# 基于配置模式匹配和层次化映射结构的高效FPGA码流生成系统研究

涂开辉[①②] 黄志洪[①] 侯峥嵘[①] 杨海钢*[①②]

[①](中国科学院电子学研究所 北京 100190)

[②](中国科学院大学 北京 100049)

**摘 要**：码流生成在FPGA电子设计自动化(EDA)流程中，提供应用电路在芯片上物理实现所需的精准配置信息。现代FPGA的发展一方面呈现出器件规模及码流容量越来越大的趋势，另一方面越来越多可变阵列大小的嵌入式应用(例如eFPGA)又要求码流生成器具备更高的配置效率以及更精简的可重构数据库。针对码流生成时间增加的问题和阵列规模任意缩放的需求，该文提出一种模式匹配和层次映射的码流生成方法，即对编程单元按配置模式进行分类建模，在配置时按模型进行调用匹配，并采用了层次化的码流映射策略，使得数据库可随阵列排布调整动态生成。该方法可有效应对FPGA嵌入式应用中码流容量的增大以及阵列规模可变所带来的挑战，同时相比平面化的建模及映射方法，码流配置的时间复杂度由$O(n)$降低为$O(\lg n)$。

**关键词**：FPGA；码流生成；嵌入式；配置模式；层次化

**中图分类号**：TN402 **文献标识码**：A **文章编号**：1009-5896(2019)11-2585-07

**DOI**: 10.11999/JEIT190143

# Research on Efficient FPGA Bitstream Generation System Based on Mode Matching and Hierarchical Mapping

TU Kaihui[①②] HUANG Zhihong[①] HOU Zhengrong[①] YANG Haigang[①②]

[①](*Institute of Electronics, Chinese Academy of Sciences, Beijing* 100190, *China*)

[②](*University of Chinese Academy of Sciences, Beijing* 100049, *China*)

**Abstract**: Bitstream generator in FPGA Electronic Design Automation(EDA) offers precise configuration information, which enables the application circuits to be implemented on the target device. On one hand, modern FPGAs tend to have larger device scale and more configuration bits, on the other hand, embedded applications (e.g. eFPGAs) require better configuration efficiency and smaller, more adaptive database. In order to meet these new requirements, a bit-stream generation method is proposed which firstly models the configurable resources by configuration modes and matches the netlist with these models, then hierarchical mapping strategy is used to search every bit on a dynamically generated database determined by the array floorplan. This method well meets the challenges that embedded applications may bring-the surge of configuration bit count and the changeable size of the array. Compared to flattened modelling and mapping method, its time complexity is reduced from $O(n)$ to $O(\lg n)$.

**Key words**: FPGA; Bitstream generation; Embedded; Configuration mode; Hierarchy

## 1 引言

FPGA码流的生成，一般需要依靠芯片厂商所提供的电子设计自动化(Electronic Design Automation, EDA)工具进行。不同的FPGA厂商，所提供的“码流生成”软件也各不相同，其在Xilinx Vivado工具链中，被称为“Bitgen”；而在Intel Quartus工具链中，被叫做“Assembler”。在现代FPGA发展中，随着芯片规模的日益扩大以及机器学习等复杂应用场景的日趋成熟[1]，为了应对生成码流大小、速度、安全性以及灵活性的挑战，码流压缩[2]、分区配置[3]、码流加密[4]以及部分重配技术[5]成为了该领域的热门研究方向。

在工业界，常用的Vivado以及Quartus工具链的码流生成系统未见公开。而在学术界，关于码流生成的研究均和其所依托的数据库(主要包括用户

收稿日期：2019-03-12；改回日期：2019-05-30；网络出版：2019-06-04
*通信作者： yanghg@mail.ie.ac.cn
基金项目：国家自然科学基金(61876172, 61704173)，北京市科技重大专项课题(Z171100000117019)

Foundation Items: The National Natural Science Foundation of China (61876172, 61704173), The Major Program of Beijing Science and Technology (Z171100000117019)

电路以及芯片结构)密切相关。JBits[6], JHDLBits[7]和BitMan[8]均开发了针对Xilinx FPGA的码流修改应用程序编程接口，通过破译码流和芯片资源之间的对应关系，从而实现通过码流对电路的重构。Torc[9]和VDI+RapidSmith2[10]，开发出了相对完整的工具链，但码流生成依然以读入和在帧层修改Xilinx码流为主要手段。这些工具均未以电路网表为出发点，给出码流生成的完整解决方案。Debit[11], BIL[12], Bit2NCD[13]则聚焦于逆向工程，从Xilinx码流中逆向还原电路网表，此类研究由于涉及版权和信息安全，已不再鼓励进行。VTR[14]，是学术界非常流行的FPGA EDA研究平台，在其最新的7.0版本中使用了XML格式参数化的描述其硬件结构，以及使用BLIF格式描述其电路设计，但由于其仅作为硬件的参数化评估工具，并未依托实际的芯片，因此未将码流生成工具集成进去。VTR-to-Bitstream[15]和其升级版本[16]借助Torc和RapidSmith试图将VTR和工业界的实际芯片相结合，但仅做了网表转换，码流生成则直接使用了Xilinx的Bitgen。Bitmerge[17]，通过修改Xilinx XDL网表文件来产生数据库，只能针对Xilinx器件，且在码流映射方面没有自己的阐述而是直接使用了Torc数据库。文献[18]仅描述了对FPGA互连资源的配置位提取方法，但并没有涉及逻辑模块内部的配置，并且码流映射使用了平面化的方法，虽然提取效率比人工高，但在后面实际的码位配置速度上依然会随着芯片规模的增大很快遇到瓶颈。

近年来越来越大的器件规模(数千万门)和更多的嵌入式应用(eFPGA)，给现有的码流生成工具带来以下新的问题：(1) 和器件结构绑定紧密，方法无法通用；(2) 器件码位容量增大带来的码流生成时间增加；(3) 数据库过大且无法动态适应不同定制芯片资源大小。本文针对以上挑战，提出了一种适用于业界先进FPGA器件的通用码流生成方法，本文方法对编程单元按配置模式进行分类建模，使其可以和器件架构解除绑定关系；采用层次化的码位映射策略以降低配置搜索时间；建立可重构的数据库以满足可变阵列大小的设计需要。

## 2 概念定义

为方便阐述，首先介绍本文中所用基本概念的定义。

### 2.1 逻辑资源结构(Logic Resource Structure, LRS)

在现代FPGA中，如图1所示，逻辑模块和互连资源可以被划分为若干列状排列的单元，称作复用单元(tile)。每个复用单元类型都包含了若干基本单元(primitive)，作为码流配置的最小单位。(复用单元和基本单元之间可能还会有中间层次(subtile)，为简明起见，本文在阐述时采用复用单元/基本单元这两层结构)，每个基本单元均包含1个或多个编程点，每个编程点拥有和其相关的配置位(bit)。

每个码位在LRS中的地址，称为其逻辑地址。典型的逻辑地址表示为：tile.primitive.bit。

### 2.2 配置码位结构(Configuration Bit Structure, CBS)

配置码位结构CBS，通常由配置控制器的设计确定，包括配置存储器(CRAM)结构、块存储器(BRAM)结构以及其他各类控制寄存器。在现代FPGA中，CBS通常以时钟域分区(sector)，每个时钟域分区内可划分为若干帧(frame)，帧通常垂直排布，避免横跨多种类型的复用单元资源，利于部分重配。每帧中包含若干配置位(bit)。由于每个分区可拥有各自的配置控制器从而进行独立配置，本文将分区看作一个小型器件进行分析(参看图1)。

每个码位在CBS中的地址，称为其物理地址。典型的物理地址表示为：frame.bit。

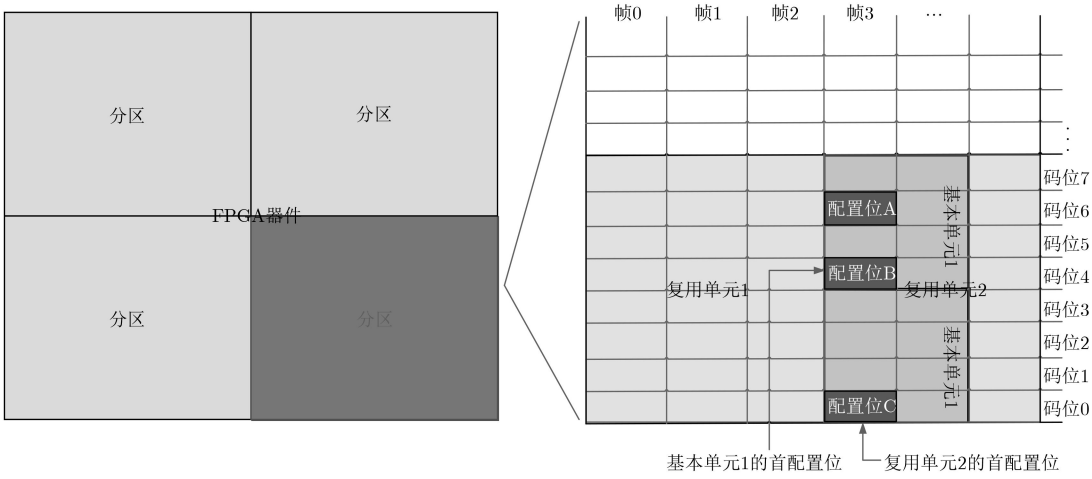


图 1 配置码位结构CBS示意

对于配置位A，其在不同层级上有着不同的物理地址：基本单元级：(frame0.bit2)，复用单元级：(frame0.bit6)，器件级：(frame3.bit6)。

## 3　依赖数据库

### 3.1　器件结构数据库(DeVice Database, DVD)

器件结构数据库DVD包含了所有和码流生成相关的FPGA芯片硬件物理信息，主要包括：

(1) 各级单元首配置位物理地址信息(primitive/tile_first_addresses)

由于同类型单元的配置位结构是完全一样的，因此该信息可以通过提取芯片复用/基本单元级设计网表来获取它们的配置位阵列尺寸(frame×bit)，并结合其逻辑资源排布情况计算得到每个坐标下单元的首配置位地址。

(2) 码流初始化信息(bitstream_initial)

该信息包含了器件中所有编程点配置成“initial”模式后生成的码流，可预先按复用单元生成，在码流配置之前直接读入以节省时间。

(3) 码流格式信息(bitstream_format)

该信息由配置控制器决定，通常包括码流长度，帧的格式，各控制寄存器的值等。

### 3.2　电路设计数据库(DeSign Database, DSD)

电路设计数据库DSD包含了所有和码流生成相关的电路网表物理实现(implementation)信息。这个数据库随着FPGA EDA流程的进行而不断变得丰富，直到完成电路在芯片上的全部物理实现。

本文方法所使用的DSD中，编程点的配置信息以簇/分子(cluster/molecule)的结构来组织(同样为简化描述，采用两层结构)，层级和电路设计数据库中的复用/基本单元(tile/primitive)对应。因此，电路的物理实现从某种程度上来说，就是为电路设计数据库中的簇/分子在器件结构数据库中找到可以容纳其功能的复用/基本单元。

## 4　码流生成流程

本文的主要思想是将码流配置的过程转化为一个模式查找匹配以及地址分级映射问题。在生成码流之前，先对所有类型编程点的配置模式进行建模，并确保上游的EDA流程已经将所有编程点的配置信息存入电路设计数据库。对于电路设计中所有的编程点，确定其处在哪种配置模式下，并获取该模式下所要配置的码位逻辑地址，最后通过映射(mapping)获取这些码位的物理地址(如图2)。

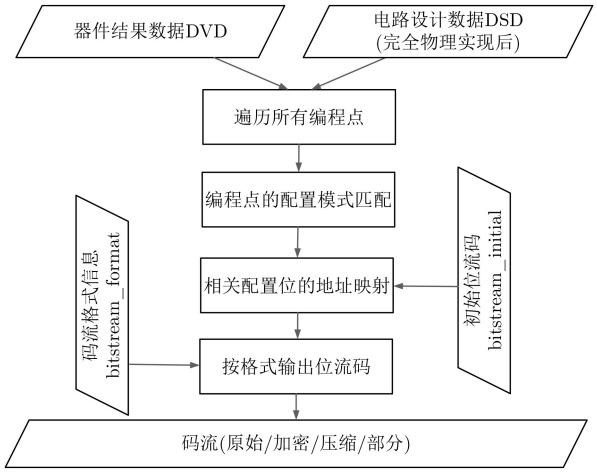


图 2 码流生成流程

### 4.1　配置模式分类建模

编程点是SRAM型FPGA中最基本的编程单元，通常由一个或多个SRAM位控制，并独立实现一个基本功能。每个编程点均可被编程为若干种状态，每一种可能的编程状态称为该编程点的一个配置模式(configmode)，即1组左值-右值对(如图3)。编程点一般可分作两类：

(1) 逻辑编程点(Programmable Logic Point, PLP)：表示逻辑块资源的编程点，其配置模式可

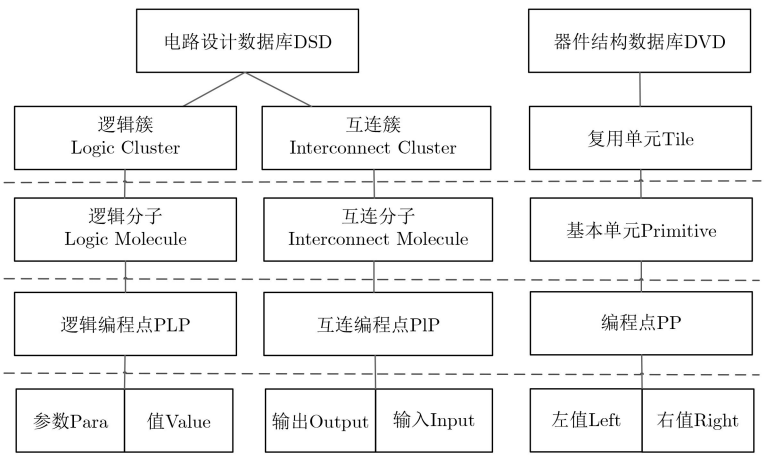


图 3 电路设计数据库-器件结构数据库层次化对应关系

被物理实现过程定义为一个1对多的参数对：左值parameter和右值value(1个parameter对应多个value)。

(2) 互连编程点(Programmable Interconnect Point, PIP)：表示互连资源的编程点，其配置模式可被物理实现过程定义为一个1对多的参数对：左值output pin/wire和右值input pin/wire(1个output对应多个input)。

配置模式可以使用XML格式来进行建模，而配置模式模型可分为以下3类：

(1) 枚举配置模型

当编程点的配置模式较少，且每个模式配置下的配置值均为确定值，通过枚举将其配置模式列出。

(2) 公式配置模型(独立)

当编程点的配置模式过多不便枚举(例如LUT的掩码值、PLL的占空比等，每一个可能值都可看作一个配置模式)时，则该模型仅列1个模式(除initial)，该模式下所需配置位(BIT)的配置值仅和该编程点的右值相关，需通过仅与该右值相关的公式计算(公式配置模型示例如图4所示)。

(3) 公式配置模型(依赖)

当编程点的配置模式较少，但配置位的值不确定时(同时依赖其他编程点的配置情况)，需通过同时与当前编程点的右值以及依赖编程点的右值都相关的公式计算。

### 4.2 码位分层映射

在典型的电路设计数据库中，本文只需依次遍历每个已确定好位置的“簇-分子-编程点”，通过簇的位置确定其所在复用单元的首配置位物理地址FAt，再通过分子的位置确定其所在基本单元的首配置位物理地址FAp，最后根据该编程点的参数对：parameter/value(PLP)和in/out(PIP)匹配获得需要配置的位在其所在基本单元中的物理地址Ap。最后将所有地址相加获得该配置位的最终物理地址Af并进行配置。对于需要配置的复用单元，在其初始码流上进行配置。

基于以上分析，如果映射的目标器件中所有的码位总数为$n$，平面化的码位映射需要在$n$的范围内搜索每一个配置位(时间复杂度为$O(n)$)，采用层次化映射后，假设最底层基本单元的平均码位地址总数为$p$，并共分$l$层级映射(本文为了简化结构说明使$l$=2，即基本单元级、复用单元级和器件级，真实器件中可能还会有中间层级，即$l$>2)，每一层的平均模块单元数为$m$(即在该层的搜索次数，常数)，因此$n$=$p$×$m^l$，而层次化的映射方法的搜索次数为

$$\begin{aligned}\mathrm{St} &= p + m \times l = p + m \times (\log_m(n/p)) \\ &= m\log_m n - m\log_m p + p\end{aligned} \tag{1}$$

因此时间复杂度降为$O(\log_m n)$，即$O(\lg n)$。就配置时间而言，假设平面化搜索的时间为T1，层次化搜索的时间为T2，每一个配置位在primitive级搜索时间为Sp(常数)，在各单元层搜索的时间是Sm(常数)，需要配置的总位数为$N$(常数)，则可得到

$$\begin{aligned}\mathrm{T2} &= (\mathrm{Sp}+\mathrm{Sm}\times l)\times N = (\mathrm{Sp}+\mathrm{Sm}\,(\log_m(\mathrm{T1}/N/\mathrm{Sp})))\\ &\times N = (\mathrm{Sm}\ \log_m \mathrm{T1} - \mathrm{Sm}\cdot\log_m N\\ &-\mathrm{Sm}\ \log_m \mathrm{Sp} + \mathrm{Sp})\times N\end{aligned} \tag{2}$$

从此关系可看出，其时间复杂度依然是$O(\lg n)$。

## 5 实验结果

本方法可以在以下4个方面进行评估：

(1) 芯片资源覆盖

配置模式几乎可以为目前所有主流的逻辑模块

```
(1) <PRIMITIVE type="lut">
(2)     <PP name="lut_mask">
(3)         <MODE name="initial">
(4)             <BIT addr="0,1" value="0">
(5)             <BIT addr="0,2" value="0">
(6)             <BIT addr="0,3" value="0">
(7)             <BIT addr="0,4" value="0">
(8)             ...
(9)         <MODE>
(10)        <MODE name="lut_mask">
(11)            <BIT addr="0,1" value="lut_mask/16">
(12)            <BIT addr="0,2" value="lut_mask%16/8">
(13)            <BIT addr="0,3" value="lut_mask%8/4">
(14)            <BIT addr="0,4" value="lut_mask%4/2">
(15)            ...
(16)        <MODE>
(17)    </PP>
(18)    ...
(19)</PRIMITIVE>
```

图 4 公式配置模型示例

和布线资源建模,包含Xilinx Slice/Intel ALM等基本逻辑单元、各类布线开关资源、时钟网络以及MEM/DSP/PLL等主流IP核。

(2) 芯片相关数据库大小

比较数据库大小所用的器件结构(DVD)模型包含ALM结构的簇(cluster)、MEM, DSP, PLL 3种异构IP以及各类IOB等10种复用单元，表1列出了在不同器件资源的情况下，码流生成所依赖的芯片数据库大小。

对于同系列芯片来说，由于共享了复用单元库，因此以复用单元为单位进行描述的primitive_first_addresses，initial_bitstream和config_modes数据库均保持不变。而tile_first_address由于在芯片层级进行描述，因此受到芯片规模(tile数目)的影响并随之增大。在器件模型为3 M Gates的情况下数据库总大小为765 kB，而在器件模型增大到90 M Gates左右时(相当于Xilinx Virtex7系列规模)，数据库总大小仅为1 MB左右。在eFPGA的评估过程中，只需根据动态生成的器件排布情况，为每个坐标下的复用单元生成相应的tile_first_addresses数据库，即可实现所依赖的数据库随芯片资源重组进行自适应调节。

(3) 码流配置时间

码流配置时间测试的用例分为器件结构(DVD)模型用例以及电路设计(DSD)模型用例，由于本实验仅测试码流配置所需时间，因此可以不考虑电路实际功能以及器件实际结构，而采用参数化的方法来构建两种模型。

其中，DVD模型选择平均码位总数$p$为100左右的5个primitive模块(平均模块单元数$m=5$，primitive由1个ALM加1个InputMux组成)拼接为第1级subtile，然后再以5个第1级subtile拼接为第2级subtile，以此类推来构建拥有任意映射层数的器件模型，映射层数$l$=subtile层级总数+1，如图5。

而DSD模型则可以通过读入图6中的Cluster XML文件在相应坐标的cluster下注入PP，来进行构建。只需指定需要添加配置信息的坐标(loc_x, loc_y, loc_s …视层级而定)即可构造出占用任意资源大小的电路设计模型。

在3 GHz 2-Core CPU, 8 GB RAM的PC平台

**表 1　芯片相关数据库大小(kB)**

| 器件模型* | config_modes | tile/primitive_first_addresses | initial_bitstream | bitstream_format_info | 数据库总大小(kB) |
|---|---|---|---|---|---|
| 器件a(3 M Gates) | 512 | 10/15 | 226 | 2 | 765 |
| 器件b(10 M Gates) | 512 | 32/15 | 226 | 2 | 787 |
| 器件c(30 M Gates) | 512 | 99/15 | 226 | 2 | 854 |
| 器件d(50 M Gates) | 512 | 158/15 | 226 | 2 | 913 |
| 器件e(70 M Gates) | 512 | 210/15 | 226 | 2 | 965 |
| 器件f(90 M Gates) | 512 | 268/15 | 226 | 2 | 1023 |

*注：器件模型均属一个系列，该系列包含10种复用单元

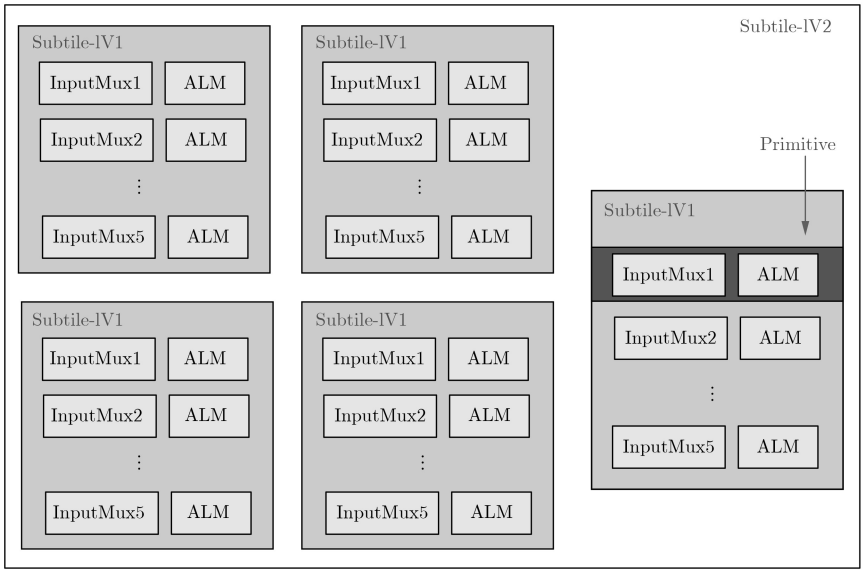


图 5 DVD测试模型示意

```
(1) <!--After Implementation-->
(2) <DESIGN name="dsd1" series="s1" device="d0">
(3)     <CLUSTER name="clb01" type="clb" loc_x="5" loc_y="4">
(4)         <MOLECULE name="alm01" type="alm" loc_s="0">
(5)             <ATTRIB name="comb_01_operation_mode" value="normal">
(6)             <ATTRIB name="comb_01_lut_mask" value="AA0000">
(7)             <ATTRIB name="ff_01_power_up" value="dont_care">
(8)             ...
(9)         </MOLECULE>
(10)        ...
(11)    </CLUSTER>
(12) </DESIGN>
```

图 6 DSD测试模型示意

上构造了6个电路设计(DSD模型)从5.8 k个配置位(分布在120个molecule)到18.4 M左右个配置位(分布在380700个molecule)不等(如表2)，6种器件结构(DVD模型)由于映射层数不同，资源规模从25.8 k Gates到80.6 M Gates不等(如表3)。

**表 2 不同电路设计、相同芯片规模(同系列)下的码流配置时间**

| 电路设计 | 需要配置的码位总数(bit) | 码流配置时间(s) |
|---|---|---|
| 电路1(用满25 k Gates器件资源) | 5.8 k | 0.016 |
| 电路2(用满1 M Gates器件资源) | 250.4 k | 0.682 |
| 电路3(用满10 M Gates器件资源) | 2.3 M | 6.360 |
| 电路4(用满30 M Gates器件资源) | 7.0 M | 19.419 |
| 电路5(用满50 M Gates器件资源) | 11.5 M | 30.334 |
| 电路6(用满80 M Gates器件资源) | 18.4 M | 50.886 |

**实验1** 均使用器件6(80.6 M Gates)时的码流配置时间如表2所示(全芯片码位总数≈39 M bit)。

**实验2** 均使用电路1(用满25 k Gates器件资源)时的码流配置时间如表3所示(需要配置的码位总数≈5.8 k bit)。

由于本文方法需要对每个待配置的码位进行匹配搜索，因此在电路需要配置的码位总数增大时，码流配置时间随其线性增大(如表2所示)，但在芯片规模增大时(如表3所示)，实验结果验证了4.2节关于层次化映射码流配置时间t2的分析(式(2))，即平面化的映射方法会使得配置效率随着芯片规模的增大而急剧恶化，而采用层次化的映射方法，码流配置时间则可得到极大地降低(如图7)。

(4) 和其他具备码流生成功能的学术工具比较，结果如表4所示。

**表 3 相同电路设计、不同芯片规模(同系列)下的码流配置时间**

| 器件模型 | 映射层数$l$ | 全器件的码位总数$n$(bit) | 平面化映射码流(传统方法)配置时间t1(s) | 层次化映射码流(本文方法)配置时间t2(s) |
|---|---|---|---|---|
| 器件1(25.8 k Gates) | 3 | 12.5 k | 1.4 | 0.013 |
| 器件2(130 k Gates) | 4 | 62.5 k | 7.0 | 0.013 |
| 器件3(645 k Gates) | 5 | 312.5 k | 34.1 | 0.014 |
| 器件4(3.1 M Gates) | 6 | 1.5 M | 155.2 | 0.015 |
| 器件5(16.1 M Gates) | 7 | 7.8 M | 820.2 | 0.015 |
| 器件6(80.6 M Gates) | 8 | 39 M | 4066.0 | 0.016 |

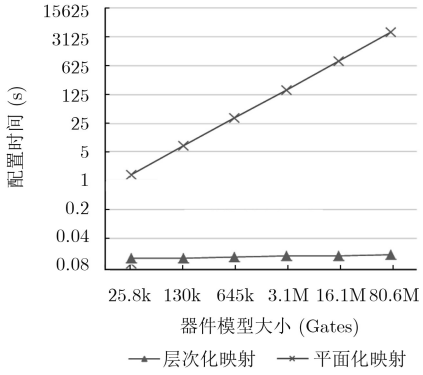


图 7 码流配置时间随器件模型的变化关系($m$=5)

## 6 结论

本文基于模块化的配置模式模型和层次化的码位映射方法，提出一种适用于主流FPGA架构的高效比特级码流生成系统设计，本文方法所依赖的数据库规模小且可随芯片资源变化自适应调节，解决了阵列规模可变且对存储资源敏感的嵌入式FPGA核的配置编译问题。实验结果表明，与传统平面化建模及映射方法相比，本文的配置模式匹配和映射结构层次化方法显著降低了码流配置的时间复杂度，即由$O(n)$减至$O(\lg n)$。

表 4　本方法和其他码流生成工具的特性比较

| | Torc[9] | RapidSmith2[10] | 文献[18] | 本文方法 |
|---|---|---|---|---|
| 数据库通用性 | 仅针对Xilinx器件 | 仅针对Xilinx器件 | 通用 | 通用 |
| 码流生成层次 | Frame级 | Frame级 | Bit级 | Bit级 |
| 涵盖资源 | 全芯片资源 | 全芯片资源 | 仅互连资源 | 全芯片资源 |
| Bit级映射方法 | – | – | 平面化 | 层次化 |

## 参 考 文 献


[1] 王俊, 郑彤, 雷鹏, 等. 深度学习在雷达中的研究综述[J]. 雷达学报, 2018, 7(4): 395–411. doi: 10.12000/JR18040.
WANG Jun, ZHENG Tong, LEI Peng, *et al*. Study on deep learning in radar[J]. *Journal of Radars*, 2018, 7(4): 395–411. doi: 10.12000/JR18040.

[2] IŠA R and MATOUŠEK J. A novel architecture for LZSS compression of configuration bitstreams within FPGA[C]. Proceedings of the IEEE 20th International Symposium on Design and Diagnostics of Electronic Circuits & Systems, Dresden, Germany, 2017: 171–176.

[3] Intel Inc. UGS10CONFIG-Intel stratix 10 configuration user guide[EB/OL]. https://www.intel.com/content/dam/altera-www/global/en_US/pdfs/literature/hb/stratix-10/ug-s10-config.pdf, 2018.

[4] ABDELLATIF K M, CHOTIN-AVOT R, and MEHREZ H. Protecting FPGA bitstreams using authenticated encryption[C]. Proceedings of the IEEE 11th International New Circuits and Systems Conference, Paris, France, 2013: 1–4.

[5] Xilinx Inc. UG909-Vivado design suite user guide partial reconfiguration[OL]. https://www.xilinx.com/support/documentation/sw_manuals/xilinx2016_1/ug909-vivado-partial-reconfiguration.pdf, 2018.

[6] PATTERSON C and GUCCIONE S A. JBits ™ design abstractions[C]. Proceedings of the 9th Annual IEEE Symposium on Field-Programmable Custom Computing Machines, Rohnert Park, USA, 2001: 251–252.

[7] POETTER A, HUNTER J, PATTERSON C, *et al*. JHDLBits: The merging of two worlds[C]. Proceedings of the 14th International Conference Field Programmable Logic and Application, Leuven, Belgium, 2004: 414–423.

[8] PHAM K D, HORTA E, and KOCH D. BITMAN: A tool and API for FPGA bitstream manipulations[C]. Proceedings of 2017 Design, Automation & Test in Europe Conference & Exhibition, Lausanne, Switzerland, 2017: 894–897.

[9] STEINER N, WOOD A, SHOJAEI H, *et al*. Torc: Towards an open-source tool flow[C]. Proceedings of the 19th ACM/SIGDA International Symposium on Field Programmable Gate Arrays, Monterey, USA, 2011: 41–44.

[10] TOWNSEND T and NELSON B. Vivado design interface: An export/import capability for vivado FPGA designs[C]. Proceedings of the 27th International Conference on Field Programmable Logic and Applications, Ghent, Belgium, 2017: 1–7.

[11] NOTE J B and RANNAUD É. From the bitstream to the netlist[C]. Proceedings of the 16th International ACM/SIGDA Symposium on Field Programmable Gate Arrays, Monterey, USA, 2008: 264.

[12] BENZ F, SEFFRIN A, and HUSS S A. Bil: A tool-chain for bitstream reverse-engineering[C]. Proceedings of the 22nd International Conference on Field Programmable Logic and Applications, Oslo, Norway, 2012: 735–738.

[13] DING Zheng, WU Qiang, ZHANG Yizhong, *et al*. Deriving an NCD file from an FPGA bitstream: Methodology, architecture and evaluation[J]. *Microprocessors and Microsystems*, 2013, 37(3): 299–312. doi: 10.1016/j.micpro.2012.12.003.

[14] ROSE J, LUU J, YU Chiwai, *et al*. The VTR project: Architecture and CAD for FPGAs from verilog to routing[C]. Proceedings of the ACM/SIGDA International Symposium on Field Programmable Gate Arrays, Monterey, USA, 2012: 77–86.

[15] HUNG E, ESLAMI F, and WILTON S J E. Escaping the academic sandbox: Realizing VPR circuits on Xilinx devices[C]. Proceedings of the IEEE 21st Annual International Symposium on Field-Programmable Custom Computing Machines, Seattle, USA, 2013: 45–52.

[16] HUNG E. Mind the (synthesis) gap: Examining where academic FPGA tools lag behind industry[C]. Proceedings of the 25th International Conference on Field Programmable Logic and Applications, London, UK, 2015: 1–4.

[17] SONI R K, STEINER N, and FRENCH M. Open-source bitstream generation[C]. Proceedings of the IEEE 21st Annual International Symposium on Field-Programmable Custom Computing Machines, Seattle, USA, 2013: 105–112.

[18] 李智华, 黄娟, 李威, 等. 一种SRAM型FPGA互连资源的位流码配置方法[J]. 太赫兹科学与电子信息学报, 2016, 14(1): 136–142. doi: 10.11805/TKYDA201601.0136.
LI Zhihua, HUANG Juan, LI Wei, *et al*. An automatic approach for bitstream configuration of routing resource in SRAM FPGA[J]. *Journal of Terahertz Science and Electronic Information Technology*, 2016, 14(1): 136–142. doi: 10.11805/TKYDA201601.0136.



涂开辉：男，1984年生，博士生，助理研究员，研究方向为大规模集成电路设计自动化.

黄志洪：男，1984年生，博士，助理研究员，研究方向为可编程逻辑芯片设计技术.

侯峥嵘：男，1994年生，研究方向为大规模集成电路设计自动化.

杨海钢：男，1960年生，博士生导师，研究员，研究方向为可编程逻辑芯片设计技术，大规模集成电路设计自动化.