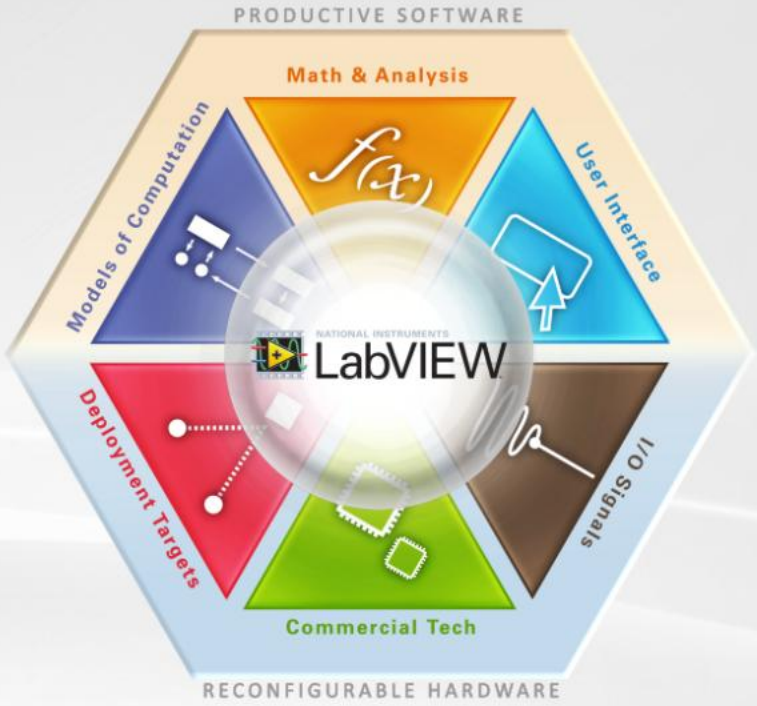
NI Graphical System Design Platform

Productive software and reconfigurable hardware for any system that needs measurement and control.

The power to
innovate - fast.



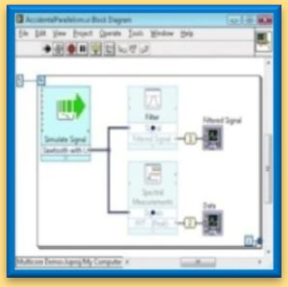
NATIONAL INSTRUMENTS

LabVIEW | Ultimate Embedded Software Design Tool

- Discussion with Chris Ciufo around:
  - Definition of “general embedded”
  - LabVIEW strengths as embedded software tool
  - LabVIEW and NI hardware
  - LabVIEW and non-NI hardware
  - Language characteristics of the LabVIEW editor
    - Including Models of Computation

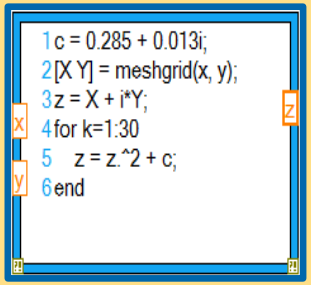
Use the Programming Model of Your Choice Within the LabVIEW Environment

Dataflow

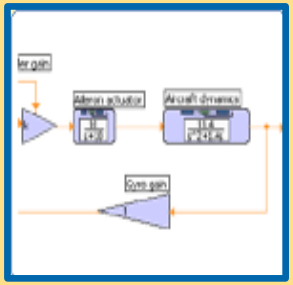


C / HDL Code

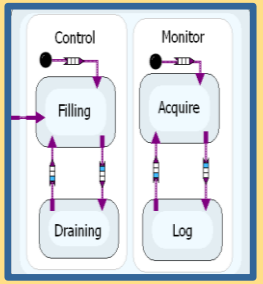
Textual Math



Simulation



Statechart



NATIONAL INSTRUMENTS

LabVIEW™

PCs

PXI Systems

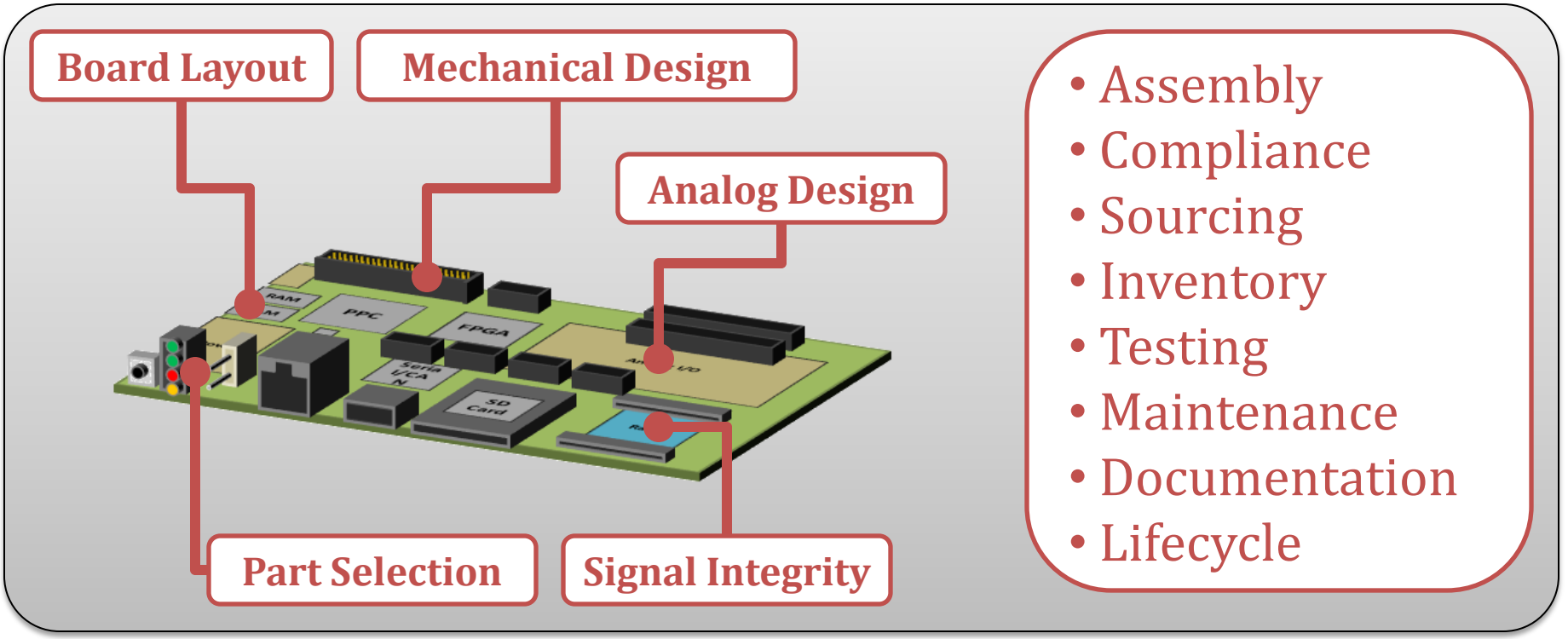
Compact RIO

Single-Board RIO

Custom Design

NATIONAL INSTRUMENTS

NI Hardware | Where We Add Value



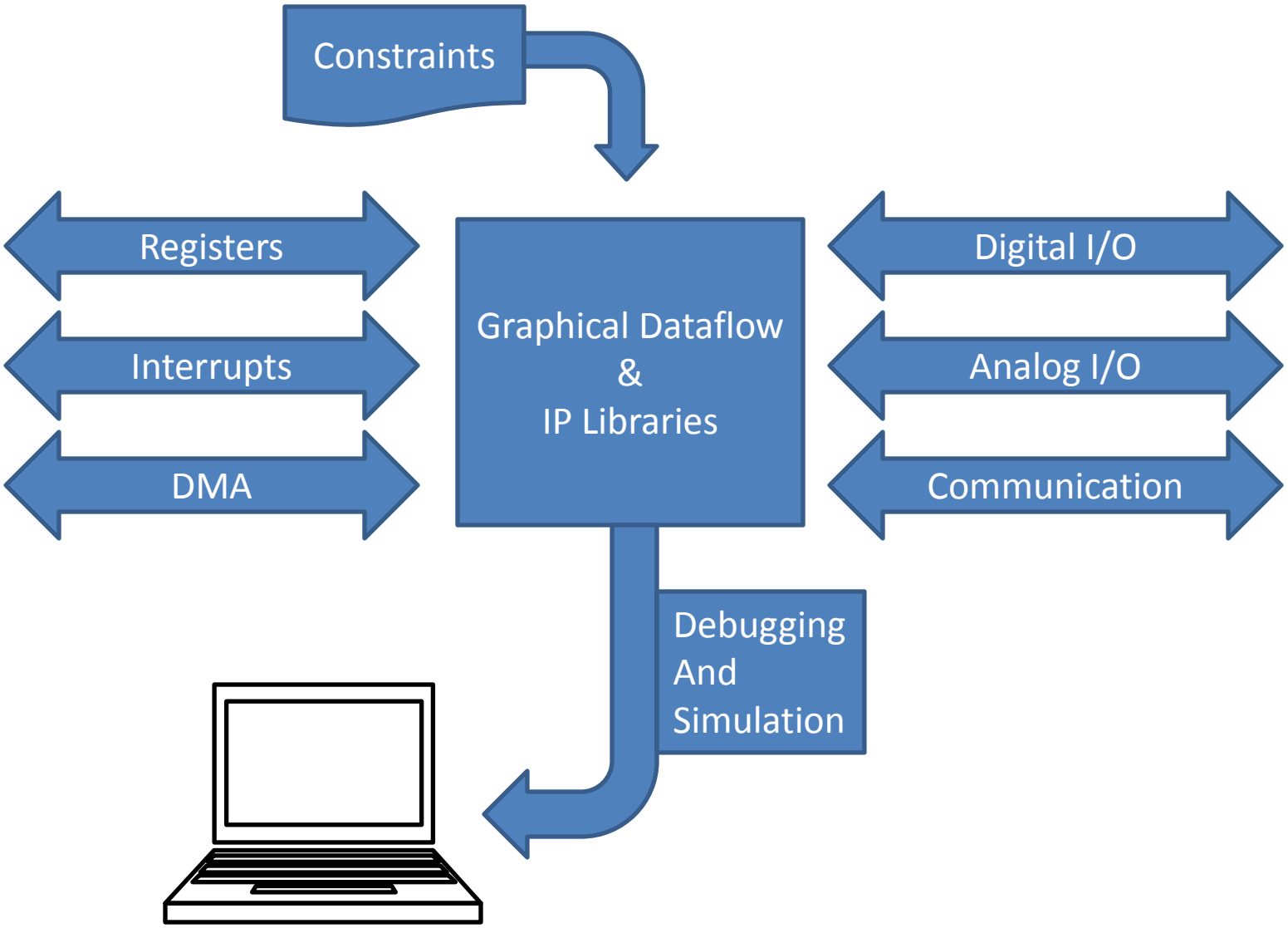
- Assembly
- Compliance
- Sourcing
- Inventory
- Testing
- Maintenance
- Documentation
- Lifecycle

NATIONAL INSTRUMENTS

NI Software | Where We Add Value

NI Software delivers unprecedented productivity on NI hardware because we provide:

- High-quality implementations of real-time drivers (network chipsets, PCI/x, USB, video, storage and so on...)
- I/O drivers (RIO, NI-DAQ ...)
- Firmware (BIOS/boot/safe mode)
- System configuration/mgt
- Fast debug cycle, automatic code download and remote debug (huge value here...)
- Remote step over, into, profiling, and rich UI with interactive front panel
- IP blocks targeted at real-time, embedded and control
- Intrinsic graphical qualities of LabVIEW the language

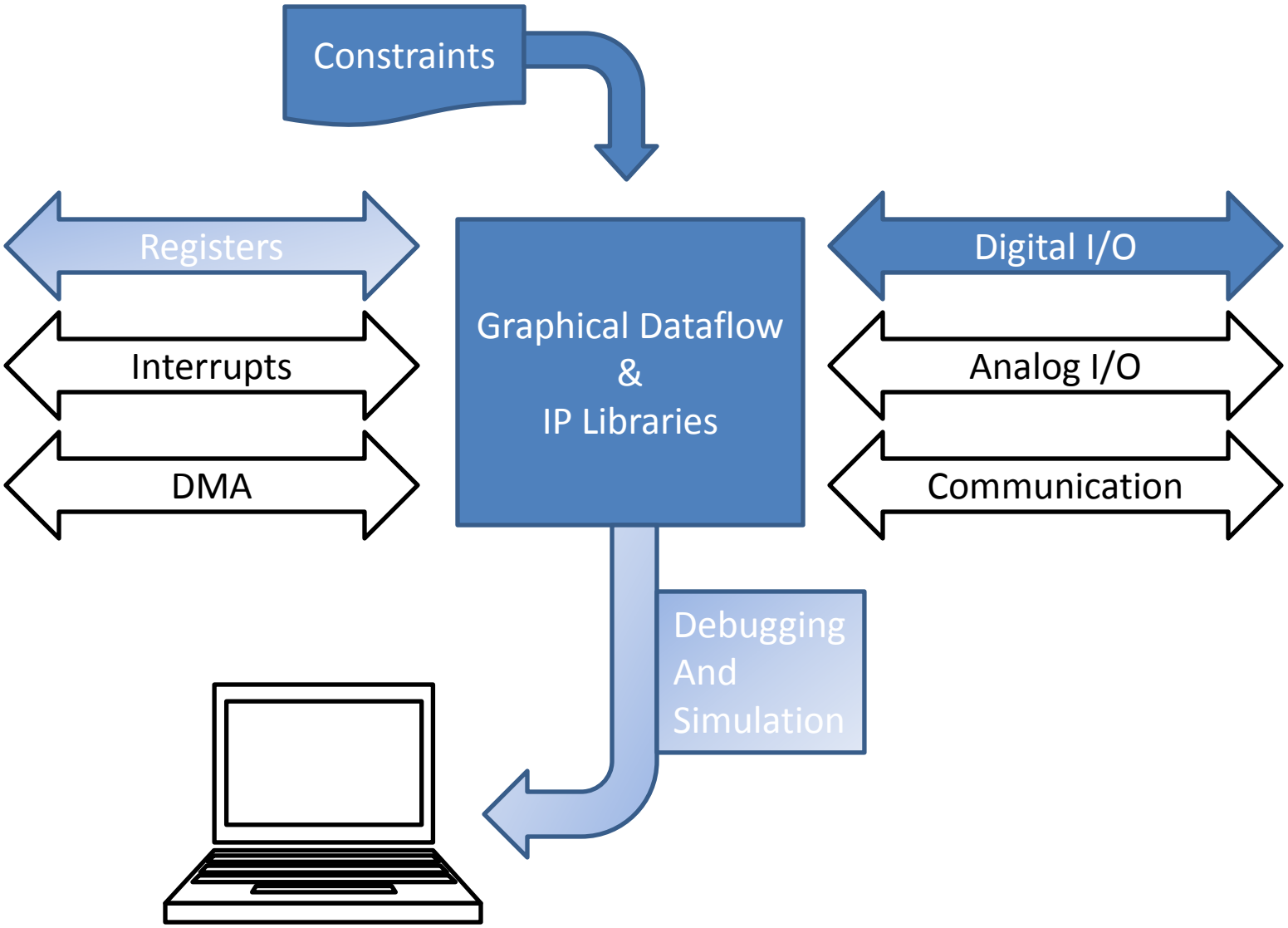


Software | With a Known HW Target

NATIONAL INSTRUMENTS

NI software on any hardware:

- ~~high-quality implementations of real-time drivers (network chipsets, PCI/x, USB, video, storage and so on…)~~
- ~~IO drivers (RIO, NI-DAQ …)~~
- ~~firmware (BIOS/boot/safe mode)~~
- ~~system configuration/mgt~~
- ~~fast debug cycle, automatic code download and remote debug (huge value here…)~~
- ~~remote step over, into, profiling, and rich UI with interactive front panel~~
- IP blocks targeted at real-time, embedded and control
- Intrinsic graphical qualities of LabVIEW the language
- One language: run on windows, real-time, and FPGA



Software | With *an Unknown* HW Target

NATIONAL INSTRUMENTS

Non-NI Targets

- C Generator
- Custom FPGA targets
- LabVIEW for Arduino (http://sine.ni.com/nips/cds/view/p/lang/en/nid/209835)
- LabVIEW for ARM
- Specific bundles: NXT, Spartan III

Custom FPGA Target

"What's really cool about this platform [Xilinx FPGA] is we've maintained PXI compatibility so that we can use chassis and accessories from National Instruments to help customers migrate from the lab to get to orbit as quickly as possible…

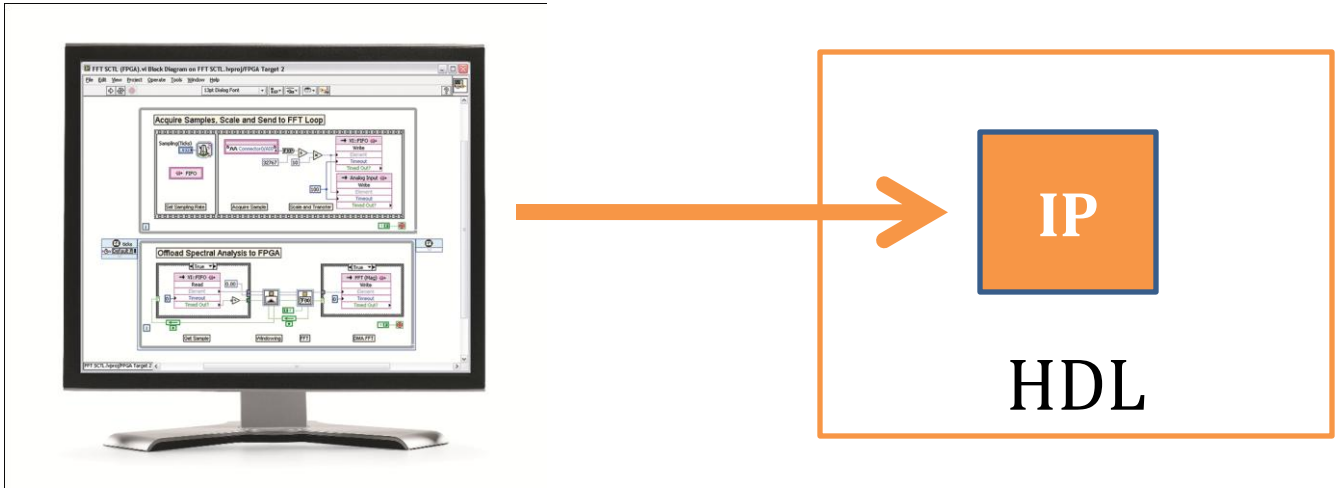
...Honeywell and the National Instruments custom engineering organization have been working very tightly for the last couple of years to bring a customized version of LabVIEW FPGA to the Responsive Digital Electronics board from Honeywell. This platform brings to the customer the ability to build their own flight IP within LabVIEW and the tool flow they're familiar with in the R series, [NI] FlexRIO, or CompactRIO platforms and deploy this to space-breed Honeywell products."

- Thomas Kreider, Honeywell

NATIONAL INSTRUMENTS

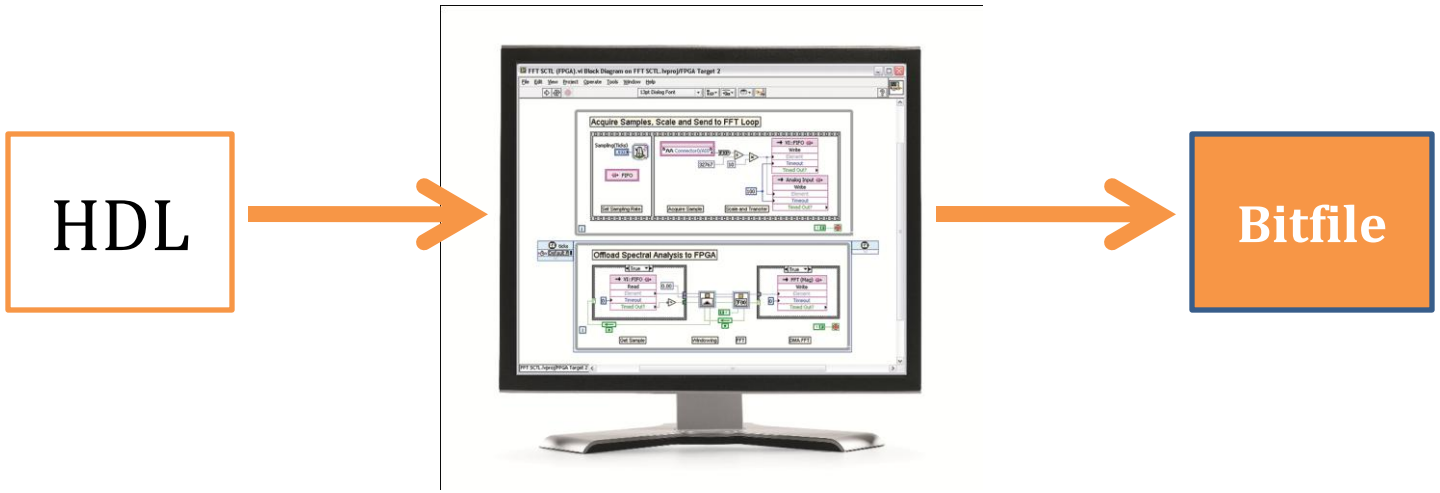
Example: IP Core Generation

Use Case: *HDL is the primary development environment where LabVIEW IP is pulled in as a component*

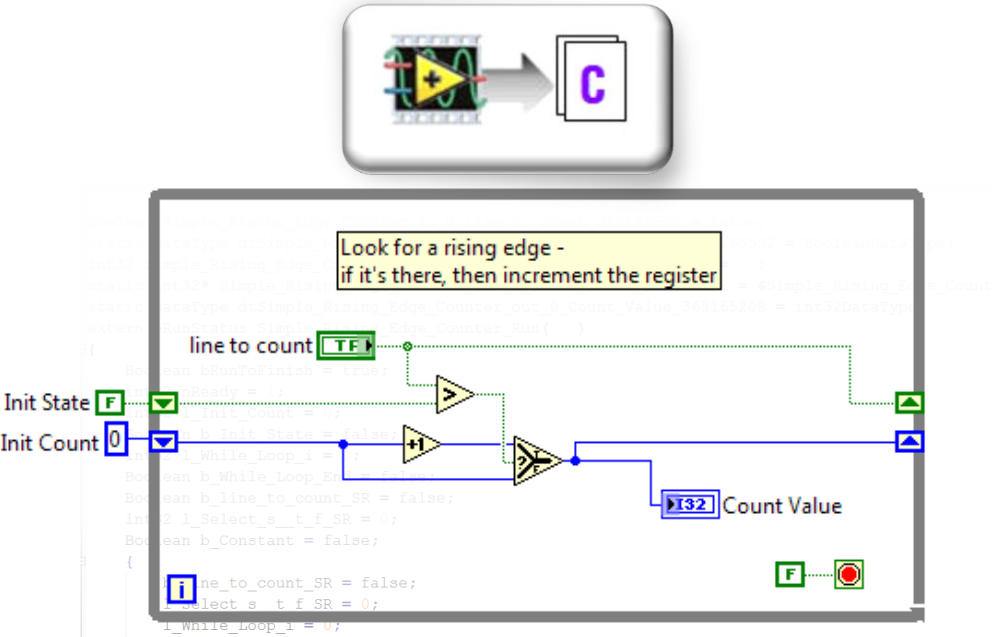


Example: Board Support Package

Use Case: *LabVIEW is the primary development environment where HDL IP may or may not be pulled in as a component*



LabVIEW C Generator

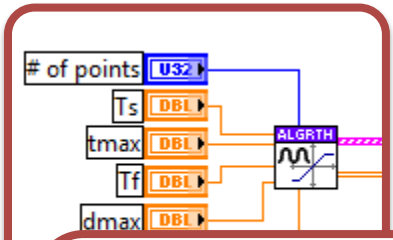



```
do {
    /*****************************************************************************/
    /* Look for a rising edge -  */
    /* if it\'s there, then increment the register */
    /*****************************************************************************/
    {
        Boolean b_line_to_count = false;
        int32 l_Select_s__t_f = 0;
        int32 l_Init_Count_SR = 0;
        Boolean b_Greater__x___y_ = false;
        int32 l_Increment_x_1 = 0;
        Boolean b_Init_State_SR = false;
        b_Init_State_SR = b_line_to_count_SR;
        l_Init_Count_SR = l_Select_s__t_f_SR;
        /**/
        /* Increment */
        /**/
        l_Increment_x_1 = (int32)(l_Init_Count_SR + 1);
        b_line_to_count =Simple_Rising_Edge_Counter_in_0_line_to_count_368165592;
        /**/
        /* Greater? */
        /**/
        b_Greater__x___y_ =  (b_line_to_count > b_Init_State_SR);
        /**/
        /* Select */
```

- Generates **professional C code** from a LabVIEW diagram
  - Correctness
  - Stability
  - Performance
  - Density
  - Readability
- Build the generated code into larger project
- Validate against desktop execution
- Augment/override code generation where necessary

NATIONAL INSTRUMENTS

LabVIEW C Generator Design Flow

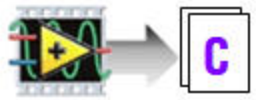


Design

- Design top-down architecture
- Develop algorithm and simulate inputs
- Validate **numeric** accuracy

Prototype

- Connect algorithm to physical signals
- Emulate design on high-performance system
- Validate **functional** accuracy



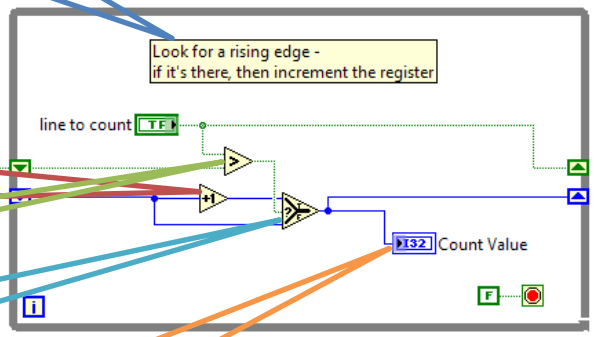
Deploy

- Generate C Functions from SubVIs
- Build using C development toolchain
- Validate **integrated system**

NATIONAL INSTRUMENTS

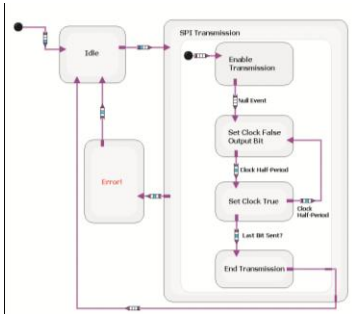
```
/****** Block diagram code **********/

Boolean Simple_Rising_Edge_Counter_in_0_line_to_count_368165592 = false;
static DataType dtSimple_Rising_Edge_Counter_in_0_line_to_count_368165592 = BooleanDataType;
int32 Simple_Rising_Edge_Counter_out_0_Count_Value_368165208_init_ = 0 ;
static int32* Simple_Rising_Edge_Counter_out_0_Count_Value_368165208 = &Simple_Rising_Edge_Counter_out_0_Count_Value_368165208_init_;
static DataType dtSimple_Rising_Edge_Counter_out_0_Count_Value_368165208 = int32DataType;
extern eRunStatus Simple_Rising_Edge_Counter_Run(    )
{
    Boolean bRunToFinish = true;
    int32 nReady = 1;
    int32 l_Init_Count = 0;
    Boolean b_Init_State = false;
    int32 l_While_Loop_i = 0;
    Boolean b_While_Loop_End = false;
    Boolean b_line_to_count_SR = false;
    int32 l_Select_s__t_f_SR = 0;
    Boolean b_Constant = false;
    {
        b_line_to_count_SR = false;
        l_Select_s__t_f_SR = 0;
        l_While_Loop_i = 0;
        do {
            /*****************************************************************************/
            /* Look for a rising edge -  */
            /* if it\'s there, then increment the register */
            /*****************************************************************************/
            {
                Boolean b_line_to_count = false;
                int32 l_Select_s__t_f = 0;
                int32 l_Init_Count_SR = 0;
                Boolean b_Greater__x___y_ = false;
                int32 l_Increment_x_1 = 0;
                Boolean b_Init_State_SR = false;
                b_Init_State_SR = b_line_to_count_SR;
                l_Init_Count_SR = l_Select_s__t_f_SR;
                /**/
                /* Increment */
                /**/
                l_Increment_x_1 = (int32)(l_Init_Count_SR + 1);
                b_line_to_count =Simple_Rising_Edge_Counter_in_0_line_to_count_368165592;
                /**/
                /* Greater? */
                /**/
                b_Greater__x___y_ =  (b_line_to_count > b_Init_State_SR);
                /**/
                /* Select */
                /**/
```

Variable declarations

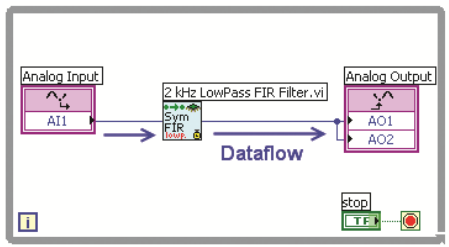


Multiple Development Approaches

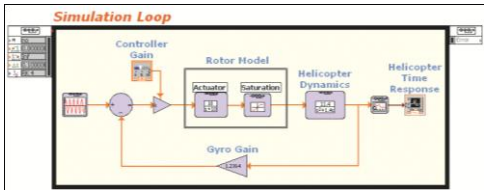
- Algorithm development using three models of computation
- Generate C function from a SubVI



Statechart



Dataflow



Simulation

Supported Features

- Supported features
  - Most LabVIEW structures and functions
  - Analysis library
  - All datatypes
- Excluded features
  - Front panels (properties)
  - On-target debugging
  - Timing and Synchronization
    - Timed Loops, Timers, Wait
  - I/O (Analog, Digital, and Communications)