MathWorks

MATLAB EXPO 2017

硬件设计的智能化
从 MATLAB/Simulink 到 FPGA/ASIC/SoC

赵志宏
全球产品市场部经理

FLIR 加速热成像 FPGA 的开发

挑战

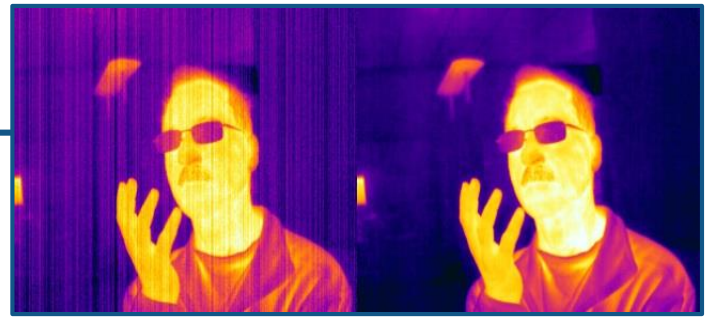
加速高级热成像滤波器的 FPGA 算法开发与实现

解决方法

使用 MATLAB 开发、仿真并评估算法，使用 HDL Coder 在 FPGA 上快速实现最佳算

结果

- 从概念的形成到构建可在现场测试的原型的时间缩短了 60%
- 仅需数小时就完成给了原有功能的提升，而不需耗费数周时间
- 代码复用的比例由零提升至30%Code reuse increased from zero to 30%



原始图像（左）和应用滤波器（通过 HDL Coder 开发而得）后的图像（右）

“借助 MATLAB 和 HDL Coder，我们能够更快地对市场需求做出响应。现在我们之所以能够坦然应对各种变局，原因在于我们可在数周内将新的创意引入具有实时性能的硬件原型上。工程设计过程有了更多乐趣，工作满意度和客户满意度也因此得到提升。”

Nicholas Hogasten

FLIR Systems

3T Develops Robot Emergency Braking System with Model-Based Design

A SCARA robot.

Challenge

Design and implement a robot emergency braking system with minimal hardware testing

Solution

Model-Based Design with Simulink and HDL Coder to model, verify, and implement the controller

Results

- Cleanroom time reduced from weeks to days
- Late requirement changes rapidly implemented
- Complex bug resolved in one day

“With Simulink and HDL Coder we eliminated programming errors and automated delay balancing, pipelining, and other tedious and error-prone tasks. As a result, we were able to easily and quickly implement change requests from our customer and reduce time-to-market.”

Ronald van der Meer

3T

Semtech加快基于FPGA和ASIC的数字收发器的开发

挑战

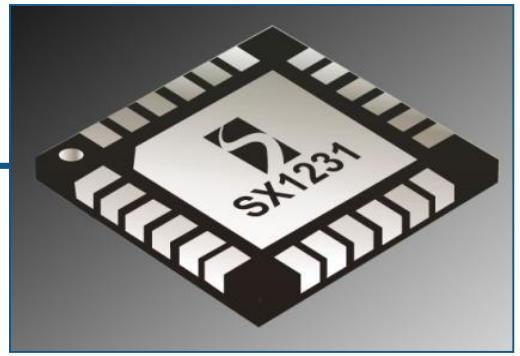
加快无线射频设备的优化数字接收机链路开发

解决方法

使用MathWorks工具进行基于模型的设计，以生成用于快速FPGA和ASIC实现的生产VHDL代码

结果

- 设计创建原型的速度提高了50%.
- 信号处理时间从数周缩短为数天.
- 交付终交付经过优化、性能更佳的设计

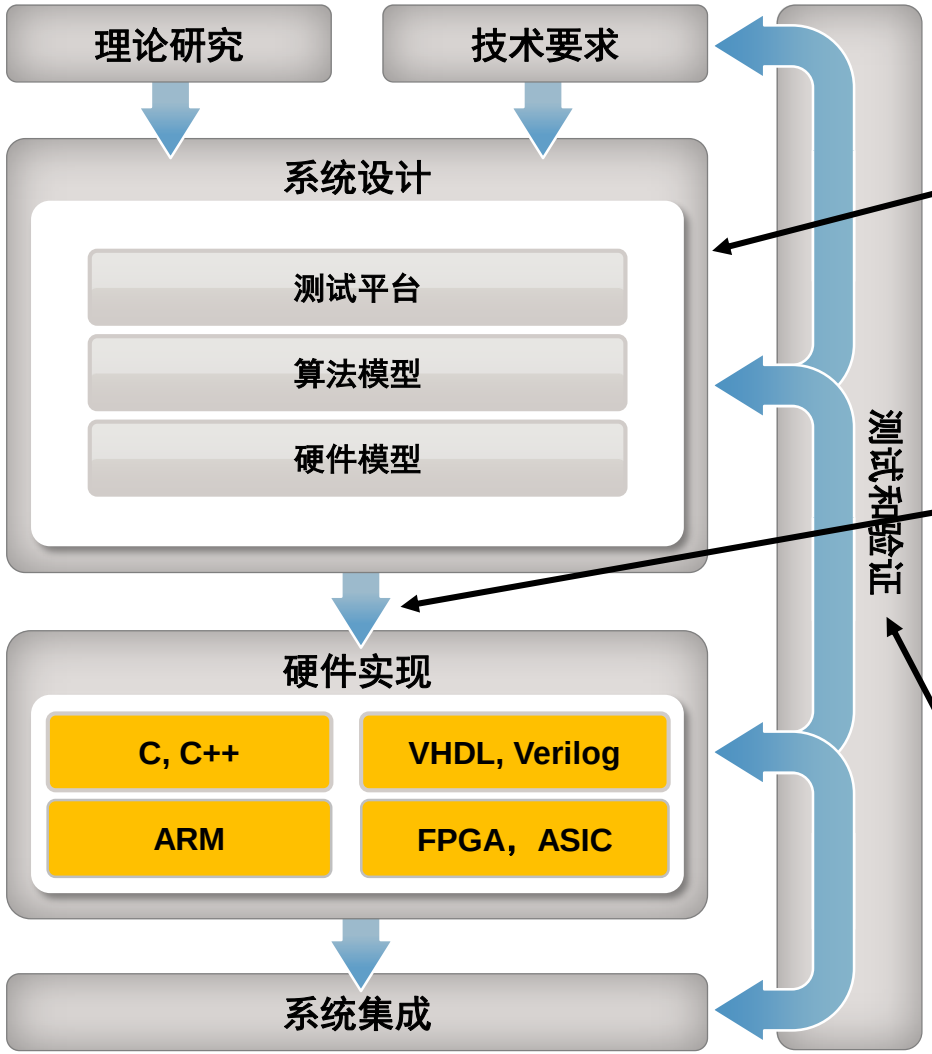


The Semtech SX1231 wireless transceiver.

“编写VHDL是一件非常枯燥的事情，还需要验证手写代码。利用Simulink和HDL Coder，一旦完成对模型的仿真，就可以直接自动生成VHDL，并使用FPGA进行原型验证。这样可以节省大量时间，并且生成的代码还包含一些我们未曾想到的优化。”

Frantz Prianon
Semtech

基于模型的设计流程



- 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

- 直接产生高效可读的代码
- 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

演示实例

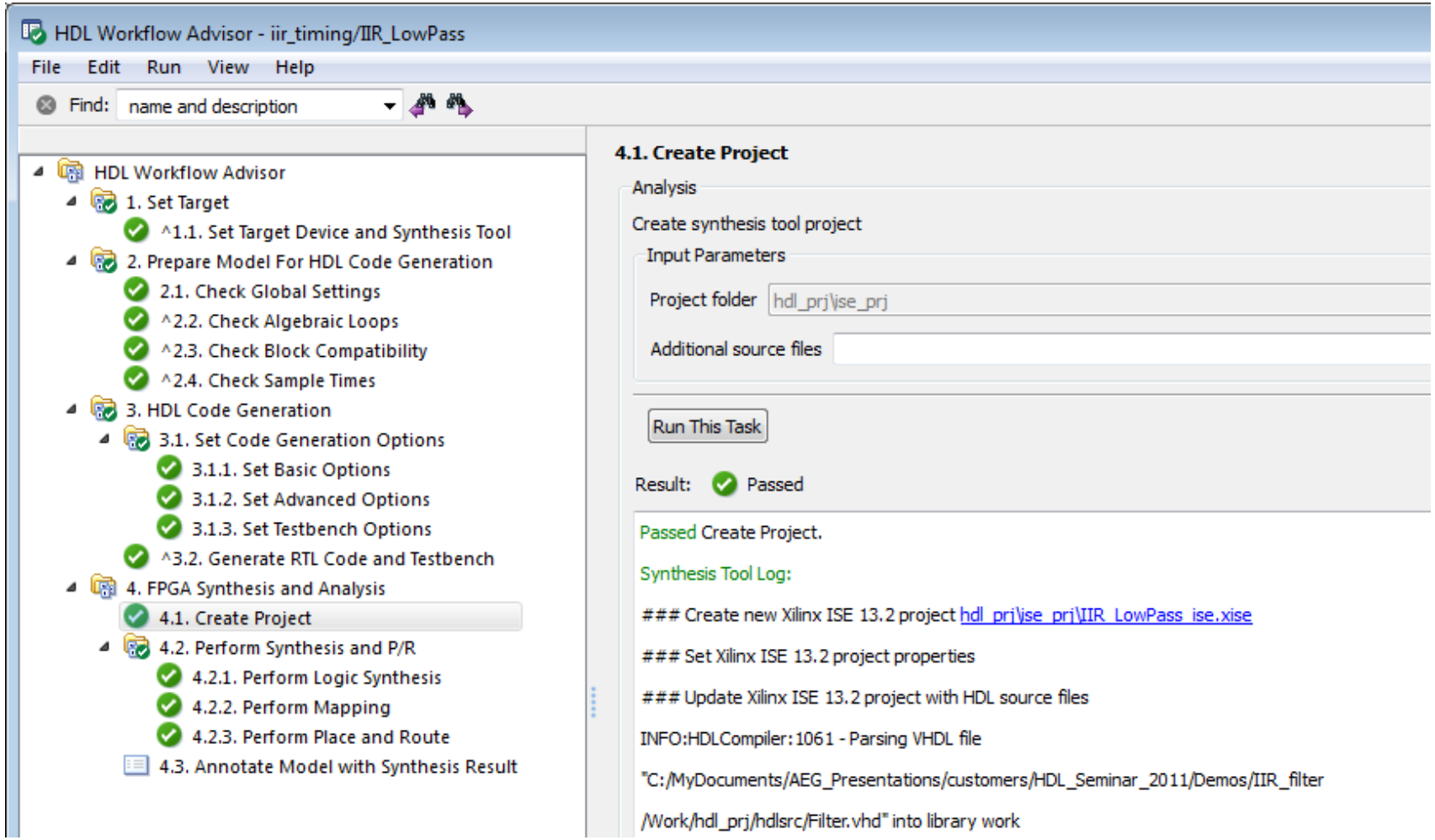
请仔细观察

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

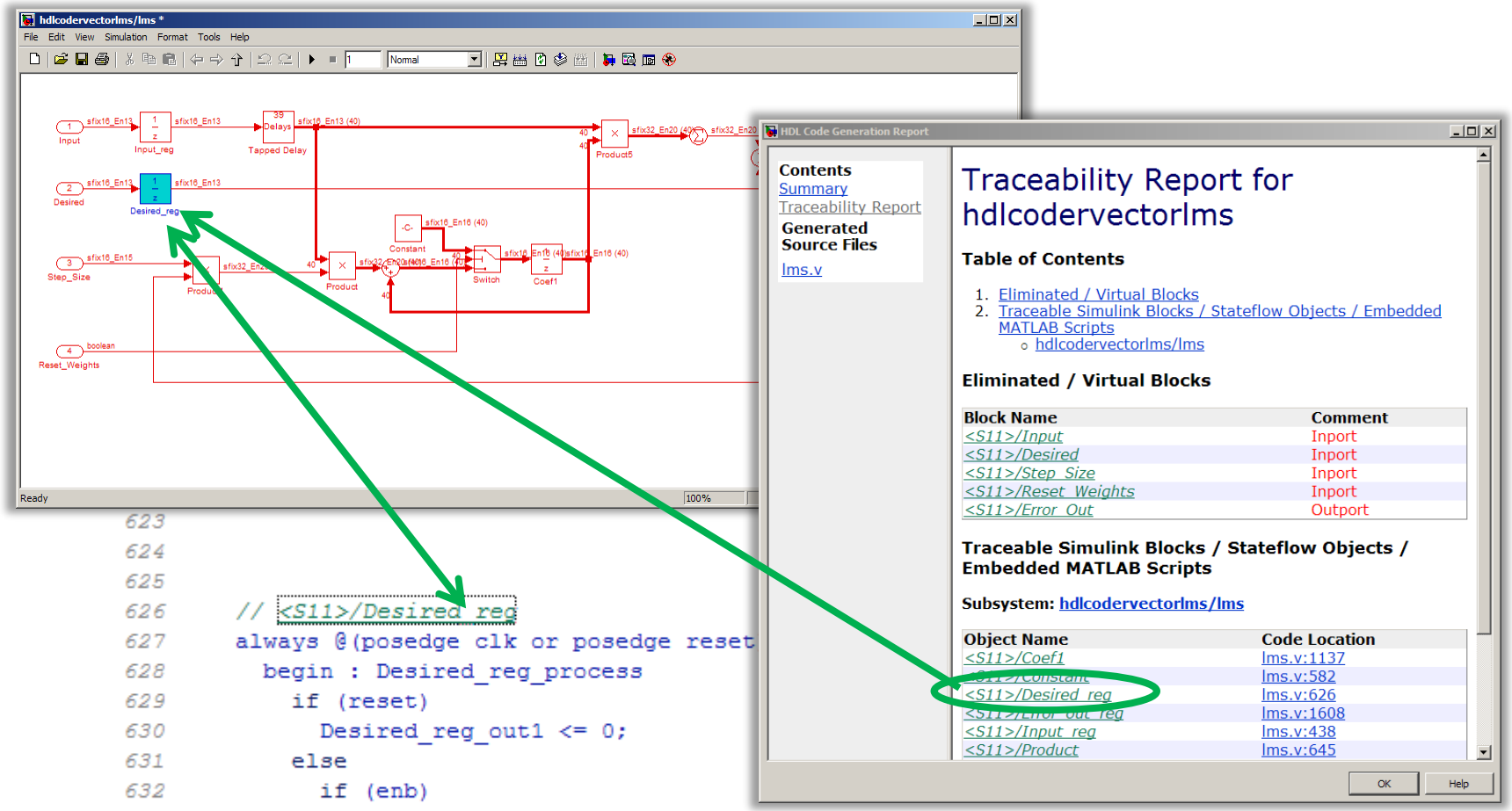
还记得刚刚演示的功能吗？

自动化的工作流程

从模型到FPGA实现和时序分析

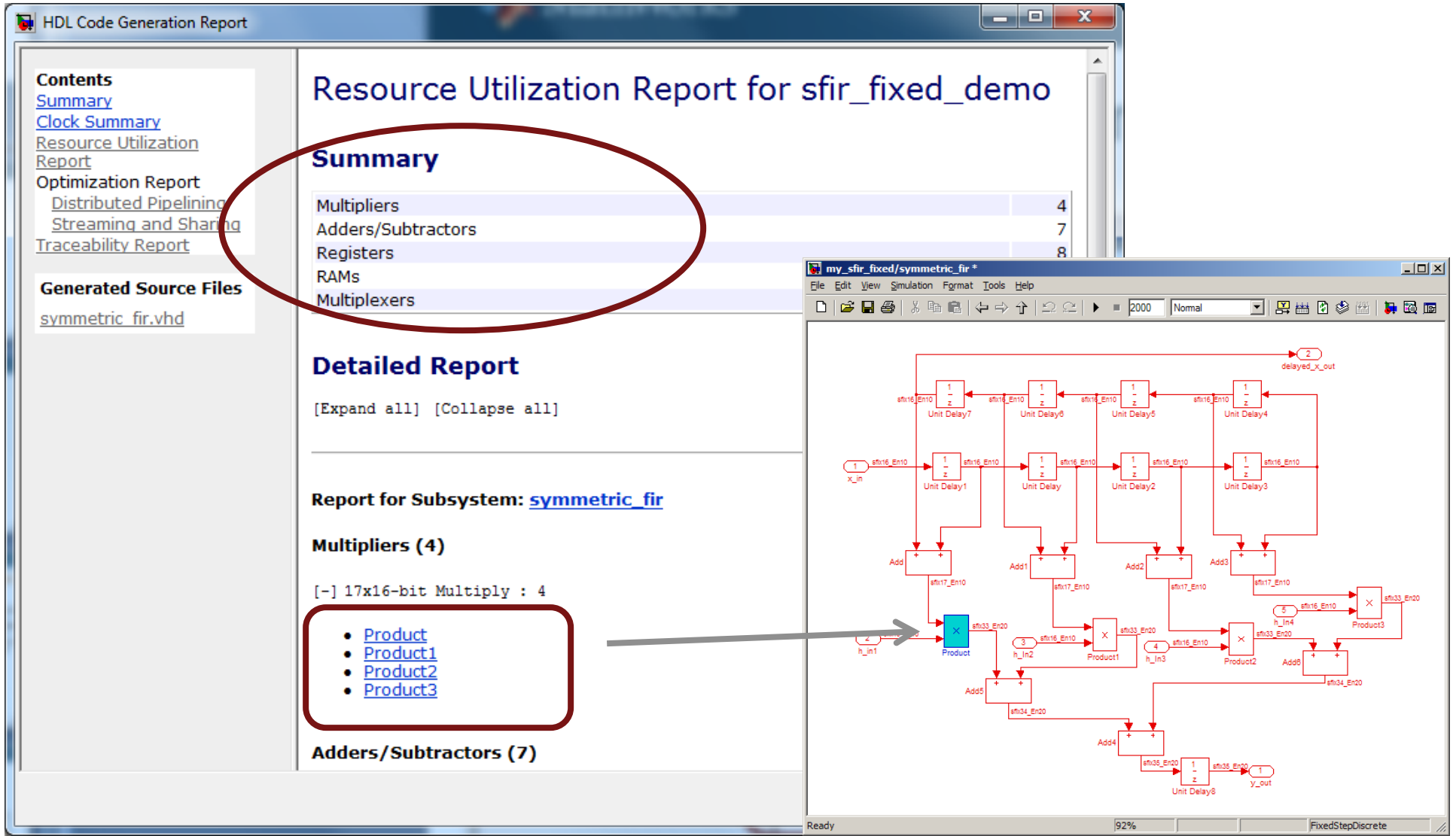


模型和代码的双重追踪性

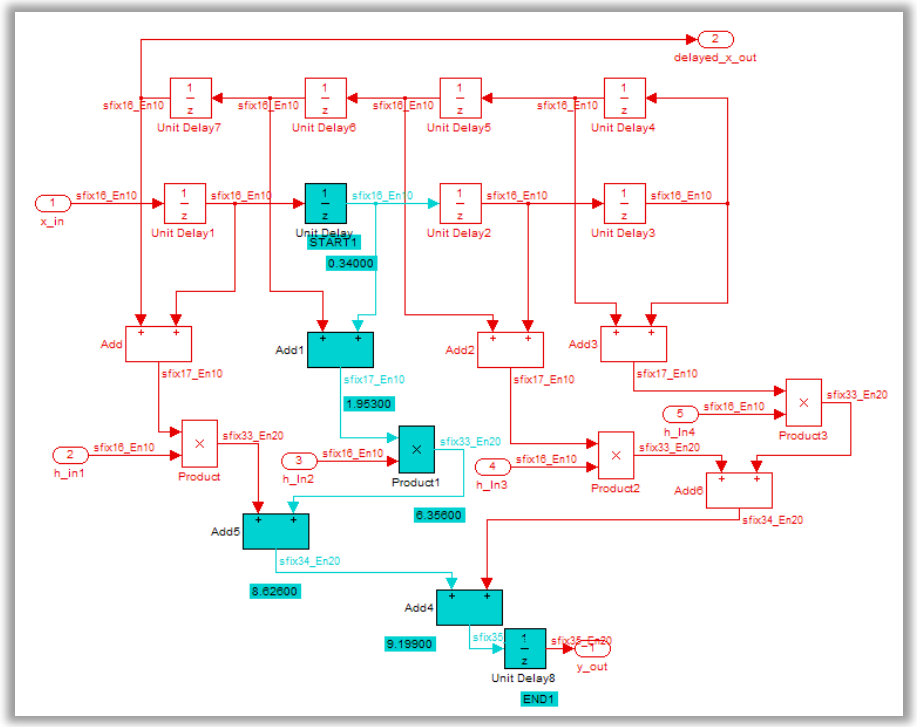



```
623
624
625
626     // <S11>/Desired_reg
627     always @(posedge clk or posedge reset
628       begin : Desired_reg_process
629         if (reset)
630           Desired_reg_out1 <= 0;
631         else
632           if (enb)
633             Desired_reg_out1 <= Desired;
634
635       end
```

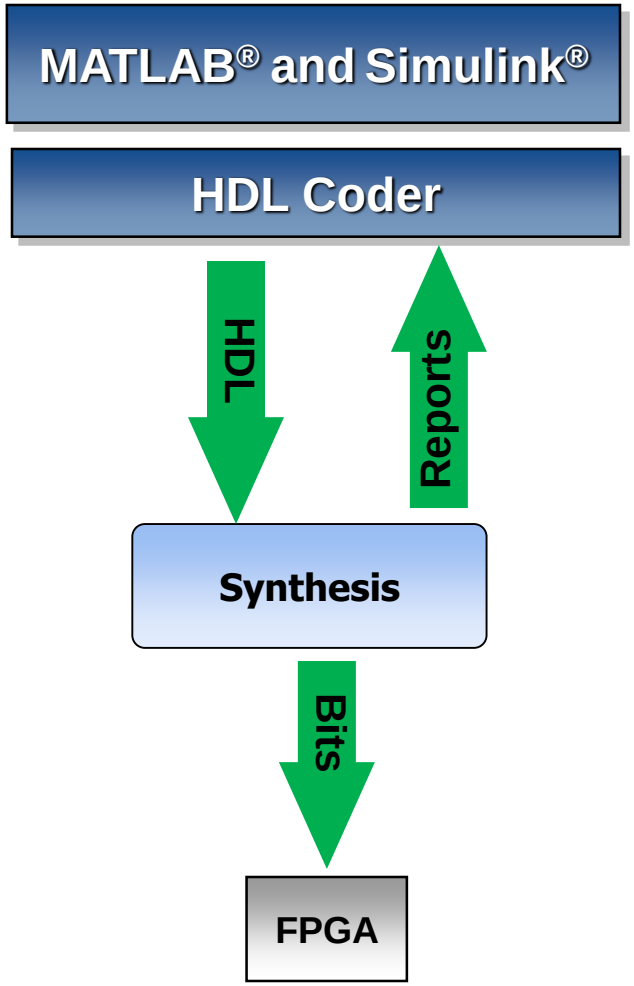
资源使用预估


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)
my_sfir_fixed/symmetric_fir *
File Edit View Simulation Format Tools Help
Normal
Ready
92%
FixedStepDiscrete

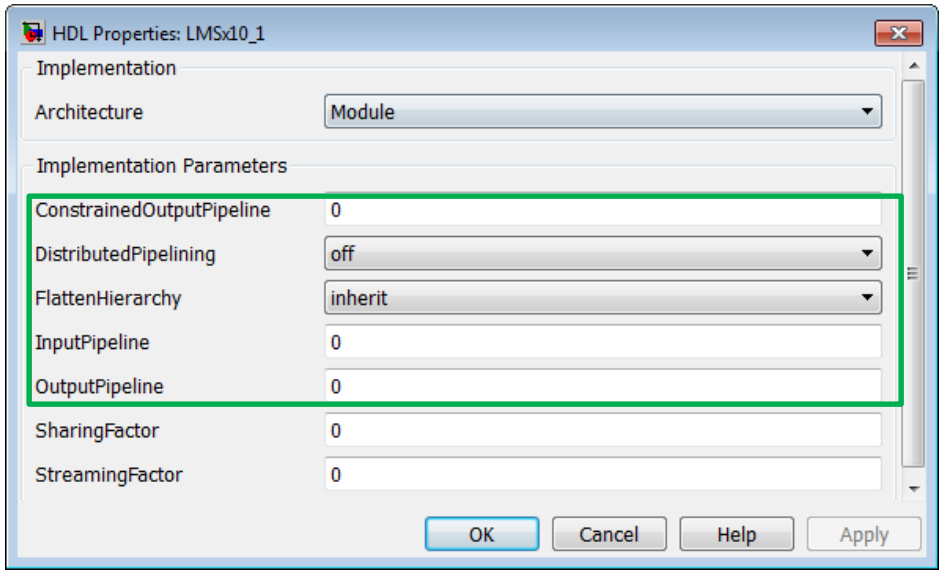
找出关键路径



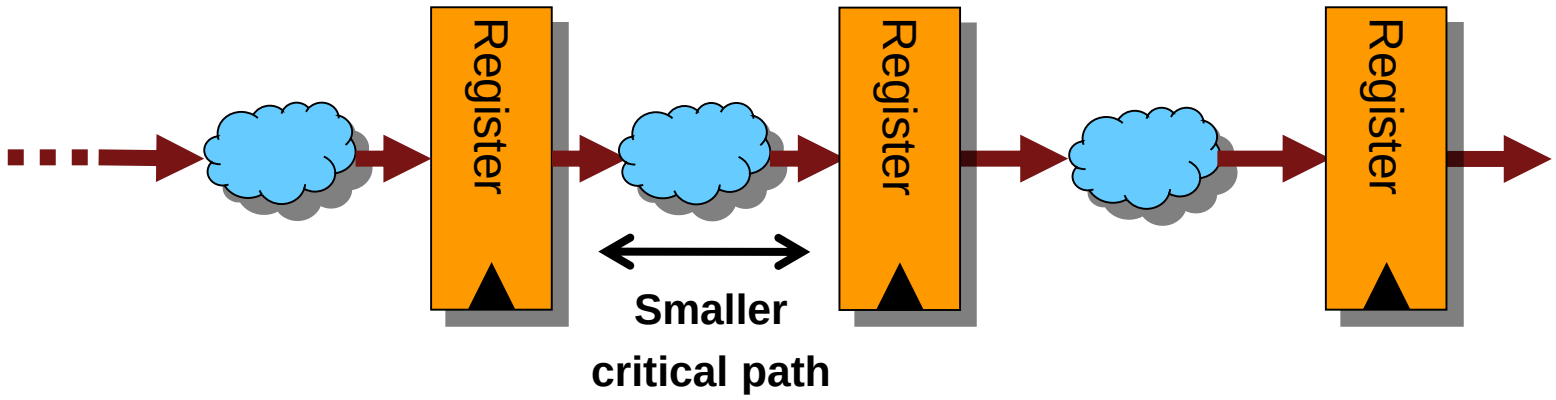
在算法结构中直接看到实现后的关键路径



满足时序约束
分布式管道寄存器

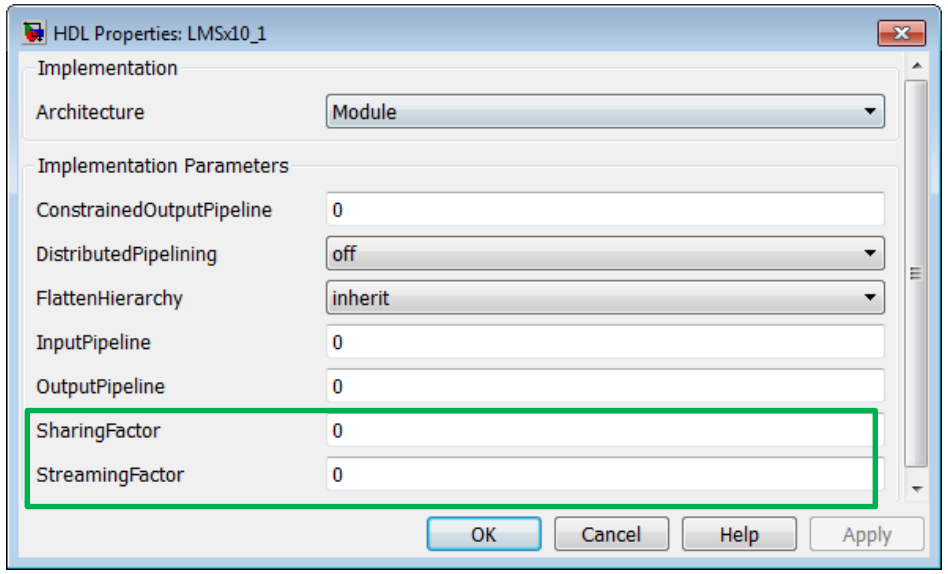


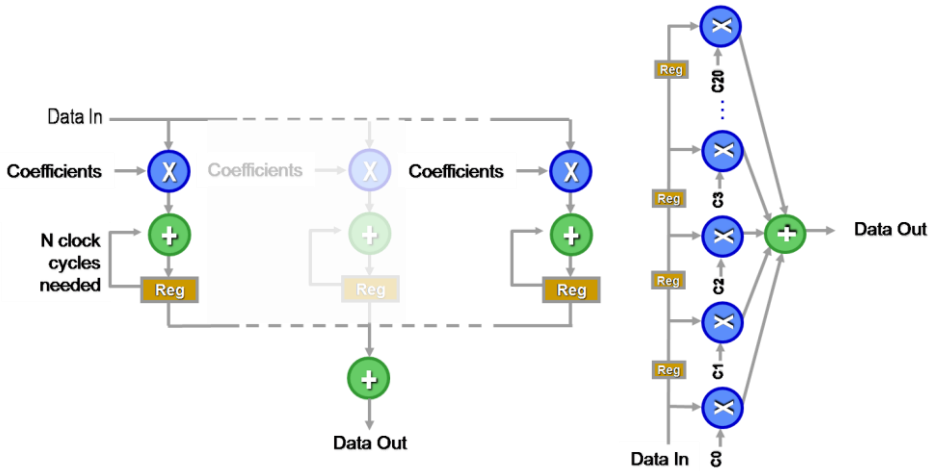
- 分布式管道寄存器 (在模型中重定时)
- 在需要时自动补偿延迟
- 用户可约束式重定时

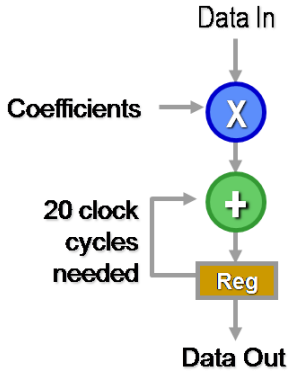


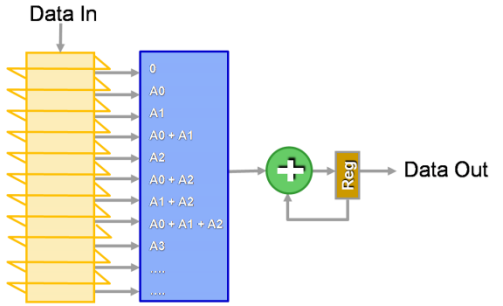
满足资源约束

资源共享

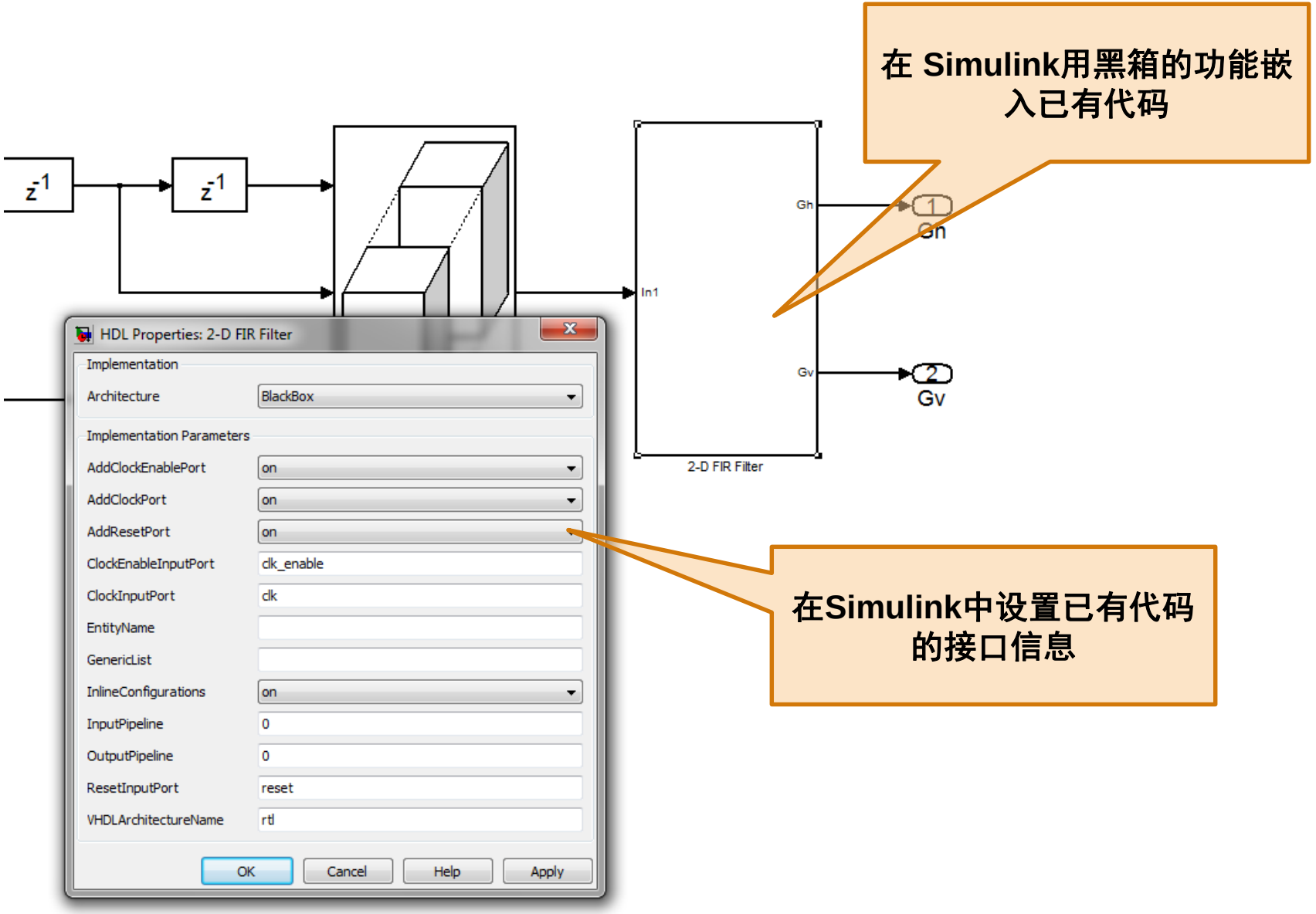

HDL Properties: LMSx10_1
Implementation
Architecture
Module
Implementation Parameters
ConstrainedOutputPipeline
0
DistributedPipelining
off
FlattenHierarchy
inherit
InputPipeline
0
OutputPipeline
0
SharingFactor
0
StreamingFactor
0
OK
Cancel
Help
Apply


Data In
Coefficients
Coefficients
Coefficients
N clock cycles needed
Reg
Reg
Reg
Data Out
Reg
C20
C3
C1
C0
Data In
Data Out


Data In
Coefficients
20 clock cycles needed
Reg
Data Out


Data In
0
A0
A1
A0 + A1
A2
A0 + A2
A1 + A2
A0 + A1 + A2
A3
Reg
Data Out

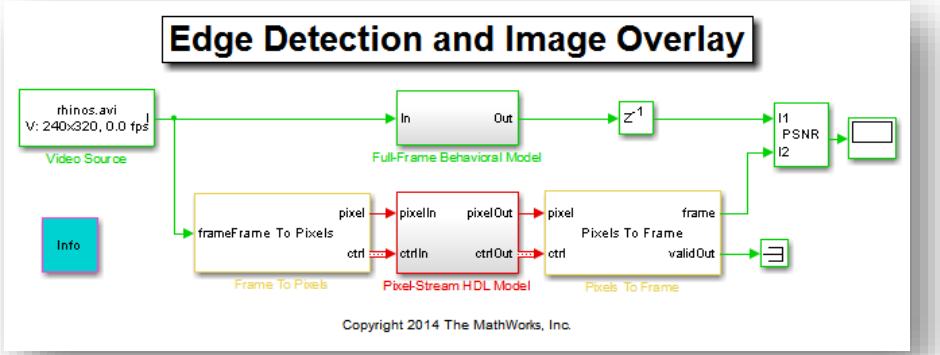
集成已有HDL代码


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

有什么样高级的算法模块吗？

视觉HDL工具箱

- **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



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

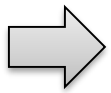
用MATLAB代码编写你自己的模块

```
%#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; uc
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;
```

能产生浮点数的HDL代码吗？

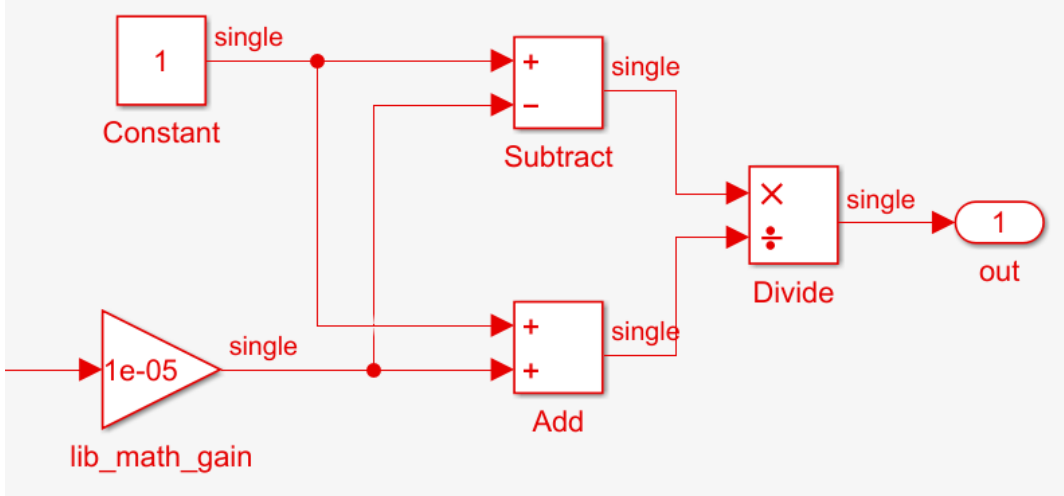
浮点数运算的硬件实现

- 算法要求很大的数值范围或很高的精度
- 在转定点数之前很快在硬件上验证算法的正确性
- 整个或部分算法可以自动转化成定点数

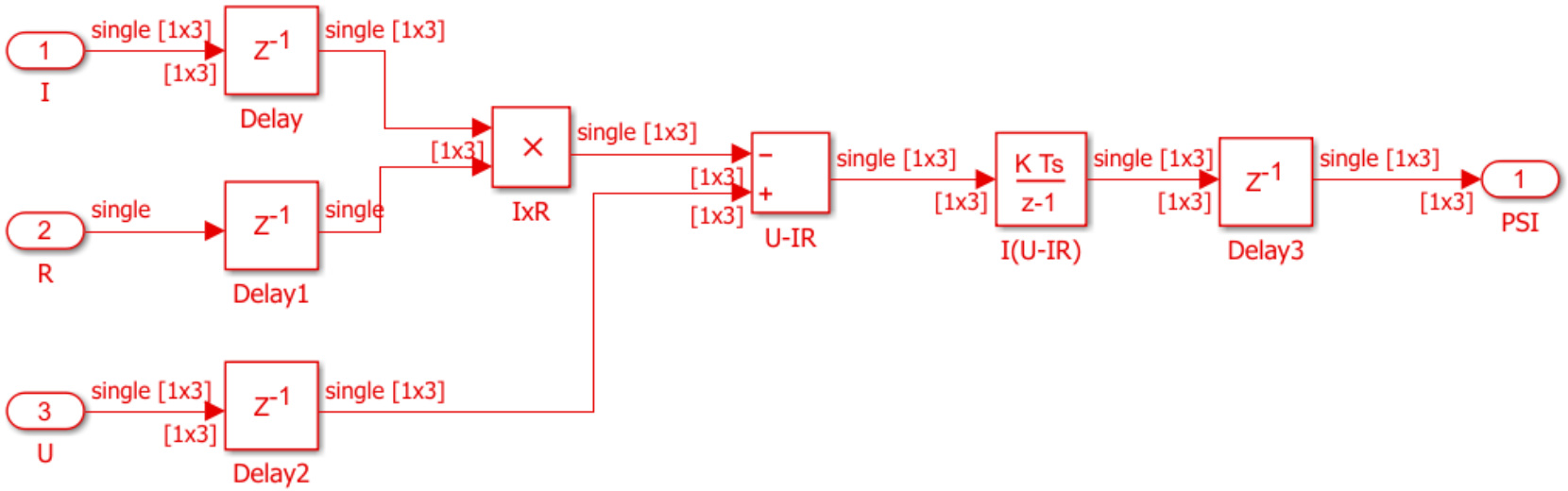
常用的数学表达式的模型

$$\frac{1-a}{1+a}$$

$RangeMin$: $-2^{24}-1$
$RangeMax$: $2^{24}+3$



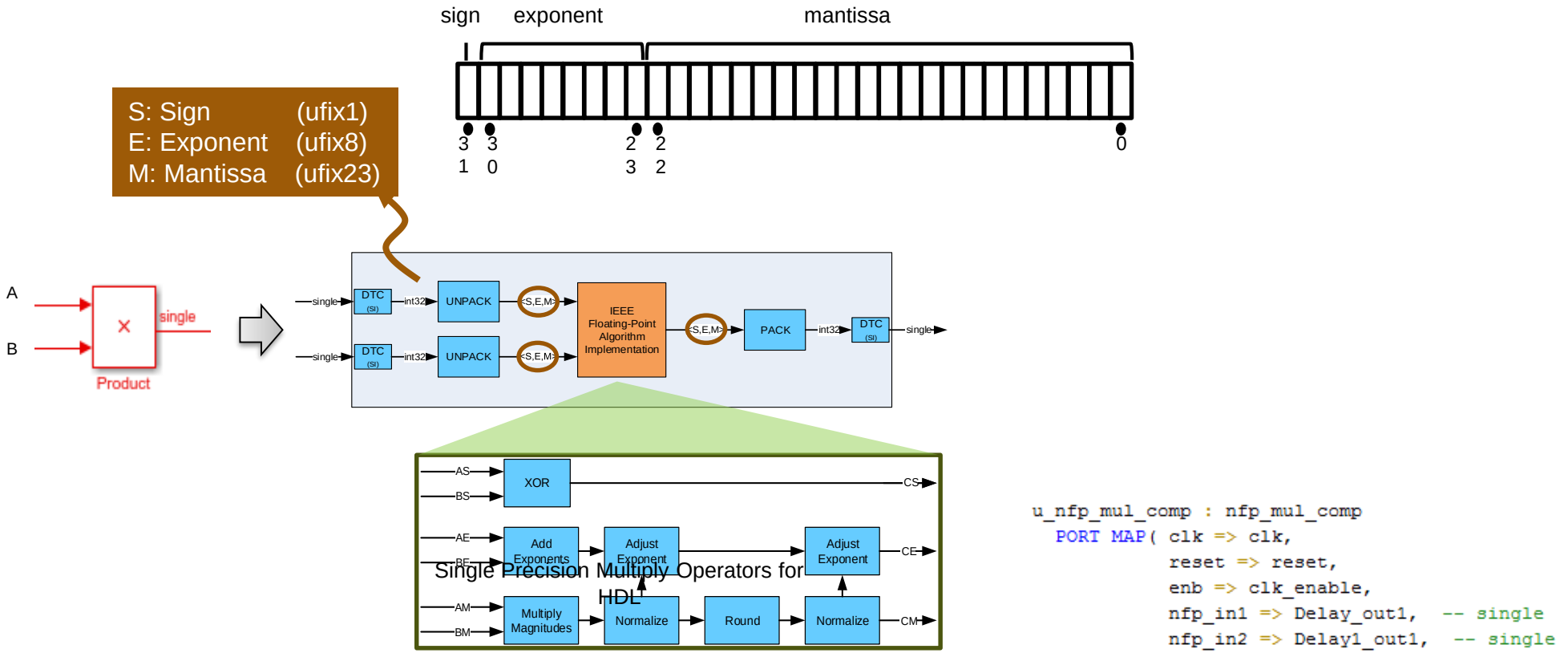
通量方程建模



$$\psi_{Mt\_ab}(t) = \int (U_{M\_ab}(t) - I_{Mt\_ab}(t) \cdot R_{Mt}) \cdot dt$$

$\psi_{Mt_ab}(t) = \int (U_{M_ab} (t) - I_{Mt_ab}(t) \cdot R_{Mt})\cdot dt$

浮点数的运算



Single Precision Multiply Operators for HDL

```
u_nfp_mul_comp : nfp_mul_comp
  PORT MAP( clk => clk,
            reset => reset,
            enb => clk_enable,
            nfp_in1 => Delay_out1,   -- single
            nfp_in2 => Delay1_out1,   -- single
            nfp_out => Product_out1   -- single
            );
```

支持浮点数代码产生的数学运算

- R2017a版本
  - Add, Sub, Mul, Reciprocal, Div, Sqrt, RecipSqrt, Mod, Rem
  - Sin, Cos, SinCos, Atan, Atan2, Exp, Log
  - SOE, POE, DTI, PID, DTF, Discrete FIR

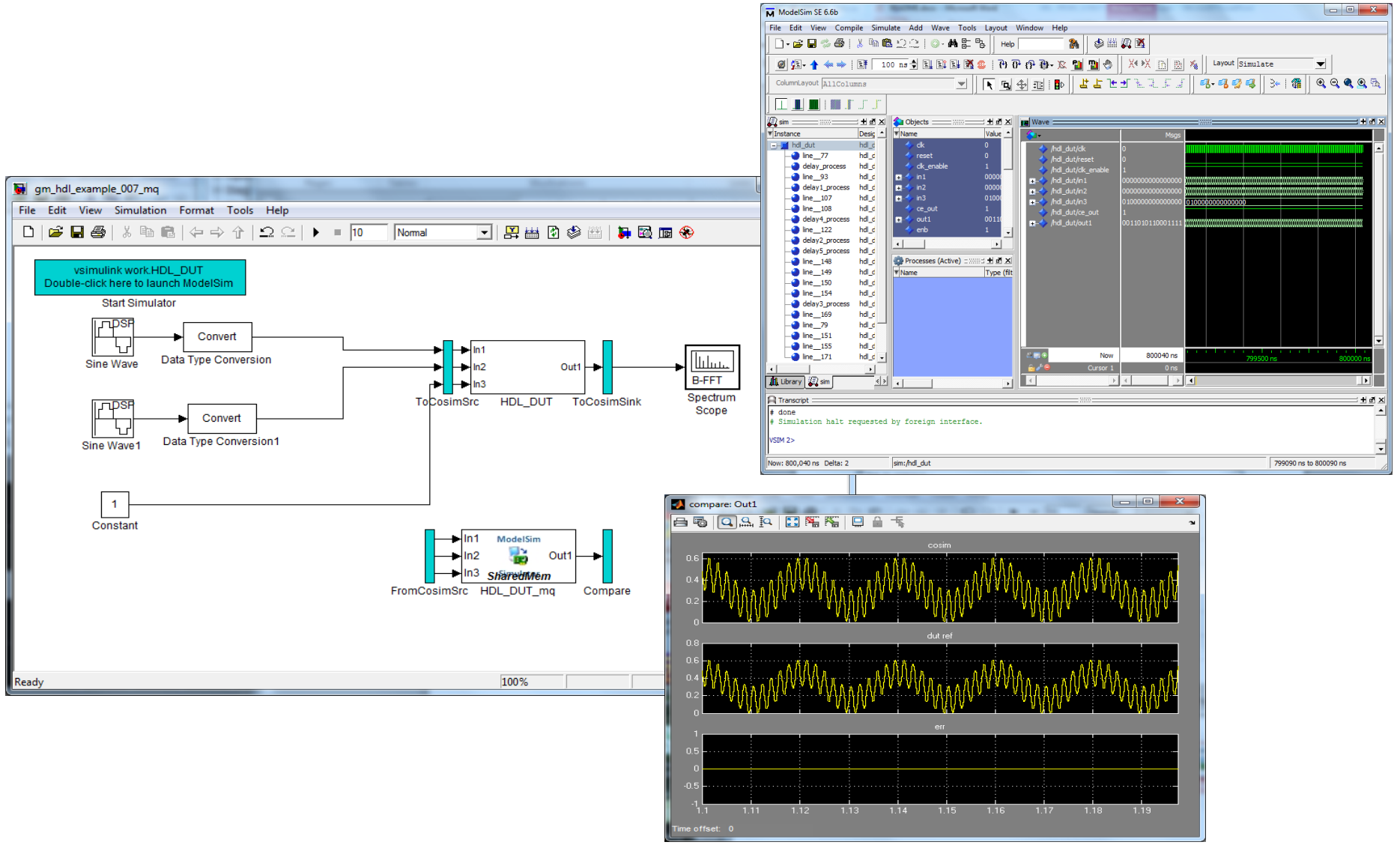
- 后续版本
  - 10^u, log10, acos, asin, tan, reciprocal
  - Mag^2, pow, conj, hypot, square,
  - hyperbolic functions (sinh, cosh, tanh, acosh, asinh, atanh)

怎么验证产生的代码？

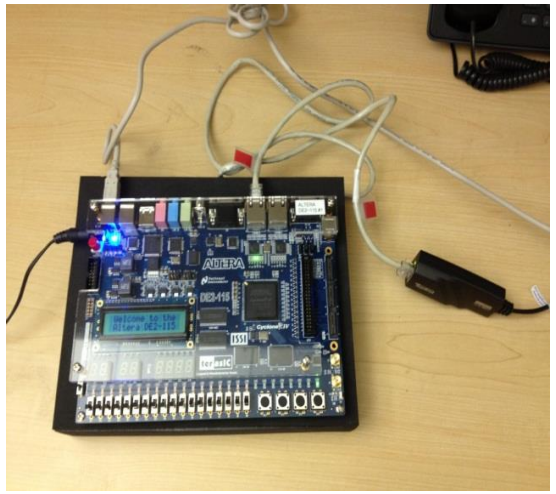
HDL 验证

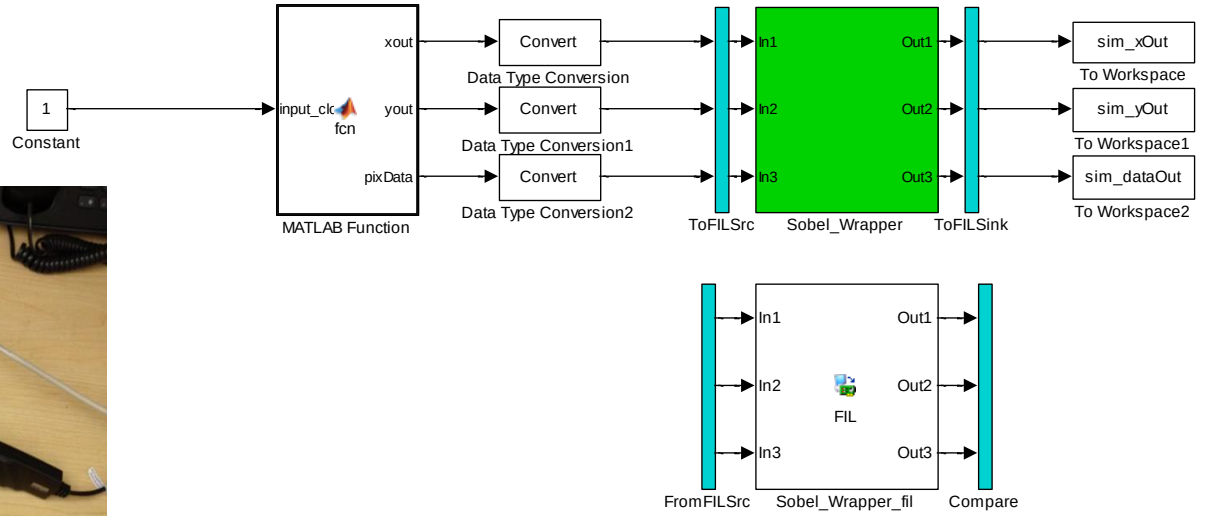
- 产生独立的测试平台和测试数据
  - VHDL或Verilog测试平台
  - 算法的输入输出记录在数据文件中
  - 可在任何VHDL和Verilog仿真器中验证
- Simulink与EDA仿真器联合仿真
  - Cadence® Incisive®,
  - Mentor Graphics® ModelSim® and Questa®
- FPGA在环仿真 （FPGA-in-the-loop)
  - 算法在FPGA 板卡上跑，测试平台在MATLAB或Simulink中
  - 通过千兆网口或JTAG连接
  - 利用是数据扑捉功能在Simulink里调试硬件

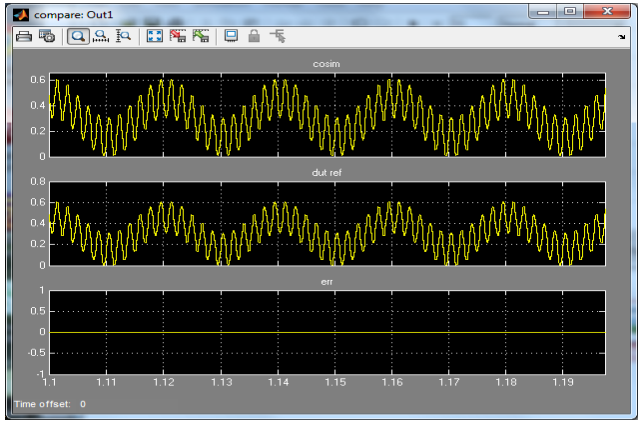
Simulink与EDA仿真器联合仿真



FPGA在环仿真



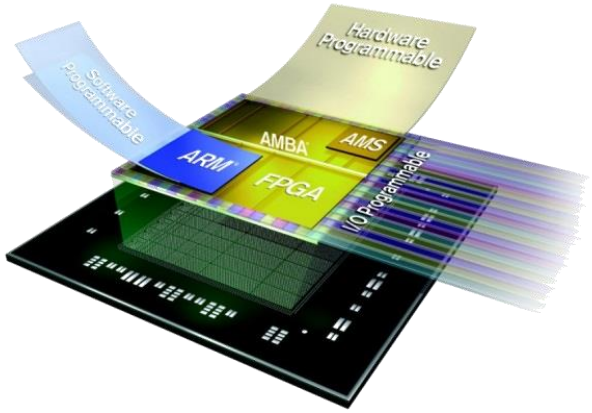




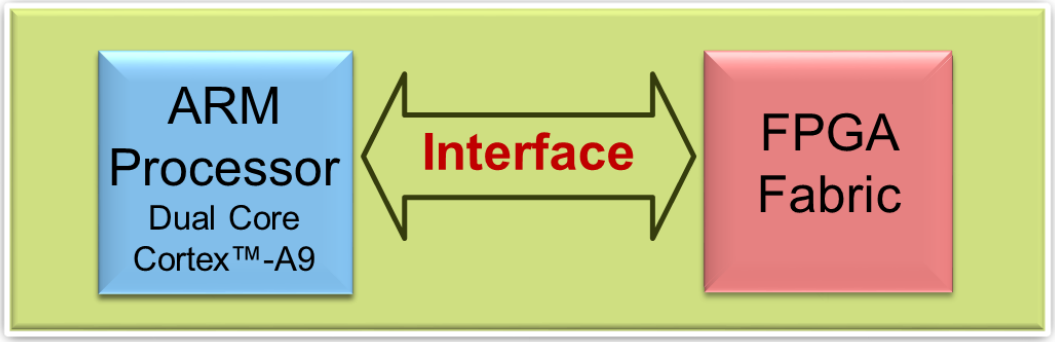
- 产生的HDL自动编译并下载到板卡上运行
- Simulink实时提供输入并采集分析输出

支持SoC的设计流程吗？

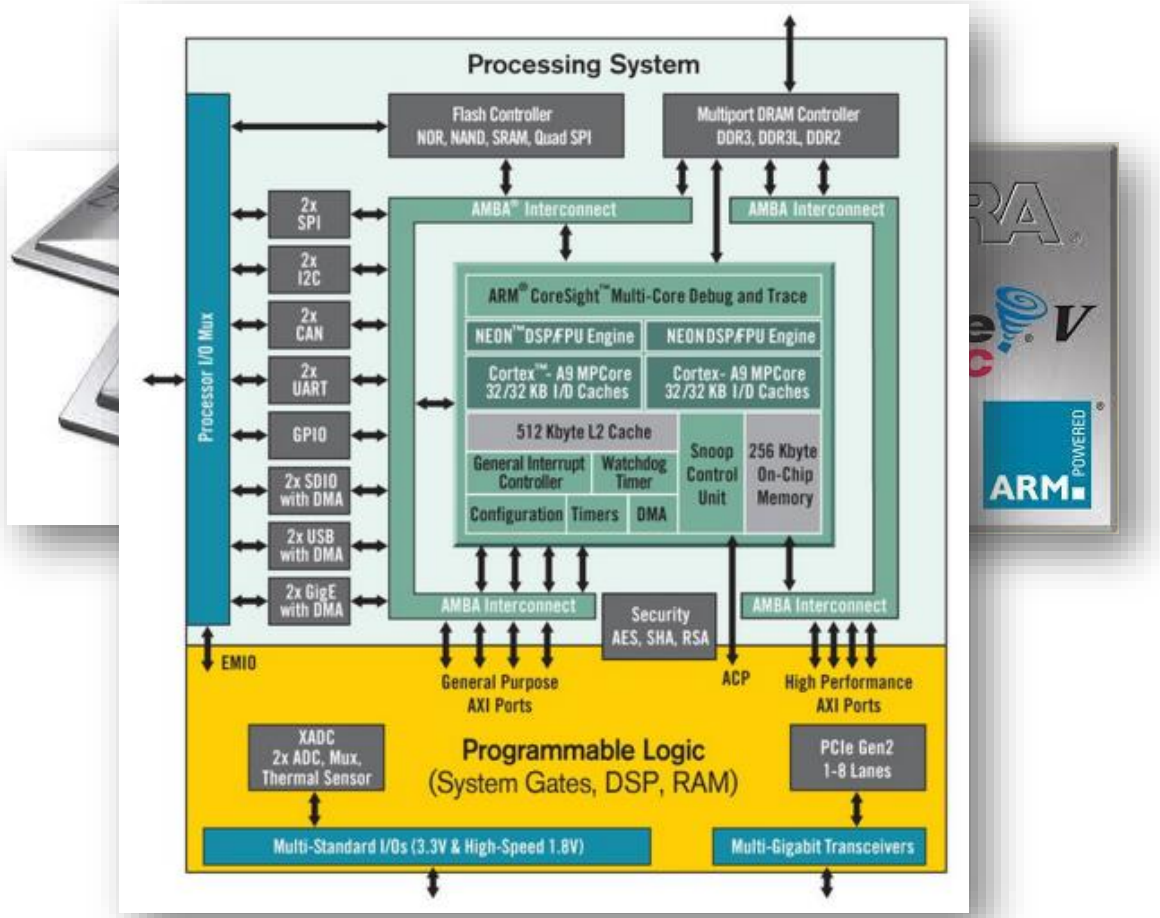
SoC的设计和实现


Hardware Programmable
Software Programmable
AMBA
AMS
ARM
FPGA
I/O Programmable

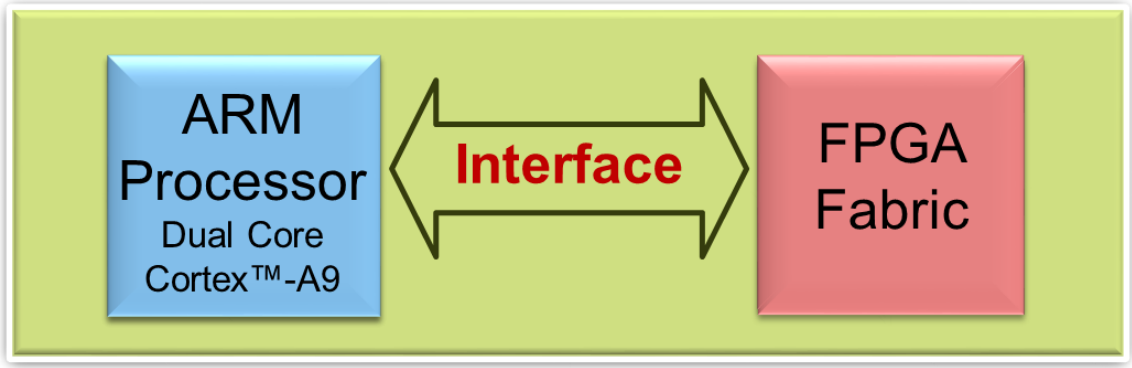

ZYNQ
7000
350
ZYNQ
7000
7045
ALTERA
Cyclone V
SoC
ARM
POWERED


ARM
Processor
Dual Core
Cortex™-A9
Interface
FPGA
Fabric

SoC 设计的挑战性


Processing System
Flash Controller
NOR, NAND, SRAM, Quad SPI
Multiport DRAM Controller
DDR3, DDR3L, DDR2
Processor I/O Mux
2x SPI
2x I2C
2x CAN
2x UART
GPIO
2x SDIO with DMA
2x USB with DMA
2x GigE with DMA
AMBA® Interconnect
AMBA Interconnect
ARM® CoreSight™ Multi-Core Debug and Trace
NEON™ DSP/FPU Engine
NEON DSP/FPU Engine
Cortex™- A9 MPCore 32/32 KB I/D Caches
Cortex- A9 MPCore 32/32 KB I/D Caches
512 Kbyte L2 Cache
General Interrupt Controller
Watchdog Timer
Configuration
Timers
DMA
Snoop Control Unit
256 Kbyte On-Chip Memory
AMBA Interconnect
Security AES, SHA, RSA
AMBA Interconnect
EMIO
General Purpose AXI Ports
ACP
High Performance AXI Ports
XADC 2x ADC, Mux, Thermal Sensor
Programmable Logic
(System Gates, DSP, RAM)
PCIe Gen2 1-8 Lanes
Multi-Standard I/Os (3.3V & High-Speed 1.8V)
Multi-Gigabit Transceivers
ARM POWERED

SoC 设计的挑战性



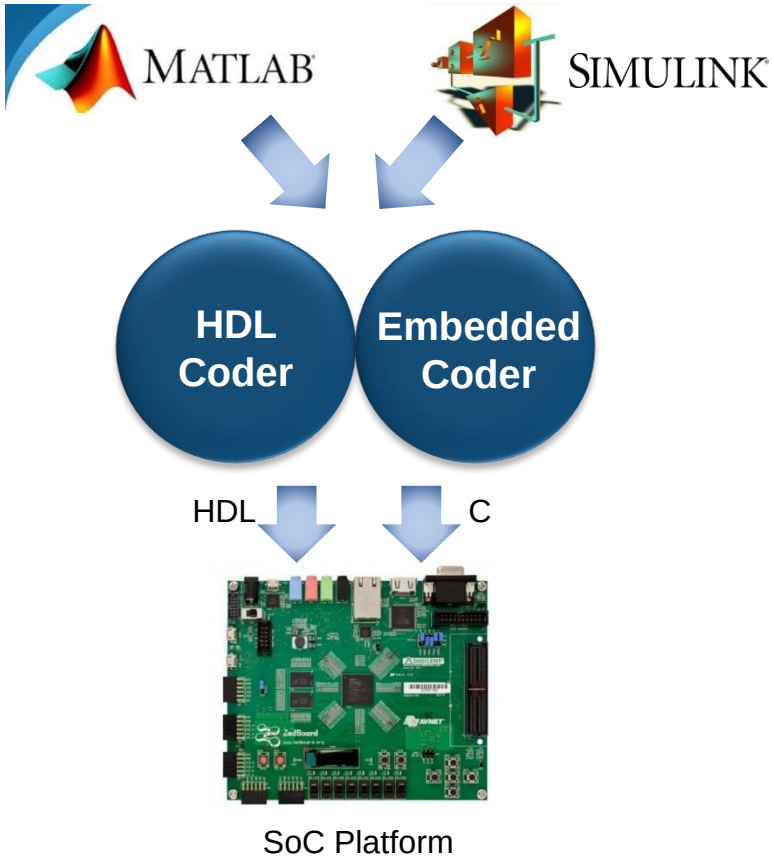
Design Challenge

- System partitioning
- Functional verification
- Performance estimation

Implementation challenge

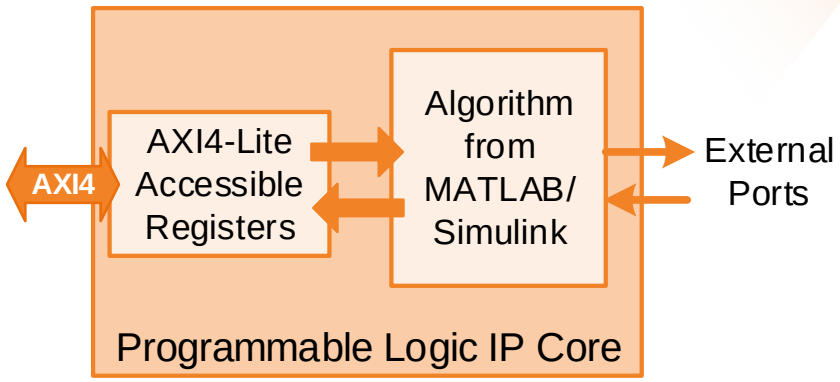
- Embedded Processor and FPGA
- Interconnect
- Peripherals

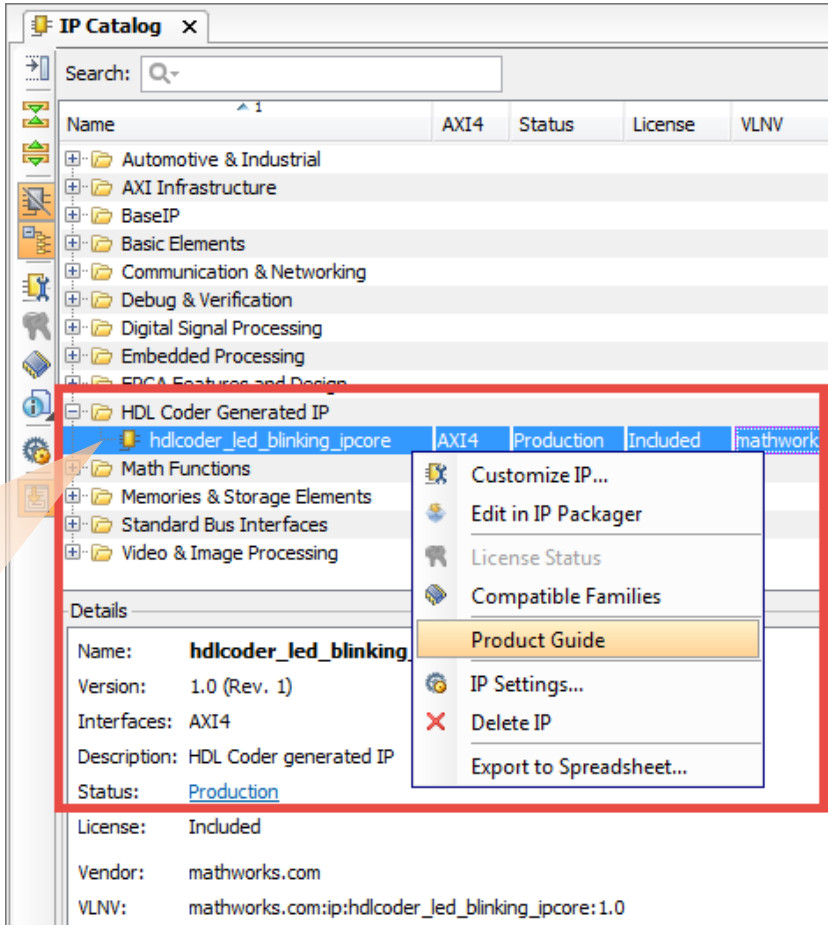
SoC的软硬件共同设计



产生Xilinx Vivado IP 核

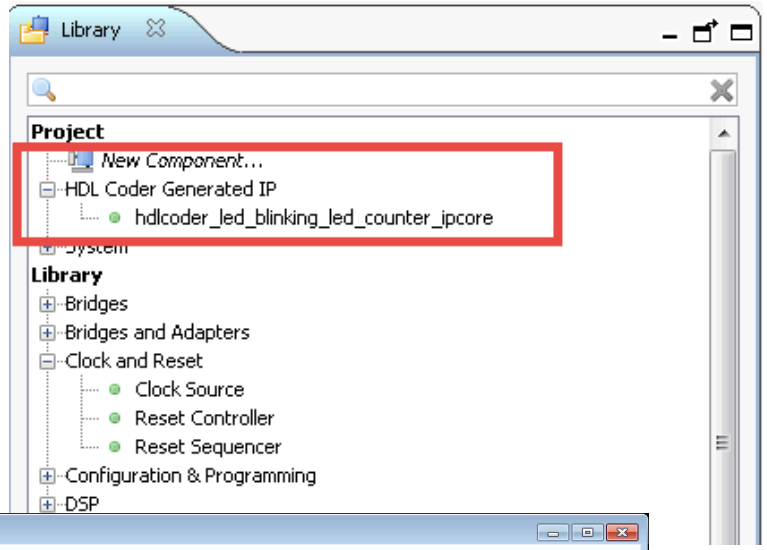
- 直接从MATLAB和Simulink产生可移植和复用的IP核
- 包含AXI4接口，直接连接 Zynq 的ARM处理器
- 产生的IP核可直接集成入Xilinx IP Catalog

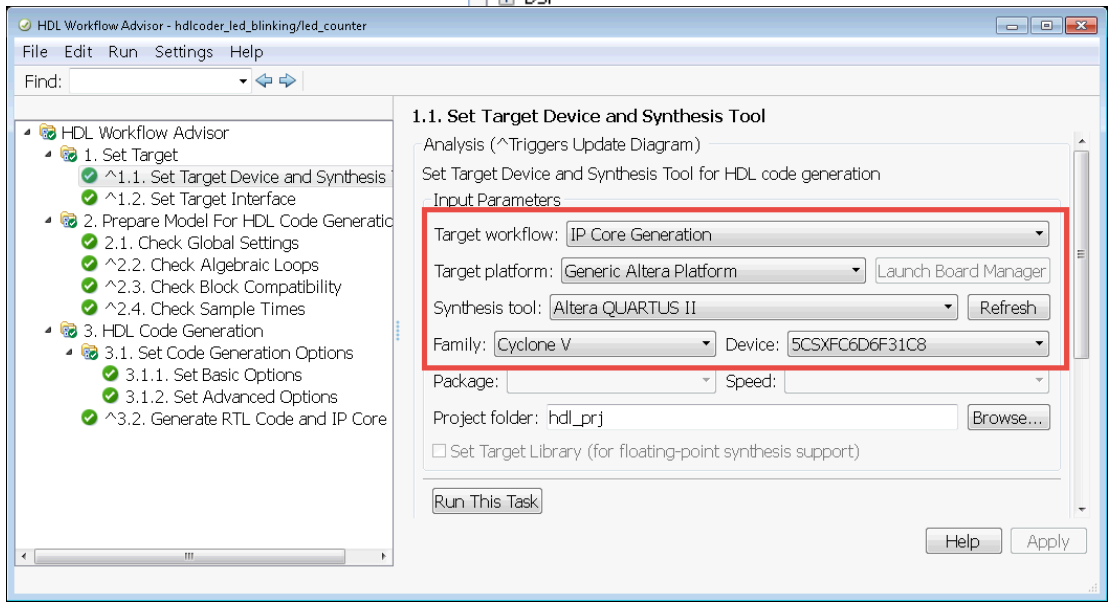




产生Altera IP 核

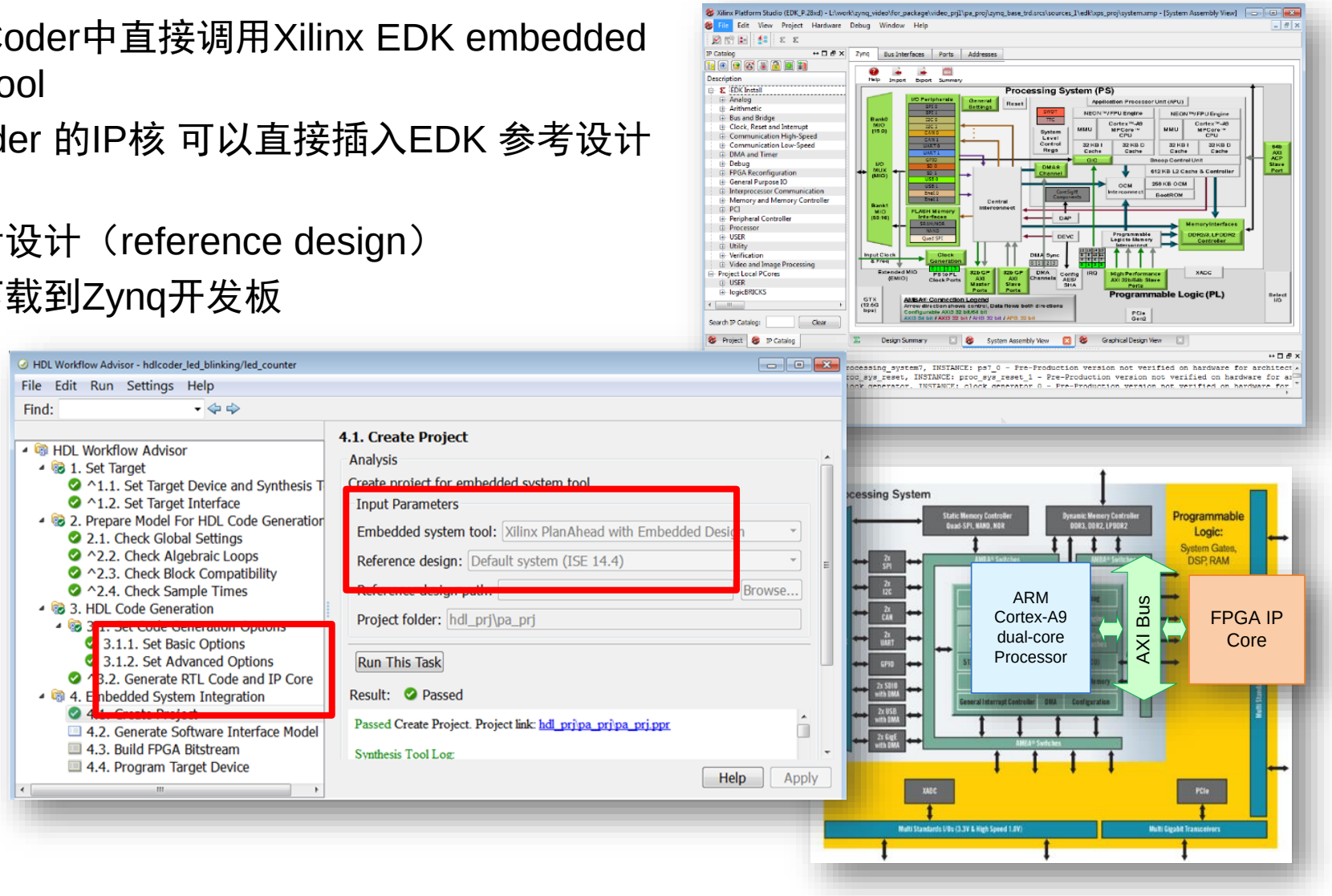
- 直接从MATLAB和Simulink产生可移植和复用的IP核
- 包含AXI4接口，直接连接 Altera SoC的ARM处理器
- 产生的报告文档a可做IP核数据表
- 与Altera的Qsys综合工具紧密集成



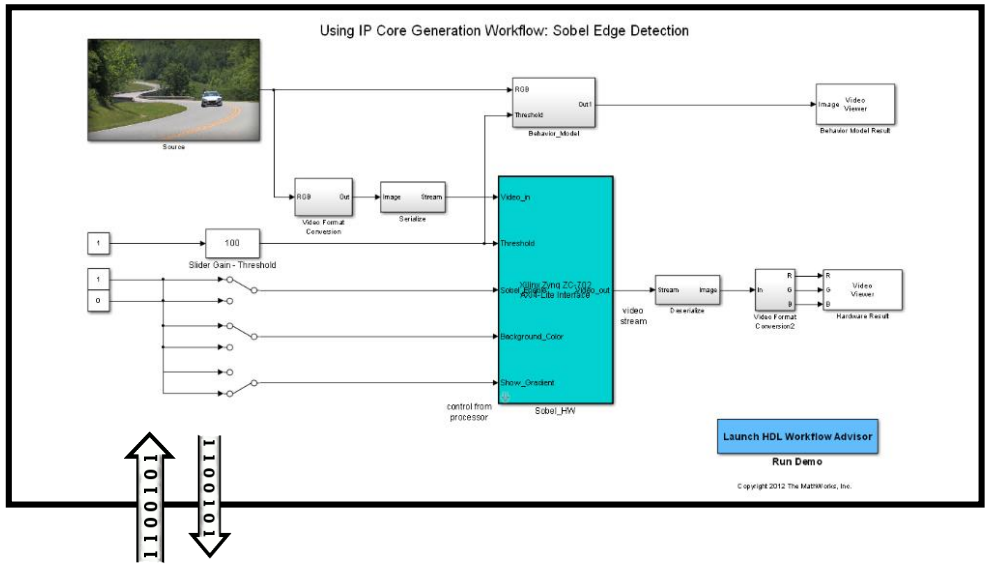


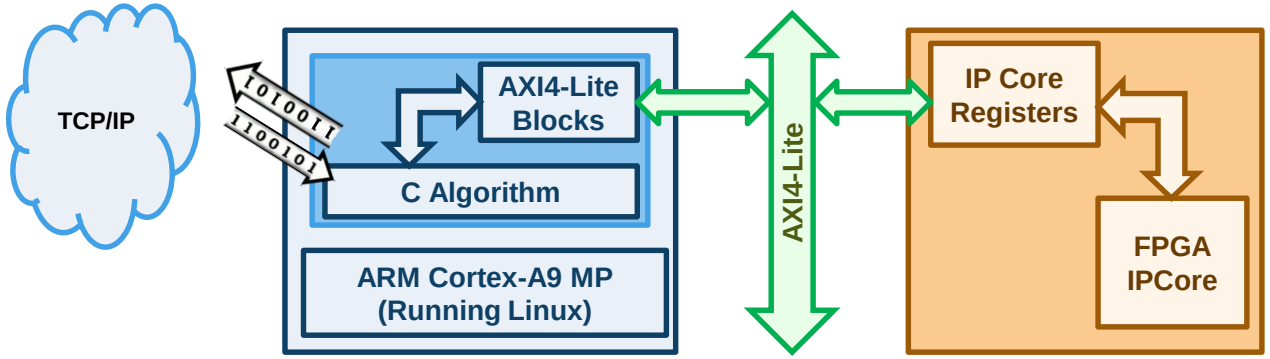
与Xilinx的 Zynq 设计工具紧密结合

- 在HDL Coder中直接调用Xilinx EDK embedded system tool
- HDL Coder 的IP核 可以直接插入EDK 参考设计使用
- 定制参考设计（reference design）
- 编译并下载到Zynq开发板



处理器在环仿真





- 在Simulink中实时控制硬件上跑的算法
- 实时调整参数
- 实时采集数据

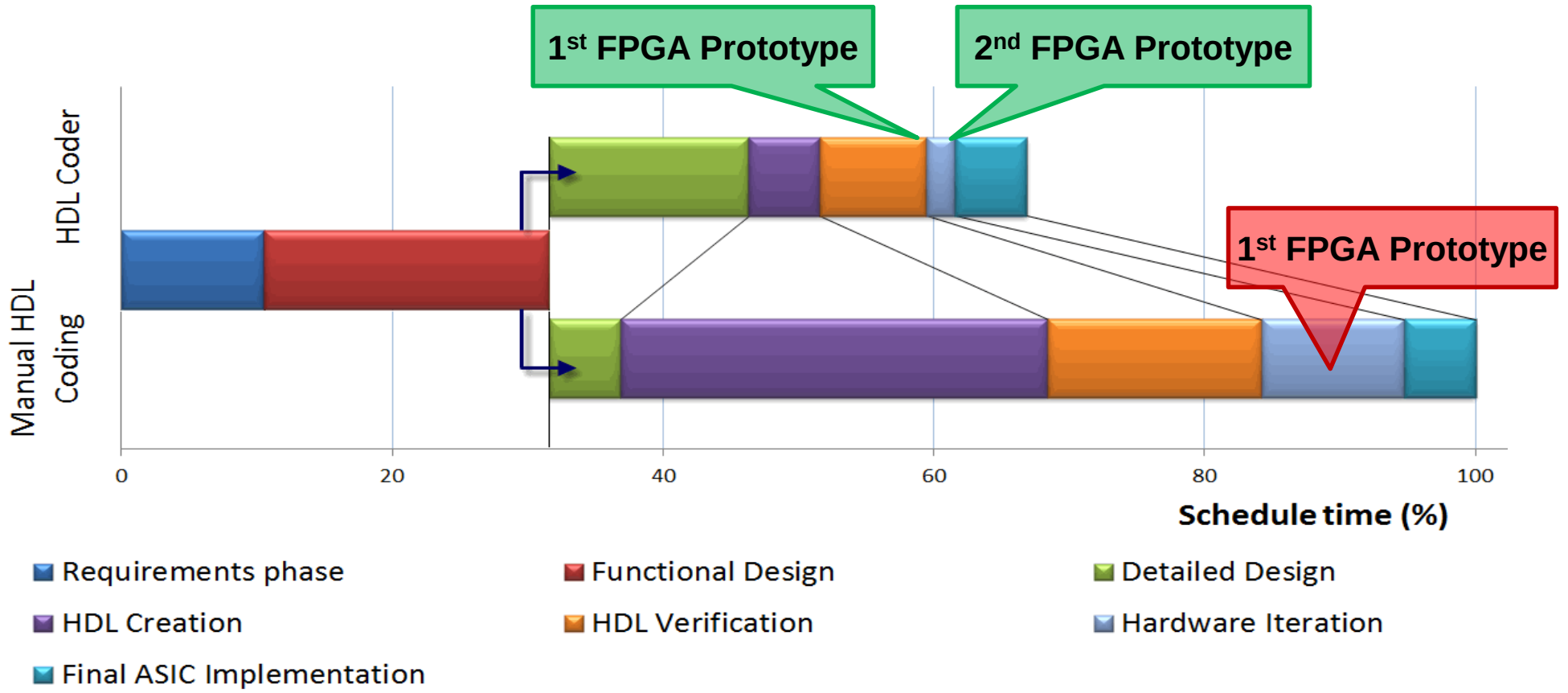
产品未来的走向是什么？

典型应用

- 图像处理
- 无线通信（5G)
- 机器学习和深度学习
- 自动驾驶
- 物理模型仿真

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

- FPGA实现所需时间缩短了48% (占项目的总耗时的33%)



问答