# 第二章 FPGA逻辑综合及技术映射

**2.1 FPGA及资源映射**

**2.1.1 FPGA**

FPGA是一种随着时代发展而产生的特殊芯片，它的出现是集成电路快速发展的一个缩影。上世纪中后期，集成电路的设计成本越来越高，各大公司都希望找到一种不仅功能强大而且设计成本低廉的芯片。在这种背景下，现场可编程门阵列FPGA应运而生。相对于其他芯片，使用FPGA进行研发的容错成本更低。在研发过程中，一旦发现芯片功能不符合所制定的要求，则马上可以对设计进行调整。在目前的研发过程中，研发团队首先会使用少量的FPGA芯片去进行设计实现，并根据实际情况进行修改直到设计达到预期的效果，最后再将达到要求的设计用ASIC实现并最终投入生产。另外，在一些功能灵活、规模较小的项目中，研发者直接使用FPGA实现设计，不仅可以满足功能需求，而且研发成本很低。

FPGA是可编程逻辑器件（PLD）的一种，是可以用来实现任意逻辑电路的集成电路。FPGA的特征从名字就可以看出是在现场可编程的门阵列。但实际上FPGA并非是单纯由“门”形成的结构。

**2.1.2 FPGA的资源组成**

FPGA大致由三部分构成：第一部分是实现逻辑电路的逻辑要素即逻辑块（Logic Block，LB）也就是我们常说的基本逻辑单元，可编程逻辑中逻辑块的实现方式有多重，比如最早的乘积项和现在的查找表、数据选择器等，但是无论哪种方式都是由可以实现任意逻辑电路的可编程部分触发器即Flip-Flop,FF等数据存储电路和数据选择器组成。逻辑要素也是在FPGA设计开发流程中比较紧俏的资源种类之一，FPGA供应商对各自的逻辑块结构有不同的称呼，赛灵思公司称其为CLB(Configurable Logic Block),但基本原理都是类似的，采用的是查找表的方式。第二部分是和外部进行信号输入、输出对的要素即（IO块），第三部分是连接前两种元素的布线要素。

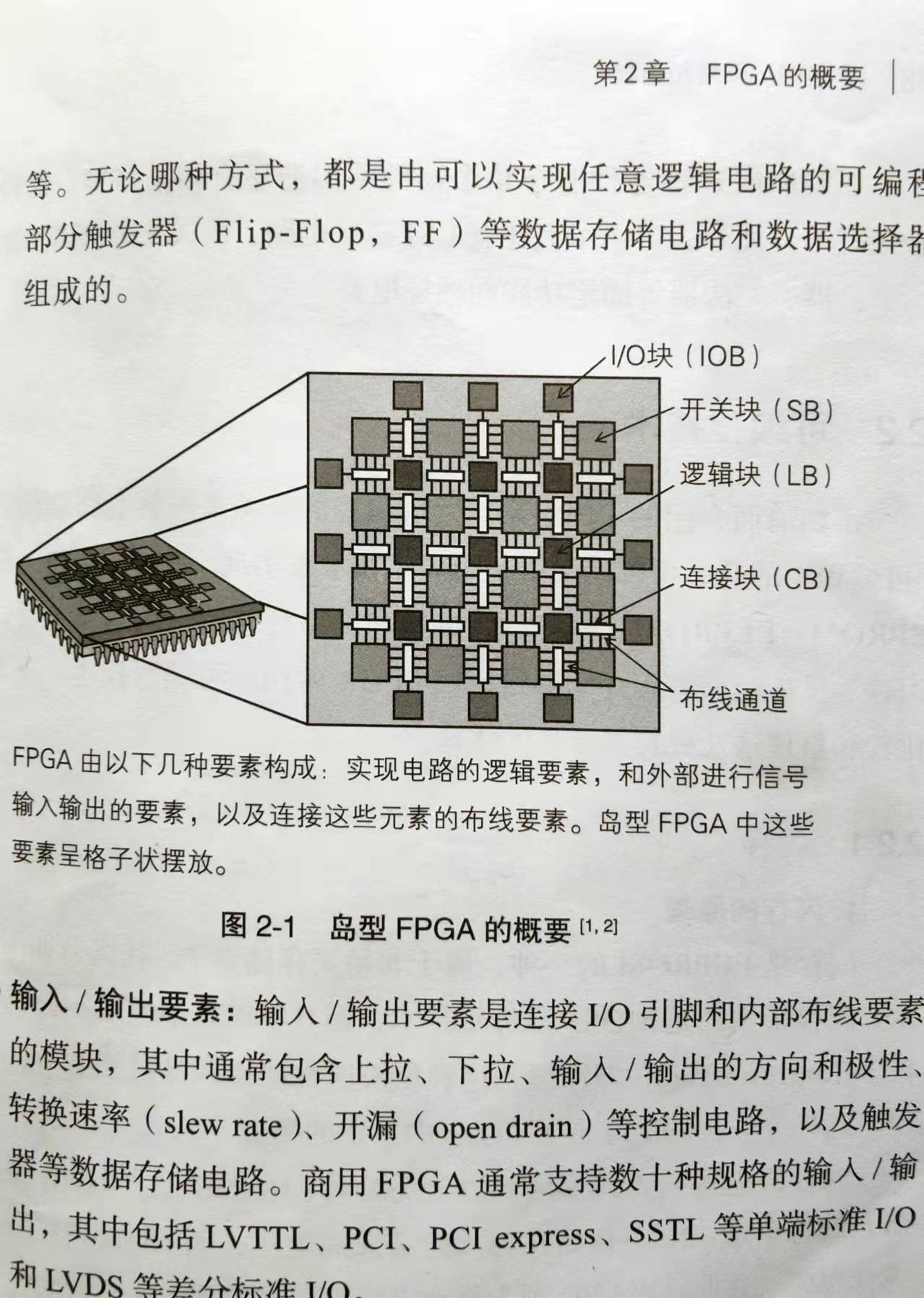


图2-1 FPGA基本组成

通常的FPGA组成主要包括：可编程逻辑单元、布线资源、可编程输入输出及内嵌RAM（Random Access Memory，随机存取存储器）等。可编程逻辑单元负责实现逻辑功能，布线资源能够连接各个逻辑单元，而内嵌RAM则可以配置为各种存储形式，输入输出负责FPGA 与外部模块的通信。在以上FPGA的结构中，最重要的是可编程逻辑单元，它主要包括触发器和查找表两部分。从实际效果看，查找表在功能上可以等效为一种特殊的静态存储器SRAM(Static Random Access Memory，静态随机存取存储器)。

可编程逻辑单元中，查找表表示的是1个字只有一位的内存表，字数取决于地址的位数。FPGA上查找表的存储单元。FPGA上查找表的存储单元大多使用SRAM实现。查找表的结构**如图所示，**

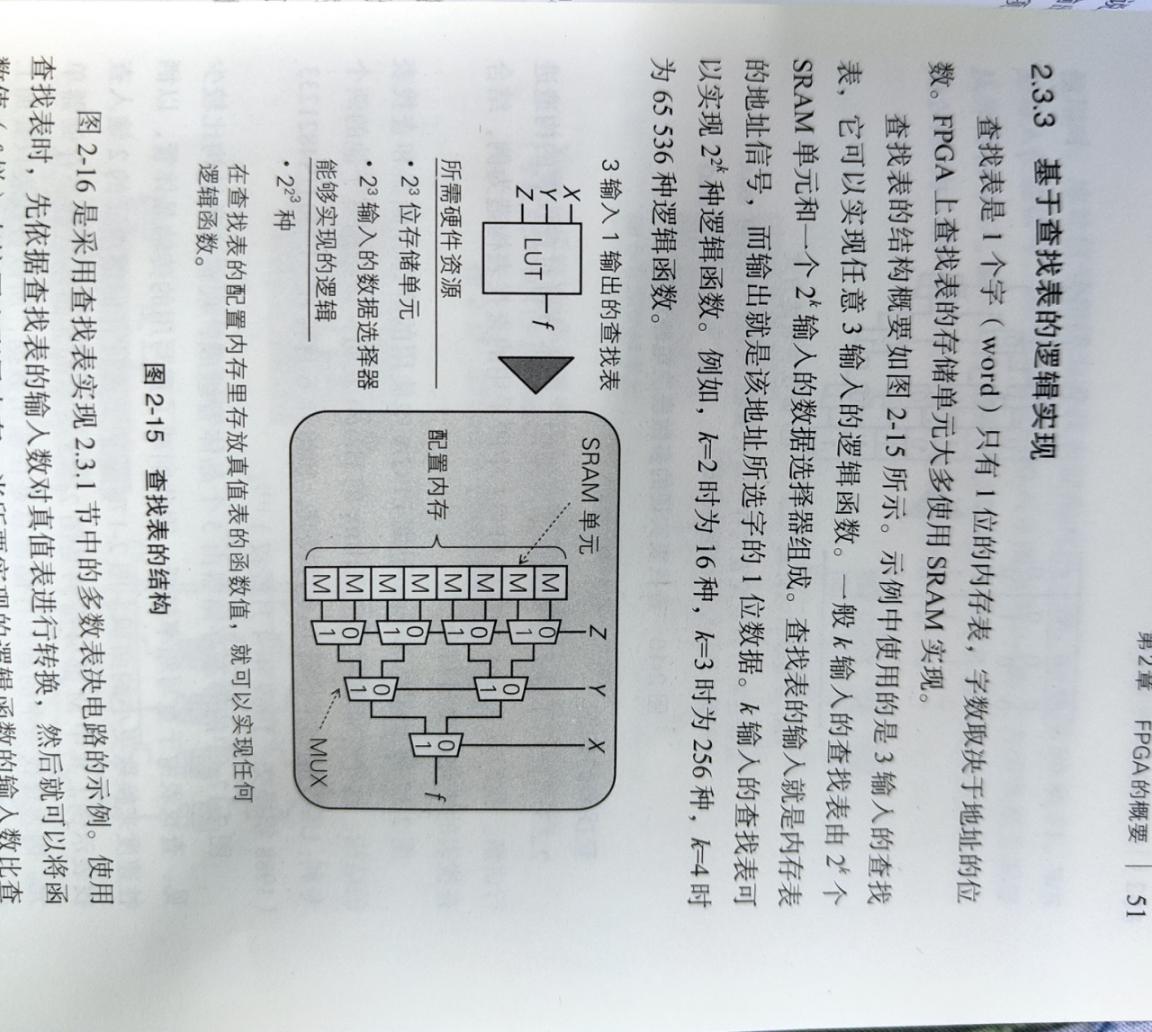
****

图2-2 FPGA基本组成

如图中所示的3输入的查找表，它可以实现任意3输入的逻辑函数。一般k输入的查找表由2k个SRAM单元和一个2k输入的数据选择器组成。查找表的输入就是内存表的的地址信号，而输出就是该地址所选字的1位数据。K输入的查找表可以实现22k种逻辑。

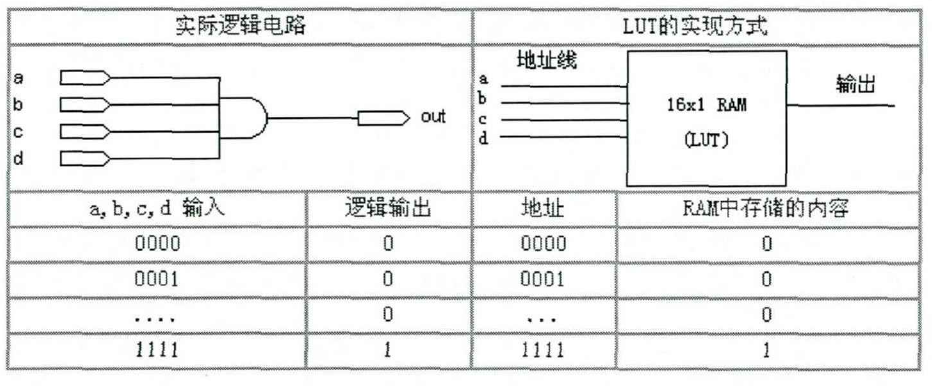


图2-3 LUT基本工作原理

LUT基本工作原理**如图所示**，左侧是期望实现的逻辑电路及其真值表，右侧是相应的查找表的内容。由图可以知，查找表中的内容只是将逻辑输入等效为存储区地址，而逻辑输出等效为存储区地址中的具体内容。最终查找表将原先复杂的逻辑运算转化为简单的输出存储区指定地址中的值，这也是 FPGA 的验证效率比软件仿真等方法高效的重要原因。此外，当需要实现时序逻辑电路时，FPGA通过在查找表后边配置触发器来完成。

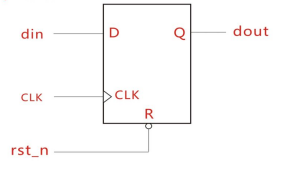


图2-4 D触发器

可编程逻辑单元中的另一种资源FF（Flip Flop）触发器。其工作原理同数字电路中所使用的触发器原理一致，也叫双稳态门，又称双稳态触发器。是一种可以在两种状态下运行的数字逻辑电路。触发器一直保持它们的状态，直到它们收到输入脉冲，又称为触发。当收到输入脉冲时，触发器输出就会根据规则改变状态，然后保持这种状态直到收到另一个触发。触发器（flip-flops）电路相互关联，从而为使用内存芯片和微处理器的数字集成电路（IC）形成逻辑门。它们可用来存储一比特的数据。该数据可表示音序器的状态、计数器的价值、在计算机内存的ASCII字符或任何其他的信息。有几种不同类型的触发器（flip-flops），如T（切换）、S-R（设置/重置）J-K（也可能称为Jack Kilby）和D（延迟）。典型的触发器包括零个、一个或两个输入信号，以及时钟信号和输出信号。一些触发器还包括一个重置当前输出的明确输入信号。触发器资源在大型RTL设计进行FPGA原型验证时也常常面临资源紧俏的情况。

EDA(Electronic Design Automation,电子设计自动化)是充分发挥FPGA及设计性能的关键技术，理论上，一款FPGA所能达到的性能上限是由制程等物理因素决定，而在实际应用中用户电路的性能很大程度上取决于硬件的架构和EDA工具。这就像汽车一样，无论引擎（制程）多么强劲，都需要配合适当的车体和驾驶技术才能发挥出极限速度。尤其是与电路实现直接相关的EDA工具，其对性能的影响不可估量。

**2.2 FPGA设计流程**

在本文中提到的设计流程的起点均为RTL级Verilog HDL代码，故我们先在此处对verilog HDL语法及综合原则进行简略的介绍。

**2.2.1 Verilog VHDL语言介绍**

Verilog HDL是一种硬件描述语言，以文本形式来描述数字系统硬件的结构和行为的语言，用它可以表示逻辑电路图、逻辑表达式，还可以表示数字逻辑系统所完成的逻辑功能。 Verilog HDL和VHDL是世界上最流行的两种硬件描述语言，都是在20世纪80年代中期开发出来的。前者由Gateway Design Automation公司（该公司于1989年被Cadence公司收购）开发。两种HDL均为IEEE标准。 根据IEEE所描述的Verilog HDL是一种用来描述和创建电路系统所有阶段的表示形式，由于它是机器可读的、人类可读的，所以它支持硬件设计的开发、验证、合成和测试、硬件设计数据的通信和维护、修改。

Verilog包含一组丰富的内置原语，包括逻辑门、用户可定义原语、开关和有线逻辑。它还具有设备端到端的延迟和定时检查。抽象级别的混合基本上由两种数据类型的语义提供：网络和变量。连续赋值提供了基本的结构构造，在连续赋值中，变量和网络的表达式可以连续地将值驱动到网络上。程序赋值提供了基本的行为结构，其中涉及变量和净值的计算结果可以存储到变量中。设计包括一组模块，每个模块都有一个I/O接口，以及对其功能的描述，这些功能可以是结构化的、行为的，也可以是混合的。这些模块形成一个层次结构，并与网络互连。

Verilog语言可以通过编程语言接口（PLI）和Verilog过程接口（VPI）进行扩展。PLI/VPI是一组例程，允许外部函数访问设计的Verilog HDL描述中包含的信息，并促进与仿真的动态交互。PLI/VPI的应用包括使用其他仿真和CAD系统连接到Verilog HDL模拟器、定制调试任务、延迟计算器和注释器。

在Verilog HDL中有两种基本语句：连续赋值语句，在此表达式中寄存器和线网都可以连续驱动线网，实现基本的结构化建模；过程赋值语句，在此表达式中寄存器和线网都将运算结果存入寄存器，实现基本的行为建模。一个设计通常包含许多模块，每一模块都有一个输入或输出接口和该模块相应的功能描述。而该功能描述既可以采用结构化的描述方式也可以采用行为化的描述方式，或者二者的结合，这些模块都被加工到一定的抽象级别，然后通过线网连接。

而在典型FPGA设计流程中，或大型FPGA原型验证流程中，首选通过使用Verilog HDL等硬件描述语言编写寄存器传输级（RTL）代码来描述电路。RTL描述通过逻辑综合得到电路网表（Netlist），通过对网表进行技术映射，FPGA布局布线，最后生成FPGA配置数据（bitstream）。其中对于大型原型验证系统最为关注的前端工程就是逻辑综合和技术映射，这个过程是验证Verilog HDL语法正确性和计算该设计所需FPGA片上资源总数的重要过程。其中逻辑综合和技术映射的过程就是将Verilog HDL语言描述的寄存器级代码转换映射至具有工艺概念的逻辑器件的过程，也是我们结合机器学习人工智能进行FPGA资源估算设计过程中重点关注的过程。

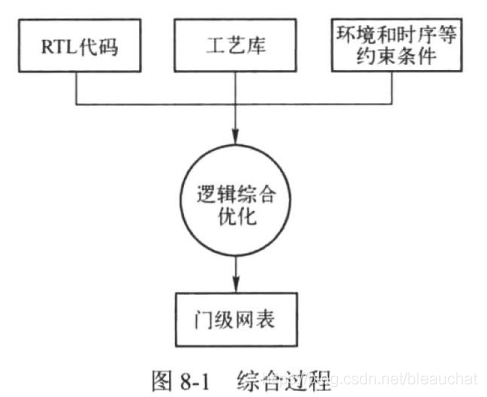
**2.2.2 Verilog VHDL综合介绍**

对于FPGA数字电路设计而言综合一句话概括就是从就是从采用 HDL语言描述的寄存器传输级(RTL)电路模型构造出门级网表的过程，Verilog HDL只是一种硬件的描述语言，并且允许在不同的抽象层次上对电路进行建模。VerilogHDL侧重于电路级，可以在门级和寄存器传输级（RTL）描述硬件，也可以在算法级对硬件加以描述。

综合的过程就是将较高层级的设计描述自动转化为较低层次描述的过程，综合分为行为综合，从算法表示、行为描述转换到寄存器传输级；逻辑综合是从RTL级描述转换到逻辑门级，包含触发器的过程，也是在资源计算中最关注的过程，还有版图综合和结构综合，是将逻辑门表示转换到版图表示的过成或转换到PLD器件的配置，版图综合和结构综合以及行为综合均不是我们主要关注的流程，后续介绍将主要介绍逻辑综合这一流程。

综合只是个中间步骤，综合后生成的网表文件，就是由导线相互连接的寄存器传输级功能块 （像是触发器、算术逻辑单元和多路选择器等）组成的。这里就需要一个叫做RTL模块构造器的程序，用来针对用户指定的目标工艺从预定义元件库中构造或获取每一个必须的RTL功能模块。综合产生网表文件后，逻辑优化器会读入网表，并且会按照用户提供的面积（即资源要求）和定时（时序要求）约束为目标来优化网表。并且，这些面积和定时约束也能够指导模块构造器恰当地选取或生成寄存器传输级功能块。 既然综合器能够将HDL语言转换成门级网表，那么综合器必定有一种映射的机制，能够实现Verliog代码中的某些变量、组件映射成对应的硬件元件。

**如图所示：**



**图2-5**

1. 综合首先需要使用RTL代码构成的电路行为级描述进行编译使其转换为GATE级描述；
2. RTL模块构造器从用户指定或综合器所携带的工艺库中里提取并生成对应的模块；
3. 逻辑优化器读取未优化的门级网表，并按照用户设置的面积和时序约束，进行优化；

由于本文中主要涉及计算RTL模块构造器从工艺库中提取对应的语法及模块进行TechMap映射时所得到的资源，所以当我们聚焦于本文所述的基于机器学习资源估算模型，更加关注于Verilog HDL语句是如何转换成硬件的，如：数据类型是如何转换成硬件的，常量是如何转换成逻辑数值的，语句是如何转换成硬件的。下面我们将详细分析下综合中各个Verilog HDL语句及器件的综合原则；

1、储值单元的综合原则

对于FPGA上的实际硬件主要有三种基本的储值单元分别是：wire、锁存器、触发器，对应于Verilog HDL语言中，变量可以是net类型也可以是reg类型，它们的综合原则分别是对于net类型的变量则只能总合成导线即wire类型，对于reg类型变量可能综合为导线即wire类型，rege类型的也可能被从合成锁存器或触发器，但是若reg类型的变量在一个always语句块中作为临时变量，则不会被总综合成存储器。

2、连续赋值语句的综合原则

连续赋值语句（Continuous Assignment）语句的硬件实现是：从赋值语句也就是等号的右边提取出逻辑，用于驱动赋值语句左边的线网即Net，**如图所示：**

****

**图2-6**

3、过程赋值语句的综合原则

过程过程赋值语句（Procejural Assignment）语句对应的FPGA的硬件实现是：从赋值语句也就是‘=’或‘<=’的右边提取出逻辑，用于驱动赋值语句左边的变量，但是有个语法前提，就是用于驱动赋值语句左边的变量是reg类型，且只有Always语句块中的过程赋值语句才可以被综合器综合，若出现在initial语句块中将被仅用于仿真。过程赋值语句有以下两种分别为：阻塞赋值语句（Blocking Assignment Statement）、非阻塞赋值语句(Non-blocking Assignment Statement)

1）阻塞赋值语句综合

阻塞赋值语句简而言之就是在一个always语句块中，语句是从上到下顺序执行，综合时将会在依据赋值语句逻辑原理结合always块的触发器来实现逻辑综合，**如图所示：**

****

**图2-7**

2）非阻塞赋值语句综合

非阻塞赋值语句简而言之是在一个always块中，语句是并行执行的，如图所示：

对比上图可以得到对于一个Always块中的阻塞或非阻塞语句的综合最大的区别在于是并行还是顺序执行，但是对于综合后器件的数量，也就是我们所关心的资源数量并没有明显的影响，主要影响使用资源数量的关键在于赋值语句右边的驱动逻辑的组成；



**图2-8**

4、逻辑运算的综合原则

Verilog HDL语句中的逻辑运算符将会被直接综合成对应的基本逻辑门,如与或非、异或门等。



图2-9

5、算术运算符的综合原则：

在Verilog HDL语句中支持各种算术运算符如+、-、/、\*等，在进行综合时RTL构造器会将其构造为加法器、减法器、除法器等源语器件。如图所示：



图2-10

6、关系运算符的综合原则

关系运算符的综合则取决于当前运算类型，综合为基本逻辑运算符，如图所示：

7、always语句块的综合原则

8、if及条件语句的综合原则

9、case语句的综合原则

10、循环语句的综合

对于循环语句，如果循环内部的赋值语句之间不存在相互依赖性，则这些语句可以展开为并行执行的语句，且语句执行先后顺序不会影响到最终的结果

11、函数的综合

在verilog HDL语句中，函数代表了纯组合逻辑。函数在综合的时候，被展开成了内联的代码。任何在函数中定义的局部变量都被当做纯粹的临时变量，仅被总合成导线

**2.2.3 技术映射**

查找表的输入数是既定的，查找表能实现输入数不大于自己的任意逻辑，而FPGA的设计就是要从目标电路的逻辑函数中不断分离出既定输入数之内的逻辑，并将其映射到查找表上。

FPGA使用统一构造的查找表

工艺映射是指将不依赖于任何工艺的门级网表转换为由特定FPGA逻辑单元所表示的网表的过程。这里所说的逻辑单元依赖于特定的FPGA架构，是由查找表或MUX等逻辑电路实现的FPGA上的最小逻辑单位。工艺映射是从HDL开始的逻辑转换的最后一步，因此对最终电路实现的质量(面积、速度、功耗等)至关重要。

下面我们将分析具有代表性的FlowMap的工艺映射原理，工艺映射包含多种工艺映射算法，既将目标电路网表转换到K输入的查找表(k-Lut)的工艺映射过程由下面两个步骤组成：

1. 分解：门级网表实际上都是以布尔网络的形式来表示的。现将布尔网络的各个节点不断分解，直至输入数小于查找表的输入数K。
2. 覆盖：基于过程1得到的布尔网络，使用某种基准对输入进行切分，使用k-Lut覆盖多个节点

覆盖的过程是一种单位时间内找到逻辑层数最优解的办法；

如下图是将网表映射至3-LUT的实例，将通过该实例对工艺映射原理进行说明。首先将图x中的门级网表转换为图x中的DAG有向图的方式来表示。最上层的节点成为主输入，最下层的节点称为主输出，从而可以得到如图x所示的映射范围，图x所示为在映射范围下进行切分和标注求解的过程及将映射范围的有向图依据主输入及主输出进行逻辑分层，从而标注拓扑序列；常见的标注算法如下：

1. 主输入标注为0层；
2. 在所有以主输入作为输入的节点中，寻找可以使用3-LUT进行覆盖的节点，并将其切分出来；
3. 顺序计算已标注节点的相邻节点。当遇到还没标注的节点时，则对其进行计算标注。
4. 当所有的相关节点都被计算完成后，再计算下一层几点的标注，依旧以一个3-LUT作为覆盖范围进行切分；
5. 通过反复计算所有节点的标注，计算终点主输出出的层数为2；

通过上述计算可以得到每一层的标注为依据上层计算而来的最小值，因此可以保证使用最少的逻辑层数。之后对电路的主输出开始进行器件映射，每映射一个后主输出将发生变更，对所有的主输出依次进行算法映射后，可以得到查找表及网表图，如图x。

在对目标电路进行切分和映射的过程中，通过改良评估函数可以实现各种工艺映射算法，如：哥伦比亚大学的Steve Wilton团队开发了改善功耗的EMap，多伦多大学的Stephen Brown开发了IMAP等。