Reconfigurable measurement systems based on real time and FPGA
Ján Šaliga
Technical University of Košice, Slovakia
(based on different sources mainly from National Instruments)

Typical nowadays system:
- Low cost instruments (system): microcontroller with fix firmware and fix hardware – no flexibility
- Quality instruments: mainless PC with signal inputs and output (signal front – and analog preprocessing circuit AD and DA convertors, digital filtering SD – fixed hardware)
- High-end instruments: general purpose PC (e.g., W7) + flexible application software (measurement functions)
- High-end, high-energy consumptions, limited flexibility given by software and fixed hardware, problematic in real-time signal processing, exact and fast timing (multitasking in Windows), and similar applications
- Large, high energy consumption, limited flexibility given by software and fixed hardware
- Easy maintained (software development and installation), multitask, data presentation and archiving.

Typical convenient applications:
- Single purpose instrumentation (osilloscope, basic spectrum analyzers, logic network analyzers, etc.)
- Typical inconvenient applications:
  - Distributed systems (sensor network), control systems of machines, robust and reliable systems continuously working for a long time, fast signal processing in real-time, reconfigurable measurement system (reconfigurable signal and pattern generation, signal analyzers, etc. that cannot be processed by software in real-time)

Modern instrumentation and control – nowadays systems
- Typical nowadays system:
  - Low cost instruments (system): microcontroller with fix firmware and fix hardware – no flexibility
  - Quality instruments: mainless PC with signal inputs and output (signal front – and analog preprocessing circuit AD and DA convertors, digital filtering SD – fixed hardware)
  - High-end instruments: general purpose PC (e.g., W7) + flexible application software (measurement functions)
  - High-end, high-energy consumptions, limited flexibility given by software and fixed hardware, problematic in real-time signal processing, exact and fast timing (multitasking in Windows), and similar applications
  - Easy maintained (software development and installation), multitask, data presentation and archiving.

Real-Time Terms
- Determinism – Describes how consistently the system responds to external events or performs operations within a given time period.
- Time Critical Code - Code that must execute on a specific schedule to function as desired
- Priority - Defines when a VI or loop should execute relative to other VIs and loops

Real time operational systems
- In general, real time systems are based on real time OS (RTOS)
- What is RTOS?
  - Any OS is responsible for managing the hardware resources of a computer and hosting applications that run on the computer
  - RTOS is specially designed to run applications with very precise timing and high degree of reliability
  - To be considered “real-time”, an operating system must have a known maximum time for each of the critical operations that it performs

General Purpose Operating Systems
- Multitasking – Processor time is shared between programs
- Can preempt high priority programs
  - Many programs run in the background—screen savers, disk utilities, antivirus software, etc.
  - Must service interrupts—keyboard, mouse, Ethernet, etc.
  - High amount of jitter
  - Cannot guarantee determinism
Real-Time Operating Systems

- Multitasking - Ensure that high-priority tasks execute first
- Generally do not require user input from peripherals

LabVIEW Real-Time Module executes VIs on the following real-time operating systems:
- NI ETS (Embedded Tool Suite)
- Wind River VxWorks
- Linux

LabVIEW Real-Time System

Host Computer

RT Target

Develop

Deploy

Execute

RTOS vs. general purpose OS (GPOS)

- GPOS such a Windows is excellent platform for developing and running your non-critical measurement and control applications but not the ideal platform for running applications that require precise timing or extended uptime.
- Priorities:
  - GPOS is fair – all running tasks (threads) will achieve a processor time
  - RTOS behavior is given by programmer decision on tasks’ priorities (static, dynamic) and on environment requirements
- Interrupts latency – measured as the amount of time between when a device generates an interrupt and when that device is serviced:
  - GPOS: variable amount of time
  - RTOS must guarantee that all interrupts will be serviced within a certain maximum amount of time (interrupt latency must be bounded)
- Performance – actual performance is given mostly by hardware
  - GPOS – virtually decreased by multitasking
  - RTOS - can provide much more precise and predictable timing characteristics

Hard and Soft RTOS

- RTOS that can absolutely guarantee a maximum time for operations are commonly referred to as “hard real-time”
- RTOS that can only guarantee a maximum most of the time are referred to as “soft real-time”

RT system components

- Components of RT system:

Real-Time System Components:

Real-Time System Components: Real-Time Targets

NI RT PXI Embedded Controllers
- High speed, high channel density, I/O variety

NI CompactRIO
- Reconfigurable Embedded System

NI Compact Vision System & NI Smart Camera
- Machine vision

NI Industrial Controller
- Rugged, fast, high-performance

NI Single-Board RIO
- Reconfigurable single circuit board form factor

Desktop PC running RTOS
- Determinism for PCI systems

NI myRIO
- Education
**Host and Target Application Architecture**

- **Host Application**
  - User Interface
  - Data Storage

- **Target Application – working independently**
  - Data Storage
  - Non-deterministic Loop
  - Deterministic Loop

**Host Application**

- Runs on the host computer
- Handles non-deterministic tasks
  - Communicates with the target application
  - User interface parameters and data retrieval
  - Data logging
  - Data analysis
  - Data broadcast to other systems

**Target and Host Application Interaction**

- Processes that require determinism are time-critical processes — set all other processes to a lower priority
- Use multithreading to set the priority of a process
- Higher priority processes preempt lower priority processes

**FPGA in instrumentation**

- Field-programmable gate arrays (FPGAs) are reprogrammable silicon chips.
- In contrast to processors that you find in your PC, programming an FPGA rewires the chip itself to implement your functionality rather than run a software application.
- Ross Freeman, the cofounder of Xilinx, invented the first FPGA in 1985.

**Top 5 Benefits of Using FPGAs**

- FPGA chip adoption across all industries is driven by the fact that FPGAs combine the best parts of application-specific integrated circuits (ASICs) and processor-based systems.
  - These benefits include the following:
    - Faster I/O response times and specialized functionality
    - Exceeding the computing power of digital signal processors
    - Rapid prototyping and verification without the fabrication process of custom ASIC design
    - Implementing custom functionality with the reliability of dedicated deterministic hardware
    - Field-upgradable eliminating the expense of custom ASIC re-design and maintenance
FPGA Technology

Configurable logic blocks (CLBs)
- Basic blocks
- Flip-flops
- Lookup tables (LUTs)
The basic block can be combine in different FPGA into more complex blocks:
- Digital signal processors (DSP) slices
- Embedded block of RAM
- Multipliers
- Etc.
Note: Most applications do not require detailed knowledge of these components

Look Up Table (LUT)
- Basic FPGA Building Block
- Common uses
  - Logic functions
  - Distributed memory
  - Shift registers

FPGA Logic implementation
Implementing Logic on FPGA: $F = (A \cdot B) \oplus C \cdot D$ 

True Parallelism with FPGA
Implemented Logic on FPGA: $F = (A \cdot B) \oplus E$
$Z = (W \cdot X) \oplus Y$
Benefits:
- Reprogrammable silicon also has the same flexibility of software running on a processor-based system, but it is not limited by the number of processing cores available.
- Unlike processors, FPGAs are truly parallel in nature, so different processing operations do not have to compete for the same resources.
- Each independent processing task is assigned to a dedicated section of the chip, and can function autonomously without any influence from other logic blocks.
- As a result, the performance of one part of the application is not affected when you add more processing.
- FPGA-based applications have high execution speed, reliability, and flexibility.
- Adding a new task does not affect the other tasks.
- Low costs in comparison with other HW realization (ASIC).

Drawback:
- Missing OS for easy adding tasks, connecting disk, drivers for IO, etc.
- Difficult design of FPGA internal structure.

Benefits and drawbacks of FPGA

- Fast real-time signal processing:
  - Fast digitizers with onboard signal processing, e.g., Agilent
  - Reconfigurable signal analyzers with demodulation and analysis in real time (e.g., wireless networks with different standards)
  - Robust components of sensor network, smart sensors
  - Reliable data acquisition and control systems for industry technologies, robots, …, including marine technology, space and aeronautic systems, etc.
  - Reconfigurable tester for product inspection and new products development.

FPGA applications in instrumentation

- Traditional tools:
  - Hardware description languages (HDLs) such as VHDL and Verilog
  - Text-oriented describing internal structure a mapping chip IO to internal signals
  - To verify the design a test bench needs to be developed simulating input signal and acquiring outputs
  - Verified design is compiled into bit form describing internal structure of FPGA

Intellectual Properties (IP) are key building blocks of Xilinx Targeted Design Platforms
FPGA Hardware to Bitfile
Function Nodes – Case Study

- Xilinx Compiler – Create bitfile

Schematic

Bitfile

High-Level Synthesis Design Tools

- Simplification of design – no HDL is needed
- Verification in user friendly environment known from software development and debugging.
- Example for projects: LabVIEW FPGA

Abstraction to the Pin

VHDL

LabVIEW FPGA

LabVIEW, LabVIEW Real Time, LabVIEW FPGA

Graphical System Design
A Platform-Based Approach

LabVIEW

System Design to Deployment
Models of Computation
Real time application is always developed on PC
- Real time hardware (target) must be first detected and configured by MAX
- Project in LabVIEW must be created
  - Real time target must be added into the project
  - Under the target a new VI running in the real time environment must be created

**Preparation**

- Host Computer
  - Displays front panel of RT Target VI
- User Interface
  - Communication
  - Only for development
- RT Target
  - Executes the block diagram logic of RT Target VI

**A. Front Panel Communication**

**B. Network Communication**

<table>
<thead>
<tr>
<th>Protocol</th>
<th>Common Use</th>
<th>Speed</th>
<th>Deterministic Read/Write</th>
<th>Advantages</th>
<th>Caveats</th>
</tr>
</thead>
<tbody>
<tr>
<td>Network Streams</td>
<td>Data streaming</td>
<td>Faster</td>
<td>No</td>
<td>Built-in functions</td>
<td>No</td>
</tr>
<tr>
<td>TCP</td>
<td>Data streaming</td>
<td>Fastest</td>
<td>No</td>
<td>High transfer rates, standard protocol</td>
<td>No</td>
</tr>
<tr>
<td>UDP</td>
<td>Broadcast latest values</td>
<td>Fastest</td>
<td>No</td>
<td>High transfer rates, standard protocol</td>
<td>String data, Lossy</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Protocol</th>
<th>Common Use</th>
<th>Speed</th>
<th>Deterministic Read/Write</th>
<th>Advantages</th>
<th>Caveats</th>
</tr>
</thead>
<tbody>
<tr>
<td>Network Publishing Shared Variable</td>
<td>Latest value, host interface</td>
<td>Fast</td>
<td>With RT FIFO enabled</td>
<td>Ease of programming</td>
<td>LabVIEW only</td>
</tr>
<tr>
<td>Network Streams</td>
<td>Data streaming</td>
<td>Faster</td>
<td>No</td>
<td>Built-in functions</td>
<td>No</td>
</tr>
<tr>
<td>TCP</td>
<td>Data streaming</td>
<td>Fastest</td>
<td>No</td>
<td>High transfer rates, standard protocol</td>
<td>No</td>
</tr>
<tr>
<td>UDP</td>
<td>Broadcast latest values</td>
<td>Fastest</td>
<td>No</td>
<td>High transfer rates, standard protocol</td>
<td>String data, Lossy</td>
</tr>
</tbody>
</table>
Deploying Real-Time Application

- Creating a stand-alone application (executable) of a LabVIEW Real-Time Module application using the LabVIEW Application Builder
- Benefits of deploying your RT Application:
  - Embed executable in non-volatile memory on the target
  - Launch executable automatically when target boots
  - Review the code for unsupported functions
  - Functions that modify front panel objects
  - Functions that use technologies specific to other operating systems

Avoid Modifying Front Panel Objects and Using OS specific Technologies

- Front panel Property Nodes and control references
- Dialog functions
- VI Server front panel functions
- Do Not Use OS-Specific Technologies

Examples:
- ActiveX VIs
- .NET VIs
- Windows Registry Access VIs
- TextStand VIs (ActiveX-based)
- Report Generation Toolkit VIs
- Cursor VIs
- Call library nodes that access an operating system API other than ETS or VxWorks
- Graphics and Sound VIs
- Database Connectivity Toolkit
- XML DOM Parser and C Web Server for CGI Support

LabVIEW FPGA

Evaluating System Requirements

- Need FPGA?
- Select RIO Architecture
  - FPGA for Windows
  - FPGA for Real-Time
  - PCI/PCle
  - PXI/PXie
- Select a Platform
  - PC/PCle
  - FPGA for Windows
  - FPGA for Real-Time
- Other Considerations
  - FPGA “size”
  - I/O
  - Connectors

FPGA – Windows

Example: FlexRIO

PXI/PXie/PCle FPGA module + adapter
Adapter: additional electronics protecting FPGA inputs and creating additional measurement functions, e.g.: RF transceiver, fast digitizer (oscilloscope), fast DIO, ...
FPGA – Real-Time

Windows System
LabVIEW Real-Time System (cRIO, sbRIO, R Series in RT PXI, etc)
FPGA
Host VI
LabVIEW FPGA
LabVIEW Real-Time
LabVIEW FPGA

Example: cRIO, sbRIO, myRIO

- cRIO: Chassis with FPGA + RT processor or embedded + slots for DAQ modules
- sbRIO: single board cRIO with integrated DAQ modules
- myRIO: educational module similar to sbRIO

LabVIEW Programming & CompactRIO or myRIO

User Interface
LabVIEW
Real-Time Processors
LabVIEW Real-Time
Reconfigurable FPGA
LabVIEW FPGA

FPGA Development Flow

Evaluate System Requirements
Select Architecture
Configure Hardware
Create a LabVIEW Project

Create FPGA VI
Simulate on PC to test
Compile to FPGA
Create FPGA Host VI(s)
Create Real-Time Host VI(s)

Project with CompactRIO Controller

IO are connected to processor across FPGA

IO access:
1. The „Scan Engine“ is a prewritten LabVIEW FPGA personality that automatically reads values from I/O modules (myRIO up to 10kHz rate) – no need to program FPGA.
2. Own personality (VI configuring FPGA and optionally including digital signal preprocessing, control, timing, fast complex calculations, etc.).

IO access:
- IO access: 1. The „Scan Engine“ is a prewritten LabVIEW FPGA personality that automatically reads values from I/O modules (myRIO up to 10kHz rate) – no need to program FPGA.
- 2. Own personality (VI configuring FPGA and optionally including digital signal preprocessing, control, timing, fast complex calculations, etc.).
Data transfer FPGA – RT application

- Direct writing/reading from virtual front panel of FPGA VI
- Simple data transfer without memory
- FIFO – DMA transfer with buffered data transfer
- Register
- Handshake
- Memory
- FPGA I/O variable
- See the demonstration later for more details

What is myRIO

- National Instruments myRIO-9100 is a portable reconfigurable I/O (RIO) device that students can use to design control, signal processing, robotics, and mechatronics systems

Xilinx Zynq 7000 family

www.ni.com/myrio

myRIO

What is inside?

Analog inputs and outputs
MXP connectors

MSP connector

NI myRIO Palette

Links

- www.ni.com/myrio
- www.ni.com/labview
- www.ni.com/crio
- www.ni.com/flexrio
- www.ni.com/...