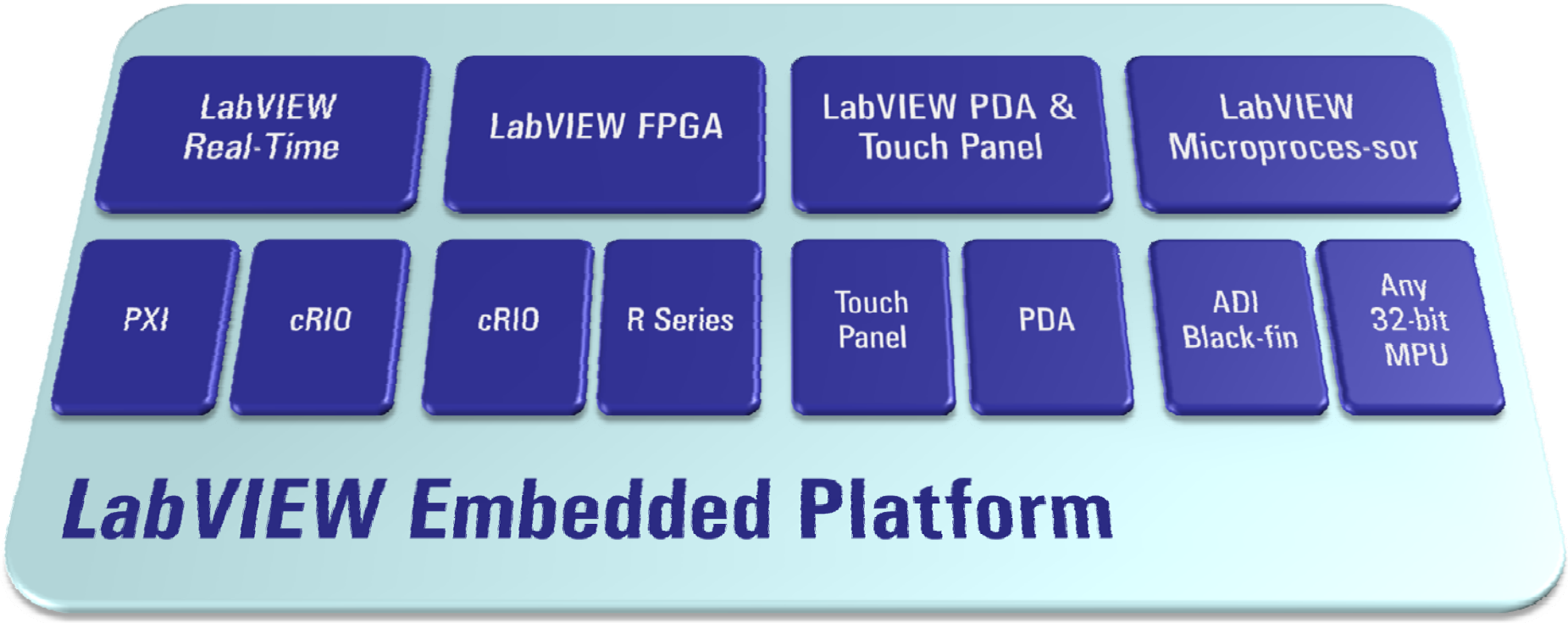
Oslo - Stockholm - Utrecht - Brussels - Copenhagen - Helsinki

ni.com/nidays

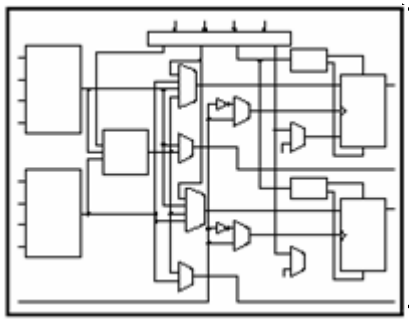
NATIONAL INSTRUMENTS

COMMON LABVIEW FPGA APPLICATIONS

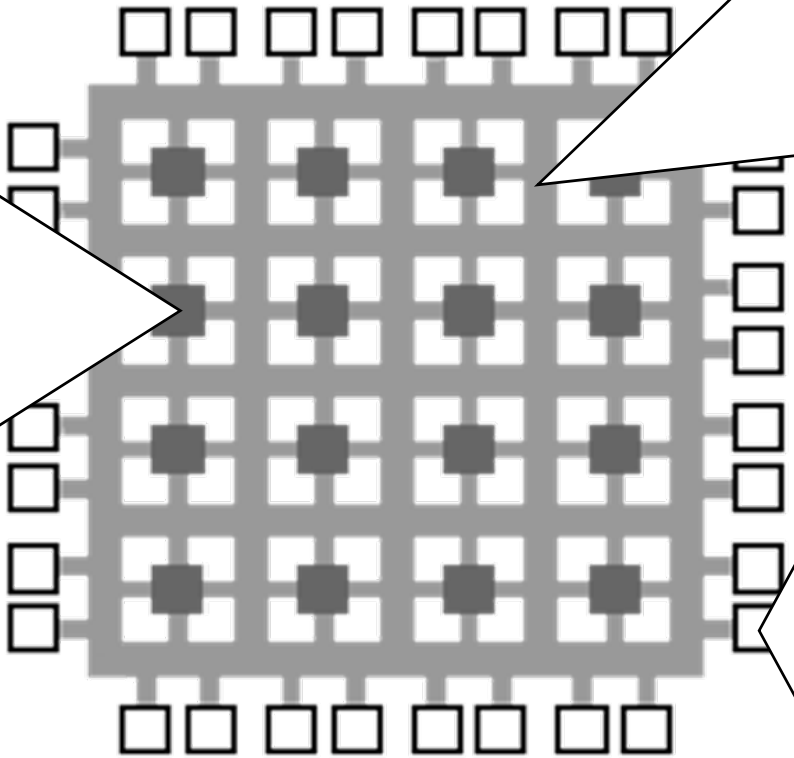
LabVIEW Embedded Product Family

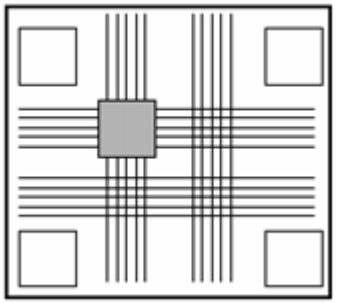


FPGA Technology

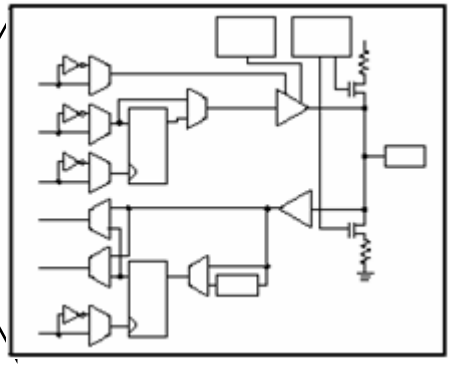


Logic Blocks



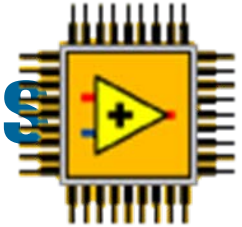


Programmable Interconnects



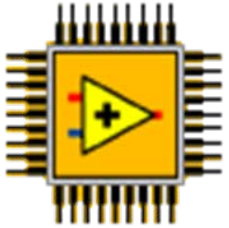
I/O Blocks

Importance of FPGA in Systems



- **High Reliability** – Designs become a custom circuit
- **High Determinism** – Runs algorithms at deterministic rates down to 25 ns (faster in many cases)
- **True Parallelism** – Enables parallel tasks
- **Reconfigurable** – Create new and alter existing task-specific personalities

Common LabVIEW FPGA Applications



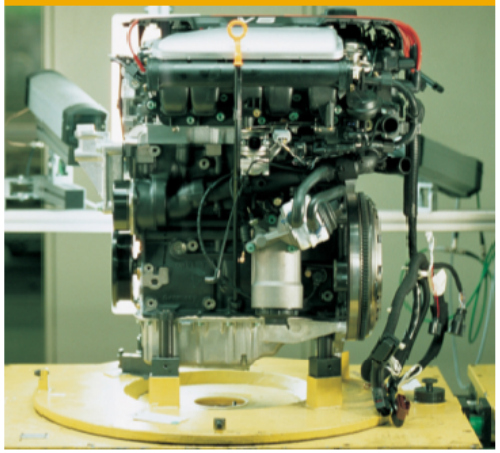
- High-speed control
- Intelligent DAQ
- Digital communication protocols
- Sensor simulation
- Onboard processing and data reduction
- Co-processing

Spectrum of Real-Time Applications

Deterministic Performance

Wind Tunnel Control

Maximum Reliability



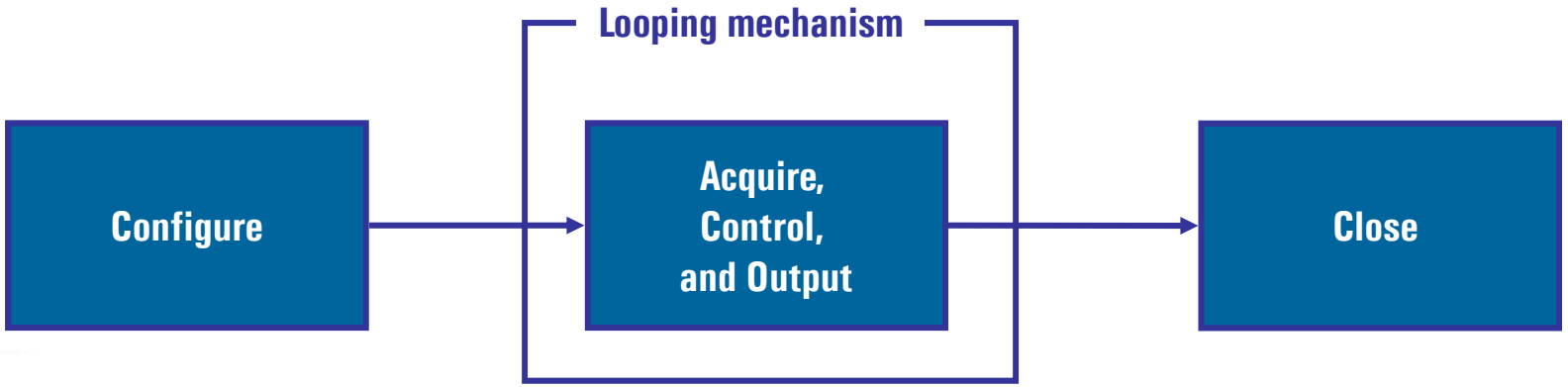
Endurance Testing

Autonomous Operation

Safety Monitoring

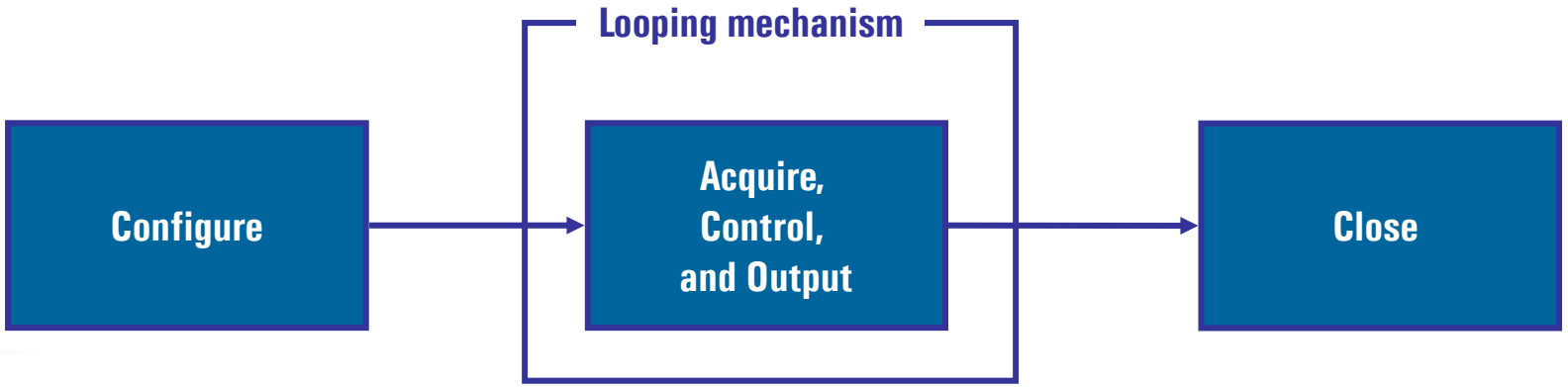
Real-Time Control Architectures

- SW Timed Solutions:
  - Insert Wait function in loop
  - Insert Wait Until Next ms multiple function in loop
  - Replace looping mechanism with timed loop



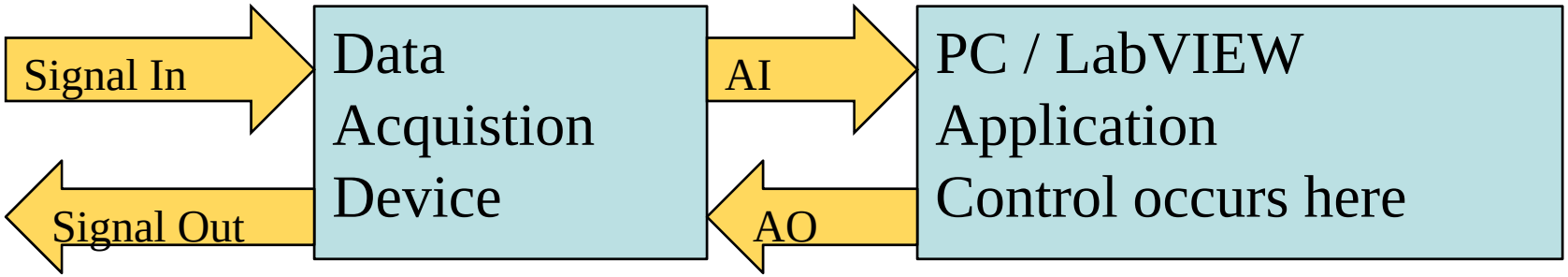
Real-Time Control Architectures

- HW Timed Solutions:
  - Insert Wait function with µs resolution in loop
  - Insert Wait Until Next Multiple function with µs resolution in loop
  - Replace loop with a Timed Loop linked to µs clock or external clock
  - Use DAQmx VIs to connect to external clock



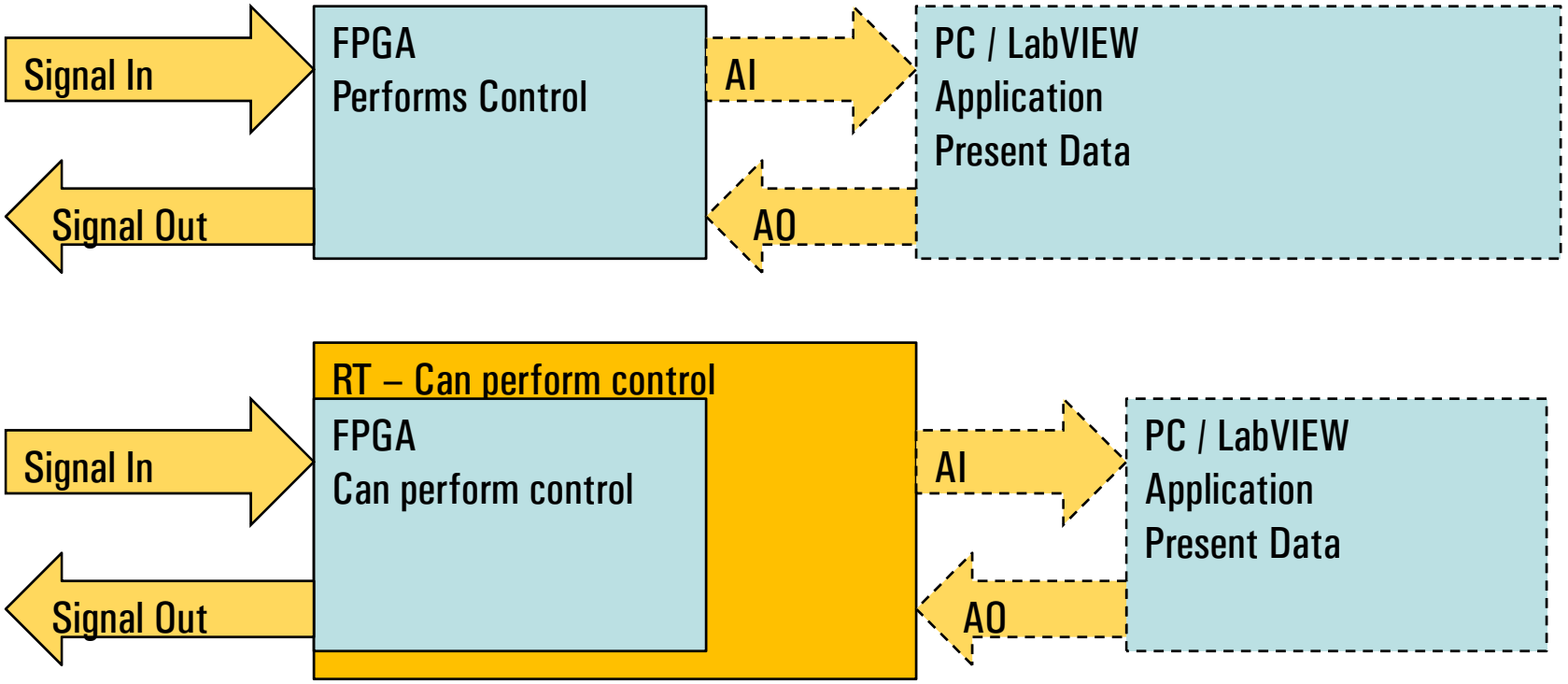
High-Speed Control

- A traditional control system consists of:
  - Target running a RT operative system for high determinism, low jitter and high stability
  - Device acquiring data such a data acquisition board
  - Application where the controller is implemented



High-Speed Control

- Possible FPGA/RT Control implementations



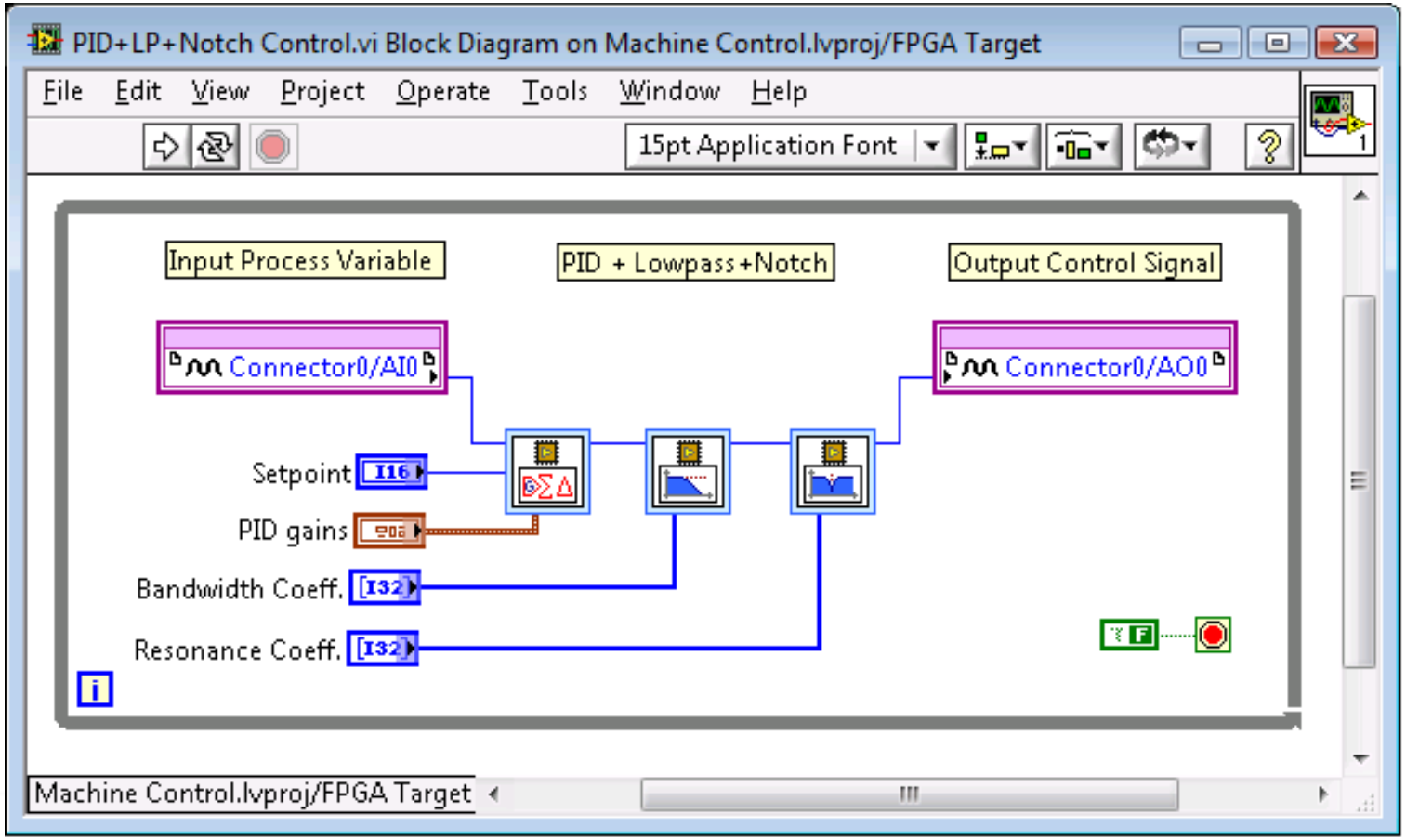
High-Speed Control

Microprocessor Based System		FPGA Based System
Performance limited to xx kHz	⇨	Closed Loop performance beyond 1MHz
Serial execution on single-core	⇨	Parallel execution, multi-rate control
Slow down when more loops added	⇨	No slow down when more loops added
OS runs control logic	⇨	Control logic in HW
I/O modules fixed functionality	⇨	I/O Functionality reconfigurable
Custom circuitry requires board layout	⇨	SW defined gate array

High-Speed Control

- Continuous-time systems - described by differential equations (Analog world)
- Discrete-time systems are described by difference equations (Digital world)
  - Signals are sampled periodically
  - Discrete controllers are implemented on FPGAs

High-Speed Control



About 200 kHz Loop Rate

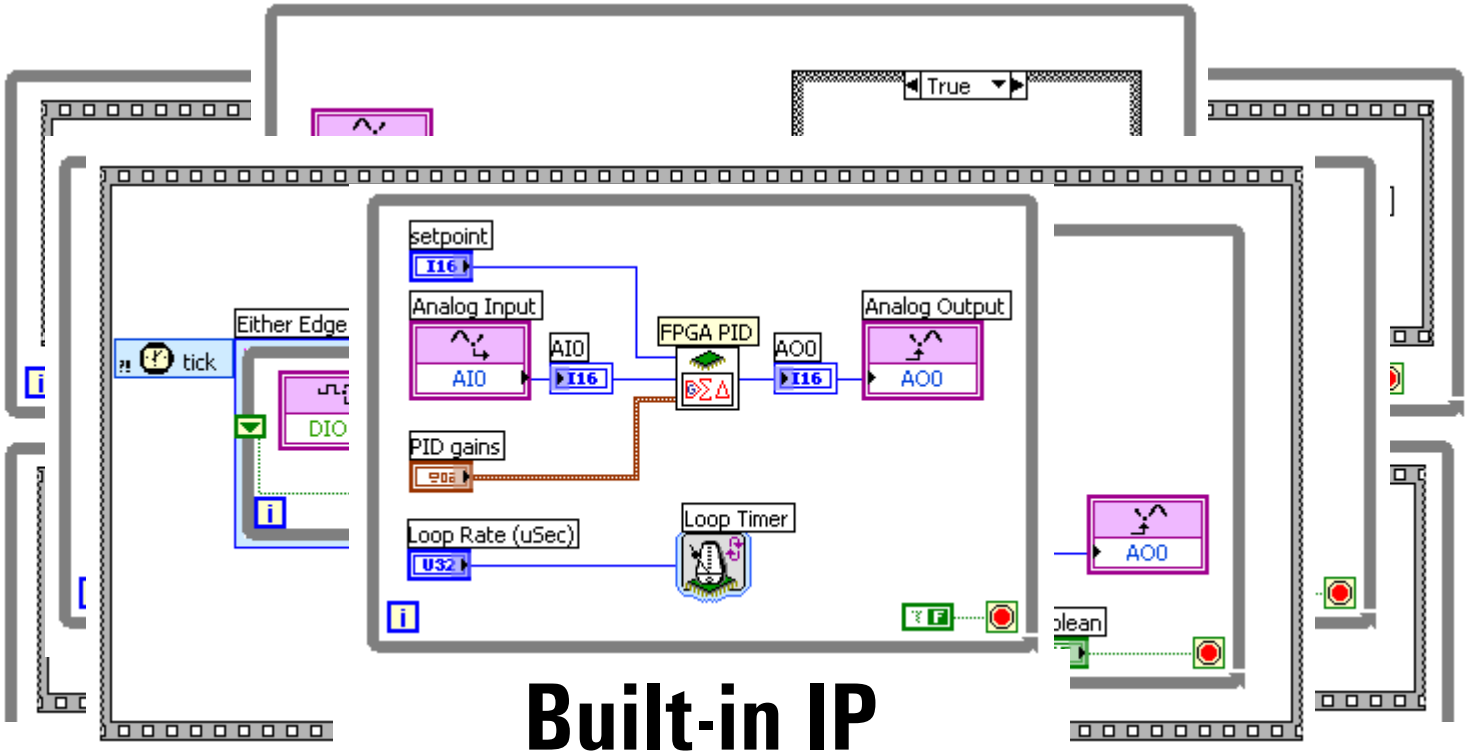
Intelligent DAQ

- Increased flexibility
  - Custom triggering
  - Create own counters (using digital lines)
  - Multi-rate applications
  - Implement custom filtering
  - Perform ‘oversampling’ and data reduction on HW

Intelligent DAQ

	Typical Multifunction DAQ Device	PXI-7831R
I/O Control	ASICs for counter/timer operations, triggering, etc.	Configured with LabVIEW FPGA Module.
Custom onboard decision making	N/A	Configured with LabVIEW FPGA Module.
Timing and Synchronization	Driver functions for signal routing, clock sharing, triggering, pulse measurement/generation.	LabVIEW constructs, such as While Loops, Sequence structures, Wait functions, etc., implemented in hardware in real time.
Analog Input	16 single-ended or 8 differential multiplexed.	8 differential, independent. Can be synchronized.
Analog Output	2 differential independent.	8 differential, independent. Can be synchronized .
Digital I/O	8 lines independently configurable as input or output.	96 lines independently configurable as input or output – static or synchronous.
Counter/timers	2 general-purpose counter/timers.	Any of above 96 lines easily configured as custom counters.

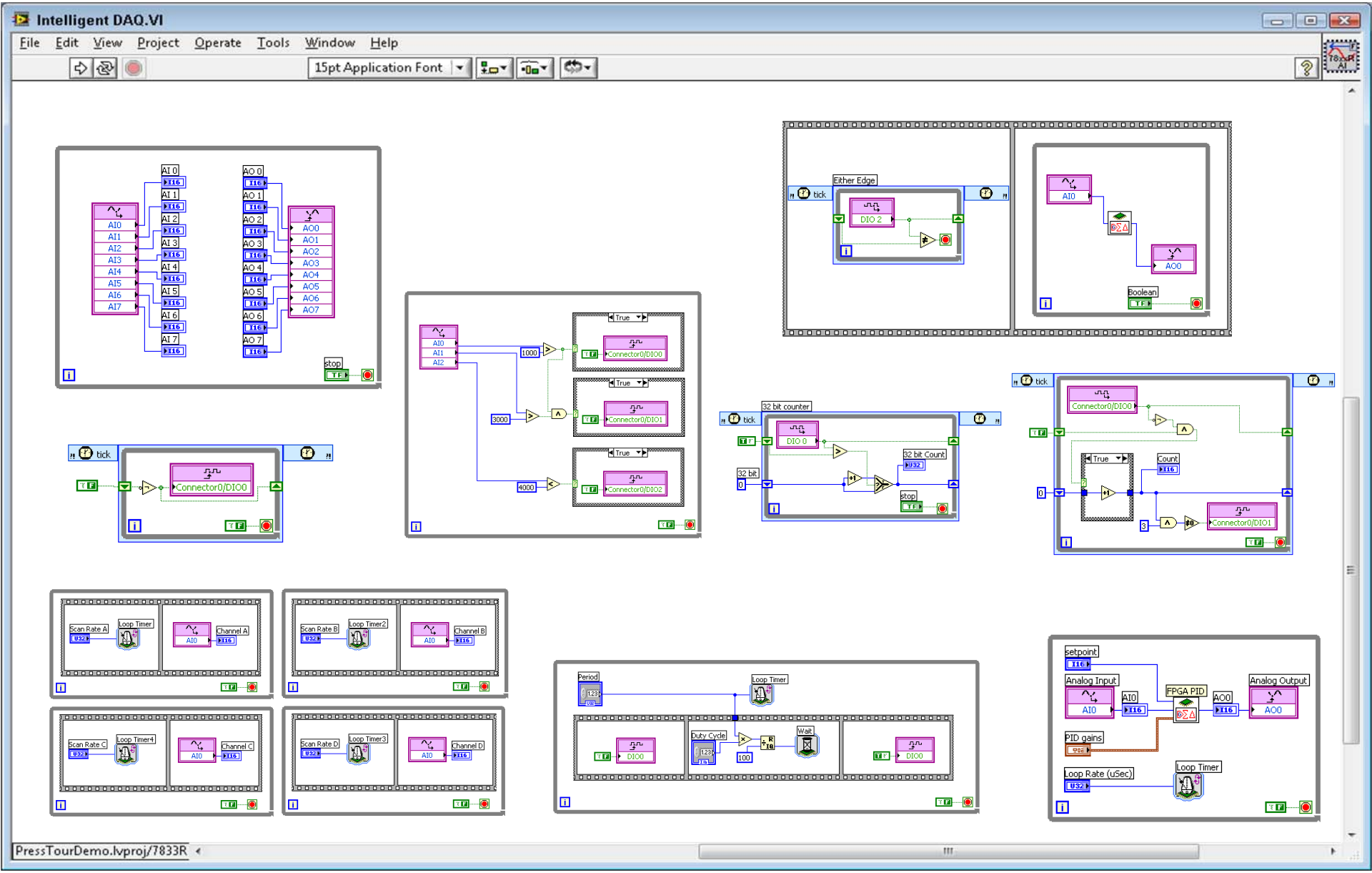
Intelligent DAQ



Built-in IP

Processing blocks

Cus hronization

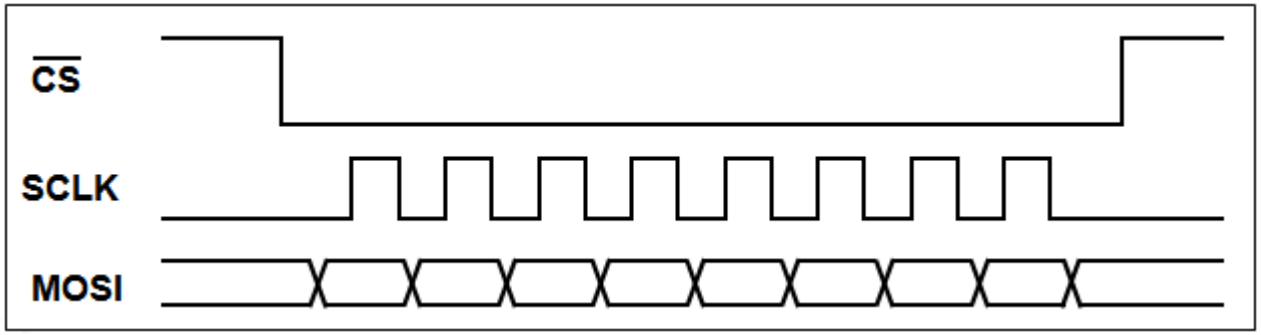


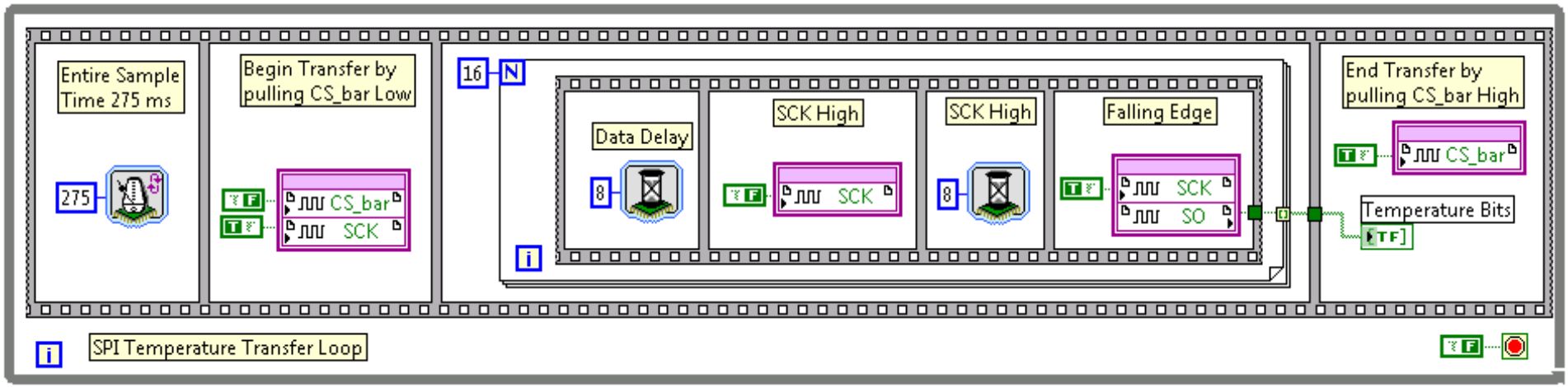
Digital Communication

- Design, implement and test digital communication protocols
  - Standard Protocols such as SPI, I2S, Slimbus, I2C, RS-232, RS-485, S/PDIF
  - Custom digital protocols
- Make use LabVIEW FPGA IP blocks or existing VHDL/Verilog code
- LabVIEW IP Cores covering digital buses and protocols available at www.ni.com/ipnet

Digital Communication

Example – SPI

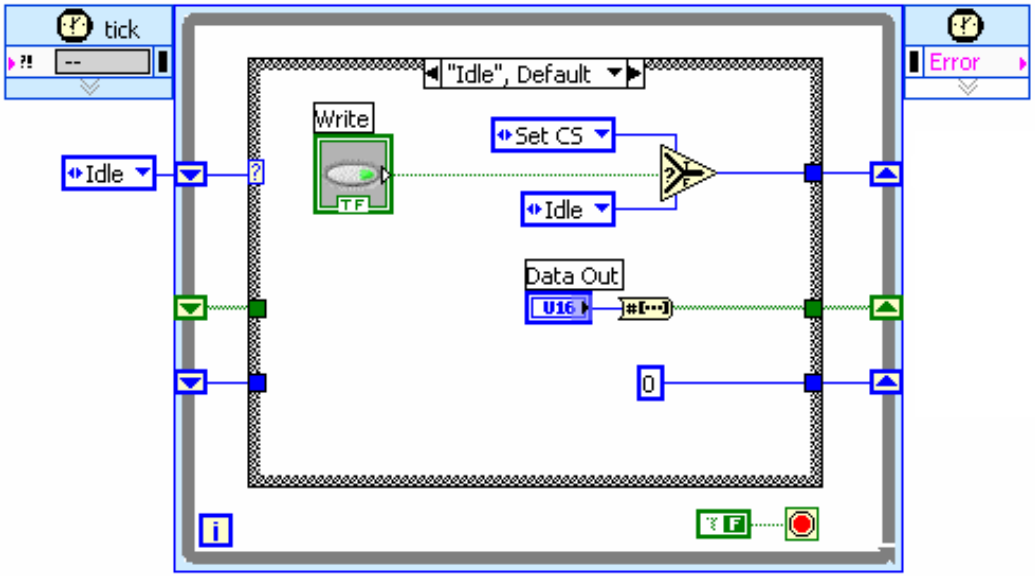




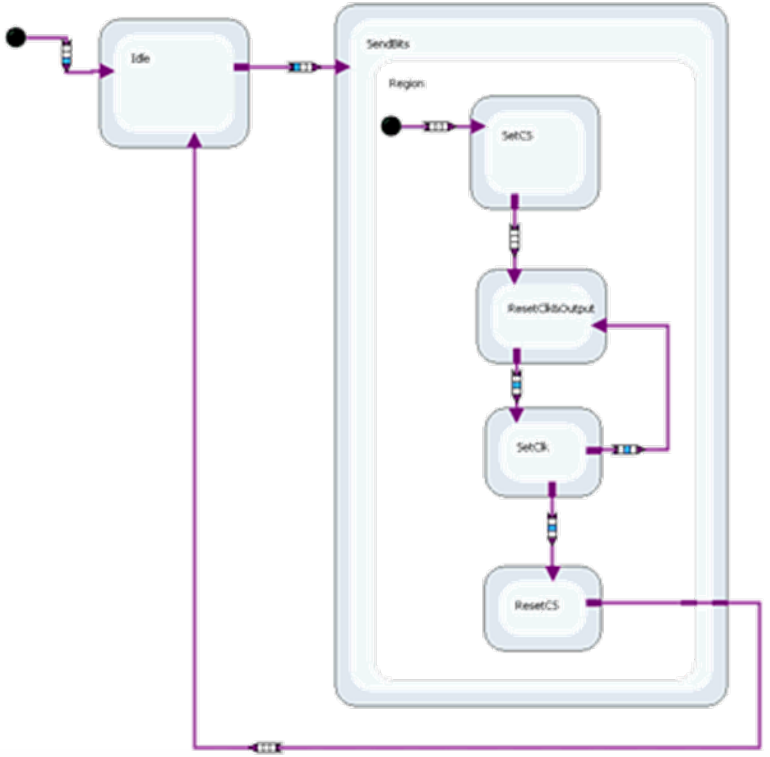
Digital Communication

Example – SPI

LabVIEW state machine



LabVIEW statecharts



Sensor Simulation and FPGA

Scenario: Company should develop a new Engine Control Unit for a vehicle

- Modeling of engine using system identification for an example
- Develop controller SW, implement it on RT target against the actual engine and verify its performance (rapid control prototyping)
- When the controller SW is implemented on final target (i.e. on the ECU) as a prototype one can either test it against the actual engine

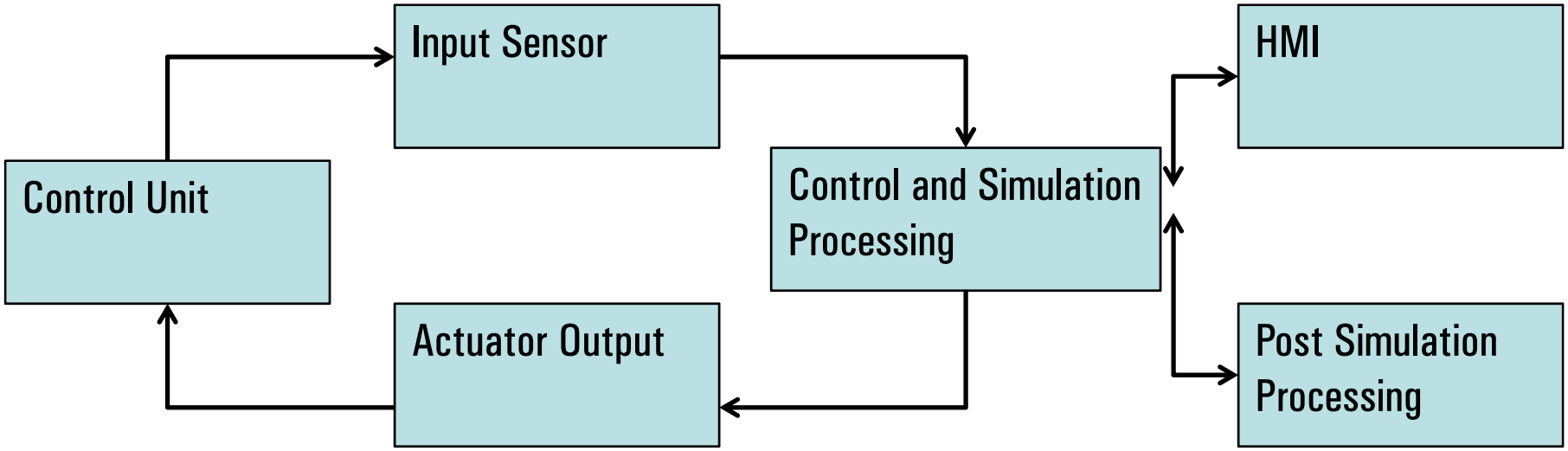
Or...

Sensor Simulation and FPGA

- Save time and money and improve safety by connecting the prototype ECU to a RT system with a FPGA board that can simulate various sensors and actuators:
- Simulate items such as:
  - Linear Variable Differential Transformer (LVDTs)
  - Resolvers
  - Thermocouples
  - Quadrature Encoders
  - Pulse-Width-Modulated (PWM) Signals
- LabVIEW IP Cores for sensor simulation – www.ni.com/ipnet

Sensor Simulation and FPGA

Typical Hardware-In-the-Loop simulation system architecture



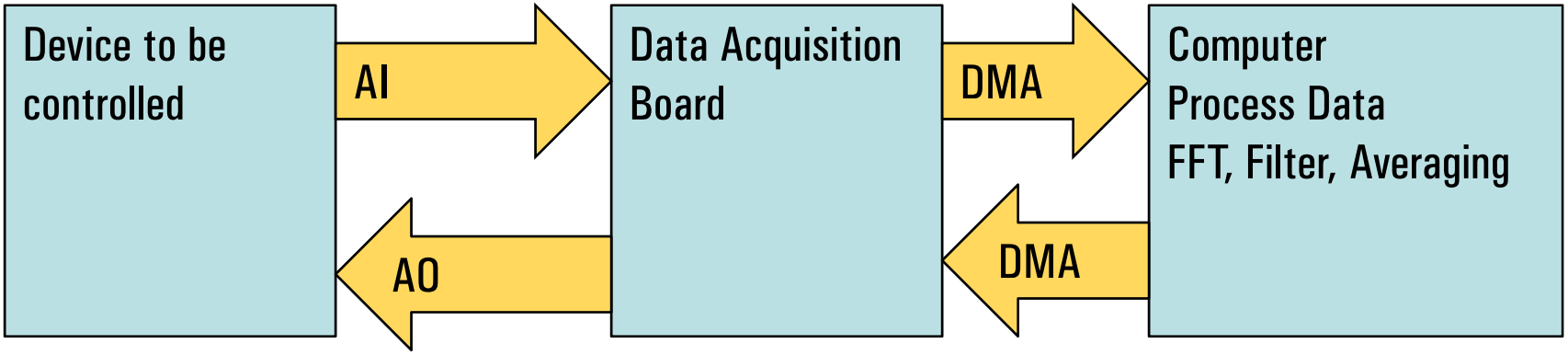
Sensor Simulation and FPGA

- Many types of sensors – fully customizable hardware
- Parallelism – many sensors on chip with no interference
- Strict timing requirements – deterministic or highly realistic
- Onboard processing – engineering units to sensor signals

Sensor Signals

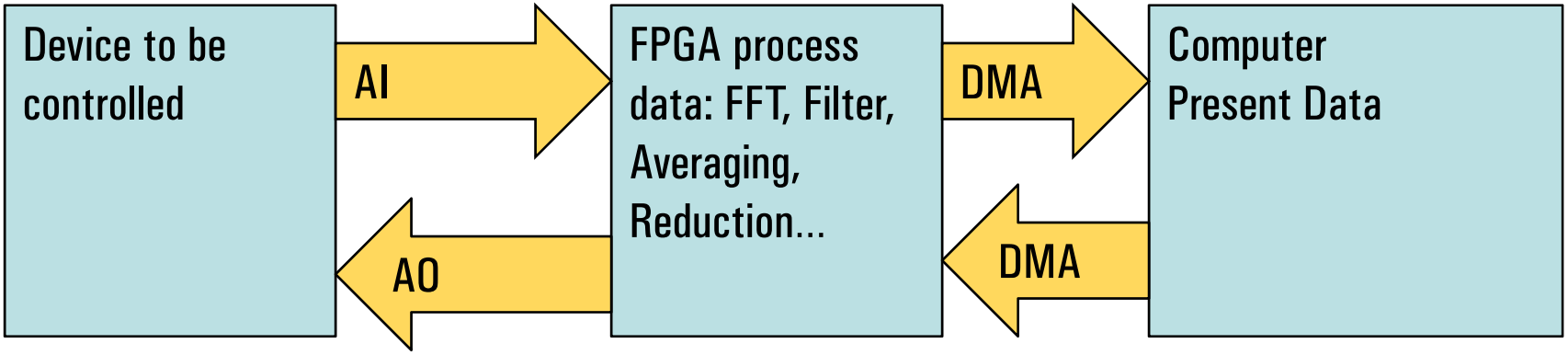
Onboard Processing and Data Reduction

Typical data acquisition + signal processing application



Onboard Processing and Data Reduction

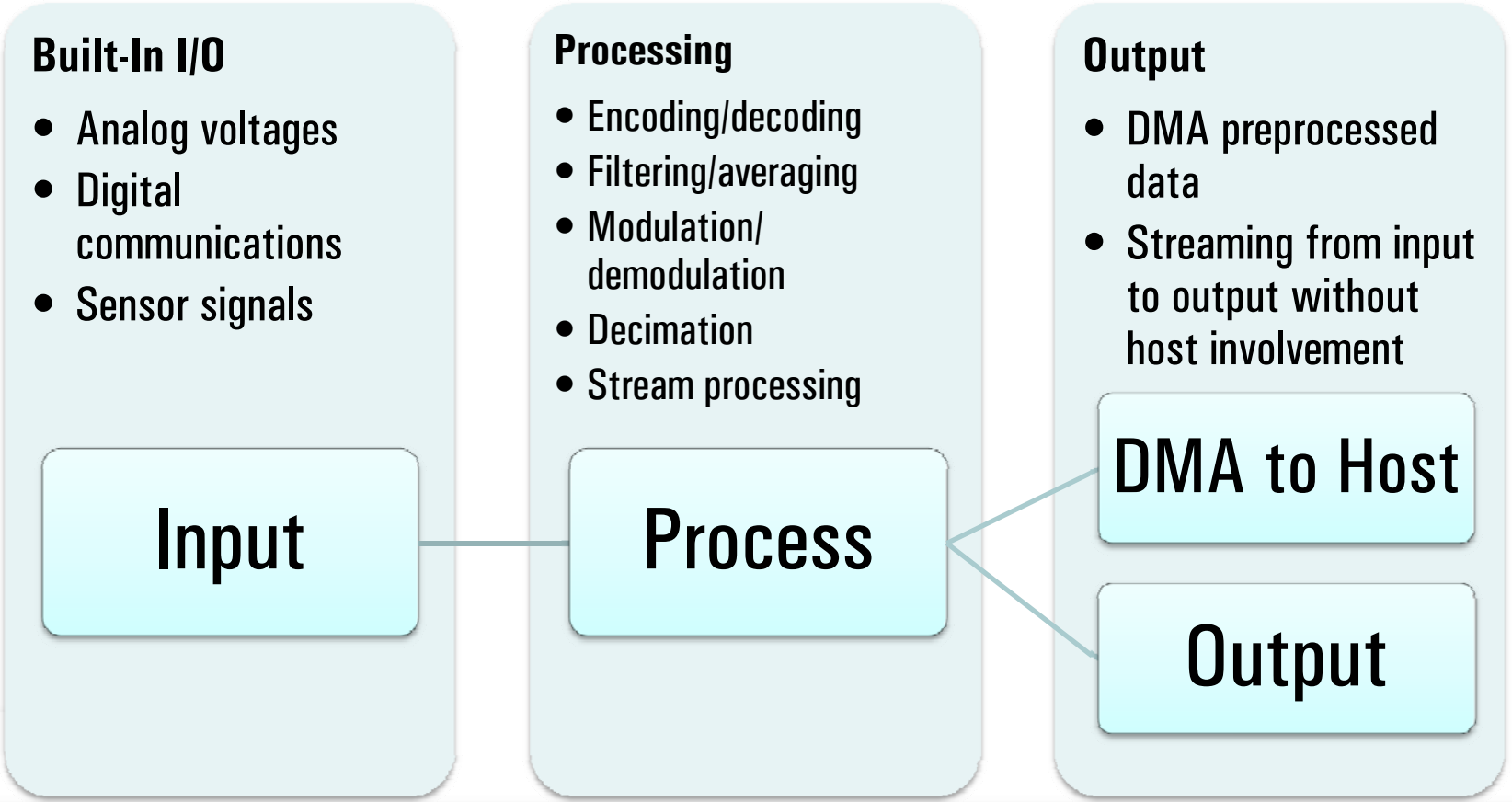
Alternative solution using FPGA



Onboard Processing and Data Reduction

- Signal Generation and Signal Processing LabVIEW IP Cores found at www.ni.com/ipnet
  - PID
  - Filters – Butterworth and Notch Filters
  - DC & RMS Measurements
  - FFT
  - Sine, Square and Sawtooth generation
- Fixed Point Math Library available at www.ni.com/labs
  - FPGA IPs for elementary mathematical functions

Onboard Processing and Data Reduction



FPGA Co-Processing

- Trends today include more and more complex applications that require:
  - High signal processing performance
  - High memory bandwidths
  - High degree of parallelism, multiple tasks in parallel
- Common application areas that require this include:
  - Video broadcasts
  - Scientific computing
  - Signal processing in consumer electronics (audio and video)
  - Telecom

FPGA Co-Processing

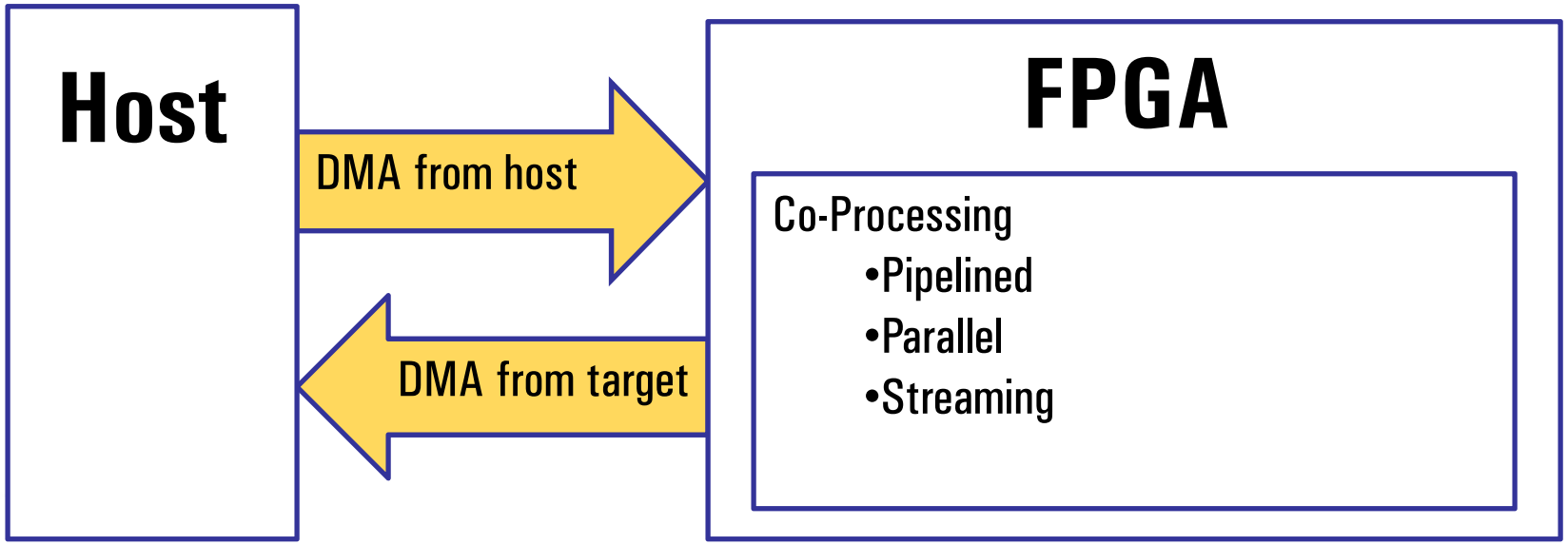
- What is FPGA Co-Processing?
  - System consisting of a microprocessor or a digital signal processor (DSP) in combination with a FPGA
- Microprocessor/DSP benefits:
  - High clock rates
  - Floating Point performance
  - Abstraction when it comes to memory and built in I/O interfaces
- FPGA Benefits:
  - High number of instructions per clock
  - Lots of multipliers
  - Truly parallel execution

FPGA Co-Processing

- What can I gain by using FPGA Co-Processing?
  - Offload the microprocessor/DSP
  - Improved performance
  - Lower Cost
- Make use of the FPGAs parallel capabilities
  - Execute computationally intensive blocks using an efficient parallel implementation on a FPGA
  - Improve performance using pipelining
  - High compute throughput even at low clock rates due to the parallelism of the FPGA

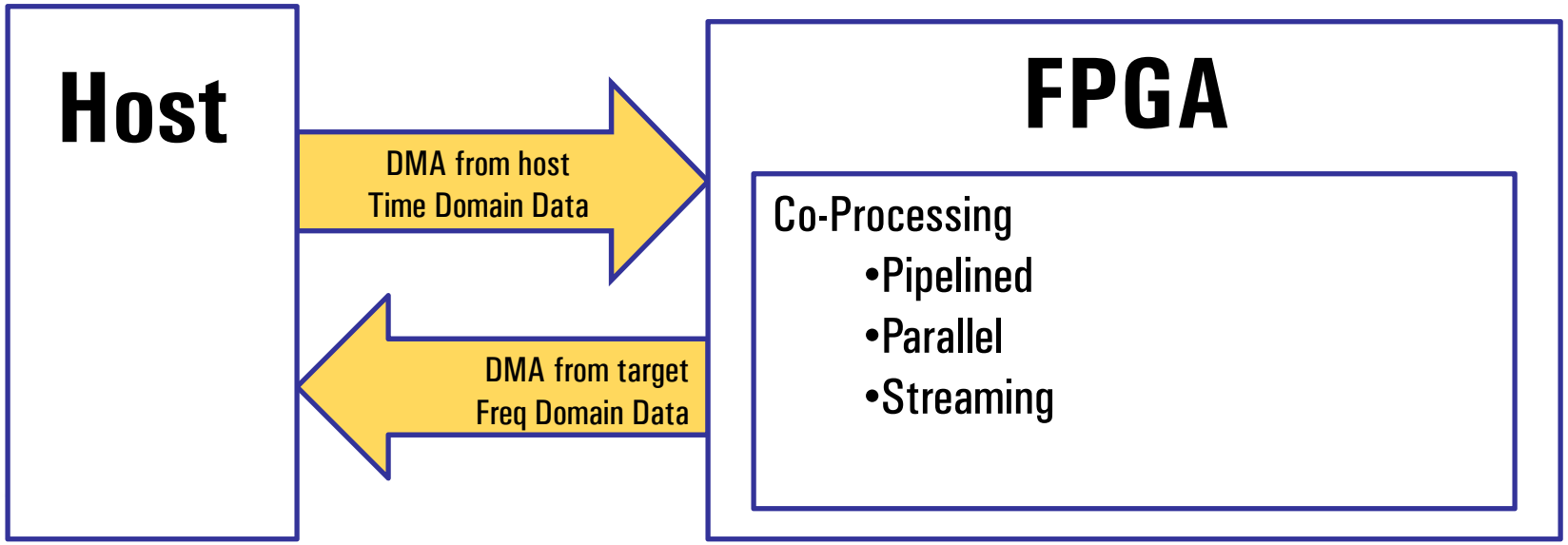
FPGA Co-Processing

Principal layout of FPGA Co-Processing



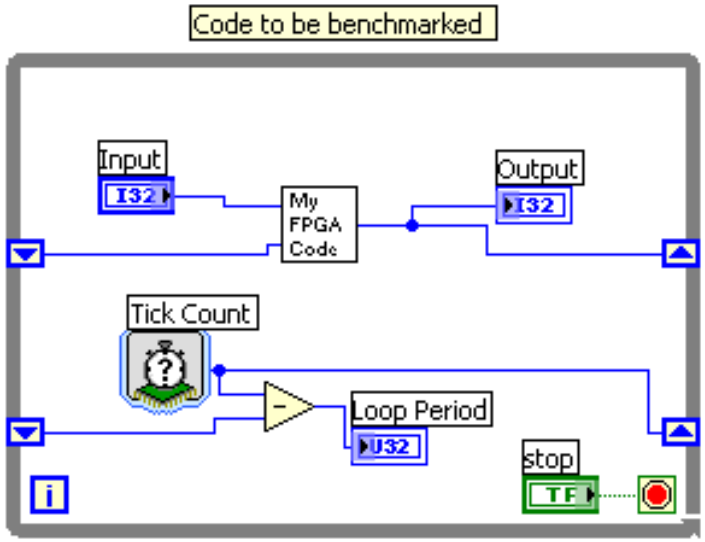
FPGA Co-Processing

Current FFT Implementation on FPGA (Beta)
Available on ni.com/ipnet
FPGA hardware target used as a co-processor.

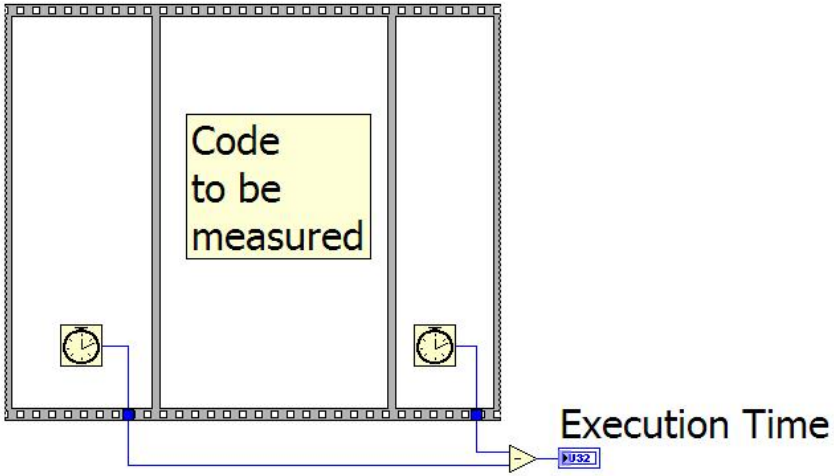


FPGA Tips to improve performance

Benchmark loop rates and code snippets!



FPGA code check loop rate



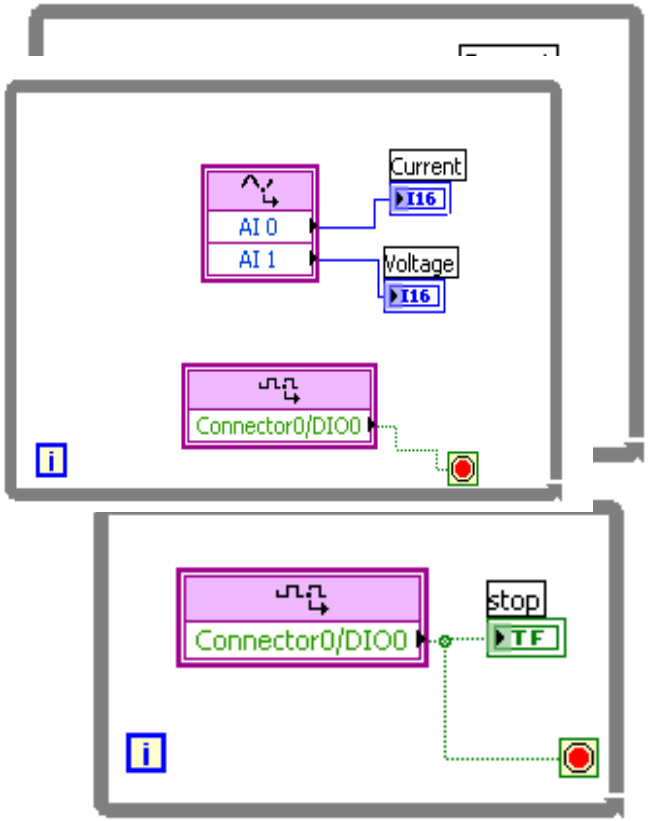
General layout for benchmarking

FPGA Tips to improve performance

Parallelism Example

- Loop rates limited by longest path
- AI takes 170 ticks, DI takes 1 tick
- Separate functions to allow DI to run independent of AI
- DI can now execute faster and detect smaller pulses

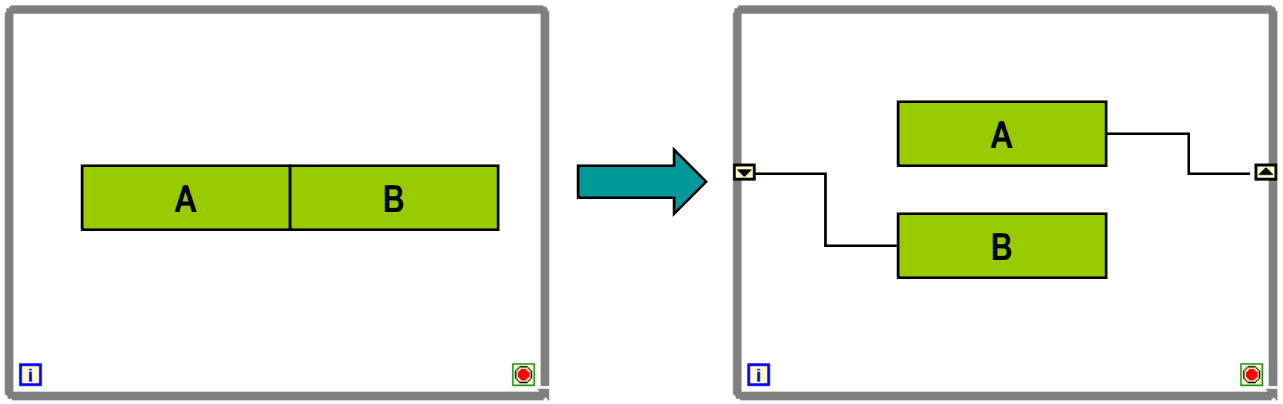
173 Ticks ~ 4.3uSec



4 Ticks ~ .1 uSec

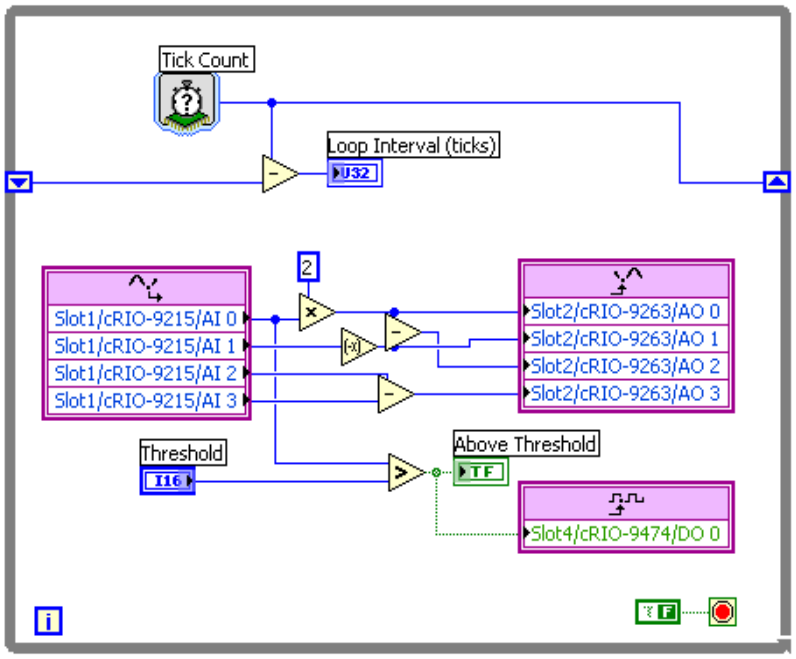
FPGA Tips to improve performance

- Within a loop you can split up your code into different loop iterations to reduce the length of each iteration
  - Handle different parts of the process flow in parallel within one loop iteration
  - Pass data to the next using shift registers
  - Critical path is reduced

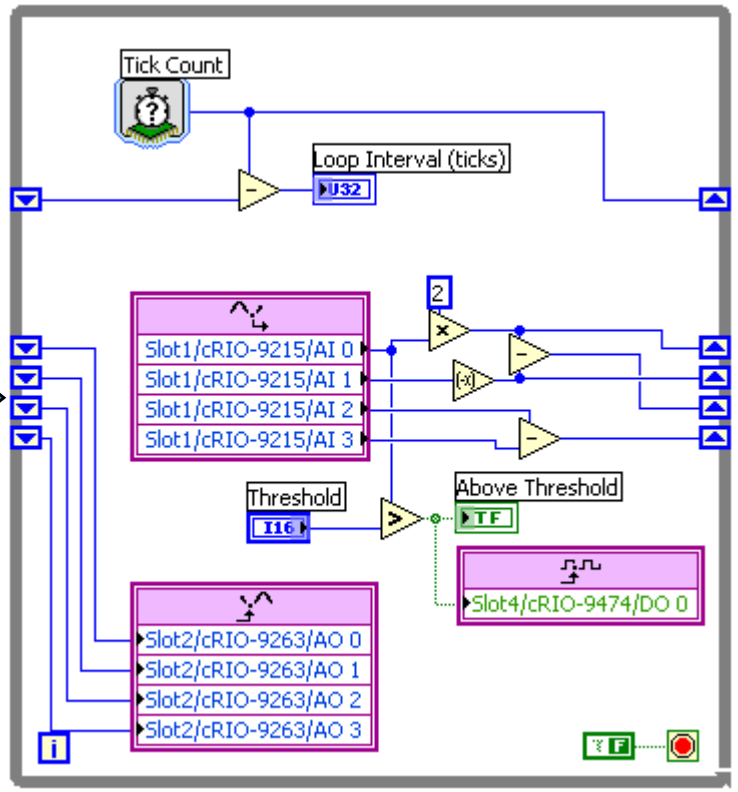


FPGA Tips to improve performance

Pipelining Example



720 clock cycles (18 µs)



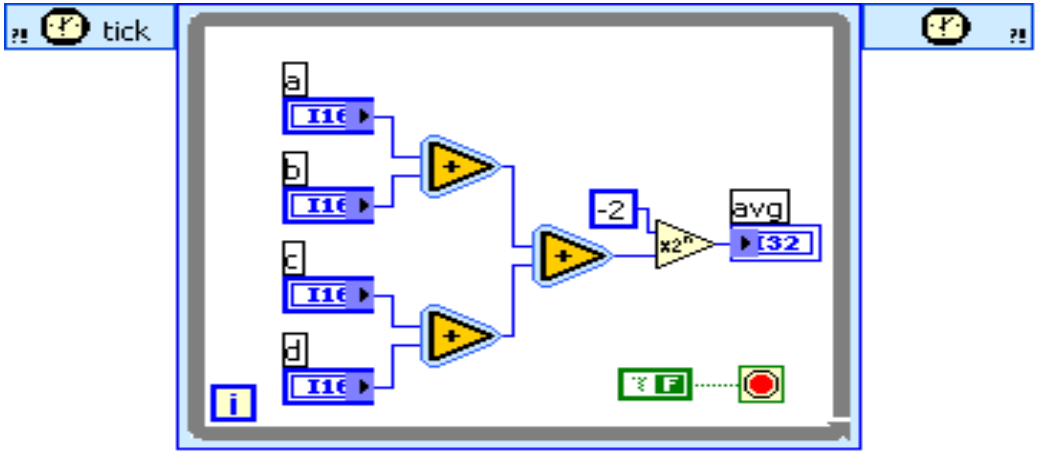
365 clock cycles (9.13 µs)

FPGA Tips to improve performance

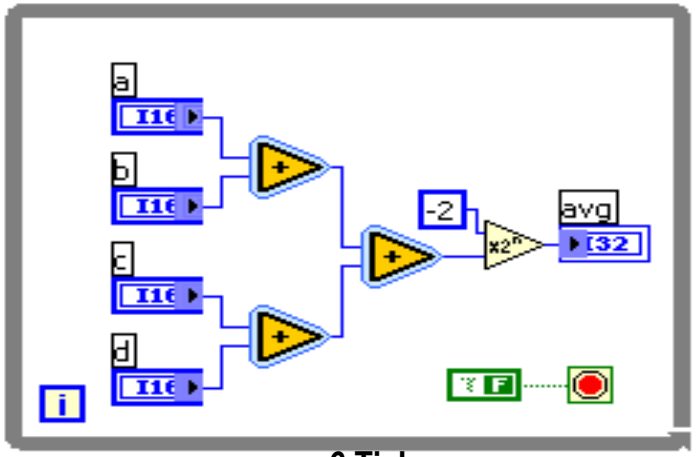
- Single Cycle Timed Loop (SCTL)
  - Minimizes synchronization and enable chain overhead
  - All operations within the SCTL loop must fit into one clock cycle
  - Code generator error message
  - Use pipelining to separate code into smaller pieces
  - Use shift register counters to implement For loop functionality within a SCTL
- Note that all functions can not be used within the SCTL!
  - Long sequences of serial code
  - Loop timer, wait functions
  - AI and AO nodes
  - Quotient and Remainder function (use **Scaly by Power of 2** or divide)
  - While Loops

FPGA Tips to improve performance

SCTL Example



1 Tick



6 Ticks