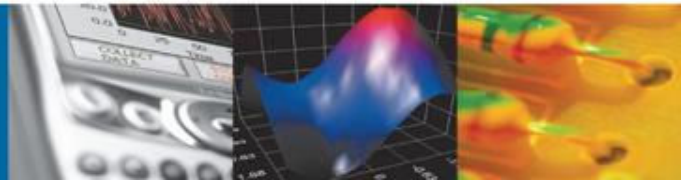
Incorporating FPGAs in Test Applications

NI Technical Conference

Long Island Nov 10, 2009

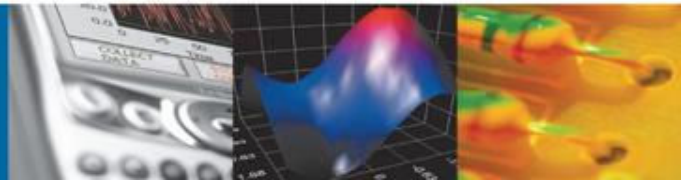
Terry Stratoudakis, P.E.

ALE System Integration

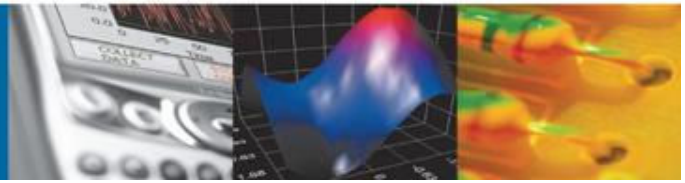


Agenda

- Introduction to FPGAs in test
- FPGAs in test applications
  - New FPGA-enabled applications
  - FPGA-enhanced applications
- FPGA for test hardware
  - Examples
- Programming in LabVIEW FPGA for test applications

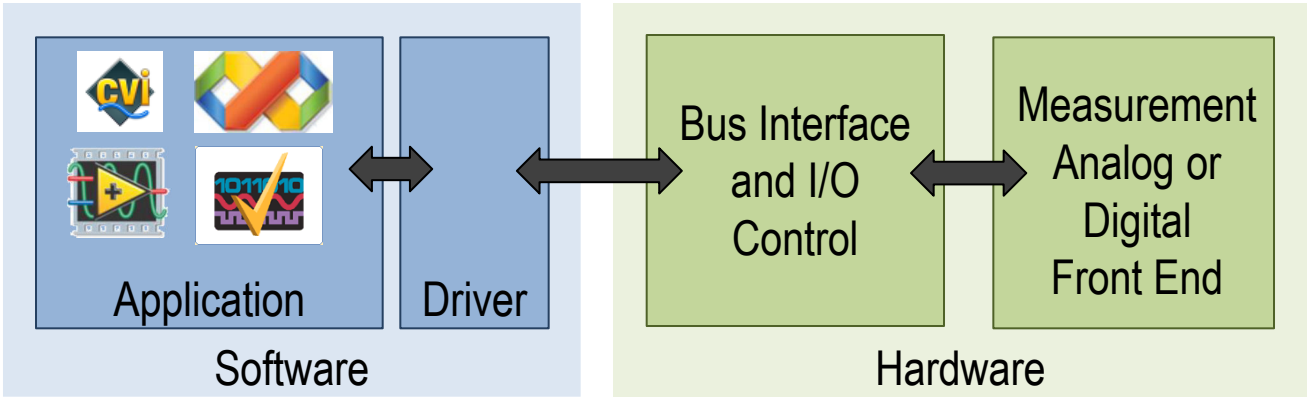


Introduction to FPGAs in Test



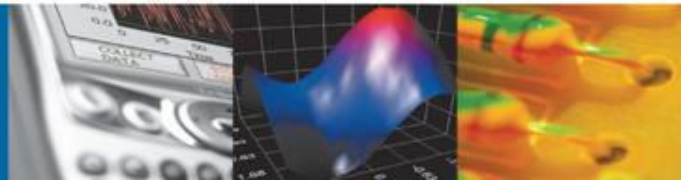
Software-Defined Test System Architecture

Standard Virtual Instrumentation Model



Field Programmable Gate Arrays

- Introduced in 1987
- Customizable Integrated Circuit (IC)
  - Similar to Application Specific ICs (ASICs)
- No Operating System
- Configured with Hardware Descriptor Language (HDL)
- Parallel Execution
- Millions of configurable gates on a single chip

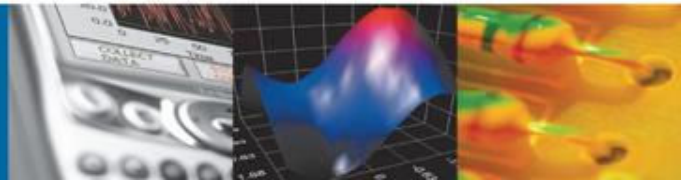


Who makes FPGAs?

- Actel
- Aeroflex UTMC
- Lattice Semiconductor
- And others...
- Altera*
- Atmel
- Xilinx*

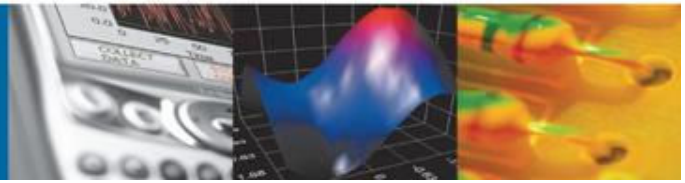
*Market leaders

http://seekingalpha.com/article/85478-altera-and-xilinx-report-the-battle-continues



FPGA Configuration Tools

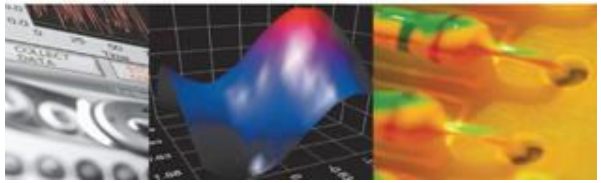
- Direct: Hardware Description Languages (HDL)
  - akin to assembly language
  - Examples: Verilog and VHDL
- High-Level
  - FPGA design accessible to software engineers
  - Text based using C or Matlab with concurrency models
    - Examples: “Impulse C” and “Mentor Graphics Catapult C”
  - Graphical/Flow based with LabVIEW FPGA



VHDL Adder Example

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;

entity test_add is
  generic ( width : integer := 17 );
  port ( clk         : in  std_ulogic;
         reset       : in  std_ulogic;
         enable      : in  std_ulogic;
         inp1, inp2  : in  std_logic_vector ( width downto 0);
         sum         : out std_logic_vector ( (width + 1) downto 0) );
end test_add;

--------------------------------------------------------------------------------
--  RTL description.  Adds two inputs together (unsigned) into an integer
--  of "width + 1" in lenght.
--------------------------------------------------------------------------------

architecture rtl of test_add is
  constant terminal_count     : integer := 2**( sum'high + 1 ) - 1;
  subtype  adder_range is integer range 0 to terminal_count;
  signal   sumx, inp1x, inp2x : adder_range;
begin  --  rtl

  sum   <= std_logic_vector( to_unsigned ( sumx, width + 2 ));
  inp1x <= to_integer ( unsigned ( inp1 ) );
  inp2x <= to_integer ( unsigned ( inp2 ) );

  adder : process ( clk, reset )
  begin
    if reset = '0' then
      sumx <= 0;
    elsif rising_edge (clk) then
      if enable = '1' then
        sumx <= inp1x + inp2x;
      end if;
```

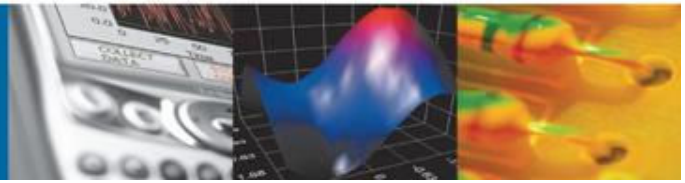
```
begin  --  rtl

  VCC    <= '1';
  GND    <= '0';
  mulout <= local_mulout;

  U1 : test_clkgen
    port map (
      clk   => clk,
      reset => reset,
      dclk  => dclk_i );

  J2 : test_counter
    generic map ( width => 5 )
    port map (
      clk    => dclk,
      reset  => reset,
      enable => VCC,
      count  => slow_count
      );

  J3 : test_reg
    generic map ( width => internal_data'high )
    port map (
      clk    => clk,
      reset  => reset,
      enable => VCC,
      sel    => VCC,
      inp1   => input_data,
      inp2   => local_count,
      outpt  => internal_data
      );

  J4 : test_counter
    generic map ( width => internal_data'high + 1 )
    port map (
      clk    => clk,
      reset  => reset,
      enable => enable,
      count  => local_count
      );
```



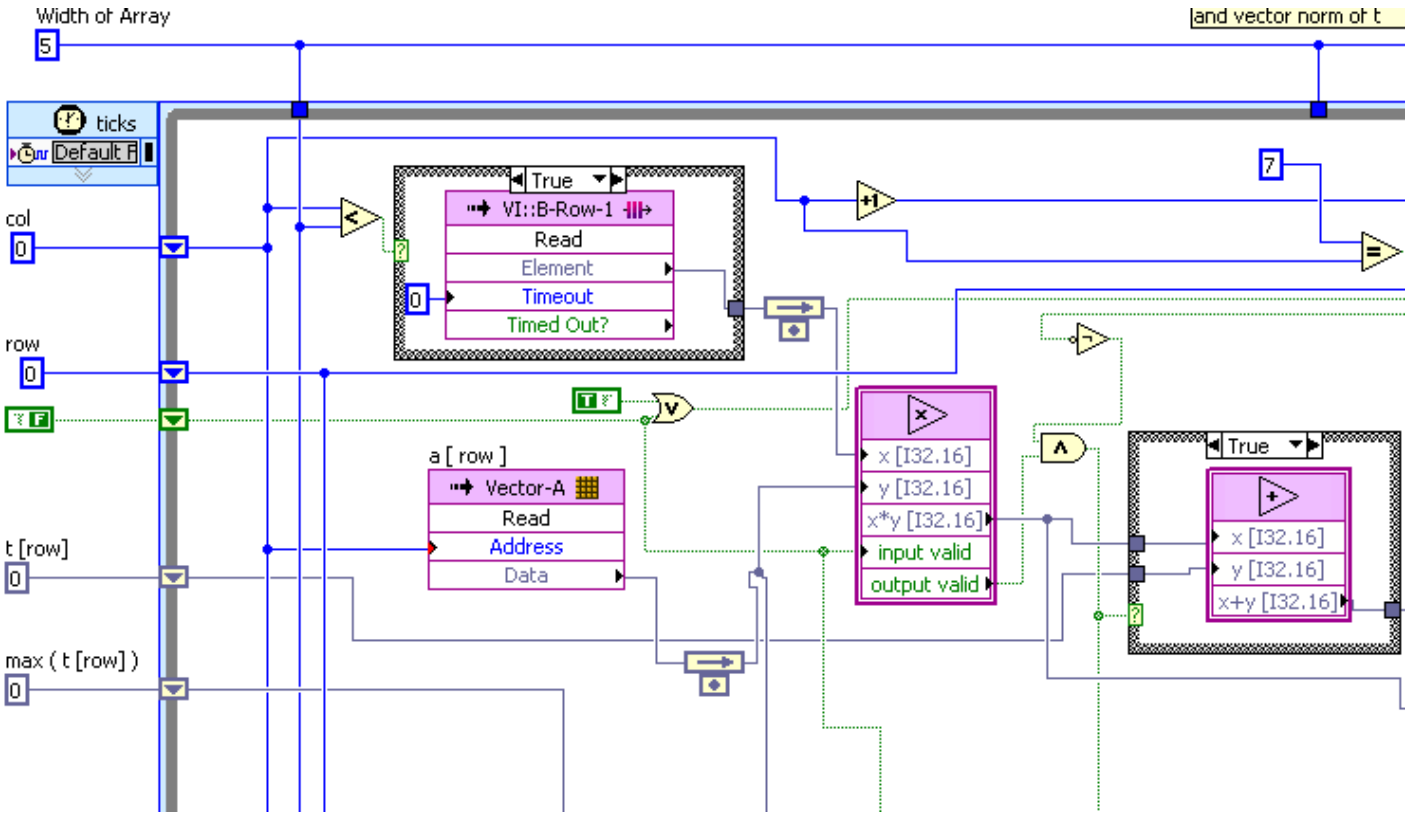
C-Based Example Image Filter excerpt*

```
do {
  for ( i = 2; i < HEIGHT; i++ ) {
    // Note: the following loop will pipeline with a rate of
    // one cycle if the target platform supports dual-port RAM.
    for (j=0; j < WIDTH; j++) {
    #pragma CO PIPELINE
        p04 = B[j];
        p14 = C[j];
        p24 = D[j];
        p34 = E[j];
        co_stream_read(input_stream, &p44, sizeof(co_uint16));
        co_stream_write(r0, &p04, sizeof(co_uint16));
        co_stream_write(r1, &p14, sizeof(co_uint16));
        co_stream_write(r2, &p24, sizeof(co_uint16));
        co_stream_write(r3, &p34, sizeof(co_uint16));
        co_stream_write(r4, &p44, sizeof(co_uint16));
        B[j] = p14;
        C[j] = p24;
        D[j] = p34;
        E[j] = p44;
    }
  }
  IF_SIM(break;)  // For simulation we break after one frame
} while (1);
```

*Using “Impulse C”

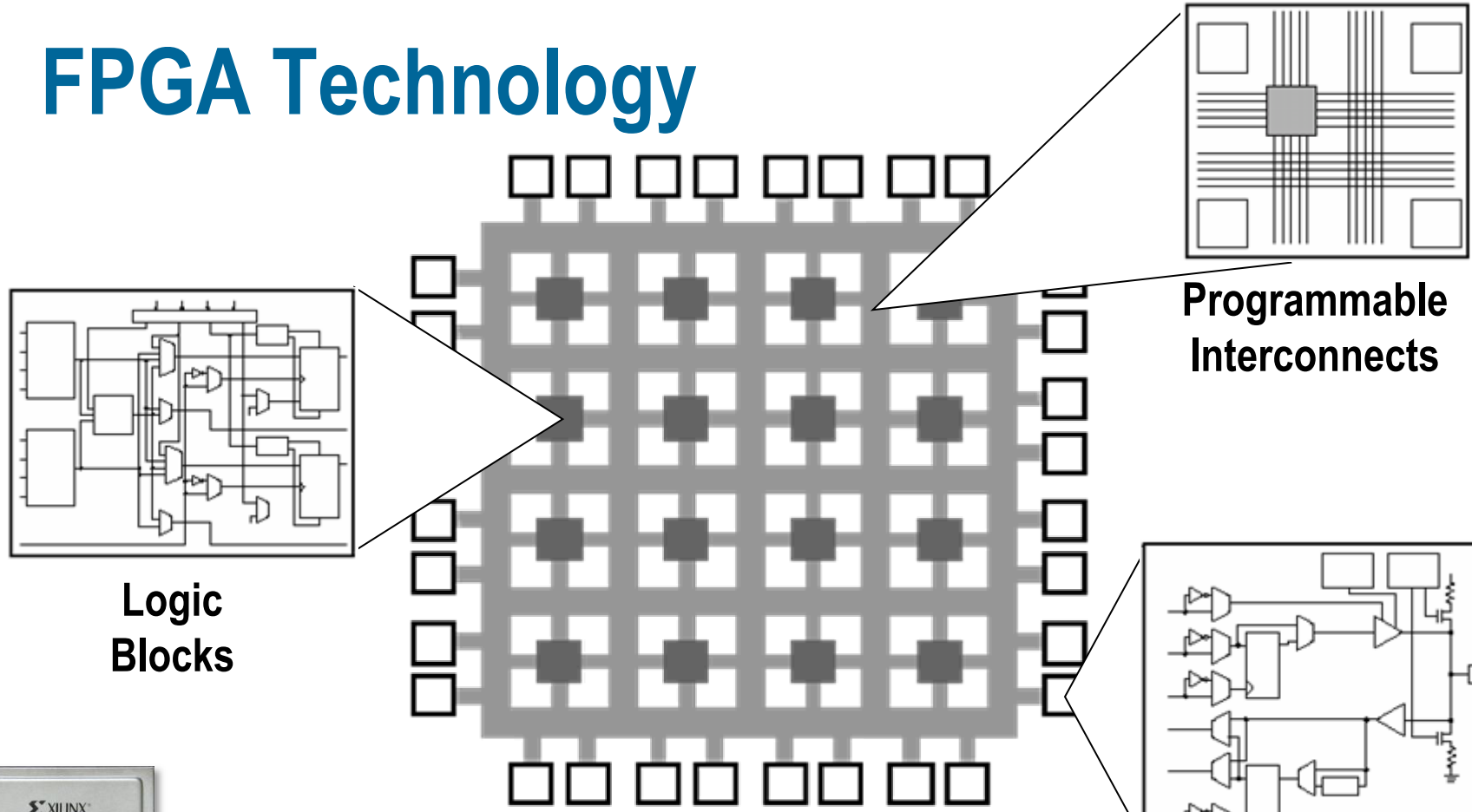


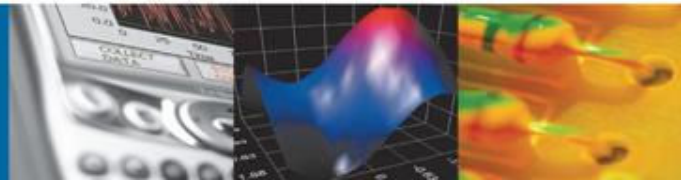
Graphical Based Matrix Math Example*



*Using “LabVIEW FPGA”

FPGA Technology

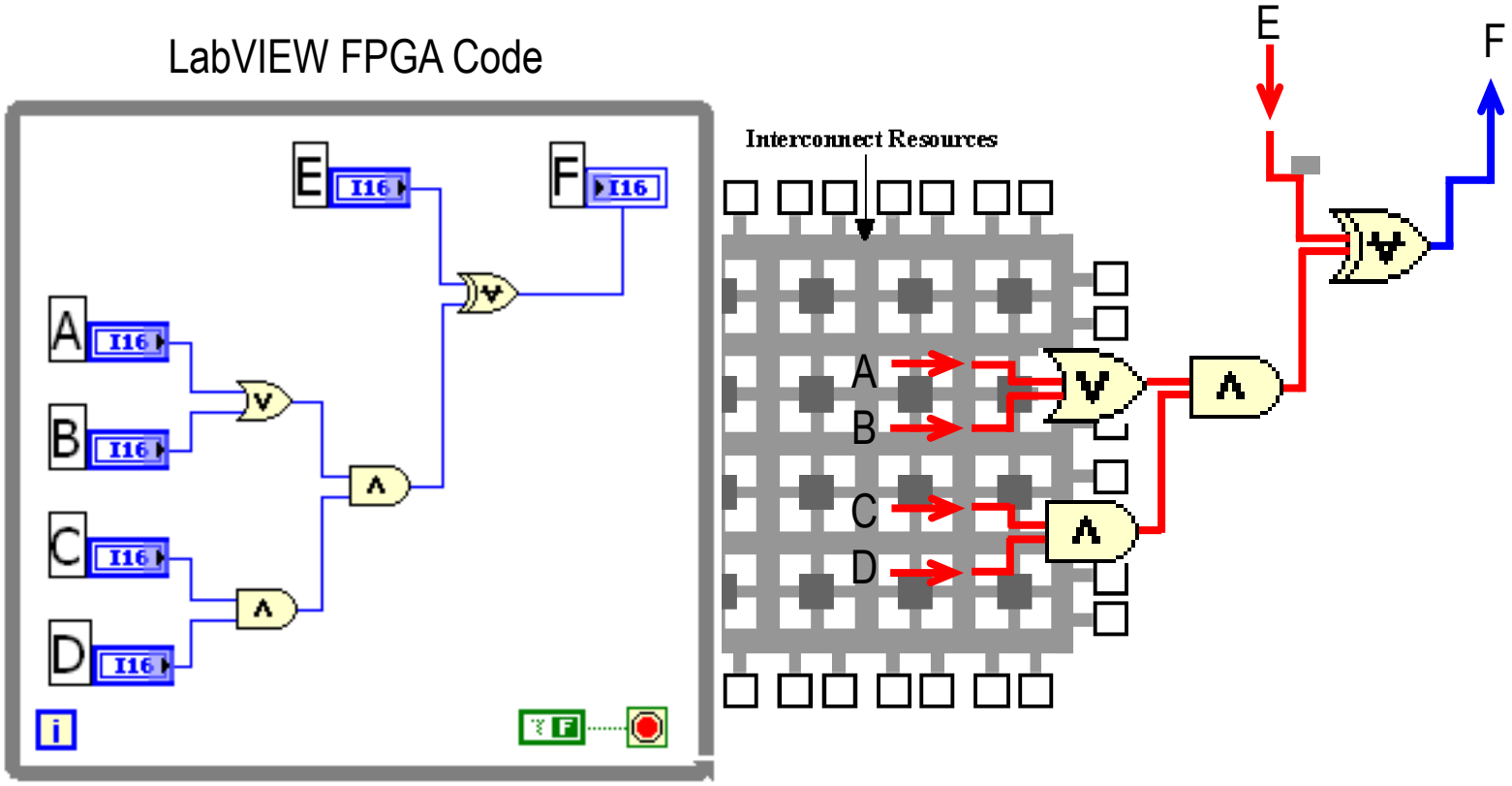




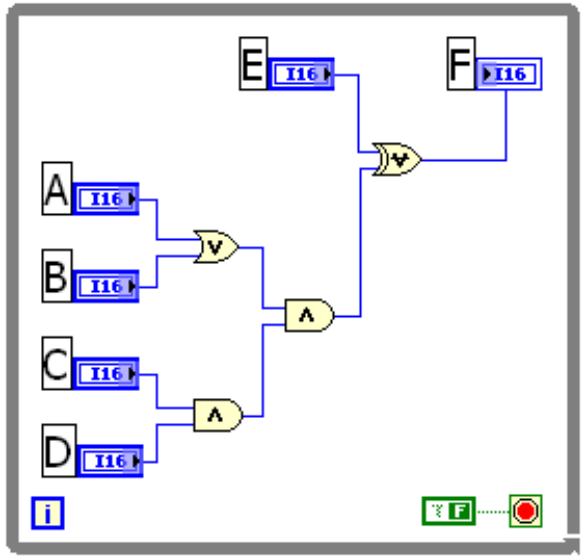
FPGA Logic Implementation

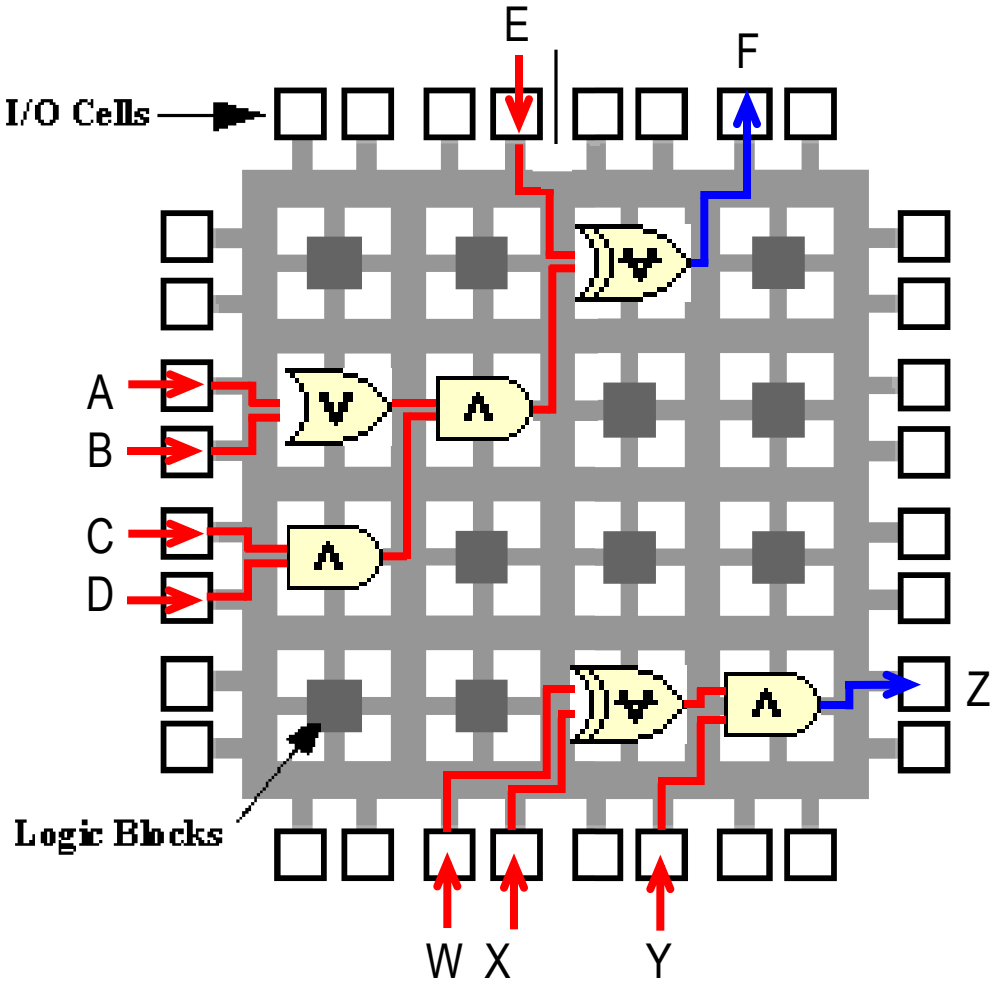
Implementing Logic on FPGA: F = {(A+B)CD} ⊕ E

LabVIEW FPGA Code



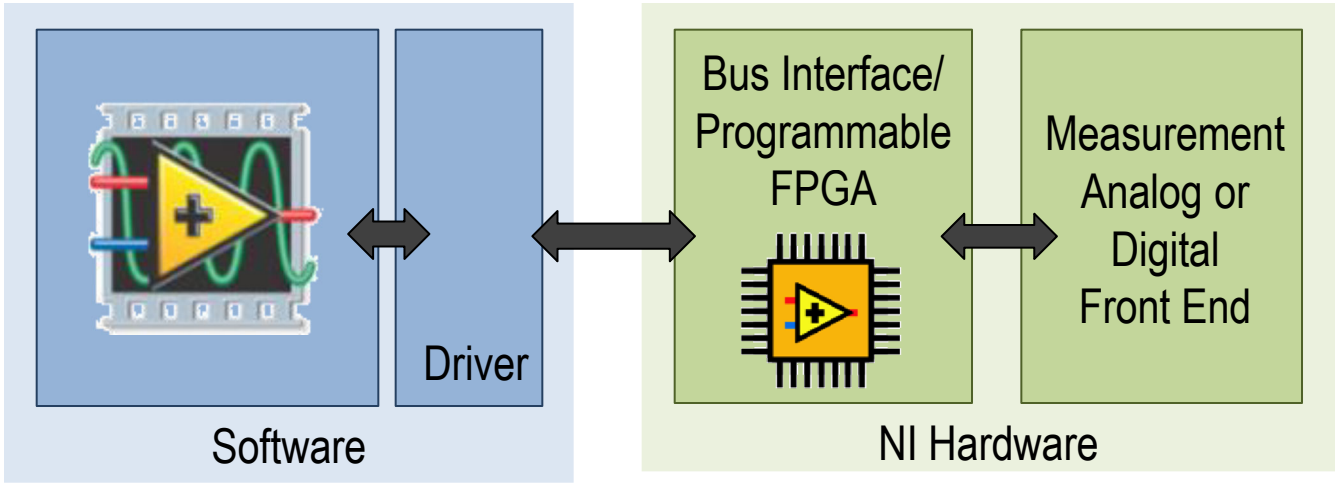
True Parallelism





FPGA-Based Test System Architecture

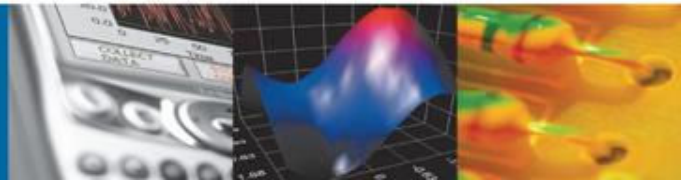
Open FPGA System Model



- System intelligence and decision making can be moved from software to hardware

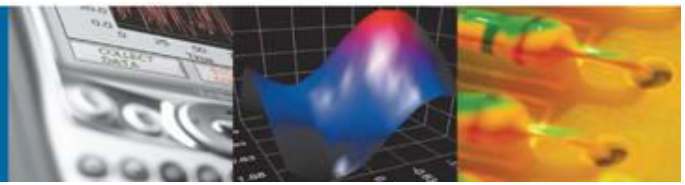
Benefits of FPGAs in Test Systems

- ***High Reliability*** – Designs implemented in hardware
- ***Low Latency*** – Run algorithms at deterministic rates down to 5 ns
- ***Reconfigurable*** – Create DUT / application-specific personalities
- ***High Performance*** – Computational abilities open new possibilities for measurement and data processing speed
- ***True Parallelism*** – Enables parallel tasks and pipelining, reducing test times



FPGAs in Test Applications

New FPGA-enabled applications



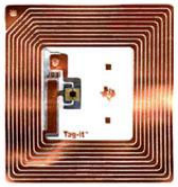
RFID Testing

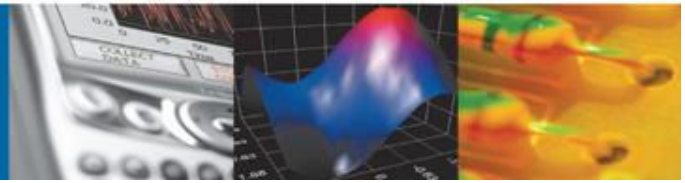
RFID
Reader (Emulated)

Stimulus

Response

RFID
Tag (DUT)

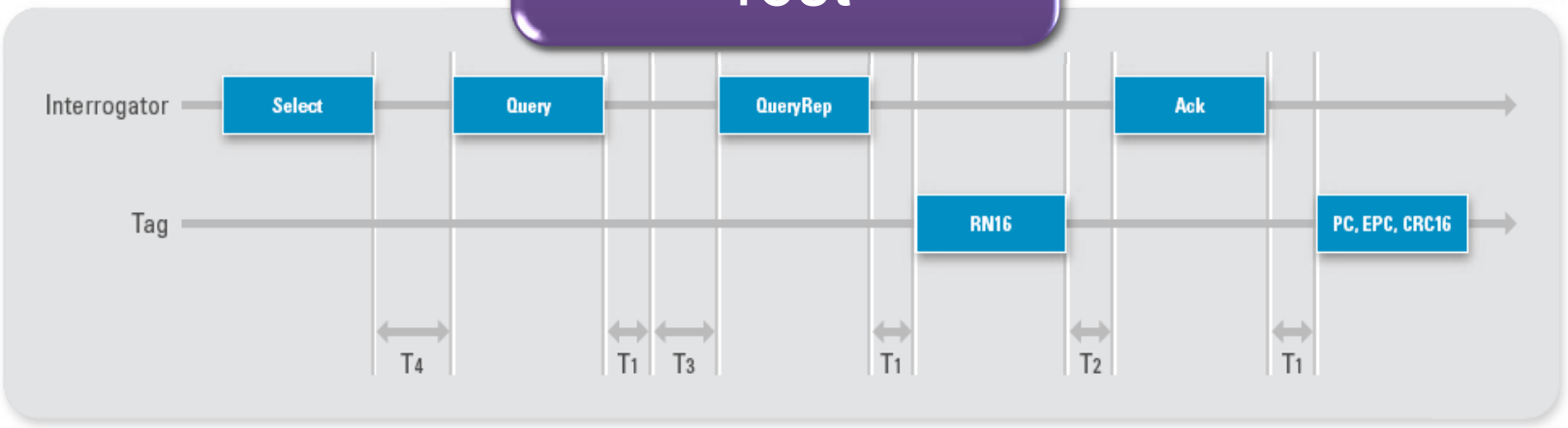




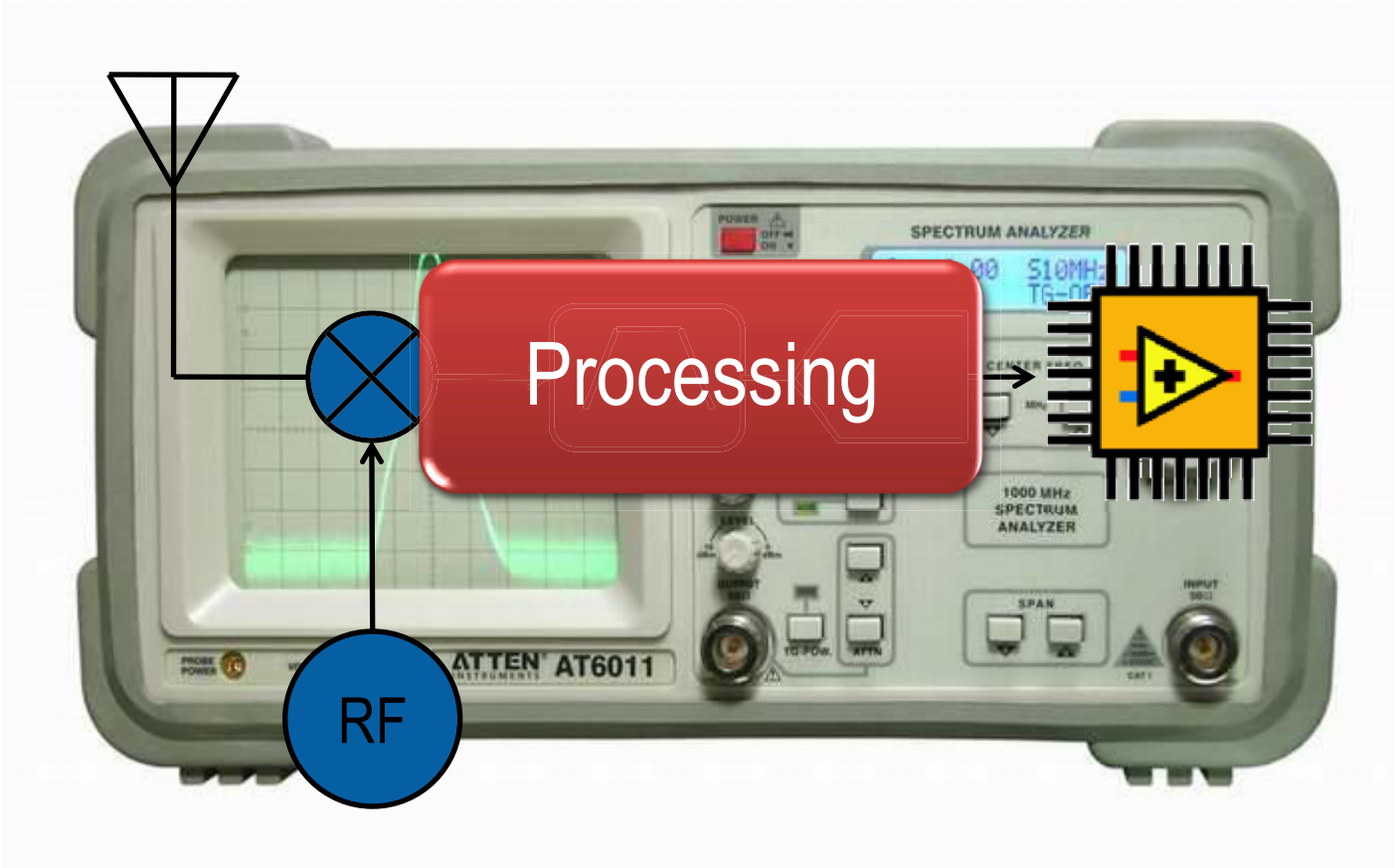
RFID Testing – Response-Stimulus

- Testing an RFID tag requires emulating the tag reader
  - Interrogates and responds to tag within microseconds
- Coding/decoding, modulation/demodulation, and decision making must be completed in hardware to meet timing

Closed-Loop Test

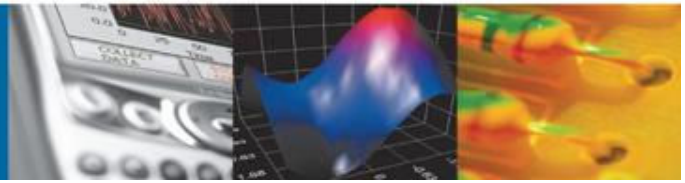


Real-time spectral measurements

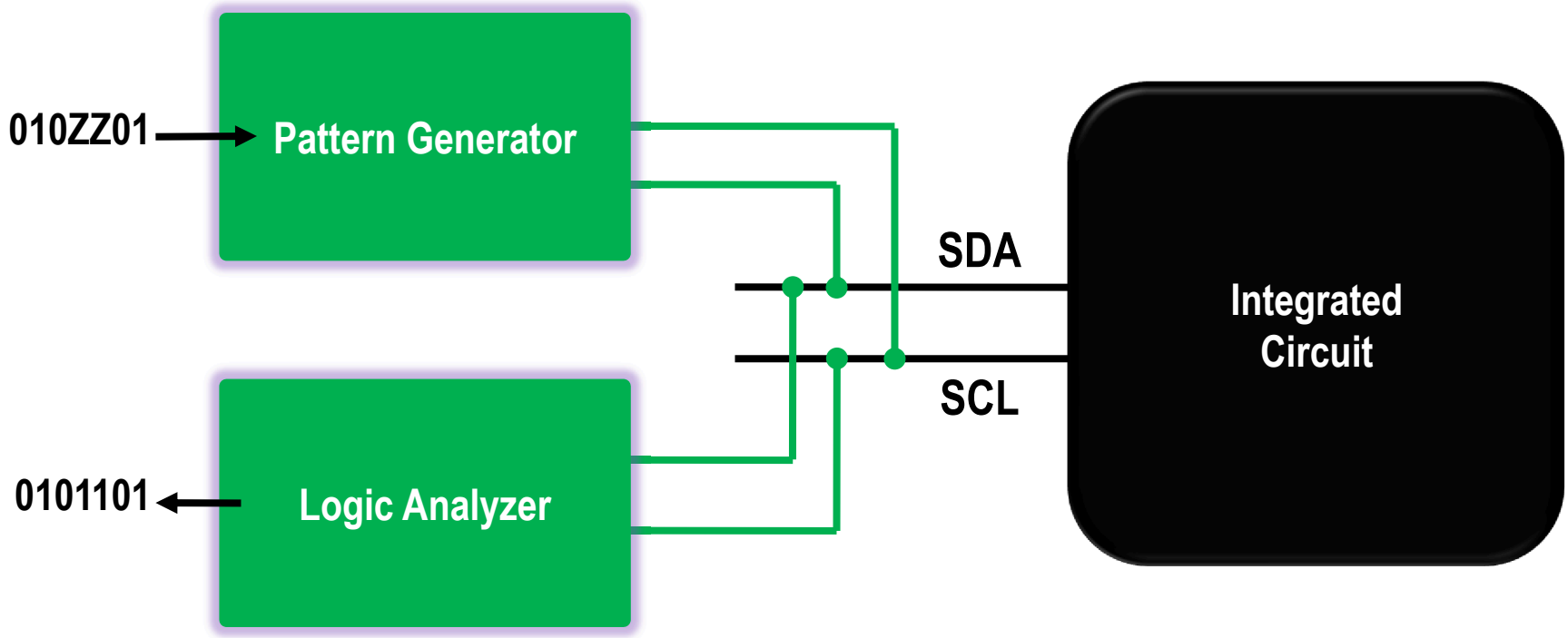


FPGAs in Test Applications

FPGA-enhanced applications



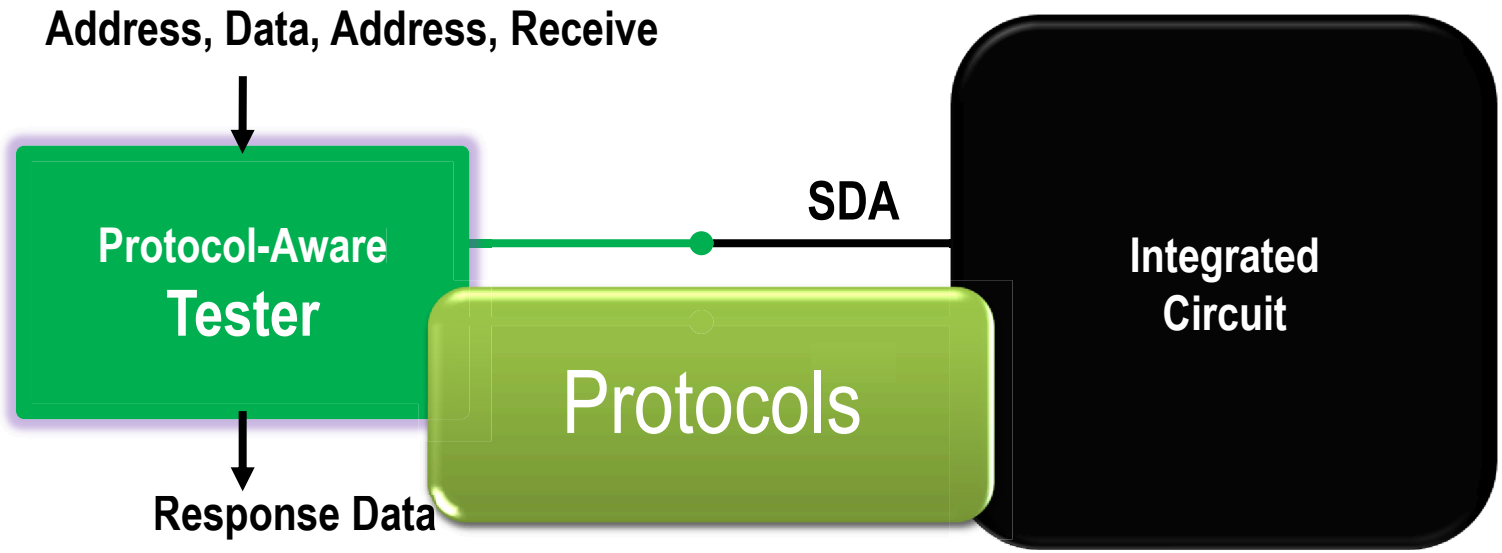
A Simple Digital Protocol: I²C



Traditional Approach

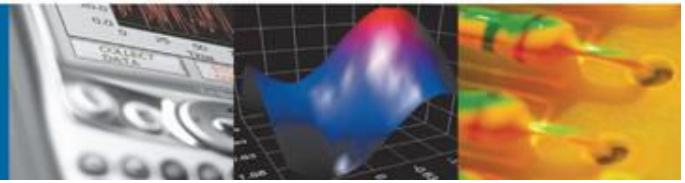
- Static stimulus and expected responses
- Difficult to accommodate multiple clock domains

A Simple Digital Protocol: I²C



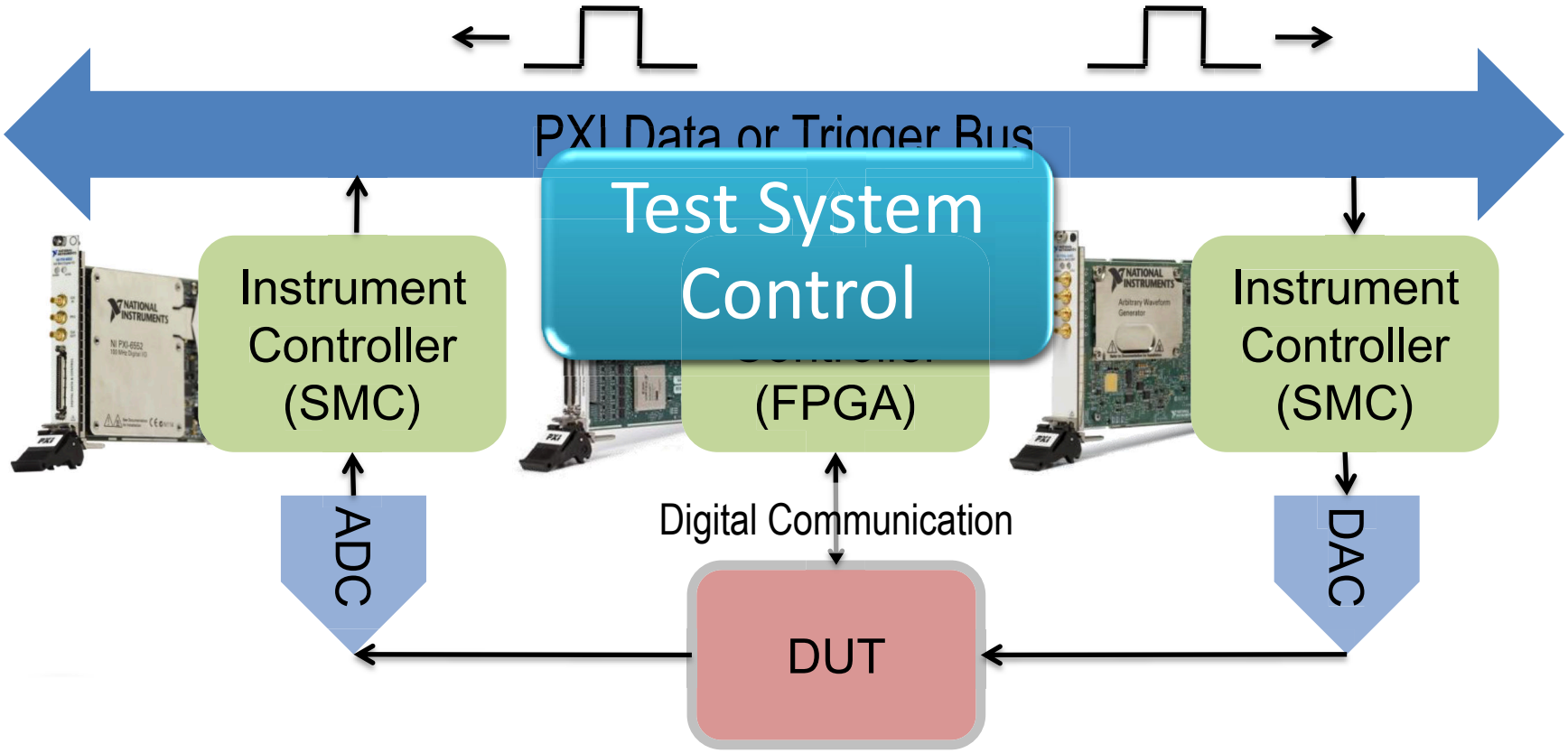
Protocol-Aware Approach

- Intelligence built into the tester
  - Accommodates wait cycles
  - Easy to cross clock domains
- Test with high-level commands
  - Real-world scenario
  - Inherently easier to program



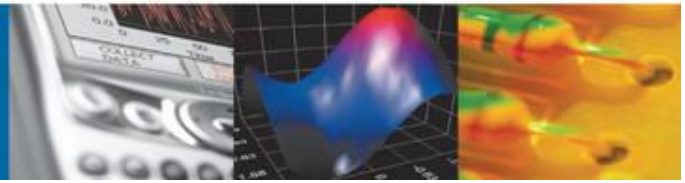
System Control

- Transfer of system timing and decision making from software to hardware

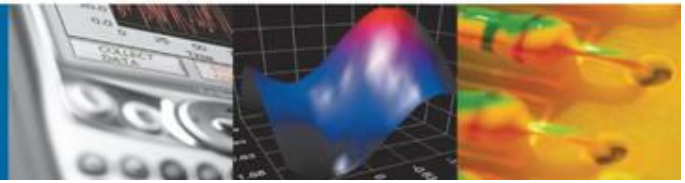


FPGA for Test Application Areas

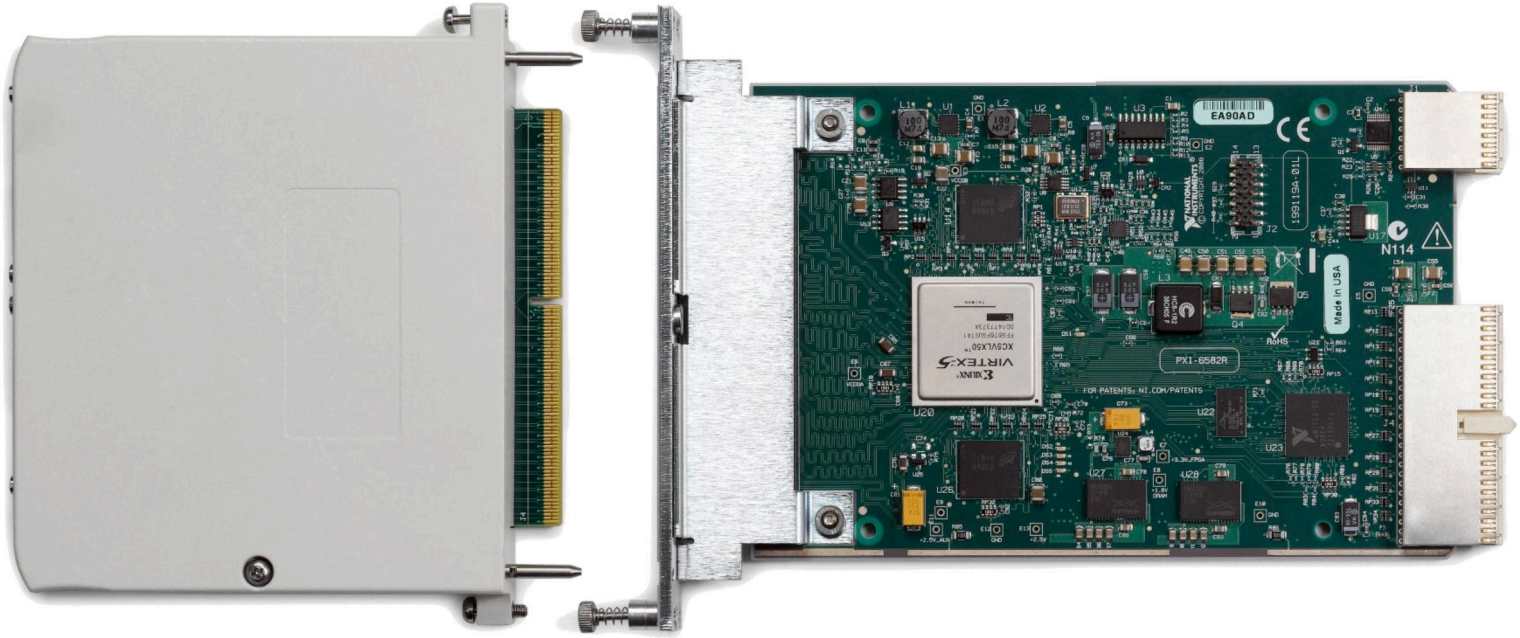
Processing	• Real-time / co-processing • Data reduction / in-line processing
Protocols	• Protocol-aware ATE • Interfacing (digital or modulated)
Closed-Loop Test	• Response-stimulus test • Hardware-in-the-loop (HIL)
Test System Control	• Digital DUT interfacing and command • Complex triggers



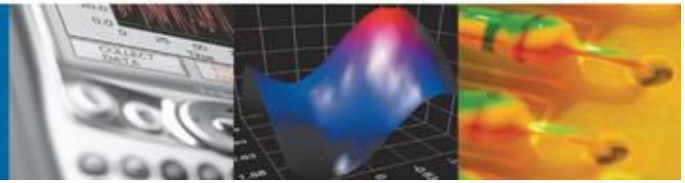
FPGA for Test Hardware



COTS Integration of FPGAs



Example: NI FlexRIO (?)



NI FlexRIO System Architecture

NI FlexRIO Adapter Module

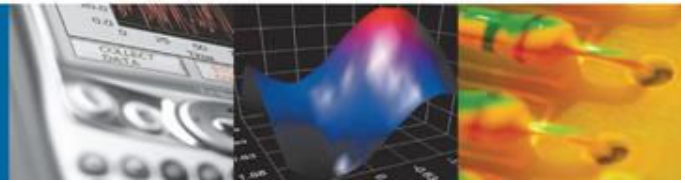
- Interchangeable I/O
- Customizable by users
- Adapter Module Development Kit (MDK)

NI FlexRIO FPGA Module

- Virtex-5 FPGA
- 132 digital I/O lines
- 128 MB of DDR2 DRAM

PXI Platform

- Synchronization
- Clocking/triggers
- Power/cooling
- Data streaming



NI Flex

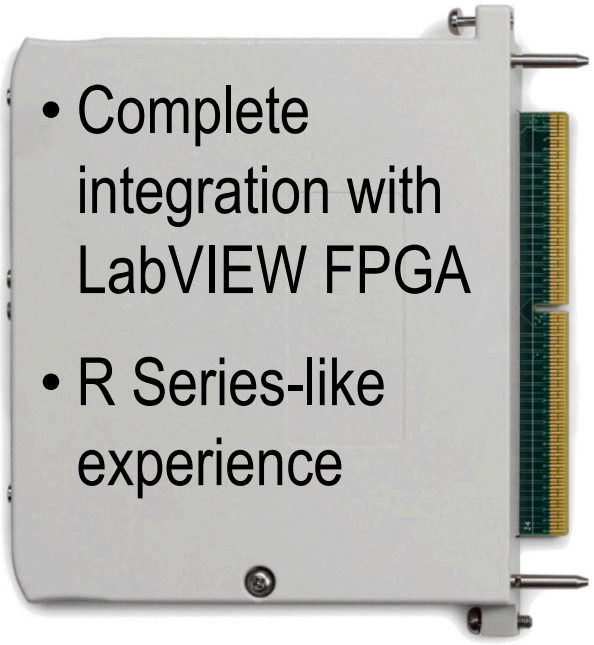
- Virtex-5
  - LX30
- Direct a
  - 132 s
    differ
  - 400 M
  - 1 Gb
- 128 MB
  - 2x 64
  - 800 M
- Adapter

NI FlexRIO Adapter Module

- Card edge connector
- Defines I/O for NI LabVIEW FPGA
- Self identification
- Custom connectivity
- Adapter Module Development Kit (MDK)

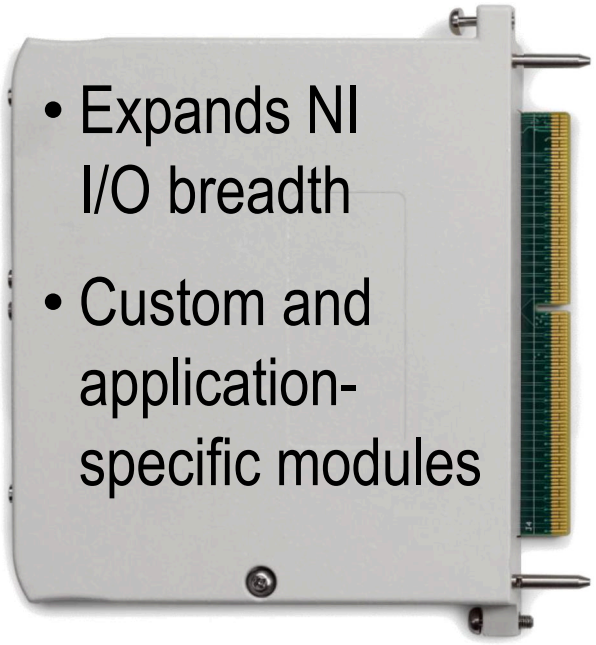
NI FlexRIO Adapter Module Options

NI Modules



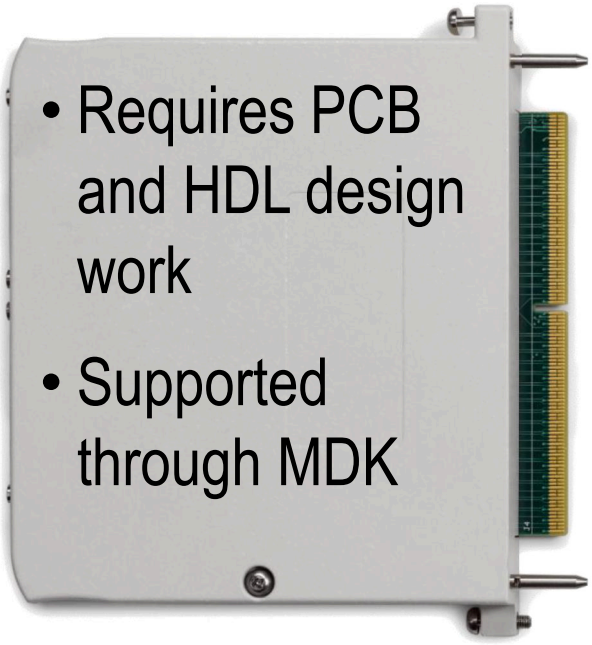
- Complete integration with LabVIEW FPGA
- R Series-like experience

Third-Party Modules

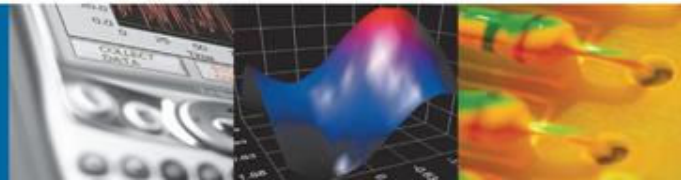


- Expands NI I/O breadth
- Custom and application-specific modules

Custom Modules

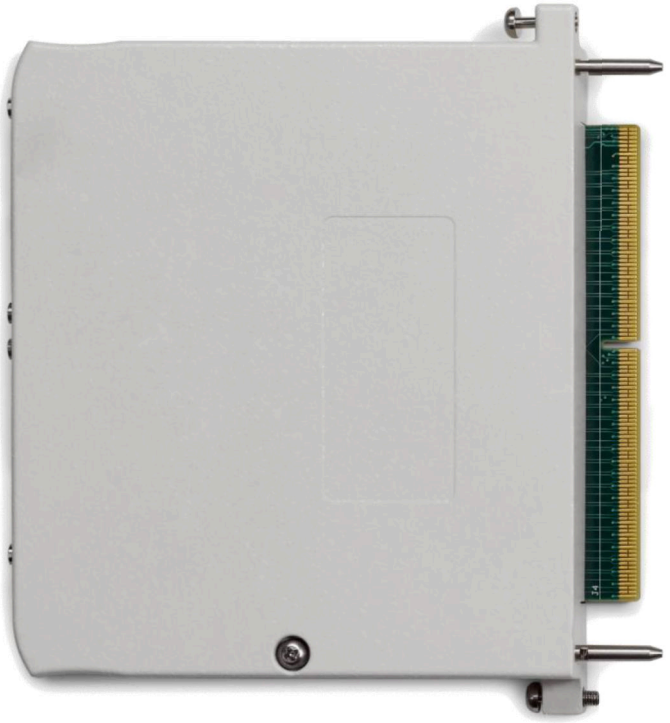


- Requires PCB and HDL design work
- Supported through MDK



NI 6581
High-Speed Digital Adapter Module

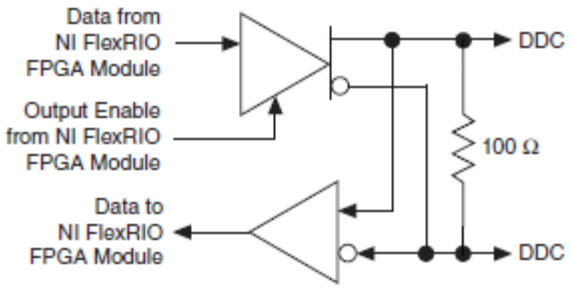
- 100 MHz digital I/O
- 54 single-ended channels
- Selectable voltage levels
  - 1.8, 2.5, 3.3 V (5 V compatible)
- External DIO voltage reference
  - 1.8 to 5.5 V
  - Configurable by connector



NI 6581

NI 6585
200 MHz LVDS Digital Instrument

- 200 MHz digital I/O
- 32 / 42 LVDS channels
- 200 Mbps SDR, 300 Mbps DDR



PXI-6585R

NI FlexRIO Partner Modules

Prevas

- Gigabit Ethernet interfaces
- MAC and Ethernet frames
- Fault-injection software

ADSYS CONTROLS INC.

- Camera Link Interface
- High-speed image processing
- Low-latency control

Averna

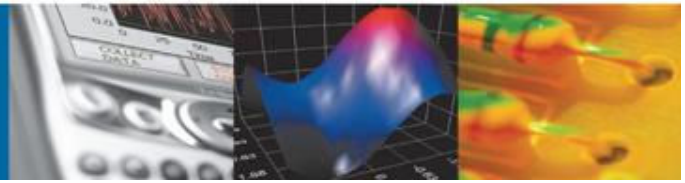
- IEEE-1394b interface
- 3 ports at 800 Mbps

NexFrontier

- 100 MHz vector digital I/O
- 8 ch. per-pin PMU

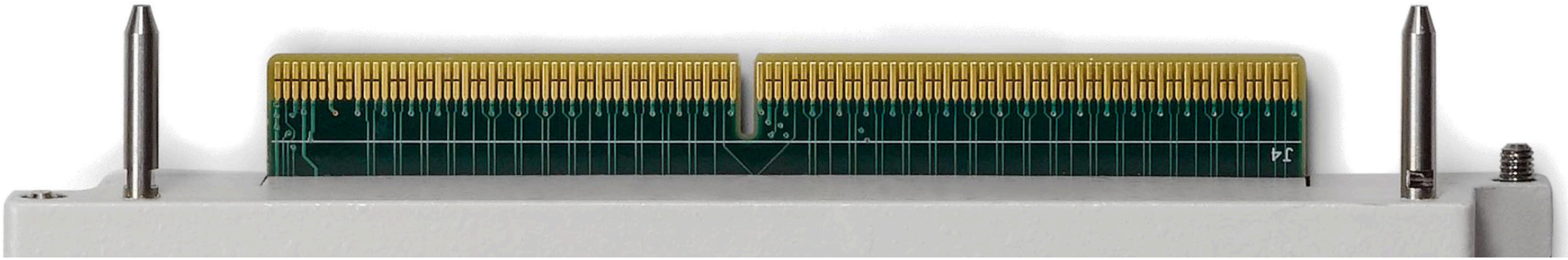
SET

cosylab CONTROL SYSTEM LABORATORY

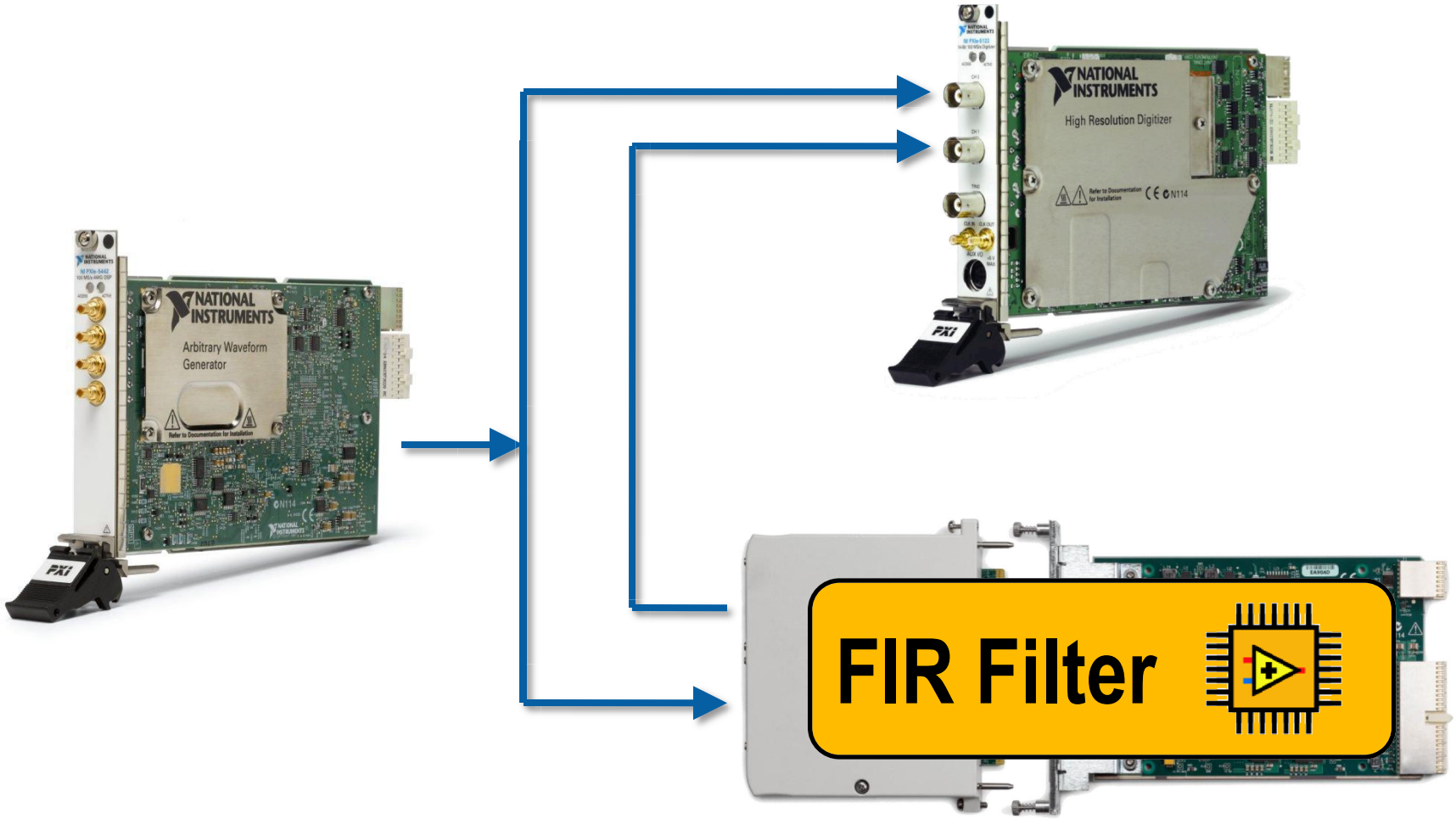


Custom NI FlexRIO Adapter Module Development Kit (MDK)

- 6 W Power – electrical and thermal limit
- 3.3 V (1 A) and 12 V (200 mA) rails
- LVTTL (3.3 V), LVCMOS (1.2, 1.5, 1.8, 2.5, 3.3 V), LVDCI (1.5, 1.8 V, 2.5, 3.3 V), LVDS (2.5 V)
  - 400 Mbps (Single-Ended), 1 Gbps (Differential)
- I^2C EEPROM for module identification and user-defined storage
- NI Mechanical Enclosures

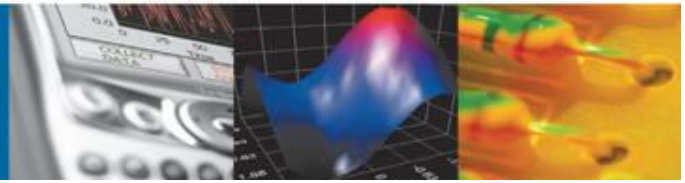


Demo: Digital Filtering



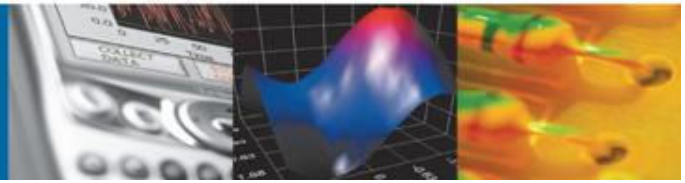
NI PXIe-5641R RIO IF Transceiver

- 2 IF Inputs and 2 IF Outputs
- 14-bit ADCs and DACs
- 20 MHz Instantaneous Bandwidth (25 MS/s I/Q)
- IFs from 250 kHz to 80 MHz
- Xilinx Virtex-5 SX95T LabVIEW-programmable FPGA



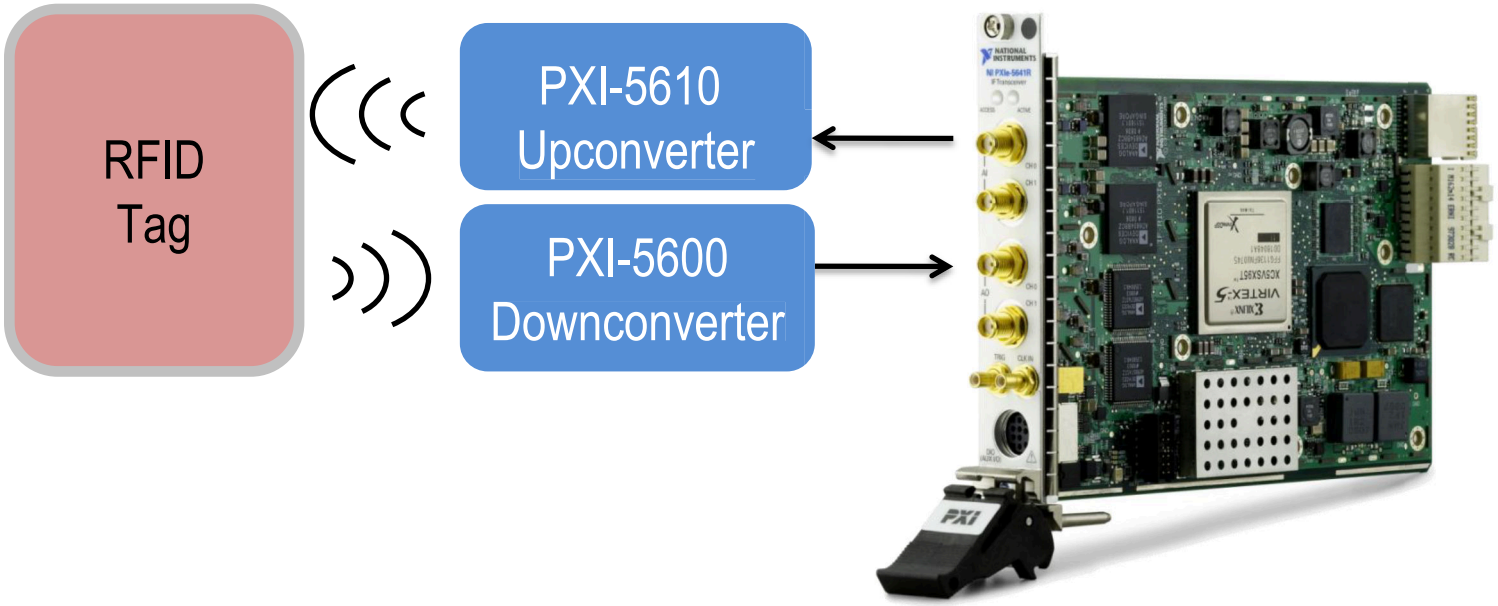
FPGA for Test Hardware

How NI FlexRIO and the PXIe-5641R fit into FPGA for test applications

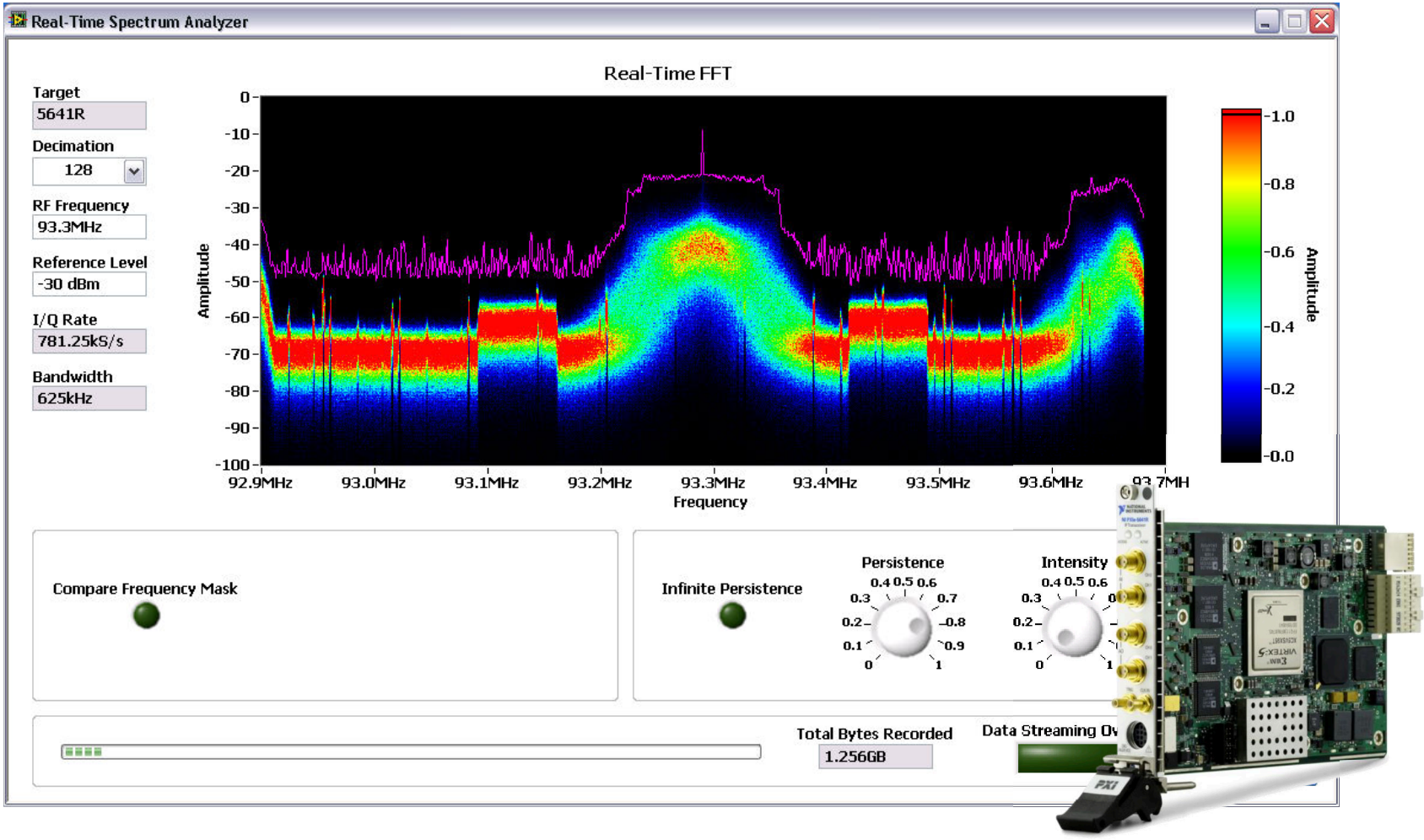


RFID Testing – Response-Stimulus

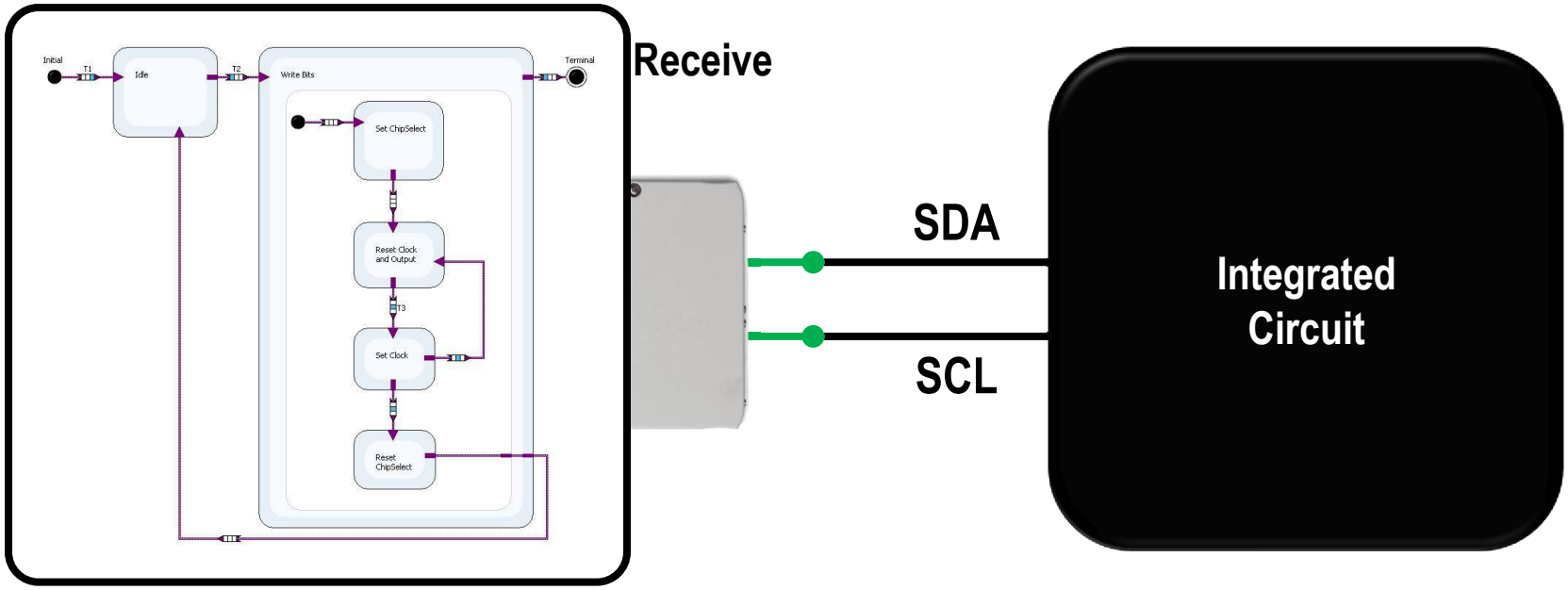
- FPGA on PXIe-5641R IF Transceiver can perform necessary processing
- Upconverter and downconverter condition the signal for the correct RF frequency of the tag



A Real-Time Spectrum Analyzer



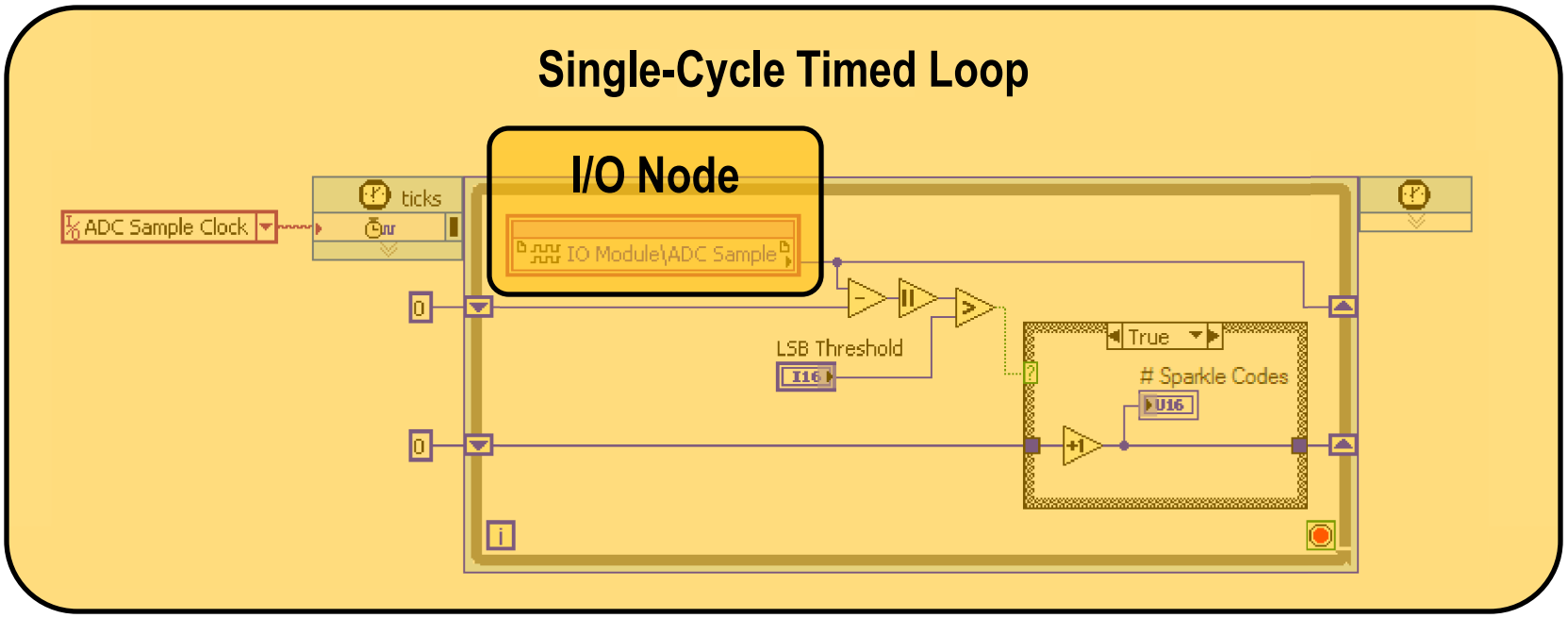
A Simple Digital Example: I²C



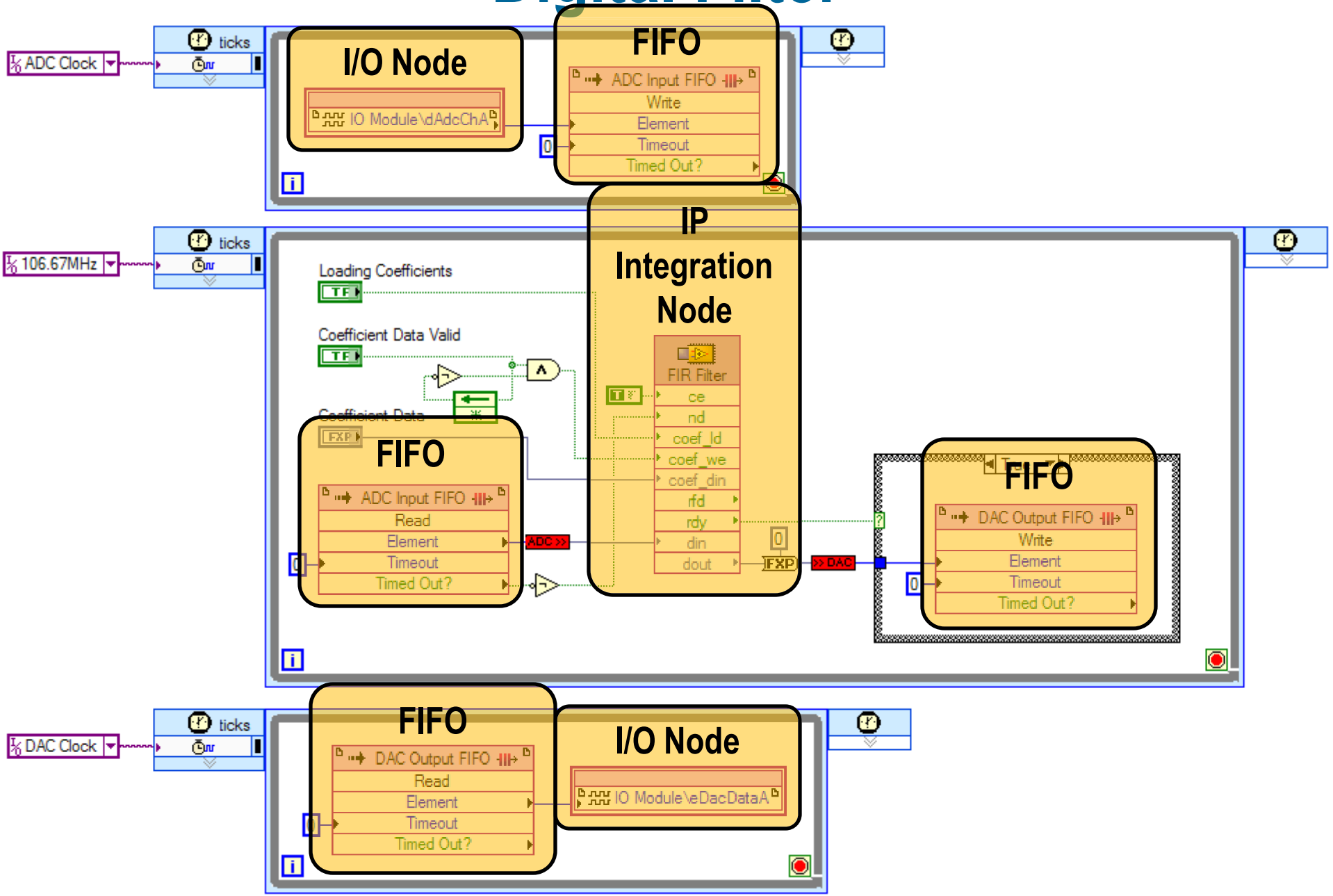
Protocol-Aware Approach

- Intelligence built into the tester
  - Accommodates wait cycles
  - Easy to cross clock domains
- Test with high-level commands
  - Real-world scenario
  - Inherently easier to program

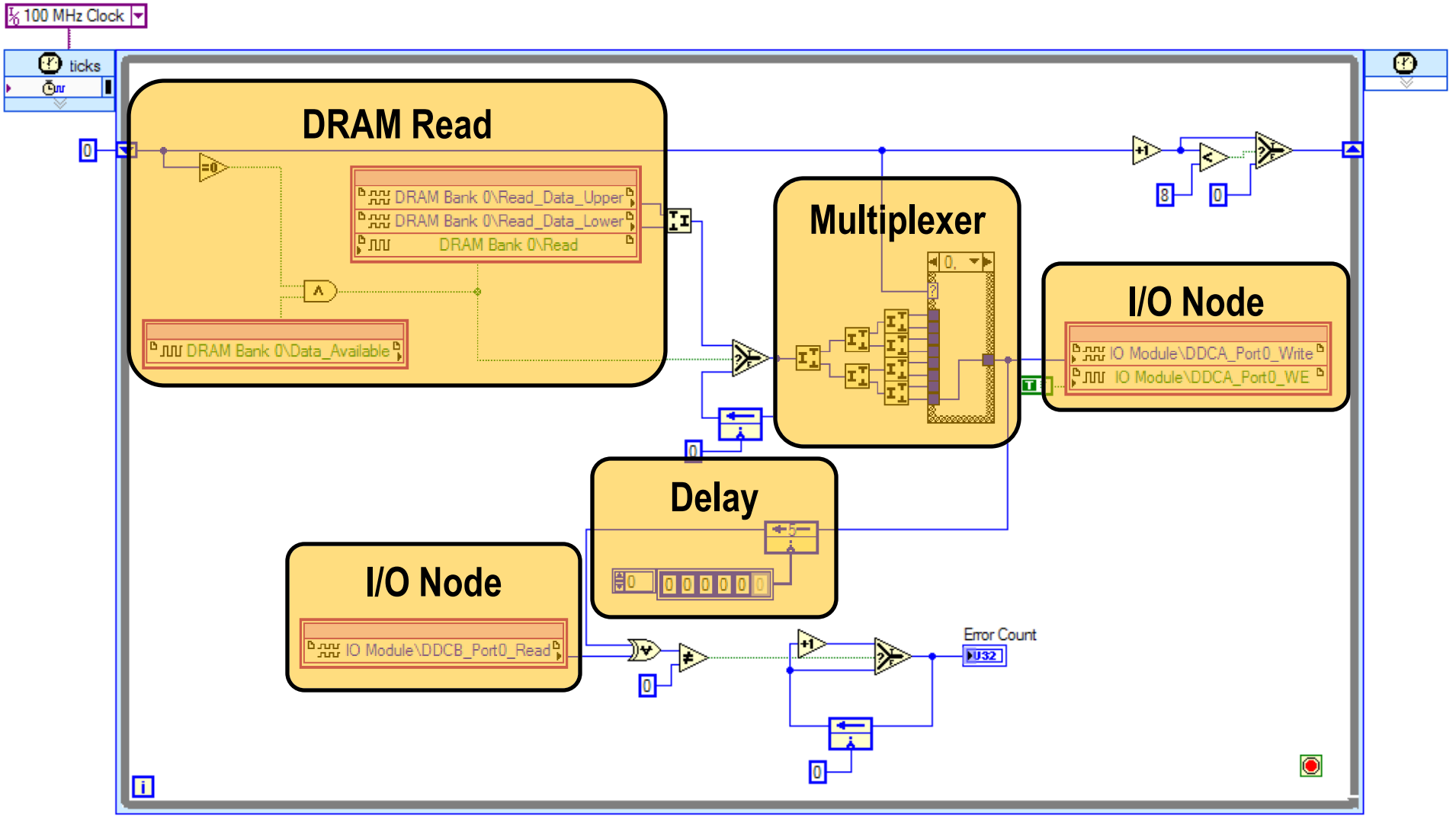
Sparkle Code Counter



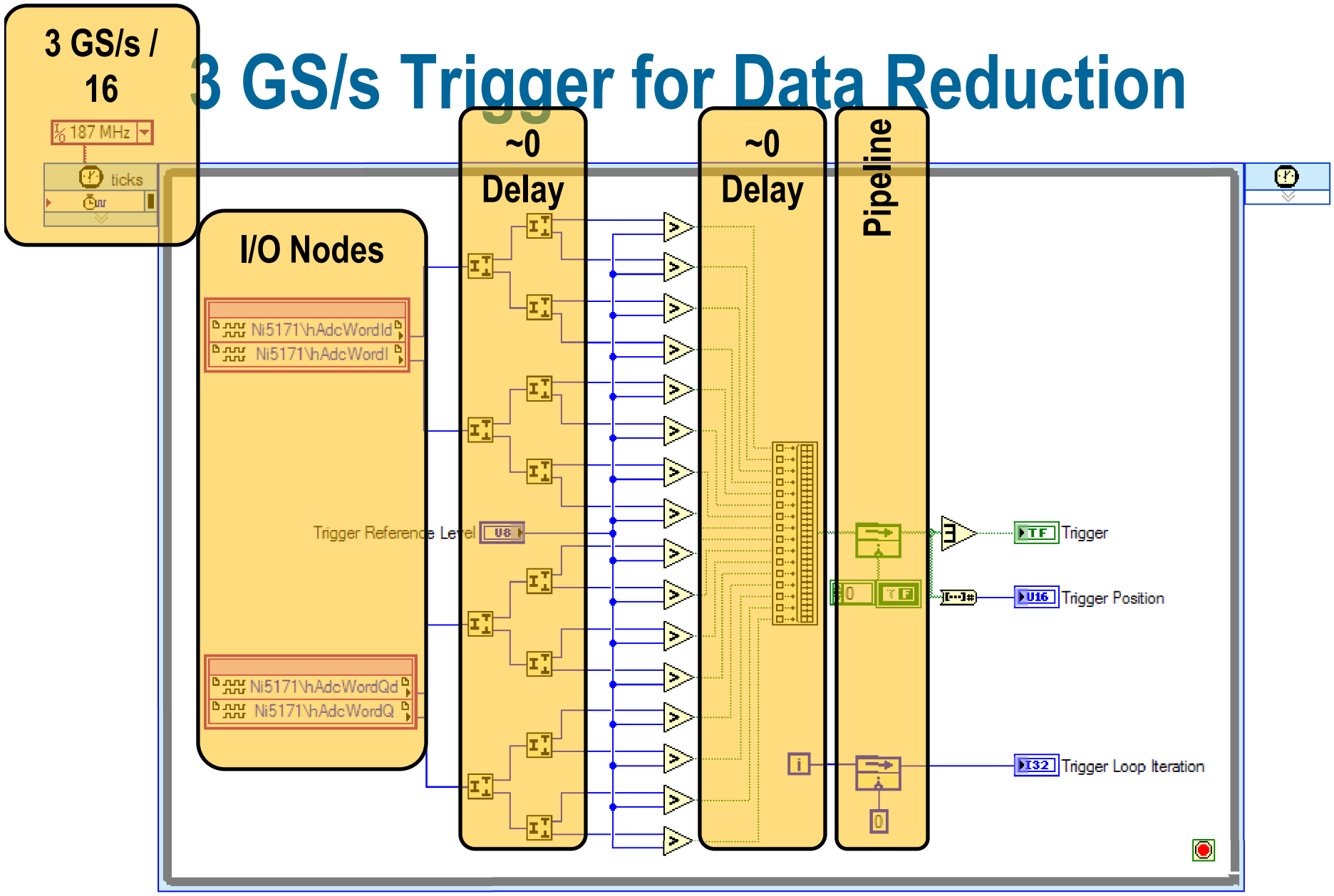
Digital Filter



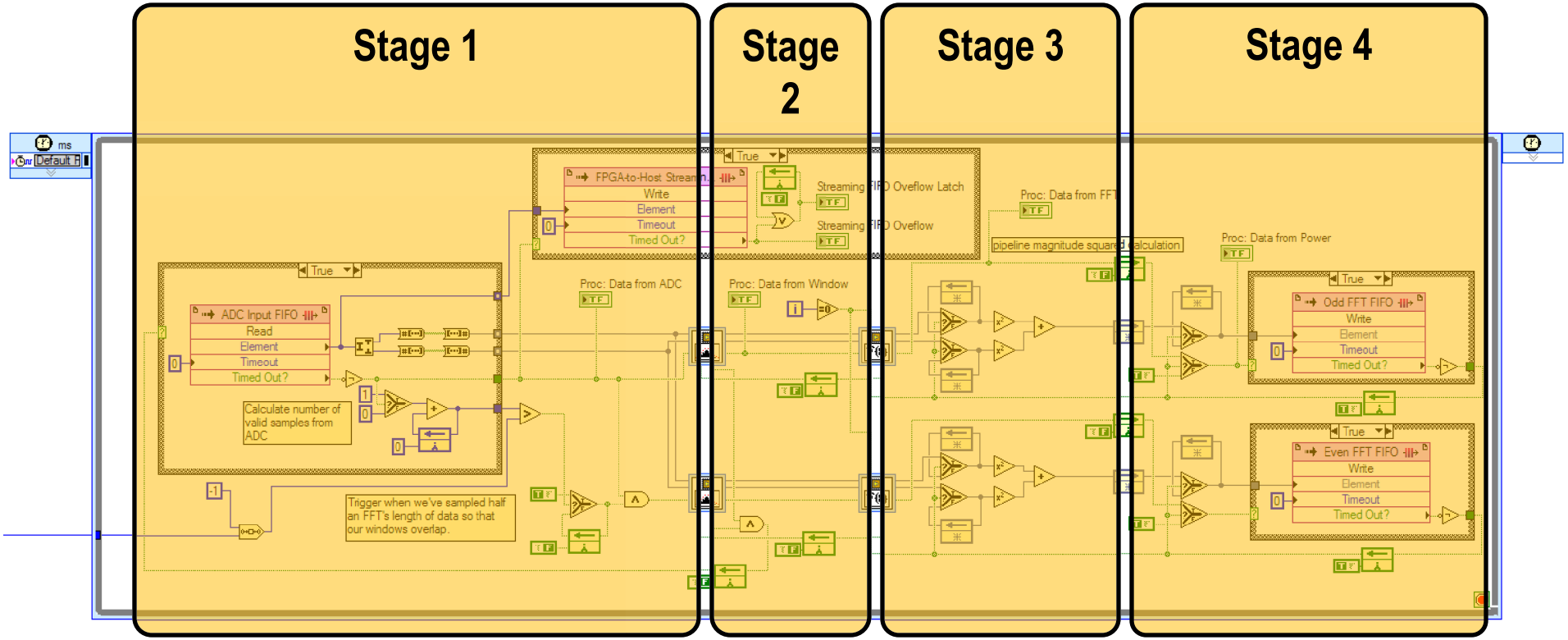
Bit Error Rate Test

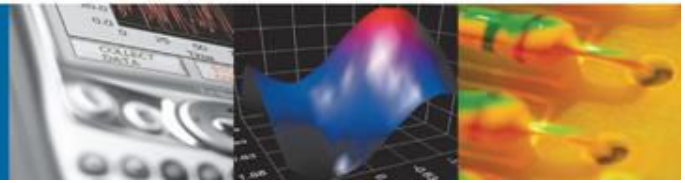


3 GS/s Trigger for Data Reduction



Real-Time Frequency Measurements





Conclusions

- FPGAs enable some types of test applications not previously possible, and make others faster

- LabVIEW FPGA represents a powerful hardware programming paradigm for test engineers and system integrators