User Manual
FPGA-based control system
5/5/2023, buffer board versions v1.2-v1.4

1. Introduction

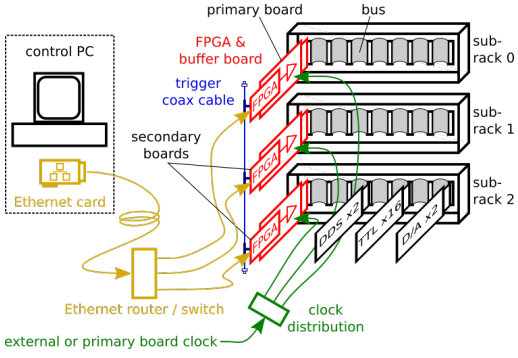
The FPGA-based control system is a replacement and upgrade of the outdated and not anymore supported DIO-64 card from Viewpoint Systems (referred as 'old control system'). While the DIO-64 card must be installed into a PCI slot of the control computer and is connected via a ribbon cable to the hardware rack, the Cora-Z7 is a single-board computer (embedded system) which runs independently of the control computer and is directly installed in the hardware rack. Data between the control computer and the Cora-Z7 is exchanged via Gigabit Ethernet. A rack contains one or several sub-racks which contains the electronics devices for controlling the experiment, like digital and analog outputs and direct-digital-synthesizers (DDS). Each sub-rack contains one FPGA board which communicates via the backplane bus of the sub-rack with these devices.

This configuration has several advantages over the previous system: the control computer does not need to have a device driver for the board installed, which permits to use any operating system of choice. The control software does not need to communicate with the device driver directly, which allows to choose from a broader spectrum of control software and facilitates development. The only requirement is the ability to communicate via Ethernet. During the execution of the experiment, the control computer is much less busy than before where the driver had to update rather small DMA buffers for the DIO-64 card over the rather slow PCI bus. Now the control computer recieves during the experimental run only status information at a programmable rate (typically 20Hz) and waits for the experiment to finish. This significantly reduces performance demands on the computer, allows to do data analysis directly on the control computer and might even increase its lifetime. On the hardware side no outdated PCI slot is required, the control computer can be at a much larger distance from the experiment and electronic isolation is intrinsically ensured via Ethernet's decoupling transformers (called "magnetics").

The FPGA control system consists of the low-cost FPGA development board (Cora-Z7-07S or Cora-Z7-10 from Digilent Inc.), and of a buffer card which hosts the FPGA development board and buffers to interface with the hardware in the sub-rack. This buffer card replaces the former opto-coupler card inserted into the sub-rack. The actual versions of the control system (v1.2-v1.4) have essentially the same firmware, only the buffer card differs. Version 1.2 is a simpler version with fewer buffers, versions 1.3 and 1.4 allow to control two sub-racks with a single FPGA development board (but with two buffer cards). Version 1.4 vs. 1.3 has optional decoupling resistors and a differential clock signal after the clock buffer into the FPGA development board. Both features are for better resilience against external perturbations like electrical spikes which can cause the external clock signal to be lost for short times.

Several FPGA development boards can be synchronized on the 10ns level or better. For this purpose the buffer card provides edge connectors with external clock input and output and trigger input and output. All boards can be locked to one common external clock. Alternatively, one of the boards provides its internal clock to the other boards. This so-called primary board also sends a trigger signal to all other boards via a single coaxial cable to synchronous start execution of an experiment.

Figure 1, system overview



Run-time delays of the trigger signal and clock phase shifts can be corrected by software. The largest system so far tested consists of two development boards at a separation of 30m and synchronization on the 1ns could be achieved with a more advanced automatic synchronization scheme (see Ref. [1]).

See the glossary for the used terms and additional information.

2. Prerequisites

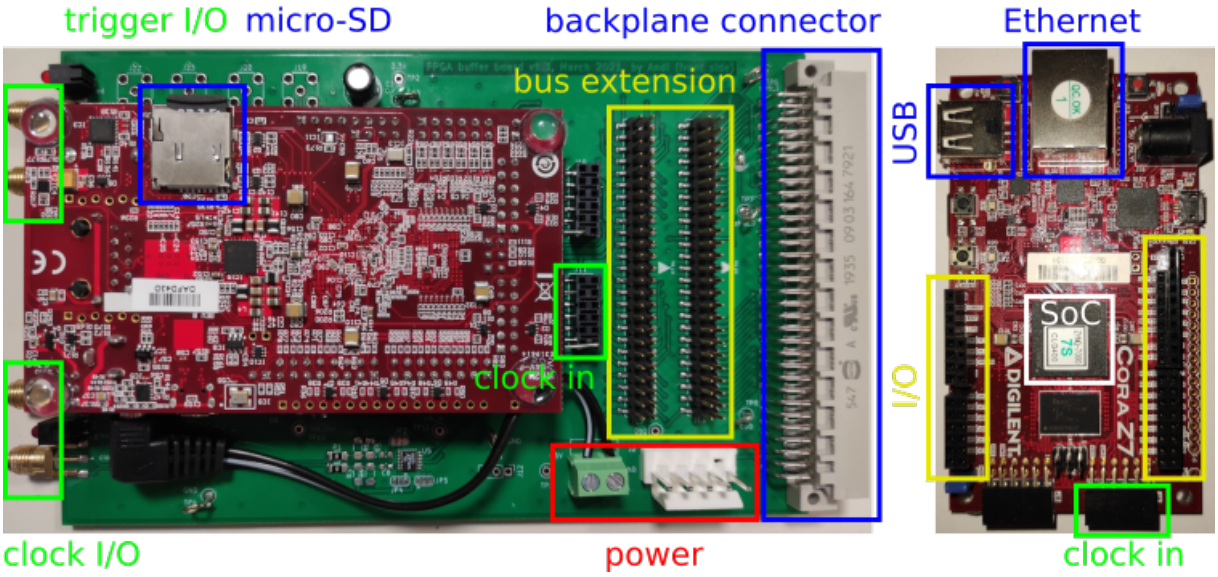
Basically, each sub-rack requires one FPGA board and one buffer card. When two sub-racks are less than about 3m apart it is possible to control two sub-racks with a single FPGA development board. In this case still two buffer cards version 1.3 or 1.4 are needed and a flat-ribbon cable (2x32, 1.27" spacing, IDN-code...) with two headers (female) have to be provided to connect the two racks (for details see section 4.2 below). Each FPGA board needs an Ethernet cable and one SD card (8GB is more than sufficient, use high-quality cards like the ones recommended for Raspberry Pi) and a short power supply cable with 4.0mm jack at 90° (RS code...). See table 1. If more than one FPGA development boards are used, the boards need to be synchronized for which additional two coaxial cables (with each 2x SMA connectors) and an Ethernet switch or router is needed to communicate between the control computer and all development boards.

Table 1
bill of material per FPGA development board (controls 1x sub-rack + optional 2nd sub-rack nearby)
1x Cora-Z7-10 or Cora-Z7-07S
1x buffer card v1.2-v1.4 (v1.3 or v1.4 needed when controlling 2nd sub-rack)
1x SD card (high quality)
1x power supply cable (>10cm, 4.0mm jack, 90°)
1x Ethernet cable
optional in addition to control a second nearby sub-rack:
1x buffer card v1.3 or v1.4
1x 2x32pin flat-ribbon cable, 1.27" spacing
2x 2x32 female headers
1x Ethernet cable
1x Ethernet switch or router

3. Buffer board

A custom PCB buffer board is needed to interface and buffer the commercial FPGA board with our standard hardware.

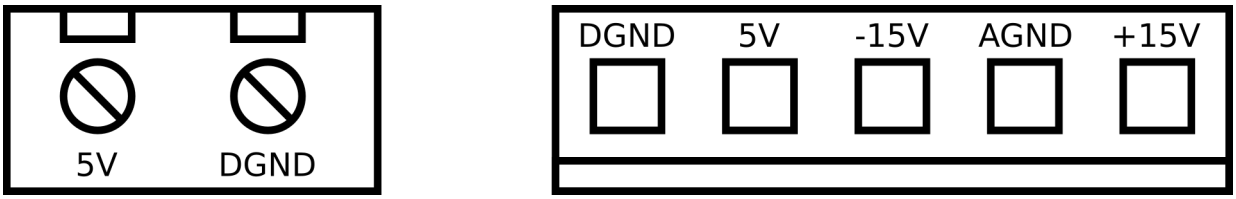
Figure 2 left shows the buffer board (version 1.3) with the FPGA board mounted on top, right shows the FPGA board from the front side:



The FPGA board is equipped with Arduino type headers (yellow I/O) which are used to stack it on top of the buffer board (green PCB) and to interface with it. The buffer board is inserted into one free slot of the sub-rack and the backplane connector (blue) is used to supply power and the digital data to the sub-rack.

3.1 power:

Figure 3 shows the power connections of the buffer board.

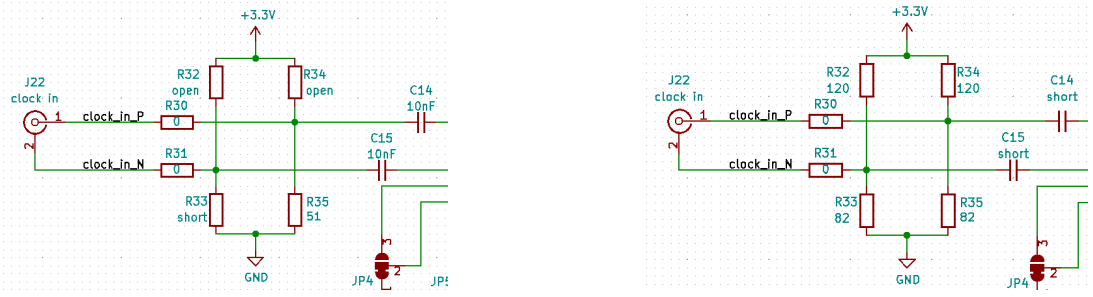


The FPGA board and the buffer board are powered via the two power connectors (red in Fig. 2) from the power supply of the sub-rack. This gives (on the 5-pole connector from left to right) digital ground, 5V, -15V, analog ground, +15V. The (2-pole connector from left to right) screw terminal provides the 5V and digital ground to the FPGA board via the 90° jack (4.0mm, outside ground, inside 5V) cable. Note that the backplane connector of the buffer board also supplies the power to the backplane bus. ATTENTION: Avoid hot-plugging of the buffer board and any of the devices of the sub-rack! I.e. always power off the sub-rack when inserting or removing any device in or out of the sub-rack.

3.2 external clock:

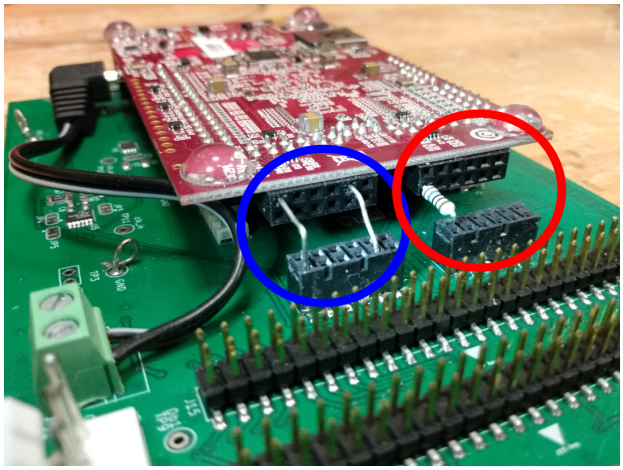
The buffer board has two SMA connections for the external clock (left-bottom green in Fig. 2) input (bottom) and output (top). The external clock input is needed to synchronize several boards in time (see Synchronization section below) or if a well-defined time scale is needed. If no external clock is provided, the 50MHz oscillator of the FPGA board is used (for more details see clocking section below). The clock output is used if no external clock is available and one or several boards are used. In this case the primary board provides the trigger signal and also the clock signal to the other boards. The clock input accepts 10-300MHz with default frequency of 10MHz. At the moment the frequency cannot be configured by software, so in case this needs to be changed please ask for the firmware update. The input clock can be either a sine wave (0.5-2Vpp, 50Ω) centered at 0V offset voltage or 3.3V LVPECL standard (0.8Vpp with 2.0V offset, 50Ω). The two different signals need different input resistors/capacitors, see Figure 4 for the sine input (left) and LVPECL input (right) for the buffer board version 1.4. Other buffer board versions are equivalent, only different part labels are used. The clock output is always a 3.3V LVPECL signal.

Figure 4, external clock input configuration for buffer board version 1.4. Left for sine wave input, right for LVPECL input.



The buffer board has a clock buffer (LTC6957) to get a square wave signal out of the sine wave input and to shift the level of the signal adequate for the FPGA board. The output of the buffer must be fed into the FPGA board via the PMOD JA connector (green "clock in" in Fig. 2; J11 on buffer board v1.4). For the buffer board version 1.2 and 1.3 this is done as shown by the blue circle in Fig. 5: use two ca. 0.7mm thick wires to connect the left-bottom pin on the PCB side with the left-top pin on the FPGA board and the 2nd-to right-bottom pin to the 2nd-to-right-top pin on the FPGA board. For buffer board version 1.4 use on the PCB the SMB connector J24 to connect a short coaxial cable to the left-top pin on the FPGA board and the next pin right of this for the inner and outer conductors of J24 respectively. When the FPGA board locks to the external clock the LED between the clock SMA connectors and a blue LED on the FPGA board are switched on.

Figure 5, clock input (blue circle) and strobe_1 output (red circle) at FPGA board.



3.3 input/outputs:

The buffer board has two SMA connections for external trigger (left-top green in Fig. 2) input (top) and output (bottom). The trigger signal is used to synchronize the starting time for several boards (see Synchronization section below). When a new experiment should be started, the primary board sends the trigger to the secondary board(s), waits a software-configurable time until the secondary board(s) have received the trigger and all boards start execution of the experiment synchronously. The required waiting time can be measured by oscilloscope and can be set as the "wait" option of the server.config file. When the trigger option is not used these I/O pins can be used as additional general-purpose I/O ports (see next paragraph).

Depending on the buffer board version it has additional general-purpose digital inputs and outputs. Buffer board version 1.3 and above have 2x I/O ports in addition to the two trigger I/O signals. These additional I/O's are located on the PCB above the FPGA board (see figure 2). All digital inputs/outputs use LVC standard, i.e. they accept 3.3V input but can be used also with 5V input without damage. The outputs are pulled up to 5V with 2.2kΩ resistors. These I/O's are not designed for fast switching with 50Ω load.

All digital I/O's can be configured by software (including the trigger ones) and can be used to represent certain states of the board or can be programmed to be controlled by certain data bits. See Section I/O configuration for details.

3.5. LEDs:

The buffer board has 3 LEDs on the edge which allows to fast check the status. During booting of the board the two LEDs on the top are switched on. After about 20 seconds they are switching off and the board is ready to communicate with the control computer. During the experimental run the green LED is switched on and switches off at the end of the cycle. If there is an error the red LED is switched on. At the moment the red LED switches off fast since the control software automatically resets the board, such that the red LED is usually only seen briefly. The third LED, located on the bottom between the clock SMA connectors is either green or blue and indicates the presence of an external clock.

All 3 LEDs have corresponding red, green and blue LEDs located on the FPGA board.

3.6. controlling two sub-racks from a single FPGA board:

One FPGA board can control two sub-racks when they are not too far from each other (about 3m). For this option either a second buffer board version >=1.3 is needed or an old opto-coupler card used for the DIO-64 card can be used. Starting from version 1.3 the buffer board is equipped with additional buffers which allow to connect a ribbon cable (2x32 pin, 1.27" spacing) from the "daughter connector" J21 (in version 1.4, J3 in v1.3) to a second buffer board J21 (J3) which is inserted into the second sub-rack and does not need to be equipped with a FPGA board. On this second buffer board the buffers which are driving the daughter connector J21 (J3) need to be disabled by setting the solder jumper JP3 (v1.4, JP2 on v1.3) from ground to 5V. This way the signal from the first FPGA board is fed into the second buffer board where another set of buffers ensure the signal is refreshed before it is put on the bus. Additionally, the FPGA board provides two independent strobe signals which can be used to control the two sub-racks independently with individual adjustable timings and optionally to use individual device address space (see 3.6).

When the two sub-racks are too far apart or when the electrical isolation is an issue or more than two sub-racks need to be controlled then it is recommended to use independent FPGA boards in each sub-rack, see Section 4 below.

3.7 pseudo-clock (strobe) selection:

The FPGA board provides two independent pseudo-clocks signals to the buffer board (labeled strobe_0 and strobe_1*; the labels might be inverted). With solder jumper JP1 the pseudo-clock signal put on the backplane bus can be selected. This allows to have independent timings of two sub-racks and to separate the address space. The default is strobe_0. When the strobe_1 should be used in either of the sub-racks then this signal needs to be connected as indicated by the red circle in Figure 5. A decoupling resistor of 100-200Ω can be used to connect the left-top pin on the PCB side to the left-bottom pin on the FPGA side.

*strobe_1 is at the moment not implemented.

The selection of the strobe bit in the I/O configuration (see below) determines independently which of the two strobe bits is enabled and which data bit is used. Both strobe bits can use the same data bit, which is by default the 8th address bit in the data. But the control software can also use one of the free data bits to generate an additional strobe bit which is useful if two sub-racks have to be controlled (see section 3.6 above). In this case the address space of 0-127 addresses is independent for the two sub-racks. If the same data bit is used for the strobe then the address space is shared between the two sub-racks.

The pseudo-clock signal is generated by the FPGA board from the strobe bit in the data sample or independently (as selected by software). When the strobe bit is used then the strobe bit must toggle state for each sample. If the strobe bit does not toggle, the sample is not put on the backplane bus. The first sample is always put on the bus (unless the NOP bit is set) regardless of the first strobe bit.

Irrespective if the strobe bit is disabled, the same or different for the two sub-racks, the timing for the generated pseudo-clock can be adjusted independently (see section configuration file below). The timing is defined by the setup time, which is the time between the data lines changing and the rising edge of the clock, and the hold time, which is the time how long the clock remains high. For the FPGA board the sample and hold times can be adjusted by software with 10ns resolution. Typical values for the bus output sample rate of 1us is 300ns sample time plus 300ns hold time.

For high output sample rates of 50MHz or more the pseudo-clock cannot anymore be generated in this way and the pseudo-clock must be configured to toggle its state instead of generating a short pulse for each data transmission. This reduces the clock rate to the output sample rate but requires that the hardware triggers on the rising and falling edge of the pseudo-clock (this is know as DDR mode) but which has to be supported by the hardware.

4. synchronization of several boards:

To synchronize several boards a common time base (clock) and start signal (start trigger) needs to be used. An external clock can be provided to all boards or one board, called the primary board, provides its internal clock to the other boards. The primary board also provides the start trigger to the other boards. The buffer board already is supplied with clock input and outputs and the start and stop trigger signals with SMA connectors on the edge of the PCB (see buffer board section above). Connect the clock and trigger outputs from the primary board to the secondary board clock and trigger inputs. Although not tested much, several secondary boards can be clocked and triggered using power splitter along the clock and trigger coaxial cables.
The propagation delay of the trigger signal and the phase shifts of the external clock can be compensated with a software configurable waiting time and phase shift. After the primary board has generated the start trigger, or after the secondary board has received the start trigger, it will wait for the a specified time before it starts executing the experiment. The waiting time is configured in units of 10ns. Additionally, the clock phase to which the PLL of each secondary board is locked can be fine-adjusted and allows a resolution of better than 1ns.

The waiting time and phase shift can be saved into the configuration file on the SD card and it can be set during the configuration of each experimental run. See Section configuration file below.

5. configuration file:

In addition to the two firmware files on the micro-SD card there is the server.config text file which allows to configure the IP address and several other parameters for each board individually. Apart of the IP address and port all other parameters are optional and can be configured also by software. The software programmed values take precedence. The file is a simple text file which can be edited with any text editor. It is recommended to start with one of the provided default configuration files and to modify this. The sharp symbol '#' is used to comment lines. After each parameter name follows the equal sign '=' and the value. Do not insert quotation marks or spaces for the values.

Table 2 contains the configuration entries. Except of the IP address all others are optional.

parameter	default value	description
info	Cora-Z7-10 (primary)	arbitrary information string printed on the console during startup
IP	192.168.1.1	static IP v4 address given as a.b.c.d with a,b,c,d decimal integers 0-127.
port	49701	port number where server listens. integer 0-65535
strobe_0 strobe_1	3:4:3:1	pseudo-clock 0 or 1 timing parameters given as s:h:r:level. s = setup time, h = hold time and s+h+r = 1/sample rate, resolution is 10ns. level = 1 or 2. if level = 1 pseudo-clock is high during hold time (s) and low during setup time and remaining time (h+r). if level = 2 pseudo-clock is toggling state after setup time (s).
wait	0	waiting time in units of 10ns before the board generates data on the bus after the start trigger is generated or received.
primary	1	1 or 2. if 1 the board is the primary board, if 2 it is the secondary board. The primary board generates the start trigger, the secondary board waits for it.
Cora_Z7	10	10 or 7. if 10 the board is the Cora-Z7-10, if 7, the board is Cora-Z7-07S. This defines how the server optimizes reading data from the Ethernet and writing the data to DMA memory. The difference is only appreciable for large amount of data.

6. I/O configuration:

Up to 3 external inputs (buffer board version >= 1.3) can be used to generate trigger events and special data bits can be used to modify the timing or to define special actions. See Table 3 below. Up to 3 external outputs (buffer board version >= 1.3) can be configured to represent internal signals or data bits. Also all LEDs on the edge of the buffer board and on the FPGA board can be configured with different signals [not yet implemented]. See Table 4 below. There are dedicated configuration sections for inputs and outputs. Each configuration has a structure of: destination = source + level_or_bit. Each destination has one source assigned with the given level or data bit. The same source can be assigned to several destinations, but not vice versa. These configurations are directly represented by the ***ctrl_in*** and ***ctrl_out*** registers. See register section below.

Table 3, input configuration

destination	source	level or data bit
start trigger	None, input 0/1/2	rising/falling edge, high/low level
stop trigger	None, input 0/1/2, data bits 20-23/24-27/28-31	rising/falling edge, high/low level, offset bit 0/1/2/3
restart trigger	None, input 0/1/2	rising/falling edge, high/low level
NOP bit	data bits 20-23	offset bit 3
IRQ bit	None, data bits 20-23/24-27/28-31	offset bit 0/1/2/3
STRB 0 bit		
STRB 1 bit*		

For the destinations see section "external triggers" and "special data bits" sections below. Source "None" means, that this option is disabled, input are the external inputs 0-2, data bits are the bits in the 4 data bytes of each sample for each sub-rack. If data bits are selected as source then the corresponding offset bit in the level or data bit entry must be selected, otherwise the input level must be selected which defines if the event is edge triggered or level triggered. The NOP bit cannot be changed or disabled, the strobe STRB 0 and STRB 1 are explained in the section about the pseudo-clock. *STRB 1 bit is not implemented so far.

Table 4, output configuration

destination	source	level or data bit
output 0	fixed, sync_out, sync_en, sync_mon, clock lost, error, run, wait, ready, restart, start trigger, stop trigger, restart trigger, data bits 20-23/24-27/28-31 [not implemented]	low/high level, offset bit 0/1/2/3
output 1		
output 2		
bus enable 0		
buts enable 1		

The outputs define which external output 0-2 are configured and the two bus enable signals are some additional signals which should be kept fixed with the default values. The possible sources are: fixed gives a constant output at the given level, sync_out is the start trigger generated from the primary board to start the secondary board(s), sync_en and sync_mon are debugging signal for the start trigger, clock_lost is active when the external clock was used and is lost during experiment, error is active when an error happened, run is active during the experiment run (not reset during

wait state), wait is active when the board waits for the start or restart trigger, ready is active when the first data is loaded in the FIFO buffer and not end state, restart toggles state when board has restarted in cycling mode, start/stop/restart trigger toggle state when start/stop/restart trigger was detected. Data bits 20-31 can be selected as source. In this case the level cannot be inverted but the data bit offset must be selected. For the other signals the level entry selects if the signal is active high or low.

7. clocking:

The internal time base of the FPGA board is f_{system} = 100MHz, i.e. 10ns per cycle. The bus output sampling rate is generated by dividing f_{system} by the content of the clock divider register f_{bus} = 100MHz/clk_div[7:0], i.e. when clk_div[7:0] = 100 the bus output sampling rate f_{bus} = 1MHz. The maximum tested f_{bus} = 10MHz. For >30-40MHz the board cannot sustain continuous output at the maximum rate, but short bursts of data output should be possible. However, the normal strobe does not work at these frequencies and one has to use toggle trigger (see section pseudo-clock above).

The external input and output clock frequencies are at the moment hard-coded for 10MHz. Other frequencies can be chosen but require compiling of the firmware. In a future version the frequency can be programmed by software.

8. External Triggers:

Similar to the old control system the board has start and stop triggers, an additional restart trigger can be defined in order to distinguish between the initial start of the board and the restart after the stop trigger. All trigger can be activated only for the first time the trigger condition is fulfilled (name with '1x') or for every time (name without '1x') [at the moment this option is not implemented and all triggers are every time active]. All triggers can use any of the external inputs, the stop trigger can be activated in addition by a programmable data bit in the data section of the sample (any of the bits 20-31). When the board is stopped (run and wait bits are set in status register) it can be restarted with the restart trigger or by flashing the run bit by software (using DIO24_STOP/DIO24_RUN). Alternatively, the execution of the sequence can be aborted (using DIO24_RESET) by software.

Several or all triggers can use the same input as the source and the levels can be different or the same.

a) ***start trigger***: when enabled after the board is started the first time (with DIO24_START), the board waits for the start trigger condition (external input) before executing the first command. During waiting time the ready, and wait bits in the status register are set. When the board is running the run bit is set in addition. In cycling mode the board waits at the beginning of each new cycle for the starting condition.

b) ***stop trigger***: when enabled the experimental sequence can be stopped by the stop trigger condition (external input or a data bit). The clock_div[15:8] register can be used to define the stop trigger window. clock_div[15:8] = 0 or 1: no window, i.e. the board stops immediately. clock_div[15:8] > 1: window time t_{wnd} = 10ns*clock_div[15:8]. The board stops at the next integer multiple of t_{wnd} after the stop trigger. E.g. for 1MHz bus output rate (clock_div[7:0] = 100) and clock_div[15:8] = 50, the board stops within the next 0.5us, i.e. at half of the bus output period $(2*f_{bus})^{-1}$. Using the stop trigger window increases the responds time to the trigger, but allows to keep several boards synchronized even with different trigger cable lengths and noise. Measurements

showed that t_{wnd} = 50-100ns gives good results in many situations. When the board is in the stopped state the run and the wait bits are both set. To restart the board either a restart trigger must be enabled or the run bit in the control register must be reset and set again (using DIO24_STOP and DIO24_START). Note that during the stopped state the board can be re-configured by software. To abort and not restart the board reset the board (using DIO24_RESET).

c) ***restart trigger***: same as the start trigger (external input), but applies only after stop trigger is active. This allows to have different input and/or levels for start and restart.

9. special data bits:

Special data bits can be configured with the input control register (***ctrl_in***). See section I/O configuration. The NOP and the STROBE 0 and STROBE 1 bits allow to ignore data, the STOP bit can be used to stop execution at a specific instruction and the IRQ bit can generate an interrupt at the specific instruction. Further bits can be implemented on request.

When the NOP bit is set the sample for this sub-rack is ignored. This allows to temporarily disable a sample without deleting it from the data which is more efficient than to move data in memory. This bit is used by the driver to mark data which it inserts for properly align buffers and can therefore not be changed or disabled. The NOP bit only prevents that the data is put on the bus, but it does not affect the timing or the other special data bits like the strobe still has to toggle state from the sample with the NOP bit to the next sample in order to execute the next sample.

When one of the two STROBE bits is enabled and not toggling state with respect to the previous sample, then the sample of the corresponding sub-rack is ignored. The first sample is always used regardless of the state of the toggle bit, unless the NOP bit is set. This prevents that even or odd number of samples impact if the first sample is executed, which was a problem with the old DIO-64 system. The existence of the strobe bit has historical reasons since it was used to generate the pseudo-clock when it was toggling state. It is supported for compatibility to the old DIO-64 system, but requires the control software to keep track of even and odd lines, especially when samples are inserted or removed. In the new design these bit can be disabled and the NOP bit can be used instead.

If the STOP bit is set the execution is stopped after the specified time given in the timestamp of the sample. If the NOP bit is set the data is ignored, otherwise it is put on the bus before the board stops.

If the IRQ (interrupt) bit is set then an interrupt is generated on the FPGA board and is transmitted to the control software which can detect it with the FPGA_irq bit set in the status register. This is useful if the control software has to execute some code at a specified time during the experimental sequence. For example, GPIB or USB controlled devices can be programmed with such a (software) interrupt at the specified time. This is much more efficient and jitters less in time than polling by software for the actual time of the experimental cycle.

10. Glossary:

buffer board:
custom printed circuit board which interfaces the FPGA board with the sub-rack. it contains buffers for the I/O signals and to generate and input external clock and trigger signals. At present there are 3 versions available: 1.2, 1.3, 1.4.

console:
The FPGA board has a micro-USB plug on which debug information of the board can be monitored in realtime from an external computer. You need some serial communication program like minicom (Linux) or PuTTY (Windows). Additionally you can execute many commands of the Linux operating system and I have created several testing applications for the FPGA and the control system. This feature can be very useful for debugging the system. Access is not password protected. ATTENTION: do not execute any of the testing applications when the control system is attached to actual hardware! Some tests create large and random data at high data rates on the bus! Before you want to do this please contact me.

CPU:
central processing unit. The FPGA-SoC has a CPU part and a logic part (see FPGA below). The CPU executes software in a linear (command by command) way and might be interrupted by interrupts (IRQ's). A classical CPU is not useful for time-critical operations (there are special microcontrollers for such applications). The CPU makes it easy to control the system via software while the FPGA part takes care of fast and precise timing.

cycling mode:
the experimental sequence can be repeated for a given number of repetitions (cycles) or infinitely (until user sends the stop command).

DDR vs. SDR:
double data rate vs. single data rate. For single data rate the data is read on the bus for each rising edge of the pseudo-clock. This requires that the pseudo-clock frequency is twice the sample rate and the pseudo-clock has a phase shift (the setup time) between when the data is changed and the rising edge. For double data rate the data is read on the bus for each rising and falling edge of the pseudo-clock. This allows to have the same frequency between the pseudo-clock and the data change but the phase shift still has to be maintained. The hardware must be capable of DDR.

DSP:
digital signal processing/processor, the FPGA part has several of such processors which allow complex manipulation of data and floating point arithmetics. In the current firmware this is not used but can be implemented when needed.

DMA:
direct memory access. This mode allows the hardware to read and write data in memory without involving the CPU. This allows much faster data rates but needs dedicated hardware channels, mapping and locking of dedicated memory sections and according support from the operating system and special care in the device driver.

experimental sequence, experimental cycle, experiment:
a sequence of experimental samples describing what should happen for each time.

firmware:
The collection of information defining the hardware and software configuration of the FPGA board. This is saved on the SD (secure data) card of the FPGA board and is loaded during booting of the board. It contains two files: BOOT.BIN which contains the bootloader and the bit stream which defines the hardware logic in the FPGA part of the FPGA-SoC chip. image.ub is the Linux image which contains the file system for the Linux operating system. The SD card contains a third file server.config which is a text file containing the IP address and other configurations of the FPGA board.

FPGA, FPGA board, FPGA-SoC board, FPGA-SoC chip:
commercial, low-cost development board with a field-programmable-gate-array (FPGA) and a CPU and dedicated hardware interfaces merged on a single chip into an embedded system-on-a-chip (SoC). The FPGA part is an ensemble of a huge number of hardware logic gates, flip-flops, memory, PLL's, DSP's etc. which allows to perform parallel complex logic operations with high frequency and well-defined timing. On the CPU part a simple embedded Linux operating system is executed which allows to execute arbitrary user software in a user-friendly environment and to easily interface with the hardware. The FPGA part appears as external hardware with which the user software like the TCP/IP server can interact via a custom driver using registers or via file access to send and receive data.

IRQ:
interrupt. Typically a hardware signal which tells the CPU to immediately stop what it did before and to execute a dedicated interrupt service routine for this specific interrupt. This allows to break the sequential execution of a CPU to mimic parallel execution, like performing a long-lasting calculation while still responding to user interrupts like mouse clicks. They are essential for modern computer systems but can be problematic for real-time applications where precise timing is needed.

NOP bit:
Special data bit in the sample. For the present firmware this is the 31bit (highest bit) and cannot be disabled since the driver in the FPGA board relies on this bit to mark padded samples which should be disregarded. If this bit is set the sample is disregarded.

PCB:
printed circuit board. The buffer board is a custom PCB with standard 160mm x 100mm x 1.6mm Eurocard size which can be conveniently inserted into the standard 19" sub-rack. See Ref. 2 for the production files.

sample:
One line in the code describing the experiment. The FPGA board supports 8 bytes/sample and 12 bytes/sample. The first 4 bytes is the timestamp, the second 4 bytes contains the data of the first sub-rack and the optional second 4 bytes contains the data of the second sub-rack. The 4 data bytes (32 data bits) of each sub-rack contains 16 data bits and 7 address bits which are directly put on the backplane bus of the sub-rack. The address bits define which device in the sub-rack should use the 16 data bits to change its state accordingly. An additional strobe bit is used to generate the pseudo-clock and the remaining 8 data bits can be used for other purposes like a NOP bit (no-operation) and a second strobe bit.

sample rate:
update rate of the backplane bus in the sub-rack in Hz. Typical value is 1MHz, which gives 1us temporal resolution. The maximum sample rate of the FPGA board is limited by the maximum DMA (direct memory access) data transmission rate of about 340Mbytes/s (see Ref. [1]) which gives a maximum sample rate of 40MHz for 8 bytes/sample. For short time the sample rate can be

increased until the FIFO buffer (first-in-first-out; typically 16k samples) on the board becomes empty. Rates above 50MHz require a change of the firmware and maybe also of the design of the sub-rack backplane bus. Rates of 200MHz on 8 digital channels are possible using reduced number of data bits and without transmission of timestamp in contiguous sample mode with one sample per tick. Generation of 1ns short pulses at a fixed repetition rate have been demonstrated.

PLL:
phase-locked loop, allows to generate from one clock another clock with a different frequency but fixed phase relation to the original clock. The FPGA board can use an external 10-300MHz input clock to generate its internal 100MHz clock used to output the data on the bus. The external clock gives a well-defined time base and allows to synchronize in time several board.

pseudo-clock:
Contrary to a "normal" clock signal the pseudo-clock is not running all the time but only when something should change. Besides of this, the pseudo-clock still resembles characteristics of a clock like that its frequency is twice of the sample rate and that it has a well-defined sample and hold time. Historically, the pseudo-clock signal on the bus is sometimes called "strobe".

strobe bit:
One data bit of the sample which can be enabled and selected for each sub-rack. If this is enabled the pseudo-clock signal is generated when this bit is toggling, otherwise the sample is ignored. If this bit is disabled for each sample the pseudo-clock signal is generated (unless the NOP bit is set). Historically, the pseudo-clock signal on the bus is sometimes called "strobe".

tick/timestamp:
the first 4 bytes of each sample contain the timestamp which is an unsigned integer counting time in units of tick = 1/sample rate where the sample rate is typically 1MHz, i.e. tick = 1us.

trigger:
external TTL signal which tells the FPGA board to start, stop or pause

TTL signal:
transistor-transistor-logic, rather old 5V digital signal level standard used in most of our experiment.

11. References:

[1] A. Trenkwalder et.al., "A flexible system-on-a-chip control hardware for atomic, molecular, and optical physics experiments", Rev. Sci. Instrum. 92, 105103 (2021), https://doi.org/10.1063/5.0058986

[2] https://github.com/INO-quantum/FPGA-SoC-experiment-control. Please contact me to get latest updates.

Appendix I. registers:

The FPGA part has 9 control and 9 status registers which are used to configure and monitor the execution. Each register contains 32 bits.

1. control register (***ctrl***):
This is the main control register which enables options and resets, starts and stops the experiment.

bit	name	description
0	ctrl_reset	when set board is reset. auto-reset after 10ns.
1	ctrl_ready	set by driver when device is opened.
2	ctrl_run	when set board is started. when reset during run board goes into wait state until is set again.
3	-	not used
4	ctrl_restart_en	when set board is restarted after end of cycle if num_cycles = 0 or > 1.
5	ctrl_as_en	when set waiting time for board synchronization is enabled
6	ctrl_as_prim	when set and ctrl_as_en is enabled sync_out pulse (start trigger) for secondary board is generated
7	-	not used
8	ctrl_bps96_en	when set 12 bytes/sample, otherwise 8 bytes/sample
9	ctrl_bps96_brd1	when ctrl_bps96_en is set indicates first (0) or second (1) sub-rack
10	ctrl_clk_ext_en	when set use external clock, otherwise internal clock. status_clk_locked must be set before enabling this bit.
11-14	-	not used
15	ctrl_err_lock_en	when set and external clock is used and lock is lost stops execution with error (default behavior). when not set and external clock lock is lost does not stop execution but switches automatically to internal clock and continues execution. This bit allows to ignore if the external clock is lost and to continue execution. ATTENTION: if this bit is disabled and clock is lost the timing of the experiment might not be correct and synchronization between several boards is lost! The status_err_lock bit is in any case indicating if the lock was lost during the experiment.
16-19	-	not used
20	ctrl_IRQ_en	when set all interrupts are enabled. The driver uses this bit to disable and reset interrupts.
21	ctrl_IRQ_end_en	when set status_IRQ_end is enabled
22	ctrl_IRQ_restart_en	when set status_IRQ_restart is enabled
23	ctrl_IRQ_freq_en	when set status_IRQ_freq is enabled
24	ctrl_IRQ_data_en	when set status_IRQ_data is enabled
25-31	-	not used

2. input control register (***ctrl_in***):
This register defines the input options used. See section I/O configuration. The register is divided into 6 sections of 5 bits each containing dest_bits[4:0] = {level_bits[1:0], source_bits[2:0]}.

Input destination bits:

ctrl_in bits	destination
4:0	start trigger
9:5	stop trigger
14:10	restart trigger
19:15	NOP bit
24:20	STROBE 0 bit (STROBE 1 not implemented)
29:25	IRQ bit

Input source bits:

source_bits[2:0]	source
0	None
1	input 0
2	input 1
3	input 2
5	data bits 20-23
6	data bits 24-27
7	data bits 28-31

Input level bits:

level_bits[1:0]	level or data bit
0	low level or offset bit 0
1	high level or offset bit 1
2	falling edge or offset bit 2
3	rising edge or offset bit 3

3. output control register (***ctrl_out***):

This register defines the output options used. See section I/O configuration. The register is divided into 5 sections of 6 bits each containing dest_bits[5:0] = {level_bits[1:0], source_bits[3:0]}.

Output destination bits:

ctrl_out bits	destination
5:0	output 0
11:6	output 1
17:12	output 2
23:18	bus enable 0
29:24	bus enable 1

Output source bits:

source_bits[3:0]	source
0	None
1	sync_out
2	sync_en
3	sync_mon
4	clock lost
5	error
6	run
7	wait
8	ready
9	restart
10	trg_start
11	trg_stop
12	trg_restart

Output level bits:

level_bits[1:0]	level
0	low level
1	high level

4. number of samples (***num_samples***):
Number of samples for experimental cycle. This needs to be a multiple of 4. The driver will automatically add samples with NOP bit set to fulfill this requirement.

5. number of cycles (***num_cycles***):
Number of cycles to be executed. If num_cycles = 0, repeates cycles for ever until ctrl_run bit is reset, default = 1. To enter cycling mode this needs to be set and ctrl_restart bit must be set.

6. clock divider register (***clk_div***):
- bits 0-7 define output bus sampling rate f_{bus} = 100MHz/clk_div[7:0], i.e. for f_{bus} = 1MHz: clk_div[7:0] = 100, for f_{bus} = 10MHz: clk_div[7:0] = 10. Max. tested f_{bus} = 10MHz.
- bits 8-15 define the stop trigger window. clk_div[15:8] = 0 or 1: no window, i.e. the board stops immediately. clk_div[15:8] > 1: window time t_{wnd} = 10ns*clock_div[15:8]. The board stops at the next integer multiple of t_{wnd} after the stop trigger.

7. strobe delay (***strb_delay***):
Contains setup and hold times for pseudo-clock 0 and 1 in units of 10ns. If end time = 0 the pseudo_clock toggles state at given setup time, otherwise hold time = end time - setup time.

strb_delay[7:0]	strobe_0 setup time
strb_delay[15:8]	strobe_0 end time
strb_delay[23:16]	strobe_1 setup time
strb_delay[31:24]	strobe_1 end time

8. synchronization delay (***sync_delay***):
Wait time in units of 10ns the primary board waits after it has generated the start trigger (sync_out) signal or the secondary board waits after receiving the start trigger.

strb_delay[9:0]	wait time

9. synchronization phase (***sync_phase***):
Phase shift of the PLL for the external and detection clock in steps of 560 = 360° = 10ns. The external clock phase allows to fine-adjust the waiting time of the secondary board for the synchronization with the primary board (see section synchronization of several boards). The detection phase does not directly impact the waiting time one but might need to be adjusted it in case that +/-10ns jitter is detected which happens when the trigger detection clock is very close to the trigger pulse such that noise severely affects the timing.

sync_phase[11:0]	detection clock phase
sync_phase[23:12]	external clock phase

10. status register (***status***):
Status bits which can be readout at any time, updated only with status_IRQ_freq at ca. 20Hz during experimental cycle.

bit	name	description
0	status_reset	set after ctrl_reset bit is set high until reset is performed
1	status_ready	set after first data received into FIFO buffer
2	status_run	set during experimental run. not reset during waiting state
3	status_end	set when experimental run has ended. in cycling mode set only after last cycle

4	status_wait	set during waiting state for start or restart trigger
5	-	used for auto-sync
6	-	used for auto-sync
7	status_ps_active	set while phase shift of PLL is active
8	-	not used
9	-	not used
10	status_clk_ext	set when external clock is used
11	status_clk_locked	set when external clock is available
12	status_err_in	error set when no data available and not in end state. indicates that RX FIFO has become empty due to fast bus data output rate
13	status_err_out	error set when data cannot be written to TX FIFO. indicates a bottle neck in the data transmission, CPU clogging, IRQ flooding, etc.
14	status_err_time	error set when the timestamp is not increasing in time
15	status_err_lock	error set when external clock is selected and lock is lost
16	-	used for debugging
17	-	used for debugging
18	-	used for debugging
19	-	not used
20	status_IRQ_error	interrupt set when ctrl_IRQ_en enabled and an error occurred
21	status_IRQ_end	interrupt set when ctrl_IRQ_en & ctrl_IRQ_end_en enabled and board is in end or in wait state
22	status_IRQ_restart	interrupt set when ctrl_IRQ_en & ctrl_IRQ_restart_en enabled and board restarts at end of cycle or resumes from stopped state
23	status_IRQ_freq	interrupt set when ctrl_IRQ_en & ctrl_IRQ_freq_en enabled with about 20Hz during experimental cycle
24	status_IRQ_data	interrupt set when ctrl_IRQ_en & ctrl_IRQ_data_en enabled and IRQ bit in data is enabled and set
25	-	not used
26	-	not used
27	-	not used
28	status_TRG_start	toggles when start trigger is active
29	status_TRG_stop	toggles when stop trigger is active
30	status_BTN_0	set when button 0 is pressed
31	status_BTN_1	set when button 1 is pressed

11. board time (***board_time***):
Actual number of ticks in units of 1/sample rate.

12. board samples (***board_samples***):
Actual number of executed samples.

13. board time extra (***board_time_ext***):
At the moment unused.

14. board samples extra (***board_samples_ext***):
At the moment unused.

15. board cycles (***board_cycles***):
Actual number of executed cycles.

16. synchronization time (***sync_time***):
Measured round-trip time from start trigger generation (sync_out) to reflected signal detection (sync_in) in units of 10ns for the auto-synchronization. This is at the moment not used and gives just the time of the timeout.

17. firmware version (***version***):
Firmware version information gives buffer board version major.minor and firmware build date:

version[31:24]	board version major is always 1 (8 bits)
version[23:16]	board version minor (2-4, 8 bits)
version[15:9]	year - 2000 (0-127, 6 bits)
version[8:5]	month (1-12, 4 bits)
version[4:0]	day (1-31, 5 bits)

18. firmware info (***info***):
Firmware information where at the moment only the FPGA board is indicated:

info[1:0]	FPGA board
0xc0	Cora-Z7-07S
0xc1	Cora-Z7-10
0xa1	Arty-Z7-10
0xa2	Arty-Z7-20

Appendix II. server commands and DLL functions [updated in the next version of the manual].