1 OpenSoC 简介

1.1 概述

OpenSoC Fabri 是一个参数化的片上网络生成器工具, 用于评估大规模多核处理器和 SoC. OpenSoC Fabri 有一组层次化的模块组成, 使用 Chisel 语言描述. 每个模块都有一个抽象接口定义, 包括输入端口/输出端口/公用函数等. 而且, 模块之间的接口也是标准化的. 每个模块有一组实例化时设置的参数.

OpenSoC Fabric 中类的层次如图 1-1所示:

- 整个片上网络是顶层 (根), 分成网络接口/注入队列/拓扑三个子类;
- 网络接口支持 AXI/PIF/其他三种;
- 拓扑由路由函数和路由器两个子类构成;

OpenSoC Fabric 的模块图如图 1-2所示:

OpenSoC Fabric 实现目前主流的 wormhole 流量控制 NoC, 采用支持流水线输入队列的路由器, 虚通道可以配置, 分配器与仲裁器分离设计, 仲裁器采用 round-robin 策略. 支持的拓扑包括 Mesh 和扁平蝶形结构, 均采用 DOR 维度路由算法.

OpenSoC Fabric 还包括一个 test harness(testbench), 用来产生并注入数据包, 并从网络中移除数据包.Test harness 可以选择产生 C++ 文件还是 Verilog 文件, 如图 1-3所示.

1.2 如何编译 OpenSoC Fabric

OpenSoC Fabric 使用 Chisel 编写, 因此需要 sbt 编译运行. 在默认配置下,OpenSoC Fabric 是一个 2X2 Mesh 网络结构, 具有虚通道路由, 集中度 (concentration) 为 1, 默认注入速率为 100%. 以下命令可以运行一个简单的随机 flit 注入测试:

```
$ sbt "run␣--sw␣true␣--harnessName␣
  OpenSoC_CMeshTester_Random_VarInjRate␣--moduleName␣
  OpenSoC_CMesh_Flit"
```

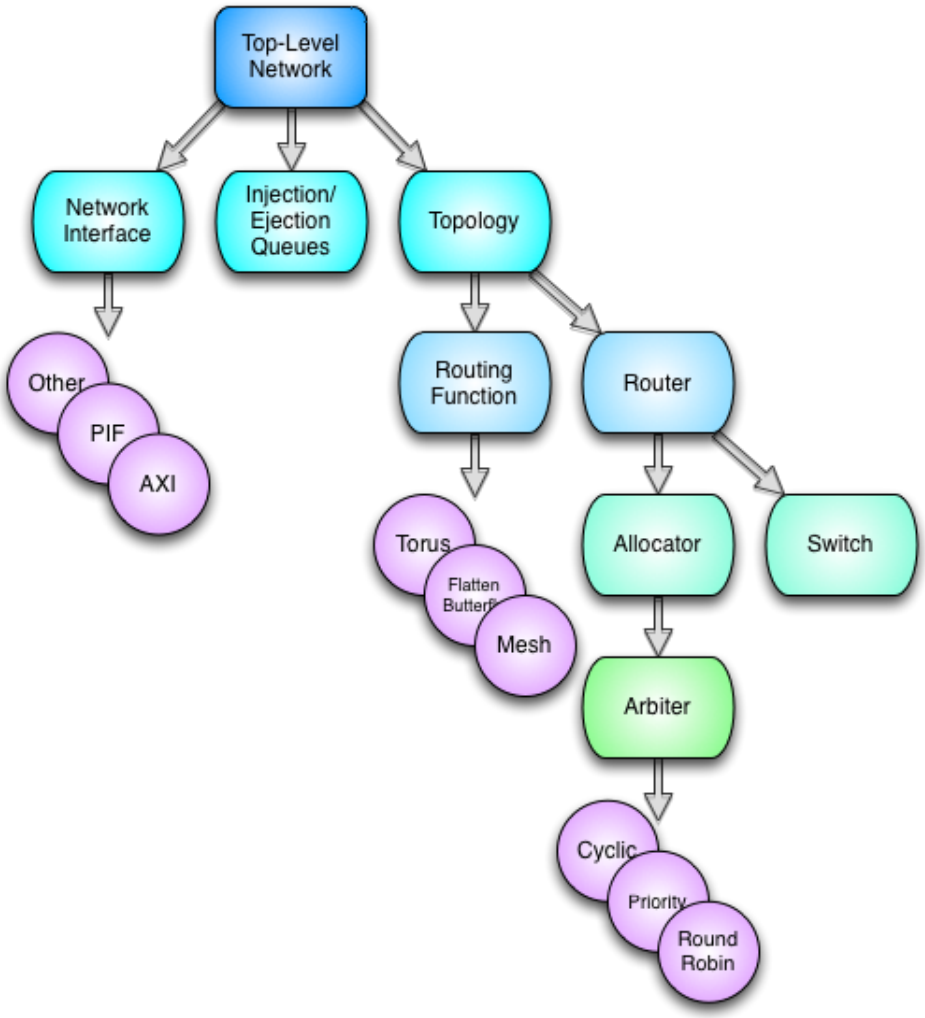

图 1-1　OpenSoC Fabri 类的层次

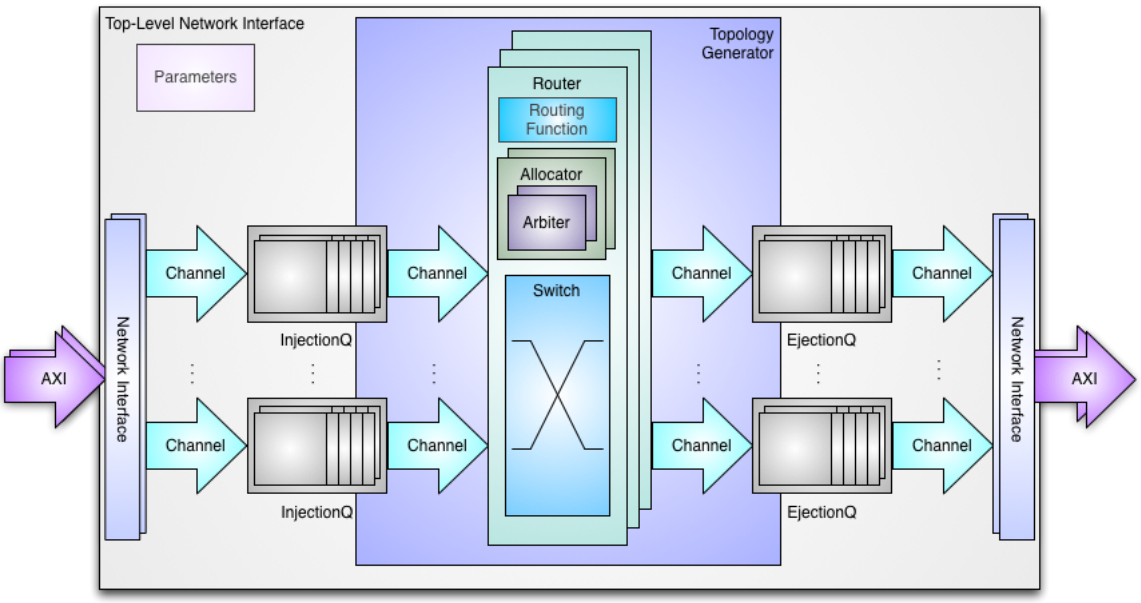

图 1-2　OpenSoC Fabri 模块结构

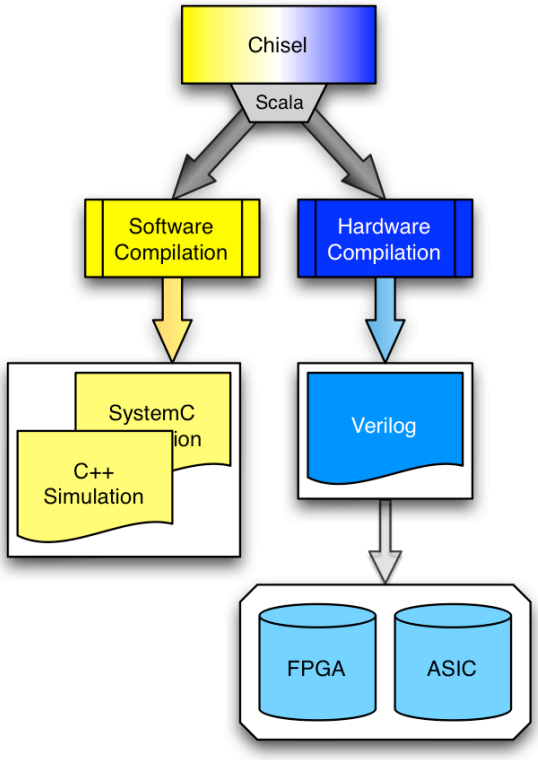

图 1-3　Chisel 编译流程

在默认情况下,OpenSoC Fabric 会生成 C++ 仿真模型. 上述命令会在编译 C++ 文件时会消耗较大内容, 如果你的 Linux 系统物理内容小于 8G, 那么建议你将虚拟内存调整到 8G 以上, 否则会报内存溢出.

在运行完成之后, 当前目录下回生成一个.vcd 文件, 你可以用 GTKwave 工具查看. 该波形文件中记录了 NoC 在仿真过程中所有顶层端口和内部寄存器的波形图.

1.3 运行参数

所有参数均定义在 hardware/src/main/scala/main.scala 文件中.

1.3.1 harnessName

运行哪种测试程序, 其取值包括:

- OpenSoC_CMeshTester_Random_VarInjRate: 目的地址随机测试
- OpenSoC_CMesh_NeighborTester_VarInjRate: 目的地址为相邻节点
- OpenSoC_CMesh_TornadoTester_VarInjRate: Tornado test pattern?
- OpenSoC_CMesh_BitReverseTester_VarInjRate: Bit reverse test pattern?
- OpenSoC_CMesh_TransposeTester_VarInjRate: Transpose test pattern?
- OpenSoC_CMeshTester_Random_Packet:?

main.scala 中的完整定义如下:

```
RingBufferTest: Ring buffer独立测试
MuxNTest: MuxN独立测试
SwitchTest: Switch独立测试
RRArbiterTest:
RRArbiterPriorityTest:
SwitchAllocTest:
RouterRegFileTest:
PacketToFlitTest:
BusProbeTest:
ChannelQTest:
PacketInjectionQTest:
```

```
VCChannelQTest :
VCIQTest :
WHCreditTest :
CMDORTester :
CFlatBflyDORTester :
SimpleRouterTester :
OpenSoC_CFlatBtflyTester_Random :
OpenSoC_CMeshTester_Random_VarInjRate :
OpenSoC_CMesh_NeighborTester_VarInjRate :
OpenSoC_CMesh_TornadoTester_VarInjRate :
OpenSoC_CMesh_BitReverseTester_VarInjRate :
OpenSoC_CMesh_TransposeTester_VarInjRate :
OpenSoC_CMesh_TraceTester :
OpenSoC_CMeshTester_Random_Packet :
OpenSoC_CMeshTester_Neighbor_Packet :
OpenSoC_CMeshTester_BitReverse_Packet :
OpenSoC_CMesh_DecoupledWrapper_Tester :
```

1.3.2 moduleName

网络模块名称, 其取值包括:

- OpenSoC_CMesh_Flit: Mesh 拓扑
- RingBuffer:
- MuxN:
- Switch:
- RRArbiter:
- RRArbiterPriority:
- SwitchAllocator:
- RouterRegFile:

- GenericChannelQ:
- PacketInjectionQ:
- VCIEChannelQ:
- InjectionChannelQ:
- PacketToFlit:
- BusProbe:
- CreditTester:
- CMeshDOR:
- CFlatBflyDOR:
- SimpleRouterTestWrapper:
- OpenSoC_CMesh:
- OpenSoC_CMesh_Flit:
- OpenSoC_CMesh_Decoupled:
- OpenSoC_CFlatBfly:

1.3.3 injRate

数据包注入速率, 取指在 0~100 之间, 默认值为 100.

1.3.4 packetCount

每个网络端口注入数据包的数量, 取指应大于 1, 默认值为 64.

1.3.5 Dim

拓扑维度, 取指应大于 1, 默认值为 2.

1.3.6 K

网络每个维度的尺度, 取指应大于 1, 默认值是每个维度尺度均为 2.

1.3.7 C

每个网络端口的集中度 (Concentration), 也就是每个路由器与几个处理器连接, 取指应大于 0, 默认值是 1(也就是每个路由器只与一个本地处理器连接).

1.3.8 numVCs

虚通道数量, 默认值为 2.

1.4 绘制图形

```
$ python statPlot.py -c channelUtilOut.csv -l latency.
  csv -r routerUtilOut.csv
```

2 模块分析

2.1 Arbiter

对应文件名为 arbiter.scala.

该模块从一定数量的请求中选择一个进行响应.Arbiter 主要用于构建 Allocator 中的部分功能.Request 同时还可以使用 lock 进行锁定, 以表示持续的请求.

2.1.1 RRArbiter

RRArbiter 是 Arbiter 的子模块, 用于实现 Round robin 仲裁策略.

2.2 Allocator

对应文件名为 allocator.scala. 该模式实现将若干资源分配给若干请求者.VC allocator 分配的资源是虚通道,Switch allocator 分配的是输出端口. 每个资源同时伴有一个输入, 标明该资源是否处于准备好可分配的状态. 最后, 请求者可以使用 lock 锁定资源. 这可以确保 VC allocator 不将 VC 分配给其他请求者, 直到前面的包已经完成传输.

2.3 Switch

对应文件名为 switch.scala.

实现了一个基于多选的交叉开关, 其输入和输出的数量均可设定.

2.4 Router

对应文件名为 router.scala.

该模块仅定义了路由器中输入和输出 flit channel. 具体功能需具体实现.

2.4.1 VCRouter

VC router 的结构如图 2-1所示.

该路由器包含一个 VC 分配器, 用于在输入端口数 *VC 数个请求者和输出端口数 *VC 数个资源之间进行分配. 分配给某个包的 VC 将锁定使用, 直到整个包离开输入

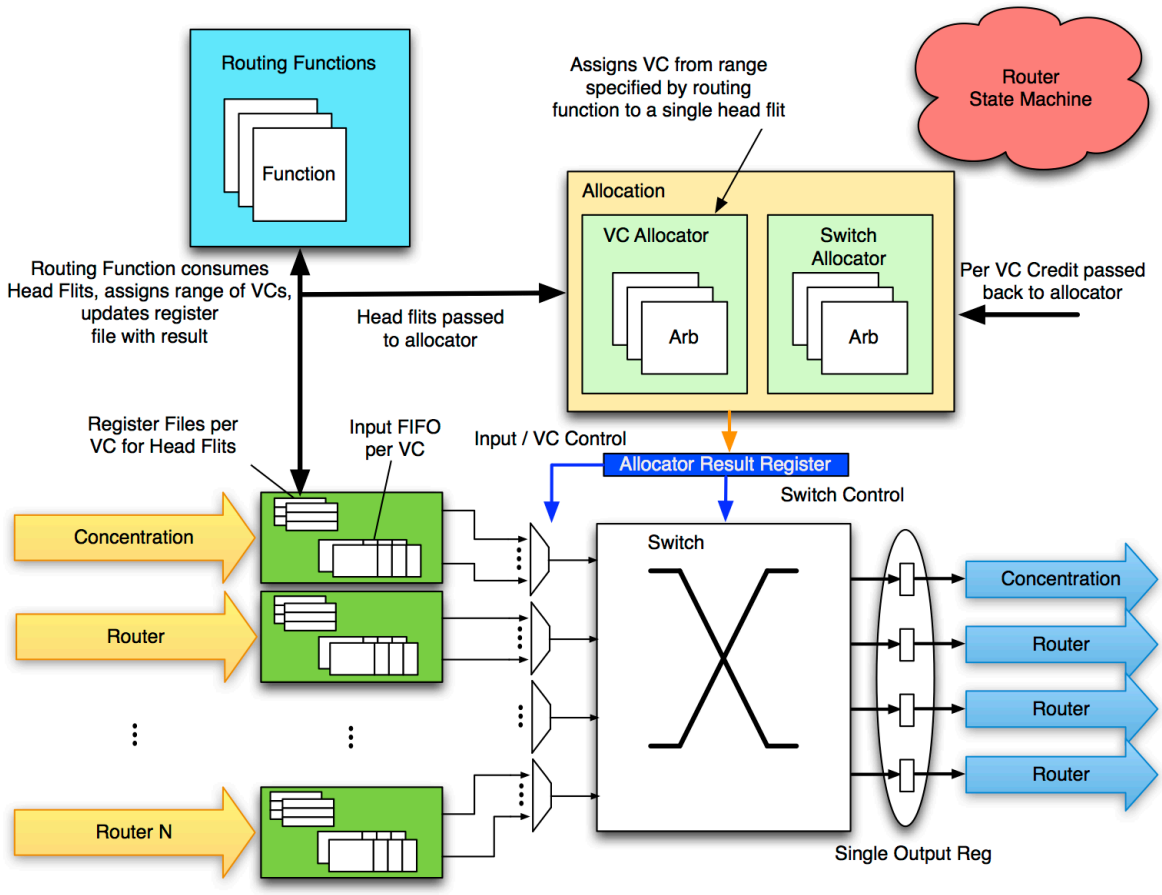

图 2-1　OpenSoC Fabri 类的层次

VC. Switch 分配器的请求者同样是输入端口数 *VC 数个, 但是资源仅有输出端口数个. 这是因为 switch 是一个共享的物理资源, 在任何一个周期, 仅有一个 flit 可以进入输入.

2.4.2 WormholeRouter

与 VCRouter 类似, 只是没有虚通道. 因此, 它的流水线中没有 VC 分配这一流水级.

2.5 RoutingFunction

对应文件名为 RoutingFunction.scala.

2.5.1 CMeshDOR

实现了 mesh 网络中的维度路由算法.

维度路由算法按维度进行路由. 以 2D mesh 为例, 包在传递过程中先沿 X 维度传递, 然后再按 Y 维度传递. 该路由具有天然避免死锁的优点.

2.5.2 CFlatBflyDOR

实现了 flattened butterfly 网络中的维度路由算法.

2.6 Topology

对应文件名为 Topology.scala.

2.6.1 CMesh

2.6.2 CFlatFly

2.7 MuxN

2.8 RingBuffer

对应文件名为 RingBuffer.scala.

参数:

bufferWidth: buffer 宽度;

totalBufferEntries: buffer 深度;

pointerCount: 应当是指针位宽?

接口:

端口名	IO	类型	备注
writeEnable	I	BOOL	1
•	•	•	•
•	•	•	•
•	•	•	•
•	•	•	•