YCPSWASYN: EPICS Driver for FPGA Register Access and Asynchronous Messaging.

J. Vasquez, E. Williams, J. D'Ewart, K. Kim, T. Straumann
SLAC National Accelerator Laboratory, Menlo Park, California, USA

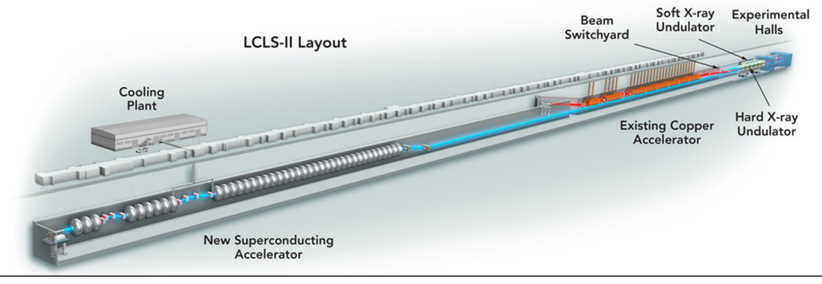
Abstract:

The Linac Coherent Light Source II (LCLS-II) is a major upgrade of the LCLS facility at SLAC, scheduled to start operations in 2020. The High Performance Systems (HPS) defines a set of LCLS-II controls sub-systems which are directly impacted by its 1Mhz operation. It is formed around a few key concepts: ATCA based packaging, digital and analog application boards, and 10G Ethernet based interconnections for controls. The Common Platform provides the common parts of the HPS in term of hardware, firmware, and software. The Common Platform Software (CPSW) provides a common interface to the common platform's FPGA for all high-level software. YAML is used to define the hardware topology as and all necessary parameters. YCPSWASYN is an AsynDriver based EPICS module for FPGA register access and asynchronous messaging. It uses CPSW for accessing the hardware which is described by YAML. YCPSWSYN has two operation modes: an automatic mode where PVs are automatically created for all registers and the record's fields are populated with information found in YAML; and a manual mode where the engineer can choose which register to expose via PVs and freely choose the record's filed information.

The Linac Coherent Light Source II (LCLS-II)

LCLS is the world's first hard X-ray free-electron laser. Its strobe-like pulses are just a few millionths of a billionth of a second long, and a billion times brighter than previous X-ray sources. Scientists use LCLS to take crisp pictures of atomic motions, watch chemical reactions unfold, probe the properties of materials and explore fundamental processes in living things.

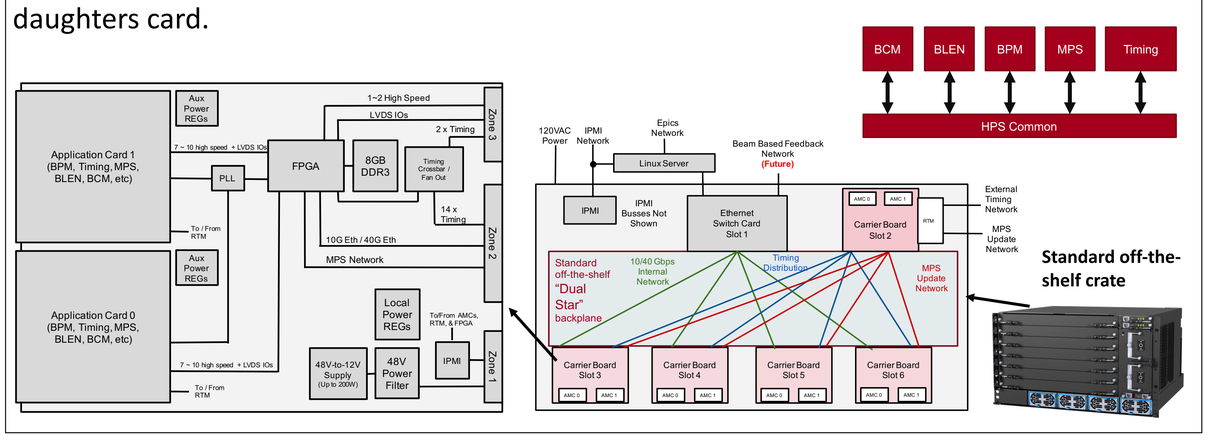
LCLS-II will provide a major jump in capability – moving from 120 pulses per second to 1 million pulses per second. This will enable researchers to perform experiments in a wide range of fields that are now impossible. The unique capabilities of LCLS-II will yield a host of discoveries to advance industry, new energy solutions and our quality of life.



LCLS-II website

SLAC High Performance System (HPS) and Common Platform

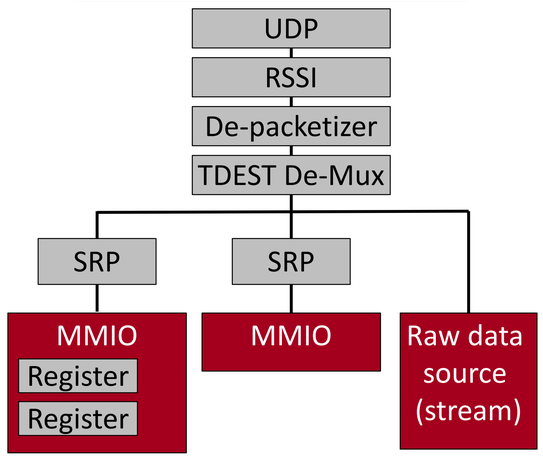
Due to the 1MHz operation rate of the LCLS-II, SLAC is implementing the High Performance System which is based on ATCA. Each application controls will be implemented in one ATCA blade. The carrier card provides the FPGA as well as all the digital and power sections which are common to all applications. The analog application-specific sections are in the AMC daughters card.



SLAC Common Platform Software (CPSW)

The CPSW is a software layer that provides a common interface to the HPS Common platform FPGA for all high level software.

Communication protocol stack

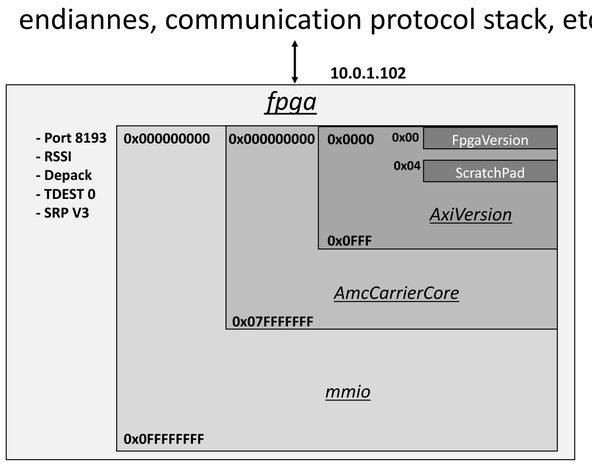


APIs

		EPICS analogy
Builder	*Assemble hierarchy, define parameters and topology.*	Database
User	*provide access to a device in the hierarchy*	CA client
Developer	*All details visible, for defining new objects*	Base

Describing the FPGA register using YAML

We chose YAML to describe the hardware topology and parameters (register offset, endiannes, communication protocol stack, etc.).



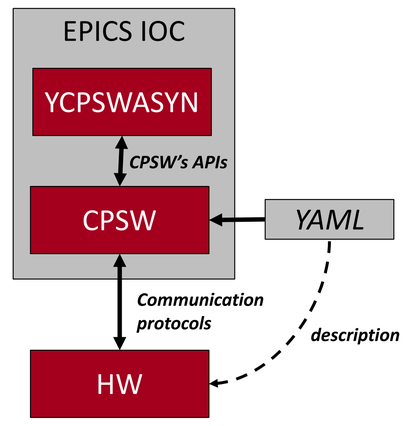

```
mmio: &mmio
    class: NetIODev
    size: 0x100000000
    children:
        AmcCarrierCore:
            class: MMIODev
            size: 0x080000000
            at:
                offset: 0x000000000
            description: Amc Carrier Core
            children:
                AxiVersion:
                    class: MMIODev
                    size: 0x000001000
                    at:
                        offset: 0x000000000
                    description: AXI-Lite Version Module
                    children:
                        FpgaVersion:
                            class: IntField
                            at:
                                offset: 0x0000
                            mode: RO
                            description: FPGA Firmware Version Number
                        ScratchPad:
                            class: IntField
                            at:
                                offset: 0x0004
                            mode: RW
                            description: Register to test reads and writes
```

Accessing the registers using the user API

- *Load Yaml description* `Path root = IPATH::LoadYamlFile(file.yaml);`
- *Navigate the hierarchy (lookup)* `Path p = root->findByName("fpga/mmio/AmcCarreierCore/AxiVersion/FpgaVersion");`
- *Instantiate user interface* `ScalVal fpgaVersion = IScalVal::create( p );`
- *Access properties* `fpgaVersion->getVal( &int32_variable );`

YAML + CPSW + ASYN: YCPSWASYN

- EPICS module based on asynPortDriver
- Uses CPSW for accessing the hardware, which is describe using YAML
- General purpose driver, doesn't have any application specific functions. It gives access to registers and asynchronous messages.
- It has 2 operations mode: auto and manual generation of PVs



Auto-generation of PVs

- In this mode, the module navigates the entire hierarchy defined in YAML and generates PVs for all elements found.
- Based on the type of register and its properties, records of appropriated type are generated and their fields are populated with the register properties defined in YAML (e.g., SCAN, DESC; Bx and MBBx state names, etc.).
- The PV name is generated from the register "path".

```
/fpga/mmio/AmcCarrierCore/AmcBay0Core/Adc16Dx370[0]/AdcReg_0x0003
```

```
AdcReg_0x0003:
  at:
    offset:    0x00C
    nelms:     1
  class:       IntField
  sizeBits:    32
  mode:        RW
  pollSecs:    5
  description: ADC control register
```

→

```
Record(longout, "$(P):C:B0:Adc0:AdcReg_0x0003:St")
{
  field(DTYP, "asynInt32")
  field(SCAN, "Passive")
  field(DESC, "ADC Control register")
  ...
}
Record(longin, "$(P):C:B0:Adc0:AdcReg_0x0003:Rd")
{
  field(DTYP, "asynInt32")
  field(SCAN, "5 seconds")
  field(DESC, "ADC control register")
  ...
}
```

Register information				EPICS PV
Register class	**N. Elements**	**Encoding**	**Enum?**	**Record type**
IntField	1	None	No	longin, longout
			Yes	bi, bo, mbbi, mbbo
		IEEE_754	No	ai, ao
	>1	None		waveform
SequenceCommand	N/A			bo
IntField (stream port)				waveform

Manual generation of PVs

- The module gives the possibility to disable the auto-generation and define PV manually for specific registers. This gives the possibility to freely choose PV name and field properties
- The manual generation of PVs requires 2 steps:

1. Map a register to an asyn parameter (dictionary file)

```
#Register path                                          asyn paramete
/fpga/mmio/AmcCarrierCore/AxiVersion/FpgaVerson         FPGA_VERSION
```

2. Create an appropriate record

```
Record(longin, "$(P):FpgaVersion")
{
  field(DTYP, "asynInt32")
  field(SCAN, "Passive")
  field(DESC, "Fpga Version number")
  field(INP,  "@asyn($(PORT))FPGA_VERSION"
}
```

ICALEPCS2017
Barcelona · Spain, October 8-13 · Palau de Congressos de Catalunya

Jesus A. Vasquez Stanescu
jvasquez@slac.stanford.edu