Intel 公司推荐 FPGA/CPLD 培训教材

Quartus Prime Design Suite

Intel FPGA/CPLD 设计

（高级篇）

王江宏 蔡海宁 颜远 王诚 吴继华 编著
Intel 公司 审校

扫描二维码
下载图书配套资源

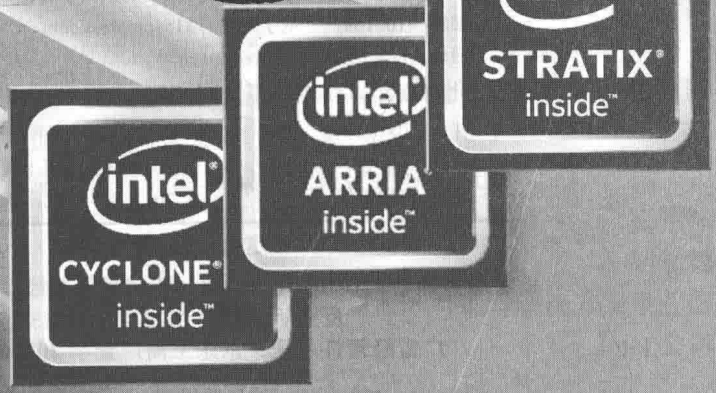
中国工信出版集团
人民邮电出版社
POSTS & TELECOM PRESS

Intel 公司推荐 FPGA/CPLD 培训教材

Quartus Prime Design Suite

Intel FPGA/CPLD 设计（高级篇）

王江宏 蔡海宁 颜远 王诚 吴继华 编著
Intel 公司 审校



人 民 邮 电 出 版 社
北 京

图书在版编目（CIP）数据

Intel FPGA/CPLD设计. 高级篇 / 王江宏等编著. --
北京 : 人民邮电出版社, 2017.9
ISBN 978-7-115-46678-5

Ⅰ. ①I… Ⅱ. ①王… Ⅲ. ①可编程序逻辑阵列－系统设计 Ⅳ. ①TP332.1

中国版本图书馆CIP数据核字(2017)第217508号

内 容 提 要

本书作者凭借多年工作经验，深入地讨论了 Intel FPGA/CPLD 的设计和优化技巧。在讨论 FPGA/CPLD 设计指导原则的基础上，介绍了 Intel FPGA 器件的高级应用；引领读者学习逻辑锁定设计工具，详细讨论了时序约束与静态时序分析的方法；针对市场应用需求，分别介绍了 SoC FPGA 和 OpenCL 系统应用技术；结合实例讨论如何进行设计优化，介绍了 Intel 的可编程器件的高级设计工具与系统级设计技巧。

本书所有实例的完整工程、源代码和使用说明文件，都以云存储的方式存放在云端，读者可以通过扫描二维码的方式进行下载。

本书可作为高等院校通信工程、电子工程、计算机、微电子与半导体等专业的教材，也可作为硬件工程师和 IC 工程师的实用工具书。

◆ 编　　著　王江宏　蔡海宁　颜　远　王　诚　吴继华
审　　校　Intel 公司
责任编辑　李永涛
责任印制　焦志炜

◆ 人民邮电出版社出版发行　　北京市丰台区成寿寺路 11 号
邮编　100164　　电子邮件　315@ptpress.com.cn
网址　http://www.ptpress.com.cn
北京鑫正大印刷有限公司印刷

◆ 开本：787×1092　1/16
印张：20.75　　2017 年 9 月第 1 版
字数：518 千字　　2017 年 9 月北京第 1 次印刷

定价：59.00 元

读者服务热线：(010)81055410　印装质量热线：(010)81055316
反盗版热线：(010)81055315
广告经营许可证：京东工商广登字 20170147 号

序

Altera 公司成立于 1983 年，后于 2016 年成为 Intel 公司可编程解决方案事业部（Programmable Solutions Group），总部位于硅谷。自从 1984 年发明世界上第一款可编程逻辑器件以来，一直为客户提供业界领先的定制逻辑解决方案。今天，分布在全球 20 多个地区的 3000 多名员工将为 12000 多家用户提供更巧妙的定制逻辑解决方案及广泛的技术支持，包括了 FPGA、SoC、CPLD 及电源管理产品等。

Intel 的全系列定制逻辑解决方案解决了很多系统级难题，包括性能、功耗、总体拥有成本、电路板面积、产品及时面市和设计团队效能等。很多不同行业的业界领先公司都采用了 Intel FPGA 产品，包括数据中心、通信、汽车、广播、工业、医疗、军事、测试和测量、消费类等行业。

Intel 公司可编程解决方案事业部的产品包括：

- Stratix 系列 FPGA 和 SoC（支持实现性能最好的系统，用于数据处理和算法加速）；
- Arria 系列 FPGA 和 SoC（以很低的功耗预算实现了高性能系统）；
- Cyclone 系列 FPGA 和 SoC（以最低功耗实现了需要较高性能的大批量应用系统）；
- MAX 系列 FPGA 和 CPLD（在大批量系统中实现了非易失、单芯片集成）；
- Enpirion 电源产品（为 Intel FPGA、SoC 和 CPLD 提供支持，在集成产品中同时实现了高效、小外形封装、低噪声性能）；
- 软件开发工具、IP、参考设计和开发套件（方便了 Intel FPGA、SoC 和 CPLD 的开发，保证了极高的设计团队效能）。

随着最终市场需求的发展，用户产品越来越复杂，我们的定制逻辑容量也随之快速增长。结果，设计工程师没有足够的设计方法和知识来满足需求的增长。设计人员在采用最新定制逻辑解决方案及 Intel FPGA 前沿产品和技术时，可以充分利用我们在中国本地开发的参考材料和指南。

我非常荣幸地向您推荐《Intel FPGA/CPLD 设计（基础篇）》和《Intel FPGA/CPLD 设计（高级篇）》。与前面版本图书相比，本版对内容进行了更新以反映 Intel 最新 FPGA 器件和设计工具。这两本书不仅介绍了传统 PLD 技术和设计技巧，而且还大量阐述了目前业界流行的硬浮点 DSP、片外高速存储器、HMC（Hybrid Memory Controller）、JESD204B 和片上系统 FPGA，以及 Intel 推出的适用于异构平台并通过 FPGA 做加速处理的 OpenCL 软件开发套件。

这两本书以独特的视角解释了设计方法，帮助您掌握高级 PLD 设计技巧，还介绍了 Intel FPGA 器件和 Quartus II 设计软件。这些书有丰富的设计实例，通过实际练习，能帮助您深入理解概念，养成良好的设计习惯。

希望您能够从这些优秀的书中受益，预祝您的可编程逻辑设计获得成功!

庄秉翰

副总裁

亚太区可编程解决方案事业部

Intel 公司

关于本书

内容和特点

FPGA/CPLD、GPU 和 CPU 被称为未来数字电路系统的 3 块基石，也是目前硬件设计研究的热点。与传统电路设计方法相比，FPGA/CPLD 具有功能强大，开发过程投资小、周期短，可反复编程修改，保密性能好，开发工具智能化等特点，特别是随着电子工艺的不断改进，低成本 FPGA/CPLD 器件推陈出新，这一切促使 FPGA/CPLD 成为当今硬件设计的首选方式之一。可以说 FPGA/CPLD 设计技术是当今高级硬件工程师与 IC 工程师的必备技能。

我国可编程逻辑器件设计技术落后于国外，目前立足工程实践，系统地介绍最新 FPGA/CPLD 设计工具的中文书籍较为贫乏。在这种情况下，为了满足广大工科在校生了解业界流行的高效 FPGA/CPLD 设计技术的需要，提高硬件工程师与 IC 工程师的工程实践技巧，我们编写了《Altera FPGA/CPLD 设计（基础篇）》和《Altera FPGA/CPLD 设计（高级篇）》。这两本书出版以来，广受读者好评，但随着技术的不断发展，器件型号和软件版本的不断更新，原有图书的内容和知识体系已经不适应目前的读者需求，为此我们根据 Intel（注：原 Altera 已于 2016 年被 Intel 收购）推出的一系列新型 FPGA，以及新版 Quartus II 软件的特性，对上述两本书进行了改版升级，升级后的书名为《Intel FPGA/CPLD 设计（基础篇）》和《Intel FPGA/CPLD 设计（高级篇）》。

升级后的图书涵盖了 Intel 主流 FPGA/CPLD 的硬件结构与特性，详尽地讨论了 Quartus II 与第三方 EDA 工具的设计方法，系统地阐述了 Intel 可编程逻辑设计优化技术。

本书共 7 章，各章内容简要介绍如下。

- 第 1 章：探讨了可编程逻辑设计的基本原则和常用思想与技巧，并详细地讨论了 Intel 推荐的 FPGA Coding Style。
- 第 2 章：分别介绍了 Intel FPGA 器件的时钟管理、硬浮点数字信号处理、片外高速存储器、HMC、JESD204B、高速串行收发器等高级硬件特性和 IP 的应用方法。
- 第 3 章：重点介绍 Intel SoC FPGA 嵌入式设计基础。
- 第 4 章：在介绍时序分析的基本概念与常用约束方法的基础上，讨论了高级时序分析的技巧。
- 第 5 章：介绍资源利用率优化、I/O 时序优化、最高频率优化等设计优化的实用技术，并讨论了如何使用 DSE 进行优化的方法。
- 第 6 章：介绍 Tcl 脚本、DSP Builder、Intel FPGA OpenCL 软件开发套件等高级工具的使用方法。
- 第 7 章：重点讨论了信号完整性、电源设计、功耗分析与热设计、SERDES 与高速系统设计等系统级设计技巧。

本书的主要特点介绍如下。

- 全面系统：涵盖了 Intel FPGA 软、硬件设计技术，基础与高级设计工具，全面系统地论述了 Intel 可编程设计技术。
- 实用价值高：本书的作者都有丰富的 FPGA/CPLD、嵌入式 SoC 和 OpenCL 设

计经验，本书立足于工程实践的需要，对工程设计有显著的指导意义。

- 内容新颖：本书的作者长期工作在可编程逻辑设计的最前沿，与 FPGA 器件制造公司和 EDA 软件设计公司联系紧密，所以有幸能够在第一时间内使用最新版本的 FPGA/CPLD 设计工具。书中涉及的所有工具均根据较新资料撰写，使图书介绍的内容新颖。
- 剖析深刻：书中对 FPGA/CPLD 设计的基本原理、方法有较为详尽的论述，对各种设计工具的介绍并不局限于操作方法，而是结合作者多年的工作经验与心得，从较深的层面对各个工具的特点进行剖析。

读者对象

本书可作为高等院校通信工程、电子工程、计算机、微电子与半导体学等理工专业的教材，也可作为硬件工程师和 IC 工程师的实用工具书。

配套资源

配套资源中提供了书中所有示例的完整工程文件、设计源文件和说明文件（读者可扫描封面上的二维码进行下载）。

每个工程示例都包括了该工程的项目文件、源文件、报告文件和生成结果等文件，读者可以用 Quartus II 或相应的软件直接打开。设计源文件根据设计输入类型分为源代码或原理图等，请读者将设计源文件复制到计算机硬盘上，并按照书中的操作步骤自行操作练习。示例说明文件包含了示例的详细信息和操作指南。

本书约定

为了方便读者阅读，书中设计了 4 个小图标，它们代表的含义如下。


行家指点：用于介绍使用经验和心得，或罗列重要的概念。


注意事项：用于提醒读者应该注意的问题。


多学一招：用于介绍实现同一功能的不同方法。


操作实例：用于引出一个操作题目和相应的一组操作步骤。

全书的各章节分别由王江宏、蔡海宁、颜远、王诚和吴继华等作者执笔，全书由 Intel 公司可编程解决方案事业部（Programmable Solutions Group）资深现场应用工程师王江宏统一修改整理。

资深高速 I/O 技术专家蔡海宁先生、资深高速 I/O 技术应用工程师董凡辉先生、资深内存接口技术应用工程师何虎刚先生、资深数字信号处理技术应用工程师王欣先生，对全书新版章节进行了审校。Intel 公司亚太区可编程解决方案事业部现场应用工程总监邓海涛先生、亚太区应用工程总监罗小锋先生、中国区大客户销售总监吕家龙先生、现场应用工程经理赵敏先生对本书提出了许多建设性意见，并给予作者多方面的帮助。在这里要特别感谢 Intel

公司亚太区可编程解决方案事业部副总裁庄秉翰先生在百忙之中亲自为本书撰写序言。感谢所有关心并支持本书的同仁佳友！

感谢您选择了本书，如果您对书中内容有任何困惑和建议，请与我们联系。

电子邮件：adeli.wang@intel.com（作者），liyongtao@ptpress.com.cn（责任编辑）。

如果您需要得到 Intel 更全面的服务与技术支持，请访问 http://www.altera.com.cn。

编者

2017 年 5 月

目　录

第1章　可编程逻辑设计指导原则

本章旨在探讨可编程逻辑设计的一些基本规律。FPGA/CPLD 的设计规律与方法是一个非常大的课题，在此不可能面面俱到，希望通过本章提纲携领的粗浅介绍，引起读者的注意。如果大家能在日后的工作实践中不断积累，有意识地用 FPGA/CPLD 的基本设计原则、设计思想作为指导，将取得事半功倍的效果。

本章主要内容如下。

- 可编程逻辑基本设计原则。
- 可编程逻辑常用设计思想与技巧。
- Altera 推荐的 Coding Style。

1.1　可编程逻辑基本设计原则

可编程逻辑设计有许多内在规律可循，总结并掌握这些规律对于较深刻地理解可编程逻辑设计技术非常重要。本章从 FPGA/CPLD 的基本概念出发，总结出 4 个基本设计原则，这些指导原则范畴非常广，希望读者不仅仅是学习它们，更重要的是理解它们，并在今后的工作实践中充实、完善它们。

(1) 面积和速度的平衡与互换原则。提出了 FPGA/CPLD 设计的两个基本目标，并探讨了这两个目标对立统一的矛盾关系。

(2) 硬件原则。重点在于提醒读者转化软件设计的思路，理解 HDL 语言设计的本质。

(3) 系统原则。希望读者能够通过从全局、整体上把握设计，从而提高设计质量，优化设计效果。

(4) 同步设计原则。设计时序稳定的基本要求，也是高速 PLD 设计的通用法则。

1.1.1　面积和速度的平衡与互换原则

这里的“面积”是指一个设计所消耗 FPGA/CPLD 的逻辑资源数量：对于 FPGA，可以用所消耗的触发器（FF）和查找表（LUT）来衡量；对于 CPLD，常用宏单元（MC）衡量。用设计所占用的等价逻辑门数来衡量设计所消耗 FPGA/CPLD 的逻辑资源数量也是一种常见的衡量方式。“速度”指设计在芯片上稳定运行时所能够达到的最高频率，这个频率由设计的时序状况决定，与设计满足的时钟周期、PAD to PAD Time、Clock Setup Time、Clock Hold Time 和 Clock-to-Output Delay 等众多时序特征量密切相关。面积（Area）和速度（Speed）这两个指标贯穿着 FPGA/CPLD 设计的始终，是设计质量评价的终极标准。这里

我们就讨论一下设计中关于面积和速度的基本原则：面积和速度的平衡与互换。

面积和速度是一对对立统一的矛盾体。要求一个设计同时具备设计面积最小，运行频率最高，这是不现实的。科学的设计目标应该是在满足设计时序要求（包含对设计最高频率的要求）的前提下，占用最小的芯片面积，或者在所规定的面积下，使设计的时序余量更大，频率更高。这两种目标充分体现了面积和速度的平衡思想。关于面积和速度的要求，我们不应该简单地理解为工程师水平的提高和设计完美性的追求，而应该认识到它们是与产品的质量和成本直接相关的。如果设计的时序余量比较大，运行的频率比较高，则意味着设计的健壮性更强，整个系统的质量更有保证；另一方面，设计所消耗的面积更小，则意味着在单位芯片上实现的功能模块更多，需要的芯片数量更少，整个系统的成本也随之大幅度削减。

作为矛盾的两个组成部分，面积和速度的地位是不一样的。相比之下，满足时序、工作频率的要求更重要一些，当两者冲突时，采用速度优先的准则。

面积和速度的互换是 FPGA/CPLD 设计的一个重要思想。从理论上讲，一个设计如果时序余量较大，所能跑的频率远远高于设计要求，那么就能通过功能模块复用减少整个设计消耗的芯片面积，这就是用速度的优势换面积的节约；反之，如果一个设计的时序要求很高，普通方法达不到设计频率，那么一般可以通过将数据流串并转换，并行复制多个操作模块，对整个设计采取“乒乓操作”和“串并转换”的思想进行处理，在芯片输出模块处再对数据进行“并串转换”。从宏观上看，整个芯片满足了处理速度的要求，这相当于用面积复制换取速度的提高。面积和速度互换的具体操作技巧很多，如“模块复用”“乒乓操作”“串并转换”等，需要大家在日后工作中不断积累。下面举例说明如何使用“速度换面积”和“面积换速度”。

【例1-1】 如何使用“速度的优势换取面积的节约”？

在 WCDMA（宽带码分多址）系统中，使用到了快速哈达码（FHT）运算，如图 1-1 所示。

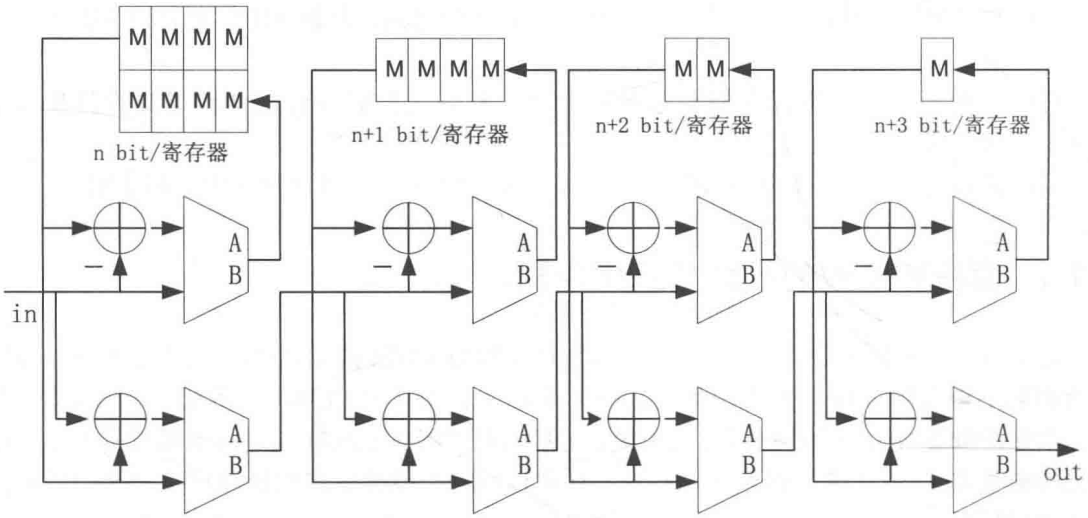


图1-1　FHT 原理图

FHT 的单步算法如下。

$$Out[2i] = In[2i] + In[2i+8]; i = 0-7;$$
$$Out[2i+1] = In[2i+1] - In[2i+1+8]; i = 0-7$$

考虑流水线式数据处理的要求，最自然的设计方法就是设计不同端口宽度的 4 个单步 FHT，并将这 4 个单步模块串联起来，从而完成数据流的流水线处理。该 FHT 实现方式的代码如下。

```
    //该模块是 FHT 的顶层，调用 4 个不同端口宽度的单步 FHT 模块，完成整个 FHT 算法
    module
fhtpart(Clk,Reset,FhtStarOne,FhtStarTwo,FhtStarThree,FhtStarFour,
            I0,I1,I2,I3,I4,I5,I6,I7,I8,
            I9,I10,I11,I12,I13,I14,I15,
            Out0,Out1,Out2,Out3,Out4,Out5,Out6,Out7,Out8,
            Out9,Out10,Out11,Out12,Out13,Out14,Out15);
    input Clk;    //设计的主时钟
    input Reset;   //异步复位
    input FhtStarOne,FhtStarTwo,FhtStarThree,FhtStarFour; //4 个单步算法的时序控制信号
    input [11:0] I0,I1,I2,I3,I4,I5,I6,I7,I8;
    input [11:0] I9,I10,I11,I12,I13,I14,I15;                    //FHT 的 16 个输入
    output [15:0] Out0,Out1,Out2,Out3,Out4,Out5,Out6,Out7;
    output [15:0] Out8,Out9,Out10,Out11,Out12,Out13,Out14,Out15;  //FHT 的 16 个输出

    //第 1 次 FHT 单步运算的输出
    wire [12:0] m0,m1,m2,m3,m4,m5,m6,m7,m8,m9;
    wire [12:0] m10,m11,m12,m13,m14,m15;

    //第 2 次 FHT 单步运算的输出
    wire [13:0] mm0,mm1,mm2,mm3,mm4,mm5,mm6,mm7,mm8,mm9;
    wire [13:0] mm10,mm11,mm12,mm13,mm14,mm15;

    //第 3 次 FHT 单步运算的输出
    wire [14:0] mmm0,mmm1,mmm2,mmm3,mmm4,mmm5,mmm6,mmm7,mmm8,mmm9;
    wire [14:0] mmm10,mmm11,mmm12,mmm13,mmm14,mmm15;

    //第 4 次 FHT 单步运算的输出
    wire [15:0] Out0,Out1,Out2,Out3,Out4,Out5,Out6,Out7,Out8,Out9;
    wire [15:0] Out10,Out11,Out12,Out13,Out14,Out15;
    //第 1 次 FHT 单步运算
```

```
fht_unit1 fht_unit1(Clk,Reset,FhtStarOne,
          I0,I1,I2,I3,I4,I5,I6,I7,I8,
          I9,I10,I11,I12,I13,I14,I15,
          m0,m1,m2,m3,m4,m5,m6,m7,m8,
          m9,m10,m11,m12,m13,m14,m15
          );

//第 2 次 FHT 单步运算
fht_unit2 fht_unit2(Clk,Reset,FhtStarTwo,
          m0,m1,m2,m3,m4,m5,m6,m7,m8,
          m9,m10,m11,m12,m13,m14,m15,
          mm0,mm1,mm2,mm3,mm4,mm5,mm6,mm7,mm8,
          mm9,mm10,mm11,mm12,mm13,mm14,mm15
          );
//第 3 次 FHT 单步运算
fht_unit3 fht_unit3(Clk,Reset,FhtStarThree,
          mm0,mm1,mm2,mm3,mm4,mm5,mm6,mm7,mm8,
          mm9,mm10,mm11,mm12,mm13,mm14,mm15,
          mmm0,mmm1,mmm2,mmm3,mmm4,mmm5,mmm6,mmm7,mmm8,
          mmm9,mmm10,mmm11,mmm12,mmm13,mmm14,mmm15
          );
//第 4 次 FHT 单步运算
fht_unit4 fht_unit4(Clk,Reset,FhtStarFour,
          mmm0,mmm1,mmm2,mmm3,mmm4,mmm5,mmm6,mmm7,mmm8,
          mmm9,mmm10,mmm11,mmm12,mmm13,mmm14,mmm15,
          Out0,Out1,Out2,Out3,Out4,Out5,Out6,Out7,Out8,
          Out9,Out10,Out11,Out12,Out13,Out14,Out15
          );
endmodule
```

单步 FHT 运算如下（仅仅举例第 4 步的模块）。

```
module fht_unit4(Clk,Reset,FhtStar,
          In0,In1,In2,In3,In4,In5,In6,In7,In8,
          In9,In10,In11,In12,In13,In14,In15,
          Out0,Out1,Out2,Out3,Out4,Out5,Out6,Out7,Out8,
          Out9,Out10,Out11,Out12,Out13,Out14,Out15
          );

input Clk;          //设计的主时钟
```

```
input Reset;       //异步复位
input FhtStar;     //单步 FHT 运算控制信号
input [14:0] In0,In1,In2,In3,In4,In5,In6,In7,In8,In9;
input [14:0] In10,In11,In12,In13,In14,In15;          //单步 FHT 运算输入
output [15:0] Out0,Out1,Out2,Out3,Out4,Out5,Out6,Out7,Out8,Out9;
output [15:0] Out10,Out11,Out12,Out13,Out14,Out15;  //单步 FHT 运算输出

//Single FHT calculation
reg [15:0] Out0,Out1,Out2,Out3,Out4,Out5;
reg [15:0] Out6,Out7,Out8,Out9,Out10,Out11;
reg [15:0] Out12,Out13,Out14,Out15;
//补码运算
wire [14:0] In8Co =~In8+1;
wire [14:0] In9Co =~In9+1;
wire [14:0] In10Co=~In10+1;
wire [14:0] In11Co=~In11+1;
wire [14:0] In12Co=~In12+1;
wire [14:0] In13Co=~In13+1;
wire [14:0] In14Co=~In14+1;
wire [14:0] In15Co=~In15+1;

always @(posedge Clk or negedge Reset)
begin
  if(!Reset)
  begin
    Out0<=0;Out1<=0;Out2<=0;Out3<=0;
    Out4<=0;Out5<=0;Out6<=0;Out7<=0;
    Out8<=0;Out9<=0;Out10<=0;Out11<=0;
    Out12<=0;Out13<=0;Out14<=0;Out15<=0;
  end
  else
  begin
    if(FhtStar)
    begin
    Out0<={In0[14],In0 }+{In8[14],In8 };
    Out1<={In0[14],In0 }+{In8Co[14],In8Co };
    Out2<={In1[14],In1 }+{In9[14],In9 };
    Out3<={In1[14],In1 }+{In9Co[14],In9Co };
    Out4<={In2[14],In2 }+{In10[14],In10 };
```