The Right Development Process for LabVIEW FPGA Success

Best Practices for Development

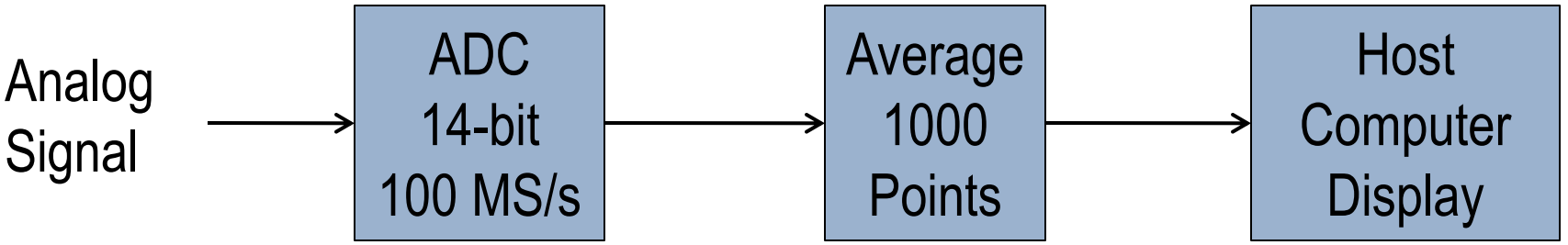
Before we start

- Do you know what FPGAs are?
- Have you programmed with LabVIEW FPGA?
- Hopefully you are eager to learn more about the most efficient way to implement your application with LabVIEW FPGA

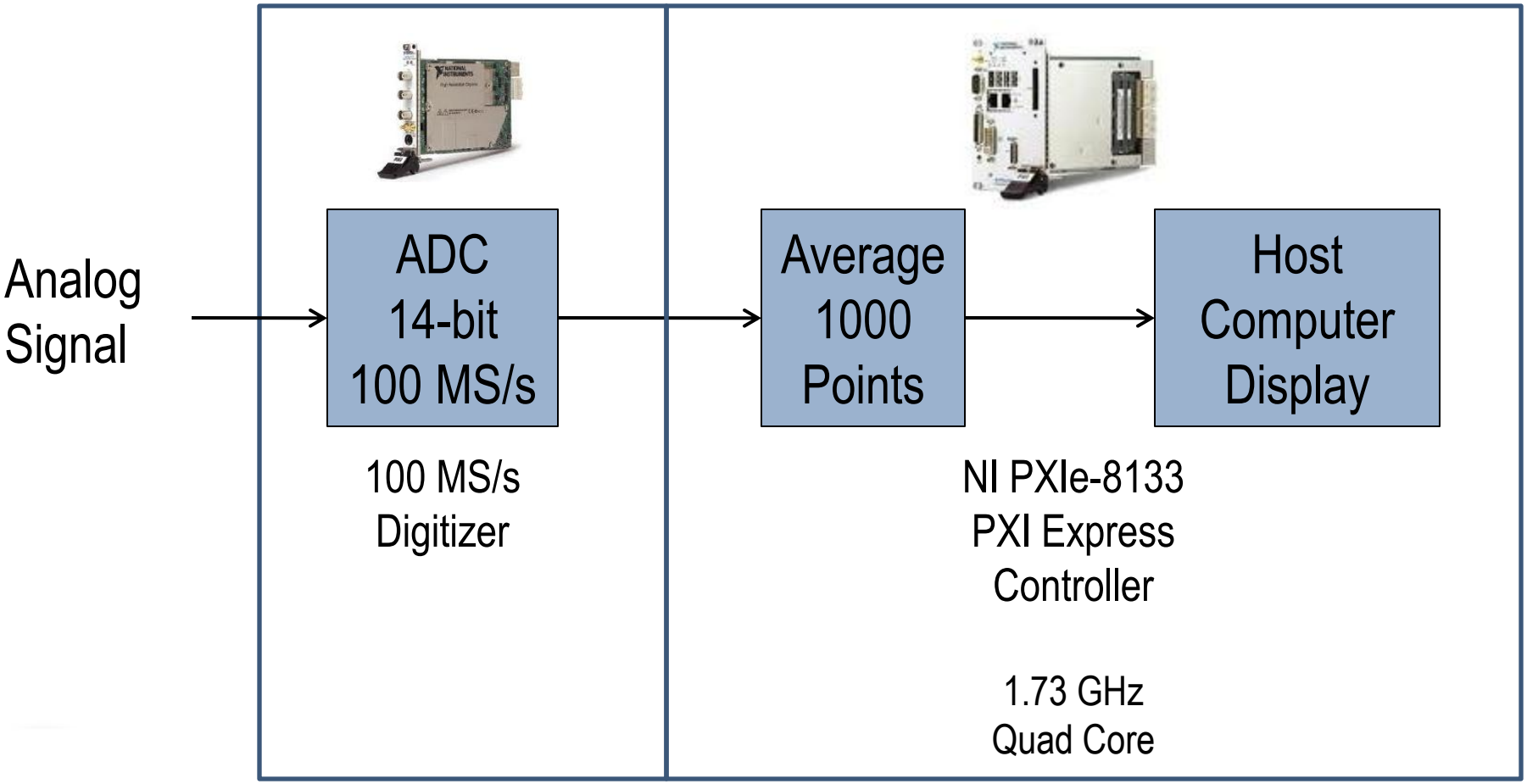
Agenda

- Design Problem
- Design Process

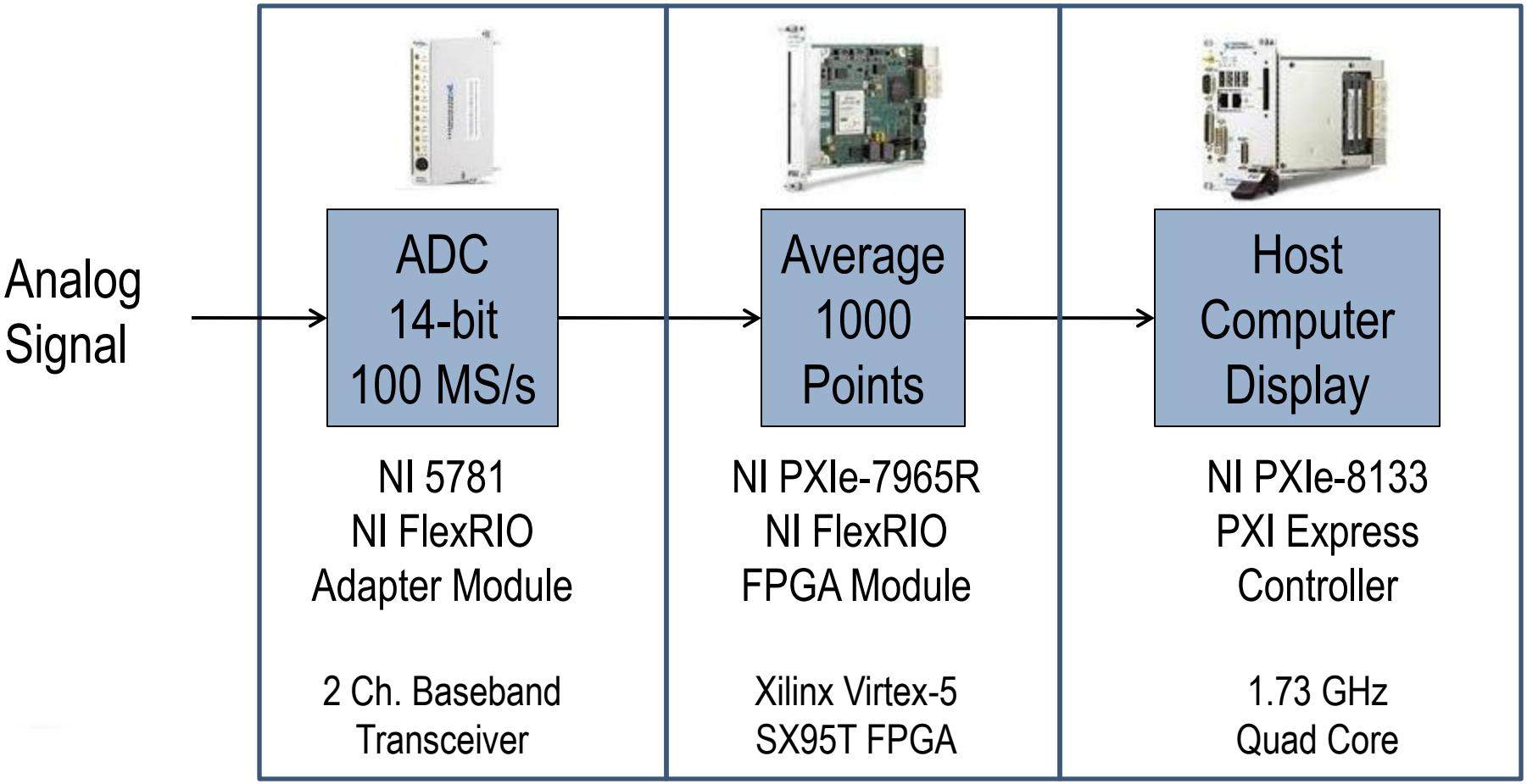
Design Problem



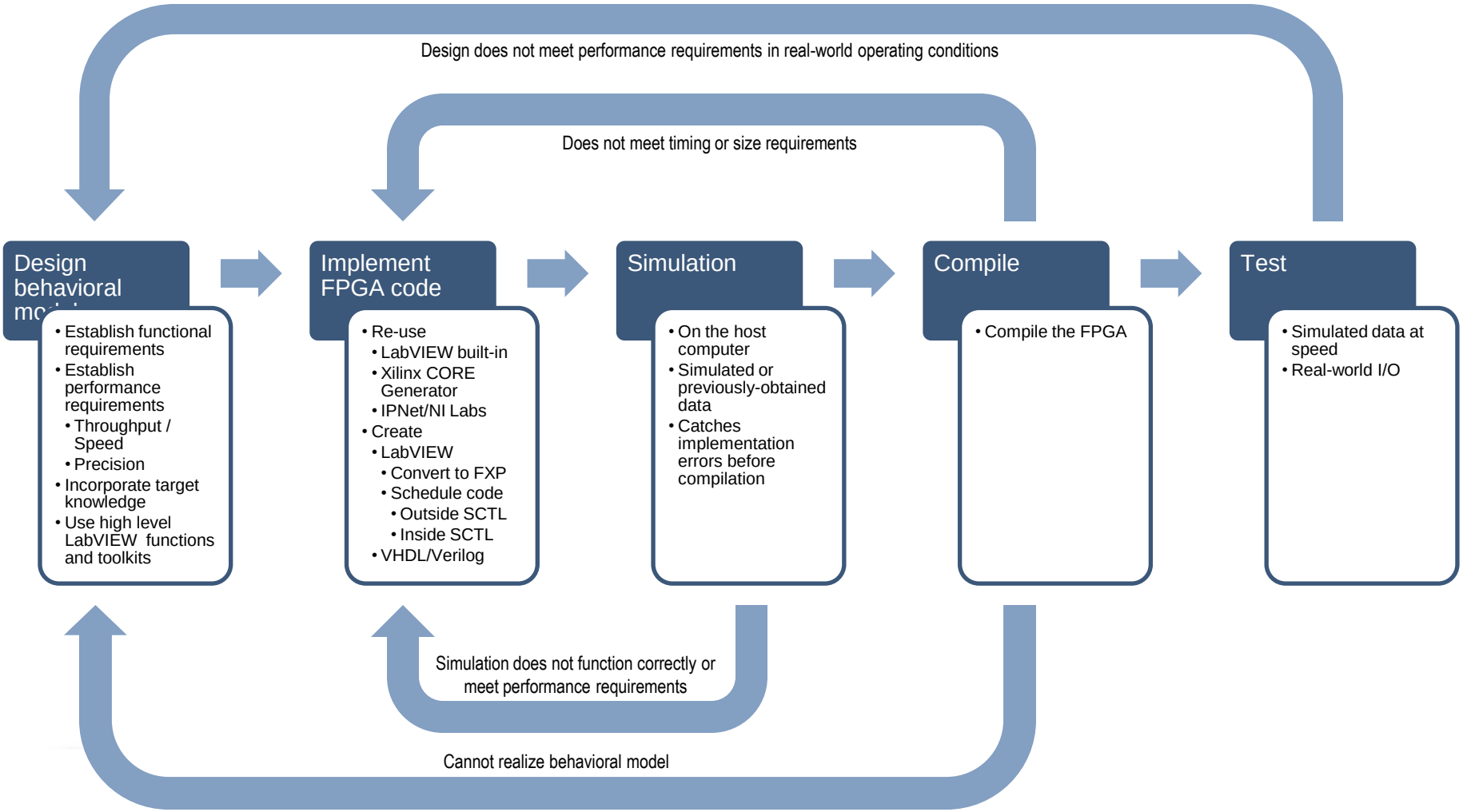
Design Problem



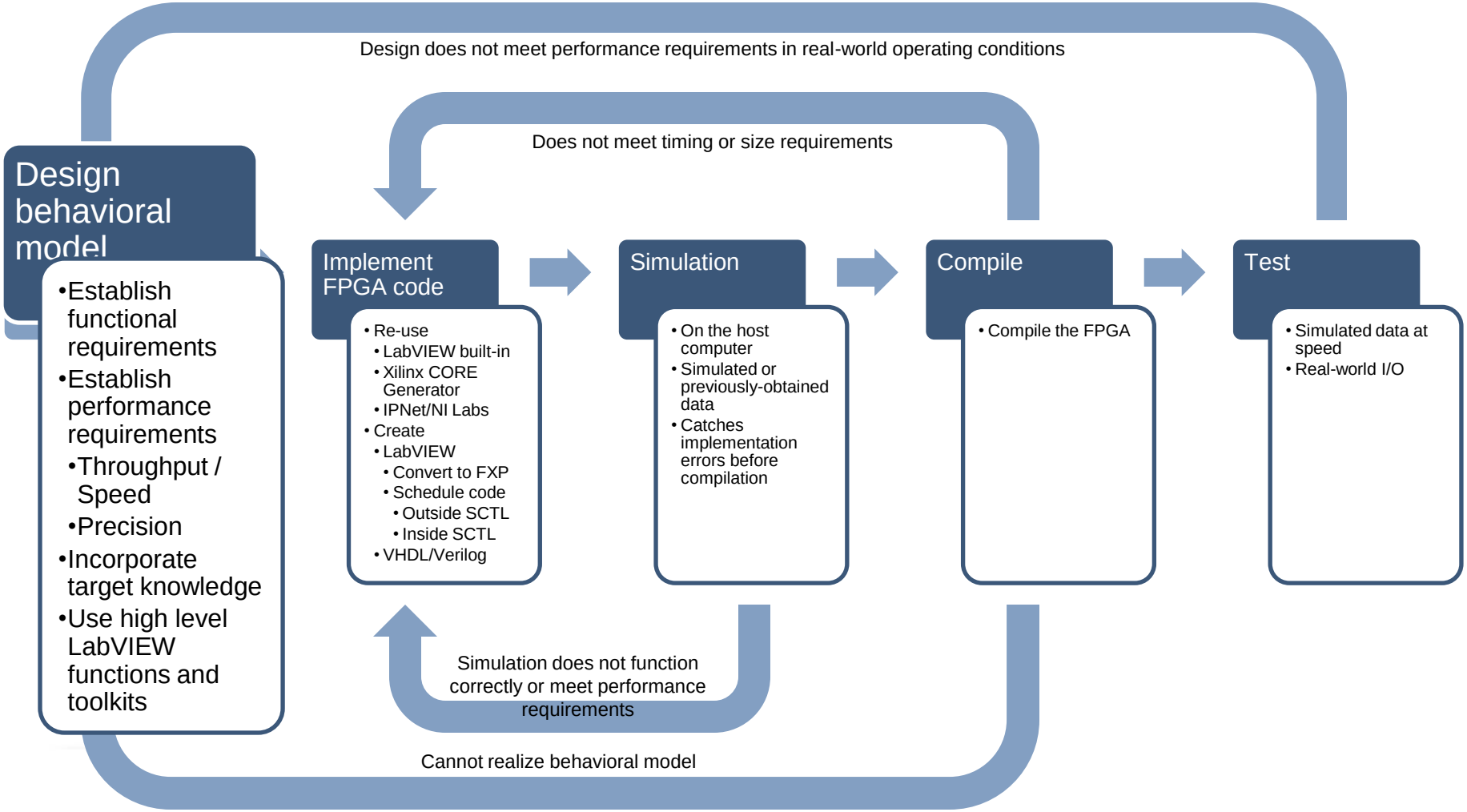
Design Problem



LabVIEW FPGA Design Flow



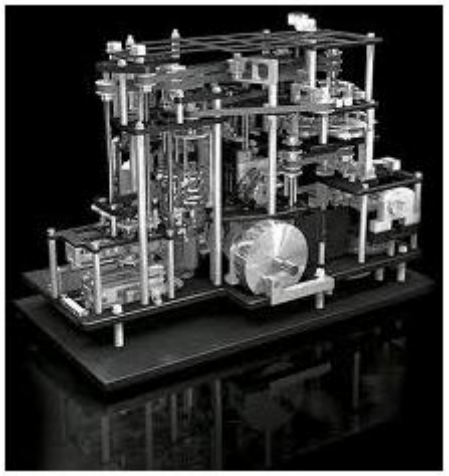
LabVIEW FPGA Design Flow



Establish Requirements

- Functional Requirements
  - Digitize analog signal
  - Average every 1000 samples
  - Display averaged data on the host computer
- Performance Requirements
  - 100 MS/s ADC sample rate
  - 14-bit ADC resolution
  - 16-bit averager output resolution

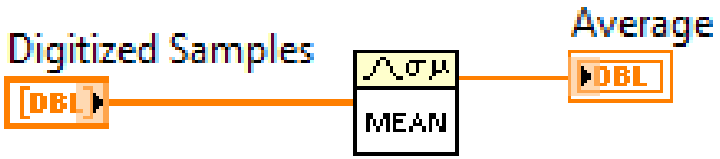
Know Your Hardware Target



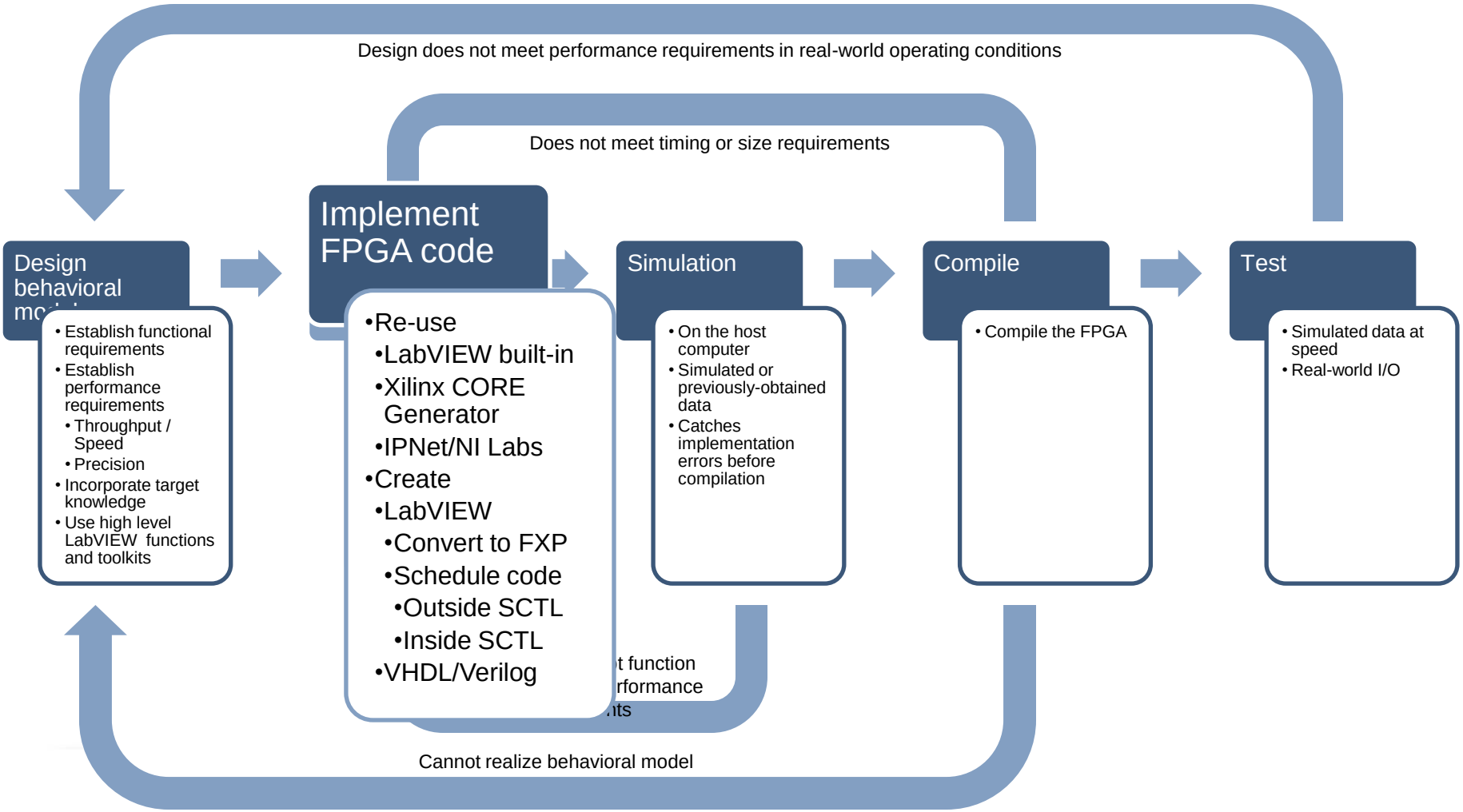
- I/O
  - Sample Rates – 100 MS/s
  - Resolution – 14-bits
- FPGA
  - Size – 58,880 FF and LUTs, 488 BRAM, 640 DSP48Es
  - Hardware components – BRAM, DSP48E
  - Speed – 100 MHz is no problem for most code
    - Do a few test compiles

Host Computer Behavior Model

- High level LabVIEW implementation of your design problem
  - Functionally accurate
  - Doesn't need to meet timing requirements
  - High level functions
  - Floating Point
  - Arrays
  - For/while loops

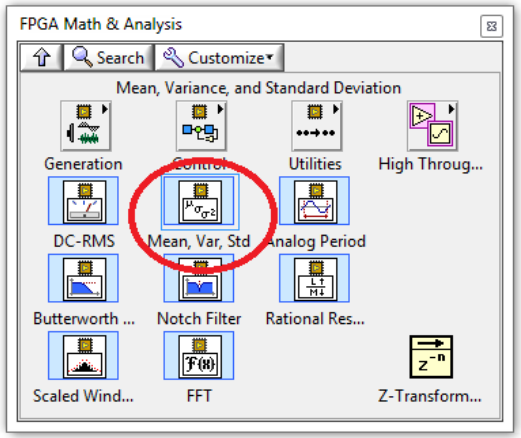


LabVIEW FPGA Design Flow

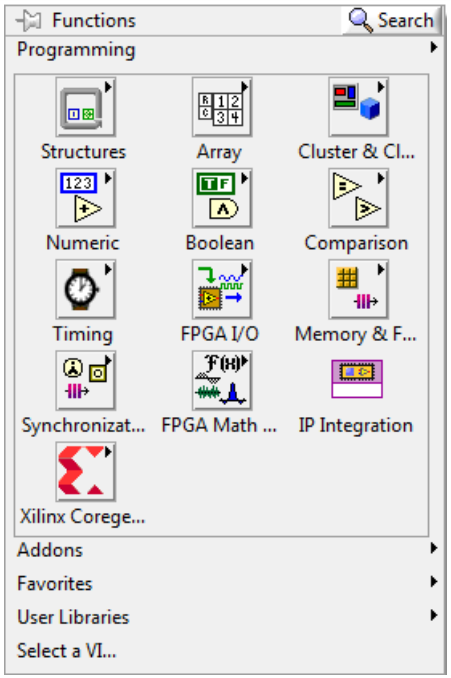


Existing LabVIEW FPGA IP

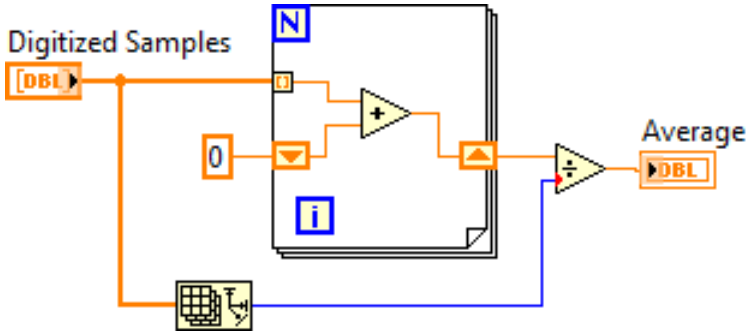
- See what LabVIEW FPGA IP is already available
- Sources
  - LabVIEW FPGA palette
  - Xilinx Coregen IP
  - http://www.ni.com/ipnet/
  - https://decibel.ni.com/content/groups/ni-labs
- Compare IP to your design constraints



Host code with Functions Available in LabVIEW FPGA



- Rewrite the code using
  - Functions available in LV FPGA
    - Host computer version
  - Floating Point
- Ensures your understanding of algorithm(s)

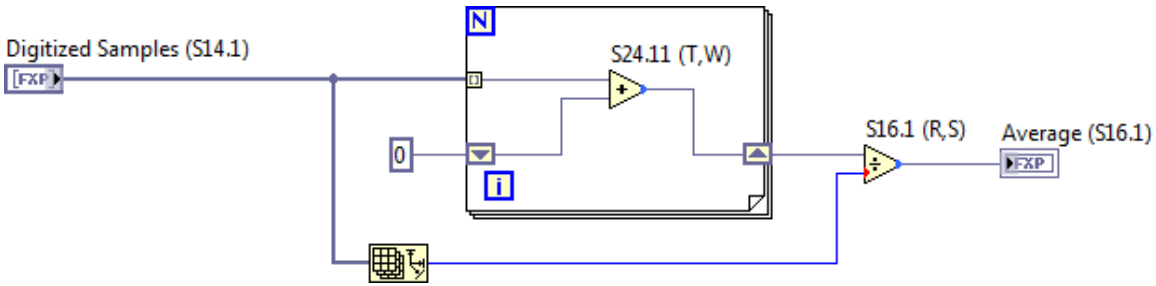


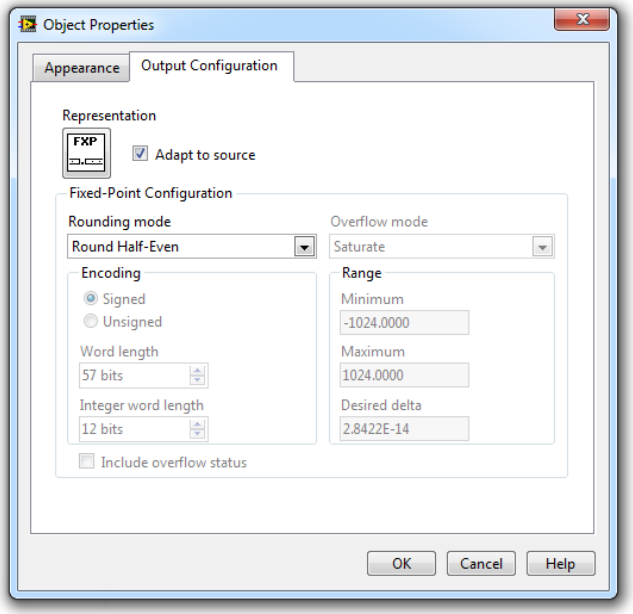
Convert to Fixed Point

- Floating point
  - Arithmetic is expensive
  - Not natively supported by LabVIEW FPGA
- Fixed point
  - Customized precision
  - Optimize the number of integer and fractional bits of each mathematical operation
  - Natively supported by LabVIEW FPGA

Convert to Fixed Point

- Use the FXP datatype
- Use “Adapt to Source” if you have the space and can meet timing

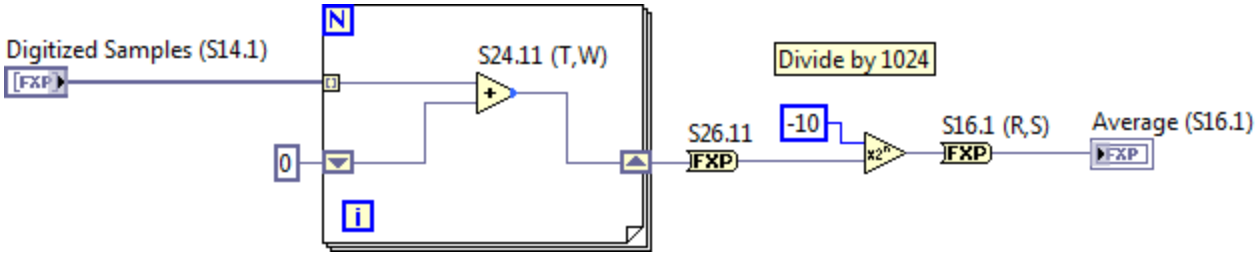




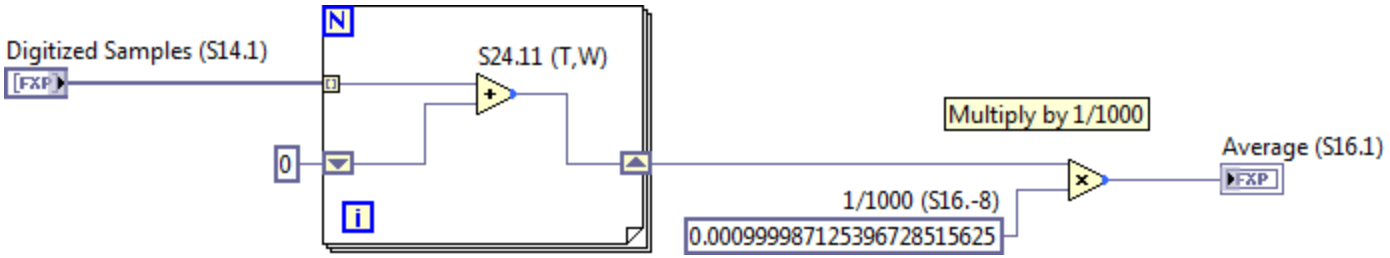
NATIONAL INSTRUMENTS

Fixed Point Optimizations

- Shift instead of multiply/divide by powers of two



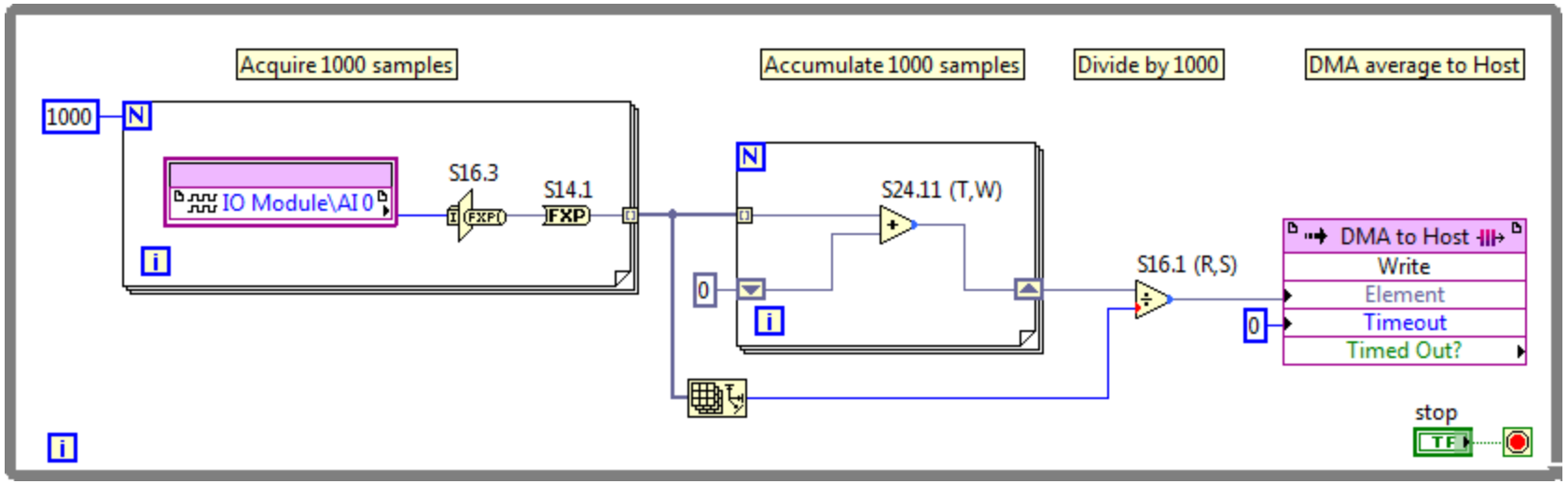
- Multiply by 1/x instead of divide by x



Scheduling

- How do you write the LabVIEW FPGA code?
  - Oustide SCTL (Single-Cycle Timed Loop)
  - Inside SCTL
- How much parallel hardware do you need?
- Driven by application timing and space requirements

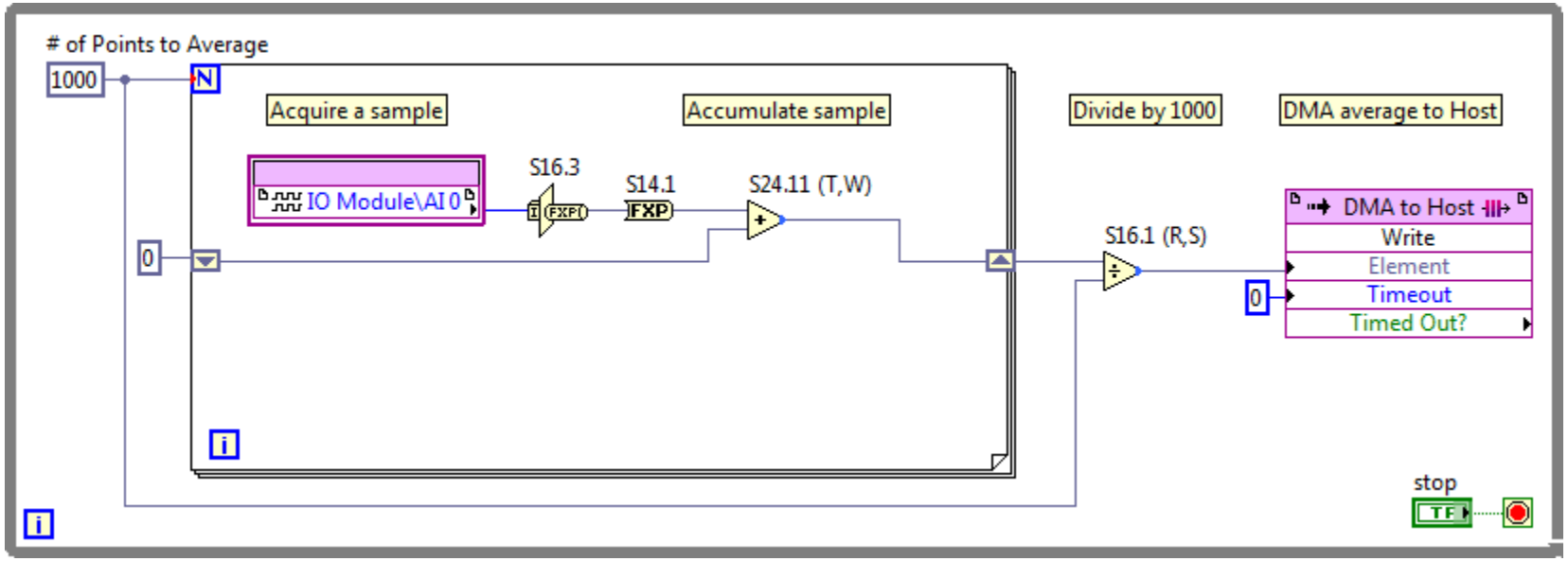
First Scheduled Solution



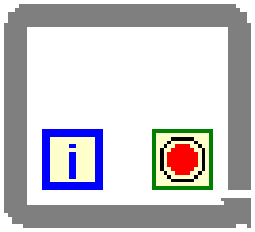
Process Data One Point at a Time

- Process the data the way it arrives
- Don’t buffer the data unless you have to
- If you have to buffer data
  - Arrays are expensive
  - Use Block RAM or Distributed RAM

Second Scheduled Solution

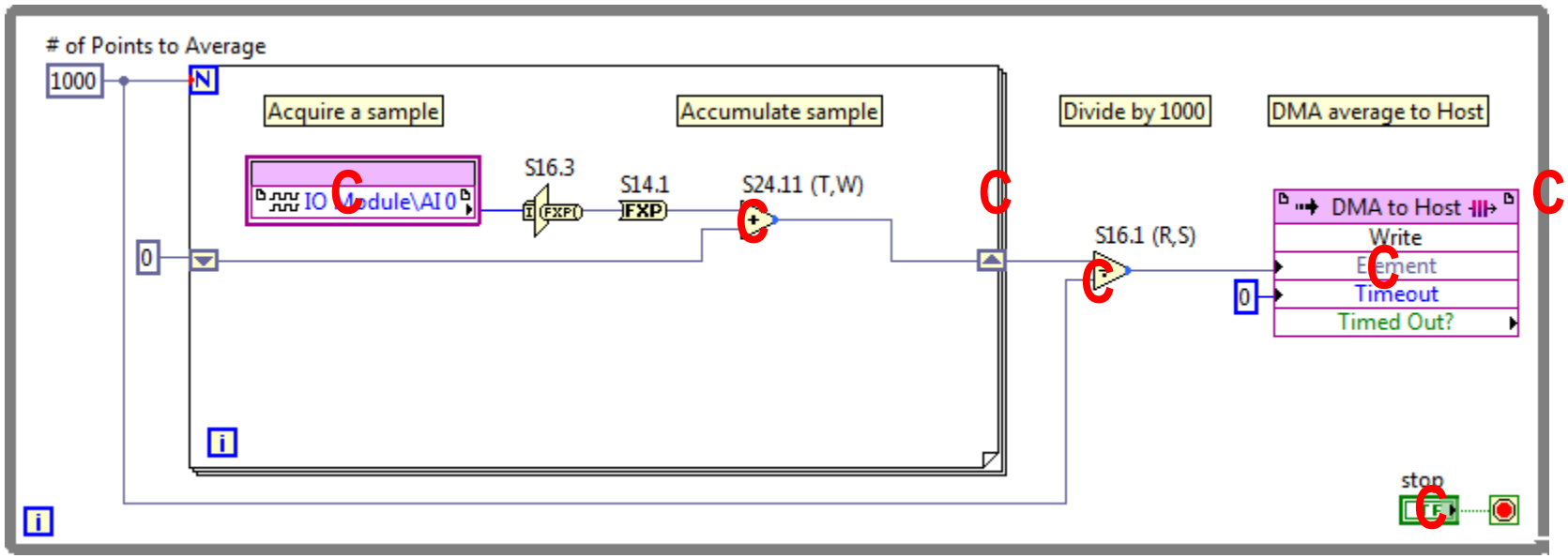


Outside SCTL



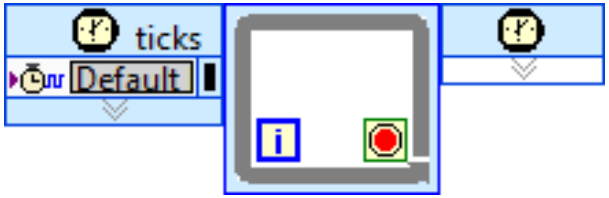
- Similar to desktop LabVIEW programming
  - For/While loops
  - More LabVIEW functions are supported (divide)
- Automatically pipelined to meet FPGA clock frequency constraints
- Application timing is abstracted away
  - Tough to determine if design meets application timing requirements

Second Scheduled Solution



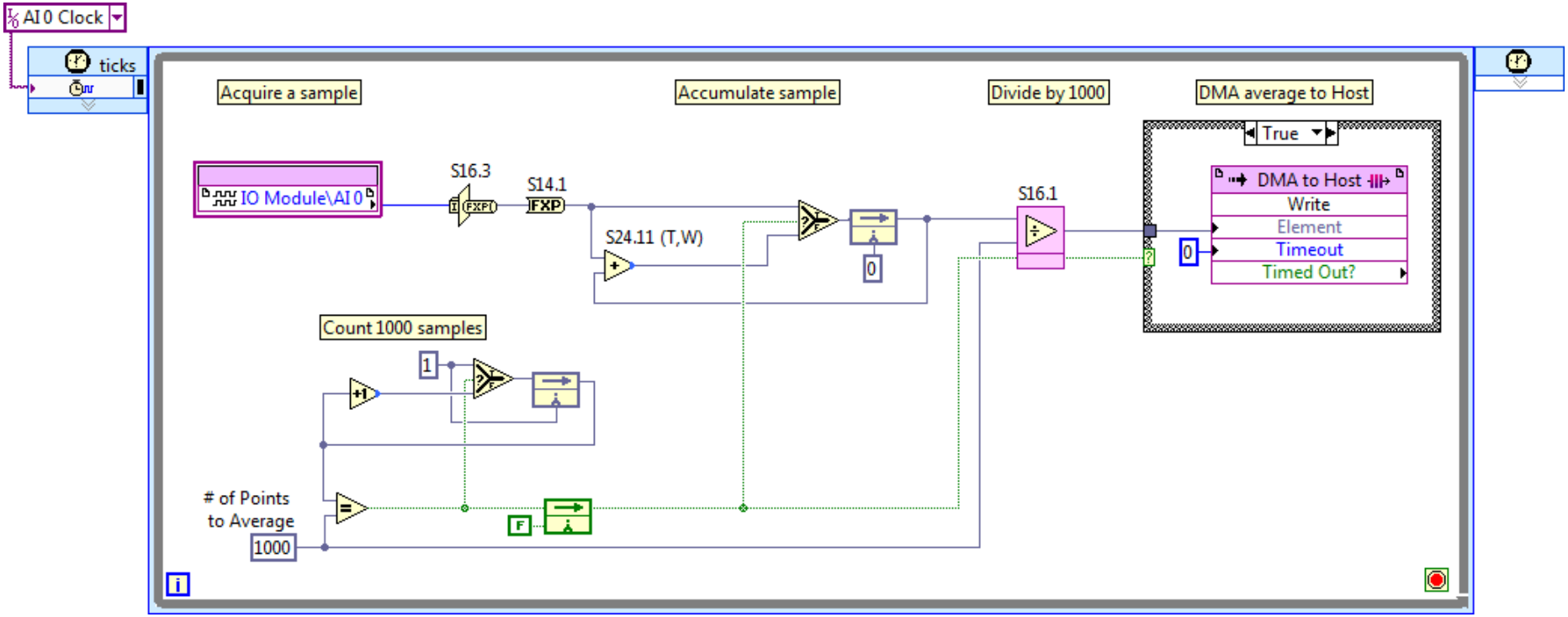
C = takes one or more clock cycles to execute

Inside SCTL



- Lower programming level
  - Program FPGA on a cycle-by-cycle level
  - No For/While loops
  - Fewer LabVIEW functions are supported
    - No native divide
- Manual pipelining to meet FPGA clock frequency constraints
- Application timing is under your control

Third Scheduled Solution



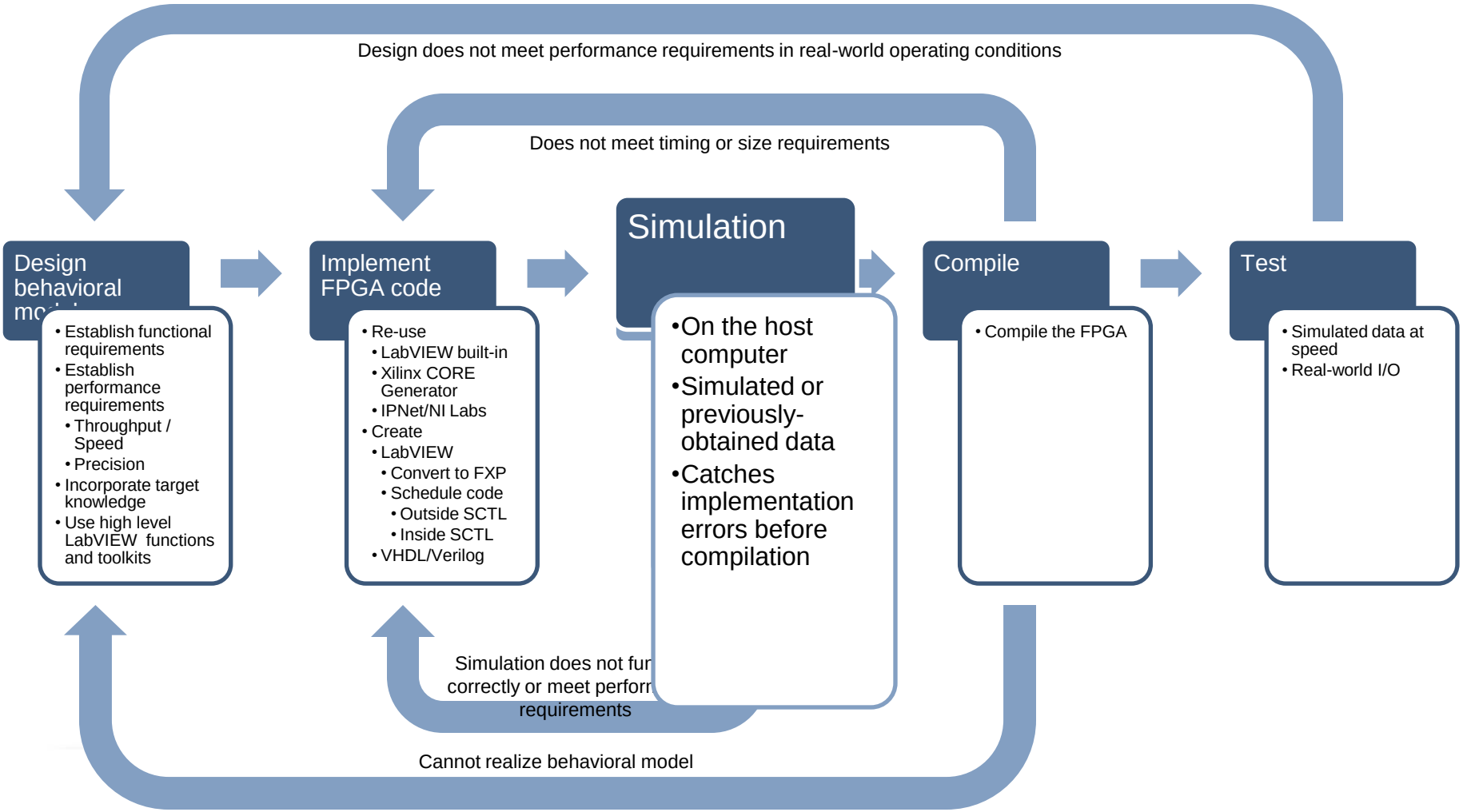
Solution Summary

Implementation	Throughput	Flip-Flops	LUTs	DSP	BRAMs
First Solution – Array, Outside SCTL	< 100 kS/s	54%	30%	0%	2%
Second Solution – Outside SCTL	< 10 MS/s	7%	7%	0%	2%
Third Solution – Inside SCTL	100 MS/s	7%	7%	0%	2%

Integrating VHDL or Verilog Code

Feature	IP Integration Node	CLIP Node
Synthesis file formats	.vhd, .ngc, **.xco**	.vhd, .ngc, .ucf
Auto-Simulation on Development Computer		
Supported LabVIEW data type	Boolean, Boolean array, Integer, Fixed-point	Boolean, Integer, Fixed-point
Execution model	Inline	Parallel
Place of declaration and behavior	Declared locally in an FPGA VI	Declared in a LabVIEW project; acts as a global
Supported execution modes	Inside SCTL Only	Inside and outside SCTL
Support for multiple clock domains	Two Clocks: Integer Multiples	Maximum number of clocks defined by FPGA
Clocks From IP		
Constraints and I/O		

LabVIEW FPGA Design Flow



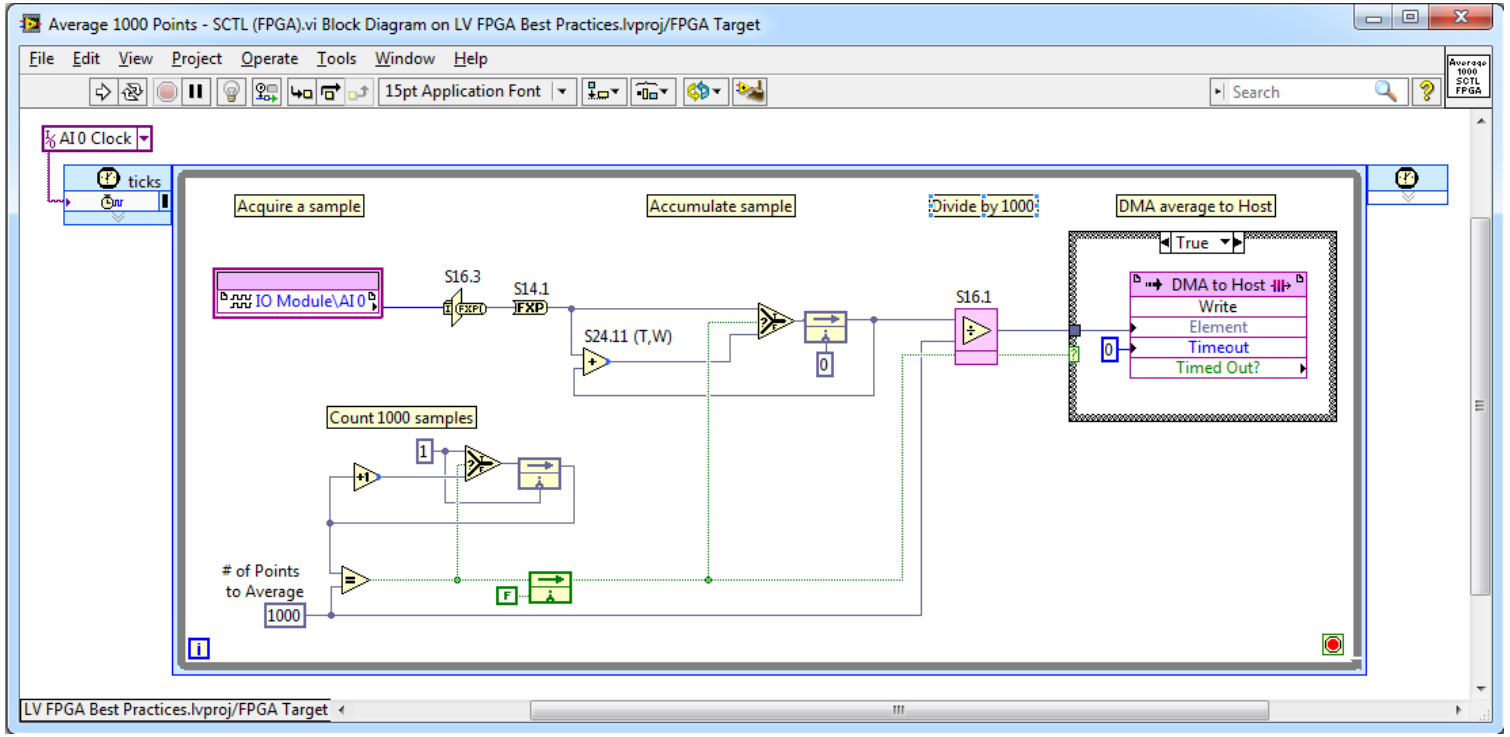
Simulation

- FPGA compiles are long
  - 30 minutes to 30 hours
- Debugging in the FPGA is difficult
- Simulate all FPGA VIs on the host computer if possible

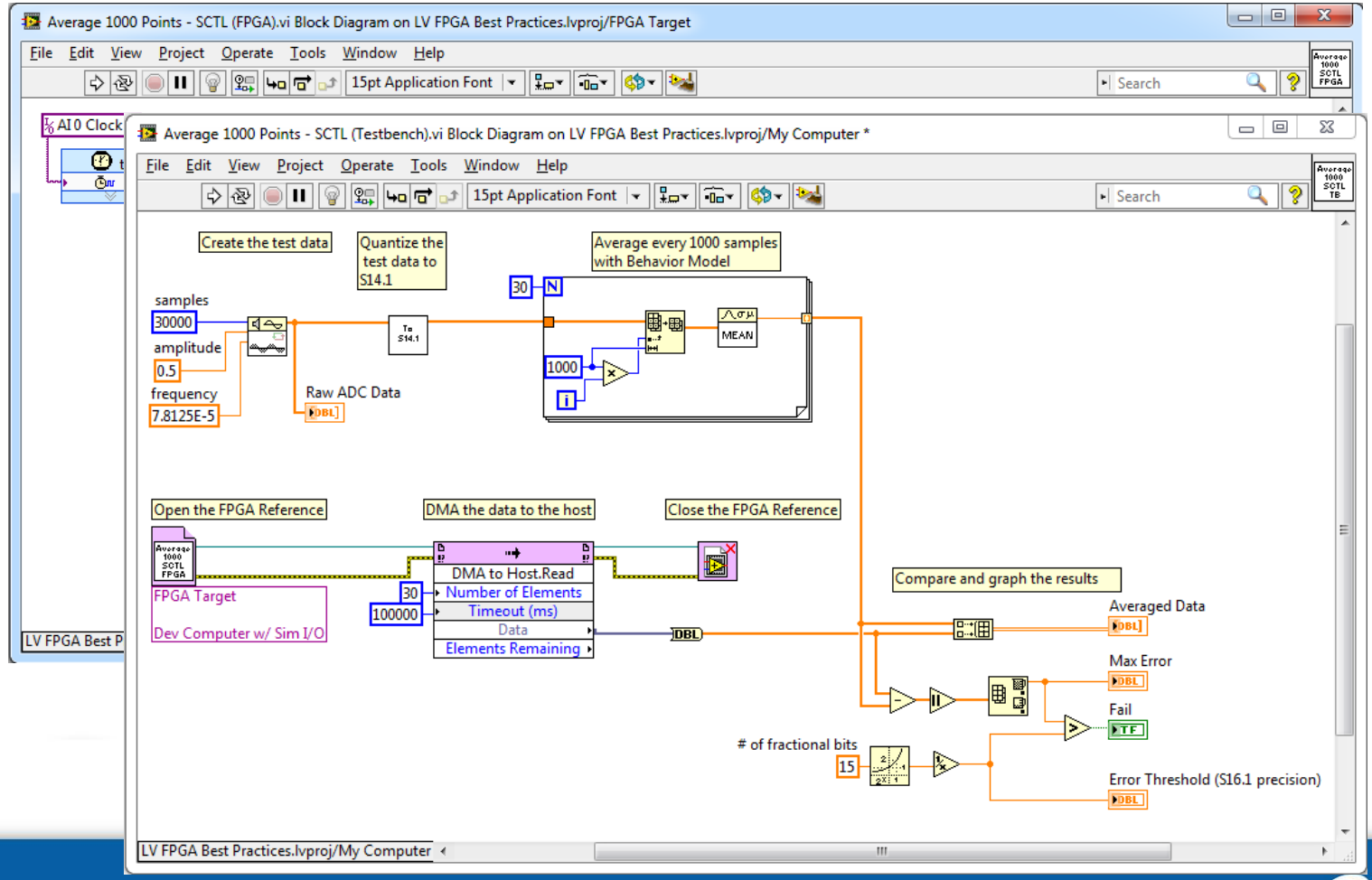
Testbench

- Compare behavior model to DUT
  - Doesn't have to test timing
- Use it to test all LabVIEW FPGA permutations
  - Regression test
- Pass or Fail
- Several focused tests

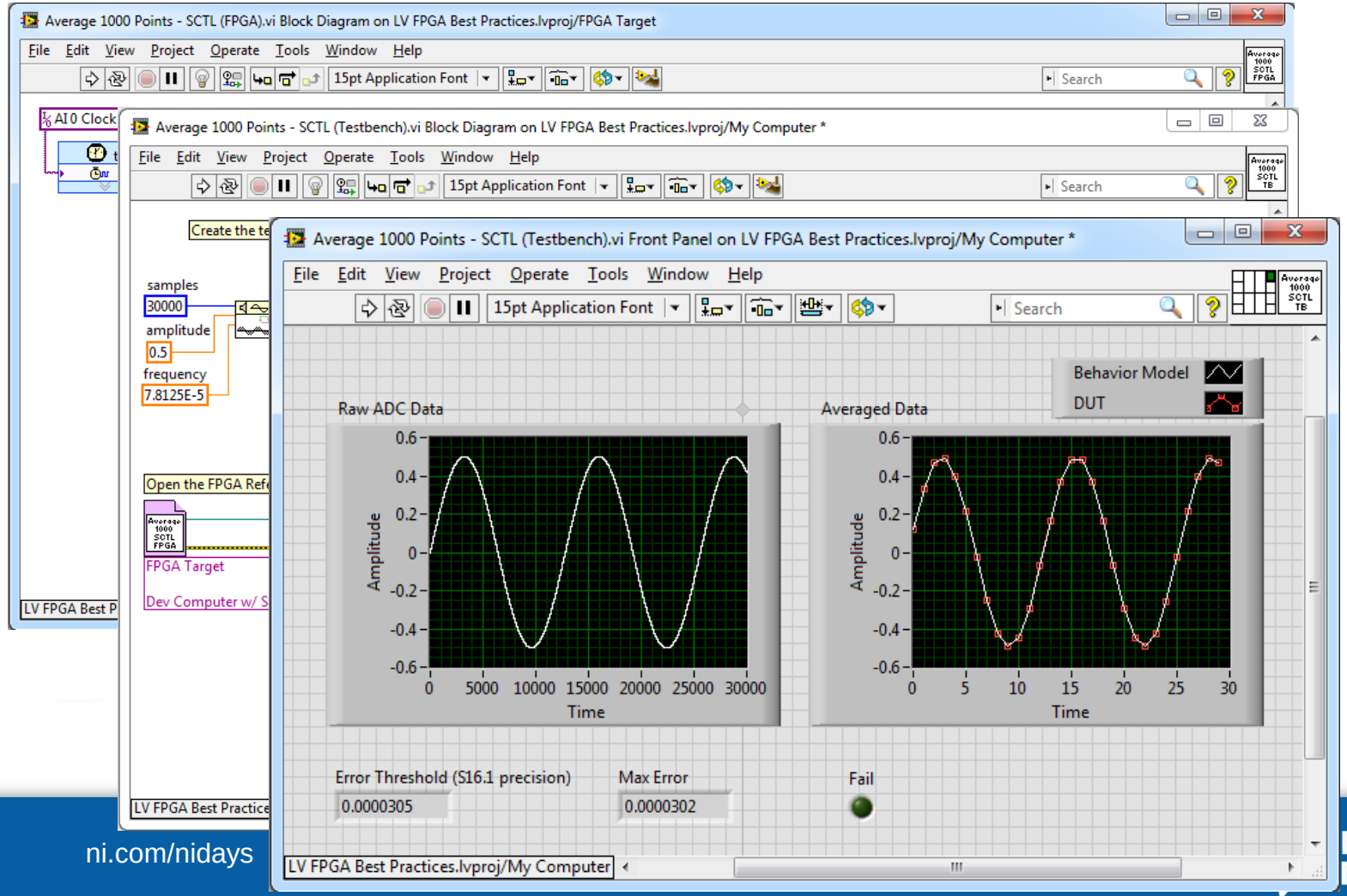
Simulation Results



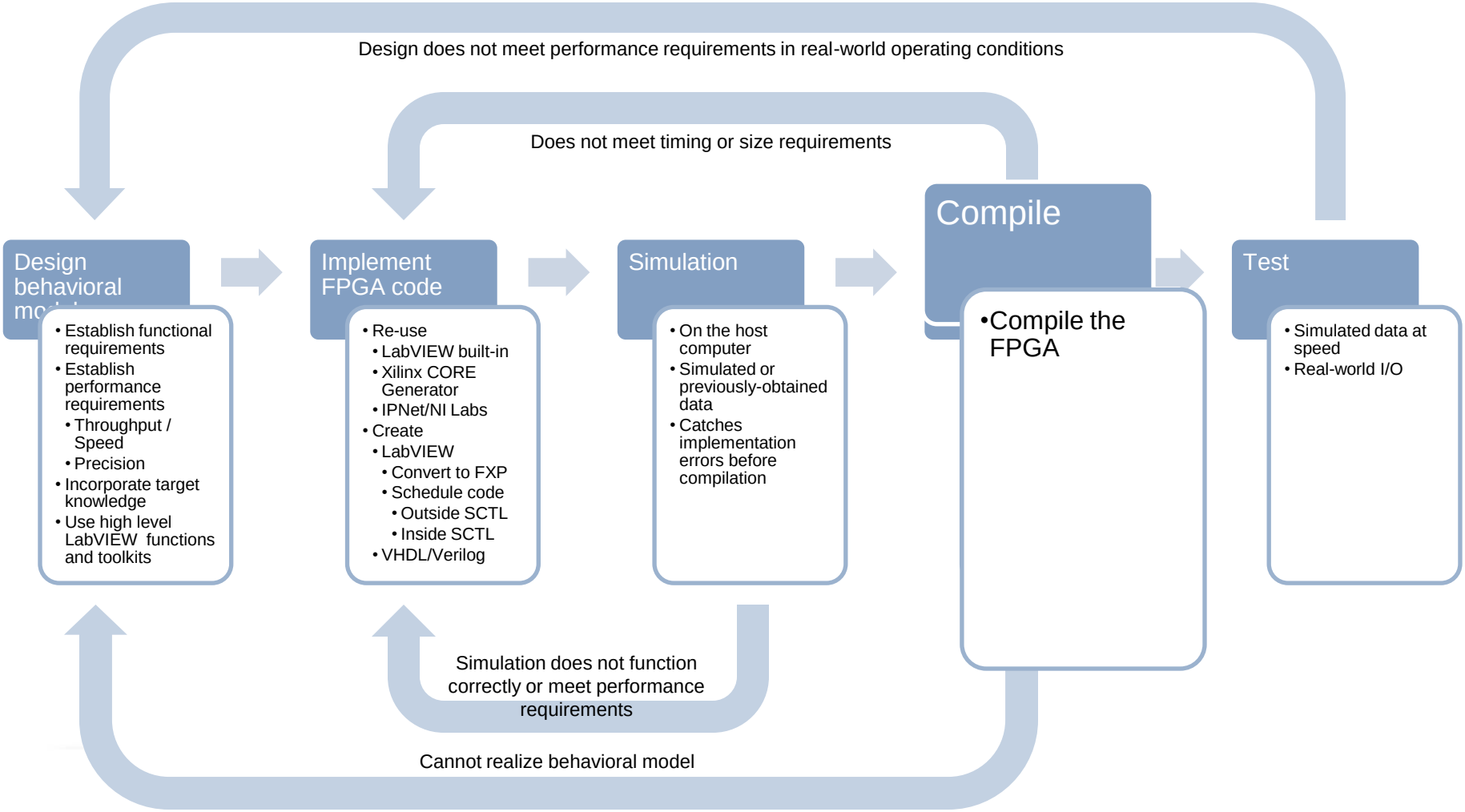
Simulation Results



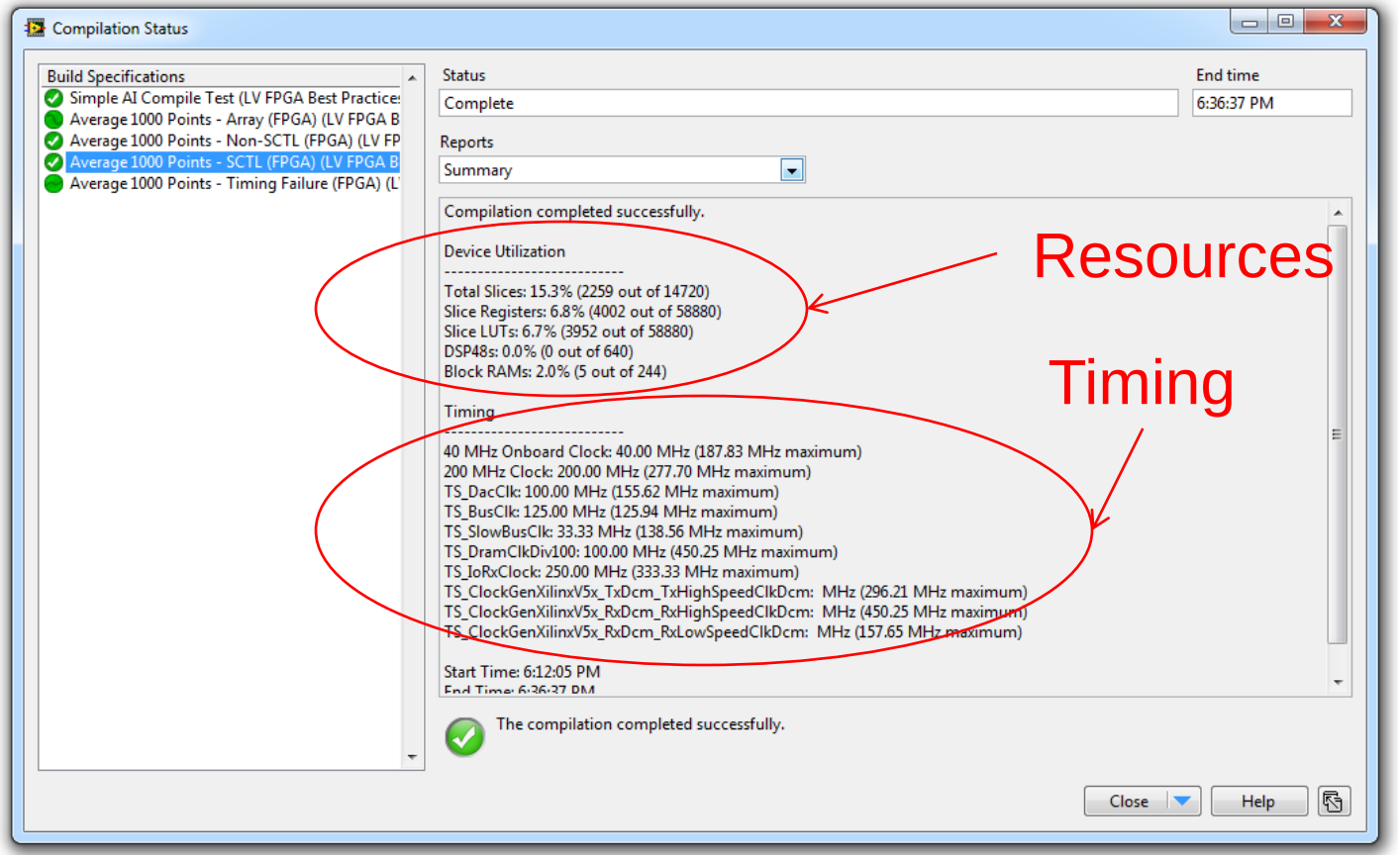
Simulation Results



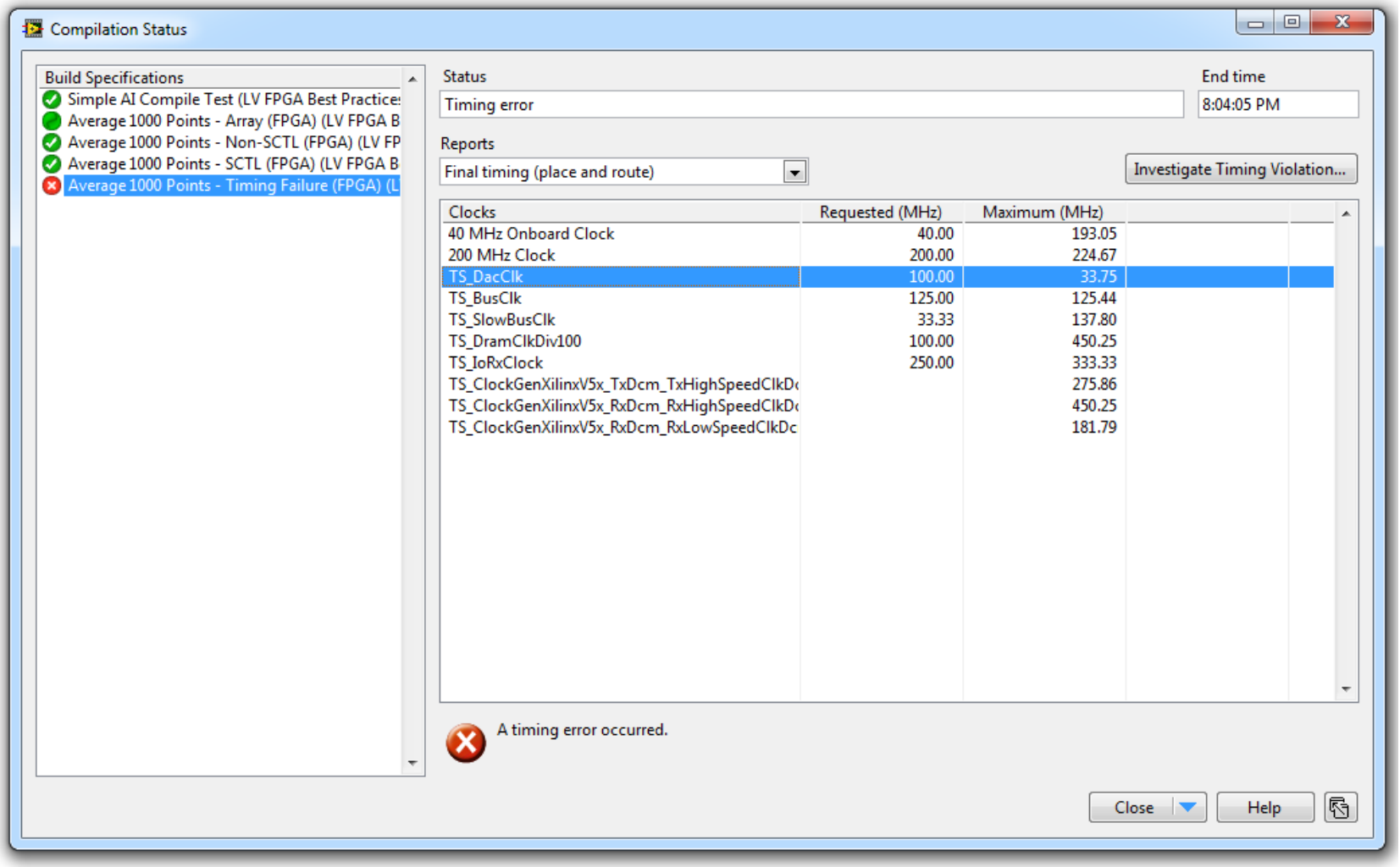
LabVIEW FPGA Design Flow



Compilation Results



Compilation Results



FPGA Design is too Large

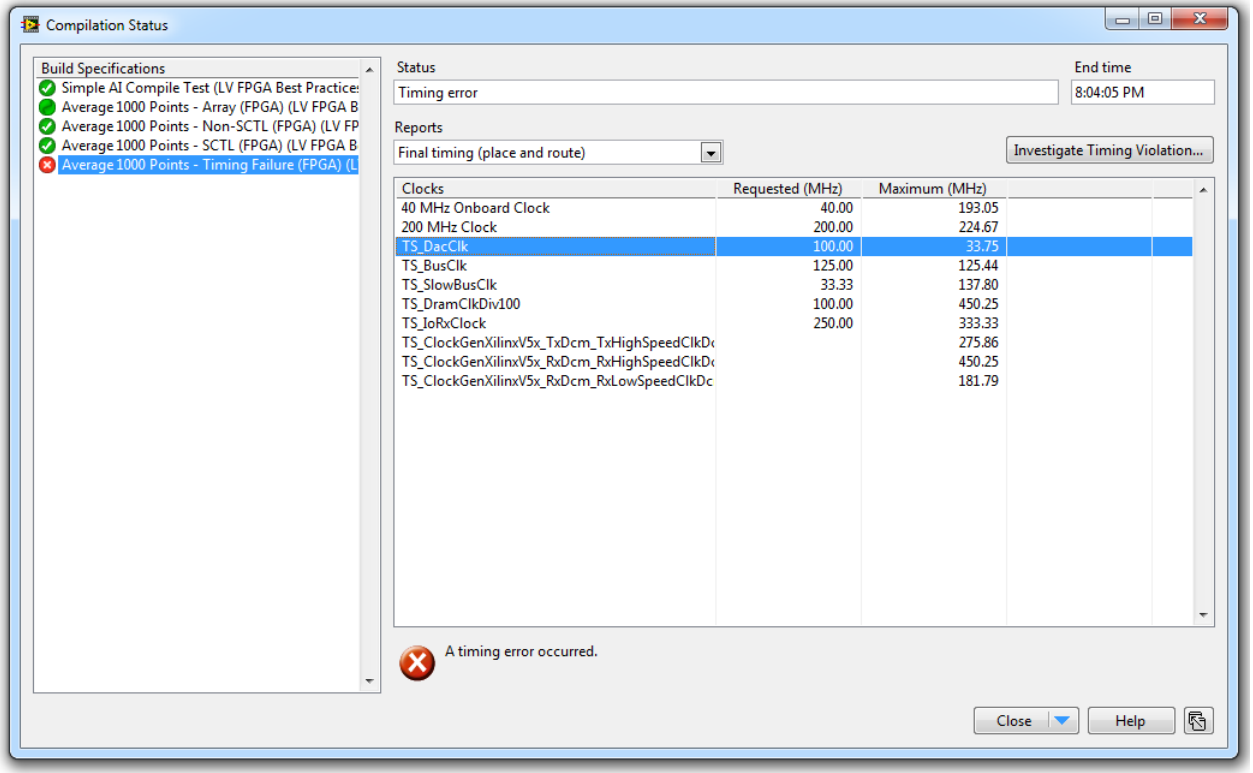
- Review your algorithm choice
- Resource balancing
- Bit-width pruning
  - Customize or reduce FXP bit widths
- Move functionality to host computer
  - Division in our design problem

FPGA Design is too Large

- Time multiplexing
- Stream the data to another LabVIEW FPGA device
- Buy a card with a bigger FPGA

FPGA Design is too Slow

- Identify failing paths



FPGA Design is too Slow

- Identify failing paths

Timing Violation Analysis (Average 1000 Points - Timing Failure (FPGA).vi on LV FPGA Best Practices.lvproj/FPG...

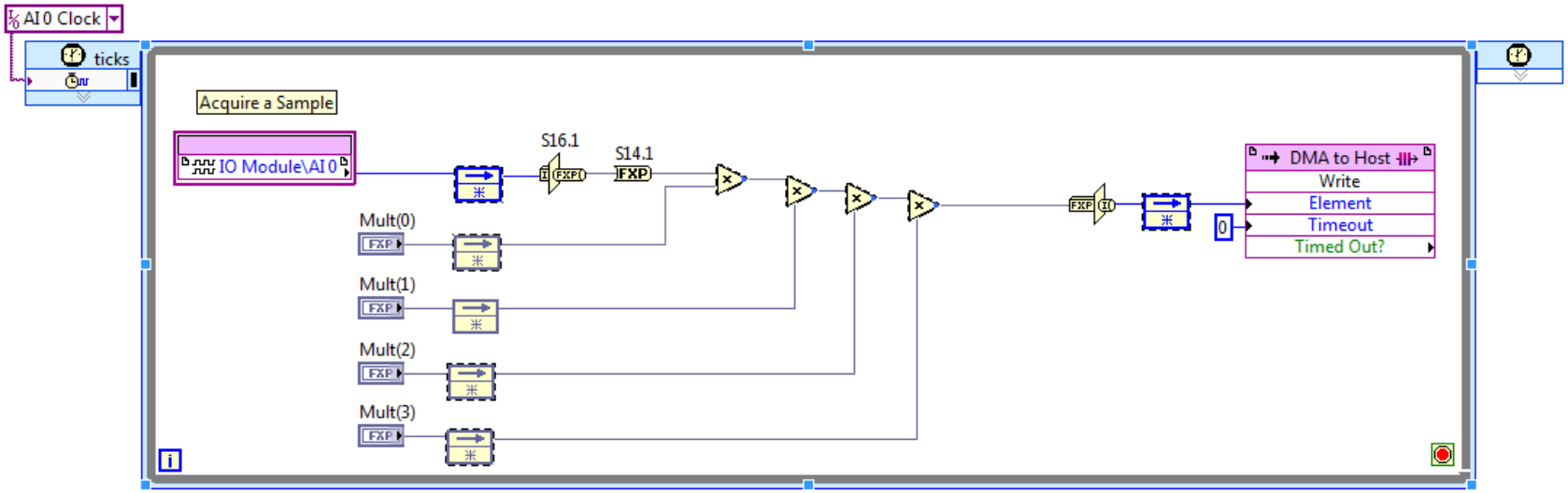
Timing Information

Paths	Total Delay	Logic Delay	Routing Delay	Max Fanout
Path 1 : Requirement 10.00ns missed by 19.46ns	29.33	16.95	12.38	16
Timed Loop	29.33	16.95	12.38	16
Feedback Node	0.66	0.66	0.00	
Multiply	4.46	3.65	0.81	1
Feedback Node	2.94	0.19	2.75	4
Multiply	7.18	4.78	2.40	15
Multiply	5.37	3.74	1.63	1
Feedback Node	1.67	0.09	1.58	16
Multiply	5.37	3.74	1.63	1
Feedback Node	1.67	0.10	1.57	16
Path 2 : Requirement 10.00ns missed by 19.46ns	29.33	17.90	11.43	16
Timed Loop	29.33	17.90	11.43	16
Feedback Node	0.66	0.66	0.00	
Multiply	4.46	3.65	0.81	1
Feedback Node	2.85	0.09	2.75	4
Multiply	7.18	4.78	2.40	15
Multiply	6.98	4.88	2.11	16
Multiply	5.53	3.74	1.79	1
Feedback Node	1.67	0.10	1.57	16
Path 3 : Requirement 10.00ns missed by 19.63ns	29.50	17.92	11.57	16
Timed Loop	29.50	17.92	11.57	16
Feedback Node	0.66	0.66	0.00	
Multiply	4.46	3.65	0.81	1
Feedback Node	2.94	0.19	2.75	4
Multiply	7.18	4.78	2.40	15
Multiply	5.37	3.74	1.63	1
Feedback Node	1.67	0.09	1.58	16

Close | Show Element | Show Path | Help

FPGA Design is too Slow

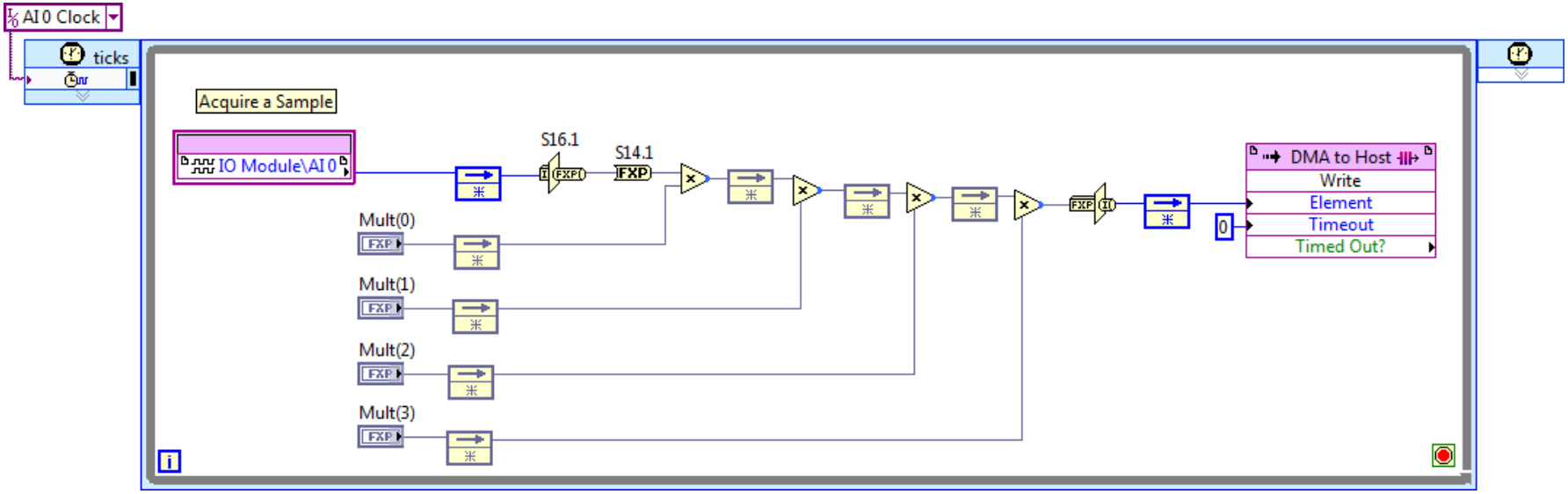
- Identify failing paths



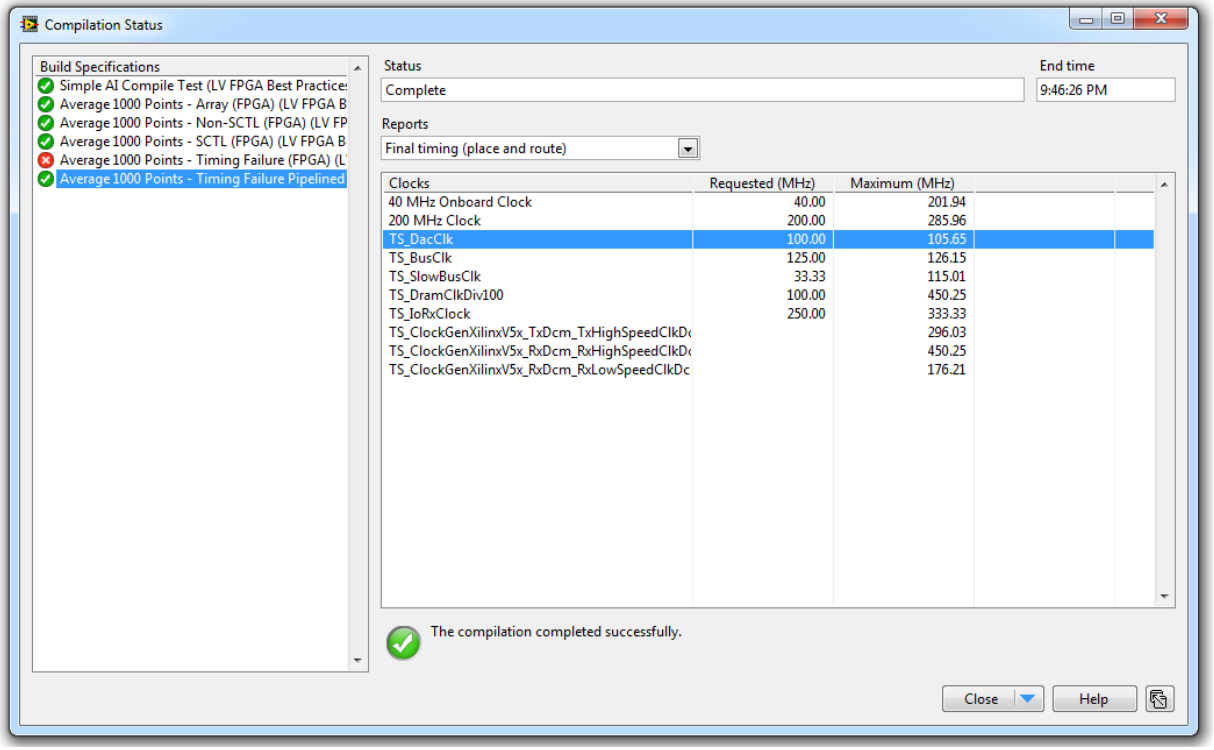
Fixing Timing Failures

- Can the logic run at a slower clock rate?
  - Divider in our design problem
- Pipeline the failing paths
- If you are close, compile again with a random seed

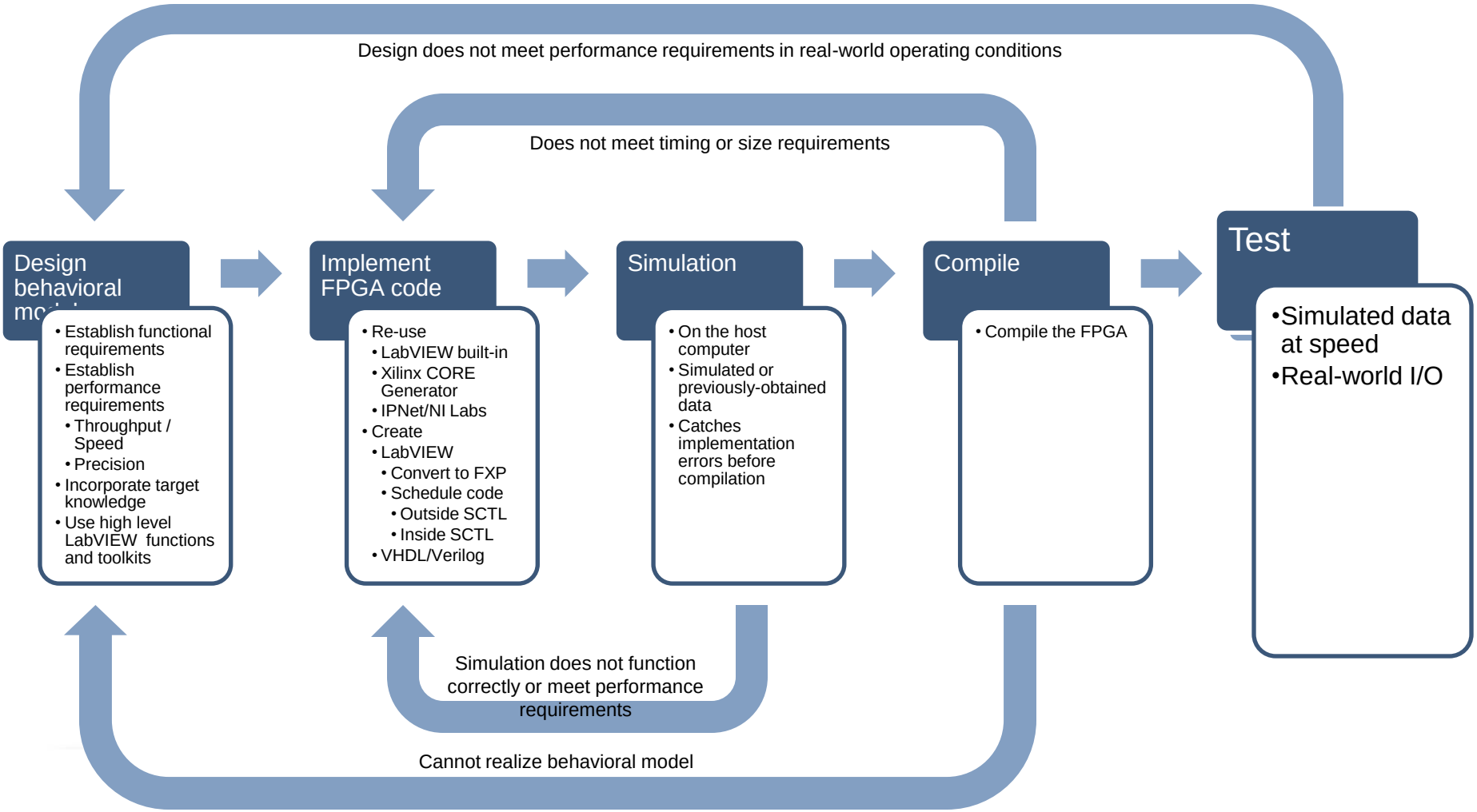
Pipelined Solution



Pipelined Solution



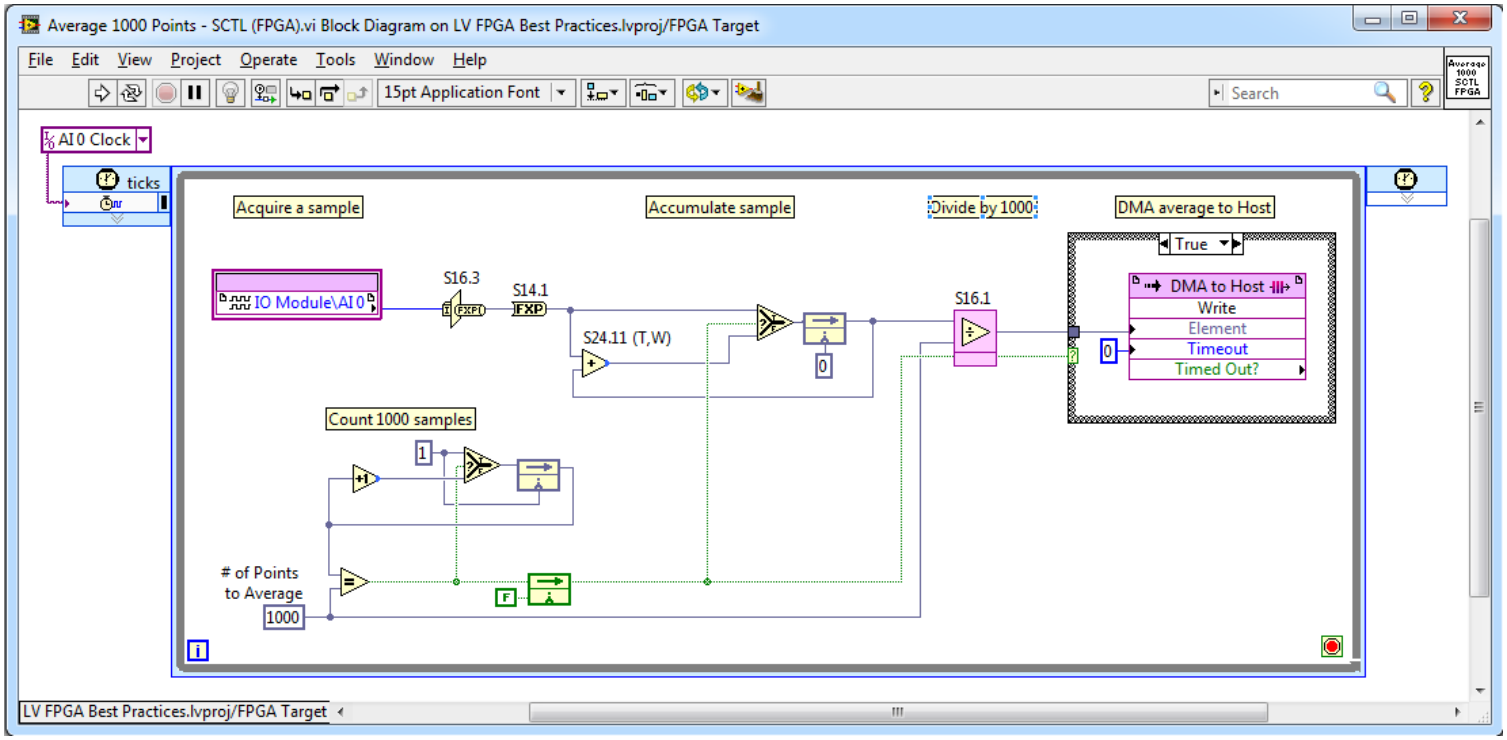
LabVIEW FPGA Design Flow



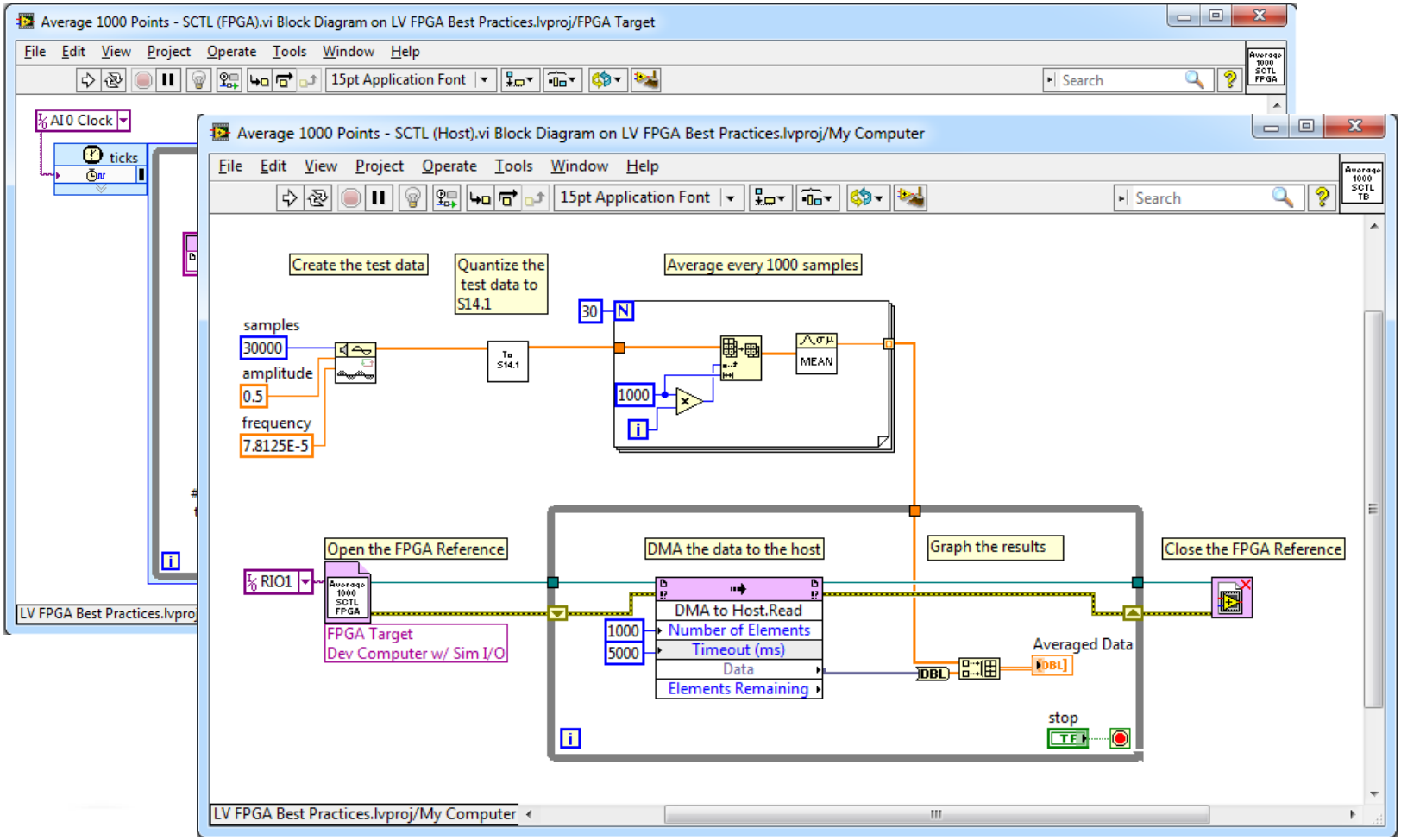
Host interface timing

- Host interface timing is much slower than the FPGA
  - Microseconds or milliseconds rather than nanoseconds
  - Not reflected in LabVIEW simulation on the host computer
- Detect rising edges in the FPGA on strobe signals from the host computer
  - Write enable signal to memory
- Latch error events in the FPGA and read occasionally on the host computer
  - DMA FIFO overflow

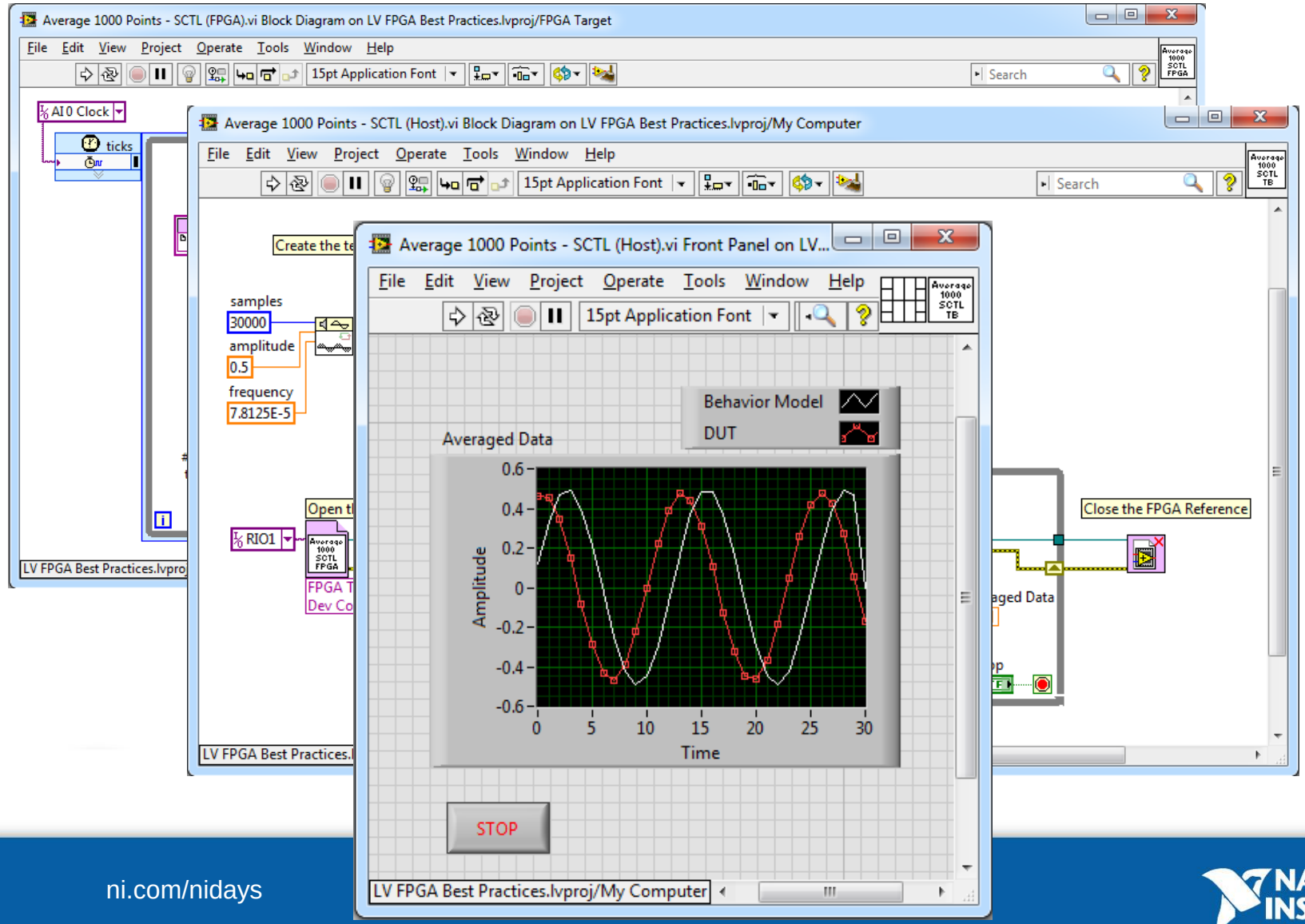
Real-World Results



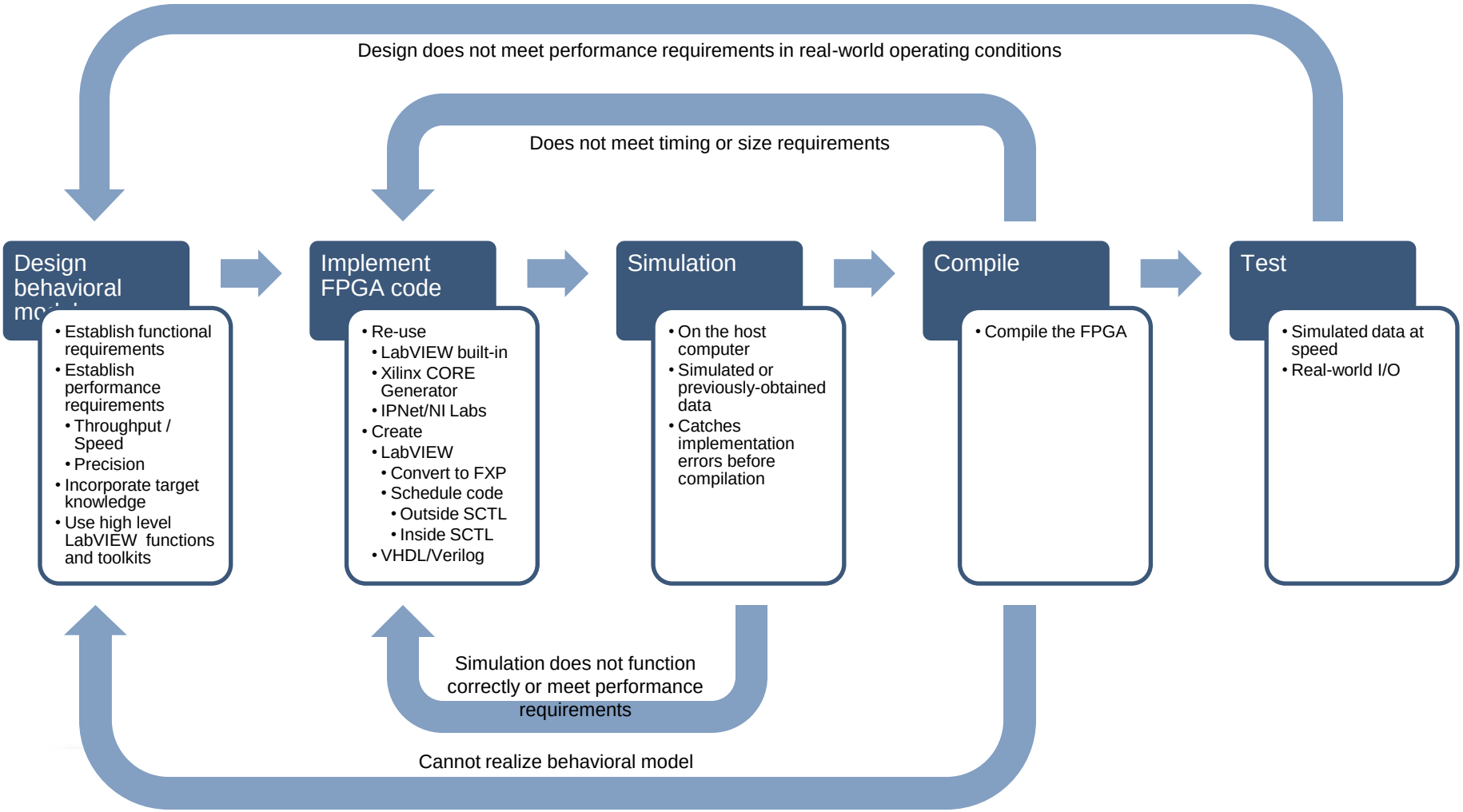
Real-World Results



Real-World Results



LabVIEW FPGA Design Flow



Thank You

Questions?