如何在 FPGA 上建立 MATLAB 和 Simulink 算法原型

芯片设计和验证工程师通常要为在硅片上实现的每一行 RTL 代码写出多达 10 行测试平台代码。验证任务在设计周期内可能会占用 50%或更多的时间。尽管如此辛苦，仍有接近 60%的芯片存在功能瑕疵，需要返工。由于 HDL 仿真不足以发现系统级错误，芯片设计人员正利用 FPGA 来加速算法创建和原型设计。

利用 FPGA 处理大型测试数据集可以使工程师快速评估算法和架构并迅速做出权衡。工程师也可以在实际环境下测试设计，避免因使用 HDL 仿真器消耗大量时间。系统级设计和验证工具（如 MATLAB 和 Simulink）通过在 FPGA 上快速建立算法原型，可以帮助工程师实现这些优势。

本文将介绍使用 MATLAB 和 Simulink 创建 FPGA 原型的最佳方法。这些最佳方法包括：在设计过程初期分析定点量化的效应并优化字长，产生更小、更高效的实现方案；利用自动 HDL 代码生成功能，更快生成 FPGA 原型；重用具有 HDL 协同仿真功能的系统级测试平台，采用系统级指标分析 HDL 实现方案；通过 FPGA 在环仿真加速验证（图 1）。

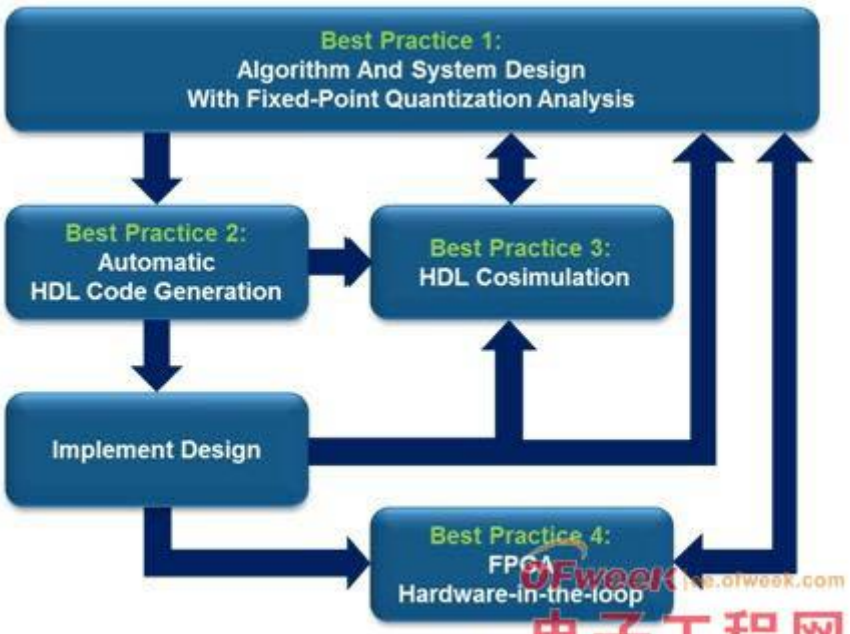

图1：基于模型设计FPGA原型开发最佳方法。

为什么在 FPGA 上建立原型？

在 FPGA 上建立算法原型可以增强工程师的信心，使他们相信自己的算法在实际环境中的表现能够与预期相符。除了高速运行测试向量和仿真方案，工程师还可以利用 FPGA 原型试验软件功能以及诸如 RF 和模拟子系统的相关系统级功能。此外，由于 FPGA 原型运行速度更快，可以使用大型数据集，暴露出仿真模型未能发现的缺陷。

采用 HDL 代码生成功能的基于模型的设计可以使工程师有效地建立 FPGA 原型，如图 2 所示。该图向我们展示了这样一种现实情况：工程师经常缩短详细设计阶段，试图通过尽快开始硬件开发阶段以符合开发周期的要求。现实中，当工程师发现定点算法达不到系统要求时，就得在 HDL 创建阶段重新审视详细设计阶段。这样的重叠工作将使 HDL 创建阶段延长（如紫色长条所示），并可能引发各种设计问题（如胶合逻辑或设计补丁）。

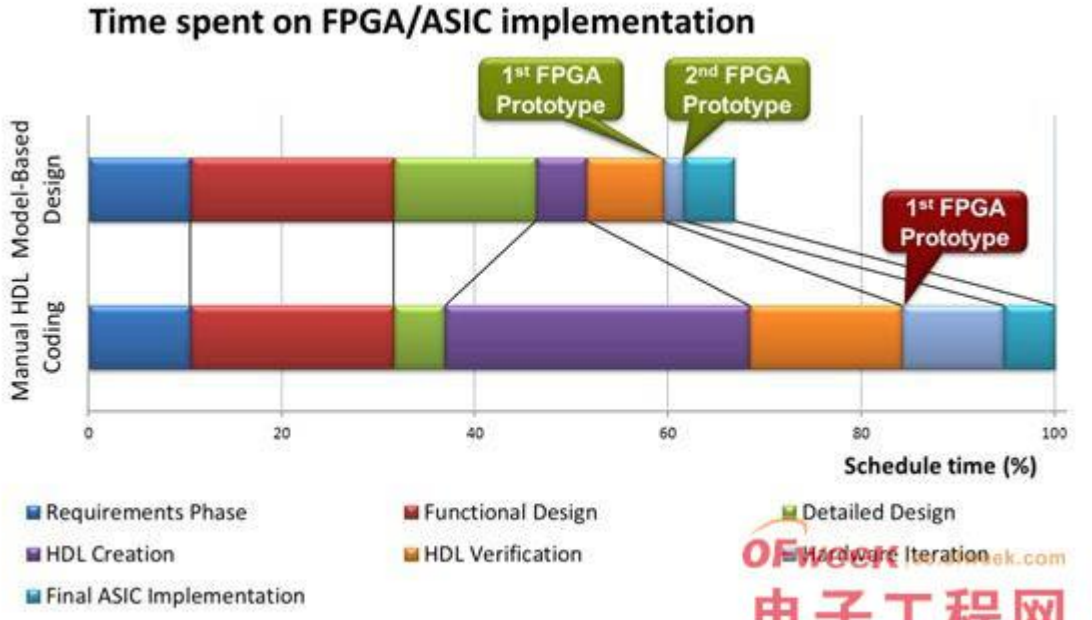

图2：建立FPGA原型和实现ASIC时，基于模型的设计与手动工作流程在时间进度上的比较。

由于自动 HDL 代码生成流程比手工编码快，工程师得以把节省下来的时间投入到详细设计阶段，生成更优质的定点算法。与手动的工作流程相比，这种方法使工程师能够以更快的速度生成质量更佳的 FPGA 原型。

数字下变频器案例研究

为了说明采用基于模型的设计建立 FPGA 原型的最佳方法，可借助数字下变频器（DDC）来进行案例研究。在众多的通信系统中，DDC 是一种普通的构建块（图 3）。该构建块用于将高速通带输入转换为低速基带输出，以便使用较低采样率时钟进行处理。这样，在硬件实施阶段便可降低功耗、节约资源。DDC 的主要部件包括：数控振荡器（NCO）、混频器和数字滤波器链路（图 4）。

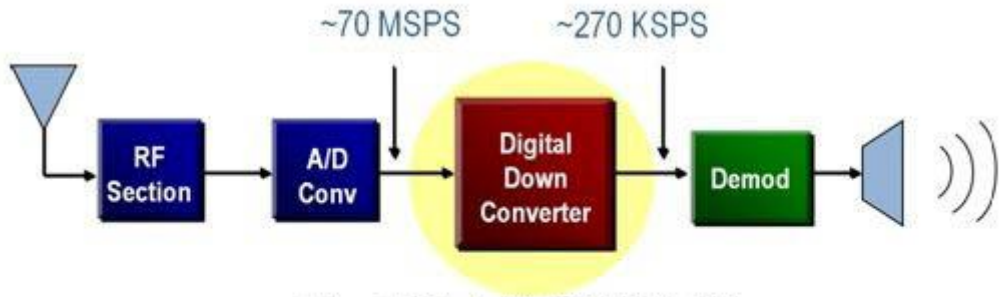

图3：采用数字下变频器的通信系统。

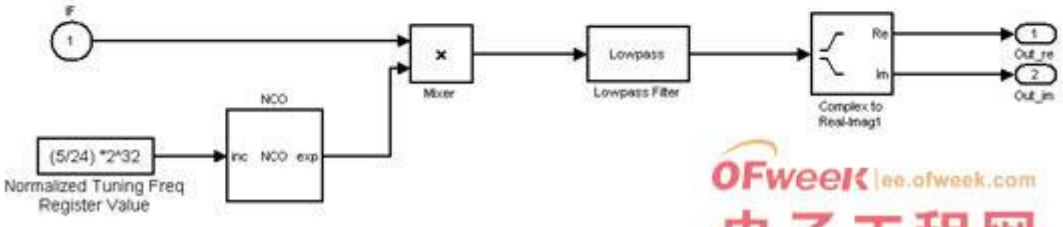

图4：数字下变频器系统模型。

在设计过程初期分析定点量化的效应

工程师通常使用浮点数据类型来测试新的构想和开发初始算法。然而，FPGA 和 ASIC 硬件实现要求转换为定点数据类型，而这往往会造成量化误差。使用手动工作流程时，通常在 HDL 编码过程中执行定点量化。在该工作流程中，工程师无法轻易地通过比较定点表示形式和浮点参考值量化定点量化的效应，而分析针对溢出的 HDL 实现也同样不易。

为了明智确定所需的小数位数，在开始 HDL 编码过程之前，工程师需要某种方法来比较浮点仿真结果与定点仿真结果。增加小数位数可以减小量化误差；不过，这种方法需要增加字长（区域增多、功耗升高）。

例如，图 5 展示了 DDC 滤波器链路中低通滤波器第一阶段浮点与定点仿真结果的差异。这些差异是因定点量化所致。上方图形显示了浮点与定点仿真结果的重叠效果。下方图形显示了图中每一点的量化误差。工程师可能需要根据设计规范来增加小数位数以减小由此引出的量化误差。

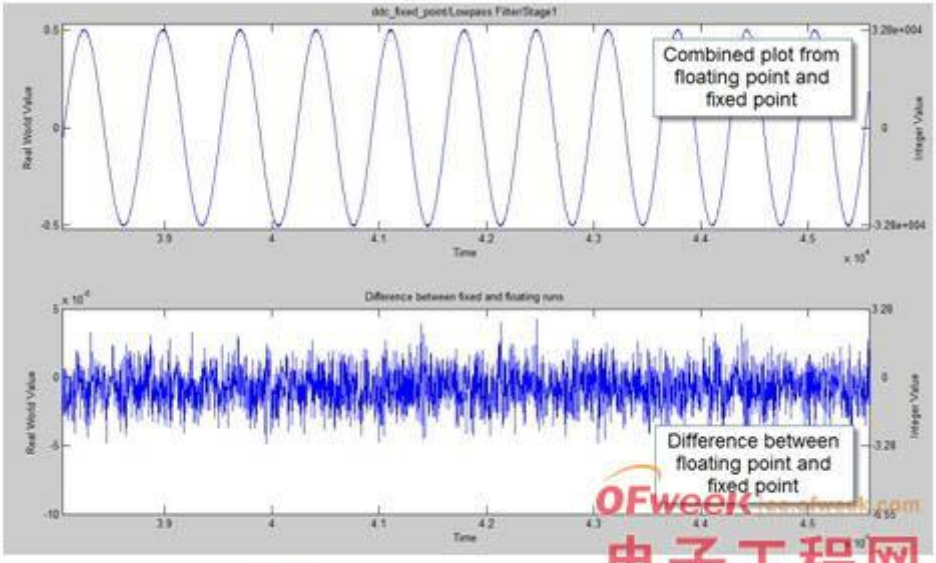

图5：使用Simulink定点模块组量化定点量化的效应。

除了选择小数位数之外，工程师还需要优化字长，实现低功耗和区域优化的设计。

在 DDC 案例研究中，工程师使用 Simulink 定点模块组将部分数字滤波器链路的字长减少了 8 位之多（图 6）。

图6：使用Simulink定点模块组优化定点数据类型。

利用自动 HDL 代码生成功能更快生成 FPGA 原型

在生成 FPGA 原型时，HDL 代码必不可少。工程师手工编写了 Verilog 或 VHDL 代码。作为替代选择，使用 HDL 编码器自动生成 HDL 代码具有众多明显优势。工

程师可以快速地评估能否在硬件中实施当前算法；迅速评估不同的算法实现，选择最佳方案；并在 FPGA 上更快地建立算法原型。

对于 DDC 案例研究而言，可以在 55 秒内生成了 5780 行 HDL 代码。工程师可以浏览并很快理解代码（图 7）。自动代码生成功能允许工程师对系统级模型进行更改，并且，通过重新生成 HDL 代码，该功能可以在数分钟之内生成更新的 HDL 实现方案。

```
BEGIN
  -- Count limited, Unsigned Counter
  --  initial value   = 0
  --  step value      = 1
  --  count to value  = 119
  --
  -- <S22>/Counter Limited
  Counter_Limited_process : PROCESS (clk)
  BEGIN
    IF clk'EVENT AND clk = '1' THEN
      IF reset = '1' THEN
        Counter_Limited_count <= to_unsigned(0, 8);
      ELSIF enb = '1' THEN
        IF Counter_Limited_count = 119 THEN
          Counter_Limited_count <= to_unsigned(0, 8);
        ELSE
          Counter_Limited_count <= Counter_Limited_count + 1;
        END IF;
      END IF;
    END IF;
  END PROCESS Counter_Limited_process;

  Counter_Limited_out1 <= Counter_Limited_count;

  -- <S22>/1-D Lookup Table

  alpha1_D_Lookup_Table_k <= to_signed(0, 31) WHEN Counter_Limited_out1 <= 0 ELSE
      to_signed(119, 31) WHEN Counter_Limited_out1 >= 119 ELSE
      signed(resize(Counter_Limited_out1, 31));
  alpha1_D_Lookup_Table_out1_re <= table_data_re(to_integer(alpha1_D_Lookup_Table_k));
  alpha1_D_Lookup_Table_out1_im <= table_data_im(to_integer(alpha1_D_Lookup_Table_k));
```

图7：利用Simulink HDL编码器生成的HDL代码实例

重用具有协同仿真功能的系统级测试平台进行 HDL 验证

功能验证：HDL 协同仿真使工程师能够重用 Simulink 模型，将激励驱动至 HDL 仿真器，并对仿真输出执行交互式系统级分析（图 8）。

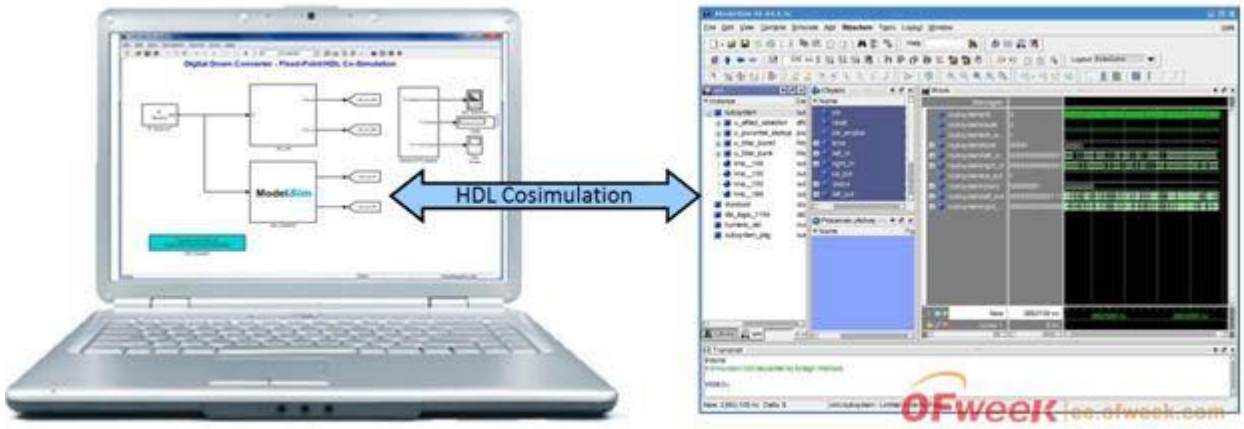

图8：Simulink和ModelSim间的HDL协同仿真。

HDL 仿真仅提供数字波形输出，而 HDL 协同仿真则提供了显示 HDL 代码的完整视图，并可以访问 Simulink 的全套系统级分析工具。当工程师观察到预期结果与 HDL 仿真结果存在差异时，可借助协同仿真进一步了解该失配所产生的系统级影响。

例如，在图 9 中，频谱仪视图可以使工程师做出明智决定，忽略预期结果与 HDL 仿真结果之间的失配，其原因是该差异位于阻带区。相比之下，数字波形输出只是将预期结果与 HDL 仿真结果的失配标记为误差。尽管工程师最终可能得出相同的结论，但这将需要更多的时间完成所需的分析。

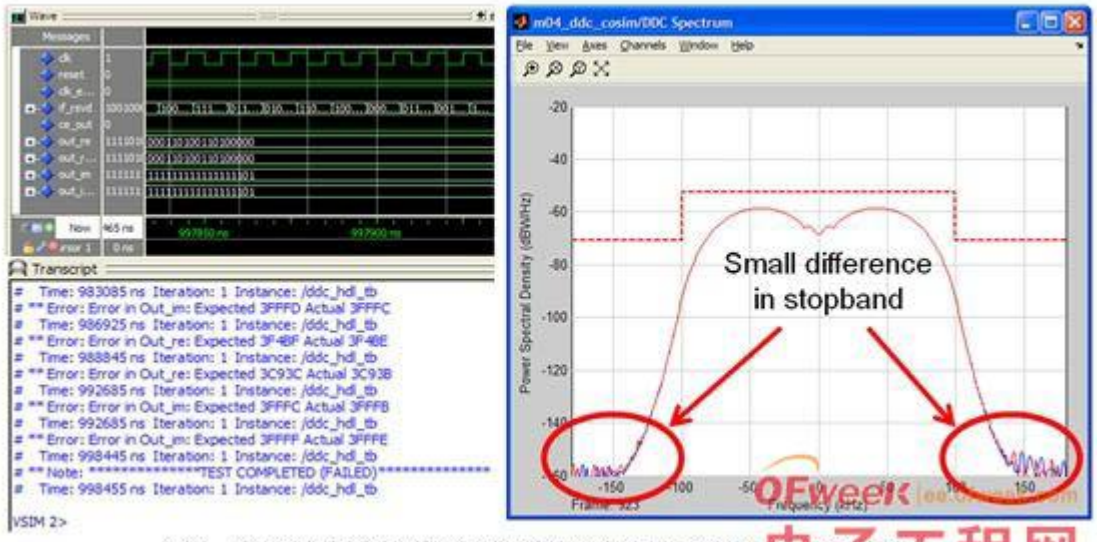

图9：使用特定域频谱仪分析系统级指标并评估HDL实现的性能。

测试覆盖率：工程师可以使用 HDL 验证工具、Simulink 设计验证工具和 ModelSim/Questa 自动执行代码覆盖率分析。在该工作流程中，Simulink 设计验证工具可针对模型覆盖率生成一套测试用例。HDL 验证工具自动使用这一套测试用例运行 ModelSim/Questa,收集代码覆盖率数据，以对生成的代码加以全面分析。

使用 FPGA 在环仿真加速验证

使用系统级仿真和 HDL 协同仿真验证 DDC 算法之后，便可以立即在 FPGA 目标平台上部署 DDC 算法。对算法执行基于 FPGA 的验证（也称为 FPGA 在环仿真）可以增强对算法在现实环境中有效运行的信心。相比基于主机的 HDL 仿真，该验证可以使工程师更快地运行测试方案。

对于 DDC 算法而言，可以使用 Simulink 模型驱动 FPGA 输入激励并分析 FPGA 的输出（图 10）。与 HDL 协同仿真一样，在 Simulink 中始终可以利用相关数据进行分析。

图10：使用Simulink和FPGA硬件执行FPGA在环仿真。

图 11 对比了 HDL 协同仿真和 FPGA 在环仿真这两种用于 DDC 设计的验证方法。在本案例中，FPGA 在环仿真的速度是 HDL 协同仿真的 23 倍。这样的速度提升使工程师能够运行更广泛的测试用例并对其设计进行回归测试。这使他们能够识别出有待进一步分析的潜在问题区域。

Verification Method	Simulation Performance	Visibility into HDL Code	System-Level Analysis Possible
HDL Cosimulation	46 sec	Yes	Yes
FPGA-in-the-Loop	2 sec	No	Yes

验证方法	仿真性能	HDL 代码的可见性	可能的系统级分析
HDL 协同仿真	46 秒	是	是
FPGA 在环	2 秒	否	是

图11：HDL协同仿真与FPGA在环仿真在DDC设计验证上的对比。

尽管 HDL 协同仿真速度较慢，但它却提高了 HDL 代码的可见性。因此，它很适合针对 FPGA 在环仿真过程中发现的问题区域进行更详细的分析。

本文小结

如果工程师遵循本文所述的四种最佳方法，开发 FPGA 原型将比传统的手动工作流程快出许多，并能使工程师信心倍增。此外，工程师还可以在整个开发过程中继续优化自己的模型，并快速地重新生成有关 FPGA 实现的代码。与依赖手工编写 HDL 的传统工作流程相比，这种能力可以显着缩短设计迭代的周期。