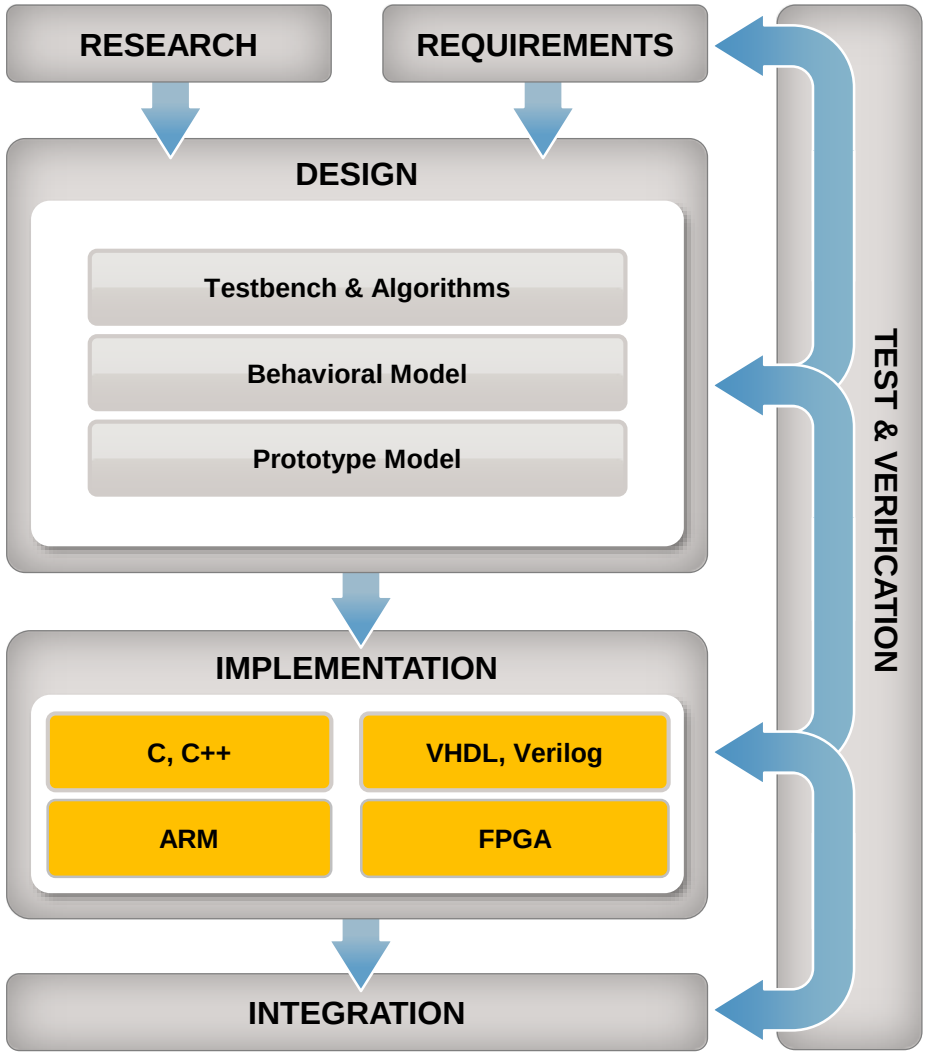
在FPGA上实现MATLAB和Simulink的算法

赵志宏（John Zhao）
全球产品市场部经理

基于模型的设计



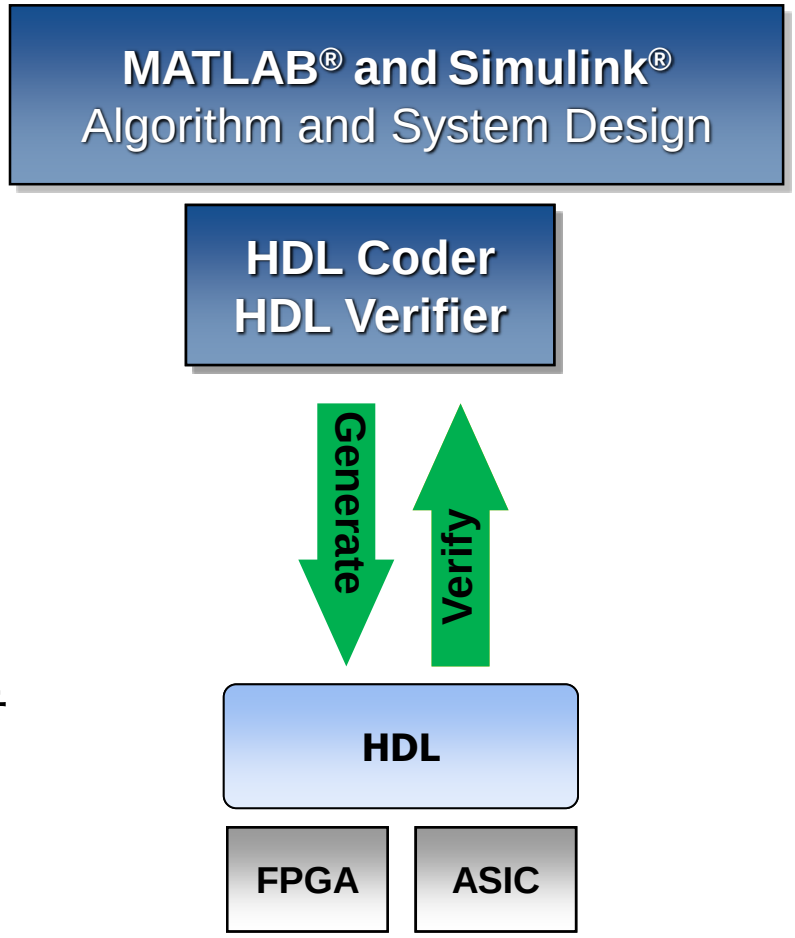
- 快速验证理论和算法的正确性
- 根据实现的要求搭建系统构架

- 直接产生可读的代码
- 快速进行性能和资源的优化

- 复用算法的测试平台和数据
- 支持多种工业测试标准

HDL 代码产生和验证

- 代码产生
  - 可读的VHDL 或 Verilog
  - 为主频和资源进行优化的多种选项
- 验证
  - 产生RTL测试平台
  - 与ModelSim 或 Incisive联合仿真
- 设计自动化
  - 集成了 Xilinx 或 Altera 的综合工具
  - 可对主频和资源进一步优化
  - 支持很多Xilinx and Altera 的开发板卡



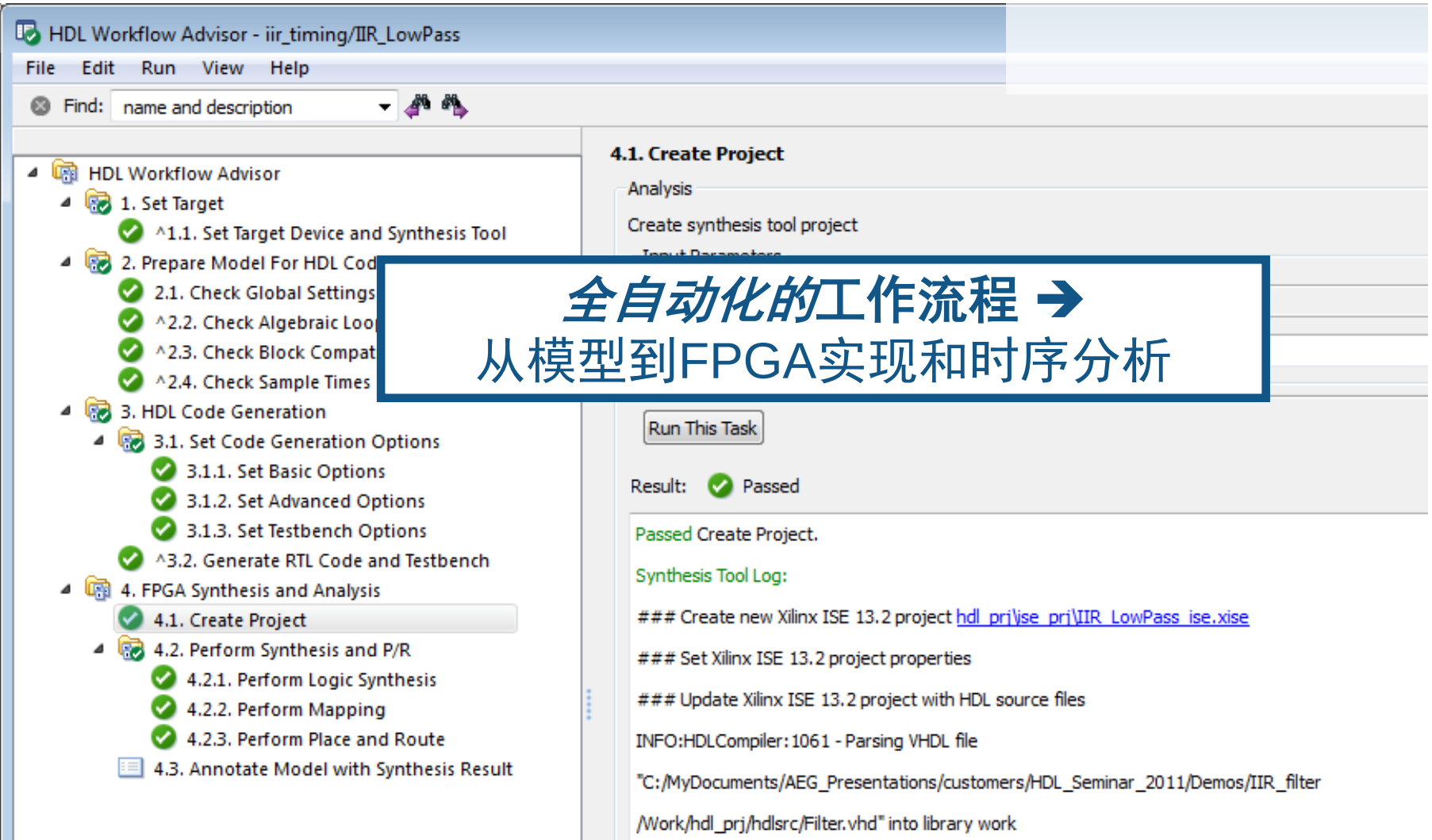
演示实例

请仔细观察

- *怎样用Simulink搭建和仿真你的算法*
- *怎样迅速产生HDL代码*
- *代码可读性如何*
- *怎样迅速把代码综合到Xilinx的芯片上*
- *怎样提高代码的效率*

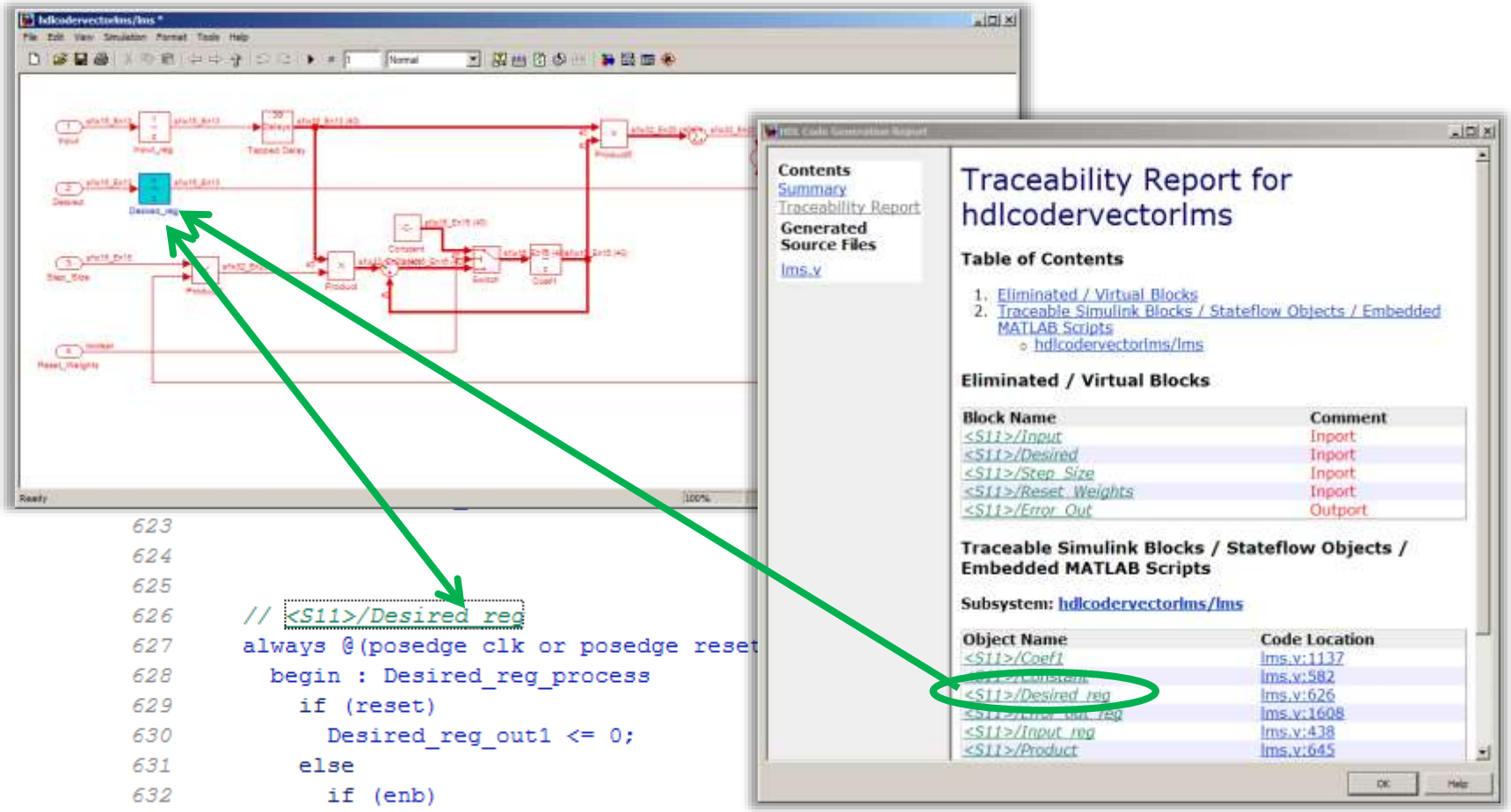
还记得刚刚演示的功能吗？

按钮式工作流程
HDL Workflow Advisor



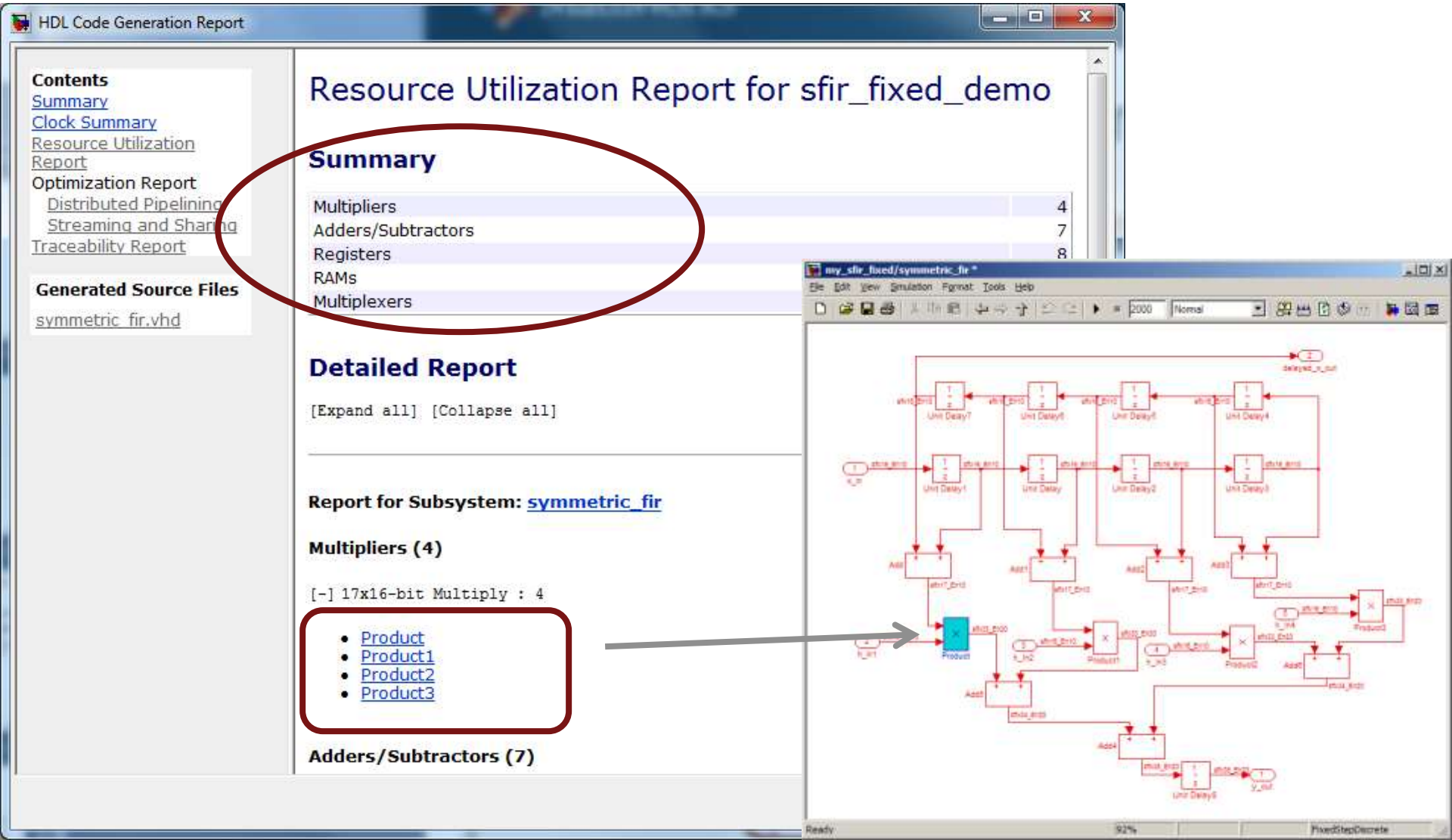
***全自动化的*工作流程 ➔**
从模型到FPGA实现和时序分析

模型和代码的双重追踪性（Traceability）

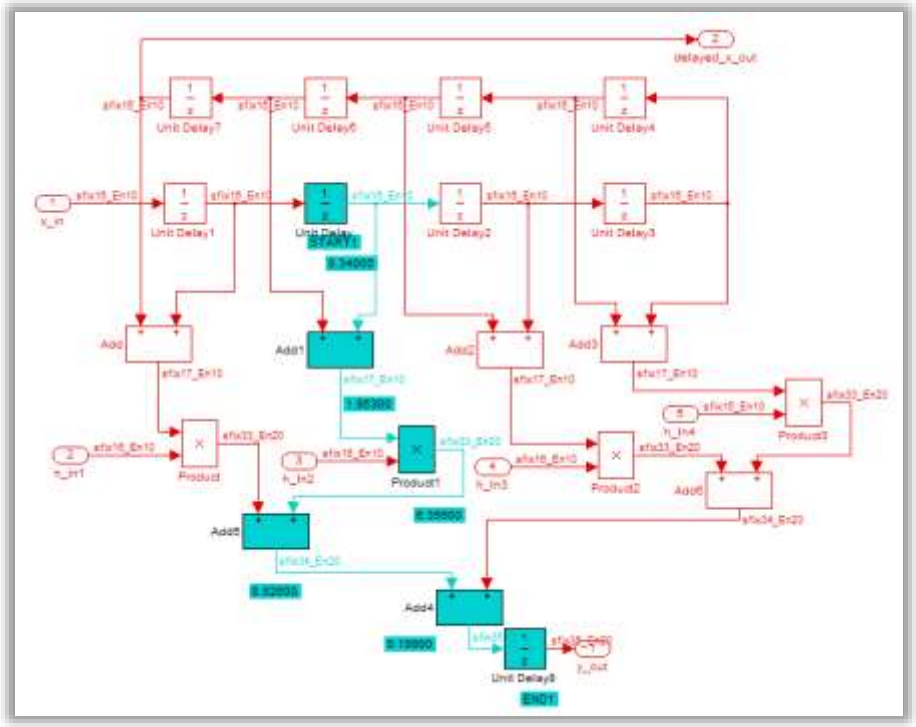


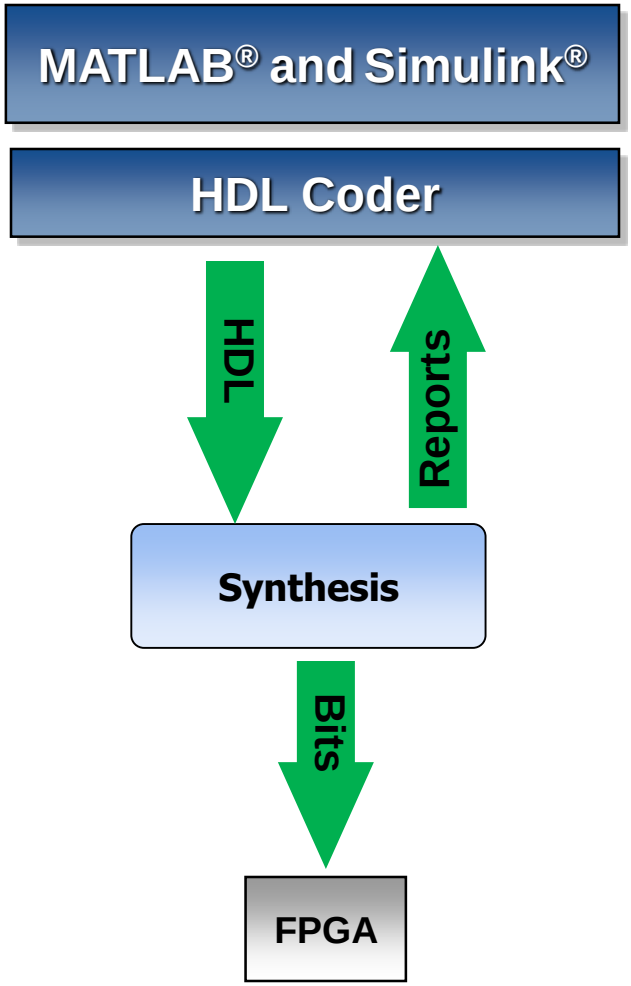

```
623
624
625
626     // <S11>/Desired_reg
627     always @(posedge clk or posedge reset
628       begin : Desired_reg_process
629         if (reset)
630           Desired_reg_out1 <= 0;
631         else
632           if (enb)
633             Desired_reg_out1 <= Desired;
634
635       end
```

资源使用预估


HDL Code Generation Report
Contents
Summary
Clock Summary
Resource Utilization Report
Optimization Report
Distributed Pipelining
Streaming and Sharing
Traceability Report
Generated Source Files
symmetric_fir.vhd
Resource Utilization Report for sfir_fixed_demo
Summary
Multipliers 4
Adders/Subtractors 7
Registers 8
RAMs
Multiplexers
Detailed Report
[Expand all] [Collapse all]
Report for Subsystem: symmetric_fir
Multipliers (4)
[-] 17x16-bit Multiply : 4
Product
Product1
Product2
Product3
Adders/Subtractors (7)

找出关键路径

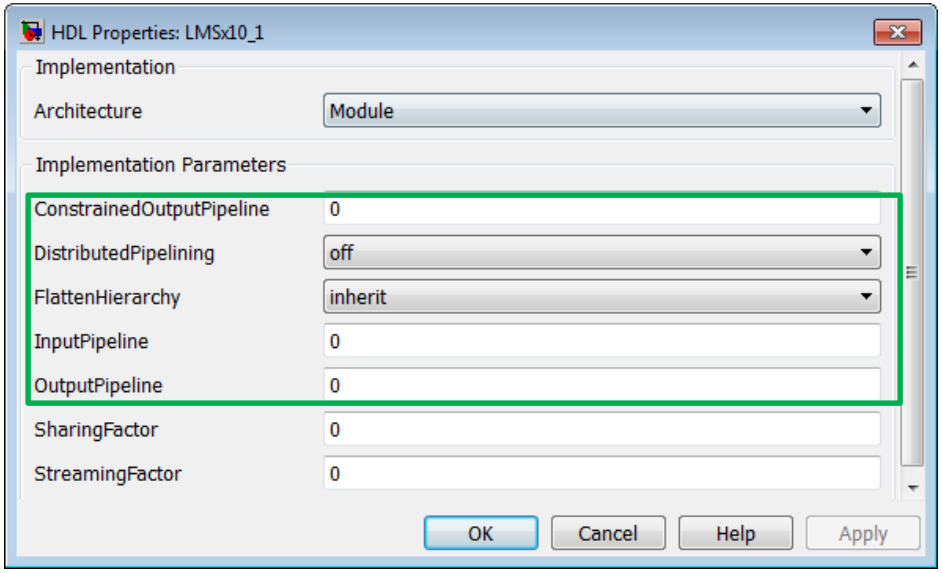




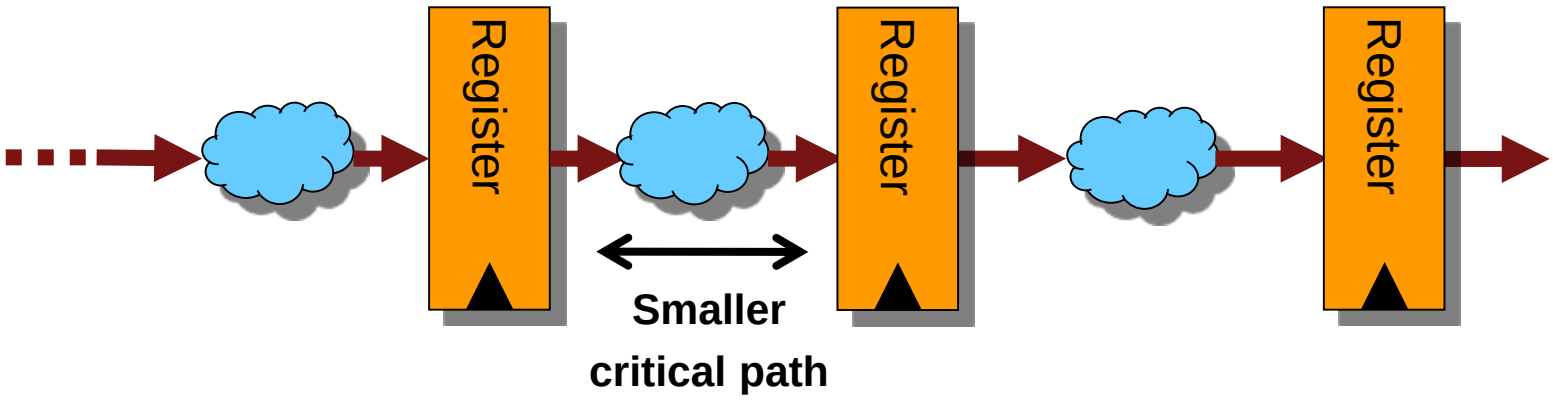
在算法结构中直接看到实现后的关键路径

满足时序约束

分布式管道寄存器（ Distributed Pipelining）

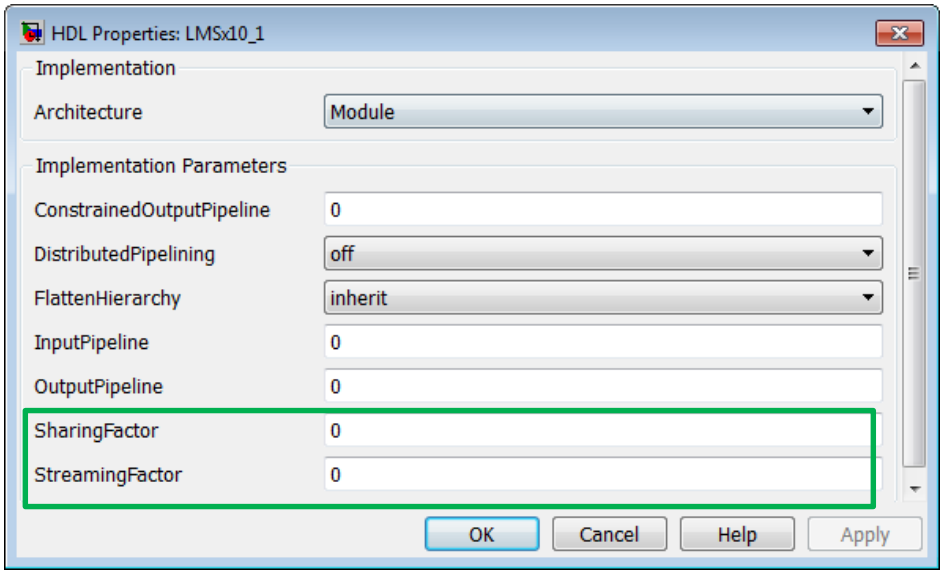


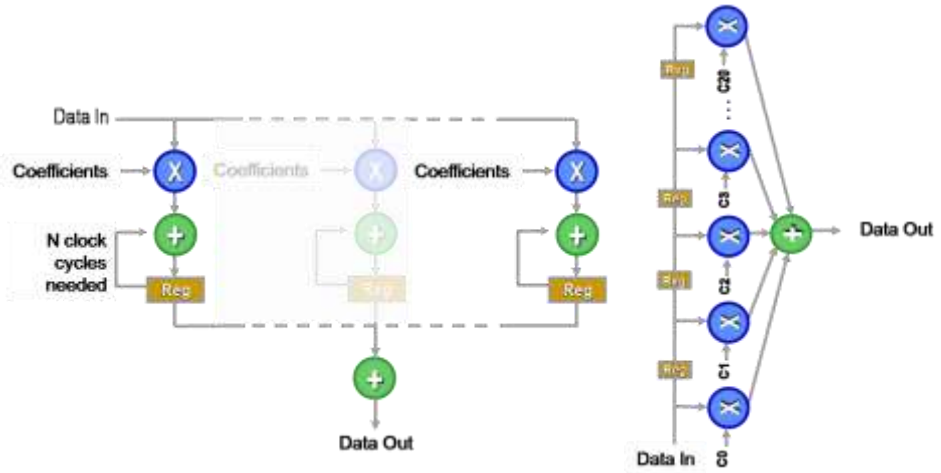
- 分布式管道寄存器 (在模型中重定时)
- 在需要时自动补偿延迟
- 用户可约束式重定时

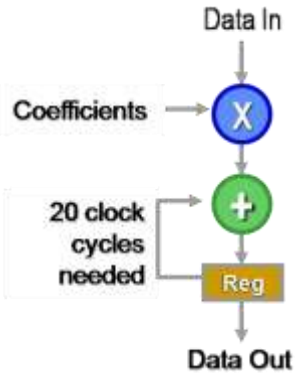


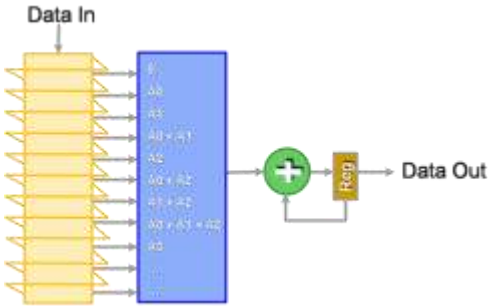
满足资源约束

资源共享 （Resource Sharing）

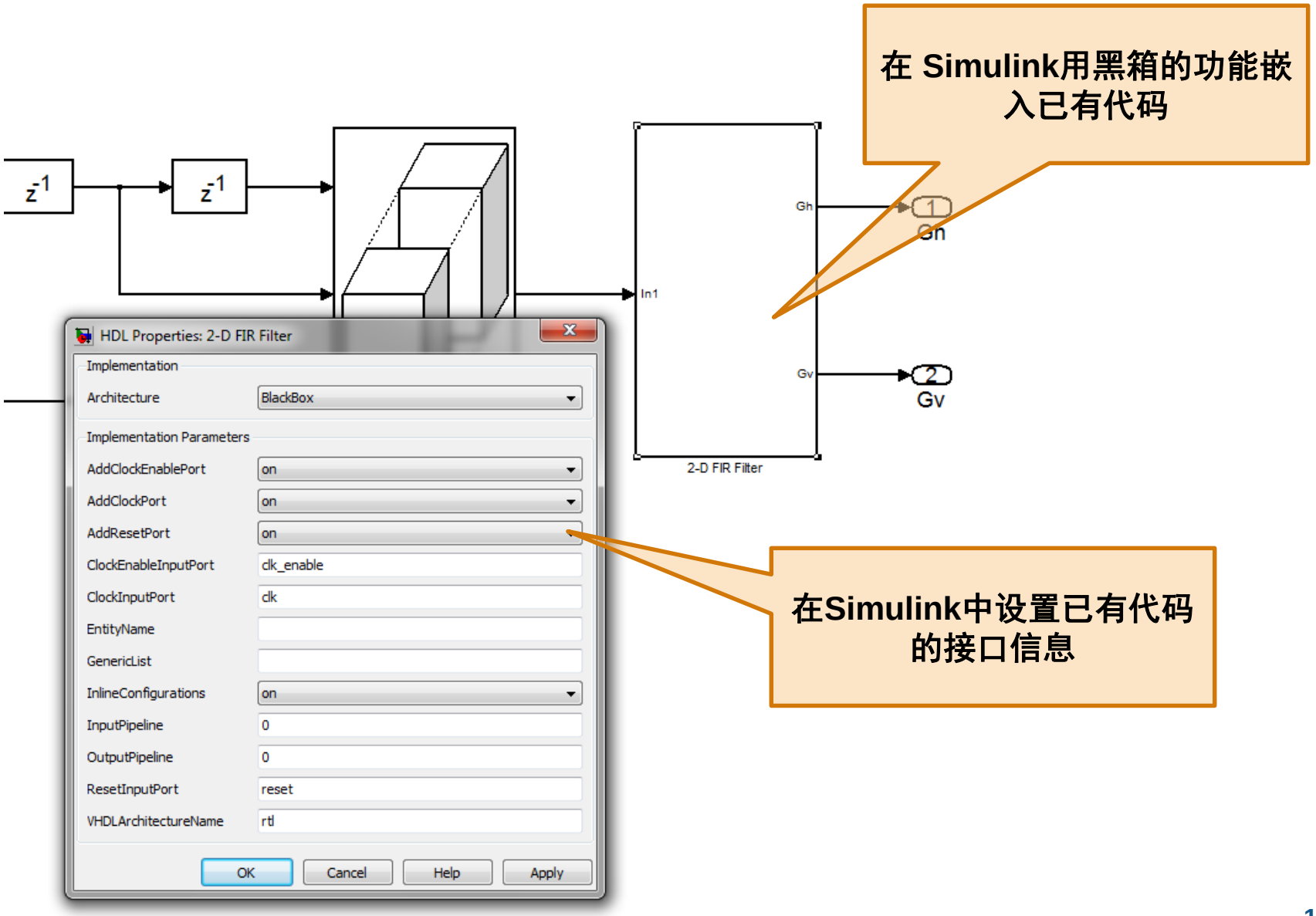








集成已有HDL代码

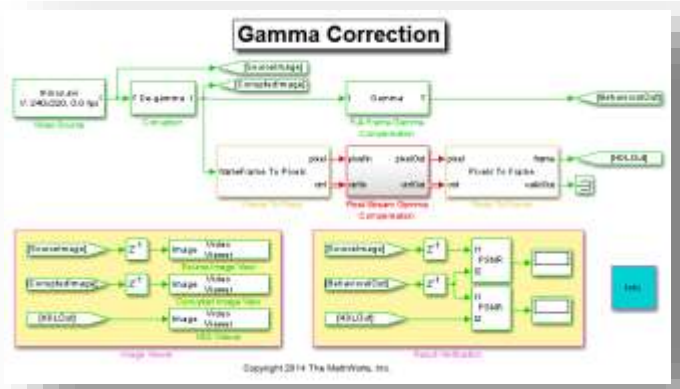

在 Simulink用黑箱的功能嵌入已有代码
z-1
z-1
In1
Gh
Gv
1
Gh
2
Gv
2-D FIR Filter
HDL Properties: 2-D FIR Filter
Implementation
Architecture
BlackBox
Implementation Parameters
AddClockEnablePort
on
AddClockPort
on
AddResetPort
on
ClockEnableInputPort
clk_enable
ClockInputPort
clk
EntityName
GenericList
InlineConfigurations
on
InputPipeline
0
OutputPipeline
0
ResetInputPort
reset
VHDLArchitectureName
rtl
OK
Cancel
Help
Apply
在Simulink中设置已有代码的接口信息

应用高级的算法模块

视觉HDL工具箱（Vision HDL Toolbox）

设计并实现视频图像处理的算法

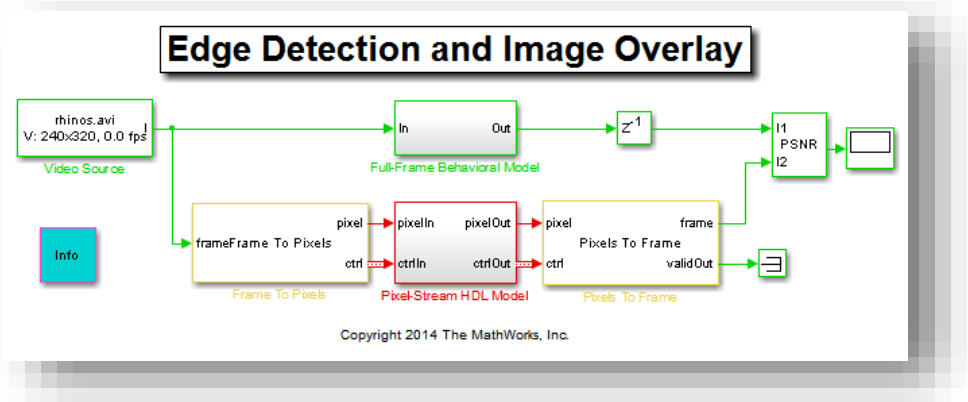
- 模拟算法在实现后的特性
  - 提供基于像素的模块库
  - 提供帧和像素流的自动转换
  - 支持标准的和定制的图像格式



- 在FPGA或SoC上建快速原型
  - (用**HDL Coder**) 可产生高效、可读的HDL代码
  - (用**HDL Verifier**) 进行FPGA在环测试和仿真加速

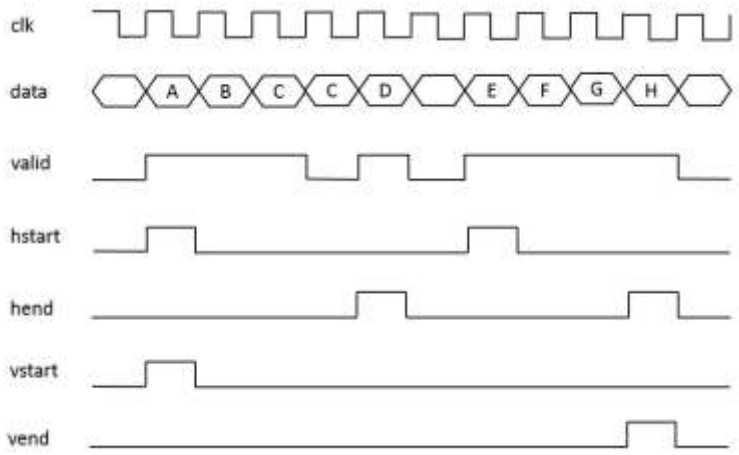
工具箱提供的模块库（基于像素的模块）

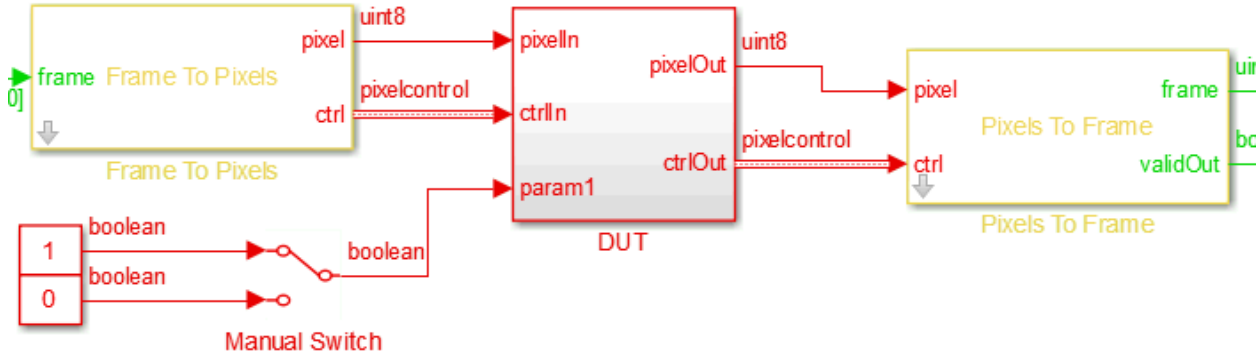
- **图像分析和加强**
  - 边缘检测，中值滤波
- **图像转换器**
  - 色度重采样, 颜色空间转换
  - 去马赛克插补
  - Gamma 校正
- **图像滤波器**
  - 图像滤波器，中值滤波器
- **图像形态运算**
  - 扩张, 侵蚀,
  - 开、闭
- **统计**
  - 直方图
  - 图像统计
- 输入输出接口
  - 帧到像素转换
  - 像素到帧转换
- **其他实用功能**
  - 像素流控制总线产生器
  - 像素流控制总线选择器



建立自己的支持像素流的模块

- 采用文档中解释的像素流控制总线的协议和时序
- 使您的模块的接口和像素流控制总线匹配





其他支持HDL代码产生的高级模块

- 滤波
  - Biquad
  - Interpolator/Decimator
  - LMS
- 无线通讯
  - FFT, NCO
  - QAM, BPSK, QPSK
  - Viterbi, Convolutional, RS, Turbo

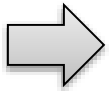
用MATLAB代码编写你自己的模块

```
%#codegen
function [y_out, delayed_xout] = mlhdlc_sfir(x_in,
% Symmetric FIR Filter

% declare and intialize the delay registers
persistent ud1 ud2 ud3 ud4 ud5 ud6 ud7 ud8;
if isempty(ud1)
    ud1 = 0; ud2 = 0; ud3 = 0; ud4 = 0; ud5 = 0; ud
end

% access the previous value of states/registers
a1 = ud1 + ud8; a2 = ud2 + ud7;
a3 = ud3 + ud6; a4 = ud4 + ud5;

% multiplier chain
m1 = h_in1 * a1; m2 = h_in2 * a2;
m3 = h_in3 * a3; m4 = h_in4 * a4;
```

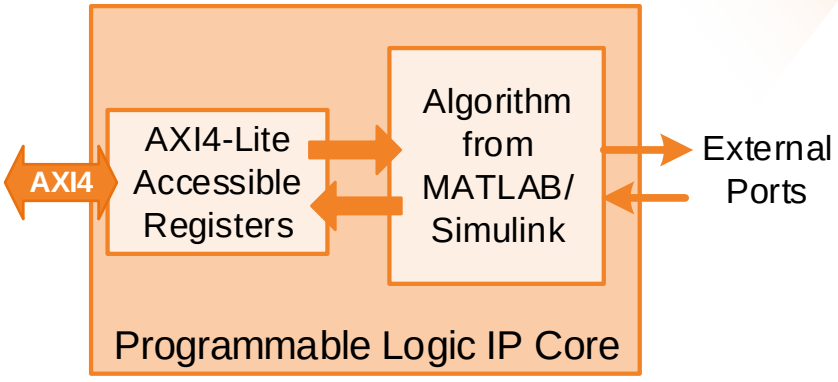


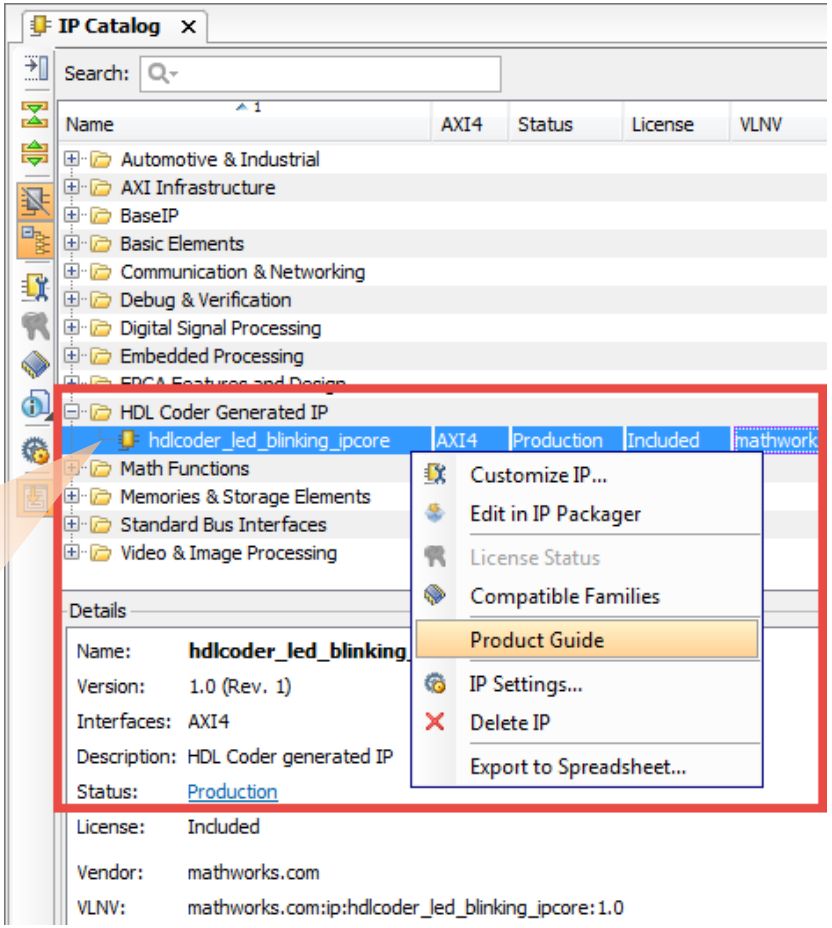
```
always @(posedge clk or posedge reset)
  begin : ud2_reg_process
    if (reset == 1'b1) begin
      ud2_1 <= 0;
    end
    else begin
      if (enb) begin
        ud2_1 <= ud2;
      end
    end
  end


assign tmp_4 = ud2_1;
```

产生Xilinx Vivado IP 核

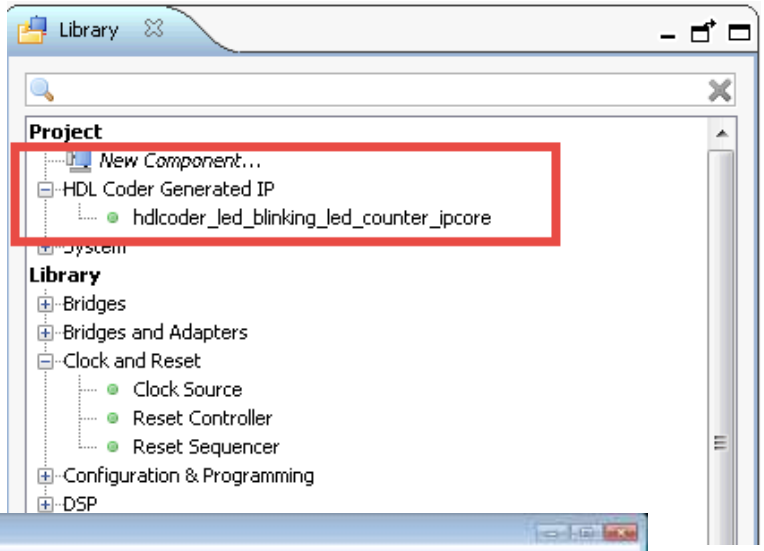
- 直接从MATLAB和Simulink产生可移植和复用的IP核
- 包含AXI4接口，直接连接 Zynq 的ARM处理器
- 产生的IP核可直接集成入Xilinx IP Catalog

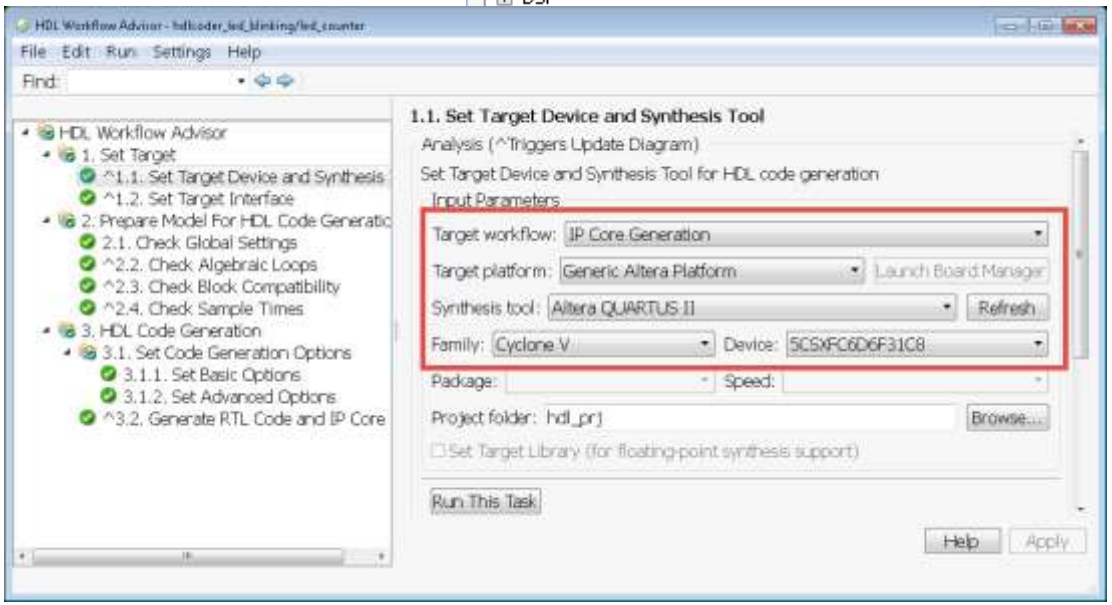




产生Altera IP 核

- 直接从MATLAB和Simulink产生可移植和复用的IP核
- 包含AXI4接口，直接连接 Altera SoC 的ARM处理器
- 产生的报告文档a可做IP核数据表
- 与Altera的Qsys综合工具紧密集成



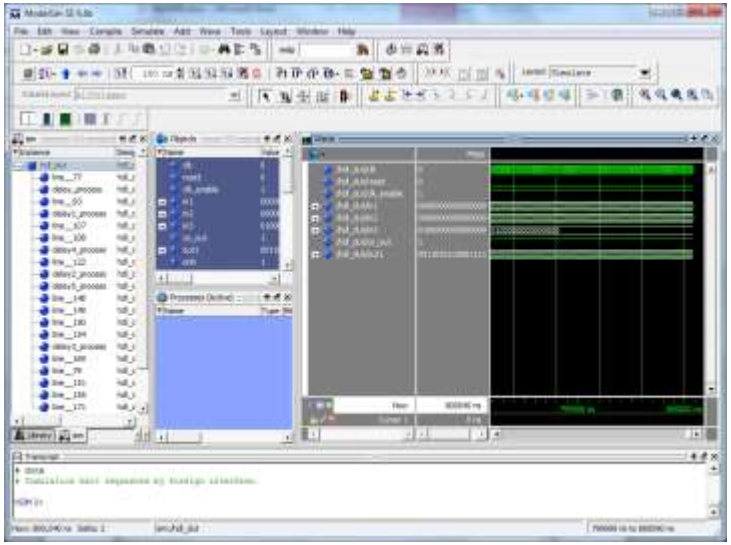


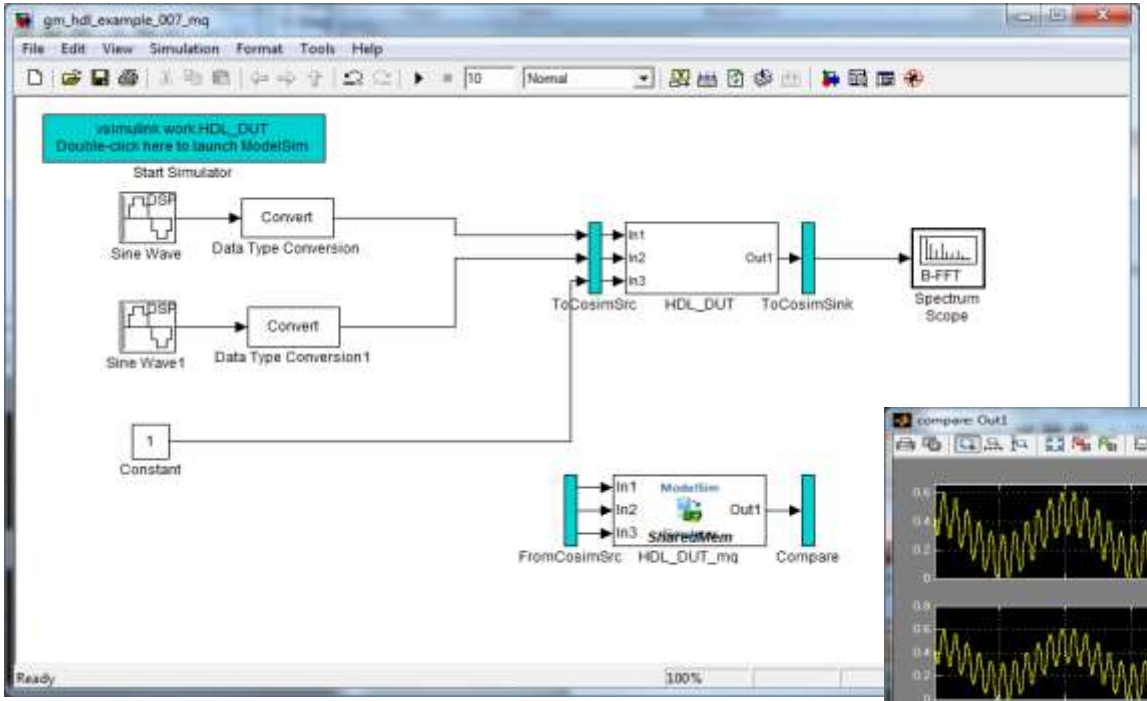
怎么验证产生的代码？

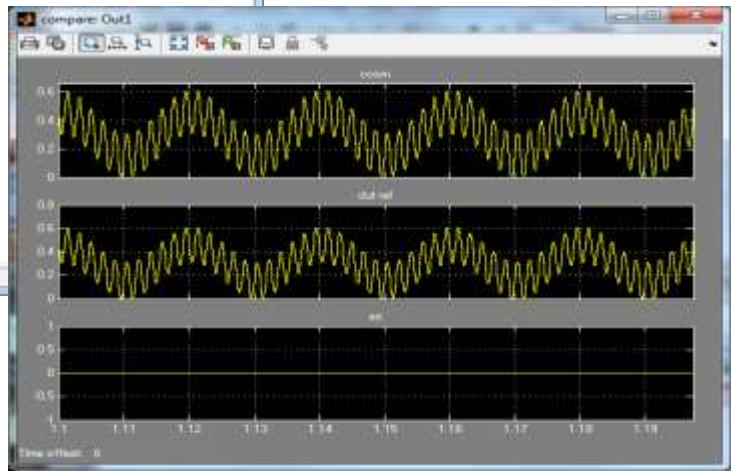
HDL 验证

- 产生独立的测试平台和测试数据
  - VHDL或Verilog测试平台
  - 算法的输入输出记录在数据文件中
  - 可在任何VHDL和Verilog仿真器中验证
- Simulink与EDA仿真器联合仿真
  - Cadence® Incisive®,
  - Mentor Graphics® ModelSim® and Questa®
- FPGA在环仿真 （FPGA-in-the-loop)
  - 算法在FPGA 板卡上跑，测试平台在MATLAB或Simulink中
  - 通过千兆网口或JTAG连接

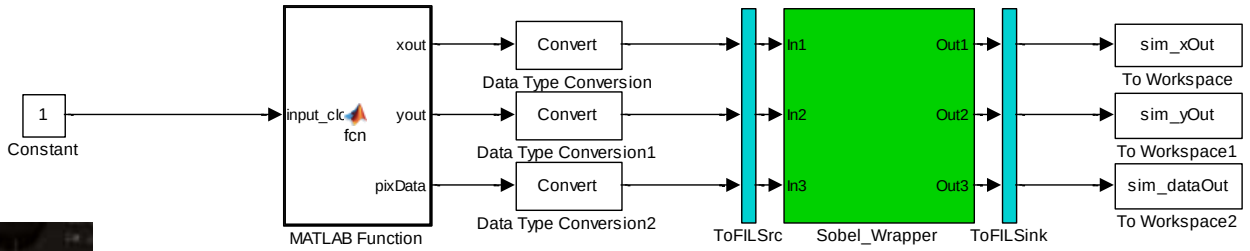
Simulink与EDA仿真器联合仿真

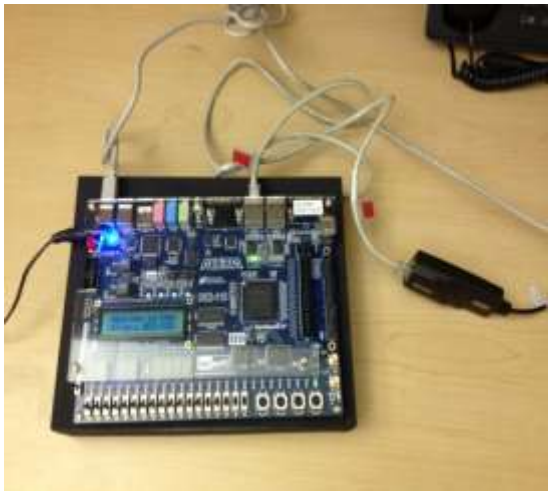


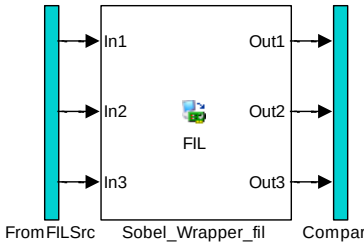


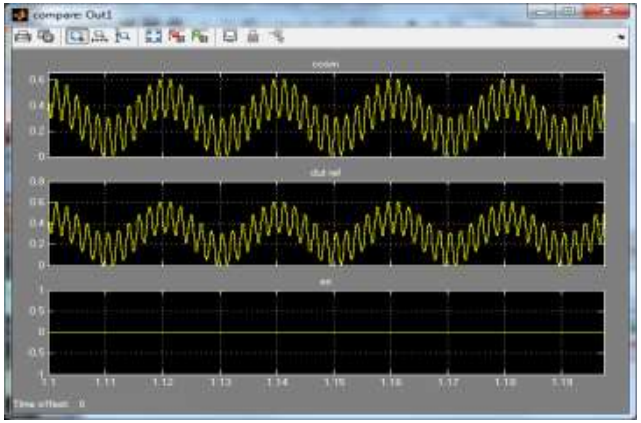


FPGA在环仿真





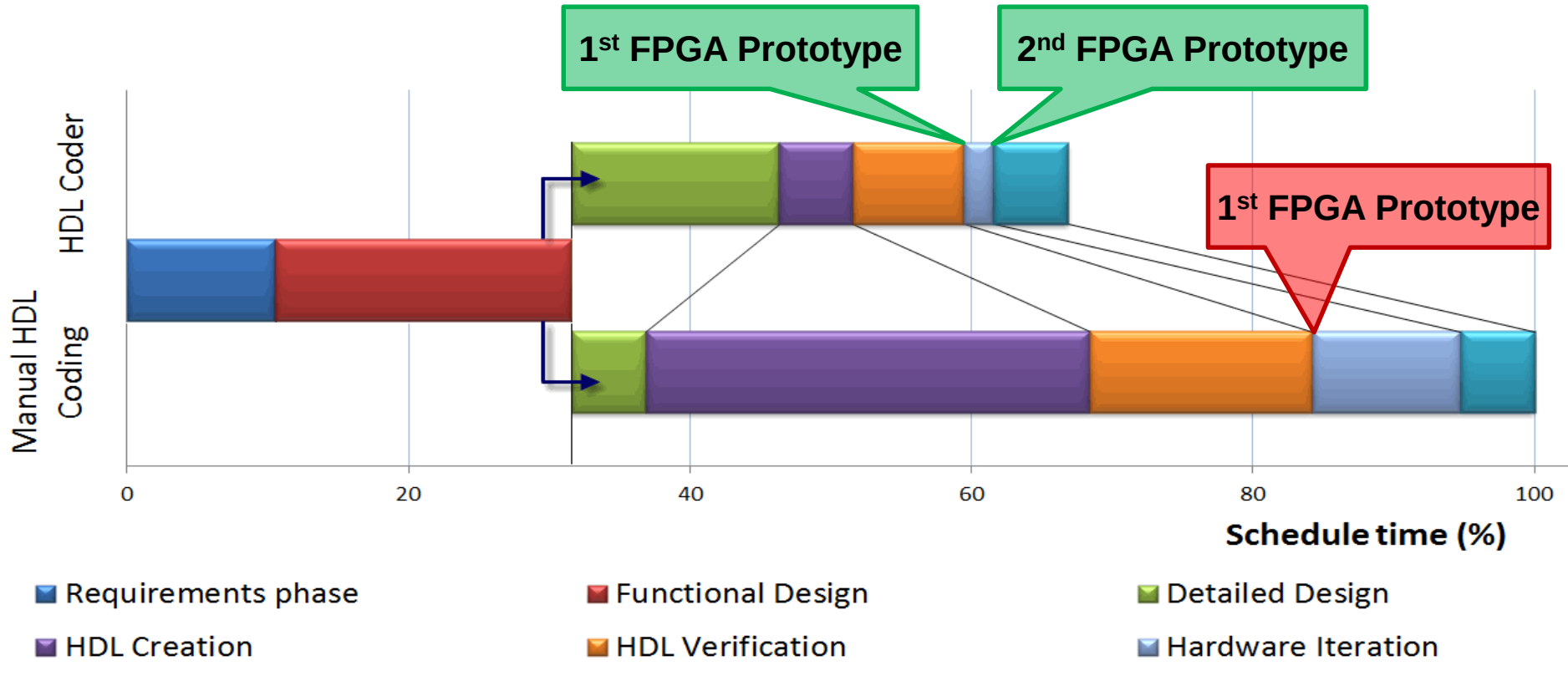




- 产生的HDL自动编译并下载到板卡上运行
- Simulink实时提供输入并采集分析输出

基于模型的设计大大缩短产品设计周期
花在FPGA实现上的时间的对比

- 将FPGA实现所需时间缩短了48% (占项目的总耗时的33%)
- 将设计循环所需的时间缩短了80%



问答