NIDays

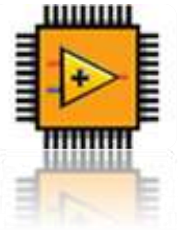
WORLDWIDE GRAPHICAL SYSTEM DESIGN

CONFERENCE

NATIONAL INSTRUMENTS

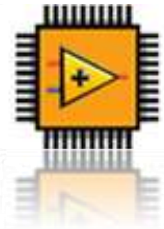
Programming Techniques for LabVIEW FPGA Developers

Agenda



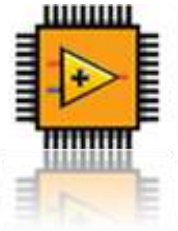
- Part 1 – Optimization Techniques
  - Pipelining (Technique 1)
  - SCTL (Technique 2)
- Part 2 – New features in LabVIEW FPGA 2009
  - Compilation improvements
  - Timing violation debugging
  - New and improved IP
  - Programmatic I/O discovery and configuration (w/RT)
  - Add FPGA data to the NI Scan Engine (w/RT)
- Part 3 – Hardware updates: Expansion I/O, controllers and chassis

Agenda



- Part 1 – Optimization Techniques
- Part 2 – New features in LabVIEW FPGA 2009
- Part 3 – Hardware updates: Expansion I/O, controllers and chassis

Agenda



- **Part 1 – Optimization Techniques**
  - **Pipelining (Technique 1)**
  - **SCTL (Technique 2)**
- Part 2 – New features in LabVIEW FPGA 2009
- Part 3 – Hardware updates: Expansion I/O, controllers and chassis

Optimization Techniques

FPGA VIs are limited primarily in two areas:

1. Speed
   - Execution rate too slow for specifications
2. Size
   - Requires too much space on the FPGA
   - Uses so much RAM that it will not compile

To increase speed and reduce size, optimize FPGA code

Optimization Techniques

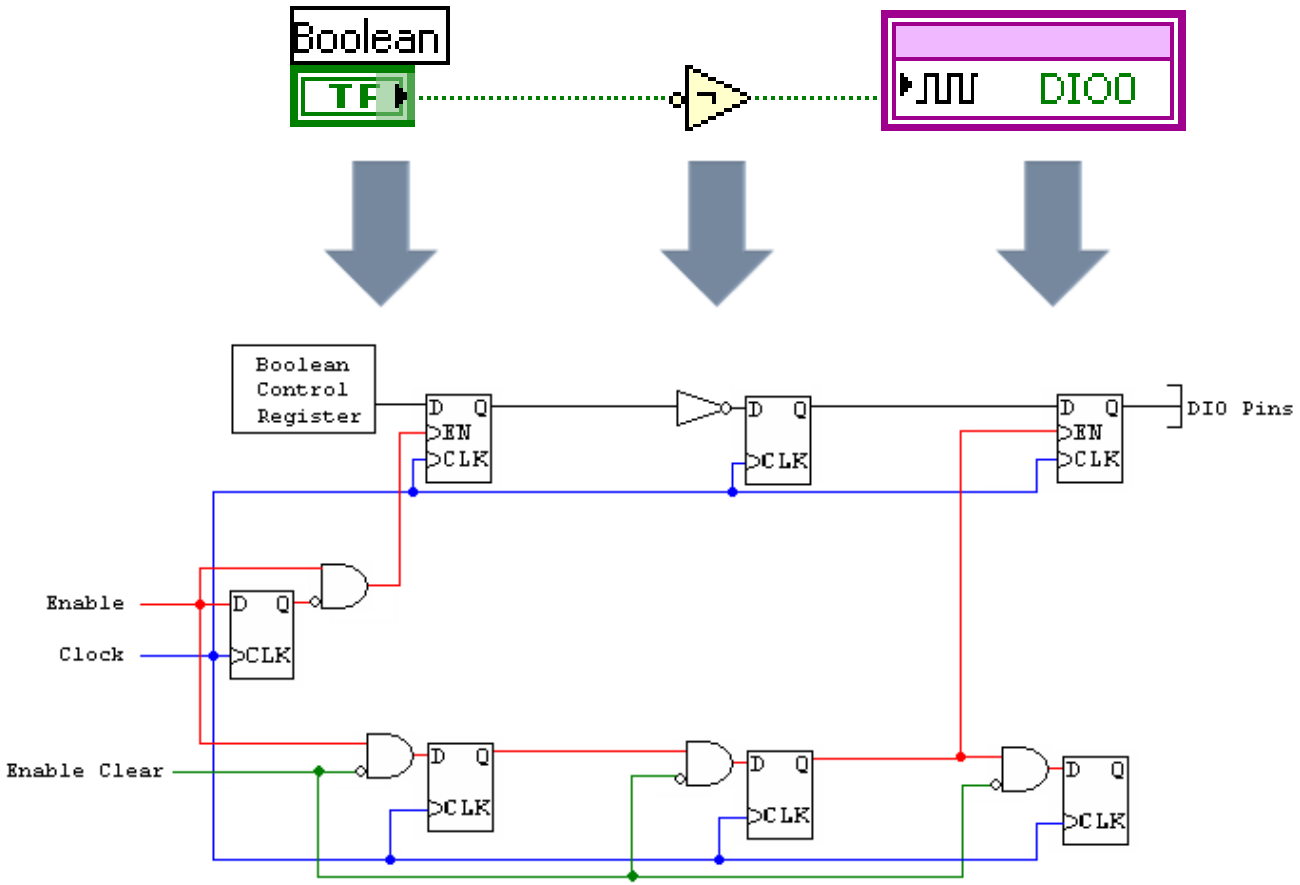
- Some techniques sacrifice speed for size and vice versa
- Can use multiple techniques in one VI
- Benchmarking techniques available
- CompactRIO training course

FPGA Optimization Technique	Speed	Size
Limit Font Panel Objects		X
Use Small Data Types		X
Avoid Large VIs	X	X
Use Non-reentrant subVIs		X
Use Reentrant subVIs	X	
Use Parallel Operations	X	
Use Pipelining	X	
Use Single-Cycle Timed Loops	X	X
Use Appropriate Arbitration		X

Dataflow within the FPGA

- Three components necessary to maintain data flow
  - The corresponding logic function
  - Synchronization
  - The enable chain

Dataflow within the FPGA



Dataflow within the FPGA

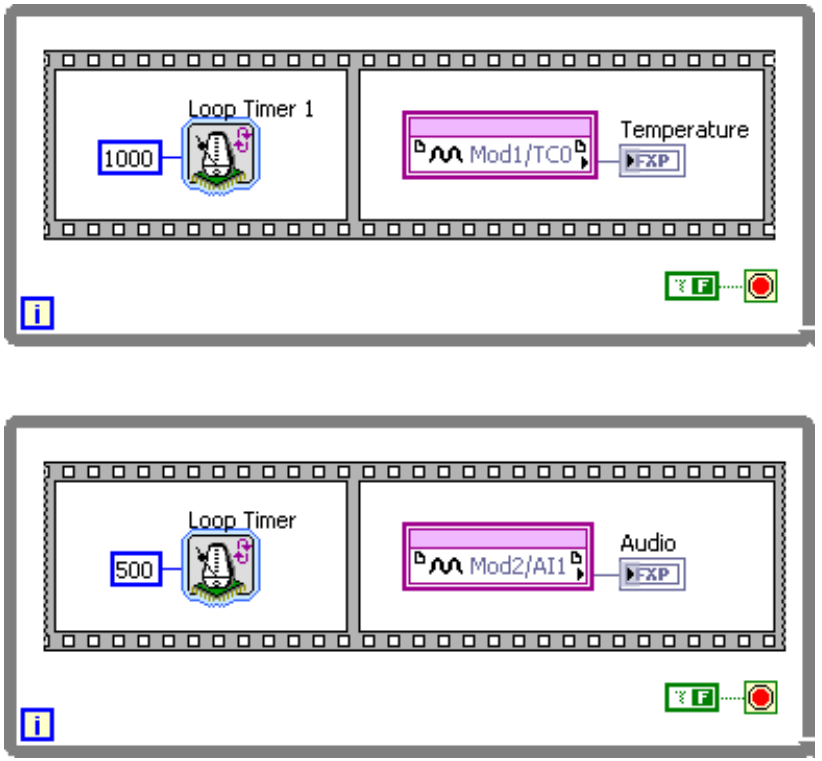
- Each function or VI takes a minimum of 1 clock tick
- Functions can run in parallel
- Some dependent functions must run in sequence
- Application can only run as quickly as the sum of items in a sequence.
- While Loops have a 2 clock tick overhead
  - If prior slide was in a loop
    - Required 3 clock ticks, plus 2 clock ticks for loop
    - Max Rate = 40 MHz / 5 = 8 MHz

Parallel operations

- Graphical programming promotes parallel code architectures
- LabVIEW Windows and Real-Time **serialize** execution
- LabVIEW FPGA implements **truly parallel** execution

Parallel Operations

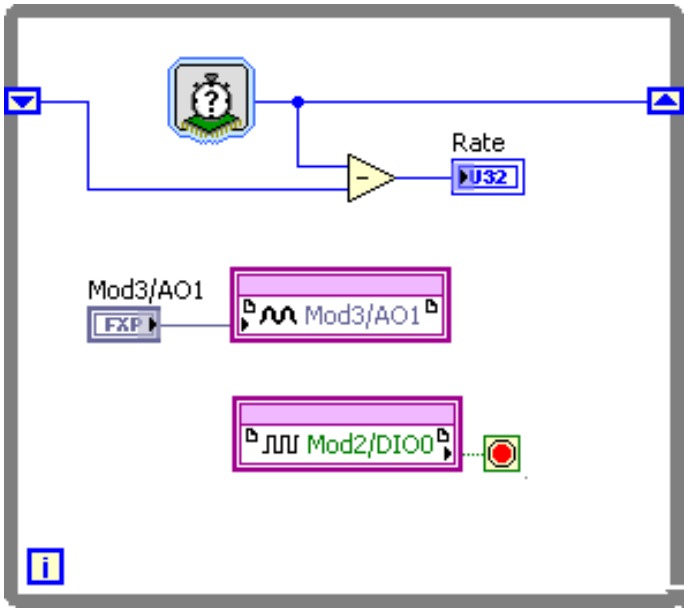
- Two parallel loops with different sampling rates
  - Run in parallel because there are no shared resources between the two loops.



Parallel Operations

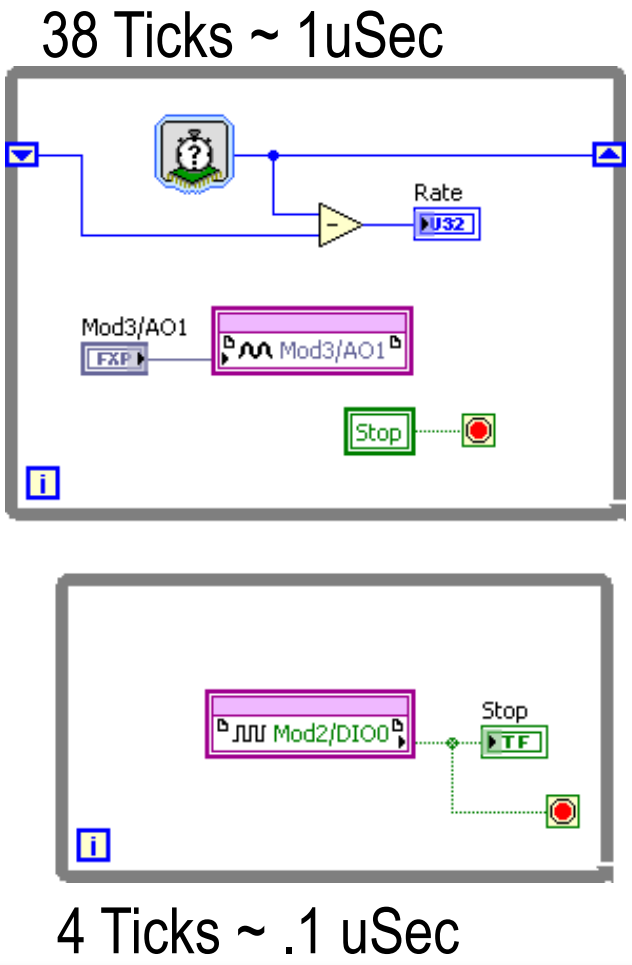
- Loop rates limited by longest path
  - A0 takes about 35 ticks, DI takes 1 tick (HW Specific)
  - DI limited by AO when in same loop

38 Ticks ~ 1uSec



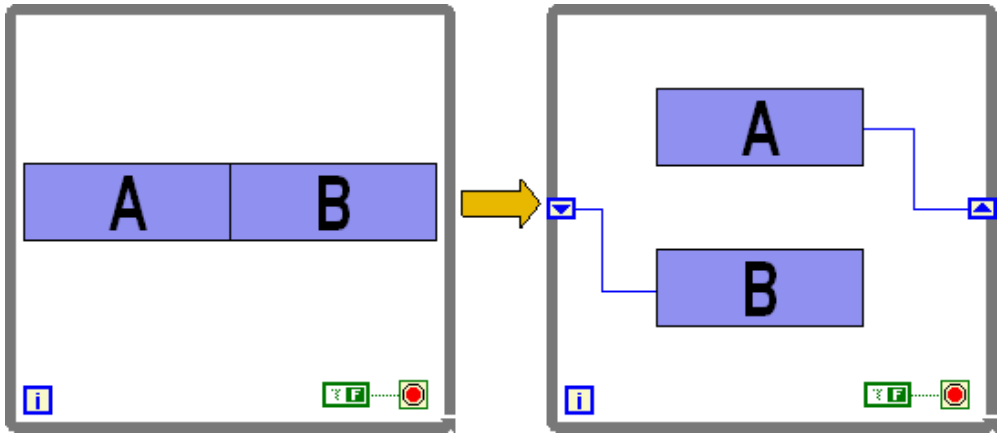
Parallel Operations

- Loop rates limited by longest path
  - A0 takes about 35 ticks, DI takes 1 tick (HW Specific)
  - Separate functions to allow DI to run independent of AO
  - This allows DI to be sampled 10 times faster by using a separate loop

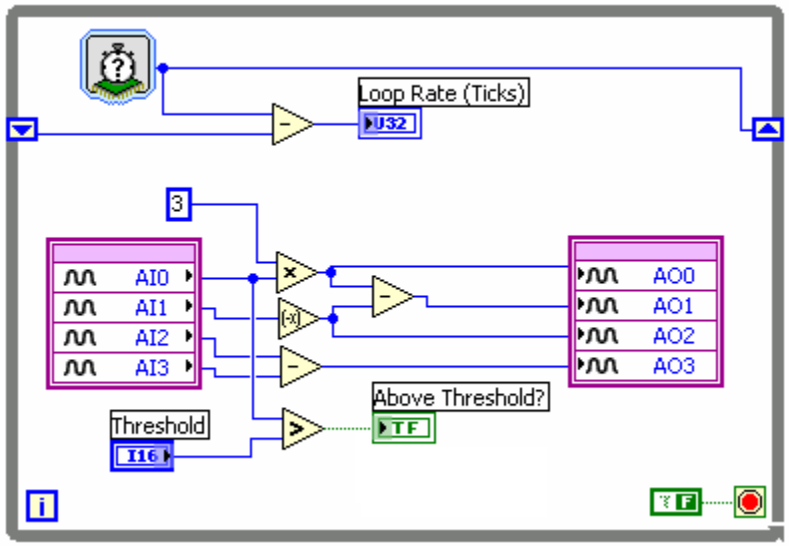


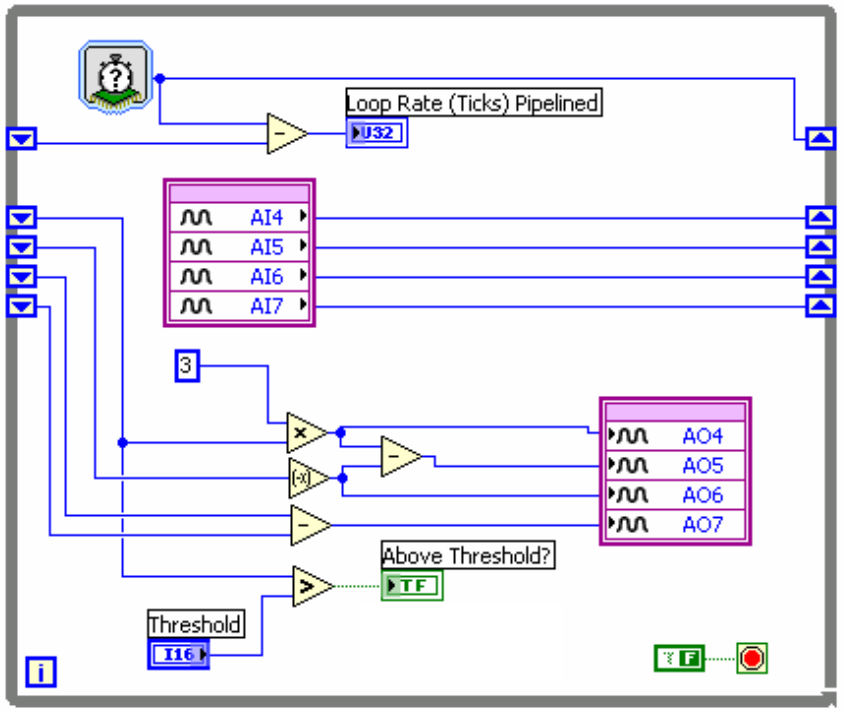
Pipelining

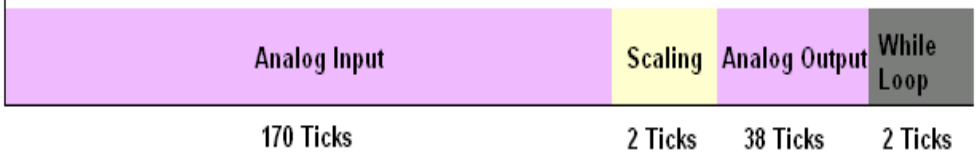
- Split up your code into different loop iterations to reduce the length of each iteration
  - Handle different parts of the process flow in parallel within one loop iteration
  - Pass data to the next using shift registers
  - Critical path is reduced



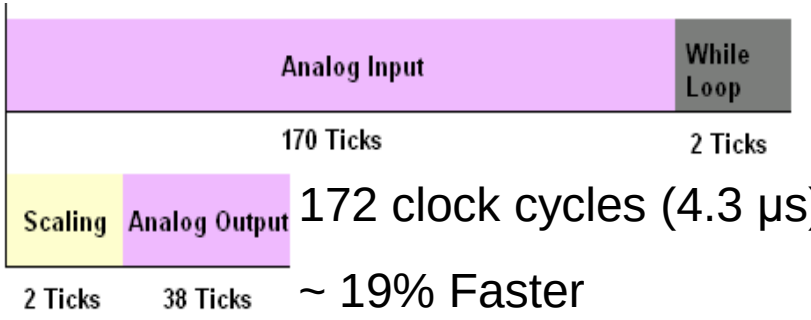
Pipelining example







212 clock cycles (5.3 µs)



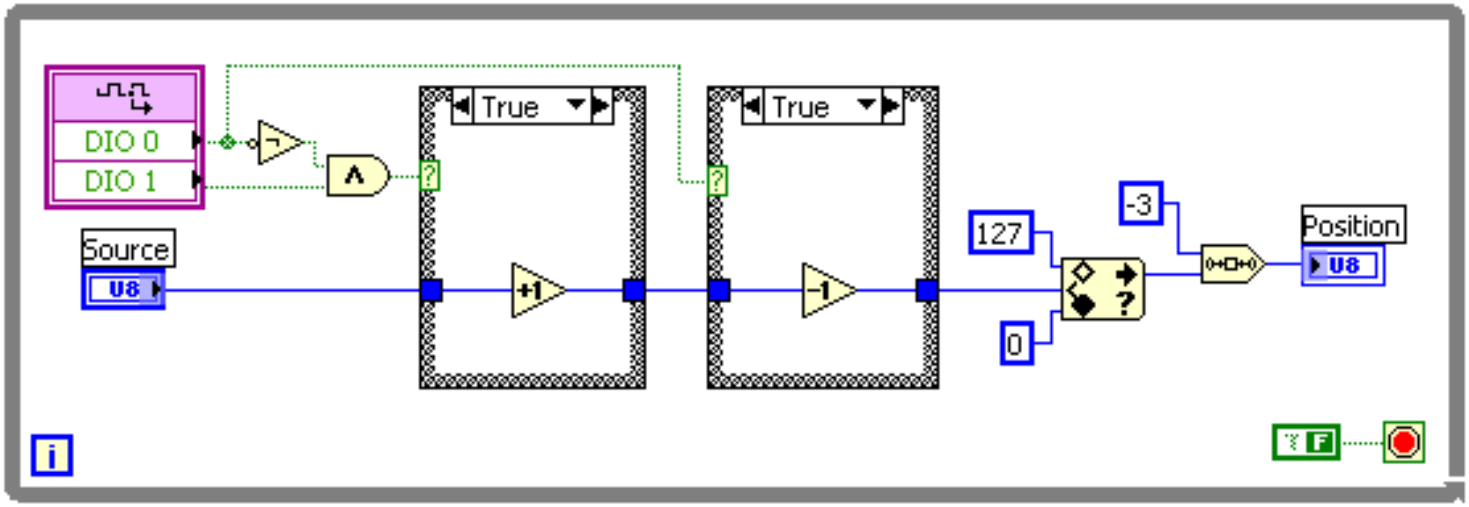
172 clock cycles (4.3 µs)

~ 19% Faster

But can we go faster?

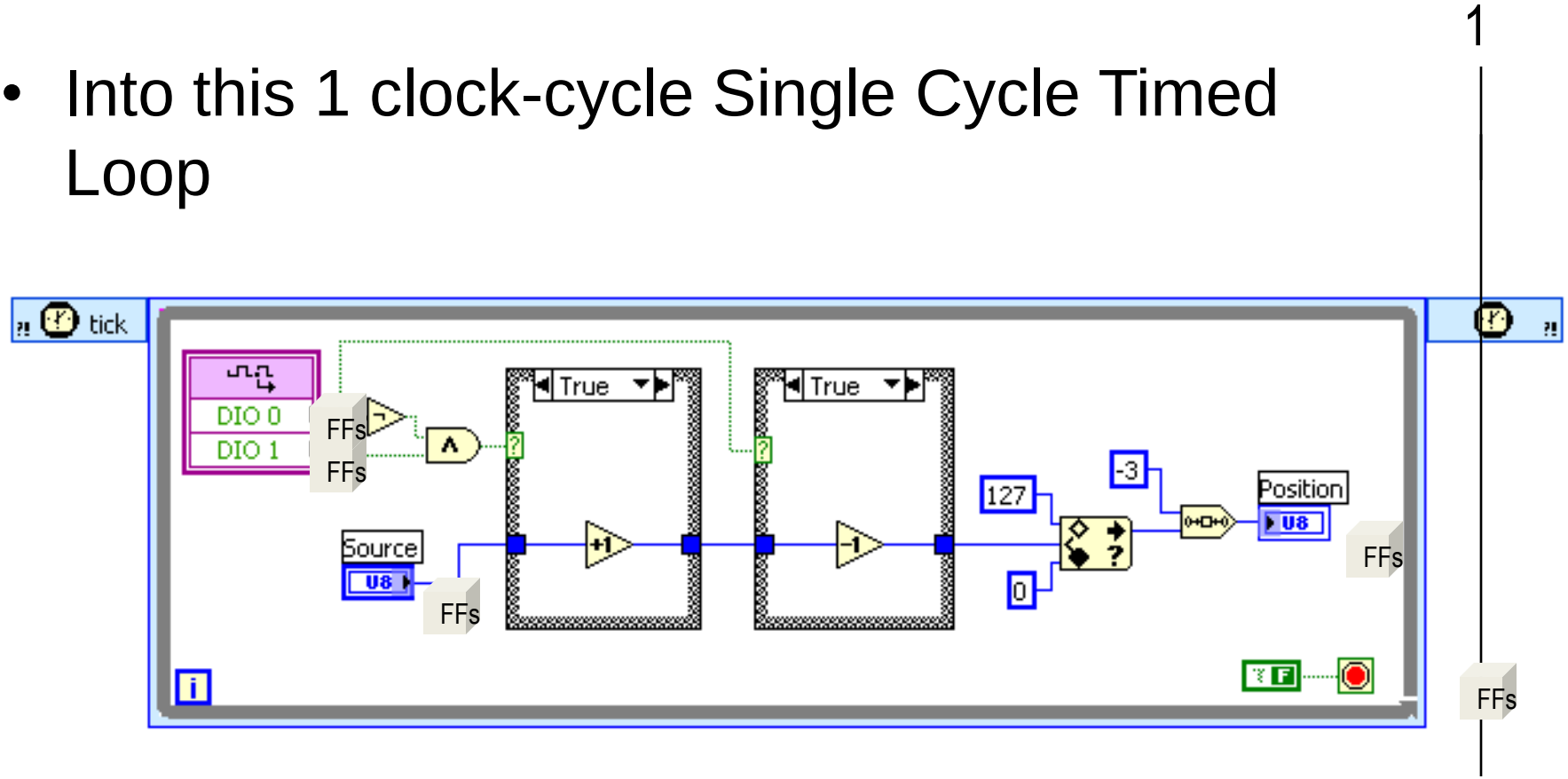
Single Cycle Timed Loop

- Can use a Single-Cycle Timed Loop (SCTL) to turn this 12 clock-cycle While Loop…



Single Cycle Timed Loop

- Into this 1 clock-cycle Single Cycle Timed Loop



Single Cycle Timed Loop (SCTL)

- All operations within the SCTL loop must fit into one clock cycle
  - Code generator error message
  - Use pipelining to separate code into smaller pieces something that minimizes the critical path
- Minimizes synchronization and enable chain overhead
- Use shift register counters to implement For loop functionality within a SCTL

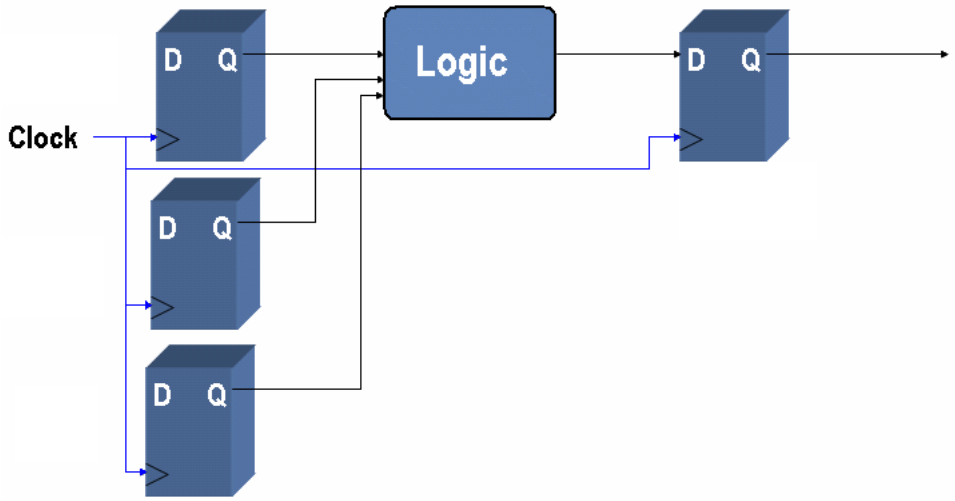
Single Cycle Timed Loop (SCTL)

- Functions not supported in SCTL:
  - Long sequences of serial code
    - Run at a lower rate using derived clocks or make the code more parallel
  - Loop timer, wait functions
  - Analog input, analog output I/O nodes
    - Place in a separate while loop and use local variables to send data
  - Loop structures (SCTL, While Loop, For Loop)
- LabVIEW Help informs which functions are supported in SCTL

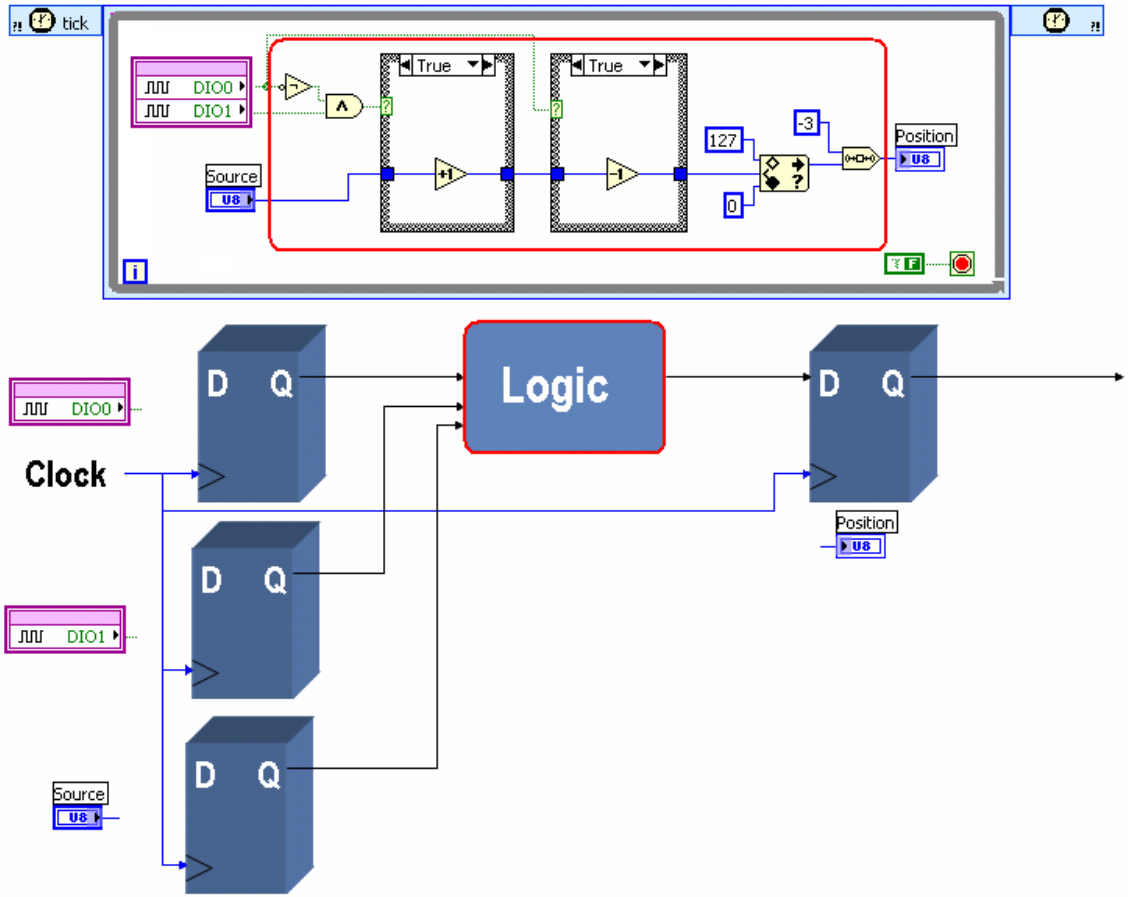
But what can be done to reduce the combinatorial path in a SCTL if it is too long?

Combining Optimizations

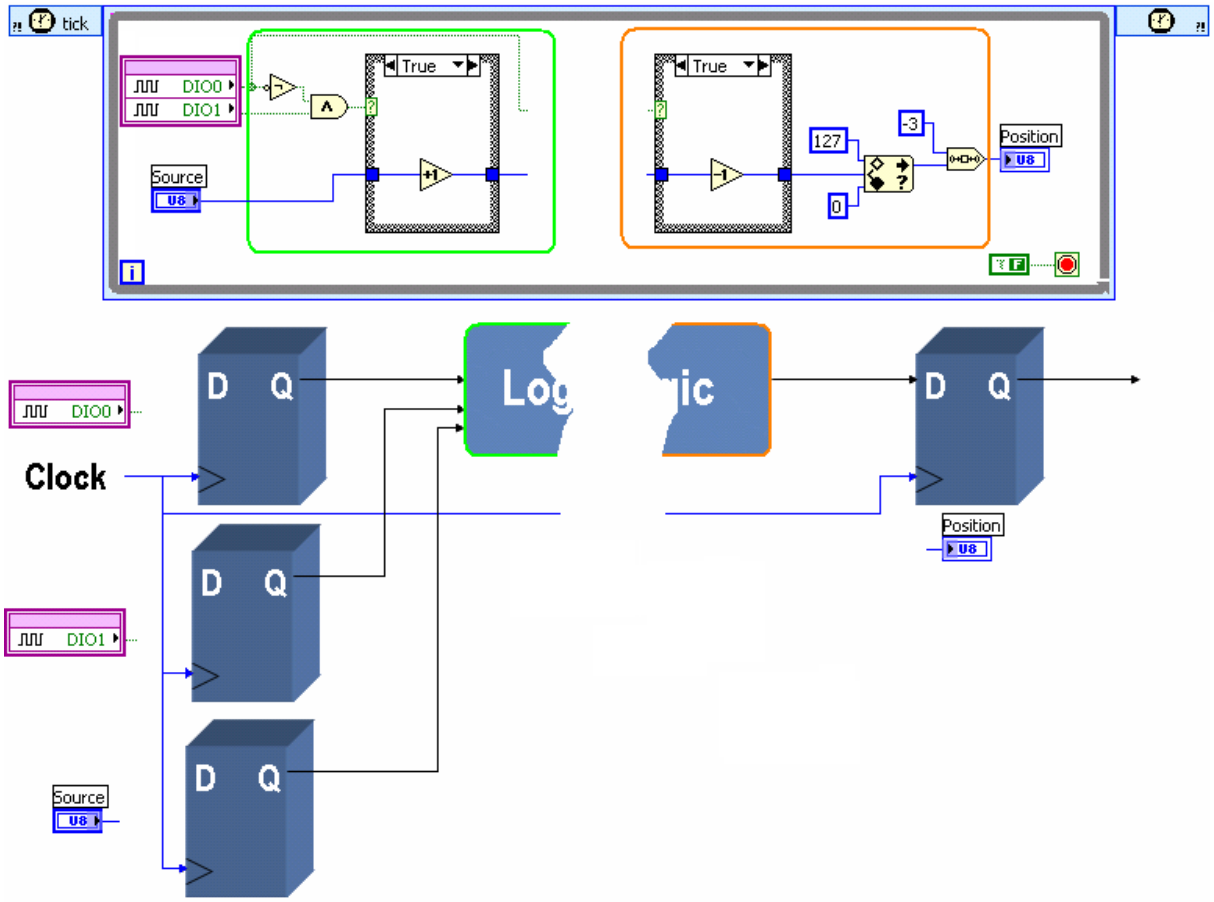
- Combinatorial Path
  - Determined by the longest path between flip-flops.
  - A long combinatorial path can lead to the compile to fail for not meeting timing constraints.
  - Use pipelining to break up combinatorial paths



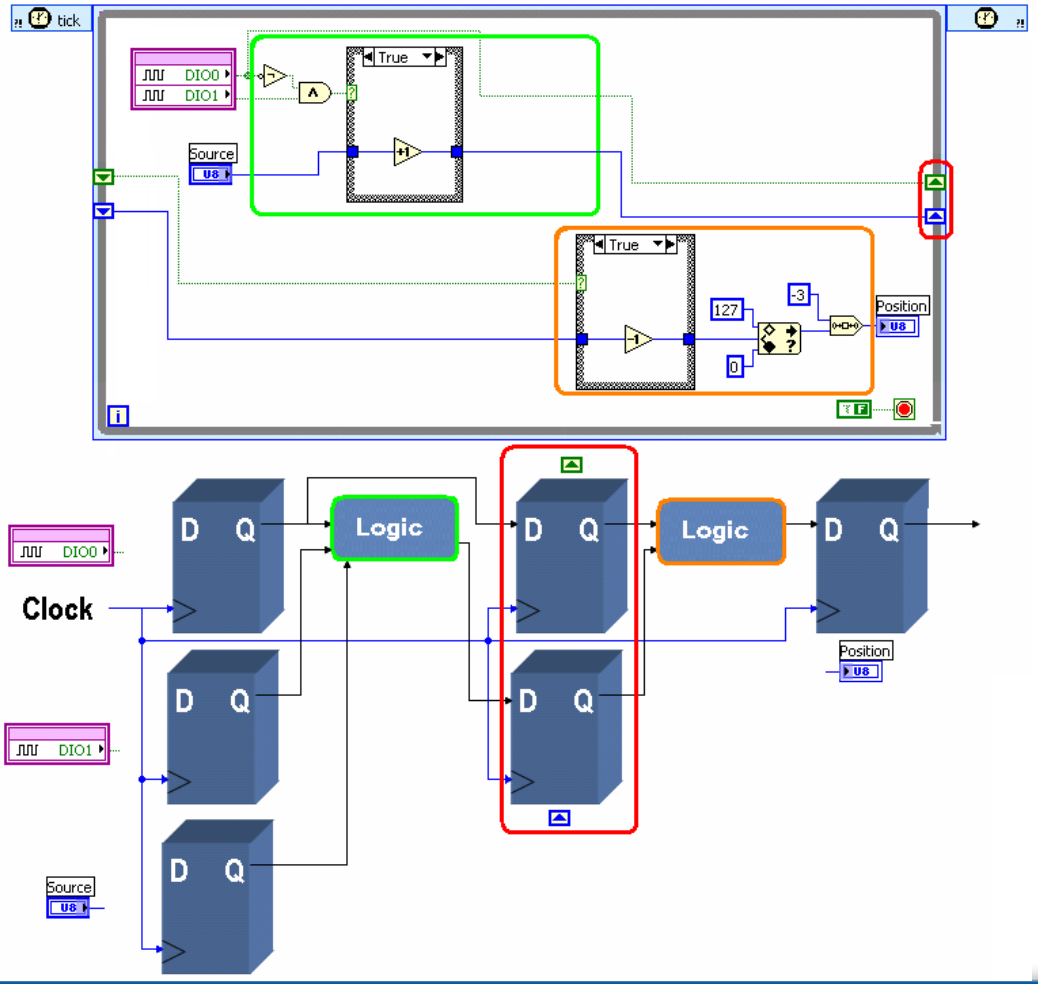
Combining Optimizations

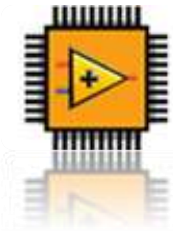


Combining Optimizations



Combining Optimizations



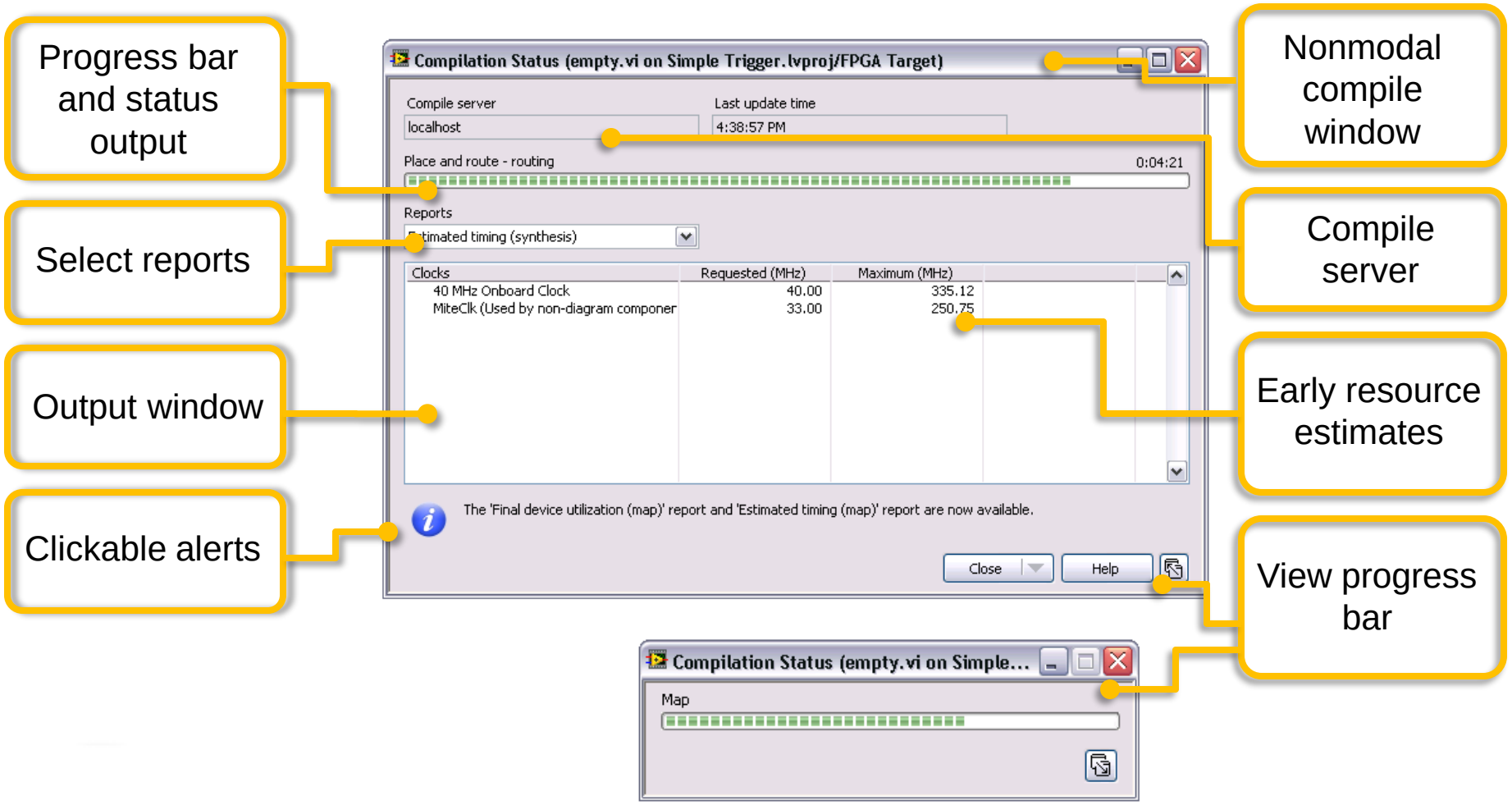


Part 2 - New features in LabVIEW FPGA 2009

Demonstration

Timing Violation Analysis

New Compilation Window

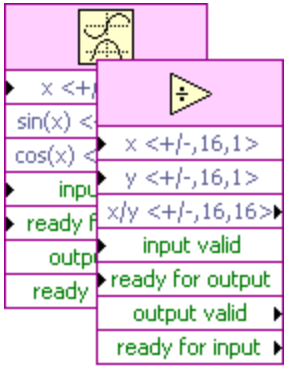


Compilation Improvement Summary

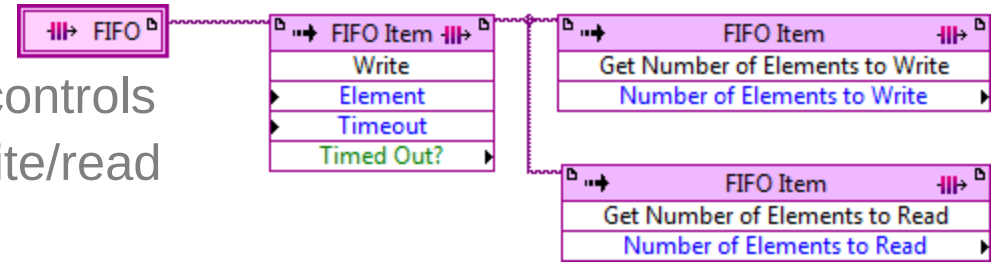
- **Early size and speed estimate –** HDL synthesis results reporting
- **Critical-path highlighting –** Debugging when design does not meet timing
- **Precompile options –** Define effort levels for size and speed
- **Clickable alerts –** When new reports are available
- **Better compiler state output –** Progress bar and state
- **Continue working without disconnect –** Window is nonmodal
- **Reduced memory –** Somewhat faster compile and larger FPGA support
- **New bit file format –** XML-based with some human-readable information

New and Improved IP

- New
  - High-Throughput Math
    - Arithmetic
    - Natural logarithm
    - Trigonometric functions



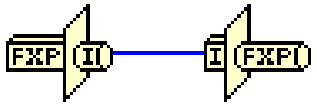
  - FIFO/Memory Profiling
    - FIFO and memory name controls
    - Number of elements to write/read

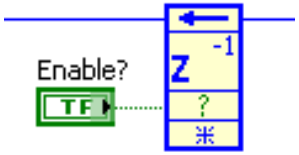


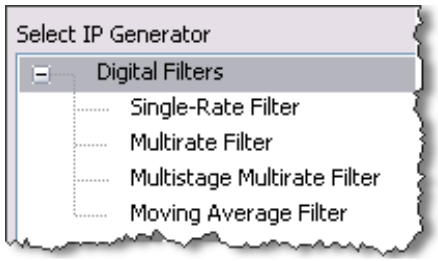
  - New IP on IPNet (ni.com/ipnet)
    - RF, digital filtering, and digital protocols

New and Improved IP

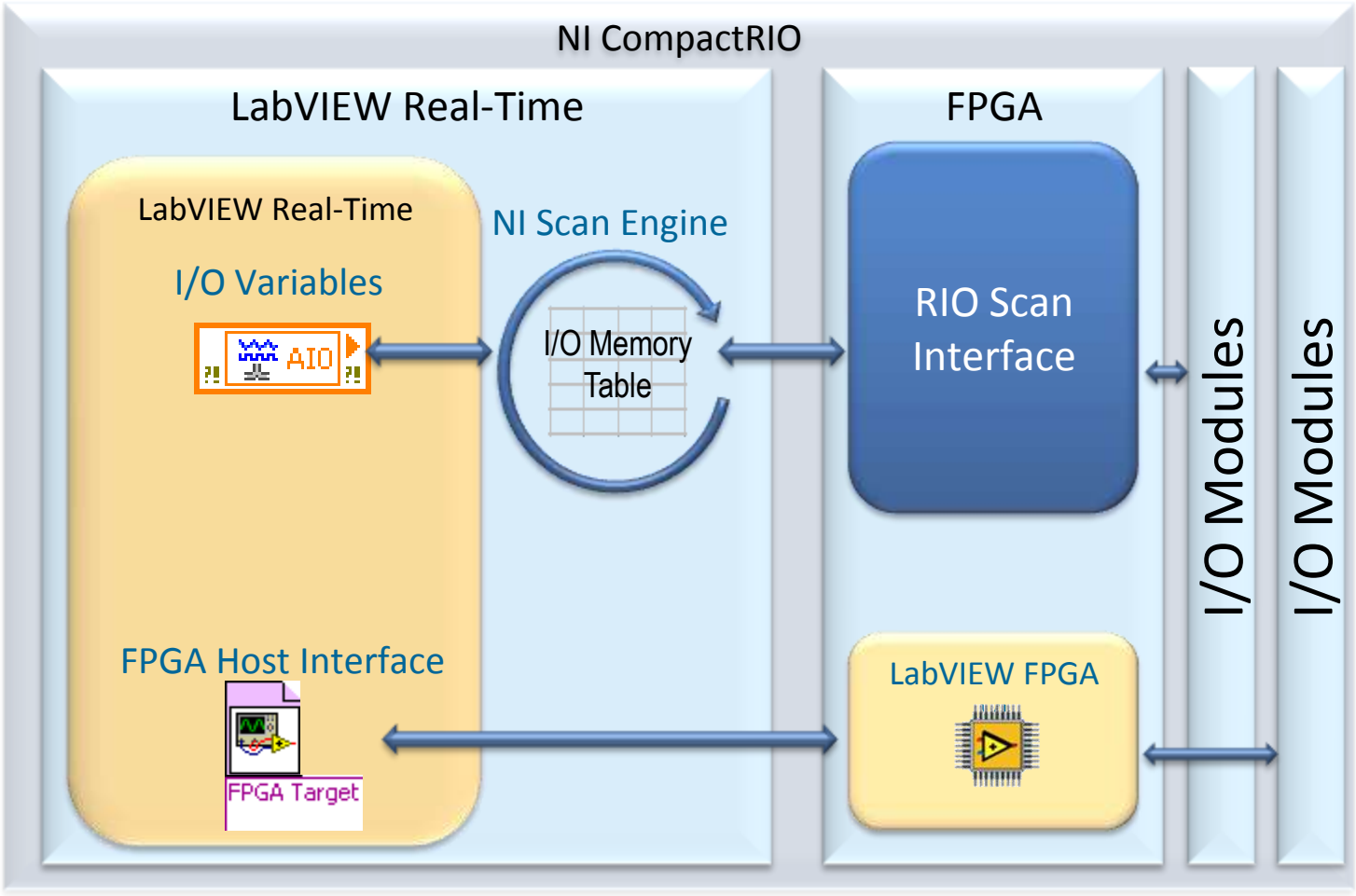
- Improved
  - FFT, Window, Resample
    - Support 100 MHz clock rates
  - Fixed-Point Cast
    - Bit-wise cast
  - Feedback Node
    - Multiple pipelines in one node
    - Enable input
  - Digital Filter Design Toolkit
    - FXP data type support
    - Integrated into IP generator





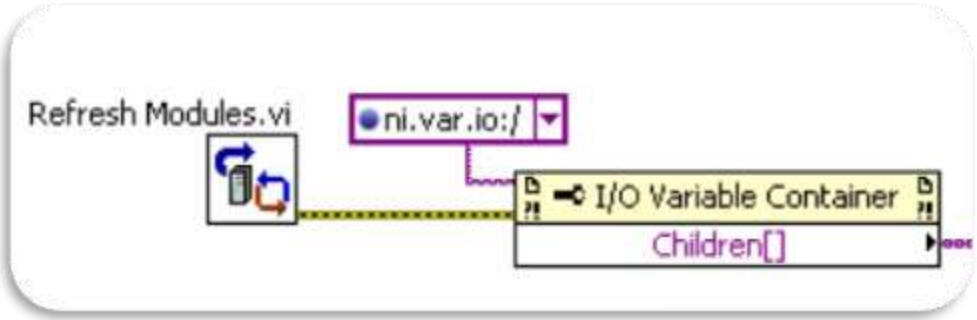


NI CompactRIO Scan Mode

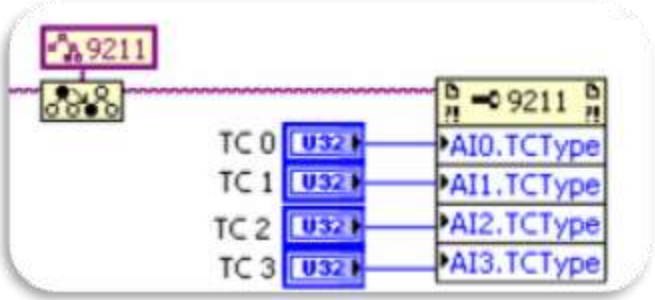


Programmatic I/O Discovery and Configuration

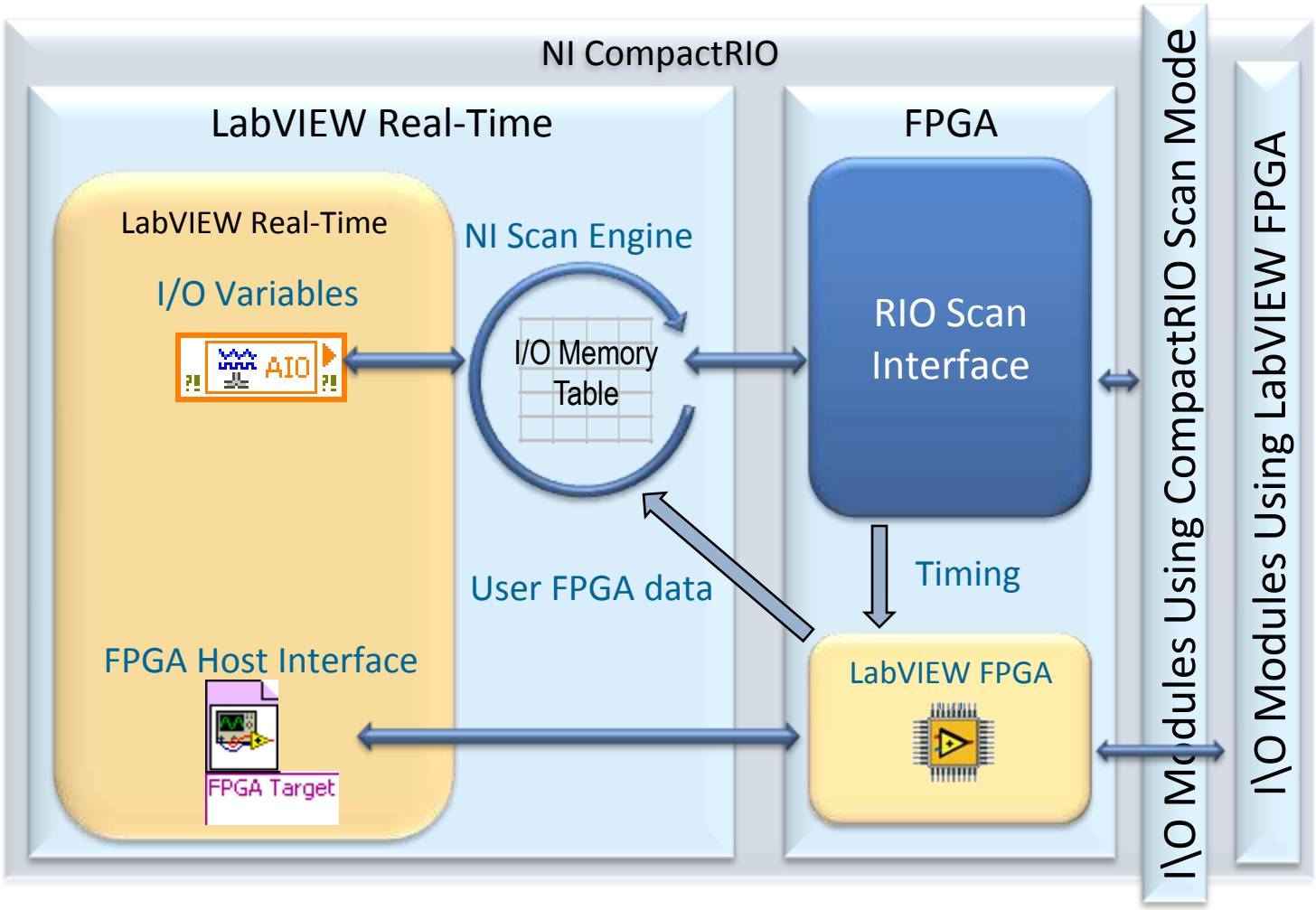
- Discover and configure I/O at run time
- Set voltage gains, thermocouple types, and specialty digital properties



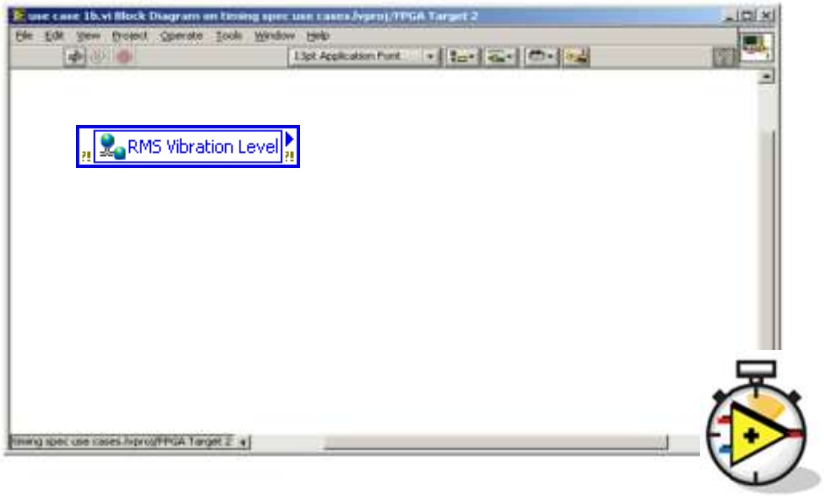
DEMO

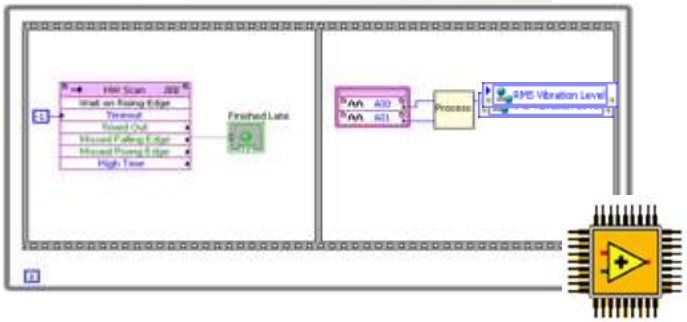


Adding FPGA Data to the NI Scan Engine



CompactRIO Expansion Chassis – Open FPGA





New FPGA Hardware

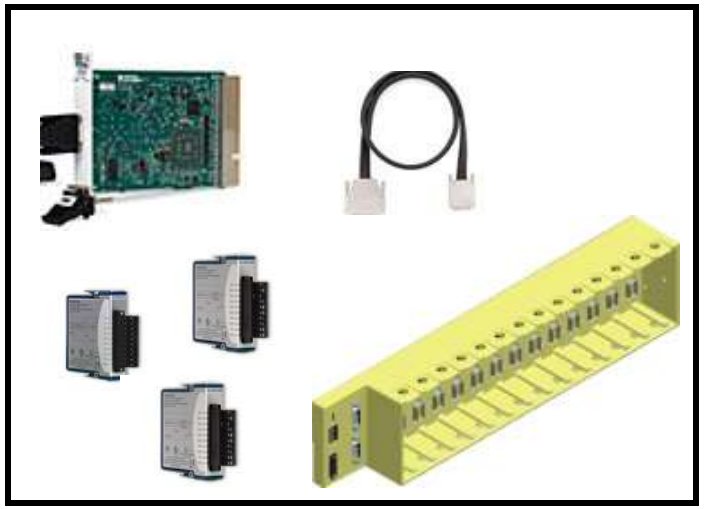
Deterministic distributed I/O with a programmable FPGA

Virtex-5 CompactRIO Backplanes

New PCI Express R Series Multifunction RIO Boards

MXI-Express RIO

- 14 C-Series Slots
- Virtex 5 FPGA
- X1 MXI-Express interface based on cabled PCI-Express (250 Mb/s)
- Daisy chainable up to 4 chassis



CompactRIO and Machine Vision

IP Camera

Basler Vision Technologies

Axis Communications

Analog Camera

With AF-1501 module from moviMED

(also comes in single-board version)

New FPGA Hardware

NI 6581 100 MHz and **NI 6585** 200 MHz LVDS digital NI FlexRIO Adapter Modules

NexFrontier 100 MHz pin electronics
Averna IEEE 1394b module

NI PXIe-5641R RIO IF transceiver

Summary

- **Improve FPGA execution –** Use pipelining and SCTL
- **New Compilation Experience –** Early size estimates, new GUI, and more
- **Timing Violation Debugging –** Critical-path highlighting
- **New and Improved IP –** High-throughput math and signal processing
- **Inline FPGA processing –** FPGA programmable IOVs
- **New Hardware –** FPGA-enable distributed I/O, CompactRIO backplanes, NI FlexRIO Adapter Modules, NI PXIe-5641R RIO IF transceiver, MXI-Express RIO