate	16.3 Gbps	6.6 Gbps

Accessing GTH transceivers on Gen 4 controller

On Gen 4, the QSFP lanes 4–7 are currently unassigned and can be used freely from the sandbox without affecting other functionalities. Lanes can be exposed to the sandbox from the **Configuration** tab in the imperix firmware IP configuration. Each enabled lane exposes the following signals:

Port Name	Direction	Description
GT_USER_TX_N_txp	Output	Transmit positive (differential)
GT_USER_TX_N_txn	Output	Transmit negative (differential)
GT_USER_RX_N_rxp	Input	Receive positive (differential)
GT_USER_RX_N_rxn	Input	Receive negative (differential)
gt_refclk	Output	250 MHz reference clock (shared across all lanes)

GT interface on Gen 4, where N is the lane number (4, 5, 6, or 7)

Accessing GTX transceivers on Gen 3 controller

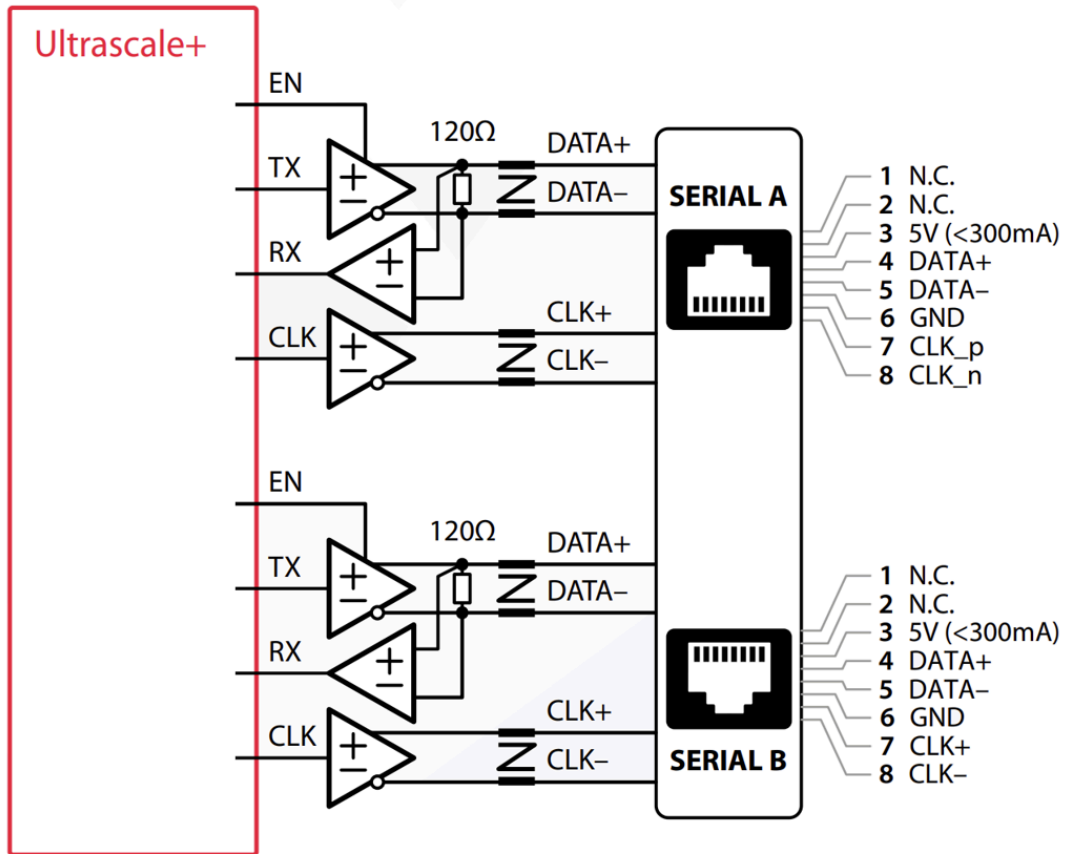
On Gen 3, the SFP ports are used by default for RealSync inter-device communication. Using a port from the sandbox requires disabling RealSync on that specific link, which removes the inter-device communication capability on that port. Lanes can be exposed to the sandbox from the **Configuration** tab in the imperix firmware IP configuration. Each enabled lane exposes the following signals:

Port Name	Direction	Description
txp_N	Output	Transmit positive (differential)
txn_N	Output	Transmit negative (differential)
rxp_N	Input	Receive positive (differential)
rxn_N	Input	Receive negative (differential)

Transceiver ports on Gen 3, where N is the lane number. 0 = SFP 0 (UP), 1 = SFP 1 (DOWN 0), 2 = SFP 2 (DOWN 1))

Serial port interfaces

The serial port interfaces are exclusive to the B-Box 4 and are typically used by the pre-implemented [SSI](#), [BiSS-C](#) and [EnDat2.2](#) FPGA drivers. The imperix IP allows bypassing these built-in drivers by providing direct access to RS485/RS422 transceiver signals from the sandbox, as illustrated below.



The serial ports (A and/or B) can be made available from the sandbox from the **Configuration** tab in the imperix firmware IP configuration. Each enabled serial port exposes the following signals:

Port Name	Direction	Description
SERIAL_X_rx	Input	Receive data from external device
SERIAL_X_tx	Output	Transmit data to external device
SERIAL_X_clk	Output	Clock signal for synchronous protocols
SERIAL_X_en	Output	Transceiver enable (active high)

Serial interface, where X is the port identifier (A or B).

BSCAN Interface to observe internal signals using ILA

The BSCAN (Boundary Scan) interface exposes the FPGA's debug scan chain to the sandbox and enables the use of Integrated Logic Analyzers (ILAs) and other Xilinx debug cores. Combined with the Xilinx Virtual Cable (XVC) protocol, this allows remote debugging of internal FPGA signals directly from Vivado's Hardware Manager over Ethernet.

For detailed instructions on adding ILAs to your design and connecting to them via XVC, refer to [How to debug an FPGA design](#).

Digital I/Os drivers

The sandbox exposes various digital I/O signals. Some ports are available for user logic, while others are reserved for internal system operation and must not be modified.

For detailed pin locations and electrical specifications, refer to the relevant datasheet:

- [B-Box 4](#) datasheet
- [B-Box RCP](#) datasheet
- [B-Box Micro](#) datasheet
- [B-Board PRO](#) datasheet
- [TPI 8032](#) datasheet

The USR pins

The USR pins are 36 user-configurable 3.3V I/Os that are routed directly to the FPGA, without any resistor or level shifter on the PCB trace. They used for interfacing external peripherals (e.g., SPI ADC).

To learn more refer to the [Accessing the USR pins in the FPGA sandbox](#) page.

Port Name	Direction	Description
USR[35:0]	Tristate	These pins are used by the motor interface and the programmable inverter . When none of these devices are used, the USR pins can freely be accessed from the sandbox.

Private Ports

These ports are required for internal communication with controller components. Modifying these connections may cause undefined behaviour.

Port Name	Direction	Description
private_in	Input	Internal system signals. Width varies by generation.
private_out	Output	Internal system signals. Width varies by generation.
DDR	–	DDR memory interface
FIXED_IO	–	Fixed I/O interface to PS

Gen 4 ports

Port Name	Direction	Description
din[23:0]	Input	General-purpose or fault feedback inputs (GPI/FLT)
dout[47:0]	Output	General-purpose outputs or PWM outputs (GPO/PWM)

Imperix **strongly discourages** the user from modifying the dout connection, as this would bypass the protection mechanism! Instead, PWM signals from the sandbox should be routed through the sb_pwm input port.

Gen 3 ports

Port Name	Direction	Clock Domain	Description
flt[15:0]	Input	clk_250_mhz	Fault inputs
gpi[15:0]	Input	clk_250_mhz	General-purpose inputs
gpo[15:0]	Output	clk_250_mhz	General-purpose outputs
pwm[31:0]	Output	clk_250_mhz	PWM output signals
BBOX[51:0]	Tristate	N/A	Internal B-Box RCP pins. Can be repurposed on B-Board PRO with custom PCB design (see PN201).

Imperix **strongly discourages** the user from modifying the pwm connection, as this would bypass the protection mechanism! Instead, PWM signals from the sandbox should be routed through the sb_pwm input port.

Disabling features to save FPGA resources

The imperix firmware IP includes several optional features that can be disabled to free up FPGA resources for custom sandbox logic. Access the resource-saving options from the **Saving FPGA resources** tab in the imperix firmware IP configuration.

Disabling a feature removes it entirely from the FPGA design. If the corresponding functionality is used in the CPU model, an error will be raised in Cockpit.

Unused PWM modulators can be removed to save logic resources. The CB-PWM logic can be disabled in groups of 8 lanes.

Feature	Gen 4	Gen 3
CB-PWM lanes 0–7	✓	✓
CB-PWM lanes 8–15	✓	✓
CB-PWM lanes 16–23	✓	✓
CB-PWM lanes 24–31	✓	✓
CB-PWM lanes 32–39	✓	–
CB-PWM lanes 40–47	✓	–
SS-PWM logic	✓	✓

PP-PWM logic	✓	✓
--------------	---	---

Resource Savings

Feature	Gen 4	Gen 3
CB-PWM (per 8 lanes)	~2100 LUTs (0.9%) ~2800 Registers (0.6%)	~1600 LUTs (2.03%) ~2000 Registers (1.27%)
SS-PWM	~6250 LUTs (2.71%) ~5500 Registers (11.93%)	~2700 LUTs (3.43%) ~3300 Registers (2.1%)
PP-PWM	~4400 LUTs (1.9%) ~9200 Registers (2%)	~4400 LUTs (5.6%) ~9150 Registers (5.82%)