FSL91030(M) chip

Principle Documentation

Manual version: **V1.2**

Wuhan Feisiling Microelectronics Technology Co., Ltd.

December 2022

Table of contents

1Port Description

Port configuration is the first stage of the ingress pipeline, and its main purpose is to obtain the port configuration information related to the data packet entering the chip. The FSL91030M chip has different port concepts, namely physical ports and packet processing ports. In a broad sense, the physical port ID number corresponds to the MAC ID one by one, and the packet processing port corresponds to the logical port (and the panel port) one by one.

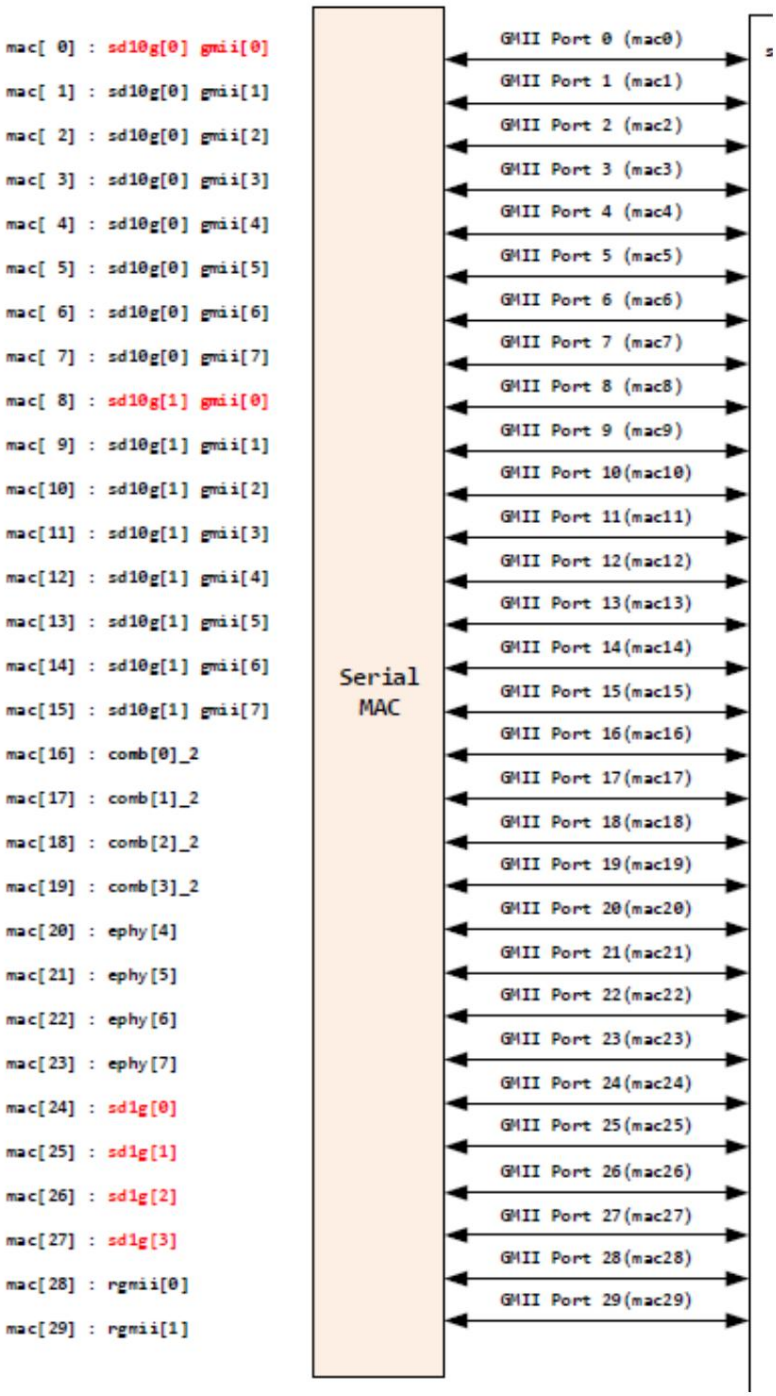
After powering on, FSL91030M first determines the chip's operating mode and port configuration based on the application scenario and configuration file information, and then Determine the configuration information of the port.

The following table shows several common working modes and port configuration information of FSL91030M single chip:

Business Portal **Physical port/mac_id** Port form		**4** electrical ports		**8** electrical ports **6*1G** optical ports				**2*10G** optical ports **+ 8** electrical ports		**4*1G** optical ports **+ 8** electrical ports		**2*10G** optical ports **+ 4** electrical ports **+ 4 comb**	
0		GE0 16	GE0 16	GE0 0	XE0 0	GE0 16	XE0 0						
1		GE1 17	GE1 17	GE1 8	XE1 8	GE1 17	XE1						8
2		GE2 18	GE2 18	GE2 24	GE0 16	GE2 18	Comb0 16						
3		GE3 19	GE3 19	GE3 25	GE1 17	GE3 19	Comb1 17						
4				GE4 20	GE4 26	GE2 18	GE4 20	Comb2 18					
5				GE5 21	GE5 27	GE3 19	GE5 21	Comb3 19					
6				GE5 22				GE4 20	GE6 22	GE0 20			
7				GE7 23				GE5 21	GE7 23	GE1 21			
8								GE6 22	GE8 24	GE2 22			
9								GE7 23	GE9 25	GE3 23			
10										GE10 26			
11										GE11 27			
31	CPU												
28	RGMII0												
29	RGMII1												

FSL91030M has two 10G serdes, four 1G serdes and eight GEPHYs, which can be physically connected to the mac.

Flutter is as follows:



2 VLAN Function

2.1 Terminology

the term	illustrate
Stag	Service VLAN Tag (outer VLAN tag)
Ctag	Customer VLAN Tag (inner VLAN tag)
SOT	Single outer tag
SIT	Single inner tag
DT	Double tag
UT	Untagged
Vid	Vlan ID
Cos	Priority (corresponding to the priority of the tag)
C	Normalized is 0, non-normalized is 1
inIsLag	Indicates whether the input internal logical port number is a LAG port, high effective
inLport	The input internal logical port number indicates a LAG port when inIsLag is valid, otherwise it indicates a common port.
outIsLag	Indicates whether the output internal logical port number is a LAG port, high effective
outLport	The output internal logical port number. When outIsLag is valid, it indicates a LAG port; otherwise, it indicates a normal port.

2.2 Ingress VLAN

The ingress VLAN function mainly includes four parts, namely VLAN Tag check, VLAN translation, VLAN search, and VLAN filtering, as shown in Figure 2-1. After the message comes in, the initial VLAN information is obtained according to the message Tag type and VlanGet module, namely ScosInit, ScfiInit, SvidInit, CcosInit, CcfiInit, CvidInit. Then enter the Vlan editing module to get upd.{svid, scos, scfi, cvid, ccos, ccfi}, when iVtPortSrm.iVtEditEn=0, Vlan editing is disabled, and Vlan information comes from the original message information; when When iVtPortSrm.iVtEditEn=1, the Vlan editing function is enabled. Vlan editing requires knowing VlanOpIdx and the port used to add or replace. vlan information (svid, scos, scfi, cvid, ccos, ccfi). There are four ways to obtain this information, namely ProtoVlan, FlowVlan, XlateVLan, Port, by looking up the corresponding table entries iVtProtoVlanSrm, iVtFlowVlanTcmSrm, iVtXlateSrm (including left 0-3 and right 0-3, a total of 8 entries, used to prevent hash conflicts), iVtPortSrm acquisition. You can enable the corresponding The Vlan editing module obtains VlanOpIdx and queries iVtVlanOpSrm by indexing VlanOpIdx to edit the original message. When performing Vlan lookup, you can use PktSvid or UpdSvid by setting iNetPortSrm.usePktSvid=1 or iNetPortSrm.useUpdSvid=1 to obtain the internal Vlan Id.

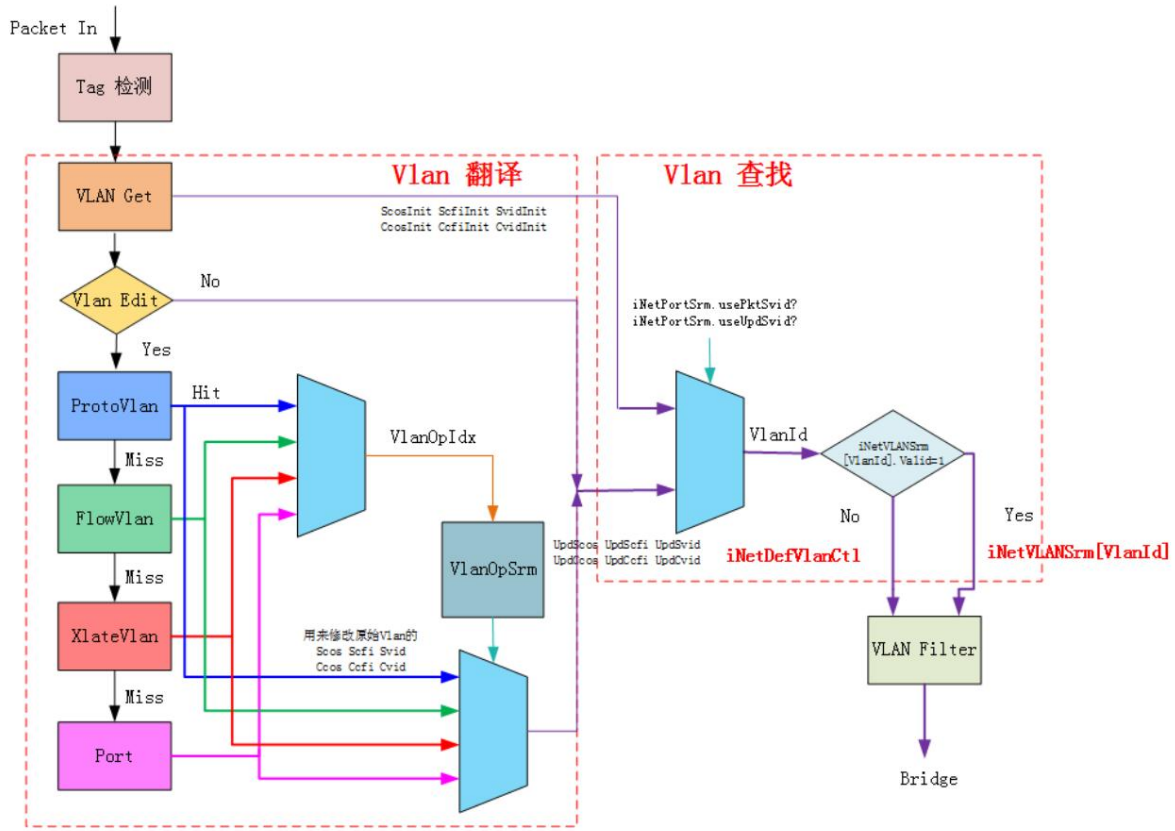


Figure 2-1 Schematic diagram of ingress VLAN function

According to the Vlan Id, the corresponding iNetVlanSrm[VlanId] table is searched. When the iNetVlanSrm[vlanId] table entry is invalid, iNetDefVlanCtl is used to obtain the Vlan broadcast domain and other information. Therefore, under the default chip configuration (Vlan filtering is not set), the iNetVlanSrm table entry is invalid. Using iNetDefVlanCtl.portbmp for flooding can forward packets normally. When iVtPortSrm.VlanFilterEn=1, the inlet Vlan filtering function is enabled. If the packet input port is not in the Vlan broadcast domain, the packet is discarded. For detailed implementation methods, please refer to each subsection.

2.2.1 **Tag** Detection

The Tag detection diagram is shown in Figure 2-2. When a message comes in, its 12th and 13th bytes are checked first. If they meet equation 1, it means that it carries stag, that is, it carries an outer VLAN. Then, the 16th and 17th bytes of the message are checked. If they meet equation 2, it means that the message carries both inner and outer VLANs, that is, DT. If it does not meet equation 2, it only carries one outer VLAN, that is, SOT. If the 12th and 13th bytes of the message do not meet equation 1 but meet equation 2, it only carries one inner VLAN, that is, SIT. If the 12th and 13th bytes of the message do not meet either equation 1 or equation 2, the message does not carry a VLAN, that is, UT.

Pcket.stagTpid = Ipr0StagTpidCtl.stagTpidx (x=1, 2, 3, 4, Stpid0 default value is 0x88A8, Ipr0StagBmpCtl.StagBmp[InPort*4+3:InPort*4] (corresponding bit enable)

Packet.ctagTpid = Ipr0CtagTpidCtl.ctagTpid (the default value of Stpid0 is 0x8100)

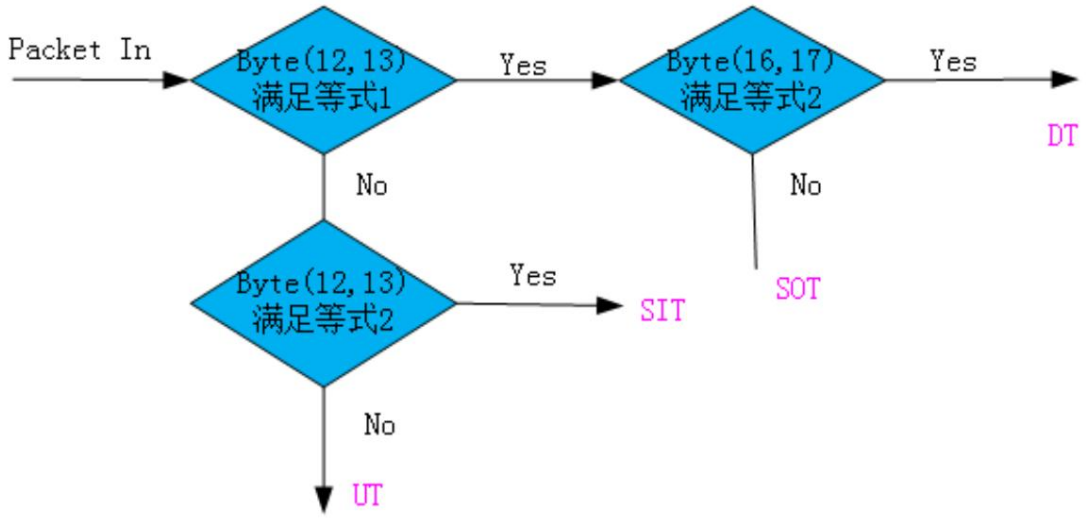


Figure 2-2 Tag detection diagram

Tag detection related registers and table entries

iPr0StagBmpCtl

Name	Width	Bits	R/W	Def	Description
stagBmp	140	139:0	R/W	0x1111111111111111 1111111111111111111	Each port's stag bmp, each port 4 bits

iPr0StagTpidCtl

Name	Width	Bits	R/W	Def	Description
stagTpid3	16	63:48	R/W	0x0	stagTpid3
stagTpid2	16	47:32	R/W	0x0	stagTpid2
stagTpid1	16	31:16	R/W	0x0	stagTpid1
stagTpid0	16	15:0	R/W	0x88a8	stagTpid0

iPr0CtagTpidCtl

Name	Width	Bits	R/W	Def	Description
ctagTpid	16	15:0	R/W	0x8100	ctagTpid

2.2.2 Vlan Translation

Vlan translation includes Vlan Get, Proto Vlan, FLowVlan, XlateVlan and Vlan Edit.
Its functions are introduced.

2.2.2.1 Vlan Get

After obtaining the VLAN type of the message through Tag detection, the VLAN information of the message can be obtained. The VLAN information comes from the original message or the
The default VLAN information of the corresponding port is shown in the following table.

Vlan Type	stag	ctag

DT	svidinit: packet.stag.svid scosinit: packet.stag.scos scfiinit: packet.stag.scfi svidinit:	cvidinit: packet.ctag.cvid ccosinit: packet.ctag.ccos ccfiinit: packet.ctag.ccfi cvidinit:
SOT	packet.stag.svid scosinit: packet.stag.scos scfiinit: packet.stag.scfi svidinit:	iVtPortSrm.defcvid ccosinitcos: iVtPortSrm.defcos ccfiinit: iVtPortSrm.defcfi cvidinit:
SIT	iVtPortSrm.defsvid scosinit: iVtPortSrm.defcos scfiinit: iVtPortSrm.defcfi svidinit:	packet.ctag.cvid ccosinit: packet.ctag.ccos ccfiinit: packet.ctag.ccfi cvidinit:
UT	iVtPortSrm.defsvid scosinit: iVtPortSrm.defcos scfiinit: iVtPortSrm.defcfi	iVtPortSrm.defcvid ccosinit: iVtPortSrm.defcos ccfiinit: iVtPortSrm.defcfi

2.2.2.2 ProtoVlan

As shown in Figure 2-3, the protocol Vlan search uses the traversal method. Under the condition that iVtPortSrm.protoVlanEn=1, the hit conditions are as follows.

inLport=iVtProtoVlanCtl[i].inLport

inIsLag=iVtProtoVlanCtl[i].inIsLag (Equation 1)

ethType=iVtProtoVlanCtl[i].ethType (i can be between 0-15)

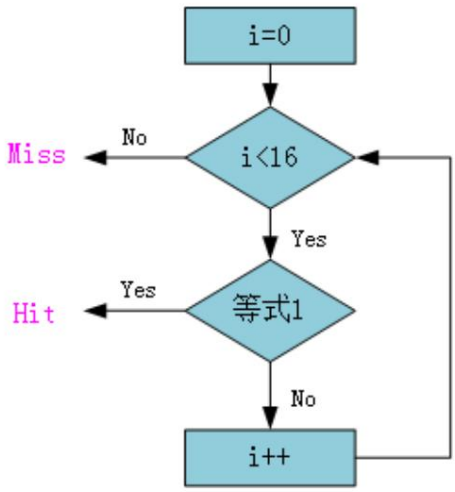


Figure 2-3 Protocol Vlan traversal method

Obtain the iVtProtoVlanSrm entry index when hitting by traversing, and then query the corresponding entry to obtain the replacement or addition svid, scos, scfi, cvid, ccos, cscfi, and vlanOpIdx values are used for VLan editing.

2.2.2.3 Flow Vlan

There are four methods for searching Flow Vlan, namely vlanKey, macKey, ipv4Key, and ipv6Key.
When iVtPortSrm.flowVlan0En=1, the corresponding
The index number of the iVtFlowVlanTcmSrm to get the replaced or added svid, scos, scfi, cvid, ccos, cscfi values and VlanOpIdx. The specific method used depends on the port configuration and the type of packet, as shown in Figure 2-4.

For specific table lookup strategies and search processes, please refer to Chapter 4 ACL Principles.

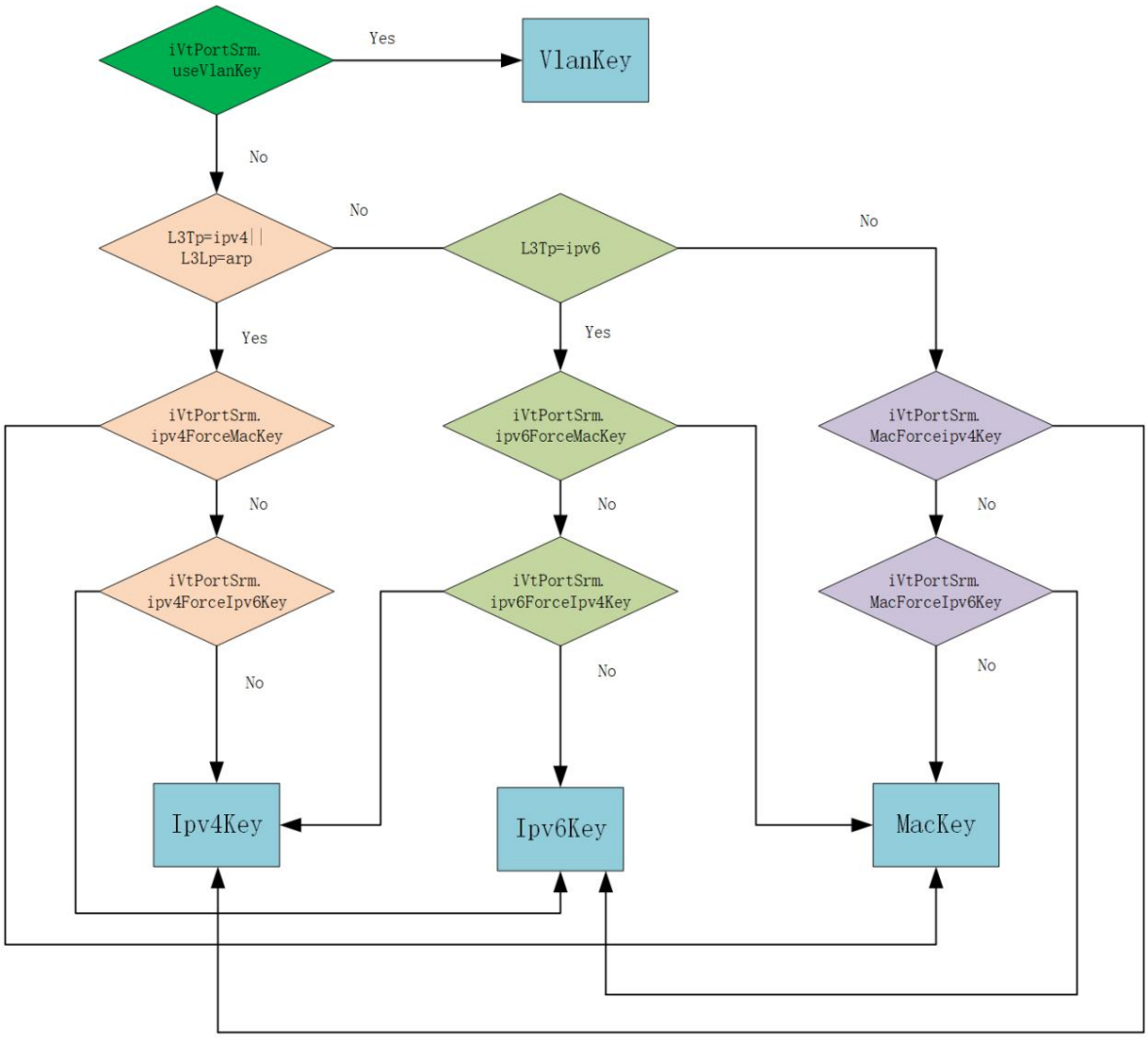


Figure 2-4 Flow Vlan port configuration

vlanKey consists of the following fields:

Name	Width	Bits
2'b11	2	139:138
meta.sip[17:0]	18	137:120
phv.eth.sa	48	119:72
meta.ethType	16	71:56

meta.l2Tp	2	55:54
meta.l3Tp	4	53:50
meta.l4Tp	3	49:47
meta.l5Tp	3	46:44
meta.oamTp	3	43:41
phv.stag.{vid,cos,cfi}	16	40:25
phv.stag.isvalid	1	24:24
phv.ctag.{vid,cos,cfi}	16	23:8
phv.ctag.isvalid	1	7:7
meta.inIsLag	1	6:6
meta.inLport	6	5:0

MacKey consists of the following fields:

Name	Width	Bits
2'b00	2	279:278
res0	36	277:242
phv.ctag.isvalid	1	241:241
meta.l4Tp	3	240:238
meta.ethType	16	237:222
meta.l5Tp	3	221:219
meta.oamTp	3	218:216
phv.stag.{vid,cos,cfi}	16	215:200
phv.ctag.{vid,cos,cfi}	16	199:184
phv.l2_slow.subType	8	183:176
meta.ptpMsgType	4	175:172
meta.l3Udf0	8	171:164
meta.l3Udf1	8	163:156
meta.l4Udf0	8	155:148
meta.l4Udf1	8	147:140
2'b00	2	139:138
phv.stag.isvalid	1	137:137
meta.l3Tp	4	136:133
meta.l2Tp	2	132:131
phv.eth.da	48	130:83
s	48	82:35
portBmp	35	34:0

iPv4Key consists of the following fields:

Name	Width	Bits
2'b01	2	279:278
res2	7	277:271
meta.ipOption	1	270:270
meta.ipHdrErr	1	269:269

meta.ipLen	16	268:253
phv.l3.ipv4.id	16	252:237
phv.l4.tcp.flag	8	236:229
meta.l4SrcPort	16	228:213
meta.l4DstPort	16	212:197
meta.l5Tp	3	196:194
meta.oamTp	3	193:191
meta.ttl	8	190:183
phv.ipv4.ihl	4	182:179
phv.ipv4.flg	3	178:176
meta.ptpMsgType	4	175:172
meta.l3Udf0	8	171:164
meta.l3Udf1	8	163:156
meta.l4Udf0	8	155:148
meta.l4Udf1	8	147:140
2'b01	2	139:138
meta.icmpType	8	137:130
meta.icmpCode	8	129:122
meta.tos	8	121:114
meta.l3Tp	4	113:110
meta.l4Tp	3	109:107
meta.ipProto	8	106:99
meta.sip	32	98:67
meta.dip	32	66:35
portBmp	35	34:0

iPv6Key consists of the following fields:

Name	Width	Bits	R/W	Def
2'b10	2	559:558	R/W	0x0
meta.ptpMsgType	4	557:554	R/W	0x0
phv.eth.da	48	553:506	R/W	0x0
phv.l4.tcp.flag	8	505:498	R/W	0x0
meta.ipLen	16	497:482	R/W	0x0
phv.l3.ipv4.id	16	481:466	R/W	0x0
meta.l5Tp	3	465:463	R/W	0x0
meta.oamTp	3	462:460	R/W	0x0
meta.l4SrcPort	16	459:444	R/W	0x0
meta.l4DstPort	16	443:428	R/W	0x0
meta.ipv6ExtHdrTp 2'b10	8	427:420	R/W	0x0
	2	419:418	R/W	0x0
phv.stag.{vid,cos,cfi}	16	427:412	R/W	0x0
phv.ctag.{vid,cos,cfi}	16	401:386	R/W	0x0
phv.ctag.isvalid	1	385:385	R/W	0x0
phv.stag.isvalid	1	384:384	R/W	0x0

meta.icmpType	8	383:376	R/W	0x0
meta.icmpCode	8	375:368	R/W	0x0
meta.l3tp	4	367:364	R/W	0x0
meta.tos	8	363:356	R/W	0x0
meta.ttl	8	355:348	R/W	0x0
phv.ipv6.ext.isValid	1	553:553	R/W	0x0
meta.l3Udf0	8	346:339	R/W	0x0
meta.l3Udf1	8	338:331	R/W	0x0
meta.l4Udf0	8	330:323	R/W	0x0
meta.l4Udf1	8	322:315	R/W	0x0
portBmp	35	314:280	R/W	0x0
2'b10	2	279:278	R/W	0x0
meta.ipOption	1	277:277	R/W	0x0
meta.ipHdrErr	1	276:276	R/W	0x0
meta.ipProto	8	275:268	R/W	0x0
meta.sip	128	267:140	R/W	0x0
2'b10	2	139:138	R/W	0x0
meta.l4Tp	3	137:135	R/W	0x0
meta.dip	128	134:7	R/W	0x0
meta.inIsLag	1	6:6	R/W	0x0
meta.inLport	6	5:0	R/W	0x0

The depth of iVtFlowVlanTcm is 128, and the fields included are shown in the following table. From the width of the key table above, we can see that vlanKey only needs Occupies 1 table entry, macKey and ipv4Key occupy 2 table entries, and ipv6Key occupies 4 table entries.

iVtFlowVlanTcm entry

Name	Width	Bits	R/W	Def	Description
validMask	1	281:281	R/W	0x0	Valid indication mask
keyMask	140	280:141	R/W	0x0	Find key mask
valid	1	140:140	R/W	0x0	Effective instructions
key	140	139:0	R/W	0x0	Find key

Therefore, iVtFlowVlanTcmSrm supports up to 128 vlanKey or 64 macKey or 64 iPhv4Key or If the mixed key method is used, it is determined according to the actual allocation situation, and the configuration register iVtFlowVlanCtl accomplish.

iVtFlowVlanCtl Register

Name	Width	Bits	R/W	Def		Description
ipv6KeyRstCtl	16	63:48	R/W	0x0		Same as macKeyRstCtl
ipv4KeyRstCtl	16	47:32	R/W	0x0		Same as macKeyRstCtl
vlanKeyRstCtl	16	31:16	R/W	0x0		{indexBase[6:0],keySize[1:0],tableBase[6:0]}
macKeyRstCtl	16	15:0	R/W	0x0		{indexBase[6:0],keySize[1:0],tableBase[6:0]}

The index counting method of iVtFlowVlanTcmSrm is as follows:

flowVlanIndex = (Index-indexBase)>>keySize +tableBase

Index is the index value calculated by hashKey.

for example:

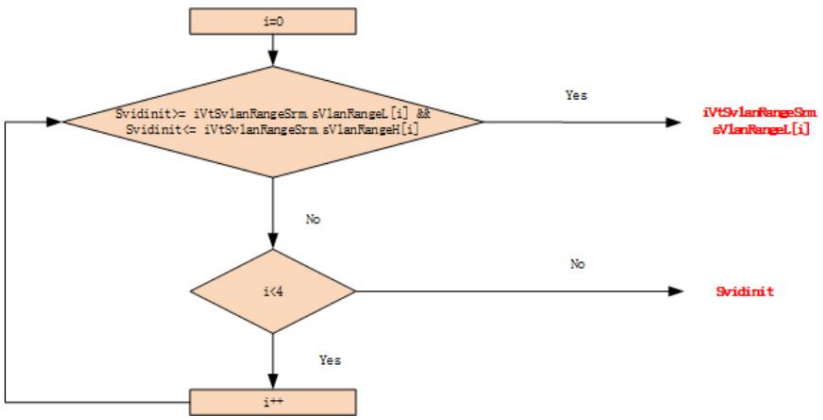
Assuming that two types of Tcm tables, iPv4 and ipv6, are configured, 32 ipv4 and 16 ipv6Key tables can be configured. Therefore, ipv4KeyRstCtl.indexBase=0, ipv4KeyRstCtl.keySize=1, ipv4KeyRstCtl.tableBase=0 can be configured; ipv6KeyRstCtl.indexBase=64, ipv6KeyRstCtl.keySize=2, ipv6KeyRstCtl.tableBase=32 can be configured. Then the ipv4Key index value is between 0-31; the ipv6Key index value is between 32-47.

2.2.2.4 XlateVlan

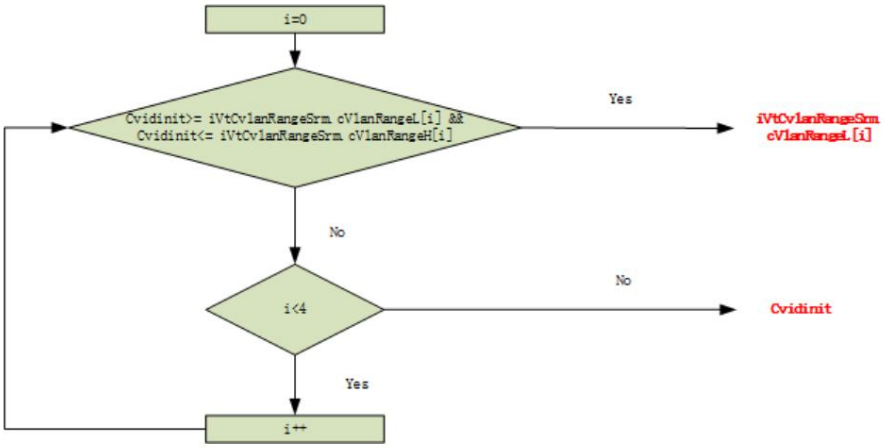
XlateVlan consists of two parts, Xlate0 and Xlate1, each of which contains 8 hash tables. Xlate0 contains iVtXlateLeftSrm[x] (x is between 0-3) and iVtXlateRightSrm[y] (y is between 0-3) (8 hash tables are to prevent hash conflicts). Before generating XlateVlanKey, you can set the VlanRange function through iVtPortSrm.svlanRangeEn and iVtPortSrm.cvlanRangeEn, and set the CosMap function through iVtPortSrm.scosMapEn and iVtPortSrm.ccosMapEn. Through the generated XlateKey and the set hash algorithm (set by iVtPortSrm.AlgTp), the index of the corresponding XlateSrm table can be calculated, so as to obtain the svid, scos, scfi, cvid, ccos, cscfi values and VlanOpIdx used for replacement or addition, which are used for Vlan editing.

1. VlanRange

VlanRange converts a certain range of VLANs to the same VLAN, including SvidRange and CvidRange operations. SvidRange is implemented by setting the iVtSvlanRangeSrm table, and the table entry depth is 16. The specific table entry to be used is obtained through iVtPortSrm.svlanRangeIdx, and then the SvidRange value is obtained through the following figure.



Similarly, get the corresponding iVtSvlanRangeSrm table through iVtPortSrm.cvlanRangeIdx and use the following figure to get the CvidRange value.



2. CosMap

CosMap maps the priority of the message to modify the priority of the message. It consists of two parts: ScosMap and CcosMap. This is achieved by setting iVtScosMapSrm and iVtCcosMapSrm respectively, with a table entry depth of 8. iVtPortSrm.scosMapIdx and iVtPortSrm.ccosMapIdx corresponds to the index value of the Cos mapping table, according to ScosInit, ScfiInit, The CcosInit and CcfiInit values are used for corresponding mapping, as shown in Table 2-1, Table 2-2, Table 2-3 and Table 2-4.

Table 2-1 ScosInit mapping relationship

ScosInit 0	**ScosMap**
1	iVtScosMapSrm[iVtPortSrm.scosMapIdx].cos0
	iVtScosMapSrm[iVtPortSrm.scosMapIdx].cos1
2	iVtScosMapSrm[iVtPortSrm.scosMapIdx].cos2
3	iVtScosMapSrm[iVtPortSrm.scosMapIdx].cos3
4	iVtScosMapSrm[iVtPortSrm.scosMapIdx].cos4
5	iVtScosMapSrm[iVtPortSrm.scosMapIdx].cos5
6	iVtScosMapSrm[iVtPortSrm.scosMapIdx].cos6
7	iVtScosMapSrm[iVtPortSrm.scosMapIdx].cos7

Table 2-2 ScfiInit mapping relationship

ScfiInit	**SfiMap**
0	iVtScosMapSrm[iVtPortSrm.scosMapIdx].cfi0
1	iVtScosMapSrm[iVtPortSrm.scosMapIdx].cfi1

Table 2-3 CcosInit mapping relationship

CcosInit 0	**CcosMap**
1	iVtCcosMapSrm[iVtPortSrm.ccosMapIdx].cos0
	iVtCcosMapSrm[iVtPortSrm.ccosMapIdx].cos1
2	iVtCcosMapSrm[iVtPortSrm.ccosMapIdx].cos2
3	iVtCcosMapSrm[iVtPortSrm.ccosMapIdx].cos3
4	iVtCcosMapSrm[iVtPortSrm.ccosMapIdx].cos4
5	iVtCcosMapSrm[iVtPortSrm.ccosMapIdx].cos5
6	iVtCcosMapSrm[iVtPortSrm.ccosMapIdx].cos6
7	iVtCcosMapSrm[iVtPortSrm.ccosMapIdx].cos7

Table 2-4 CcfiInit mapping relationship

CcfiInit	CcfiMap
0	iVtCcosMapSrm[iVtPortSrm.ccosMapIdx].cfi0
1	iVtCcosMapSrm[iVtPortSrm.ccosMapIdx].cfi1

3. XlateKey

The index value of each hash table is calculated from the corresponding XlateKey table. There are three types of XlateKey tables: VlanKey, MacKey and MacIPBandKey are introduced below respectively.

When the xlateKey table is VlanKey, the hash table is as follows:

Name	Width	Bits	R/W	Def	Description
keyTp	2	60:59	R/W	0x0	Lookup type for xlate0: 0: vlan xlate; 1: Vlan Mac; 2: macip bind
keyMode	3	58:56	R/W	0x0	key component type
res	16	55:40	R/W	0x0	Reserve
inIsLag	1	39:39	R/W	0x0	Indicates whether the input internal logic port number is a LAG port, high effective
inLport	6	38:33	R/W	0x0	The internal logical port number of the input. When inIsLag is valid, it indicates LAG port, otherwise it indicates a normal port.
ccos	3	32:30	R/W	0x0	ccos value
ccfi	1	29:29	R/W	0x0	ccfi value
cvid	12	28:17	R/W	0x0	cvid value
scos	3	16:14	R/W	0x0	scos value
scfi	1	13:13	R/W	0x0	scfi value
svid	12	12:1	R/W	0x0	svid value
valid	1	0:0	R/W	0x0	Effective instructions

When the xlateKey table is MacKey, the hash table is as follows:

Name	Width	Bits	R/W	Def	Description
keyTp	2	60:59	R/W	0x0	Lookup type for xlate0: 0: vlan xlate; 1: Vlan Mac; 2: macip bind
keyMode	3	58:56 R/W 0x0 55:55 R/W 0x0			key component type
inIsLag	1	Indicates whether the input internal logic port number is a LAG port, high effective			
inLport	6	54:49	R/W	0x0	The internal logical port number entered. When inIsLag is valid, it indicates that the LAG port Otherwise, it indicates a common port.
smac	48	48:1	R/W	0x0	Source mac address
valid	1	0:0	R/W	0x0	Effective instructions

When the xlateKey table is MacIPBandKey, the hash table is as follows:

Name	Width	Bits	R/W	Def	Description
keyTp	2	60:59	R/W	0x0	Lookup type for xlate0: 0: vlan xlate; 1: Vlan Mac; 2: macip bind

ipv4Ind	1	58:58	R/W	0x0	0: SIP for IPv6; 1: SIP for IPv4
sip	57	57:1	R/W	0x0	When using ipv4, the lower 32 bits are sip; when using ipv6, the lower bits of SIP
valid	1	0:0	R/W	0x0	Effective instructions

The domain information contained in the XlateKey table is obtained by configuring iVtXlateKeyCtlx (x is 0-7). 8 different configurations can be configured. The Key table generation mode is selected through XlateKeyMode in the iVtPortSrm table. The configuration information of iVtXlateKeyCtlx is as follows:

Down:

Name	Width	Bits	R/W	Def		Description
usePort1	1	17:17	R/W	0x0		xlate1 Checks whether to use the port as the key. 1: Use
useCrange1	1	16:16	R/W	0x0	xlate1	Whether to use the value processed by cvid range when searching. 1: Use
useSrange1	1	15:15	R/W	0x0	xlate1	Whether to use the value processed by svid range when searching. 1: Use
useSvid1	1	14:14	R/W	0x0		xlate1 checks whether svid is used as the key. 1: Use
useScos1	1	13:13	R/W	0x0		xlate1 checks whether scos is used as the key. 1: Use
useScfi1	1	12:12	R/W	0x0		xlate1 checks whether scfi is used as the key. 1: Use
useCvid1	1	11:11	R/W	0x0		xlate1 checks whether cvid is used as the key. 1: Use
useCcos1	1	10:10	R/W	0x0		xlate1 checks whether ccos is used as the key. 1: Use
useCcfi1	1	9:9	R/W	0x0		xlate1 checks whether ccfi is used as the key. 1: Use
usePort0	1	8:8	R/W	0x0		xlate0 checks whether to use the port as the key. 1: Use
useCrange0	1	7:7	R/W	0x0	xlate0	Whether to use the value processed by cvid range when searching. 1: Use
useSrange0	1	6:6	R/W	0x0	xlate0	Whether to use the value processed by svid range when searching. 1: Use
useSvid0	1	5:5	R/W	0x0		xlate0 checks whether svid is used as the key. 1: Use
useScos0	1	4:4	R/W	0x0		xlate0 checks whether scos is used as the key. 1: Use
useScfi0	1	3:3	R/W	0x0		xlate0 checks whether scfi is used as the key. 1: Use
useCvid0	1	2:2	R/W	0x0		xlate0 checks whether cvid is used as the key. 1: Use
useCcos0	1	1:1	R/W	0x0		xlate0 checks whether ccos is used as the key. 1: Use
useCcfi0	1	0:0	R/W	0x0		xlate0 checks whether ccfi is used as the key. 1: Use

The generation of XlateKey is shown in Figure 2-5. The corresponding Xlate table is enabled through iVtPortSrm.XlateEn. iVtPortSrm.XlateTp determines which XlateKey to use, and iVtPortSrm.XlateKeyMode is used to obtain the table entry of iVtXlateKeyCtl. After obtaining XlateKey, the index value of the XlateSrm table is calculated through the hash algorithm. The VLAN information and VlanOpIdx used to add or delete are obtained by looking up the table, so as to edit the VLAN.

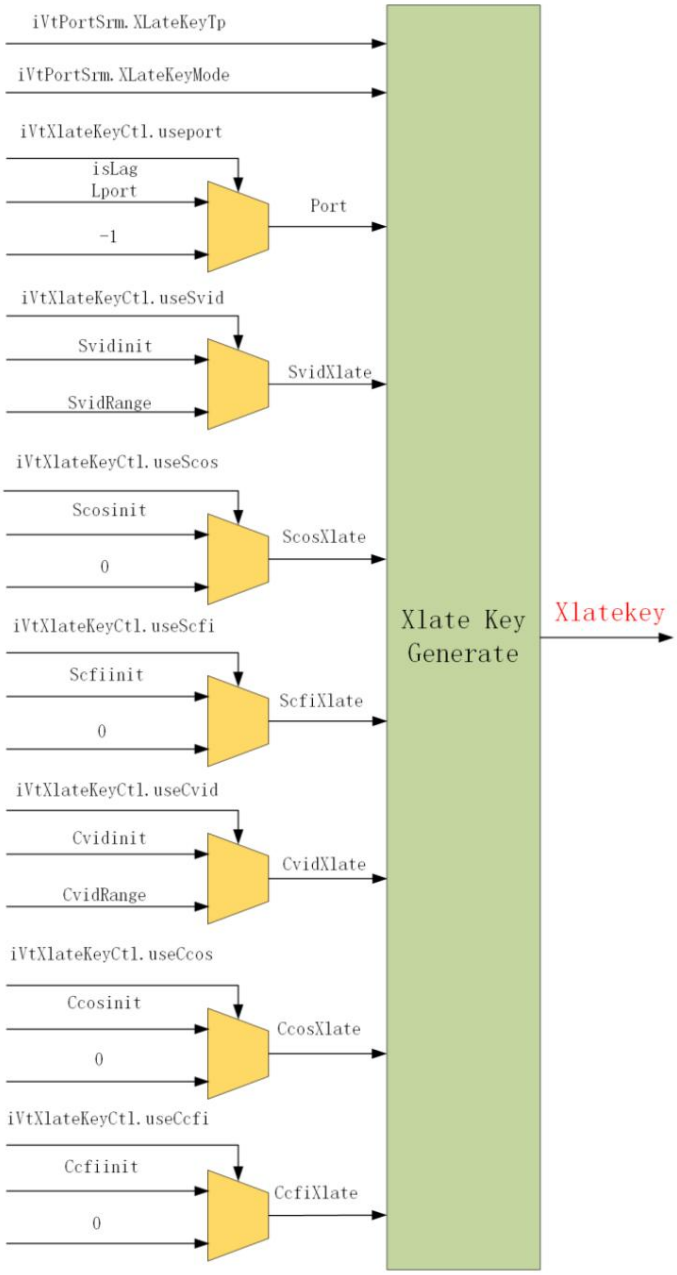


Figure 2-5 XlateKey generation diagram

2.2.2.5 Port

When ProtoVlan, FlowVlan, and XlateVlan are not hit, the port default svid, scos, scfi, cvid, ccos, cscfi values and VlanOpIdx for VLan editing, query table entry iVTPortSrm by port index Obtain.

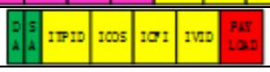
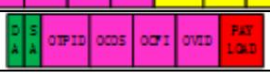
2.2.3 Vlan Edit

When the iVtEditEn bit in the iVtPortSrm table is 0, the VLAN information is obtained by Vlan Get; when it is 1, the VLAN information is obtained by VlanOpIdx The index is used to find the iVtVlanOpSrm table entry to determine the corresponding editing operation. The VLAN information to be added or replaced (i.e. Newsvid, Newscos Newcvid, Newccos, Newccfi) and VlanOpIdx from iVtProtoVlanSrm, iVtVlanTcmSrm, iVtXlateLeftSrm[x](x can be 0-3), iVtXlateRightSrm[y](y can be 0-3), iVtPortSrm

For the specific implementation, please refer to the corresponding chapter. The following is a detailed introduction to the Vlan editing operation in iVtVlanOpSrm.

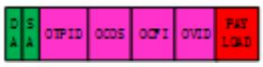
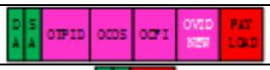
2.2.3.1 DT Edit

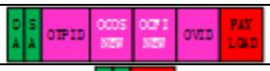
The Vlan information after DT message editing is shown in the following table, where defcvid and difsvid come from two fields of iVtPortSrm.

Vlan edited before the message	**iVtVlanOpSrm** field value	0 (nop)	**Vlan** edited message
	dtPovid (packet.stag.vid=0) dtOvid (packet.stag.vid != 0)		
		1(copy)	
		2(replace)	
		3(delete)	
	dtOpri	0 (nop)	
		1(copy)	
		2(replace)	
		3(delete)	
	dtPIvid (packet.ctag.vid = 0) dtIvid (packet.ctag.vid != 0)	0 (nop)	
		1(copy)	
		2(replace)	
		3(delete)	
	dtI	0 (nop)	
		1(copy)	
		2(replace)	
		3(delete)	

2.2.3.2 SOT Edit

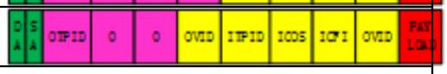
The Vlan information after SOT message editing is shown in the table, where defcvid and difsvid come from two fields of iVtPortSrm.

Vlan edited before the message	**iVtVlanOpSrm** field value	0 (nop)	**Vlan** edited message
	SotPovid (packet.stag.vid=0) SotOvid (packet.stag.vid != 0)		
		2(replace)	
		3(delete)	

	SotOpri	0 (nop)	The message remains unchanged
		2(replace)	
		3(delete)	
	SotIvid	0 (nop)	The message remains unchanged
		1(add)	
		2 (copy)	
	SotIpri	0 (nop)	The message remains unchanged
		1(add)	
		2 (copy)	

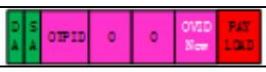
2.2.3.3 SIT Edit

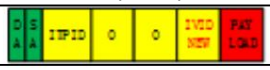
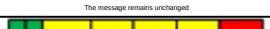
The Vlan information after SIT message editing is shown in the table, where defcvid and difsvid come from two fields of iVtPortSrm.

Vlan edited before the message	**iVtVlanOpSrm** field value 0 (nop)		**Vlan** edited message
	SitOvid		The message remains unchanged
		1 (add)	
		2 (copy)	
	SitOpri	0 (nop)	The message remains unchanged
		1(add)	
		2 (copy)	
	SitPivid (packet.stag.vid=0) SitIvid (packet.stag.vid != 0)	0 (nop)	The message remains unchanged
		2(replace)	
		3(delete)	
	SotIpri	0 (nop)	The message remains unchanged
		2(replace)	
		3(delete)	

2.2.3.4 UT Edit

The Vlan information after UT message editing is shown in the table, where defcvid and difsvid come from two fields of iVtPortSrm.

Vlan edited before the message	**iVtVlanOp** **Srm** domain	Domain Value	**Vlan** edited message
	UtOvid	0 (nop)	The message remains unchanged
		1 (add)	
	UtOpri	0 (nop)	The message remains unchanged

		1(add)	
	UtIvid	0 (nop)	
		1 (add)	
	UiP	0 (nop)	
		1(add)	

2.2.4 Vlan Lookup

After VLAN translation, two different VLAN values, SvidInit and updSvid, are obtained. The selection is made according to iNetPortSrm.usePktSvid and iNetPortSrm.useUpdSvid. The VLAN value is used to query iNetVlanSrm to obtain some configuration information of the VLAN, as shown in Figure 2-6.

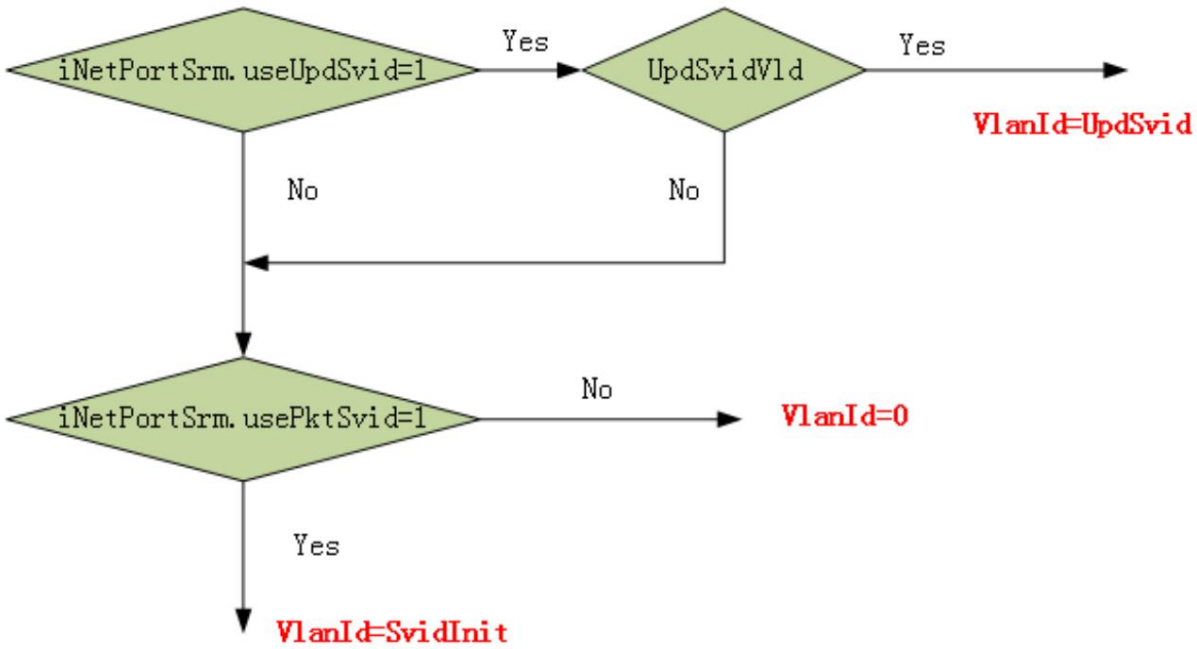


Figure 2-6 Vlan search function flow chart

Through the Vlan number, you can query the corresponding iNetVLANSrm table to obtain various attribute values of Vlan, such as setting Stp, Erps Function: Prevent loops, configure Pbmp to limit broadcast domains, and use qosProfileVld and qosProfileIdx to manage network quality.

When iNetVLANSrm.valid=0, it means that the corresponding VLAN attribute table is not configured. In order to forward the message normally, the default Vlan attribute table, iNetDefVlanCtl.

2.2.5 Vlan Filter

When iNetPortSrm.vlanFilterEn=1, the ingress VLAN filtering function is enabled, as shown in Figure 2-7. When isLag is set to 1, it means that the port is a Lag port. When the Lport bit in iNetPortSrm.lagBmp is 0, the message of the port is discarded; when isLag is set to 0, it means that the port is a normal port. When the Lport bit in iNetPortSrm.PortBmp is 0, the message of the port is discarded.

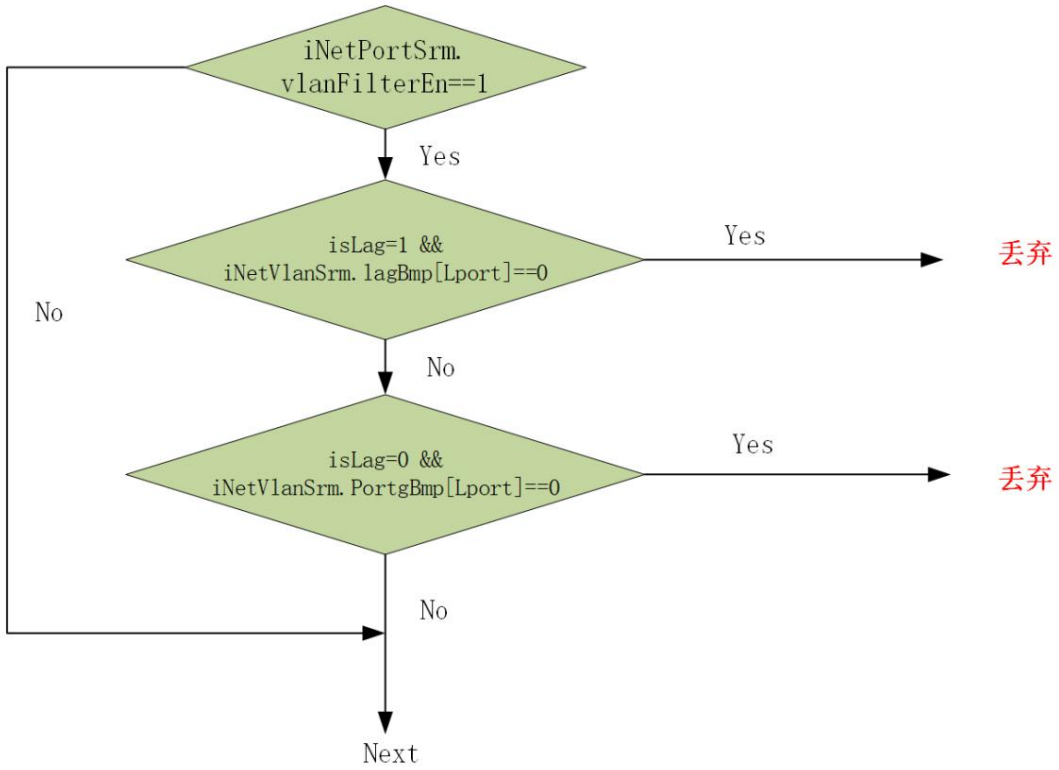


Figure 2-7 Vlan filtering function flow chart

2.3 Egress **VLAN**

The egress Vlan function mainly includes EVlan filtering, EVlan Get, and EVlan editing, as shown in Figure 2-8. After obtaining the internal Vlan ID through the ingress Vlan operation, the message first searches for eEeVLanSrm through the internal VlanId. When eEeVlanSrm.valid=0, the default Vlan attribute table eEeDefVlanCtl is used.

When eEePortSrm.vlanFilterEn=1, EVlan filtering is enabled, and packets that are not in the Evlan broadcast domain will be filtered out. Then, Evlan information is obtained based on whether the packet carries Tag information. When the packet lacks EVlan information, the default Vlan information in eEeportSrm is used. Finally, eEePortSrm.eVtEditEn is used to determine whether to perform EVlan editing. Evlan editing has two modes: EXlateVlan and Eport. VlanOpIdx and the vlan information (svid, scos, scfi, cvid, ccos, ccfi) used to add or replace are obtained by searching the corresponding table entries. The original packet is edited by querying iVtVlanOpSrm through the index VlanOpIdx. The detailed functions of each module are shown in the following chapters.

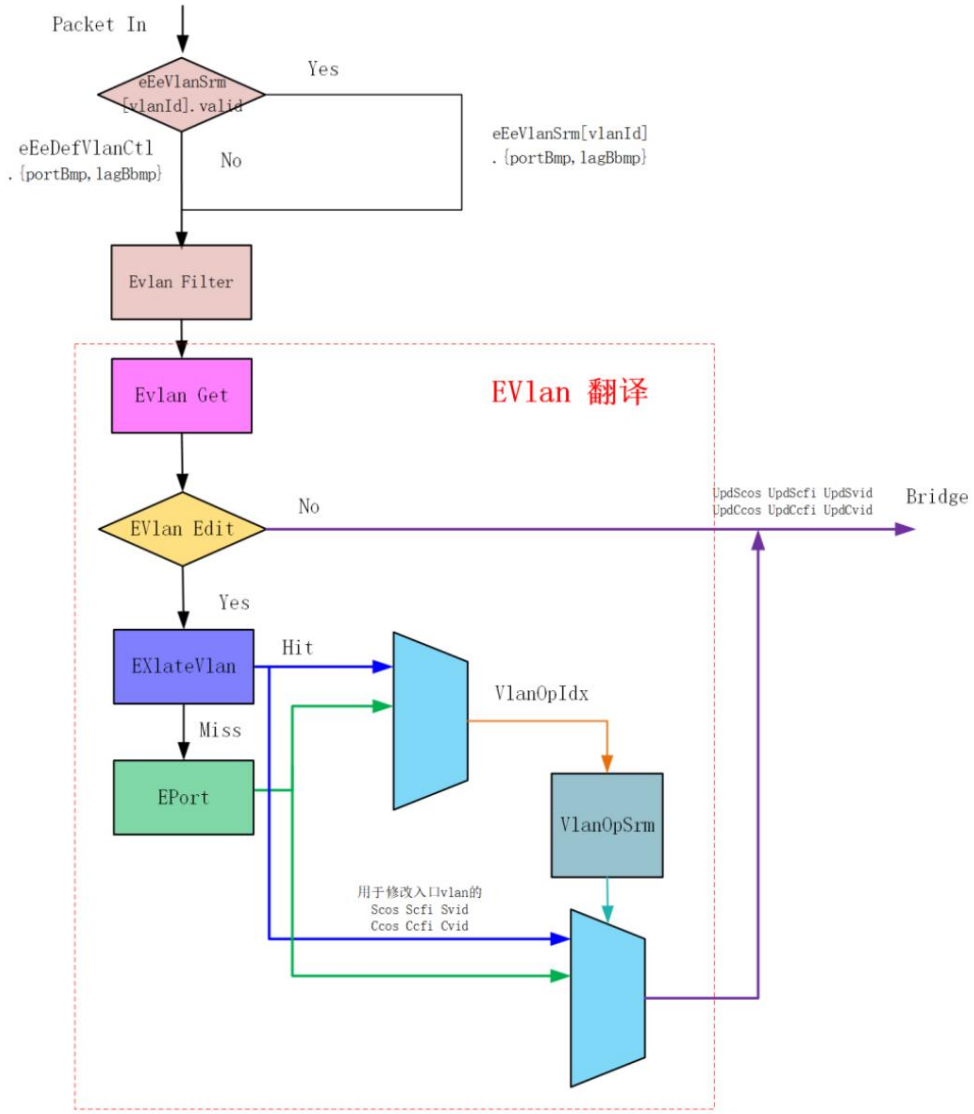


Figure 2-8 Egress VLAN function diagram

2.3.1 EVlan Filtering

When eEePortSrm.vlanFilterEn=1, the ingress VLAN filtering function is enabled, as shown in Figure 2-9.

Indicates that the port is a Lag port. When the Lport bit in eEePortSrm.lagBmp is 0, the message of the port is discarded.

When set to 0, it means that the port is a normal port. When the Lport bit in eEePortSrm.PortBmp is 0, the message of the port is discarded.

abandoned.

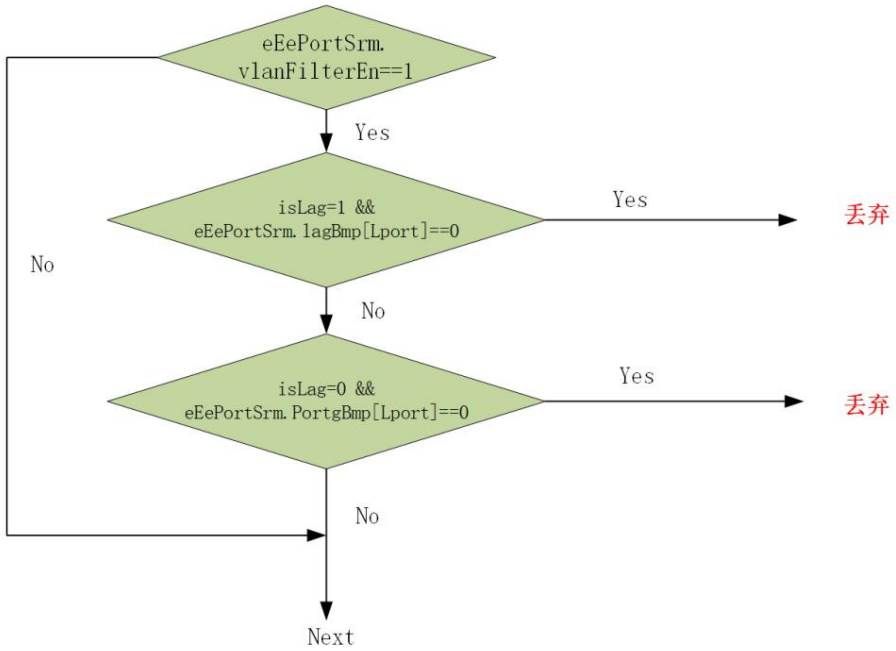


Figure 2-9 EVlan filter function flow chart

2.3.2 EVlan Translation

2.3.2.1 Evlan Get

After the message passes through the ingress VLAN related operations, Upd.{Svid, Scos, Scfi, Cvid, Ccos, Ccfi} is obtained. If
If Stag or Ctag is used, the default VLAN information of the corresponding egress is used, which is obtained by looking up the table eEePortSrm.

2.3.2.2 EXlateVlan

EXlateVlan is similar to XlateVlan, and also consists of two parts, EXlate0 and EXlate1. Each part contains 8 hash tables, namely eEeXlateLeftSrm[x](x is between 0-3), eEeXlateRightSrm[y](y is between 0-3) (8 hash tables are to prevent hashing
However, the VlanRange and CosMap functions are no longer included, and the Key table is different, with only Vlan-based Keys.
In addition, the EXlateTcm table function is added.

For EXlate0 and EXlate1, the EXlateKey and the set hash algorithm (eEePortSrm.AlgTp) can be calculated.
The index corresponding to the EXlateSrm table to obtain the svid, scos, scfi, cvid, ccos, cscfi values used for replacement or addition, and VlanOpIdx. For the EXlateTcm table, you can perform a mask search using ExlateKey and EXlateTcm to get
EXlateTcmsrm table index value to obtain the svid, scos, scfi, cvid, ccos, cscfi values used for replacement or addition, and VlanOpIdx.

1. EXlate

The Key table corresponding to EXlate0 and EXlate1 is as follows:

Name	Width	Bits	R/W		Def	Description
oeLh	2	41:40	R/W		0x0	evt key type
outIsLag	1	39:39	R/W		0x0	Indicates whether the output internal logical port number is a LAG port. Highly effective

outLport	6	38:33 R/W	0x0	The internal logical port number of the output. When outIsLag is valid, it indicates indicates a LAG port, otherwise it indicates a common port.
ccos	3	32:30 R/W	0x0	ccos
ccfi	1	29:29 R/W	0x0	ccfi
cvid	12	28:17 R/W	0x0	cvid
scos	3	16:14 R/W	0x0	scos
scfi	1	13:13 R/W	0x0	scfi
svid	12	12:1 R/W	0x0	svid
valid	1	0:0 R/W	0x0	Effective instructions

The domain information contained in the EXlateKey table is obtained by configuring eEeXlateKeyCtlx (x is 0-3). There are 4 different configurations. The different Key table generation modes are selected through eVtTp in the EXlateKey table. The configuration information of eEeXlateKeyCtlx is as follows:

eEeXlateKeyCtlx

eVtEn0	1	16:16 R/W 0x0		eVt0 search enable, high effective
eVtEn1	1	15:15 R/W 0x0		eVt1 Search enable, high effective
eVtTcamEn	1	14:14 R/W 0x0		eVt tcam search enable, high effective
xlate0UsePort 1		13:13 R/W 0x0		xlate0 whether to use port, high effective
xlate1UsePort 1		12:12 R/W 0x0		xlate1 Whether to use port, high effective
xlate0UseSvid 1		11:11 R/W 0x0		xlate0 whether to use svid, high effective
xlate1UseSvid 1		10:10 R/W 0x0		xlate1: whether to use svid, high effective
xlate0UseScos 1		9:9 R/W 0x0		xlate0 whether to use scos, high effective
xlate1UseScos 1		8:8 R/W 0x0		xlate1 whether to use scos, high effective
xlate0UseScfi	1	7:7 R/W 0x0		xlate0 whether to use scfi, high effective
xlate1UseScfi	1	6:6 R/W 0x0		xlate1 Whether to use scfi, high effective
xlate0UseCvid 1		5:5 R/W 0x0		xlate0 whether to use cvid, high effective
xlate1UseCvid 1		4:4 R/W 0x0		xlate1 whether to use cvid, high effective
xlate0UseCcos 1		3:3 R/W 0x0		xlate0 whether to use ccos, high effective
xlate1UseCcos 1		2:2 R/W 0x0		xlate1: whether to use ccos, high effective
xlate0UseCcfi	1	1:1 R/W 0x0		xlate0 whether to use ccfi, high effective
xlate1UseCcfi	1	0:0 R/W 0x0		xlate1 whether to use ccfi, high effective

The generation of EVlanXlate Key is shown in Figure 2-10. Upd.{Svid, Scos, Scfi, Cvid, Ccos, Ccfi} information, the EVlanXlate Key table can be obtained through the corresponding configuration information of eEeXlateKeyCtl, and the The index information of eEeXlateSrm is used to obtain the Vlan information and VlanOpIdx used to add or delete, so as to to edit.

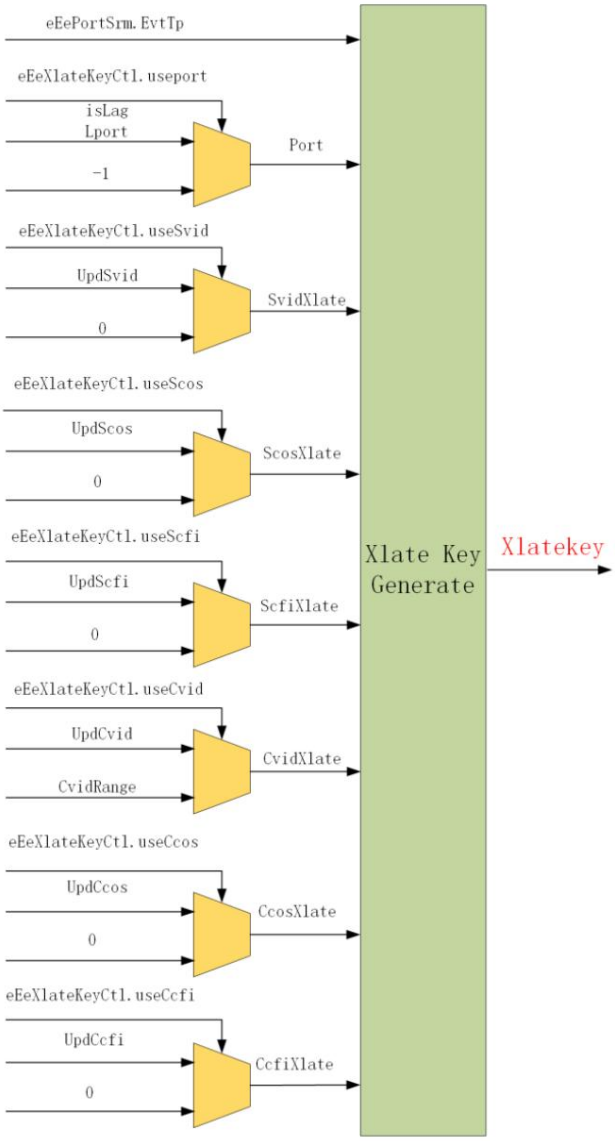


Figure 2-10 EVlanXlate Key generation method

2. EXlateTcm

Enable Tcam table lookup by eEeXlateKeyCtlx.eVtTcamEn=1. The fields contained in TcamKey are shown in the following table. Vlan related fields come from EVlan Get. When TcamKey matches eEeXlateTcm, the index value of eEeXlateTcmSrm is obtained.

By querying the corresponding eEeXlateTcmSrm, the Vlan information and VlanOpIdx used for modification are obtained.

TcamKey field composition

res	3	0x0	
outIsLag	1	0x0	Indicates whether the output internal logical port number is a LAG port, high effective
outLport	6	0x0	The internal logical port number of the output. When outIsLag is valid, it indicates the LAG port. Otherwise, it indicates Indicates a normal port.
svid	12	0x0	Upd.svid

scos	3	0x0	Upd.scos
scfi	1	0x0	Upd.scfi
cvid	12	0x0	Upd.cvid
ccos	3	0x0	Upd.ccos
ccfi	1	0x0	Upd.ccfi
oeLh	2	0x0	evt key type

2.3.2.3 EPort

When EXlateVlan is not hit, use the port default svid, scos, scfi, cvid, ccos, cscfi for replacement or addition

The value and VlanOpIdx are used to edit EVLan, and the eEePortSrm entry is obtained through the port index query.

2.3.3 EVLAN Edit

The EVLAN editing function is the same as VLAN Edit, which is to add, modify, and delete VLANs.

The inbound VLAN becomes the outbound VLAN, which will not be described here.

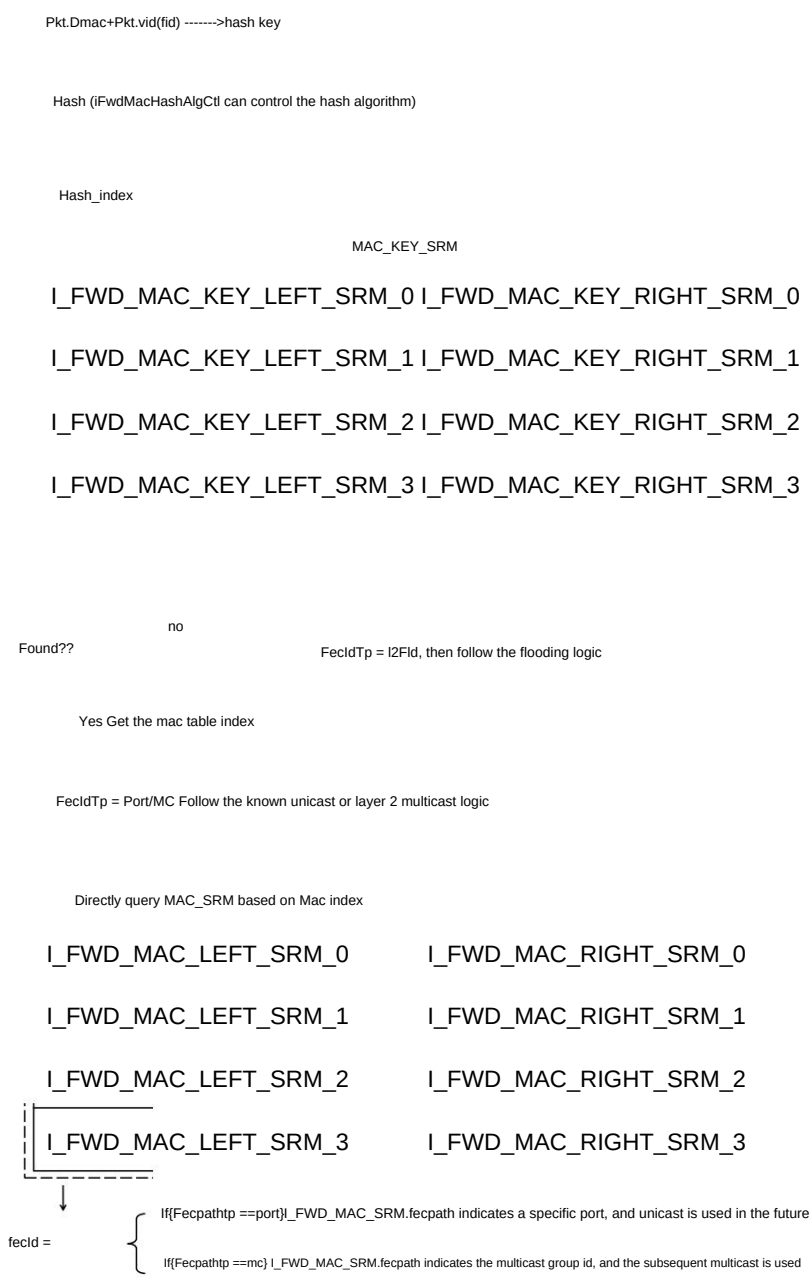
3 L2 forwarding

After the message is processed by the previous vlan, it will enter the l2 forwarding module. In general, the key information currently obtained is: message smac, message dmac, vlan information, (to be precise, the fid information of I_NET_VLAN_SRM), etc. Combined with this information, plus our related configuration of the l2 function, the message will be forwarded by l2 in the chip. In general, L2 forwarding includes 1) forwarding 2) learning 3) aging functions, as well as derived site shifting, blacklist and whitelist functions, which are described in chapters below.

3.1 Bridging

As shown in the figure below, when the message enters the forwarding module, dmac+vid(fid) will be extracted for forwarding. Through hashing, a hash index is obtained to query mac_key_srm. Since hashing involves hash conflicts, 8 slots, 4 on the left and 4 on the right, are used here to solve the conflict problem. Therefore, the obtained hash index will be searched at the index position corresponding to these 8 slots. As long as the valid of the corresponding table entry is 1, it means that the table entry is valid. Then compare the hash key (dmac+fid) with the mac key in the table entry. If they are the same, it means that the I_FWD_MAC_KEY_SRM table hits and the message follows the known unicast logic. Then, the index of the I_FWD_MAC_KEY_SRM hit table entry will be used to obtain the I_FWD_MAC_SRM table (I_FWD_MAC_SRM and I_FWD_MAC_KEY_SRM are one-to-one corresponding) to obtain corresponding information such as the forwarding port to guide subsequent forwarding behavior (the mac table can also mark discards, indicate behaviors such as color, and these fields can be inserted by configuring a static mac table instead of specifying them through learning.); If the I_FWD_MAC_KEY_SRM table does not hit, the subsequent flooding logic will be used. Here, it is also explained that the Layer 2 multicast logic is also implemented by inserting a multicast type mac table. However, the meaning of the I_FWD_MAC_SRM.fecpath found here is the multicast group id, which will be used as the index of the iDstMcGrpSrm table to obtain portBmp for flooding forwarding.

The organization of mac address is divided into two modes: svl and ivl. Ivl means that mac addresses are independent of each other in different vlans; svl means that all vlans share the same mac address table. Our implementation method is: in the I_NET_VLAN_SRM table, there is a field called fid, which forms a vid--->fid mapping. Then when forwarding and looking up the mac key, fid is used as the hash key. When you want to implement the ivl mode, you can use the vid--->fid one-to-one mapping method to implement it; when you want to implement the svl mode, you can use the vid--->fid many-to-one mapping method to implement it.



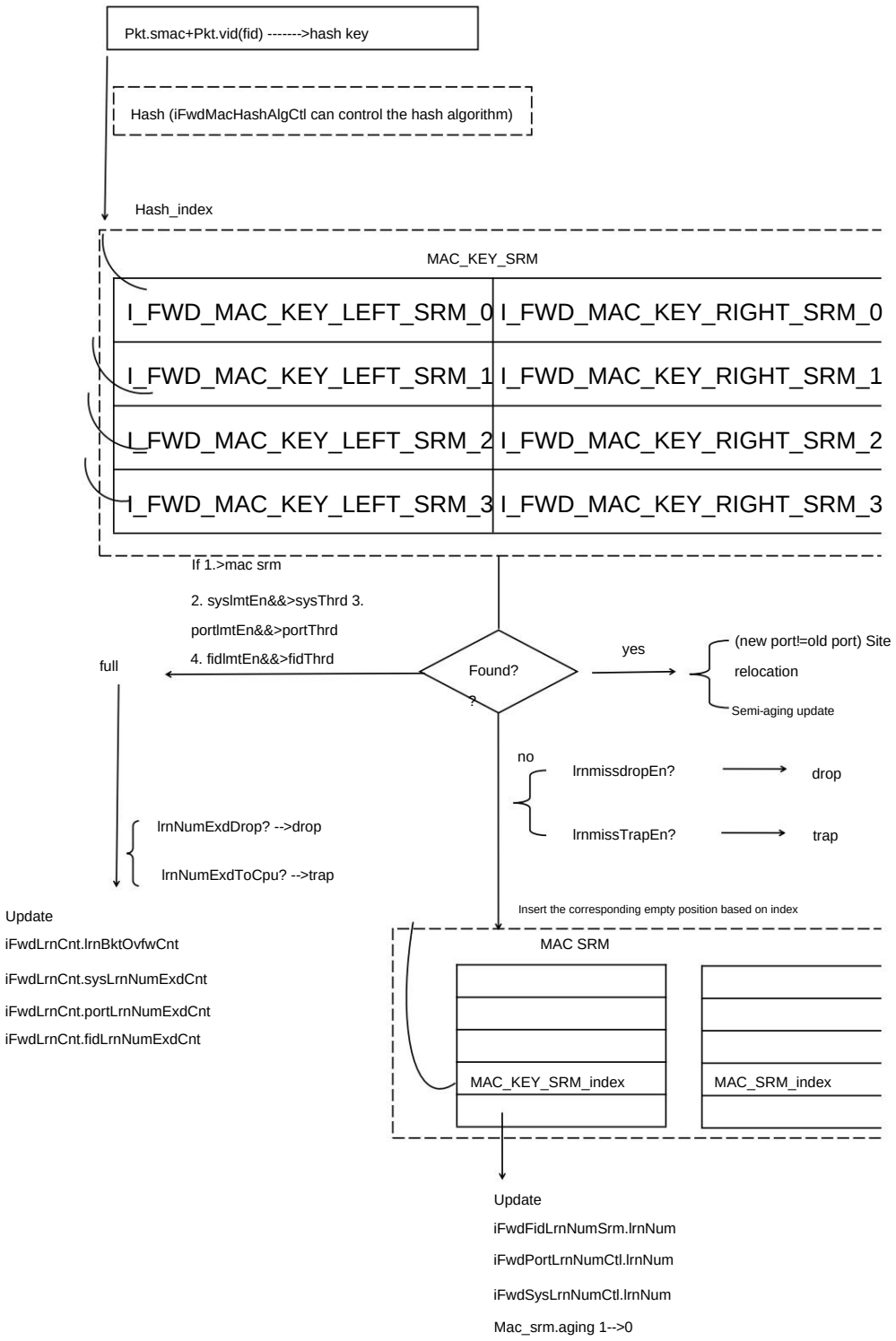
3.2 Learning

As shown in the figure below, at the beginning of the learning process, similar to forwarding, the MAC table needs to be searched. When a message comes in, the MAC table will be queried based on the SMAC address of the message.

Hit: It will determine whether the site is shifted or aged and updated. This will be introduced later;

Missed and no overflow: The smac and ingress port will be inserted into the corresponding table entry, which is learning. At the same time, the table entry of the relevant learning count will be updated.

Miss and overflow: If the mac table is full or exceeds the limit set based on port, fid and system, it will not be learned and the corresponding overflow count will be increased by 1. You can also control whether to turn off learning based on the global setting.



3.3 Aging

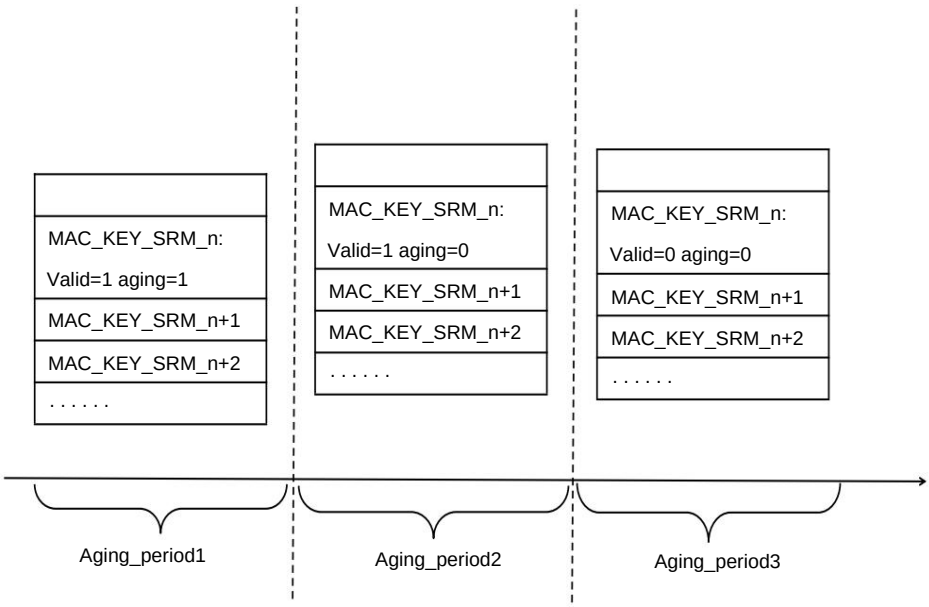
The aging process is shown in the figure below. Since the aging cycle is determined uniformly at the aging cycle time point, in order to avoid some entries being As soon as you learn, you will encounter the aging time point. Usually, aging is carried out in two steps:

As shown in the figure below, at each aging cycle node, the chip does two things: 1) It will set aging to 0 for all items in the table with Valid=1 and aging=1; 1) It will set Valid to 0 for all items in the table with Valid=1 and aging=0 (Note: Valid=1 means that the mac table is valid and will take effect in the learning and forwarding stages; Valid=1 means that the mac is invalid, which is equivalent to the entry being aged.) Then all learned entries in the mac table will experience: 1) Valid and aging are both set to 1 when they are just learned; 2) When the first aging cycle time point is encountered, aging will be set to 0 and Valid will remain at 1; 2) When the second aging cycle time point is encountered, Valid is pulled to 0 and aging is achieved.

Aging time:

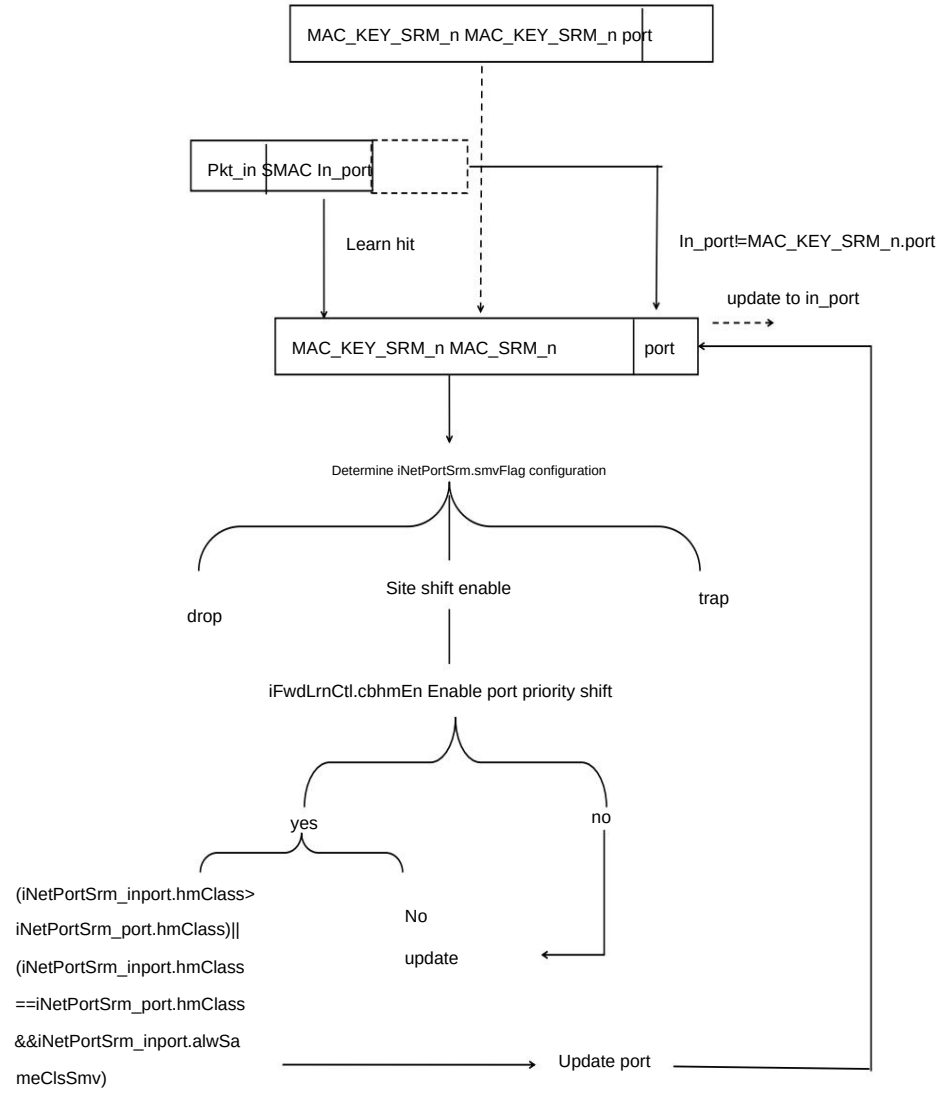
iFwdAgingTimerCtl.counterThrd and iFwdAgingCtl.sCounterThrd determine the aging time (default 5 minutes).
This value is not directly equal to the time, and the proportional relationship has been handled in the SDK code.

You can control whether to turn off aging based on the global or port, iFwdAgingCtl.agingEn and iFwdAgingPortCtl.en (bitmap). In addition, the chip also supports fast aging. If you write 1 to iFwdFastAgingCtl.fastAgingEn, all currently learned MAC addresses will be aged out (except for ports with aging turned off). In addition, static addresses (MAC_SRM.static == 1) will not be aged.



3.4 Site shift

When the incoming message goes through the learning logic, SMAC hits the original MAC table entry, but the inbound port is inconsistent with the port in the table entry, the site shift logic will be generated. As shown in the figure above, according to the configuration, it can be determined whether to perform site shift at this time, or whether to perform port priority site shift, or whether to trap the message that attempts site shift (to CPU). (Note: Here it is determined whether to trap, but not the specific port of the trap. dp trap set port=31 can be used to specify the trap as the CPU port, and port 31 is our internal CPU port). In addition, the above drop and site shift enable are conflicting. If drop is set, site shift will not be performed.



4 ACL Principle

The chip includes multiple flow identification engines, including ACL and EACL. When the data flow passes through the ACL module, it will first determine which fields are matched and pass them to the search engine in the form of Key. Only when there is an entry matching the Key, the corresponding policy engine behavior will be executed.

4.1 Key Composition

Key can contain information parsed from the data packet, such as MACSA/MACDA, SRCIP/DSTIP, VLANID or TCP/UDP port number, etc. It can also contain fields reflecting the packet switching status, such as DROP indication message, loopback message type (such as multicast loopback, mirror loopback), etc.

ACL can classify traffic based on port, L2, L3, and L4 content. Different Keys can be used for different input packet types.
Combination, including MacKey, Ipv4Key, Ipv6Key, MixKey, the specific contents of various Keys are shown in the following table.

Table 4-1 ACL_KEY

Package Type	content
Mac	DMAC[47:0], SMAC[47:0], STAG[15:0], CTAG[15:0], ETHERTYPE[15:0], PORTS[34:0], L3UDF[7:0], etc.
IPv4	DIP[31:0], SIP[31:0], PROTOCOL[7:0], TOS[7:0], PORTS[34:0], L4SrcPort[15:0], L4DsrPort[15:0], L3UDF[7:0], L4UDF[7:0], etc.
IPv6	DIP[127:0], SIP[127:0], PROTOCOL[7:0], TOS[7:0], PORTS[34:0], L4SrcPort[15:0], L4DsrPort[15:0], L3UDF[7:0], L4UDF[7:0], TTL[7:0], DMAC[47:0], TTL[7:0], TCPFLAG[7:0], STAG[15:0], CTAG[15:0], etc.
Mix	DIP[127:0], SIP[127:0], PROTOCOL[7:0], TOS[7:0], PORTS[34:0], L4SrcPort[15:0], L4DsrPort[15:0], L3UDF[7:0], L4UDF[7:0], TTL[7:0], DMAC[47:0], SMAC[47:0], ICMPCODE[7:0], ICMPTYPE[7:0], TTL[7:0], TCPFLAG[7:0], STAG[15:0], CTAG[15:0], IPOPTION[0:0], IPHDRERR[0:0], L3TP[3:0], L4TP[2:0], etc.

4.2 Search Engine

The search engine uses a TCAM table lookup method. Each TCAM table has multiple entries, and each entry (sometimes called a rule) checks the data according to the Key. The entry contains a corresponding bit-by-bit mask field that allows any bit of the data to be ignored during the matching process (Don't Care). This is important because the user can determine which part of a given field is critical and the other parts will be ignored. The search engine will only consider a match when all non-ignored bits match.

The matching process starts from the first entry of the search engine. If the first entry does not match, the next entry is searched until a matching entry is found. At this time, the search engine sends the offset of the corresponding entry to the policy engine.

The entries of the lookup engine are composed of four TCAMs arranged vertically next to each other. Together, these TCAMs determine the depth of the lookup engine (e.g., 512 entries). This default mode is called single-width mode. To provide greater flexibility, the TCAM can also be configured in double-width mode and quad-width mode.

Double-width mode allows the bottom TCAM to be configured to the right of the upper TCAM, forming a left-right positioning. In order for a given entry to match, the key must match TCAM A and TCAM B. The advantage of this configuration is that the width of the key and the match entry is twice as wide, allowing more bits to be used in the match condition. However, this configuration means that the number of available entries is halved. So if single-width mode provides 512 entries, then double-width mode will have 256 entries.

Similarly, the width of the key in quad-width mode is four times that of the single-width mode. Only when all four TCAMs match at the same time can the corresponding behavior be executed. action, the number of available entries is one quarter of that in single-width mode.

4.3 Policy Engine

A policy engine contains all actions associated with matching entries in a lookup engine. Each policy engine contains the same number of entries as a lookup engine.
The offset (or index) of the matching entry in the lookup engine is used as an index into the policy engine to obtain the behavior action information.

The behavior of the policy engine includes adding, modifying, deleting VLANTAG, hitting count statistics, discarding messages, redirecting messages, and sending to the CPU. Sampling, specifying queue numbers, specifying flow-based rate limit templates, specifying QOS templates, etc.

4.4 Search process

Figure 4-1 shows the ACL search process.

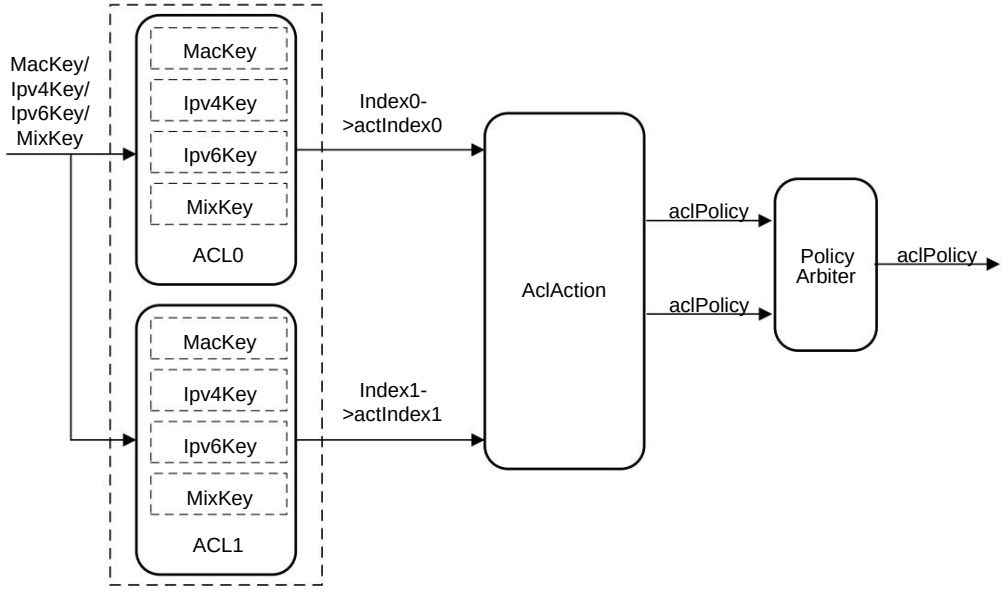


Figure 4-1 ACL search process

ACL includes two flow search engines ACL0 and ACL1, which share all ACL search entries, with a total of 512 entries. The iAclTcm table can be divided into MacKey, Ipv4Key, Ipv6Key, and MixKey according to the configuration. The key type is determined by the two bits iAclTcm[index].key[159:158]. When the value is 0, it means that the current index entry is MacKey; 1, it means Ipv4Key; 2, it means Ipv6Key; 3, it means MixKey. The number of entries allocated to each key type is configured through the iAclCtl register. The composition of the iAclCtl register is shown in Figure 4-2.

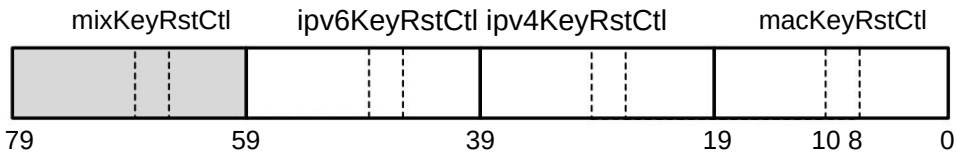


Figure 4-2 Schematic diagram of iAclCtl register composition

The iAclCtl register consists of four fields: macKeyRstCtl, ipv4KeyRstCtl, ipv6KeyRstCtl, and mixKeyRstCtl.
Each domain has 20 bits. The values of these domains can determine the entry allocation of various keys in the iAclTcm table. Take the macKeyRstClt domain as an example.
It consists of {indexBase[8:0], keySize[1:0], tableBase[8:0]}, where indexBase indicates the MacKey type in the iAclTcm table.
The base address in keySize indicates the Key width mode (0 indicates single-width mode, 1 indicates double-width mode, and 2 indicates quad-width mode). tableBase
Indicates the base address of the AclAction behavior table (iAclTcmSrm) corresponding to the Key type (usually the tableBase value is the same as the indexBase value).
The number of entries occupied by a certain Key type can be calculated based on the difference in indexBase between adjacent Key types. For example, assuming MacKey
The indexBase of MacKey is 0, and the indexBase of the adjacent Ipv4Key is 100, so the number of iAclTcm entries occupied by MacKey is 100.
strip.

The ACL search process selects the search keys of the two flow engines, ACL0 and ACL1, based on the different types and configurations of the data packets.
(MacKey/Ipv4Key/Ipv6Key/MixKey), where the search keys of ACL0 and ACL1 can contain different ACL templates respectively.
After the search operation is completed, if the message matches the entries of ACL0 and ACL1, the indexes index0 and index1 are returned respectively, and the search indexes actIndex0 and
index1 of the behavior table iAclTcmSrm are obtained through the operation actIndex = (index-indexBase)>>keySize+tableBase.
actIndex1, where the iAclTcmSrm table stores the processing behavior of the flow. According to the characteristics of TCAM, each search engine only outputs the first
Matching items, so when applying, you need to configure them according to the priority of the items, and configure the items with higher priority in the front items.

The two ACL lookup engines will obtain two processing behaviors of the business flow. At this time, arbitration is required. The principle of arbitration is that if the two processing behaviors
If there is a conflict between the behavior items, the corresponding processing behavior of the ACL1 search engine shall prevail.

Figure 4-3 shows the detailed service processing flow of ACL TCAM lookup.

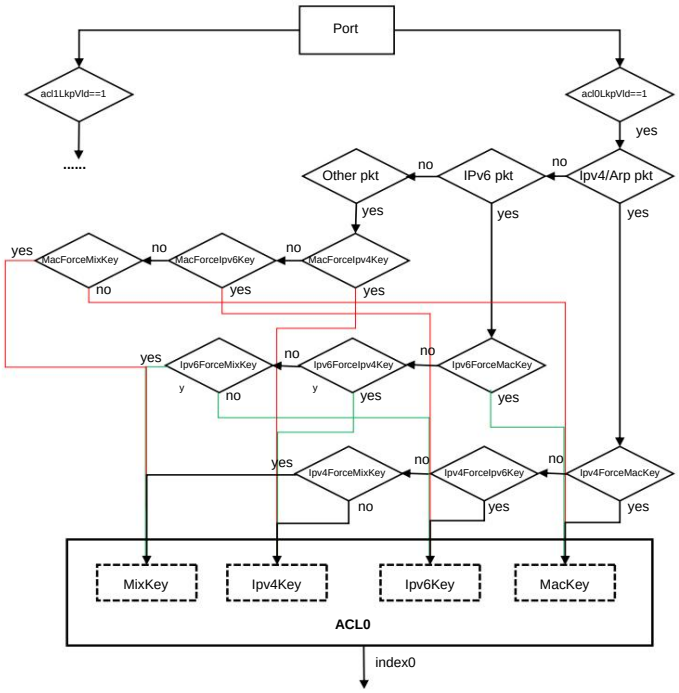


Figure 4-3 ACL TCAM service processing flow

Enable the search switch of ACL0 and ACL1 based on the port, taking ACL0 as an example. If the packet type is IPv4 or Arp, By default, Ipv4Key is used for search. If the message type is IPv6, Ipv6Key is used for search. If the message type is non-IPv4 or Arp When using Ipv6, MacKey is used by default. You can also control whether to force MacKey, Ipv6Key or MixKey based on the port. Perform an ACL lookup.

The key composition and search method of EACL are basically the same as those of ACL.

5 Qos map and policy

5.1 Map

The schematic diagram of the chip internal priority generation module is shown in Figure 5-1. From high to low priority, they are: priority mapping of the Policing module, Acl specifies the priority, the priority is modified after the mac table in the Bridge is hit, the internal priority is specified in the VLAN forwarding table, and the priority is modified in the VLAN translation. The internal priority is mainly used for the subsequent VLAN and DSCP re-marking of packets and the queue scheduling in the TM module.

The most widely used one is to use the map function of the policing module to map the L2 and L3 priorities of the incoming message into the message in the chip pipeline. The specific process is as follows:

According to the priVld set to 0 in the iVtPortTblAct table, the pri map process in the policing module is executed. According to the priority mapping template set in the iPolQosProSrm table, it is determined whether the mapping uses the L2 header or the L3 header (if the L3 header is to be used, the L3 type of the message must be IPV4 or IPV6). If the L2 header is used, whether to use the cos in the stag or ctag as the initial value of the mapping.

Use L3 header information mapping, according to the table index pointer phbPtr in the iPolQosProSrm table and the dscp value in the L3 header information Calculate the index {phbPtr, dscp} of the DSCP mapping table, and obtain the mapped internal priority and color through the lower 2 bits of tos;

Use L2 header information mapping, determine whether to use scos or ccos according to the stag and ctag usage information in the template, calculate the index {phbPtr, cos, cfi} of the cos mapping table according to the index pointer in the template and the L2 header information, and use the index to query the iPolCosPriMapSrm table to obtain the mapped internal priority and color.

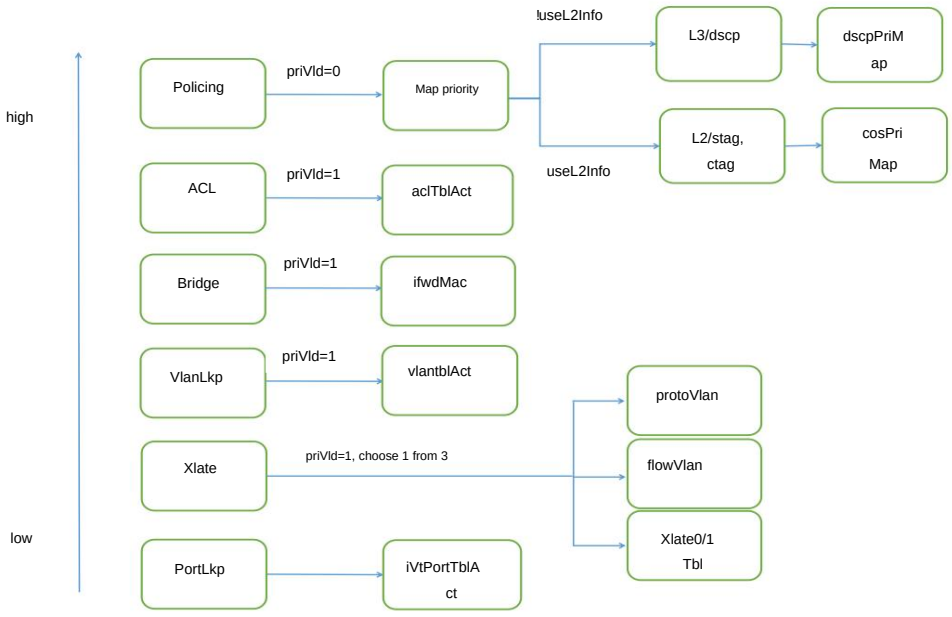


Figure 5-1 Schematic diagram of the chip internal priority product module

5.2 ReMark

The Remark function is mainly to re-mark the priority of the message in the chip to the outgoing message, and modify the priority of the outgoing message. The chip Remark function supports L2 and L3 priority remarking of vlan and tos. In the eDstRmkInfoSrm table, the remarking enable flags of scos, ccos and tos can be configured. It is also possible to modify only the dscp of the message when remarking tos. One important point is that remarking tos can only take effect when the L3 type of the message is IPV4 and IPV6.

By configuring the re-marking index index in the eDstRmkInfoSrm table and combining the priority pri of the message in the chip, the index {index, pri} of the eDstPriRmkSrm table is obtained. There are three sets of tos, pri and cfi values in this table entry, which correspond to the priority re-marking values of messages of different colors (green, yellow, and red).

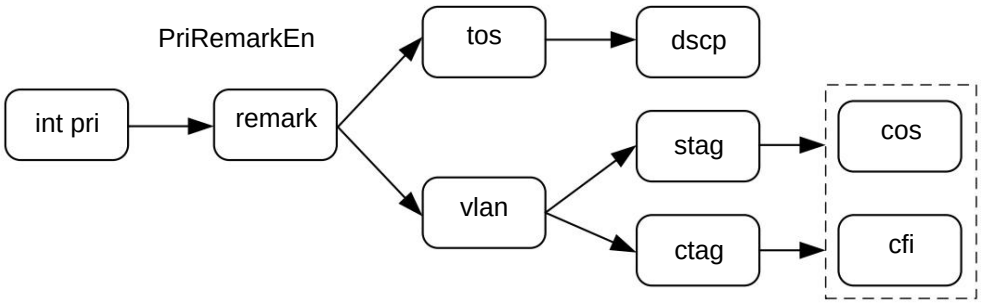


Figure 5-2 Remark function diagram

5.3 Policing

This chip supports two types of policing, port-based and flow-based policing, and supports hierarchical superposition of the two. Single-layer policing supports two modes: srTCM and trTCM, and two modes: color blindness and color sensitivity. Hierarchical policing supports three modes: MIN_ONLY, MAX_ONLY, and MIN_MAX.

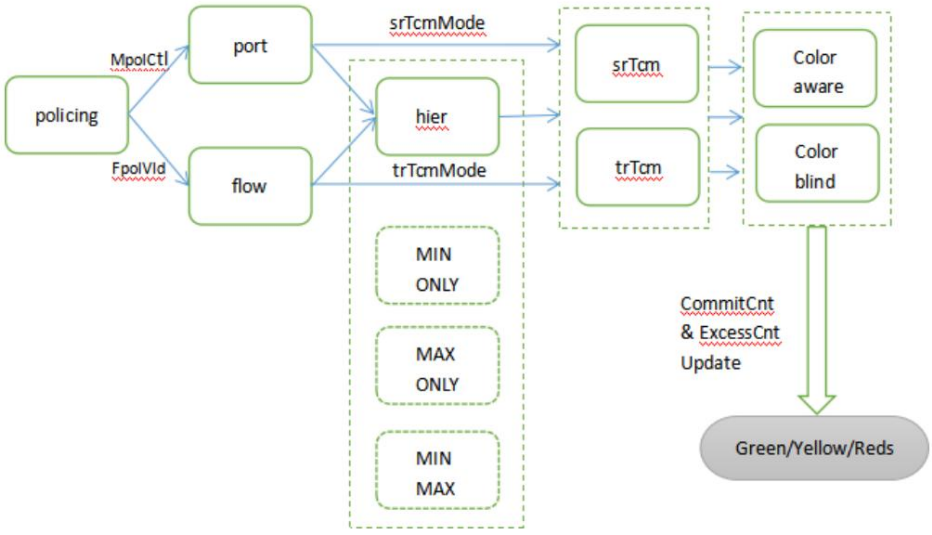


Figure 5-3 Introduction to policing types

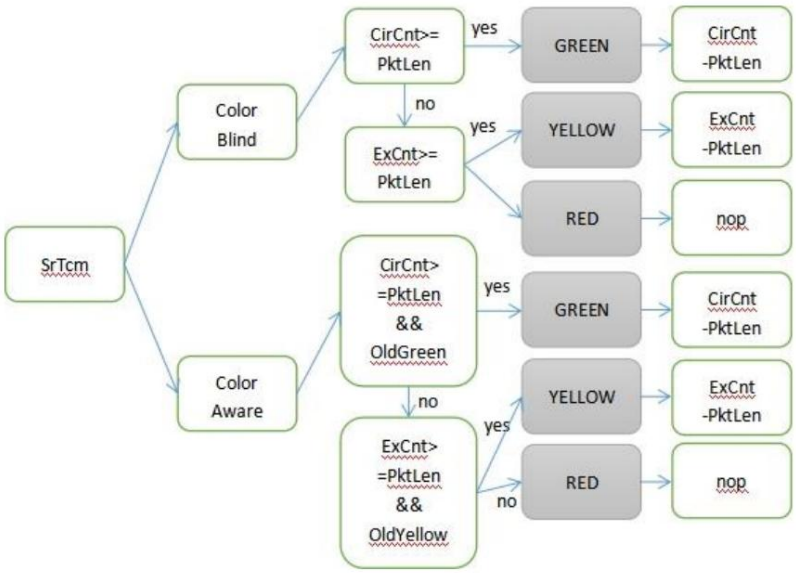


Figure 5-4 SrTcm mode introduction

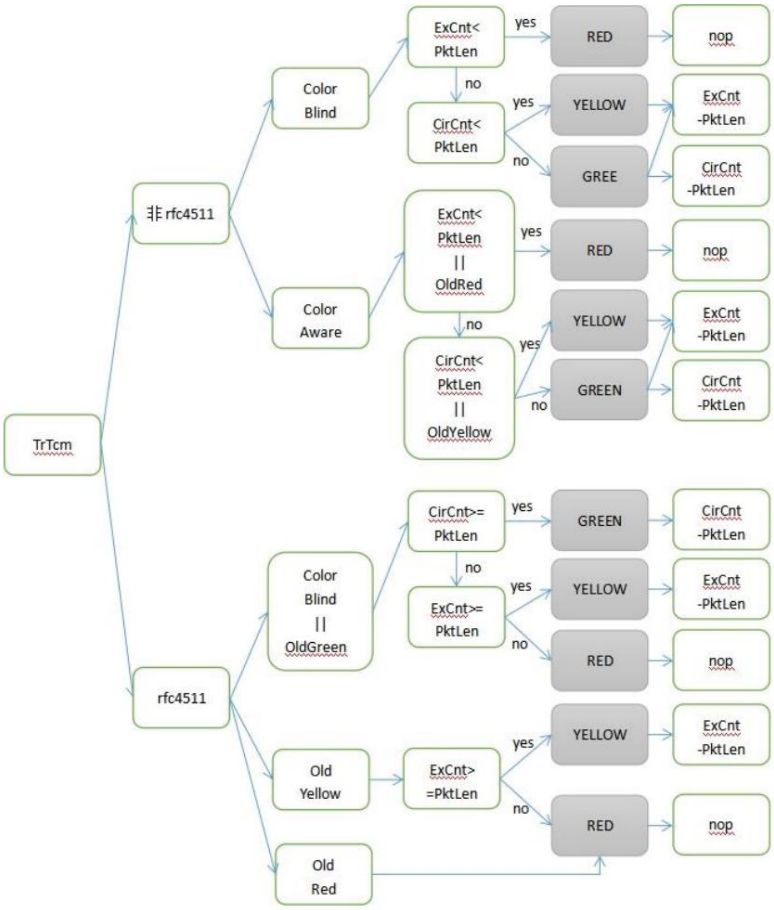


Figure 5-5 TrTcm mode introduction

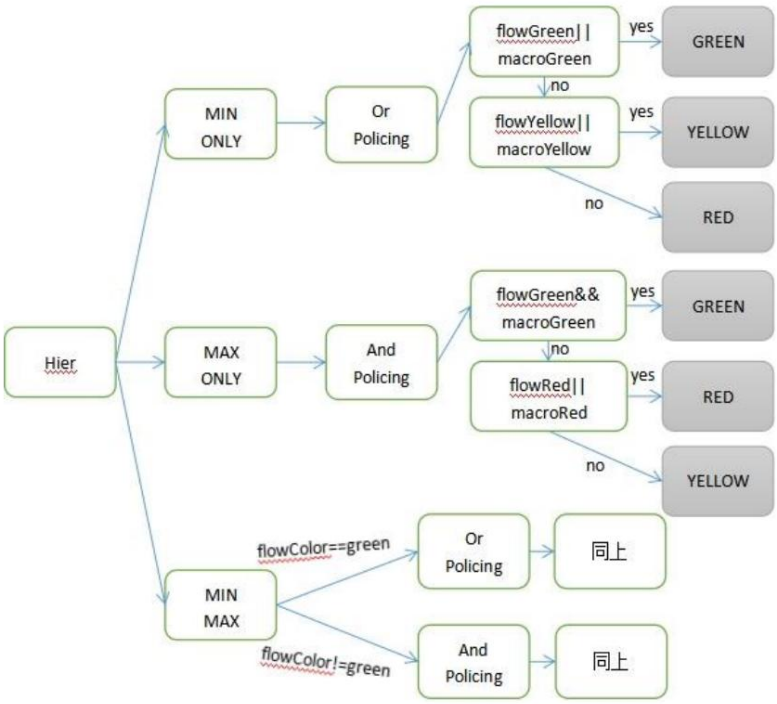


Figure 5-6 Hier mode introduction

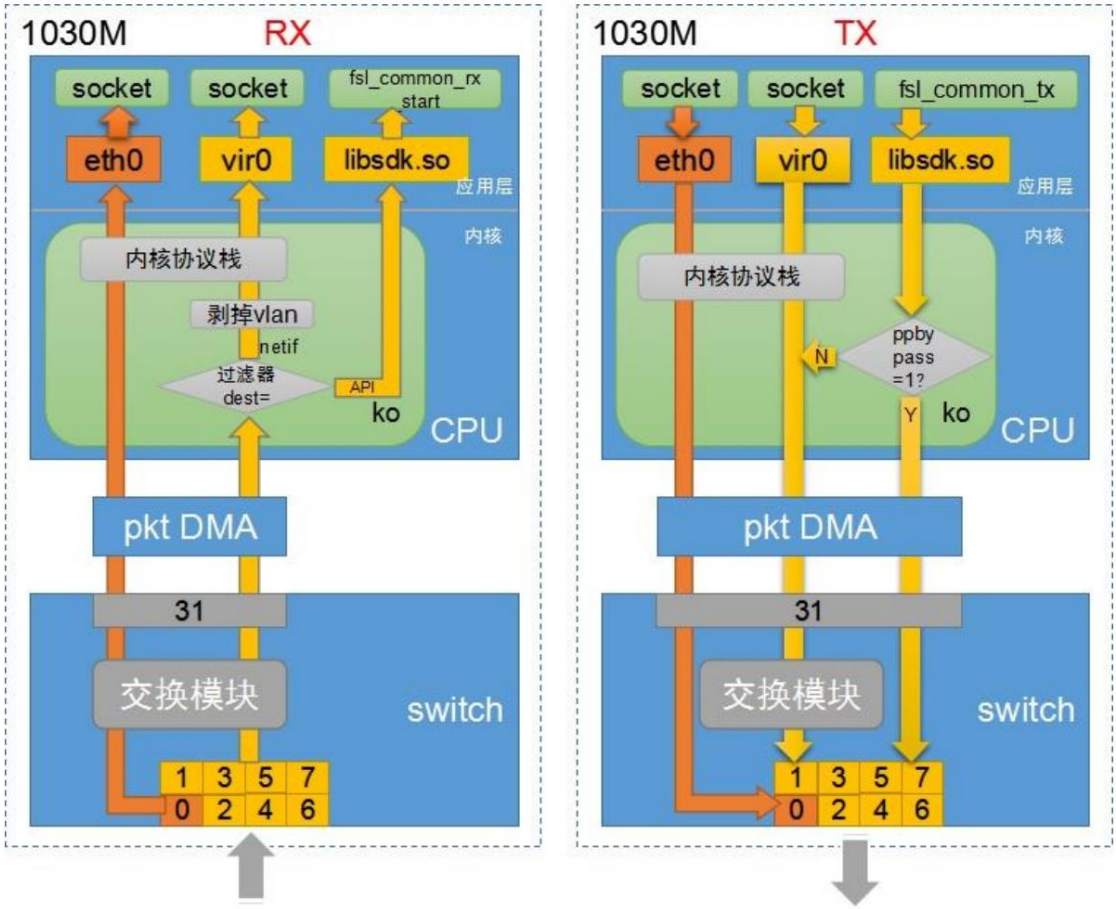
The speed limit of Policing is closely related to the clock frequency of the chip. In ipolFlowUpdCtl and ipolMacroUpdCtl, you can set the token bucket fill enable, token bucket update cycle and other parameters (timer0, timer1, timer0Num, timer1Num). These parameters and the clock frequency together determine the control granularity of commitRate, excessRate, cbs, ebs, etc. in iPolFlowMeterSrm and iPolMacroMeterSrm. The specific calculation formula is as follows:

base=clock*(timer0Num+timer1Num)/(timer0*timer0Num+timer1*timer1Num)

The actual update rates of bucket c and bucket e are: commitRate*8*base, excessRate*8*base, in bit/s.

After the chip usage scenario is fixed, clock is a fixed value. You can adjust the update granularity of cir and eir by modifying timer0, timer1, timer0Num and timer1Num to adjust the speed limit accuracy and speed limit range. Timer0 and timer1 cannot be set too small. If there are too many entries that need to be filled in the token, time0 and timer1 are set too small, which will result in failure to completely fill all entries within the time of time0 and time1, resulting in inaccurate speed limit. It is recommended to set the value to more than 2000.

6 CPU sends and receives packets



1、eth0和vir0为网络驱动（xy1000_net.ko）创建的2个网络设备，有独立的ip地址和mac地址。
2、API (fsl_common_rx_start 和 fsl_common_tx) 为 libsdk.so 提供的收发包接口函数，该接口通过 netlink 将包绕过协议栈进行收发。
3、eth0处理经由调试网口，且经过协议栈的包。
4、vir0处理经由非调试网口，且经过内核协议栈的包。
5、API处理经由非调试网口，且不经过内核协议栈的包。
6、该调试网口可任意设置。

7 Revision Information

Revision time	Version	describe
2021.5.8	V1.0	initial version.
2021.5.21	V1.1	Added overall block diagram.
2022.12.23	V1.2	Content optimization.