快速通往量产的四个步骤：利用基于模型的设计开发软件定义无线电

第四部分—利用 Zynq SDR 套件和 Simulink 代码生成工作流程快速完成原型开发

作者：Mike Donovan、Andrei Cozma 和 Di Pu

简介

本系列文章的前几部分介绍了Zynq SDR快速原型开发平台[1]，说明了利用MATLAB和Simulink开发算法以成功处理和解码ADS-B传输的步骤[2]，并展示了如何在仿真中和利用SDR平台获得的实时数据验证该算法[3]。所有阶段的最终目标是创建一个经验证的模型，其可以转换为C和HDL代码，并且能够方便地集成到SDR平台的软件和硬件基础设施中。

本系列第二部分（“利用MATLAB和Simulink进行S模式检测和解码”）[2]讨论的Simulink模型是一个具有足够高精度硬件细节的仿真模型，可验证该设计将能成功解码ADS-B消息。以该模型为出发点，本部分将讨论为了产生一个能够在Zynq SDR快速原型开发平台上运行的有效接收机设计所需的最后步骤。像前面几篇文章一样，开发该有效设计所需的技能包括：熟练使用MATLAB和Simulink，了解Zynq无线电硬件，以及软硬件集成技能。

本文提出的步骤包括：

- 以Zynq SoC上的FPGA结构和ARM®处理系统为目标，将Simulink模型划分为多个功能。
- 引入对Simulink模型的设计变更，以改善所生成的HDL代码的性能。
- 生成ADS-B接收机算法的HDL和C语言源代码。
- 将生成的源代码集成到Zynq无线电平台设计中。
- 在目标硬件上利用实时航空器信号测试该嵌入式设计。

此过程结束时，就会产生一个经全面验证的SDR系统，其运行从Simulink ADS-B模型自动生成的C和HDL代码，可实时接收和解码商用航空器信号。

将模型划分为硬件和软件组件

生成实现代码过程的第一步是划分设计功能，以便在Zynq SoC的可编程逻辑和ARM处理系统上运行。

功能划分通常是从明确设计的不同组件的处理要求和所需的执行速率与时间开始。需要以采样速率实时运行的计算密集型组件（如数据调制/解调算法），最适合在可编程逻辑中实现。计算量相对较少的处理任务（如数据解码和渲染，以及系统监视和诊断），更适合通过软件实现。其它需要考虑的方面有：运算的数据类型和复杂度，以及输入和输出数据的精度。所有以可编程逻辑为目标的运算都采用定点、整数或布尔数据类型。对于更复杂的运算，如三角函数和平方根，须在可用硬件资源的约束下，利用近似来高效实现。所有这些约束都会导致精度损失，若不加以适当评估和处理，可能会对系统功能带来不利影响。但是，以处理系统为目标的组件可以采用浮点数，并以最高保真度实现任何复杂度的运算，不过通常要以降低执行速度为代价。

以上述约束作为指导原则，ADS-B解码算法的划分是相当明显的。ModeS_Simulink_Decode.slx模型中的检波器模块的功能，包括I/Q样本的前端处理一直到校验和计算，非常适合在Zynq SoC的可编程逻辑上实现（图1）。改良缓冲器和解码与显示模块中的消息位解码功能，很容易在处理系统中实现。

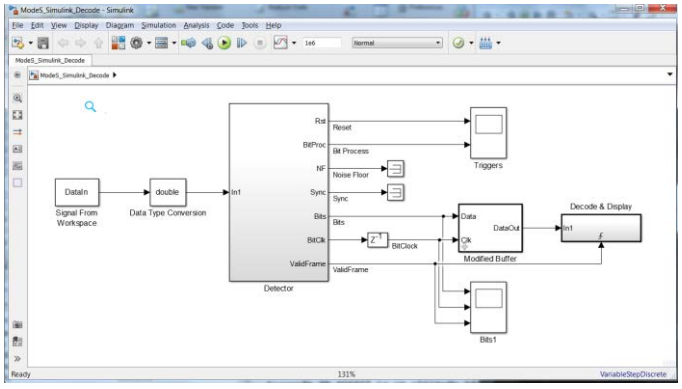


图1. ModeS_Simulink_Decode.slx：FPGA和ARM处理器划分

对下述内容和Simulink模型感兴趣的读者，可在Analog Devices GitHub库[4]中找到相关文件。

从Simulink模型生成HDL代码

S模式解码器模型中的检波器模块（图2）包括多个子系统：CalcSyncCorr、CalcNF、SyncAndControl、BitProcess、CalcCRC和FameDetect。MathWorks的HDL编码器[5]用于产生此设计的HDL源代码。

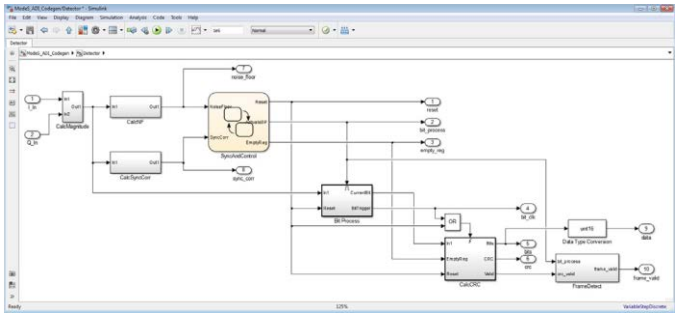


图2. 用于HDL代码生成的检波器模块

为了利用HDL编码器成功生成HDL代码，Simulink模型必须满足一些条件。下面是其中几个最重要的要求：

- 使用支持HDL代码生成的模块。HDL编码器支持大约200个Simulink模块的代码生成[6]。在检波器设计中，所有模块都支持HDL代码生成，包括状态流程图和数字滤波器模块。
- 使用定点数据类型。在检波器设计中，信号使用12位、24位和布尔数据类型。12位数据类型与ADI公司AD9361收发器上的模数转换器的位宽一致。
- 使用标量或矢量信号。矢量信号可用于多通道信号或资源共享。
- 避免模型中出现代数环。HDL编码器软件不支持存在代数环条件的模型的HDL代码生成。

ModeS_Simulink_Decode.slx模型并不满足所有这些条件，因此将比较接收位与计算校验和的CalcCRC模块部分移出检波器模块，最终用C语言实现。由此得到的模型ModeS_ADI_CodeGen.slx用于生成HDL代码。与手动编码过程相比，它只需几分钟便能生成数千行HDL代码。HDL编码器产生的源代码是Simulink模型的位真、周期精确版本。这是使用模型进行设计在生产力提升方面带来的重大好处之一，所生成的代码是Simulink模型的精确转译。

此外，这些代码易读且可追溯，工程师可以轻松地将生成的代码映射到设计模型。这是通过多种方法来实现的（图3）：

- 生成的HDL代码文件中保留了该模型的层次。本例中，顶层模块被命名为Detector.vhd，处在下一级的子系统被命名为CalcNF.vhd、Bit_Process.vhd等。
- 生成的代码中保留了该模型所用的模块名称、端口名称、信号名称、数据类型和复杂度。

模型与源代码之间存在关联，设计人员点击Simulink模型中的某个模块，便可自动导航到相应的HDL代码。同样，生成的代码中也有超级链接，点击它便会打开Simulink模型，并高亮显示与该代码段相关的模块。

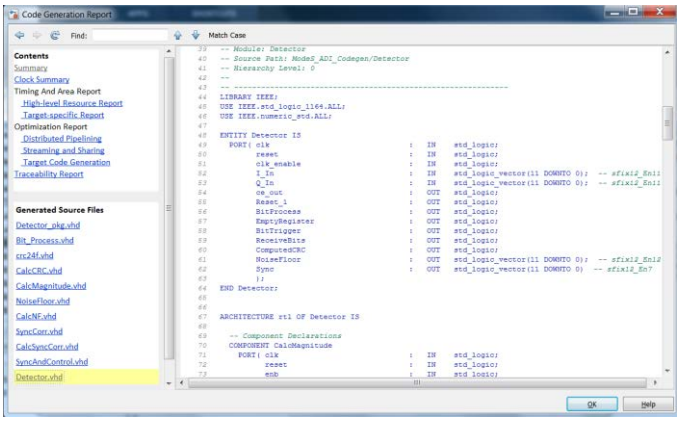


图3. ModeS_ADI_CodeGen.slx的HDL源代码

优化ADS-B模型以产生具有更高时钟速度的HDL代码

虽然ModeS_ADI_CodeGen.slx模型成功生成了HDL代码，但在绝大多数情况下，设计人员会希望改善初始结果。设计人员通常要满足速度和面积约束条件，这就需要优化初始Simulink模型以实现所需的效果。Simulink和代码生成的一个重要优势是，设计人员可以在模型中进行优化，并运行仿真以确保变更没有破坏算法，然后重新生成HDL代码。这种方法比修改HDL源代码（可能会破坏算法）更为简单，而且不易出错。

对于本设计，模型生成的HDL代码很容易适应可用的FPGA结构，但运行的时钟速率相对较低。这在许多初步设计中是常见现象。HDL编码器内置的分析工具表明，模型中的关键路径从I/Q样本输入延伸到CalcCRC子系统中的第一个寄存器。在设计中插入流水线寄存器是提高时钟速度的一种常见方法（图4）。流水线缩短了信号操作之间的路径，代价是增加了整体处理的延迟。这种折中通常是可以接受的，因为相对于更高的时钟速度，延迟略微增加是很小的代价。

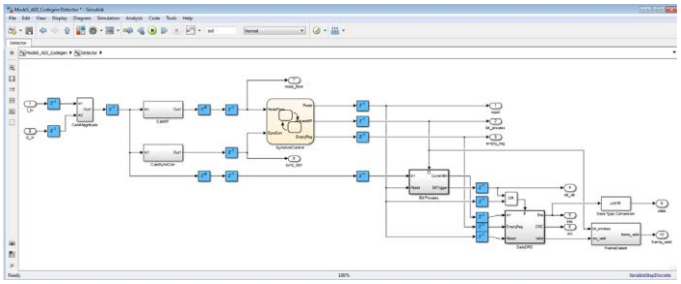


图4. 检波器设计中插入流水线寄存器

位于子系统之间的流水线寄存器有助于改善设计的时钟速率，但如果数字滤波器模块选择优越的架构，则可以实现更好的时钟速率。许多Simulink模块都有架构选择，设计人员可以藉此优化设计的速度或面积。对于计算噪底和前同步码相关性所用的数字滤波器（图5），输出乘法器的流水线化可以缩短数字滤波器内的关键路径，提高设计时钟速率。

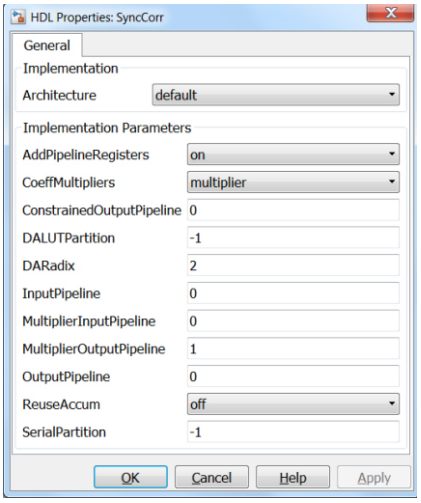


图5. 数字滤波器模块的HDL模块选择

采取这两个简单的流水线变更之后，生成的HDL代码的时钟速率便超过140 MHz。这对于使用代码生成工具的工程设计是一个有用的启示：在代码生成模型上应用一点硬件设计原理知识，便可对生成的代码结果产生相当大的影响。对该设计还可以做进一步的优化，但并无必要，因为HDL代码很容易满足该设计相对简单的时序和资源目标。

在传统无线电设计过程中，大量开发时间花在HDL代码的测试和调试上。而在基于模型的设计方法中，更多时间是花在开发仿真和代码生成模型上，本例就是如此。然而，开发时间会节省很多，因为生成的源代码与经验证的仿真行为完全一致，只需对嵌入式硬件执行极少量的调试。

利用MATLAB编码器[7]生成C语言代码

与生成HDL代码相似，为了生成用于本设计解码功能的C语言代码，也有几个条件必须满足。下面是两个最重要的要求：

- 使用MATLAB编码器支持的函数。MATLAB编码器支持大部分MATLAB语言和众多工具箱[8]，但您可能无意中使用了代码生成所不支持的函数。MATLAB编码器提供了“代码就绪工具”[9]等工具来帮助找出不支持的函数。
- 一旦声明一个MATLAB变量，其大小和类型便不得改变。这是为了确保在生成的代码中正确分配存储器。

从MATLAB生成C代码的最简单方法是从MATLAB工具列上的Apps选项卡新建一个MATLAB编码器项目。MATLAB编码器项目的最终输出如图6所示。

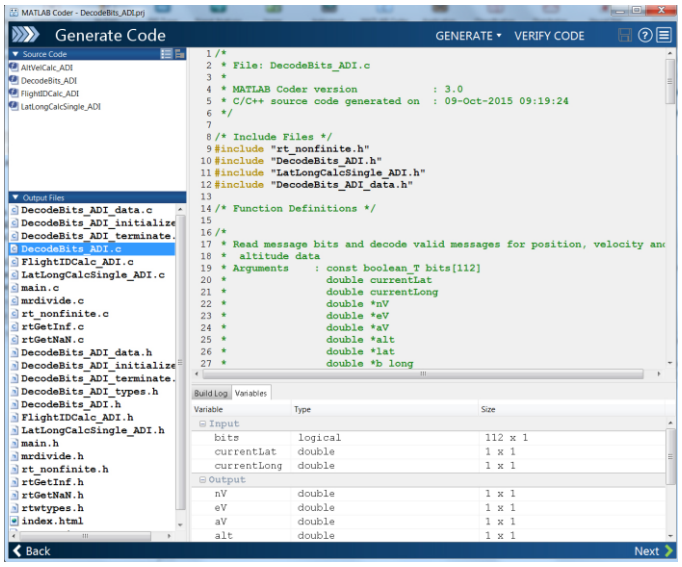


图6. 针对的DecodeBits_ADI.m的MATLAB编码器项目

在该项目中，顶层MATLAB函数是DecodeBits_ADI.m。用户需要指定此函数所需的数据类型和大小作为输入参数。图6显示，此函数的输入参数为112个布尔数据位和2个双精度值（用以提供用户当前的经度和纬度）。DecodeBits_ADI.m的输出大小和数据类型（例如：*nV表示向北速度，*eV表示向东速度，*alt表示高度）由MATLAB编码器自动确定。MATLAB编码器会找出顶层入口文件DecodeBits_ADI.m调用的所有其他函数，包括AltVelCalc_ADI.m和LatLongCalc_ADI.m，然后生成整个解码算法的C语言源代码。

MATLAB编码器生成的C语言代码是MATLAB功能的直接转译。如同HDL代码生成，MATLAB编码器产生的源代码也是易读且可追溯的，工程师可以轻松发现原始MATLAB代码与生成的C代码之间的关系。本例的C代码可从MATLAB命令提示产生，并且可由任何ANSI C编译器编译。

HDL代码平台部署

完成上述步骤（将设计划分为不同功能以在Zynq的可编程逻辑和处理系统上运行，针对HDL和C语言代码生成优化设计，以及通过仿真验证优化后的设计能够有效工作且满足性能标准）之后，现在便需要将设计部署到实际SDR硬件平台上，并验证系统在实际条件下的功能。

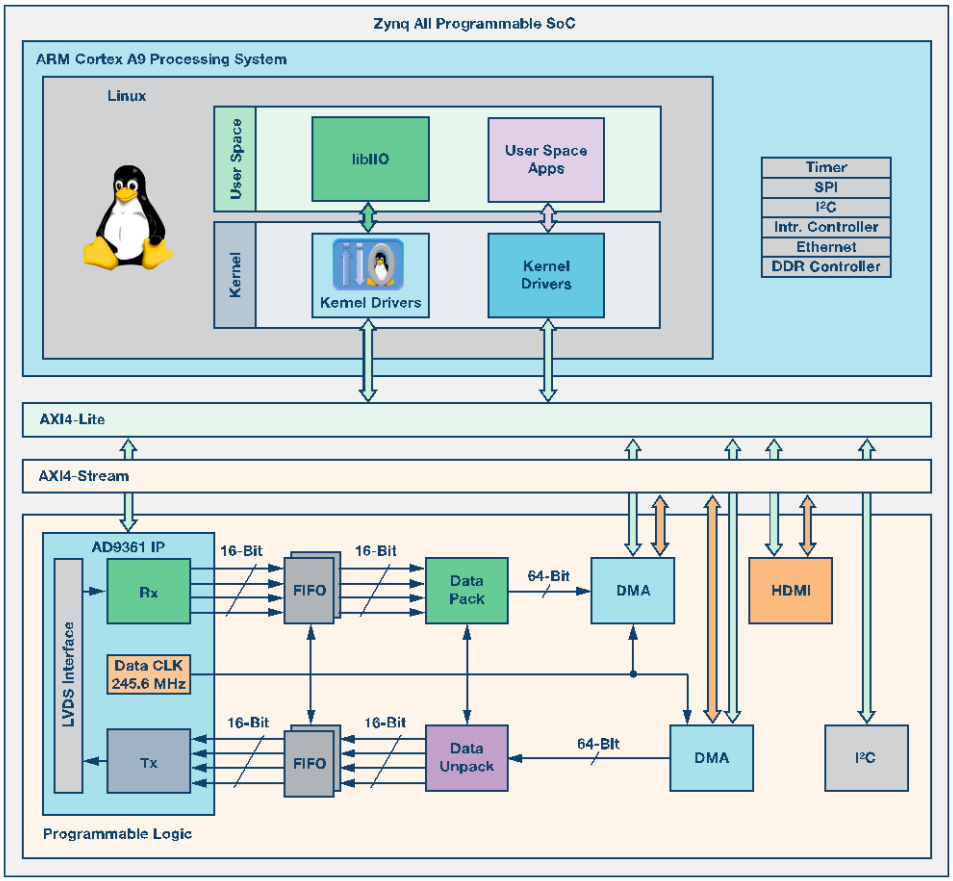


图7. HDL参考设计框图

为此，我们使用ADI公司的AD-FMCOMMS3-EBZ SDR平台[10]，其连接到一个运行ADI Linux发行版的Xilinx ZC706板[11]。

AD-FMCOMMS3-EBZ板附带一个ADI公司提供的开源Vivado HDL参考设计[12]。该参考设计包含用以配置AD-FMCOMMS3-EBZ板上的AD9361收发器及传输数据所需的全部IP模块。图7为该HDL参考设计的框图。

AD9361 IP内核实现了AD9361收发器芯片与Zynq器件之间的LVDS接收和发送数据接口，以及与设计其余部分的数据接口。DMA模块用于AD9361 IP与DDR存储器之间的高速数据传输。AD9361 IP模块的数据接口包括4条用于接收的数据线和4条用于发送的数据线，对应于AD9361的两个接收通道和两个发送通道的I&Q数据。每条数据线都是16位宽。为使系统内部的数据传输效率更高，接收和发送数据被包装到由DMA模块管理的64位宽总线中。AD9361 IP的16位并行数据线通过打包和解包模块连接到DMA。

为了将ADS-B模型的HDL代码部署到SDR平台的现有HDL基础设施中，需要创建一个可插入数据路径中的IP内核，从而实时处理收到的数据并将处理过的数据传送到软件层。部署过程是一个困难且耗时的任务，因为它要求对HDL设计的功能有深刻的理解，同时需要娴熟的HDL编程技能。为了简化这些步骤，MathWorks在HDL编码器中集成了一个称为HDL Workflow Advisor的实用工具，ADI公司为AD-FMCOMMS2-EBZ/AD-FMCOMMS3-EBZ SDR平台和Xilinx ZC706板提供了一个板支持包 (BSP) [13]。

HDL Workflow Advisor可引导用户一步一步地从Simulink模型生成HDL代码。用户可以选择不同的目标工作流程，包括“ASIC/FPGA”、“FPGA在环”和“IP内核生成”。目标平台选择包括Xilinx评估板、Altera评估板和FMCOMMS2/3 ZC706 SDR平台。余下的代码生成和目标集成过程可由HDL Workflow Advisor自动完成。

ADI公司提供的BSP是板定义和参考设计[14]的集合，用以为HDL Workflow Advisor提供必要的信息和工具来产生与现有HDL参考设计兼容的IP模块，以及将生成的IP插入HDL参考设计。图8显示了如何配置Workflow Advisor来产生ADS-B模型的IP内核。请注意：必须选择IP内核生成工作流程，并以ADI公司的AD-FMCOMMS3-EBZ SDR平台和Xilinx ZC706板为目标。

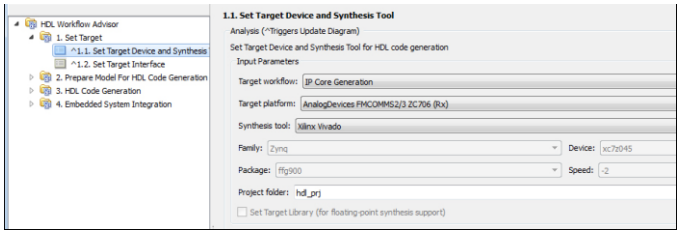


图8. Workflow Advisor配置

下一步是配置IP与参考设计之间的接口。在输入侧，该模型接受原始I&Q样本，模型的输入端口与AD9361接收器数据端口直接相连。在模型的所有输出信号中，当前阶段感兴趣的是数据、frame_valid和bit_clk信号。数据和frame_valid为16位宽，由bit_clk信号提供时钟。这些信号可以连接到BSP的“DUT Data x Out”接口，意味着它们将接收对DMA模块的直接访问；然后可以将数据传输到DDR供软件层访问。bit_clk信号连接到BSP的“DUT Data Valid Out”接口，用以控制DMA采样速率。图9显示了HDL接口必须如何配置。

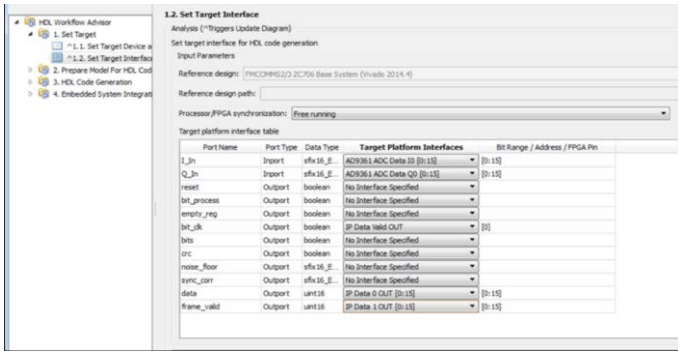


图9. HDL接口配置

一旦定义了目标接口，HDL Workflow Advisor的第2步和第3步便可保持默认状态，然后通过运行步骤4.1（创建项目）来启动项目生成过程。此步骤的结果是产生一个Vivado项目，其ADS-B IP内核已集成到ADI公司的HDL参考设计中。图10显示了ADS-B IP内核与设计中其余模块的连接。

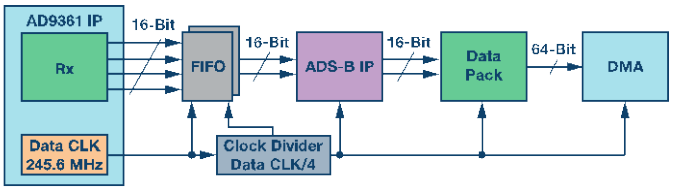


图10. HDL参考设计中的ADS-B IP连接

从Vivado项目生成位流便可结束HDL集成过程，但最终目标是让Linux在系统上运行。为此，在生成位流之后，可按照标准Xilinx SDK第一阶段引导加载程序（fsbl）和Linux引导文件创建过程创建一个Linux引导文件。与新创建HDL设计对应的Linux设备树和映像文件随同AD-FMCOMMS3-EBZ BSP发布。所有文件都必须与SD卡引导分区上的Linux引导文件一同复制；引导分区用于存储在Xilinx ZC706板上运行ADI公司Linux发行版所需的全部文件。

C语言代码平台部署

将ADS-B HDL IP集成到SDR平台的HDL设计并创建Linux SD卡之后，便需要实现用来解码ADS-B数据的软件应用程序。此应用程序基于第5部分生成的C代码，执行如下任务：

- 配置AD9361以接收ADS-B信号。
- 从ADS-B IP内核读取数据。
- 在读取的数据中检测有效ADS-B帧。
- 解码并显示ADS-B信息。

实现任务1和任务2的最简单方法是使用libiio库[15]提供的功能。此库提供了接口函数，允许用户轻松配置AD9361以及接收和发送数据。配置过程设置如下系统参数：

- LO频率—1.09 GHz
- 采样速率—12.5 MHz
- 模拟带宽—4.0 MHz
- AGC—快速启动模式

除上述参数外，一个数据速率为12.5 MSPS、通带频率为3.25 MHz、阻带频率为4 MHz的数字FIR滤波器也被加载到AD9361中，确保收到的数据仅包含目标频段。该FIR滤波器的系统参数和设计方法详见本系列文章第三部分[3]所述。

ADS-B IP的输出数据通过DMA模块传输到系统的DDR存储器。libiio库提供如下功能：将从ADS-B IP获取的数据放置到指定大小的存储缓冲器中；等待缓冲器填满；通过指针访问该缓冲器。一旦缓冲器填满，ADS-B解码算法便可处理数据。ADS-B IP内核有两个输出通道：一个通道对应于ADS-B位流，另一个通道指示一个有效数据帧在位流中的何处结束。两个通道均包含相同的数据速率，彼此同步。有效通道中一个等于1的样本表示数据通道中一个有效帧的最后一位。通过解析这两个通道，软件可以从位流中提取有效的ADS-B数据帧，并将数据传送到MATLAB编码器生成的解码函数。当计算航空器坐标时，解码函数利用ADS-B数据帧和当前位置的经纬度作为输入。当前经纬度被指定为应用程序的参数。ADS-B解码数据的显示与Simulink模型相似。

ADS-B数据解码程序是在Linux下利用makefile构建。该应用程序的源代码和makefile可在Analog Devices github库中下载[16]。

这样就完成了利用HDL编码器从ADS-B模型生成的HDL代码和利用MathWorks MATLAB编码器生成的C代码的平台部署步骤。下一步是验证系统功能并评估结果。

系统验证

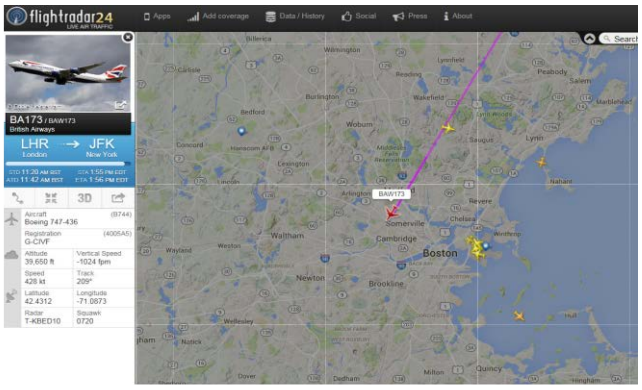
为了验证系统功能，首先要在AD-FMCOMMS3-EBZ板的一个接收端口与一个发送端口之间建立一个回送连接，并发送仿真期间使用的相同ADS-B信号。通过接收和解码此数据，可以验证SDR平台上运行的算法输出是否与仿真结果一致。图11显示了ADS-B数据解码程序的输出，结果与本系列文章第三部分中利用预先捕捉的数据进行HDL仿真所获得的结果完全相同。这说明系统运行符合预期，可以利用实际数据进行测试。

```
Aircraft ID: 400927 is at altitude 39000
Aircraft ID: 400927 is at latitude 42.324, longitude -71.143

Aircraft ID: 400927 is travelling at 468.363107 knots
Direction West at 230.000000 knots, direction South at 408.000000 knots
Aircraft ID: 400927 is going Up at 0.000000 feet/min
```

图11. 回送结果

现场实际测试时，SDR接收机放在MathWorks位于美国马萨诸塞州纳蒂克的总部外面，系统解码的ADS-B信息与飞机实时跟踪网站（如flightradar24.com）提供的数据进行对比。结果证实：在天线的视线范围内，系统能够解码从飞机收到的数据。图12显示了系统检测到的航空器信息与在线飞机跟踪数据的对比情况。可以看到，解码算法给出了正确的航空器ID、高度、速度和经纬度坐标。




```
Aircraft ID: 4005a5 is at altitude 40300
Aircraft ID: 4005a5 is at latitude 42.398, longitude -71.112

Aircraft ID: 4005a5 is travelling at 427.375713 knots
Direction West at 205.000000 knots, direction South at 375.000000 knots
Aircraft ID: 4005a5 is going Down at 1216.000000 feet/min

Aircraft ID: 4005a5 is at altitude 40275
Aircraft ID: 4005a5 is at latitude 42.396, longitude -71.113

Aircraft ID: 4005a5 is at altitude 40250
Aircraft ID: 4005a5 is at latitude 42.393, longitude -71.115

Aircraft ID: 4005a5 is travelling at 428.253430 knots
Direction West at 205.000000 knots, direction South at 376.000000 knots
Aircraft ID: 4005a5 is going Down at 1344.000000 feet/min

Aircraft ID: 4005a5 is at altitude 40150
Aircraft ID: 4005a5 is at latitude 42.386, longitude -71.121

Aircraft ID: 4005a5 is at altitude 40025
Aircraft ID: 4005a5 is at latitude 42.375, longitude -71.128
```

图12. 实时数据结果

结论

本系列文章展示了如何利用基于模型的设计来实现SDR平台从仿真到生产的全过程，这是其中的最后一篇。本系列说明了开发一个“硬件准备就绪”的ADS-B Simulink模型的所有阶段。我们设计了一个仿真模型来证明我们能够解码记录到的ADS-B消息，然后利用从SDR硬件平台获取的实时数据验证该模型。这不仅验证了该模型，而且验证了SDR平台的模拟前端和数字接收机信号链的设置。同时，它令我们确信该平台已调整好，可用于接收ADS-B信号。然后，我们将该模型划分为不同的功能，以便在Zynq处理系统和可编程逻辑上运行，并优化了该模型以自动生成C和HDL代码。最后，我们将C和HDL代码集成到SDR设计中，并利用实时商业空中交通数据验证了系统的功能。最终成果是一个设计流程—使用MathWorks建模和代码生成工具，并结合Zynq SDR平台来创建全面有效的SDR系统。

示例系统说明：相比于传统设计方法，基于模型的设计工作流程与ADI公司的集成RF捷变收发器可编程无线电硬件AD9361/AD9364相结合，可以帮助设计团队更快开发出有效的无线电原型，成本也更低。文中的原型是由笔者在相对较短的时间内制作出来的，遇到的障碍极少，使用了如下资源：

- 在MATLAB和Simulink中能够创建ADS-B接收机模型，并生成可用的C和HDL源代码。
- HDL Workflow Advisor中的功能，它们使很多软硬件集成步骤自动完成。
- libiio等库，帮助完成其余集成步骤以便部署SDR原型。
- MathWorks和ADI公司提供的产品帮助和技术支持。

ADS-B是一个相对简单的标准，为通过这种方法构建SDR原型提供了一个很好的测试案例。采用基于模型的设计和Zynq SDR平台的工程师应当能够按照本系列文章所提出的工作流程，开发出更复杂、更强大的QPSK、QAM和LTE SDR系统。

参考文献

[1] Di Pu、Andrei Cozma和Tom Hill，“快速通往量产的四个步骤：利用基于模型的设计开发软件无线电，第一部分—ADI/Xilinx SDR快速原型开发平台及其能力、优势和工具”，*模拟对话*，第49卷，第3期。

[2] Mike Donovan、Andrei Cozma和Di Pu，“快速通往量产的四个步骤：利用基于模型的设计开发软件无线电，第二部分—利用MATLAB和Simulink进行S模式检测和解码”，*模拟对话*，第49卷，第4期。

[3] Di Pu和Andrei Cozma，“快速通往量产的四个步骤：利用基于模型的设计开发软件无线电，第三部分—利用硬件在环验证S模式信号解码算法”，*模拟对话*，第49卷，第4期。

[4] Analog Devices GitHub库。

[5] HDL编码器。

[6] HDL编码器模块支持。

[7] MATLAB编码器。

[8] MATLAB工具箱。

[9] MATLAB代码生成就绪工具。

[10] AD-FMCOMMS3-EBZ用户指南。

[11] Xilinx Zynq-7000 All Programmable SoC ZC706评估套件。

[12] AD-FMCOMMS2-EBZ/AD-FMCOMMS3-EBZ/AD-FMCOMMS4-EBZ HDL/AD-FMCOMMS5-EBZ HDL参考设计。

[13] Analog Devices BSP for MathWorks HDL Workflow Advisor。

[14] 电路板和参考设计注册系统。

[15] 什么是Libiio?

[16] MathWorks目标模型—ADSB。

Mike Donovan [mike.donovan@mathworks.com]是 MathWorks 公司应用工程部门经理。他拥有巴克内尔大学电气工程学士学位和康涅狄克大学电气工程硕士学位。加入 MathWorks 之前，Mike 开发过雷达和卫星通信系统，并在宽带电信行业工作过。

Mike Donovan

Andrei Cozma [andrei.cozma@analog.com]是 ADI 公司工程设计经理，负责支持系统级参考设计的设计与开发。他拥有工业自动化与信息技术学士学位及电子与电信博士学位。他参与过电机控制、工业自动化、软件定义无线电和电信等不同行业领域的项目设计与开发。

Andrei Cozma

该作者的其它文章：

基于FPGA的系统提高电机控制性能
第49卷第1期

Di Pu [di.pu@analog.com]是 ADI 公司系统建模应用工程师，负责支持软件定义无线电平台和系统的设计与开发。她与 MathWorks 密切合作解决双方共同客户的难题。加入 ADI 公司之前，她于 2007 年获得南京理工大学 (NJUST) 电气工程学士学位，于 2009 年和 2013 年分别获得伍斯特理工学院 (WPI) 电气工程硕士学位和博士学位。她是 WPI 2013 年博士论文 Sigma Xi 研究奖获得者。

Di Pu