A High Level FPGA Framework for Application Development

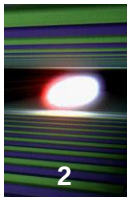
B. Fernandes

European XFEL
HPSP 2014 - Valletta, Malta
Institute of Space Sciences and Astronomy (ISSA)

29 October 2014

European XFEL

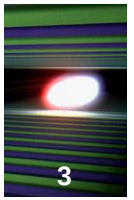
Summary



- FPGA VHDL Development approach
- High level FPGA Application development
- Development Environment
- Example application
- Future development
- Conclusion

European XFEL

FPGA VHDL Development environment



Board Modules

Everything related and specific to configuration/monitor of Board features and provide Input/Output FPGA signals to the user applications module;

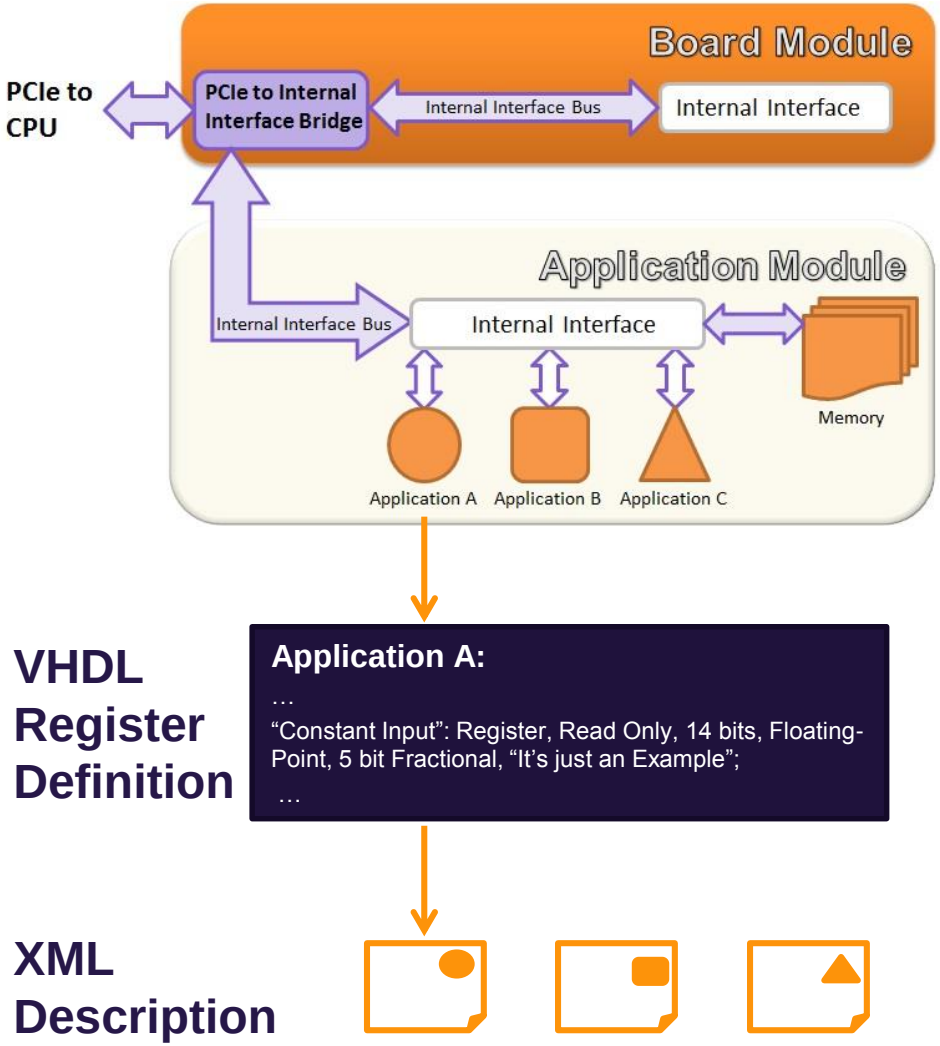
Application Modules

Includes user algorithms modules only (may or may not be Board specific)

- Fixed structure divided in Board and Application modules
- Allows for easy porting and update of both board and application code.
- It provides an environment for firmware programming for standardized and future hardware.
- Independent development.

European XFEL

FPGA VHDL Development environment



- Internal Interface bus protocol which includes a VHDL API package that eases register access and definition;
- Application have a specific set of registers and memories with definition being preformed at the hardware level;
- A XML file for each application is generated which is then used by our Software tools to access the defined registers/areas.

European XFEL

High Level FPGA Application Development

The FPGA should bend time and space according to

$$R_{\mu\nu} - \frac{1}{2} R\, g_{\mu\nu} + \Lambda\, g_{\mu\nu} = \frac{8\pi G}{c^4} T_{\mu\nu}$$

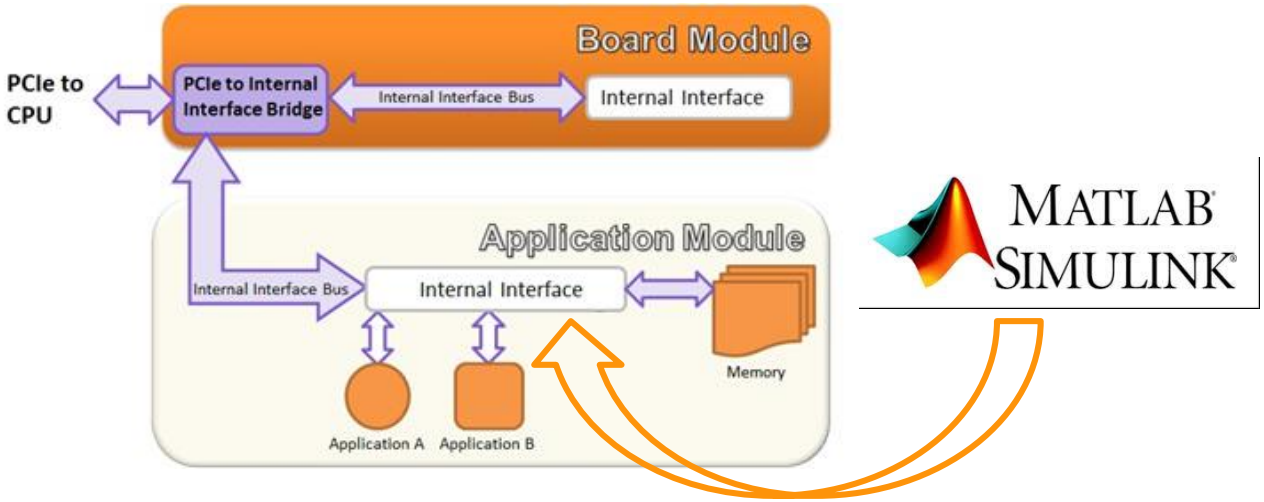
- FPGA programming is time intensive and requires specialists, however **most applications and algorithms are conceptualized by users unfamiliar with FPGA programming**.
- For the European XFEL, a high level FPGA framework is being developed that allows for users with no prior HDL knowledge to develop their algorithm modules which can be integrated in a top VHDL project.

European XFEL

Development Environment

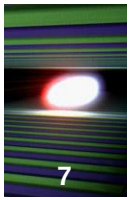
Application development in Simulink

- Graphical environment allows for people unfamiliar with HDL to quickly start developing hardware compatible algorithms
- Structured algorithms modules, reusability of existing blocks and reduced time for debugging
- Integration of Matlab allows for powerful test environments



European XFEL

High Level FPGA Application Development



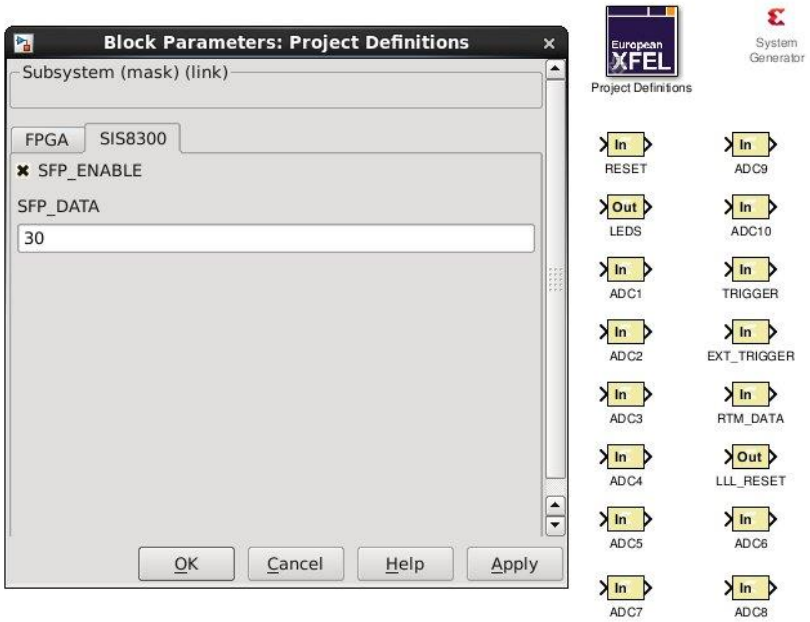
Simulink Developer

- User design environment automatically setup for the target board;
- Abstract end user from hardware programming languages and concepts such as Pin placement, clock routing, etc.;
- No restriction in Register generation and definition;
- Library with blocks that simulate the behavior of available features;
- Easy to port and distribute applications to other projects/boards;

VHDL Developer

- Easy to define/manage the available I/O in the Simulink;
- Integrate Generics in the Simulink;
- User defined registers and memories are managed in Simulink - generation of necessary logic to communicate with the bus protocol;
- Standard import of VHDL blocks to Simulink;

Development Environment



```
component SIS8300_SIMULINK is
  port (
    adc1                : in  std_logic_vector(15 downto 0);
    adc10               : in  std_logic_vector(15 downto 0);
    adc2                : in  std_logic_vector(15 downto 0);
    adc3                : in  std_logic_vector(15 downto 0);
    adc4                : in  std_logic_vector(15 downto 0);
    adc5                : in  std_logic_vector(15 downto 0);
    adc6                : in  std_logic_vector(15 downto 0);
    adc7                : in  std_logic_vector(15 downto 0);
    adc8                : in  std_logic_vector(15 downto 0);
    adc9                : in  std_logic_vector(15 downto 0);
    bus_model_mcw_clk_1 : in  std_logic;
    ext_trigger         : in  std_logic_vector(3 downto 0);
    ii_addr             : in  std_logic_vector(31 downto 0);
    ii_datain           : in  std_logic_vector(31 downto 0);
    ii_wr               : in  std_logic;
    ------------------------------------------------------------------------
    -- DDR
    ------------------------------------------------------------------------
    ddr_busy1           : in  std_logic;
    ddr_addr1           : out std_logic_vector(30 downto 0);
    ddr_data1           : out std_logic_vector(255 downto 0);
    ddr_reset1          : out std_logic;
    ddr_str1            : out std_logic;

    ddr_busy2           : in  std_logic;
    ddr_addr2           : out std_logic_vector(30 downto 0);
    ddr_data2           : out std_logic_vector(255 downto 0);
    ddr_reset2          : out std_logic;
    ddr_str2            : out std_logic;
```

- The Project Definitions block: the user defines for which board the algorithm is going to be develop as well as the bus protocol:
  - Generate the IO available for the chosen board and expected input signals;
  - Include defined VHDL Generics (size and availability IO);
  - Includes the System generator block and defines its parameters according to the FPGA board.

European XFEL

Development Environment

BUS block allows for users to define registers and memories which are later accessible by the chosen protocol:

- Write/Read cycles, data latching and address indexing are not taken into account during development phase

```
|  Type  |  Name  |  N. Bits  |  N. Elements  |  Wr. Permisson  |  Rd. Permission  |  Data Type  |  Frac. Bits  |  Default Value  |  Display Name  |  Description  |
```

Subsystem (mask) (link)

```
(VII_WORD  , VIIRegName("WORD_CLK_MUX")       ,  2  ,   4  , VII_WINTERNAL , VII_RINTERNAL , VII_UNSIGNED , 0 , X"00000000"
                                              , VIIDisplayName("Clock Multiplexer") , VIIDescription("Configuration of the clock multiplexers"))  ,
(VII_WORD  , VIIRegName("WORD_CLK_RST")       ,  1  ,   1  , VII_WINTERNAL , VII_RINTERNAL ,     VII_BOOL , 0 , X"00000000"
                                              , VIIDisplayName("Clock Reset")       , VIIDescription("Reset divider chips circuits"))           ,
(VII_WORD  , VIIRegName("WORD_CLK_LOCK")      ,  2  ,   1  , VII_WNOACCESS , VII_REXTERNAL ,     VII_BOOL , 0 , X"00000000"
                                              , VIIDisplayName("Clock Lock Pins")   , VIIDescription("Clock PLL Lock pins"))                    ,
(VII_WORD  , VIIRegName("WORD_ADC_RESET")     ,  1  ,   1  , VII_WINTERNAL , VII_RINTERNAL ,     VII_BOOL , 0 , X"00000000"
                                              , VIIDisplayName("ADC reset")         , VIIDescription("Reset the ADCs"))                         ,
(VII_WORD  , VIIRegName("WORD_ADC_FIFO_RESET"),  10 ,   1  , VII_WINTERNAL , VII_RINTERNAL ,     VII_BOOL , 0 , X"00000000"
                                              , VIIDisplayName("ADC FIFO reset")    , VIIDescription("Reset ADCs FIFOs"))                       ,
(VII_WORD  , VIIRegName("WORD_ADC_FIFO_DELAY"),  10 ,   1  , VII_WINTERNAL , VII_RINTERNAL ,     VII_BOOL , 0 , X"00000000"
                                              , VIIDisplayName("ADC FIFO delay")    , VIIDescription("Add delay to ADC FIFO input data"))       ,
```

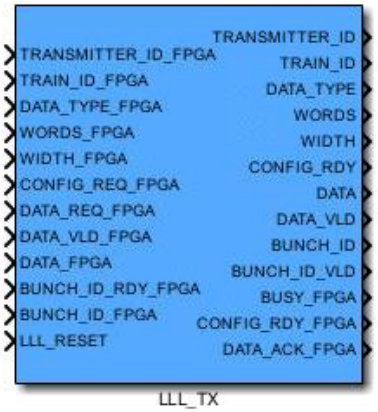
Number of Bits

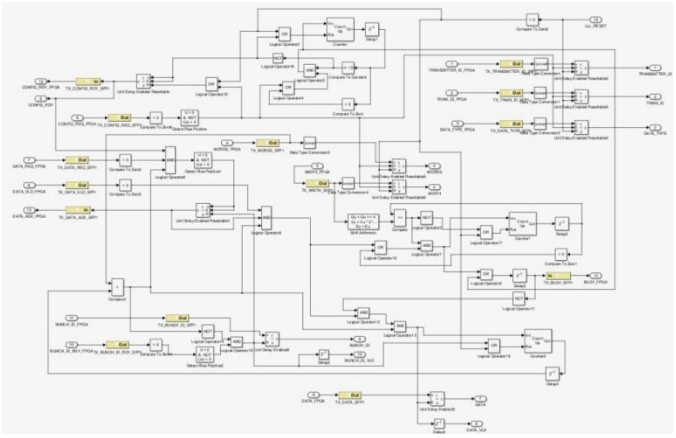
```
P_O_MUX1A_SEL <= IIGetItem(CON_II_AS, SIG_VECINT, "WORD_CLK_MUX", "SIS8300_STARTUP_BOARD", 0);
P_O_MUX1B_SEL <= IIGetItem(CON_II_AS, SIG_VECINT, "WORD_CLK_MUX", "SIS8300_STARTUP_BOARD", 1);
P_O_MUX2A_SEL <= IIGetItem(CON_II_AS, SIG_VECINT, "WORD_CLK_MUX", "SIS8300_STARTUP_BOARD", 2);
P_O_MUX2B_SEL <= IIGetItem(CON_II_AS, SIG_VECINT, "WORD_CLK_MUX", "SIS8300_STARTUP_BOARD", 3);

SIG_LOC_ADDR <= SIG_II_ADDR (9 downto 2);
SIG_LOC_DATA <= SIG_II_DATA_IN(7 downto 0);
SIG_LOC_STR  <= IIGetItemStr(CON_II_AS, SIG_VECWENA, "AREA_SPI_DIV", "SIS8300_STARTUP_BOARD", 0, SIG_II_STROBE);

IIPutItem(CON_II_AS, SIG_VECEXT, "WORD_CLK_LOCK", "SIS8300_STARTUP_BOARD", 0, P_I_CLK_LOCK, SIG_LOW);

SIG_CLK_RST <= SIG_RESET_II or IIGetItem(CON_II_AS, SIG_VECINT, "WORD_CLK_RST", "SIS8300_STARTUP_BOARD", 0)(0);
```

European XFEL

Development Environment


```
% Ports
LLL.addPort('RX_DATA','In',30,1); %, 'LLL_DATA');
LLL.addPort('RX_DATA_NEW','In',1,1);
LLL.addPort('RX_DATA_ERR','In',1,1);
LLL.addPort('DATA_ERR_CNT','In',8,1);
LLL.addPort('RX_CONFIG_RDY','In',1,1);
LLL.addPort('RX_WIDTH','In',8,1);
LLL.addPort('RX_WORDS','In',8,1);
LLL.addPort('RX_DATA_TYPE','In',8,1);
LLL.addPort('RX_TRAIN_ID','In',64,1);
LLL.addPort('RX_TRANSMITTER_ID','In',128,1);
LLL.addPort('RX_BUNCH_ID_NEW','In',1,1);
LLL.addPort('RX_BUNCH_ID','In',16,1);
LLL.addPort('RX_CONFIG_ERR','In',1,1);
LLL.addPort('CONFIG_ERR_CNT','In',8,1);
LLL.addPort('TX_BUSY','In',1,1);
LLL.addPort('TX_DATA','Out',30,1); %,'LLL_DATA');
LLL.addPort('TX_DATA_REQ','Out',1,1);
LLL.addPort('TX_DATA_VLD','Out',1,1);
LLL.addPort('TX_DATA_ACK','In',1,1);
LLL.addPort('TX_BUNCH_ID_RDY','Out',1,1);
LLL.addPort('TX_BUNCH_ID','Out',16,1);
LLL.addPort('TX_WIDTH','Out',8,1);
LLL.addPort('TX_WORDS','Out',8,1);
LLL.addPort('TX_DATA_TYPE','Out',8,1);
LLL.addPort('TX_TRAIN_ID','Out',64,1);
LLL.addPort('TX_TRANSMITTER_ID','Out',128,1);
LLL.addPort('TX_CONFIG_REQ','Out',1,1);
LLL.addPort('TX_CONFIG_RDY','In',1,1);
```



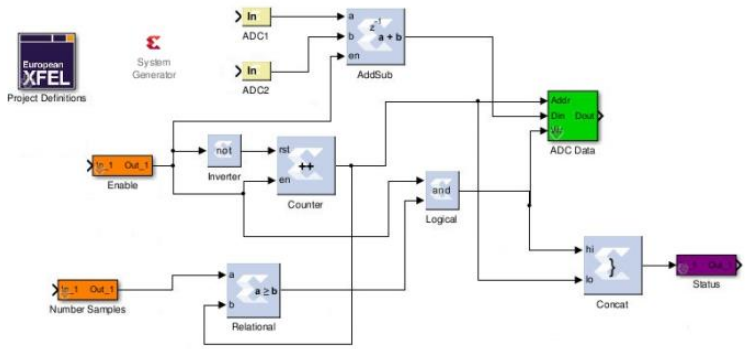


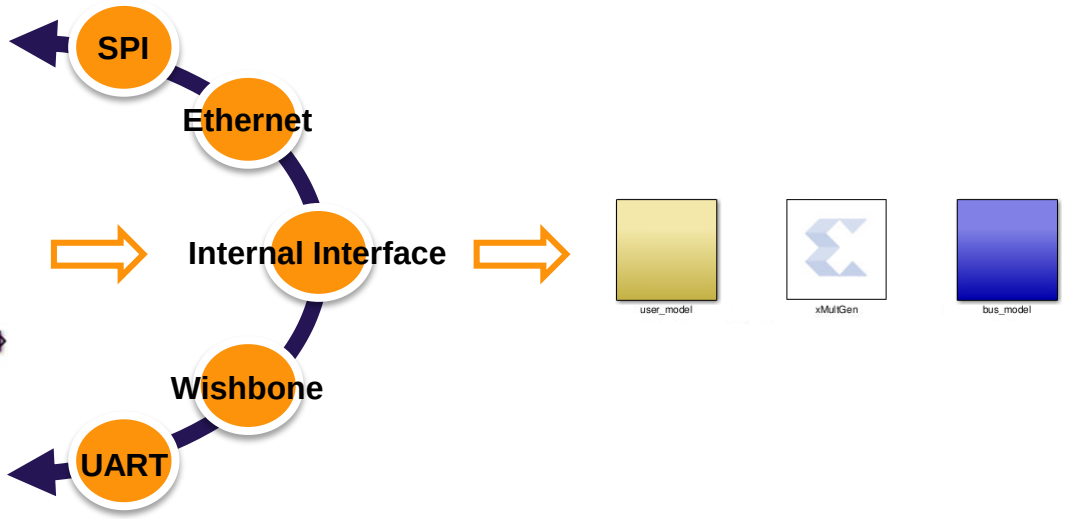
- Specific blocks are available on the library which will accurately simulate the behavior of features available on the board;
- In Hardware, the block is replaced with the real implementation.

Development Environment

The BUS logic is generated automatically:

- Based on defined user registers and chosen protocol;
- Compatible with different facilities;
- XML file with register information similar to the VHDL developed applications;
- Script to compile Simulink project with top ISE project;

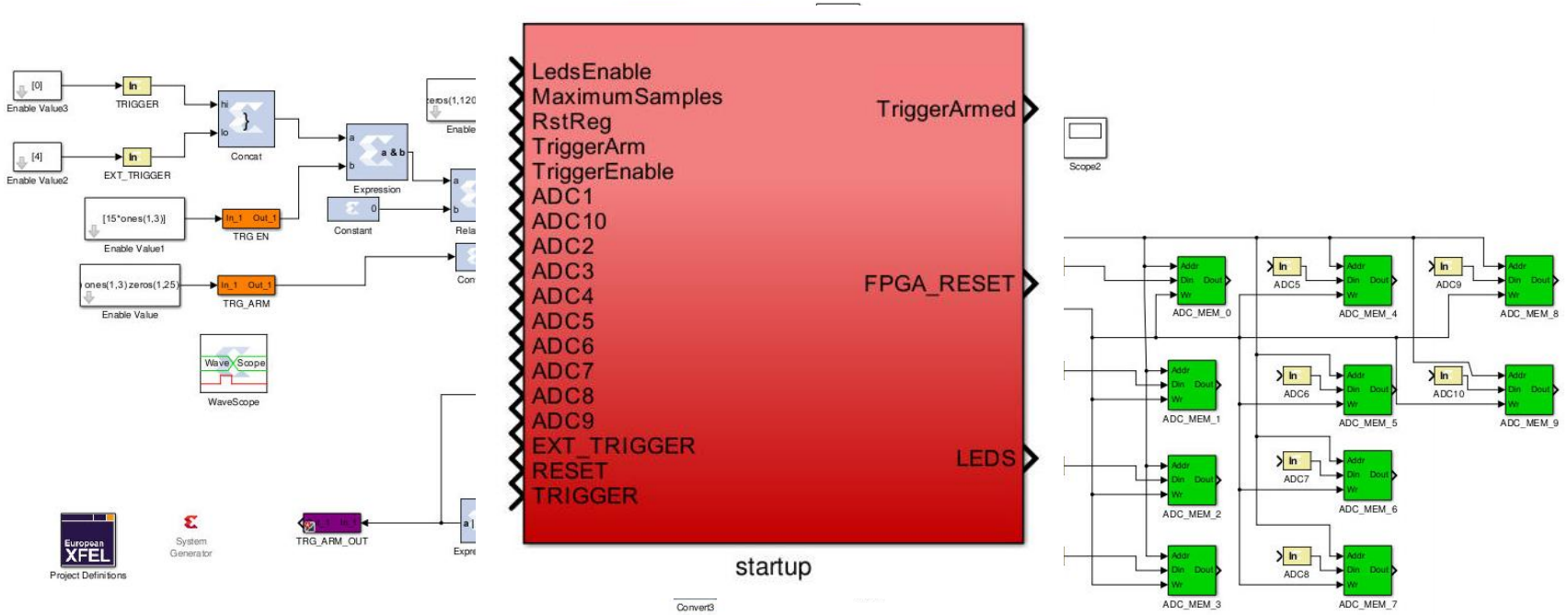




European XFEL

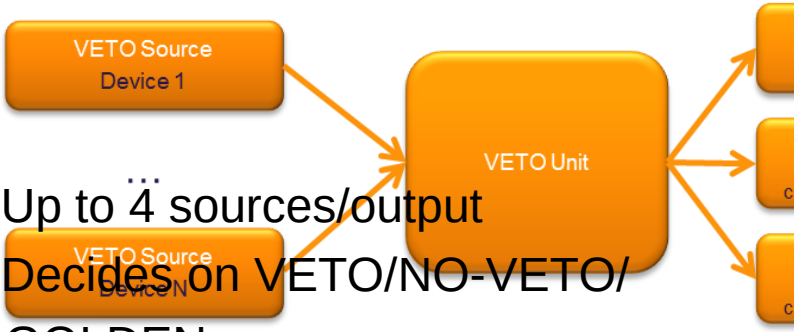
Distribute Applications

- Framework gives the option to save the Application as an standalone module while keeping all the defined registers and memories

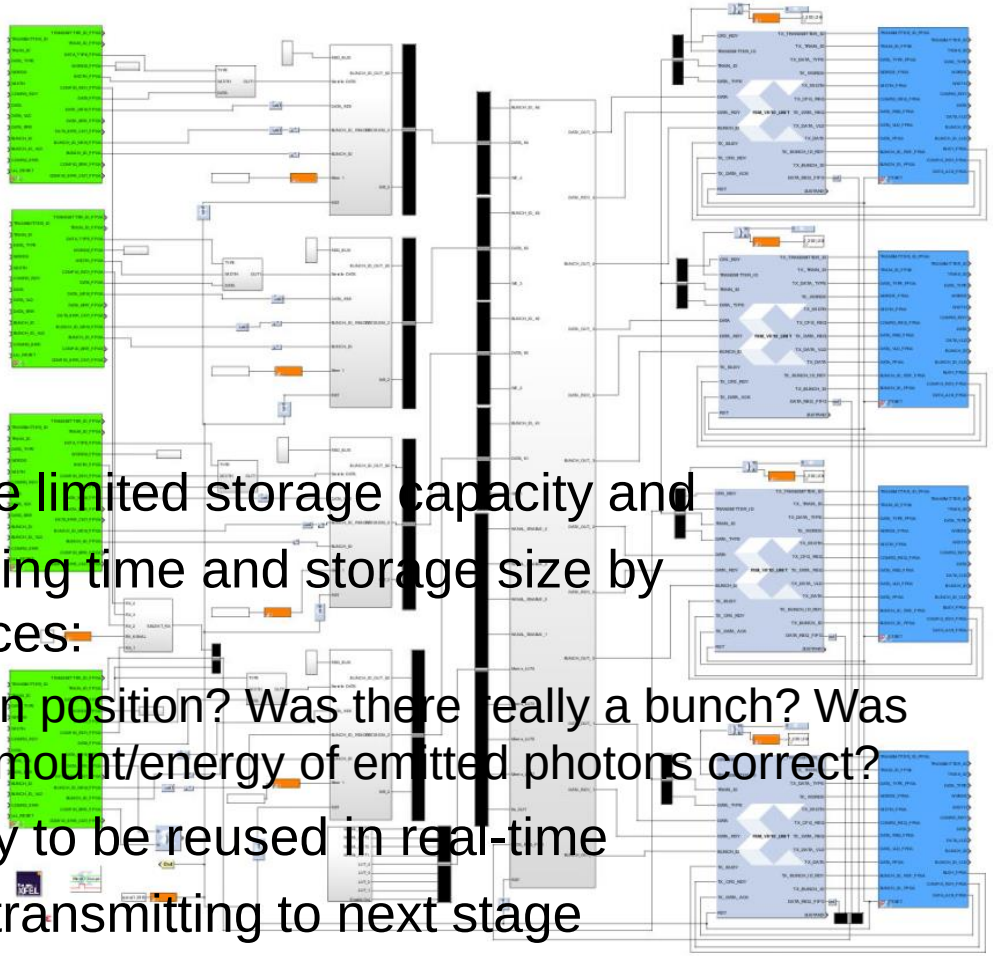


Example application

VETO Unit

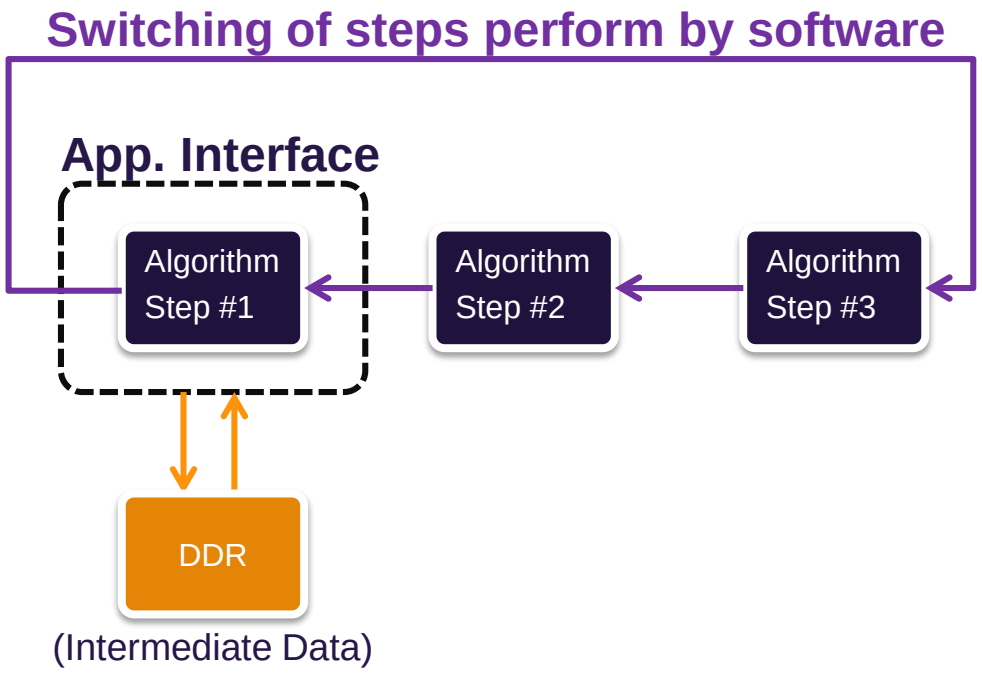


- Up to 4 sources/output
- Decides on VETO/NO-VETO/ GOLDEN
- Decision matrix fully configurable via software
- Certain sources could be ignored per output to reduce latency
  - Make most efficient use of the limited storage capacity and
  - Reduces bandwidth, processing time and storage size by considering information sources:
    - Is a pulse foreseen at a certain position? Was there really a bunch? Was the timing correct? Was the amount/energy of emitted photons correct?
  - Flag storage cells accordingly to be reused in real-time
  - Remove certain data before transmitting to next stage



Future development

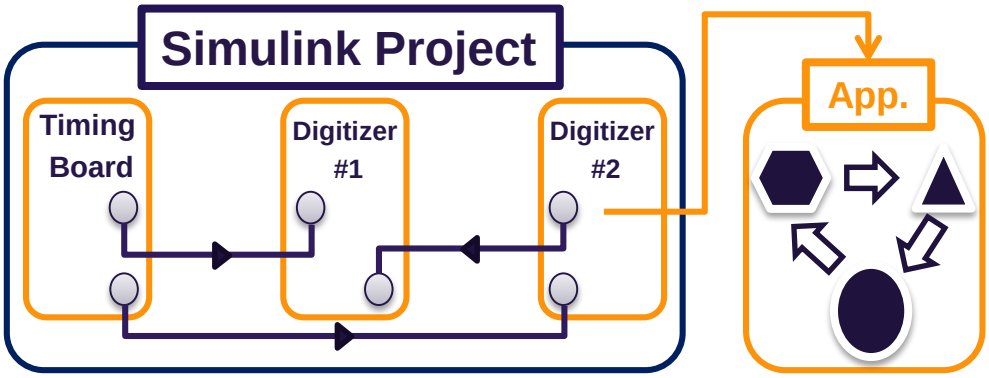
- **Partial reconfiguration**
  - Dynamically modify blocks of logic with partial bit files;
  - Reduce synthesis time and flexibility of FPGAs;
  - Partition large algorithms into smaller parts



European XFEL

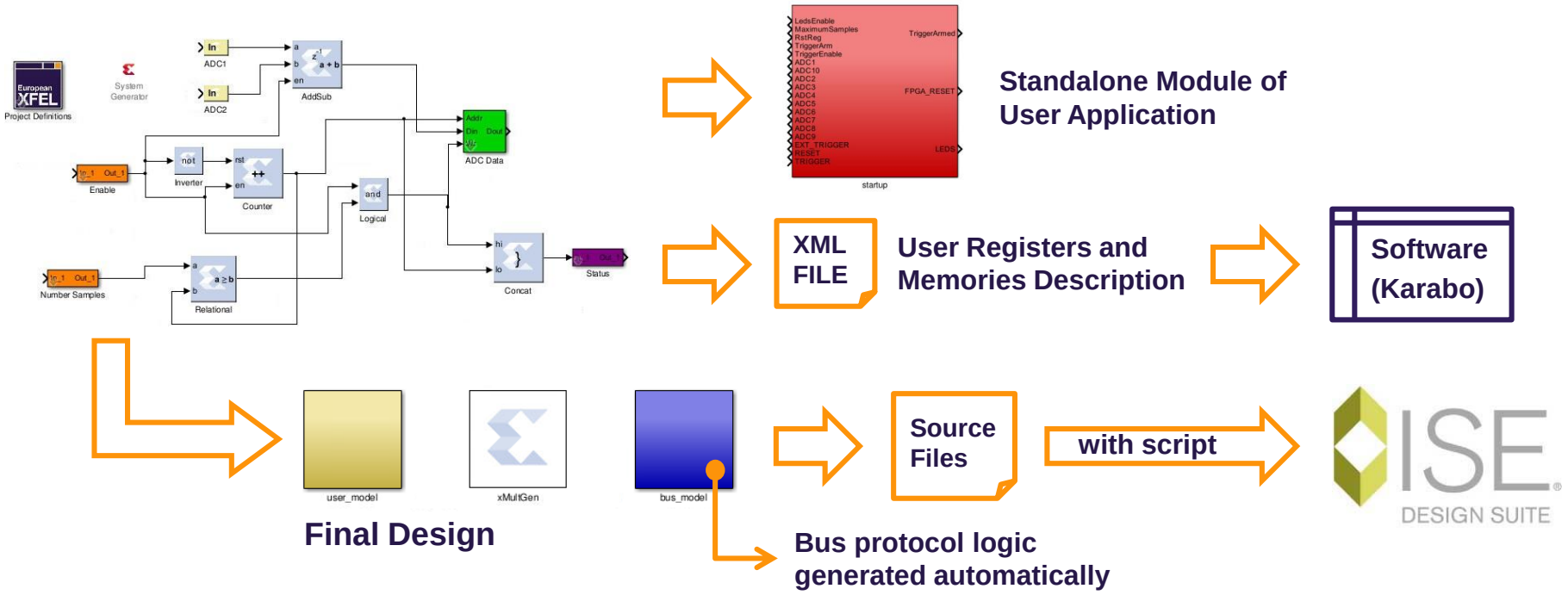
Future development

- **Partial reconfiguration**
  - Dynamically modify blocks of logic with partial bit files;
  - Reduce synthesis time and flexibility of FPGAs;
  - Partition large algorithms into smaller parts
- **Integrate additional protocols, namely Ethernet and Wishbone and IPBus (CERN)**
  - Integration with CASPER framework already tested;
- **Complete hardware systems in Simulink Design**
  - Inclusion of different boards and available interfaces to communicate between them;



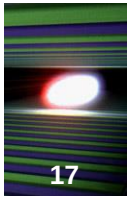
Conclusion

- The High-level FPGA programming framework allows users unfamiliar with FPGA programming to easily develop, simulate and integrate their algorithms into large FPGA projects;
- Current European XFEL FPGA projects are already being develop in this framework with positive results;



European XFEL

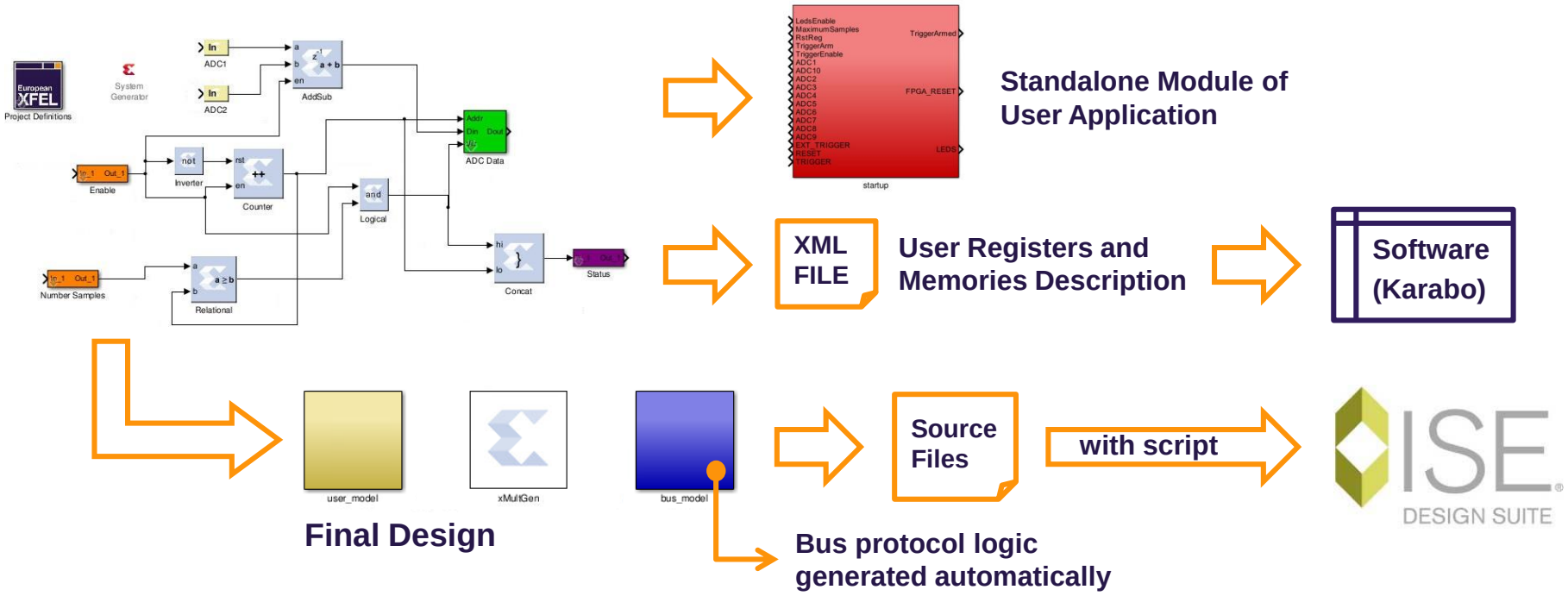
Questions?

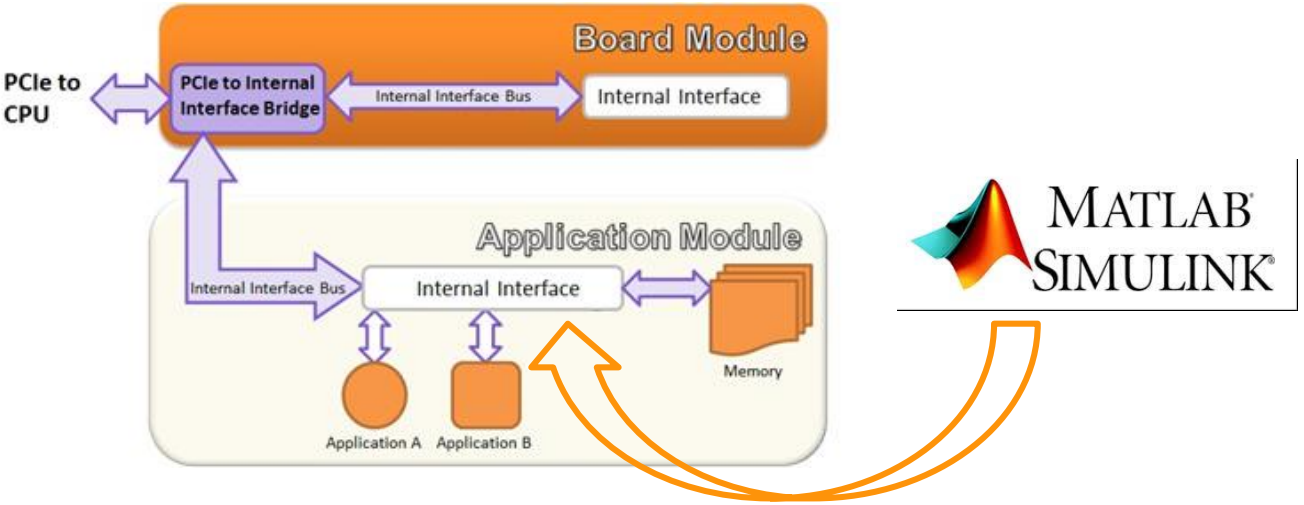


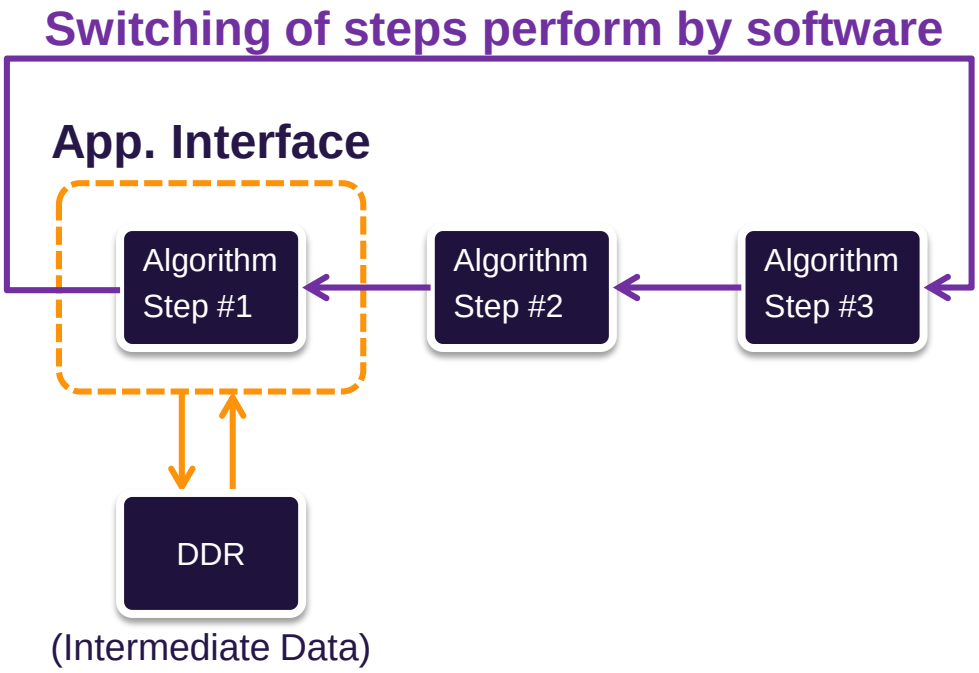
Check out the Demo!

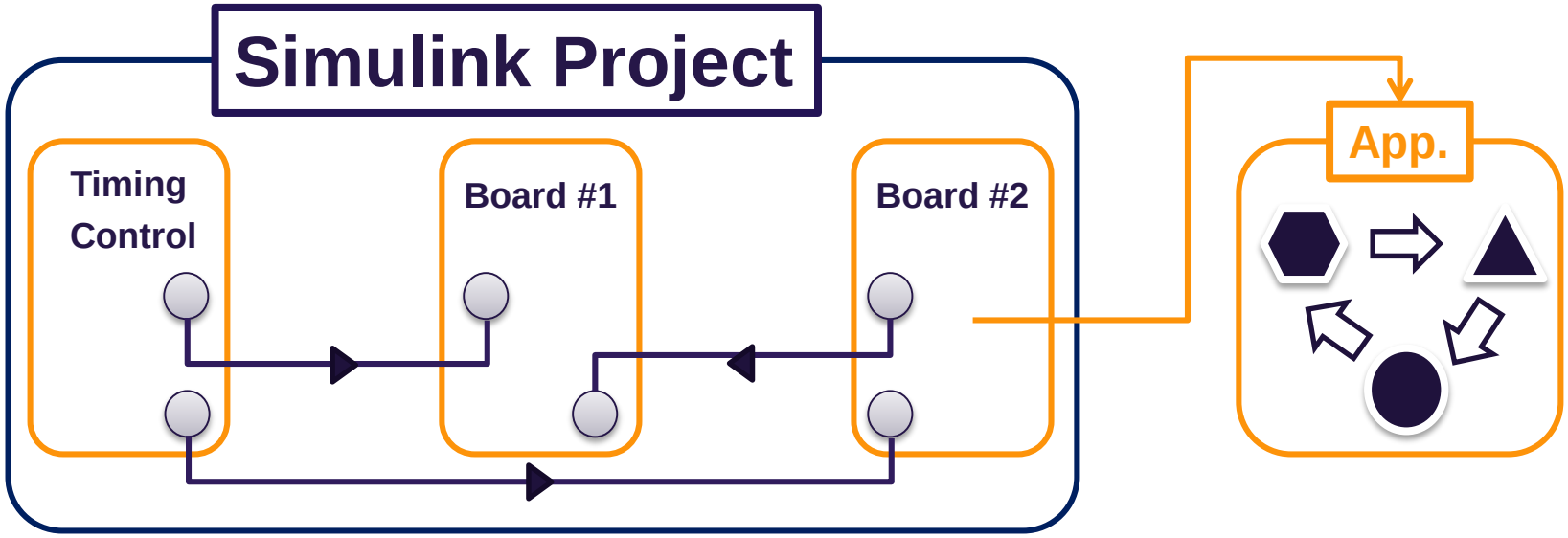
Framework Workflow

- The final Algorithm is processed to integrate the top level FPGA Project:
  - Generate Final Design with bus interface logic based on defined user registers and chosen protocol;
  - Simulink source files inserted in ISE top project;
  - Standalone Module of User application to share and distribute
  - XML file with register information similar to the VHDL developed applications.

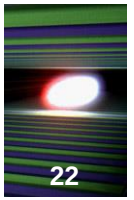



Board Module
PCIe to CPU
PCIe to Internal Interface Bridge
Internal Interface Bus
Internal Interface
Application Module
Internal Interface Bus
Internal Interface
Memory
Application A
Application B
MATLAB
SIMULINK


Switching of steps perform by software
App. Interface
Algorithm Step #1
Algorithm Step #2
Algorithm Step #3
DDR
(Intermediate Data)


Simulink Project
Timing Control
Board #1
Board #2
App.

Clip board – copy and paste



Headline

- first level
  - second level
    - ➔ third level

Headline

Texttext texttext
texttext texttext
texttext texttext

Keyword

1. Keyword
2. Keyword

- keyword
- keyword

Result Headline

- result text
- result text

Result headline

Result text, result text, result text

Result headline

- result text
- result text
- result text