一种新的高速报文解析结构研究

董永吉*① 郭云飞①② 黄万伟① 夏军波③
①(国家数字交换系统工程技术研究中心 郑州 450002)
②(解放军理工大学 南京 210007)
③(解放军防空兵学院 郑州 450052)

摘 要：随着新协议的不断涌现和网络速率的迅猛增长，报文解析结构在解析灵活度和解析速率上面临挑战。该文结合流水线设计和二叉 trie 树查表思想，提出一种应用于路由转发的报文协议解析结构(Parsing Pipeline Architecture for Forwarding, PPAF)，通过构建协议二叉 trie 树来支持报文协议解析的灵活度，利用硬件多级流水查表提升报文协议解析处理速率，采用节点映射算法解决协议二叉 trie 树节点到流水线映射过程中存储资源不均衡的问题。基于 NetFPGA 平台的仿真结果表明，相对于现有的高速解析结构，PPAF 在处理速率和资源占用上取得较好的均衡的同时，能够提供基于接口的独立灵活解析能力。

关键词：信息处理；报文解析；二叉 trie 树；网络虚拟化；NetFPGA
中图分类号：TP393 **文献标识码**： A **文章编号**：1009-5896(2013)05-1083-07
DOI: 10.3724/SP.J.1146.2012.00344

A New High-speed Packet Parsing Architecture

Dong Yong-ji① Guo Yun-fei①② Huang Wan-wei① Xia Jun-bo③
①(*National Digital Switching System Engineering Technological R & D Center*, *Zhengzhou* 450002, *China*)
②(*The PLA University of Science & Technology, Nanjing* 210007, *China*)
③(*Air Forces Defence Command Academy of PLA*, *Zhengzhou* 450052, *China*)

Abstract: With the increasing number of new protocols and the rapid growth of the network link rate, the packet parsing architecture has been greatly challenged on its flexibility and rate. While combining the idea of pipeline design and binary-trie, a new parsing architecture is proposed in this paper, namely Parsing Pipeline Architecture for Forwarding (PPAF). It flexibly analysis packet protocol by constructing Forwarding Protocol-trie, improved the processing rate by employing hardware pipeline look-up table, and solved the unbalance of node mapping storage resource by using the node to pipeline mapping algorithm. The simulation results through the NetFPGA platform suggest that PPAF is superior than the extant high speed parsing architecture in two ways: PPAF achieves ambidexterity in processing speed and resource consumption; and it can provide independent interface-based flexible protocol parsing capabilities.

Key words: Information processing; Packet parsing; Binary-trie; Network Virtualization; NetFPGA

1 引言

近年来，大量的新协议和技术[1]的涌现及随着网络虚拟化[2,3]研究的兴起，网络节点需要能够实时地调整报文解析能力，以适应网络创新实验及网络业务动态变化的组网需求。斯坦福大学推出的 NetFPGA[4]平台，Anwer 等人[5]提出的 Swithblade 平台，Xie 等人[6]设计的可编程虚拟化路由器 PEARL 都为互联网创新提供了共享性、专用性和基础性的试验环境。同时，为了加快新协议的解析和部署，Kozanitis 等人[7]提出利用 TCAM (Ternary Content Addressable Memory)和 hash 配合查表的 Kangaroo 报文解析系统，可达到 40 G/s 的链路处理速率，但由于 TCAM 高功耗的缺点[8]，限制了系统的可扩展性；Attig 等人[9]提出了一种可以简单直观描述报文解析的语言 PP(Packet Parsing)，根据 PP 编译器可以离线生成一个协议解析处理器，经在 XILINX FPGA 上测试可达到 400 G/s 的处理能力，但由于算法占用资源巨大，导致通用可移植性较差。

本文将流水线设计和二叉 trie 树查表思想加以

2012-03-29 收到，2013-03-22 改回
国家 973 计划项目(2012CB315901)，国家 863 计划项目(2011AA01A103)和国家科技支撑计划(2011BAH19B01)资助课题
*通信作者：董永吉 yongjid@gmail.com

结合，设计了一种应用于路由转发的协议解析结构(Parsing Pipeline Architecture for Forwarding, PPAF)，通过 NetFPGA 实验平台对结构进行了仿真验证，取得了良好的效果。

2 PPAF 结构介绍

PPAF 由转发协议二叉 trie 树(Forwarding Protocol-trie，FP-trie)和流水线结构两部分组成，如图 1 所示。转发协议二叉 trie 树将传统报文解析过程抽象成一种二叉 trie 树的查表描述，达到解析的灵活性表示；并采用节点映射算法(Node To Pipeline, NTP)将 FP-trie 树节点映射到流水结构上，通过硬件并行流水查找 FP-trie 树节点信息，实现报文协议解析。

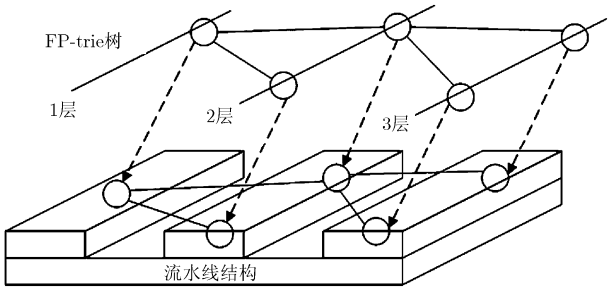

图 1 PPAF 结构组成图

2.1 转发协议二叉 trie 树

二叉 trie 树是一种用于快速检索的二叉树结构[10]，FP-trie 树依据二叉 trie 树的特点，对链路层和网络层协议以网络转发为目的进行解析，将报文头部的相应协议字段转化称为待查找的关键词，结合网络协议的层次结构和协议公开标准，将协议解析过程转换成一个有序的查找序列。

在 FP-trie 树结构中每个内部节点都包含一个判定协议的规则方法及两个指针，两个指针分别指向两个子节点，最后一级叶子节点包含与当前解析路径匹配的解析结果。在基于 MPLS(Multi-Protocol Label Switching)和 IP 协议作为转发关键词的场景下，如图 2 中(a)协议树 1 和(b)协议树 2 分别描述了协议树 Ethernet→VLAN→MPLS→IPv4→IPv6 和 802.3 SNAP→IPv4 的 FP-trie 树结构。

图 2 中 $P1$ 和 $P13$ 为无法识别的解析结果；$P4$, $P8$ 分别代表了 MPLS 和 IPv6 的协议识别结果，$P6$ 和 $P12$ 为 IPv4 的协议识别结果；其余节点均为协议的判定节点，表 1 对两个 FP-trie 树的节点进行了解释，具体如下所示。

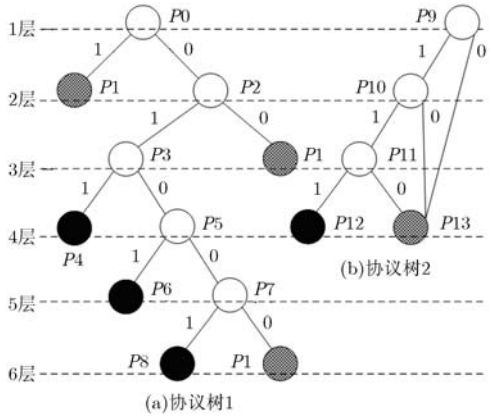

图 2 FP-trie 树结构

结合表 1 给出 FP-trie 树的定义：

定义 1 FP-trie 树叶子节点只包含解析结果，其中包含正常解析内容的称实叶子节点，反之包含无法解析的称虚叶子节点。

定义 2 节点深度为该节点到根节点的最大节点数，定义根节点的深度为 1，则其余节点深度为双亲深度的最大值加 1，并定义 FP-trie 树中最大节点深度为 FP-trie 树的深度。

定义 3 若二叉 trie 树满足如下的约束：

约束 1 每个协议的识别判定可能需要多个定

表 1 FP-trie 树节点内容表

节点名称	判定协议	比较位宽(bit)	规则/结果	判定内容	左子节点数据偏移 (bit)	右子节点数据偏移(bit)
$P0$	Eth II	16	大于	0x05DC	0	32
$P2$	VLAN	16	等于	0x9100	32	0
$P3$	MPLS	16	等于	0x8847	0	0
$P5$	IPv4	16	等于	0x0800	48	0
$P7$	IPv6	16	等于	0x08DD	208	0
$P9$	802.3	16	大于	0x05DC	16	0
$P10$	802.3	16	等于	0xAAAA	48	0
$P11$	IPv4	16	等于	0x0800	144	0

位规则，但每个非叶子节点内只存储一个定位规则和两个跳转的指针，其中定位规则包含该协议相关字段的判定取值及判定方法；

约束 2　每个 FP-trie 树有且仅有一个虚叶子节点；

约束 3　任一非叶子节点必须有两个不同为实叶子节点的子节点；

则称该二叉 trie 树称为 FP-trie 树。

由上述的定义可知，协议二叉 trie 树具有如下的性质：

性质 1　任一个 FP-trie 树描述的协议可以被有限个非叶子节点判别表示。

证明　用反证法证明。假设协议 P 可以被无限个 FP-trie$e$ 树的非叶子节点表示，由约束 1 可知，每个非叶子节点都存储一个协议相关字段判别值，则该协议 P 需要无限个协议字段来表示判定，这与协议的物理实现相悖，故不可能出现。所以 FP-trie 树描述的协议可以被有限个节点来判别表示。 证毕

性质 2　深度为 L 的 FP-trie 树的节点数 S_L，满足 $L \le S_L \le 2^L - 1$。

证明　用数学归纳法证明。

归纳基础：当 L=1 时，L=2^1-1=1 ，则当深度为 1 时，FP-trie 树只有 1 个节点，由定义 2 可知，有且仅有根节点的深度为 1，故命题成立。

归纳假设：假设对所有的 $j(1 \le j < L)$命题成立，即 $j \le S_j \le 2^j - 1$，证明 j=L 时命题亦成立。

归纳步骤：根据归纳假设，深度为 $L-1$ 的 FP-trie 树的节点数 S_{L-1} 满足 $L-1 \le S_{L-1} \le 2^{L-1}-1$。由约束 3 可知，深度为 L 的 FP-trie 树节点数至多比深度为 $L-1$ 的点数多 2^{L-1} 个，故即 j=L 时，深度为 L 的 FP-trie 最多有节点 $2^{L-1}-1+2^{L-1} = 2^L-1$ 个，满足 $S_L \le 2^L-1$；由定义 2 可知，FP-trie 树深度为 L 意味至少有一个节点的父节点的深度为 $L-1$，故 $L-1+1 = L \le S_L$，综上可知，当 j=L 的时候，满足 $L \le S_L \le 2^L-1$，故命题成立。　证毕

综合性质 1 和性质 2 可知，若待解析协议可以用标准的 FP-trie 树表示，则生成的 FP-trie 树节点数存在界限，即表明 FP-trie 树资源占用有限。

2.2 PPAF 流水结构

PPAF中流水线结构用来承载FP-trie树节点，通过流水查找FP-trie树节点的方式实现协议报文解析，如图3所示，硬件流水线结构主要由节点的存储空间双通道RAM、比较器和移位器3个部分组成。其中，双通道的RAM可以在同一时刻并发访问两个

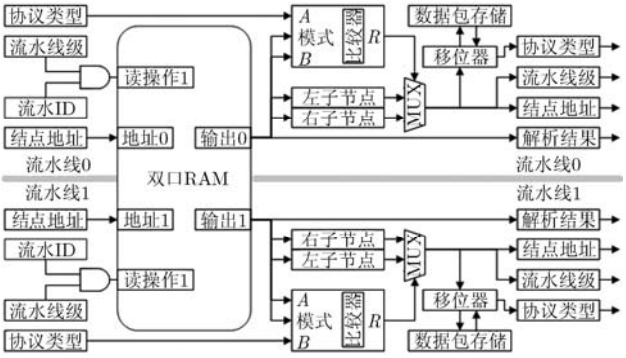

图 3 硬件流水线结构图

地址，因而每一级流水线上任意时刻可以支持两个查表操作，使得报文解析算法提高了一倍的吞吐率。

在每一阶段的流水处理中，首先要判断输入的流水线级内容和本级流水线 ID 是否匹配，如匹配命中则进行下一步操作，否则跳过本级流水线；其次若匹配，则根据输入节点地址在双通道 RAM 中读取出对应的节点内容，按照图 4 描述的数据格式判定该节点是内部结点还是叶子节点，如果是叶子节点则输出解析结果，并且不再进行查找；如果是内部节点则与输入的协议类型在比较器中进行判别比较，选择匹配节点的左或是右子节点作为输出，并根据数据偏移量使用移位器在数据包存储中提取新的数据，最后将节点地址、流水线级和协议类型作为输入，在下一时刻送入后一级流水线阶段。

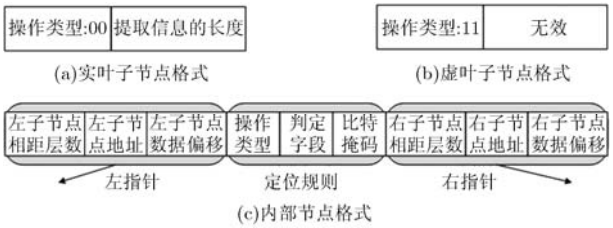

图 4 FP-trie 树的节点数据格式

双通道 RAM 中存储经过映射转换后 FP-trie 树的节点，每个节点的具体字段如图 4 所示：操作类型(2 bit)，用于标识参与比较的协议类型字段与预定值的判定关系，同时也指示节点的类型。

当操作类型为“00”的时候，代表该节点是一个实叶子节点，后续内容为提取信息的长度，如图 4 (a)所示；当操作类型为“11”的时候，如图 4 (b)所示，该节点代表一个虚叶子节点。当操作类型为“01”或是“10”的时候，代表该节点是一个内部节点，“01”对应着比较类型为大于，“10”对应着比较类型为等于，如图 4(c)所示的内部节点格式中的判定字段(16 bit)，为报文协议头部中的相关的协

议字段的判定值，比特掩码(4 bit)用于指示判定字段的无效长度(0–15 bit)；左/右子节点相距层数用于指示 FP-trie 树左/右子节点所在流水线级与其父节点所在流水线级之间的相对距离，左/右子节点地址用于指示 FP-trie 树左/右子节点在流水线上的地址；左/右子节点数据偏移用于指示提取下一个字段需要偏移的比特位个数。

PPAF 结构中比较器用于流水查找时选择匹配的子树，为协议解析判断选择出正确的路径。将报文头部中提取出的 16 bit 协议字段与 RAM 中存储的协议字段预定值进行比较，比较的位宽由比特掩码决定，最长为 16 bit 比较、最短为 1 bit 比较；比较的关系根据操作类型来决定，分为大于、等于两种关系。比较操作的结果若是为真，则选择左子节点对应的存储信息，并输出下一次待解析节点相距本级流水线的距离，下一个解析节点所在流水级上的地址以及下一次从数据包存储中提取相关字段的相对偏移比特位；比较结果若为假，则选择右子节点进行相应操作。

PPAF 结构中移位器用于实现报文信息的萃取，为了节约存储空间，采用相对偏移累加的方法完成萃取报文头部信息。如图 5 所示，使用移位累加寄存器记录上一次移位偏移的量值，并采用该值参与下次偏移值的计算；比特移位器根据前一级的计算结果在数据包存储中寻找萃取信息的起始位置，并按照指定的长度，从起始位置提取连续的比特串作为结果输出。

3 PPAF 结构实现

PPAF 通过在每个接口都对应的建立 FP-trie 树，实现接口独立灵活的解析能力。如图 6 所示，硬件流水线上映射了两个网络接口的 FP-trie 树，$P0$ 和 $P9$ 分别为两个 FP-trie 树的根节点。每次到达系统的报文，首先根据网络接口的不同，将报文送至对应的解析树根节点，每经过一级 FP-trie 树的节点，都会被进行一次协议规则的判断，进而在两个两子树中选择一路进行下一步查找，直到找到叶子节点为止；若在流水线的最后一级匹配到叶子节点，则直接输出结果，若是在前几级流水线上匹配到叶子节点，则该报文仍需要等待未处理的流水线长的时间，用以保证报文的处理时延等长，进而达到结构对报文的顺序处理。

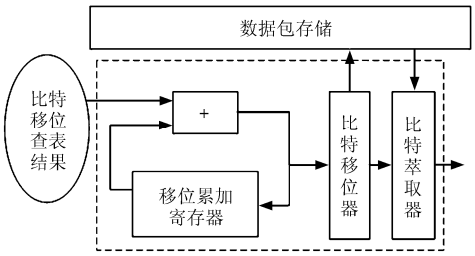

图 5 移位器结构图

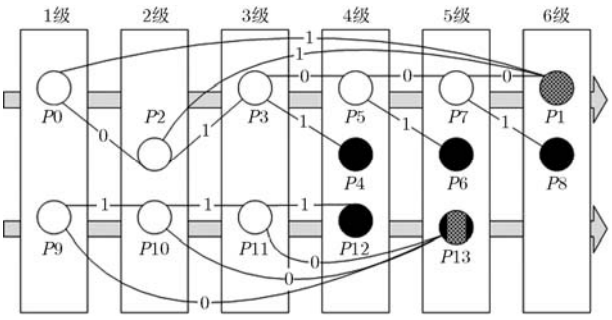

图 6 PPAF 流水查表示意图

3.1 节点映射算法

FP-trie 树节点需要被映射到流水线上才能完成协议解析的功能，最简单的映射方式就是按照 trie 树的深度将节点顺序分配到硬件流水级上，虽然这种简单的方式可以保证流水线正常的线速操作，但是 FP-trie 树结构的不规则性，会导致映射后的流水线存储空间不均衡的占用，降低存储空间的利用率，同时也会影响节点维护的效率，甚至影响系统整体的性能。针对 FP-trie 树映射过程中存储空间的均衡问题，提出最优化的数学模型，首先给出模型中的符号含义，如表 2 所示。

节点映射的最优化数学模型为

$$\min \max_{i=1,2,\cdots,H} D_i \tag{1}$$

条件 1 若 FPtrie_A 和 FPtrie_B 为 FP-trie 树中

表 2 公式及符号的含义

符号	含义
D_i	第 i 级流水线上映射的节点数
n	接口对应结构不同的 FP-trie 树的总数
H	流水线的深度
M_j	第 j 个接口对应的 FP-trie 树节点数
T_i	第 i 个 FP-trie 树
FP-trie	FP-trie 树中的节点
child_node (•)	FP-trie 树节点的子节点
tree_root (•)	FP-trie 树的根节点
tree_size (•)	FP-trie 树的节点总数
tree_depth (•)	FP-trie 树中节点的深度
pipe_depth (•)	FP-trie 树节点映射在流水线的深度
map_condition (•)	节点映射的条件判定规则

节点，$\exists \text{FPtrie}_A=\text{child\_node}(\text{FPtrie}_B)$，则 $\text{pipe\_depth}(\text{FPtrie}_A) \le \text{pipe\_depth}(\text{FPtrie}_B)$。即为了保证硬件流水查找的有序性，FP-trie 树的父节点(ancestor)一定映射在子节点(descendant)所在流水线之前。

条件 2　若 FPtrie_R 为 FP-trie 树的根节点 $\text{FPtrie}_R=\text{tree\_root}(T_i)$，$\exists \text{pipe\_depth}(\text{FPtrie}_R)=1$。

根据条件 1 和条件 2 可知有且仅有 FP-trie 树的根节点映射在第 1 级流水线上，即 $D_1=n$。

条件 3

$$\sum_{i=2}^{H} D_i + n = \sum_{j=1}^{n} M_j \tag{2}$$

结合建立的最优化模型，本文提出了一种类似二叉树前序遍历的启发迭代的节点映射算法 NTP，实现所有 FP-trie 树节点到流水线上的映射关系，算法流程如表 3 所示。其中，$\text{map\_condition}(\bullet)$ 为映射节点的条件判定规则，判断结果为真的节点必须映射到当前对应的流水线上。D_{ave} 代表除了第 1 级流水线外，其余流水线级上节点数量的理论平均值。

表 3 节点映射算法 NTP

	节点映射算法 NTP
(1)	Sort the all the FP-tries in decreasing order of node amount, $\{T_1,T_2,\cdots,T_n\}$, $\text{tree\_size}(T_1) \ge \text{tree\_size}(T_2) \ge \cdots \ge \text{tree\_size}(T_n)$
(2)	for i=1 to n
(3)	FPtrie←tree_root(T_i);
(4)	FPdepth←tree_depth(FPtrie);
(5)	MAP(FPtrie, FPdepth) begin
(6)	if(FPtrie is not mapped)
(7)	if map_condition(FPtrie)=TRUE or $D_{\text{FPdepth}} \le D_{\text{ave}}$ then
(8)	pipe_depth(FPtrie)←FPdepth;
(9)	$D_{\text{FPdepth}} \leftarrow D_{\text{FPdepth}}+1$;
(10)	else
(11)	FPdepth←FPdepth+1
(12)	MAP(FPtrie, FPdepth);
(13)	end if
(14)	end if
(15)	FPdepth ←FPdepth+1
(16)	MAP(FPtrie→lchild, FPdepth);
(17)	MAP(FPtrie→rchild, FPdepth);
(18)	end
(19)	end fo

3.2 FP-trie 树节点更新

根据不同网络业务的组网需求，PPAF 需要动态调整 FP-trie 树节点内容来实现解析能力的灵活变更。而 FP-trie 树的更新对应到节点的变化有 3 种情况：(1)修改协议节点的内容；(2)插入新的节点；(3)删除已有的节点。第(1)种类型的更新相对容易操作，只需要根据节点的存储地址，在相应的流水线上定位到该节点，并更改相应的内容即可完成更新。而第(2)和第(3)种类型的更新相对复杂，由于 FP-trie 树的非叶子节点都有两个子节点，所以任一个节点的插入或删除操作至少涉及到两个节点，而在一些复杂情况下，大量的节点插入或删除操作甚至会导致整个流水线节点分布不均衡，需要及时迅速地完成节点的重新映射。

PPAF 采用了一种在流水线空闲作业中通过插入读写双口 RAM 操作实现节点内容的快速更新。在更新过程中，首先根据流水级 ID 找到匹配的流水线，然后根据节点地址找到需要更新的节点，利用正常协议解析流水查找的空隙，按照读/写指示在存储空间上更新节点。如图 7 所示，利用流水空闲在 t_{i+1} 时刻可以对节点 F 进行更新，在 t_{i+2} 时刻可以更新节点 D，这种方法既能保证协议解析的吞吐量，又可以完成协议解析能力的快速更新。

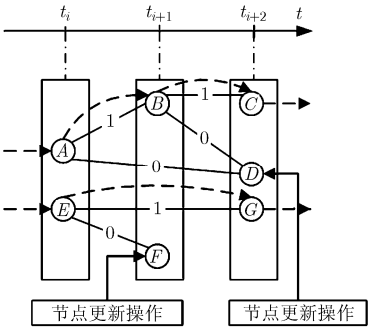

图 7 节点更新方法

4　测试结果

4.1 实验环境设计

本文采用 NetFPGA-10 G 板卡上 Xilinx Virtex-XCVTX240T-2FF1759-2 FPGA 来验证 PPAF 算法和结构。该 FPGA 可用的逻辑资源为 18,720 可编程逻辑单元(configurable logic blocks)、2.400 kbit 分布式存储单元(Distributed RAM)和 11.664(324×36) kbit 块存储单元(Block RAMs)，实验验证使用 Xilinx ISE 13.3 和 Synplify Pro E 2011-3 SP1 工具软件。

鉴于 NetFPGA 板卡的接口数量被限制为 4×10 G，无法进行扩展，为了验证 PPAF 结构的有效

性，在 FPGA 内部模拟搭建了一个 32 路接口的 PPFA 结构，该结构为 32 接口分别提供独立的解析能力，并有一个可配的深度最大为 8 的流水结构。

在流水线的构建上，已知操作类型、判定字段和比特掩码的长度一定，分别为 2 bit, 16 bit 和 4 bit，令 H 为流水线深度，则相邻两个节点间的相距层数用 $\lceil \log_2 H \rceil$ bit 表示，令左/右子节点地址宽度为 w bit，左/右子节点数据偏移分别为 v bit，则图 4 显示的节点占用的比特位宽 m 可以表示为

$$m = 2 + 16 + 4 + 2\lceil \log_2 H \rceil + 2w + 2v \tag{3}$$

$$m = 22 + 2\left(\lceil \log_2 H \rceil + w + v\right) \tag{4}$$

整个结构占用的存储空间 M 大小为

$$M = m \times H \times 2^w \tag{5}$$

由于实验中流水线的最大深度为 8，所以左/右子节点相距层数分别为 $\log_2 8 = 3$。令左/右子节点地址分别为 10 bit，左/右子节点数据偏移分别为 9 bit，则每个节点总共占用 62 bit 的存储空间，根据式(4)，式(5)可知，整个结构共占用 $M = 66 \times 8 \times 2^{10} = 540.472$ kbit 空间。XilinxVirtex-5 FPGA 中每块 Block RAM 最多存储 36 kbit 数据，且可以灵活的配置成两个独立的 18 kbit RAM 或一个 36 kbit RAM(36×1000 bit)。考虑到硬件节点的位宽($36<66<72$)，所以每级别流水线需要 2 块 BRAM 来构建。

为验证节点映射算法的均衡性，对应着虚拟的接口模拟构造产生了 32 个 FP-trie 树，所有 FP-trie 树的最大深度为 8，最小深度为 1，共有 396 个节点，其中实叶子节点 85 个，虚叶子节点 32 个。采用 NTP 算法将全部 32 个接口的 FP-trie 树映射到深度为 8 的流水线上。

4.2 结果与分析

图 8 给出了两种节点映射算法映射后流水线节点数目分布图，其中 simple 方法采用直接对应映射的方式，将 FP-trie 树节点按照其深度对应映射到相同深度的流水线级。通过观察可以发现，NTP 算法除了第 1 级节点相对少一些外，其他流水线级的节点基本按照算法的设计，在流水线上基本均匀分布。而 NTP 算法第 1 级的节点明显少于其他级的流水线节点数目是因为根据映射算法的约束条件 2，第 1 级流水线能且只能存放每个接口 FP-trie 树的根节点，所以第 1 级流水线共存放了 32 个节点。同时，对比可以发现，NTP 算法相对于 simple 方法可以有效解决节点占用存储资源分布不均的问题，并充分优化使用存储空间。

表 4 给出了基于 NetFPGA 平台的不同深度流水线的 PPAF 结构资源与性能对比，其中资源逻辑、BRAM 的占用和时钟频率来源于 ISE 的 Post place and route report 的报告。观察可以发现，随着硬件流水线层数的递增，PPAF 结构中 BRAM 使用的个数逐渐递增，始终与流水线的深度保持 2 倍的关系，与前文关于 BRAM 资源占用的分析一致；而不同深度流水线的对应的时钟频率会随着 BRAM 及逻辑资源的增多而减小，但即使是流水线深度达到 8 时，PPAF 也仅占用了不到 3%的片上逻辑资源，同时 PPAF 的时钟频率仍然可以达到 4.844 ns(206.44 MHz)，由于本结构采用双通道 BRAM，可以支持两个流水查找过程并行进行，则对于最小的以太网包(64 byte)，吞吐量至少可以达到 206.44 MHz×64 ×8 bit×2= 211 Gbps。

表 4 不同深度流水线的 PPAF 结构资源与性能对比表

流水线深度	slice 资源占用(%)	BRAM 占用(36 kbit)	时钟频率(ns)	吞吐量(Gbps)
2	6.4	4	4.472	228
4	8.0	8	4.730	216
6	9.6	12	4.754	215
8	10.8	16	4.844	211

图 9 给出了 PP, Kangaroo 和 PPAF 3 种协议解析结构在 NetFPGA 实验平台下，对协议树 Ethernet→VLAN→MPLS 解析时资源和性能上的对比。如图 9(a)所示，相对于 Kangaroo, PPAF 在 slice 资源的占用上基本持平，但是节约了 10%左右的 BRAM 资源，但在处理能力上，PPAF 的处理能力达到 228 Gbps 远高于 Kangaroo 的 10 Gbps；图 9(b)中的 PPAF^2 代表两个并行处理的 PPAF 结构，由于硬件并行处理的特性，所以 PPAF^2 在资源的占用上是 PPAF 一倍的同时，处理能力也为单个 PPAF 的一倍，达到 228 Gbps×2=456 Gbps，远高于 PP 的 341 Gbps 的处理能力，虽然占用了少量不到 4%的 BRAM 资源，但相对于 PP 系统，也节省了 27%的 slice 资源。

综上，相对于两种已有的算法，PPAF 结构在资源占用较小的同时具有较强的处理能力，在资源占用和处理速率上取得了较好的均衡，并且不同于其他两种解析结构，PPAF 结构为每个接口都建立专用的 FP-trie 树，所以 PPAF 可以为每个接口提供独立的解析能力，更适合未来网络柔性的需求。

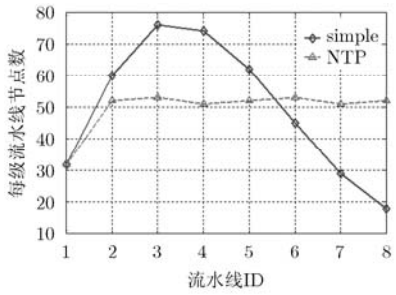

图 8　流水线节点数目对比图

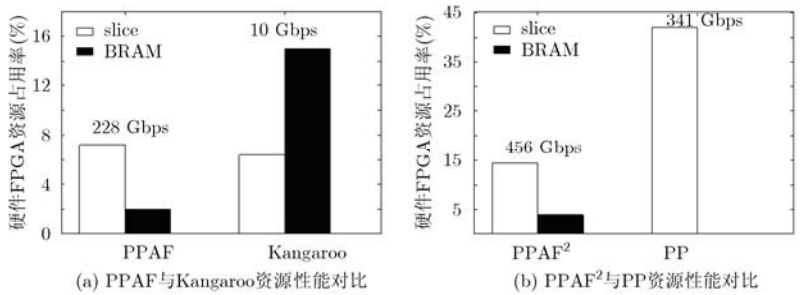

图 9　3 种解析结构资源和性能对比

5　结论

针对传统的报文解析结构在功能上无法适应业务变化、协议处理上无法灵活扩展及资源上不支持虚拟化等问题，本文提出了一种应用于路由转发的报文解析结构 PPAF，该结构采用流水线配合二叉 trie 树来实现协议解析功能高效性和灵活性，采用节点映射算法来均衡各级流水线上的节点数目，以达到优化存储空间的目的，并基于 NetFPGA 实验平台仿真验证了 PPAF 的可行性和有效性，为可重构信息通信基础网络体系研究的实验平台设计提供了参考依据。

参 考 文 献

[1] Kafle V P and Inoue M. Introducing multi-ID and multi-locator into network architecture[J]. *IEEE Communications Magazine*, 2012, 50(3): 104–110.

[2] Khan A, Zugenmaier A, Jurca D, *et al*.. Network virtualization: a hypervisor for the Internet[J]. *IEEE Communications Magazine*, 2012, 50(1): 136–143.

[3] Palkopoulou E, Schupke D A, and Bauschert T. Shared backup router resources: realizing virtualized network resilience[J]. *IEEE Communications Magazine*, 2011, 49(5): 140–146.

[4] Naous J, Erickso D, Covington A, *et al*.. Implementing and deploying an openflow switch on the NetFPGA platform[C]. Proceedings of the 4th ACM/IEEE Symposium on Architectures for Networking and Communications Systems, San Jose, CA, USA, Nov. 2008: 1–9.

[5] Muhammad Bilal Anwer, Murtaza Motiwala, Mukarram bin Tariq, *et al*.. Switchblade: a platform for rapid deployment of network protocols on programmable hardware[C]. Proceedings of the ACM SIGCOMM Conference, New Delhi, India, Aug. 2010: 183–194.

[6] Xie G G, He P, Guan H T, *et al*.. PEARL: a programmable virtual router platform[J]. *IEEE Communication Magazine*, 2011, 49(7): 71–77.

[7] Kozanitis C, Huber J, Singh S, *et al*.. Leaping multiple headers in a single bound: wire-speed parsing using the Kangaroo system[C]. Proceedings of the 29th IEEE Conference on Computer Communications, San Diego, CA, USA, Mar. 2010: 830–838.

[8] Zane F, Narlikar G, and Basu A. CoolCAMs: power-efficient TCAMs for forwarding engines[C]. Proceedings of the 22nd IEEE INFOCOM, San Francisco, USA, 2003: 42–52.

[9] Attig M and Brebner G. 400 Gb/s programmable packet parsing on a single FPGA[C]. Proceedings of the ACM/IEEE Symposium on Architectures for Networking and Communications Systems, Brooklyn, NY, USA, Oct. 2011: 12–23.

[10] Pao D, Lu Z, and Poon Y H. Bit-shuffled trie: IP lookup with multi-level index tables[C]. Proceedings of IEEE International Conference on Communications, Kyoto, Japan, June 2011:1–5.

董永吉：　男，1983 年生，博士生，研究方向为宽带信息网络.

郭云飞：　男，1963 年生，教授，博士生导师，主要研究领域为宽带信息网络.

黄万伟：　男，1979 年生，讲师，研究方向为宽带信息网络.

夏军波：　男，1981 年生，讲师，研究方向为宽带信息网络.