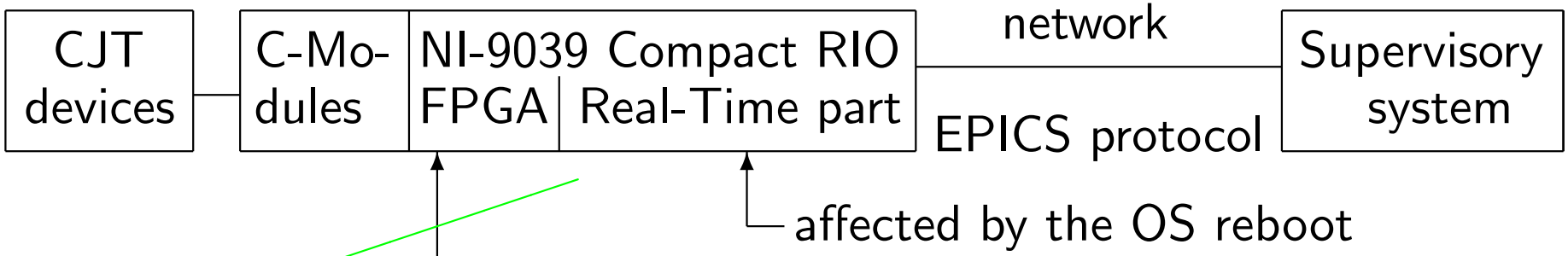
Highlights of the Cluster-Jet Target Control Program

Jerzy Tarasiuk

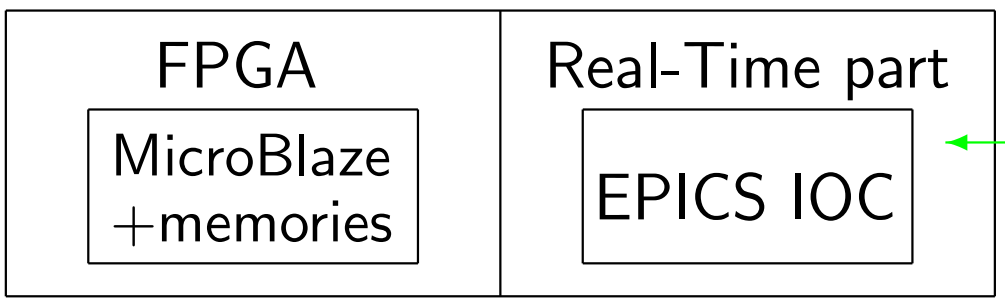
National Centre for Nuclear Research, Warsaw
and Faculty of Physics, University of Warsaw

PANDA CM 21/3 Target Session 2021-10-27

Introduction: the goal, where to put the control logic?



- FPGA should be programmed in LabVIEW - NI does not provide tools for accessing FPGA I/O ports except in NI LabVIEW;
- LabVIEW for FPGA provides means to include VHDL programs; and Xilinx Vivado provides MicroBlaze™ processor with a possibility to export it in VHDL - we employed the processor in the FPGA:



MicroBlaze and IOC in the Compact RIO

Note these gaps between inner rectangles and outer outlines - both MicroBlaze and IOC need additional software pieces for any connection.

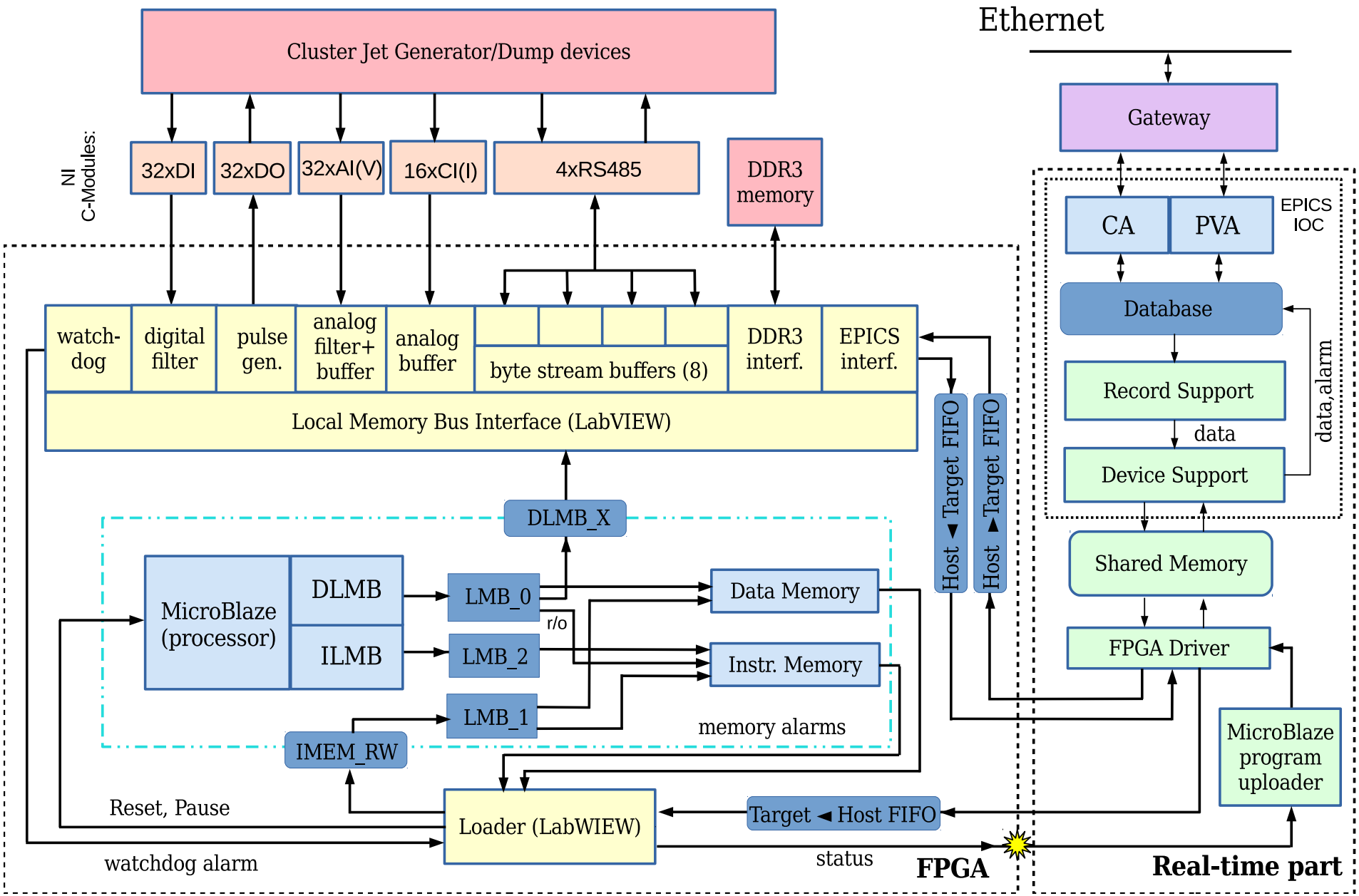
Advantages of using a processor supported by GCC

- GNU C Compiler (GCC) is widely used on many platforms.
- C has wider descriptive power than LabVIEW in FPGA applications (LabVIEW is like RTL, or assembly programming).
- Program coherency automation: using e.g. #include in C.
- Verification: more techniques developed for C; a textual form (C) is better suited for systematic analysis than a pictorial (LabVIEW).
- More possibilities for testing/debugging:
  - a C program can be initially tested on a PC;
  - FPGA can emulate devices for the program, and another C program can test processor+connections - a way to test thoroughly both program parts.
- More flexibility to perform corrections:
  - no special tools required: usual text editor, free GCC...
  - short compilation time (seconds as compared with minutes);
  - knowledge of C is more widespread than of LabVIEW.

The target system and the requirements for its control are complex
$\Rightarrow$ the above advantages are essential for us.

On the next slide, an overall structure of the system is presented.

Overall structure of the control system



Signal paths inside cRIO

- MicroBlaze+memory subsystem has connections via LMB (Local Memory Bus) - one master (it gets all processor data accesses) and one slave port; plus control and status signals (Pause, Reset, alarms...). Except the master LMB these signals are used by the program loader.
- The loader gets the MicroBlaze program from the RT part, puts it into instruction memory and starts the processor.
- Data transfer: a LabVIEW part connected to the master LMB accesses C-Modules and buffers the data from/to them; it also has connections to the RT part for passing EPICS data - the processor stores the data item to a variable in a dedicated part of its own memory, and the data reaches EPICS.
- In the RT part, the IOC has a support module for MicroBlaze; a separate task communicates with the FPGA, they use circular buffers in a shared memory for passing data.

C-Modules interfacing to MicroBlaze

- Memory-mapped I/O interfaces (via memory load/store instructions).
- Clock domain barrier: processor signal has to be synchronized with its clock; C-Modules don't allow for clock, and cannot provide data at the speed of the processor access (reading 32-channels of DI-module 9425 lasts 7μs, the processor cycle is 25ns); buffers are necessary.
- A status word indicating which interfaces request an attention.
- DO (Digital Out) signals are to be pulses; the processor writes polarity and the duration of the required pulse.
- DI (Digital In) signals are filtered; the processor is informed after a change occured and then these signals became stable.
- AI (Analog In 0-10V) signals are to be filtered - need an algorithm that will efficiently remove noise; the processor has to read a value cleared from noise and averaged over a time, say, 20ms.
- CI (Current In 4-20mA) module is slow - its data is buffered only.
- RS485 data is buffered to avoid data loss or pauses of sending.

MicroBlaze - EPICS communication: input

- The MicroBlaze memory has an area for the EPICS communication.
- Any write to the area causes the data and address to be sent in a single 64-bit word to the RT part via the Target-to-Host FIFO.
- All variables in the area should be global (to have global names).
- The MicroBlaze interface module in EPICS IOC receives these 64-bit words. During initialization, it gets names and addresses of variables in the "EPICS area", and maps them to EPICS fields (a configuration file defines mapping of MicroBlaze names to fields in EPICS).
- For ordinary fields, dbPutField() function is then used; for alarm setting or clearing, recGblSetSevr() or recGblResetAlarms().
- Alarms use a separate part of the EPICS area, have their own variable names, and definitions of EPICS connection in the configuration file.
- The simple picture gets complicated when the value to be passed has a size greater than the 32-bit MicroBlaze word: the data must be stored in parts, and the interface module must know when the data is complete - the MicroBlaze must store the "end" of the value.

MicroBlaze - EPICS communication: output

- EPICS records that are outputs to the MicroBlaze use a device support that sends their values with identifiers to the MicroBlaze via a Host-to-Target FIFO.
- The device support code uses:
  a `DTYP="microblaze"` in the EPICS record concerned,
  a `device(`*rectype*`,INST_IO,`*dsname*`,"microblaze")` in DBD file,
  an `epicsExportAddress(dset,`*dsname*`);` and
  a `static` *rectype*`dset` *dsname* `= {...};` in C source.
- The MicroBlaze gets them the same way as values from C-Modules, except that it reads the identifier first, and after range verification of the identifier it gets the value.
- The MicroBlaze can also request for reading of a field from the EPICS, with same definition as for the input - the MicroBlaze sends an address of its variable (necessarily global, and having a mapping to the EPICS field to be obtained), then gets the value in the same form; the EPICS module uses dbGetField() to obtain the value.

Indicators of the control system complexity

- 87 devices to be controlled or monitored of 25 different types.
- Total of 60 ministates in their startup/shutdown sequences (22+2 for CJG start to Standby + to Running state; 1+12 for CJG stop to Standby + to Off state; 15+1 and 2+5 for CJBD start and stop, respectively). Plus 6 for valve testing, and few for VP20 control.
- 4 non-serial connection types: digital input 0/24V, digital output 0/24V, analog input (voltage 0-10V), and analog input (current 4-20mA).
- Serial connections use 3 signal types (RS232, RS485, Ethernet), and 9 different protocols (2 of these for signal type converters), used to control or monitor 38 devices.

MicroBlaze - program structure

- The heart: a scheduler/scanner, and flags that trigger actions.
- Simple actions for bits in the status word mentioned before - e.g. just get a data item and store it; the highest priority.
- Limited medium-priority actions - they must end/pause quickly. One of them awakes sleeping actions at times they requested.
- Low-priority actions: long sequences (some never terminate).
- Except for the first kind, actions are usually multi-stage: procedures performing them are state machines, with the context pointer passed as an argument and returned as a value; NULL as an argument means initialization; NULL returned - an end of the action.
- All inter-stage data kept in the context, except for a pointer to free context list (when terminating an action, the context is saved there and later reused for a new initialization).
- A possibility to request timed actions (like Unix 'at' command).
- Three layers: basic I/O, protocol, application (on the next slide).
- Circular buffers are used for data transfer within the first two layers; the third one uses the current values and their timestamps.

MicroBlaze program structure - processing layers

Basic I/O (drivers): procedure invocation is triggered by a specific event external to the processor subsystem. These events are: a DI signal change (verified by the filter), an analog input data available, a serial port input data available or output data request, a data arriving from EPICS, and a timer tick (its driver counts the elapsed time).

Protocol: used mainly for serial communication; a communication via a converter involves one of them for the converter protocol and another one for the device itself. Also, this layer provides action synchronization (monitor functionality) and synthesis withing the protocol units. Actions taken on requests coming from the other two layers.

Application (synthesis): Actions in this layer provide synthesis of data over devices and over time. They include initiating serial port communication (e.g. by a command "get pressures from the Center Three 2 on serial port 3"), monitoring device status information and data for errors or abnormal values, getting and handling shifter requests, handling preparing and shutdown sequences. This layer uses the data obtained by the other two layers and performs respective actions.

Thank you for your attention