PM7322

PROGRAMMING THE RCMP VC SEARCH TABLE

Issue 1: January 30, 1996

TABLE OF CONTENTS

1. INTRODUCTION

1.1 PURPOSE AND SCOPE

This document provides a detailed guide for programming the VC search tables for cell processing in the RCMP. It is intended for software and system designers who are using or planning to use the RCMP, as well as for audience in general who wants to have a better understanding of the algorithm that RCMP uses to identify the virtual connection (VC) of an incoming ATM cell.

It is assumed that the reader has a basic understanding of the RCMP functionalities. Please refer to the datasheet (PMC-940904[1]) for a detailed technical description. In addition, basic knowledge of the ATM protocol would be helpful[1].

First, the background of the search is given, followed by an overview of the RCMP software architecture. The data structures used and the search algorithm itself are then described in detail. This is followed by an explanation on how to initialize the table, and how to add/drop VC connections. Finally, a list of considerations for binary search is provided. A Frequently-Asked-Question (FAQ) section is included at the end.

1.2 WHY BINARY SEARCH?

When an ATM cell arrives at a switch, it needs to be processed before it is actually routed to the destination. The header of the cell needs to be translated, since the VPI/VCI address is only valid for any one link between switches. The new header, together with the optional appended bytes called routing tags, are used by the switch to determine the routing path and to perform flow control on the various ATM VC's. Policing (ie. enforcing the agreed-upon maximum cell rate and cell delay variation) is done on each VC to ensure conformance to the user traffic contract and to avoid congesting the switch. OAM cells are identified and processed to support the network control function. Furthermore, a variety of statistics are kept on either the overall cell traffic or on a per-VC basis.

As such, there is a large amount of information associated with the processing of a cell for each VC. Since the RCMP can support up to 64K VC's, this large amount of information (up to 5 Mbytes) needs to be stored in an external database (on SRAM's). That is, every time a cell arrives, the RCMP needs to identify which VC it belongs to, and access the database to retrieve/update the information to process the cell.

The RCMP allows full flexibility in VPI/VCI address assignment and it identifies a cell by examining up to 55 bits[2], which includes the physical layer device identifier, the VPI/VCI and the appended routing tag. This corresponds to approximately 3.6×10^{16} combinations. Out of this 3.6×10^{16} possible combinations, the RCMP has to quickly

[1] A text by McDysan and Spohn [2] is recommended to gain basic knowledge of the ATM protocol.

[2] 55 bits includes the 28-bit VPI/VCI. It consists of the 16-bit Primary Search Key and the 39-bit Secondary Search Key, both of which will be explained in Section 3.

identify which VC Table Record in the 64K VC Table the incoming cell belongs to, in order to minimize the cell processing delay. This VC identification process is implemented by the VC binary search algorithm.

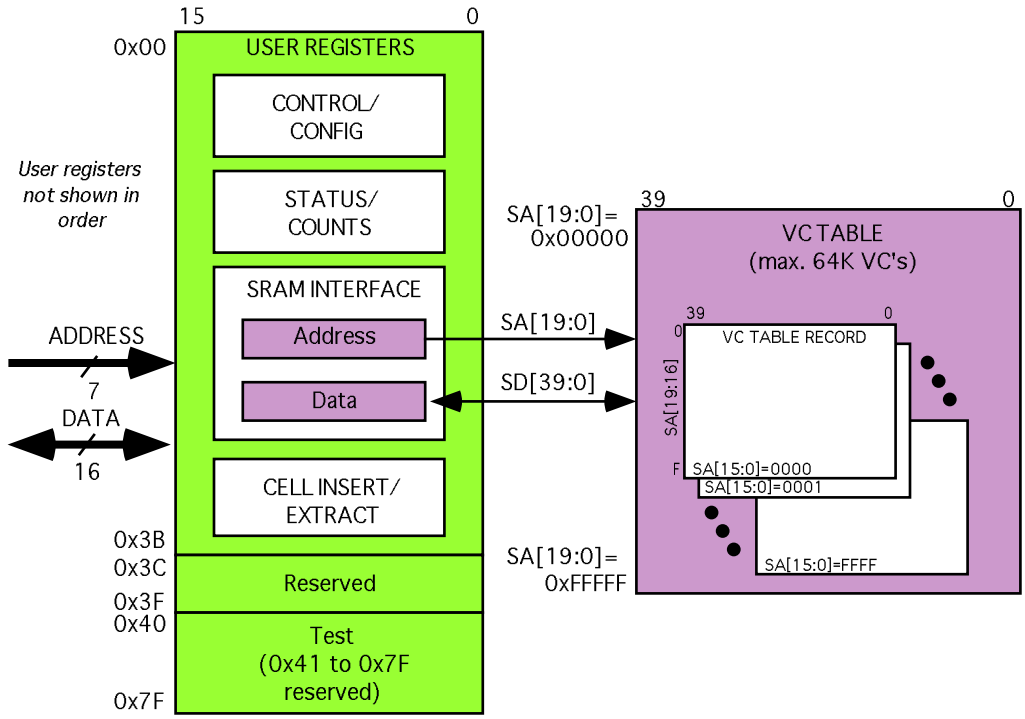
1.3 DEFINITION OF TERMS

Root	The root of the binary search tree, which is the starting point of the binary search process. There are in general many roots (ie. binary search trees) to start at. The particular root is selected by the Primary Search Key.
Node	The branching points in the binary search tree. The starting node of the tree is called the root. The end nodes of the tree are the leaves.
Branch	The path followed by the binary search algorithm at each node down the search tree. Two branches emerge from the root and from each of the nodes that follows.
Leaf	The end of the binary search path, where the VC is identified (ie. the 16-bit SRAM address pointing to the VC Table Record is found).
Primary Search Key	The first part of the two-part search process to identify the VC is a direct look-up to find the root of the binary search tree. It uses the primary search key as a pointer to the root. The Primary Search Key consists of up to 5 bits of the PHY (physical layer) ID and up to 16 bits of the Routing Word, making a total of 16 bits for this key.
Secondary Search Key	The second part of the two-part search process to identify the VC is the binary search. It uses the Secondary Search Key to determine the correct branching to reach the leaf. The Secondary Search Key consists of up to 11 bits of the Routing Word, and the 28 bit VPI/VCI address, making a total of 39 bits.
PHY ID	The number that uniquely identifies which physical layer device sources the incoming cell.
Routing Word	A 128-bit word which consists of the appended bytes (up to 11 bytes) and the cell header. This word is used to construct the Primary and Secondary Search Keys. For instance, for the Primary Search Key, up to 16 bits are chosen (16 bits consecutively) from anywhere in the Routing word.

2. RCMP SOFTWARE MODEL

The RCMP register space and the VC Table are organized as shown in Fig. 1. The RCMP uses only 16-bit registers. All accesses to the RCMP is performed through a 16-bit data bus and a 7-bit address bus.

Fig. 1 RCMP Software model



The user register space is divided into four main parts: 1) Control and configuration registers (eg. for policing), 2) Status registers and a variety of counts (cell counts, traffic statistics), 3) SRAM interface, and 4) Cell Insert/Extract Buffer.

The VC Table consists of VC Table Records, each having 16x40-bit words[3]. Each table contains all the information necessary for the processing of one VC. A 16-bit address, SA[15:0], or sometimes referred to as the VC Table Index, is used to point to each table. SA[19:16] is used to point to one of the 16 40-bit words in the table. The RCMP supports a maximum of 64K VC Table Records. If fewer than 64K VC's are supported, the amount of external memory needed can be reduced. Memory can also be reduced if certain functions within the VC Table Record are not required (eg. if performance

[3]Please refer to PMC-940903[1], the RCMP engineering document, for a detailed description of the VC Table.

monitoring is not required, than the last 3 words in the VC Table Record can be removed).

Read operation

The VC Table Records are accessed through the SRAM interface only. For a read operation, the user writes the 20-bit SRAM address into Register 0x22: Bits [3:0] (for the most significant 4 bits of the SRAM address) and Register 0x21 (for the least significant 16 bits of the SRAM address), and writes a 1 to the RWB bit in Register 0x22: Bit 15. This initiates the read operation, and the RCMP will perform the read at the earliest clock cycle that the SRAM interface is free[4]. The BUSY bit (Register 0x22: Bit 14) is asserted while the SRAM access is pending. The user would then wait for this BUSY bit to be deasserted, indicating a successful read. The user can now read from the SRAM data registers (Register 25: Bits [7:0] (MSB), 24 and 23 (LSB)) to retrieve the SRAM data. Alternatively, the BUSYB pin (pin #74) can be monitored to indicate when the SRAM read is finished (BUSYB = 0 means read is pending).

The RCMP performs a parity-check on the read data. If the parity check fails, a maskable interrupt (Register 0x04, Bits[8:4],each bit corresponding to one of 5 bytes of SRAM data) is asserted. However, the read operation will still be completed, with the BUSY bit deasserted.

Write operation

For a write operation, the user writes the SRAM address in the same way as in a read operation, and also writes the SRAM data into the SRAM data registers (Register 25: Bits [7:0] (MSB), 24 and 23 (LSB)). This initiates the write operation, and the RCMP will perform the write at the earliest clock cycle that the SRAM interface is free. The BUSY bit (Register 0x22: Bit 14) is asserted while the SRAM access is pending. The user would then wait for this BUSY bit to be deasserted, indicating a successful write.

[4]In addition to user accesses, the SRAM is accessed by the RCMP for numerous functions, such as performing the binary search, retrieving data from the VC table to police the cell and updating the cell counts in the VC table. The RCMP guarantees one SRAM access after the completion of the processing of every cell.

3. VC SEARCH ALGORITHM

This section describes the VC search algorithm, the associated data structures, and gives a detailed example of the search. The actual search algorithm is implemented in the RCMP, but it is a software function to configure the data structures to enable the search and also to make the search efficient. Refer to Section 4 for a description of the procedures to configure the data structures.

3.1 GENERAL DESCRIPTION

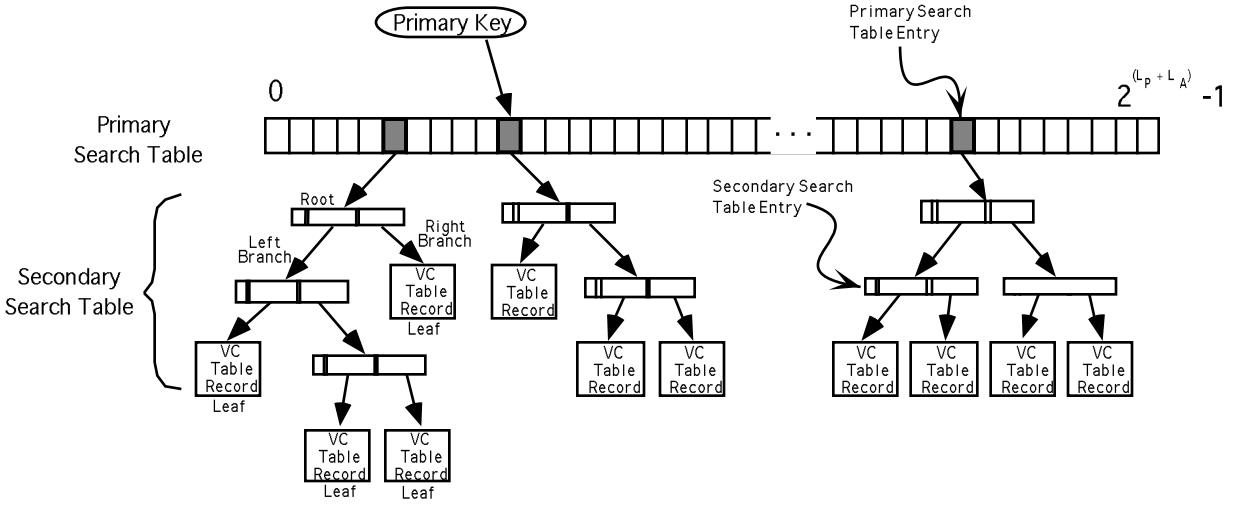
The goal of the VC search is to identify an incoming cell with a VC Table Record stored in memory as quickly as possible. It is also an objective to allow full flexibility in VPI/VCI address (ie. all 28 bits, or any subset, can be used).

This translates into mapping the cell PHY ID, the VPI/VCI address and selected portions of appended bytes (a total of 55 bits) to a 16-bit VC Table index or address (which points to the VC Table Record for the cell). This is the same as finding which VC Table Records out of up to 64K records stored to match the one of 3.6×10^{16} possible combinations.

To achieve this, the RCMP uses a two-part search algorithm. Refer to Fig. 2 for an illustration of the search algorithm. The first part is called the primary search, which is a direct look-up to find the root of the binary search tree. The second part, called the secondary search, is the actual binary search, starting from the root of the tree and ending at a leaf where the 16-bit VC Table index resides. Since the primary search is a direct look-up which takes only 1 SRAM access, it is the secondary search that takes up the bulk of the search time, and is the main factor in determining the cell throughput[5].

[5] Eg. to guarantee a cell throughput sufficient for a 622Mbps (STS-12/STM-4) connection, the number of bits used in the secondary search should be fewer than 18.

Fig. 2 VC Search Data Structure



In each VC Table Record, there are three items used for the search:

1. The Primary Search Table entry
2. The Secondary Search Table entry
3. The Secondary Search Key Confirmation word, which is a 39-bit identifier of the VC

Items 1 and 2 form the search tables used by the primary and secondary searches. They contain the SRAM addresses of the root and nodes of the search tree. There are locations in each VC Table Record reserved for the entries of the search tables. These entries are completely independent of the other information stored in the particular VC Table Record (ie. they do not belong to that particular VC). Item 3 identifies the VC Table Record itself and thus is associated with the rest of the information in the VC Table Record. It is used for comparing to the Secondary Search Key to confirm that the VC Table Record does correspond to the incoming cell at the end of the binary search. A maskable interrupt called INVALI (Register 0x02, Bit 0) is asserted if there is a mismatch in the confirmation. Section 3.3 describes in detail the search table data structures.

In the primary search, a *Primary Search Key* constructed from the 55-bit "address" is used as the 16-bit SRAM address of the Primary Search Table entry, where the SRAM address of the root of the search tree is read (Item 1 in the above).

In the secondary search, a *Secondary Search Key* (up to 39 bits long) constructed from the 55-bit "address" is used for the binary search.

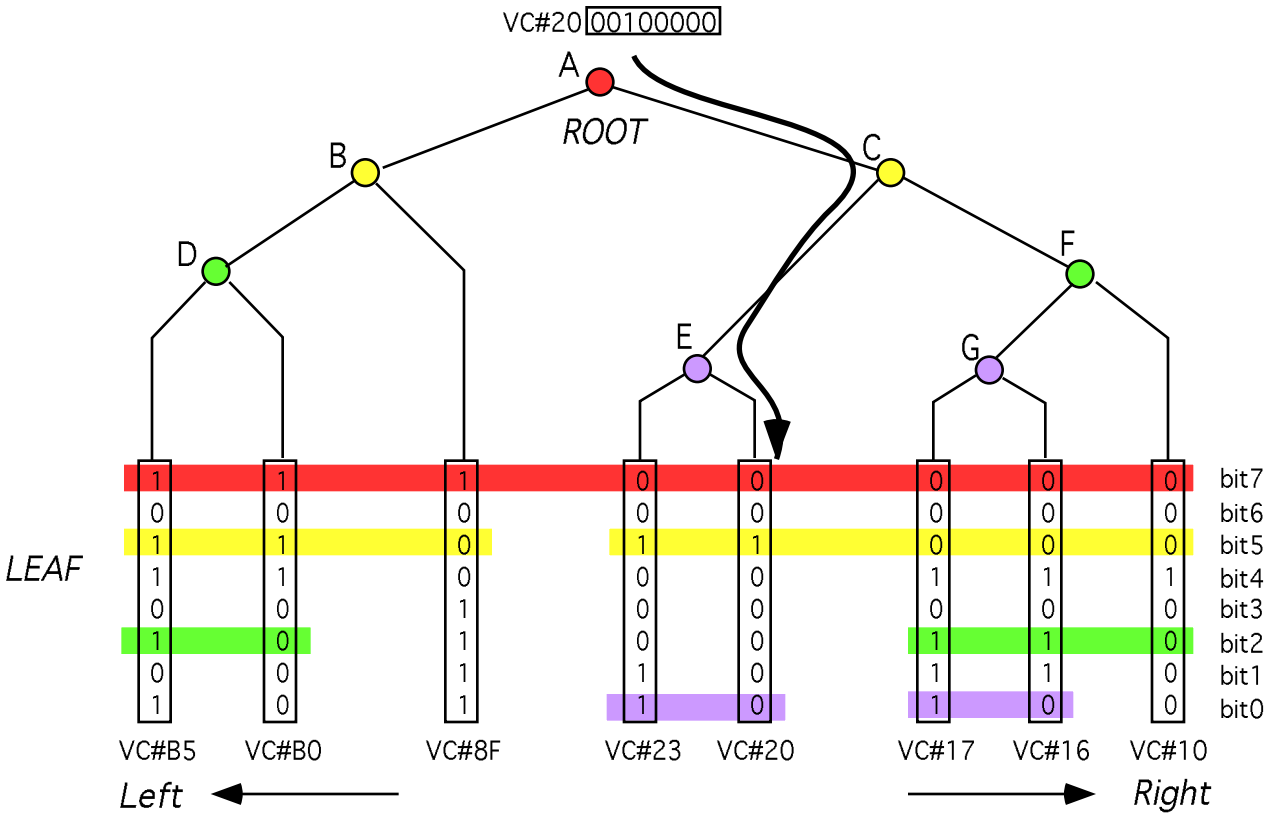
Typically, not all the Secondary Search Key bits are used to identify incoming cells. This allows a faster binary search, by specifying only a subset of bits to be used to distinguish the VC's. At each node, a 6-bit field, called the *Select* field, which is part of Item 2 mentioned above, indicates which bit in the subset is used to make the branching decision.

All of the above concepts will become clear after the reader reads through Section 3.4, a detailed example of the VC search algorithm.

The Binary Search Tree Organization

In the construction of the binary tree, the VC Table Records are organized such that at each node, the branching decision is made by examining only one bit in the Secondary Search Key. That is, at each node, all the VC's below the node are separated into two groups based on the first bit of the address at which the two groups differ. Consider the hypothetical binary search tree in Fig. 3.

Fig. 3 Hypothetical Binary Search Tree



For this hypothetical tree, we have a total of 8 VC's, each uniquely identified by an 8-bit address. Here, a subset of the 8-bit address (bits 7, 5, 2, 0) is used for the binary search. These 8 VC's are organized as follows: Starting at the root, node A, all VC's with bit 7 equal to 0 are grouped to the right, and all VC's with bit 7 equal to 1 are grouped to the left. At node B, all VC's with bit 5 equal to 0 are grouped to the right, and all VC's with bit 5 equal to 1 are grouped to the left. This is repeated at each node (ie. from node A to G). A group that has only one VC becomes a leaf. The horizontal bars

highlight which bit is used at a certain node to separate the VC's. The VC's are arranged effectively in an ascending order starting from the right.

This search tree only contains 8 out of the 256 possible 8-bit addresses. In the worst case, if all 8 bits are used to distinguish the VC's, one would have to go through a maximum of 8 nodes to arrive at the leaf. That is, the search tree would have a maximum depth of 8. By using fewer bits, namely *n*, one would only go through a maximum of *n* nodes to find the leaf. As such, the binary search time is linearly proportional to the number of bits used in address to distinguish the VC's.

A quick run through a binary search

Given a binary search tree organized like the one in Fig. 3, the actual binary search is quite simple. Starting at the root of the tree, bit 7 of the Secondary Search Key is examined and it determines which branch to take (0 = right, 1 = left). After this branching, we arrive at the next node, where bit 5 is examined, which determines the next branch to take. This repeats until a branch identifies itself as a leaf, which means the search has found the VC Table Record that matches the incoming cell. Fig. 3 shows the path that the binary search will follow to identify VC#20.

A final step in the binary search is to compare the Secondary Search Key with the 8-bit identifier (corresponding to Item 3 mentioned above) to confirm that the VC Table Record does match the cell. The reason for this confirmation step is that not all of the 8 bits in the secondary key are necessarily used to arrive at the leaf. For example, if the incoming cell has a VC# of 22, the search algorithm will still arrive at the leaf of VC#20; but the confirmation step will indicate that the incoming VC does not belong the 8 VC's included in this tree.

Section 3.4 gives a detailed example of the search algorithm.

3.2 SEARCH KEYS

The RCMP creates an internal *Routing Word* which is the concatenation of the cell header, cell prepend and cell postpend. The RCMP is programmed to select portions of the Routing Word plus the PHY address to create the Primary and Secondary Search Keys. The search keys, therefore, consist of portions of the cell's header, prepend, postpend and PHY ID. See Fig. 4.

Fig. 4 Search Key Composition

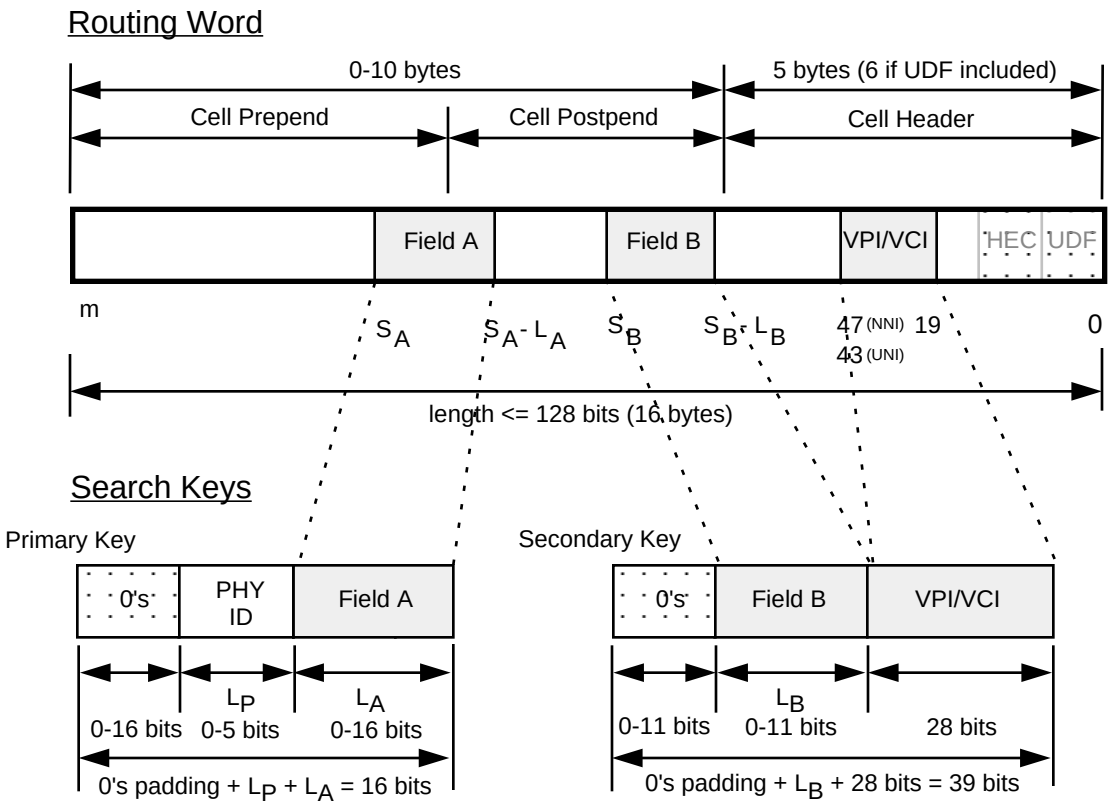


Fig. 4 is not intended to imply any restrictions on the positioning of Field A and Field B. These fields may occur anywhere within the appended bytes or the ATM header. The Primary Key and Secondary Key may also intersect. For ex., Field A can be used to cover the VPI address.

The Primary Search Key is constructed from two fields - the *PHY ID* and *Field A*. The PHY ID field and Field A can be programmed to be 0-5 bits and 0-16 bits long, respectively. The PHY ID identifies which physical layer device is sourcing the cell and must, therefore, include sufficient bits to encode all the PHYs at the PHY Layer interface of the RCMP. The number of PHY ID bits is L_P, which is programmed in Register 0x27. Field A starts at location S_A of the Routing Word and has length L_A. S_A and L_A are programmed in Register 0x28. The Primary Key is right-justified and is pre-padded with 0's. That is, the number of bits in Field A plus the number in the PHY ID field and the 0 padding must be equal to 16. This is because the Primary Key is the 16-bit SRAM address pointing to the root of the binary search tree.

The Secondary Search Key is 39 bits long and is composed of two fields. The first field, *Field B*, is 0 to 11 bits long and may start anywhere in the routing word. Field B parameters include starting position S_B, and length L_B. S_B and L_B are programmed in Register 0x29. The second field is the 28-bit VCI/VPI. This field is always taken from the

cell's header. Like the Primary Key, the Secondary Key is right-justified and is pre-padded with 0's. Note that for a UNI connection, only 8 bits of VPI are valid. The first 4 bits in the 28-bit VPI/VCI field are still placed in the Secondary Search Key but are ignored the binary search.

3.3 SEARCH TABLE DATA STRUCTURES

The following is a detailed description of the primary and Secondary Search Tables, as in Fig. 2.

Primary Search Table

The Primary Search Table contains an array of pointers which point to the roots of binary trees. The table is directly pointed to by the contents of the Primary Search Key, as defined in Section 3.2. The Primary Search Table entry is located in the SA[19:16] = 0001 word, Bits [15:0] in a VC Table Record. This entry is not related to the VC Table Record itself.

The entire Primary Search Table must be initialized to all zeros. A table value of zero represents a null pointer; therefore, the initial state means no provisioned VC's are defined. If a VC is added which results in a new binary search tree (i.e. It is the only connection associated with a particular Primary Search Key.), the appropriate Primary Search Table location must point to the newly created binary search tree root. If the last connection associated with a particular Primary Search Key is taken down, the associated Primary Search Table location must be set to all zeros.

Secondary Search Table

The Secondary Search Table consists of a set of nodes in a binary search tree. Each node in the tree is represented by a 40-bit Secondary Search Table entry located in SA[19:16]=0000 word in a VC Table Record. This entry is not associated to the VC Table Record itself. The Secondary Search Table entry is encoded as follows:

MSB LSB

SD[39:34]	SD[33]	SD[32:17]	SD[16]	SD[15:0]
Select	Left Leaf	Left Branch	Right Leaf	Right Branch

Select — The index of the Secondary Search Key bit upon which the branching decision is based. An index of zero represents the LSB. If the selected bit is a logic one, the "Left Leaf" and "Left Branch" fields are subsequently used. Likewise, if the selected bit is a logic zero, the "Right Leaf" and "Right Branch" are subsequently used. Typically, the Select value decreases monotonically with the depth of the tree, but other search sequences are supported by the flexibility of this bit. (ie. typically, one starts from the most significant bit side and

heads towards the least significant bit when selecting the bits to be used for branching decisions)

If a VC belongs to a *multicast*[6], the select field is set to an all ones pattern, except the last in the linked list. For a multicast entry, the Left Branch gives the VC Table Record address of the multicast VC . The Right Branch points to the Search Table address of the next VC in the multicast. The VC search table therefore forms a linked list and may multicast an arbitrary number of cells. The linked list is terminated by setting the Select field to value that is not all ones. A non all-ones value in the Select field iindicates that the Left Branch field provides the final VC Table Record Address of the multicast set.

Left Leaf

This flag indicates if this node is a leaf. If "Left Leaf" is a logic one, the left branch is a leaf and the binary search terminates if the decision bit is a logic one. If "Left Leaf" is a logic zero, "Left Branch" value points to another node in the binary tree.

If the VC pointed to by the Left Branch is the first VC in a multicast set, the Left Leaf must be set to a logic 1. For the remaining VCs in the multicast set, the Left Leaf value is arbitrary, but it is recommended to be set to a logic 1 for future compatibility.

Left Branch

The 16-bit SRAM address pointing to the node accessed if the decision bit is a logic one. If "Left Leaf" is a logic one, "Left Branch" contains the SA[15:0] address identifying the VC Table Record for the incoming cell. If "Left Leaf" is a logic zero, "Left Branch" contains the SA[15:0] value pointing to another Secondary Search Table entry.

If the Search Table entry is part of a multicast linked list, the Left Branch is the VC Table Record address of one VC in the multicast.

Right Leaf

This flag indicates if this node is a leaf. If "Right Leaf" is a logic one, the Right branch is a leaf and the binary search terminates if the decision bit is a logic zero. If "Right Leaf" is a logic zero, "Right Branch" value points to another node in the binary tree.

[6]Multicast is a function whereby an incoming cell is replicated N times, each with its unique VPI/VCI address. The resulting N VC's will be sent to the output of the RCMP. N can be either unlimited or can be restricted to 64 by writing a 1 to Register 0x20, Bit 3.

If the VC pointed to by the Left Branch belongs to a multicast set, the Right Leaf value is arbitrary, but it is recommended to be set to a logic 0 for future compatibility.

Right Branch — The pointer to the node accessed if the decision bit is a logic zero. If "Right Leaf" is a logic one, "Right Branch" contains the SA[15:0] address identifying the VC Table Record for the incoming cell. If "Right Leaf" is a logic zero, "Right Branch" contains the SA[15:0] value pointing to another Secondary Search Table entry.

If the Search Table entry is part of a multicast linked list (except the last element of the list), the Right Branch is the Search Table address of the next element in the list. If the Search Table entry is the last element in the linked list, this field is arbitrary.

The above encoding defines the binary search tree recursively.

The following special cases must be respected:

1.) A binary tree with only one connection must have both the Left and Right Branches pointing to the solitary VC Table Record. Both Leaf flags must be a logic one.

2a.) If the Primary Search Table is not used (i.e. $L_P = L_A = 0$), the root (ie. the VC Table Record) of the single resulting binary search tree must be located at SA[15:0]=0x0000.

2b.) If the Primary Search Table is in use, no root node shall use location SA[15:0]=0x0000, although this location may be used for nodes at least one level down. A value of 0x0000 in the Primary Search Table represents a null pointer.

3.4 DETAILED EXAMPLE

Based on the hypothetical search tree in Fig. 3, this section gives a detailed example of the VC Search algorithm. Fig. 5 shows the binary search tree. This example illustrates how the search tree is constructed and what happens during an actual binary search.

The following assumptions are made:

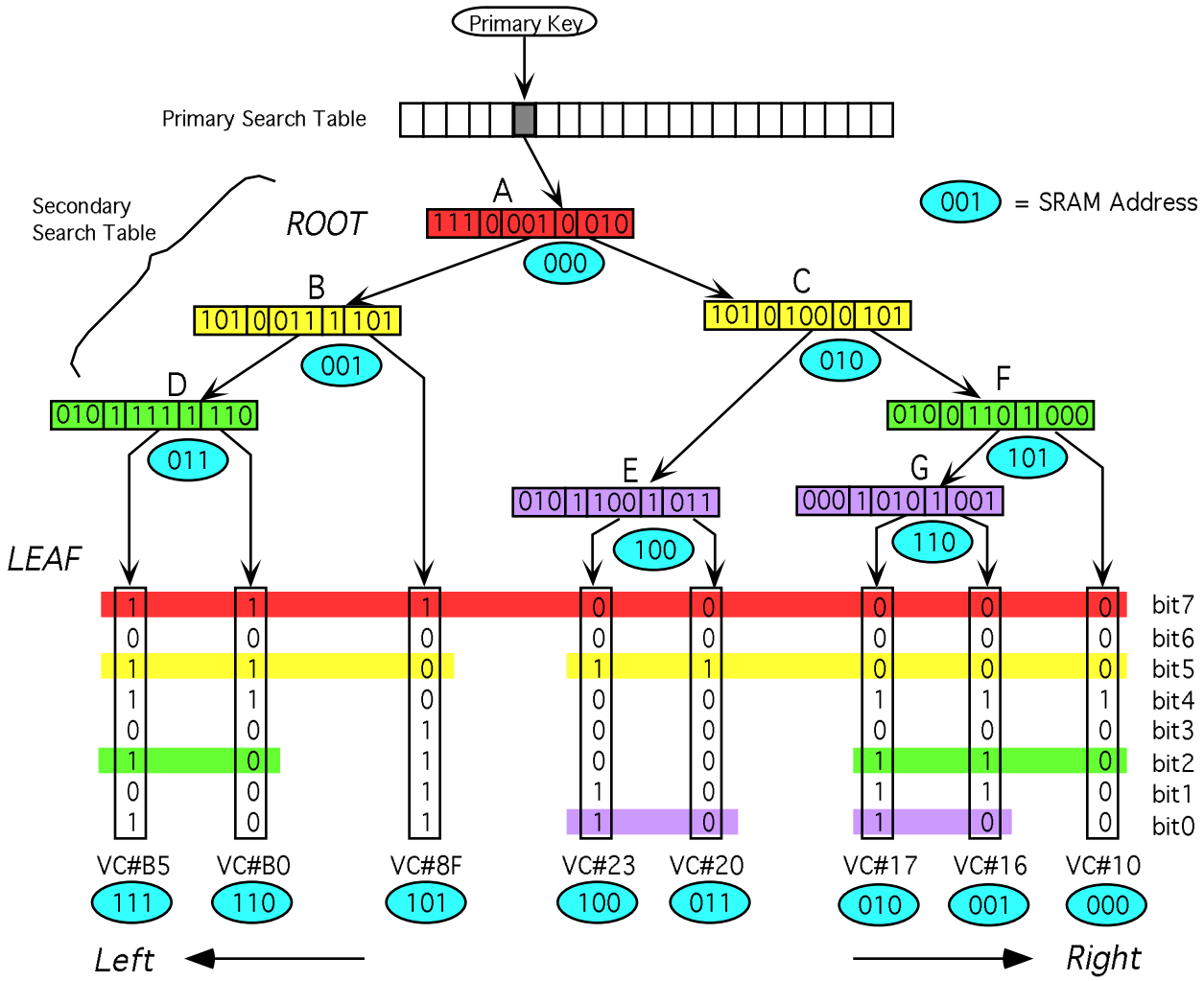
1. Total number of VC's = 8.
2. 8 bits used to identify each VC. (ie. the Secondary Search Key is 8-bits long)
3. 4 bits used for distinguishing VC's: Bits 7, 5, 2, 0[7]

[7]These bits are selected arbitrarily for this example. In general, they are completely dependent on the VC's that exist in the particular binary tree.

4. Since there are only 8 VC's, only 3 bits are used for SRAM address.

Secondary Search Table entries are shown at each node. Each is 11-bit long, which includes a 3-bit Select field, 3-bit addresses for the left and right branches and the 2 leaf indication bits. These entries are arranged such that their vertical positions correspond to which bit is used to make the branching decision.

Fig. 5 Detailed VC Binary Search Tree



The 3-bit SRAM addresses are shown for the search table entries and the VC Table Records themselves. The SRAM addresses for the Secondary Search Table entires can be assigned arbitrarily. The ones in the figure correspond to the alphabetical node names (ie. 000 for A, 001 for B, etc). The SRAM addresses for the VC Table Records can also be assigned arbitrarily. In this case, they are assigned in the same ascending order as the VC#'s.

Example search

Suppose a cell comes in with VC#B0 (1011000), and its associated primary key points to the binary search tree in Fig. 5. At the root (node A), the Secondary Search Table entry is read from the SRAM. The *Select* field indicates that bit 111, or bit 7 should be examined. It is a one, which means the left branch should be taken. The leaf indicator is 0, meaning the leaf has not been found yet. Thus, the left branch address, 001, is used to read the Secondary Search Table entry of the next node, node B. Here, the *Select* field is 101, meaning bit 5 should be used. Bit 5 is a one, meaning the left branch should be taken. The leaf indicator is 0, meaning the leaf has not been found yet. Thus, the left branch address, 011, is used to read the Secondary Search Table entry of the next node, node D. Here, the *Select* field is 010, meaning bit 2 should be used. Bit 2 is a zero, meaning the right branch should be taken. The leaf indicator is 1, meaning the leaf has been found. The right branch address is used to read the 8-bit identifier of the leaf, which is used to compare to the incoming cell. They are the same, and thus the 8-bit address is found that points to the correct VC Table Record that corresponds to VC#B0.

In this example, the depth of the binary search is 4, which is equal to the number of bits in the secondary key as specified by the *Select* field. On one extreme, the 8 VC's can be uniquely identified using only 3 bits, giving a minimum depth of 3 for the binary tree. On the other extreme, a maximum of 7 bits are used, giving a maximum depth of 7 for the binary tree.

4. OPERATIONS

Having explained the VC Search algorithm in the RCMP, this section describes the processes on how to initialize and build up the primary and Secondary Search Tables. VC's can be added or removed on the fly without corrupting a binary search in progress.

It is assumed that there is a replica of the VC Table structure kept by the microprocessor, such that the microprocessor can determine how to add/remove VC's based on this replica. *Any modification to the actual VC Table Records (through the RCMP) has to be duplicated in the replica structure*.

4.1 INITIALIZATION

The following are the microprocessor actions required to initialize the Search Tables and VC Table Records:

1.) Set the STANDBY bit of the Master Configuration register (0x01) if not already set by an asynchronous reset.

2.) Write zeros (null pointer) to every Primary Search Table location (SA[19:16] = 0001).

3.) Write zeros to the fourth word (SA[19:16]=0011) of all VC Table Records. This clears the "Active" bit in the CONFIG field[8].

4.) Clear the STANDBY bit of the Master Configuration register (0x01).

The remaining SRAM locations can be initialized when required.

4.2 ADDING VC's

The following are the microprocessor actions required to provision a connection.

1.) Determine the next available VC Table Record address. This address can simply be one higher than the highest existing VC Table Record address, or it can be the address of a VC Table Record that has been removed. Initialize the contents of the VC Table Record via the Microprocessor RAM Address and Data registers (0x21 through 0x25).

2.) Perform a binary search (using the replica VC Table structure) to determine the insertion point. The last pointer accessed in the search shall be the one modified, be it a Primary Search Table entry, left branch or right branch.

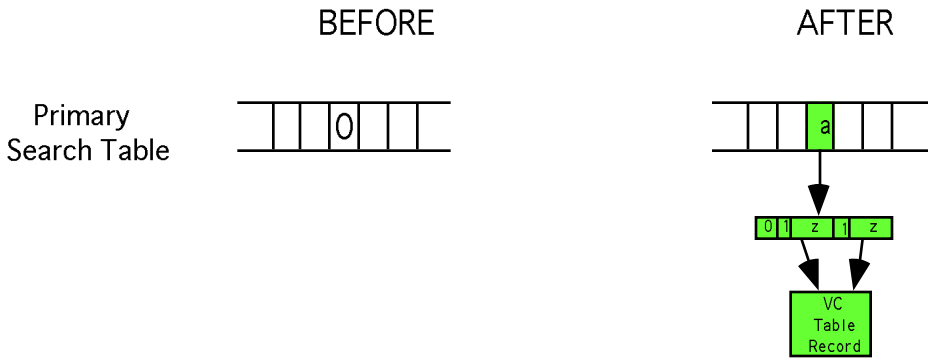
[8]The "Active" bit is only used by the RCMP in its polling process to determine whether the particular VC Table needs servicing. It does not affect the binary search process at all (ie. as long as the VC Table is pointed to by a branch, it is considered valid).

3.) Find a free Secondary Search Table entry[9] and initialize it. The only exception to this is when a single VC Table Record exists in a tree, in which case the solitary Secondary Search Table entry is modified.

4.) Perform a single SRAM write (via the Microprocessor RAM Address and Data registers) to incorporate the new Secondary Search Table entry in the existing tree structure. This step must be performed last to ensure a binary search in progress is not corrupted.

Five distinct types of insertions are possible based on the existing tree structure:

1.) The binary tree is empty. In this case, the null Primary Search Table pointer is modified to point to a newly created Secondary Search Table entry. Because no bits within the Secondary Search Key are required, both the left and right branches of the Secondary Search Table entry point to the same VC Table Record. The "select" field should be set to zero.

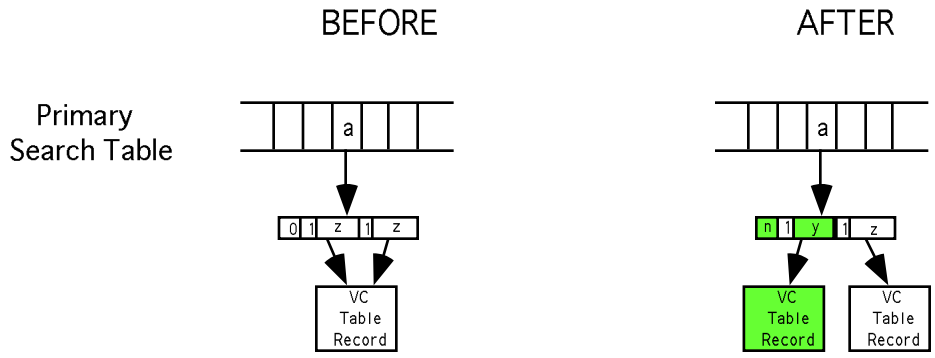


Key to data structure diagrams:

a, b, c - pointers to Secondary Search Table entries
w, x, y, z - pointers to VC Table Records
k,m,n - "select" field contents
The shaded boxes indicate those fields which have been created or modified.

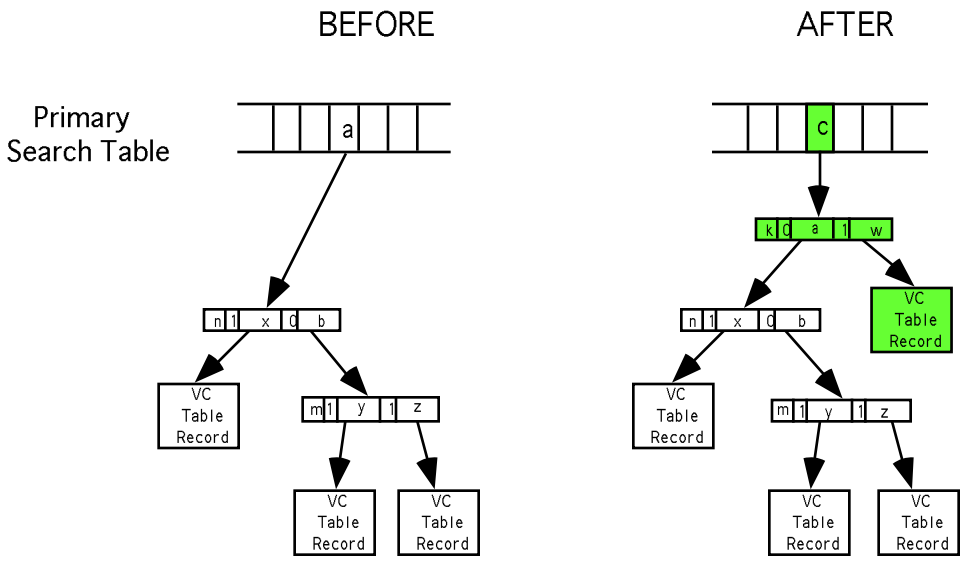
[9]As mentioned in Section 3.3, the Secondary Search Table entry is no associated to the VC Table where it is located. Therefore, *any* free Secondary Search Table entry can be used.

2.) The binary tree contains only a single VC Table Record. Modify the "select" field to index the most significant bit of the Secondary Search Key which differs between the new and existing connection. Modify the left or right branch, as appropriate, to point to the newly created VC Table Record.



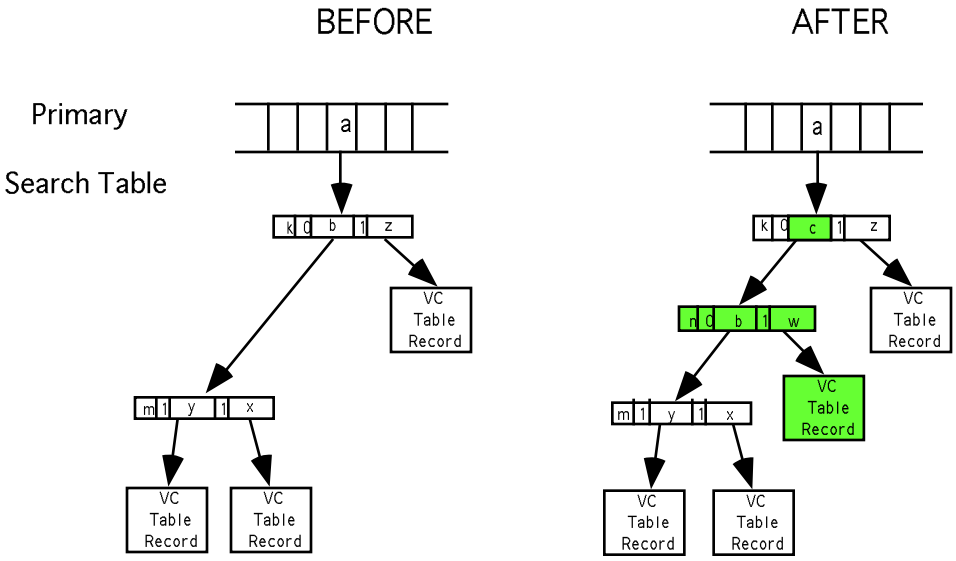
The diagram illustrates the case where the new VC has a one in the decision bit position and the existing VC has a zero in the same bit position. If the new VC had a zero in the decision bit position, the right branch would have been modified instead.

3.) The insertion point is at the root of the tree. This occurs when the new decision bit index is greater any of the indices currently in the search tree. In this case, the Primary Search Table entry is modified to point to the newly created Secondary Search Table entry. The New Secondary Search Table entry points to the new VC Table Record and the old tree root.

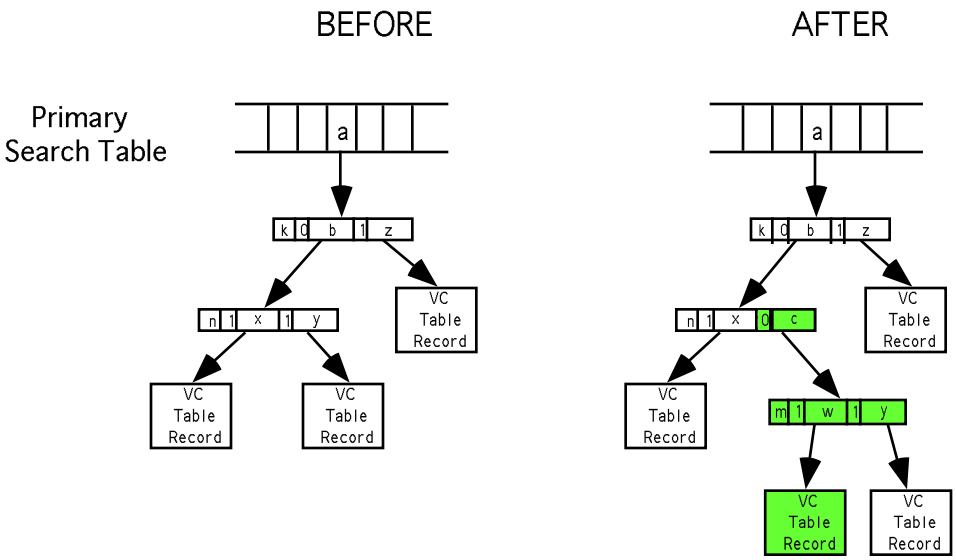


4.) The insertion point is in the middle of the binary tree. The new Secondary Search Table entry points to the new VC Table Record and an existing node

in the tree. The parent of the existing node is modified to point to the new Secondary Search Table entry in the final step of the insertion.



5.) The new Secondary Search Table entry is inserted at a leaf. The search for a candidate insertion point ends on a node which already points to a VC Table Record. The new Secondary Search Table entry points to the existing VC Table Record and the new VC Table Record. The existing Secondary Search Table entry is modified to point to the new Secondary Search Table entry in the final step of the insertion.



4.3 REMOVING VC's

The following are the microprocessor actions required to remove a connection:

1.) Find the location of the Secondary Search Table entry pointing to the connection's VC Table Record.

2.) Modify the parent node (be it the Primary Table entry or another Secondary Search Table entry) of the Secondary Search Table entry being removed to point to the node remaining after the connection removal. The only exception to this is when two VC Table Records exist in a tree, in which case the solitary Secondary Search Table entry is modified. The VC is now considered unprovisioned and any cells belonging to the VC will be discarded.

3.) Tag in software the removed Secondary Search Table entry as free.

4.) Read the final statistics for the connection from the VC Table Record and tag in software the VC Table Record address as free. Also, clear the "Active" bit in the VC Table Record.

The connection removal process examples are not illustrated because the results are exactly the reverse of the connection provisioning. (Swap the "BEFORE" and "AFTER" labels.)

5. BINARY SEARCH CONSIDERATIONS

The following are some considerations in constructing the search keys:

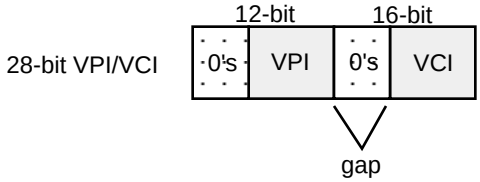
1. Even though the RCMP allows complete flexibility in constructing the search keys, **it is more efficient to examine the VPI first and then the VCI**. This is because, if there is a combination of VC's to be processed: virtual path connections (VPC) and virtual channel connections (VCC), it takes less time to search for a VPC since only the VPI address is used (ie. 12 bits VPI vs 28 bits VPI/VCI).

2. **The number of bits used in the Primary Search Key (ie. number of non-zero bits), which includes the PHY ID and Field A should be less than or equal to N, where 2^N is the maximum number of VC's.** With 2^N VC's, enough SRAM space is provided for 2^N VC Table Records**.** Each VC Table Record has space for one Primary Search Table entry, ie. there are 2^N Primary Search Table entries. Since these entries are pointed to by the Primary Search Keys, this corresponds to a maximum of N bits used in the Primary Search Key. If more than N bits are used, extra SRAM space will be needed just to accommodate the Primary Search Table entries.

3. Given the constraint described in 2. in the above, **it is most efficient in terms of search time to maximize the number of bits used in the Primary.** This is because the primary search is a direct-lookup (ie. it takes only one SRAM access). This is the same as saying the more you can minimize the binary search time (ie. the fewer bits in the Secondary Search Key) the better.

4. Standards[10] require that when assigning VPI/VCI addresses to VC's, the allocated addresses will be the least significant bits of the VPI and VCI fields. Also, the allocated bits will be contiguous.

   Implication: In general not all the bits of the VCI address will be used, ie. the most significant bits are not used. Then given a VPI/VCI address, there will be a un-used gap between the VPI and VCI fields. If we try to maximize the number of bits used in the Primary Search Key, the most we can do is to select the VPI field to be put into the Primary Key, because Field A can only select a contiguous set of bits. The VCI field should then be used in the Secondary Search Key.



[10] Please refer to ITU-T I.150[3] and ITU-T I.361[4] for VPI/VCI assignment.

5. **NNI vs UNI**. The MSB of the third word (SA[19:16]=0010), the same word containing the 39-bit VC identifier, is the NNI bit. The NNI bit identifies if the VC belongs to a Network-Network Interface. If the NNI bit is set to 0, the connection is part of a User-Network Interface (UNI) which means that the four MSB's of the VPI (aka. the GFC field) are excluded from use for the Secondary Search Key. Otherwise, these four bits are included, forming the 28-bit VPI/VCI address.

   If there is a combination of UNI and NNI connections, then only the least significant 8 bits of the VPI should be used in the Primary Search Key. The reason for this is that the most significant 4 bits of the VPI may or may not be valid, and i f they are not (in the UNI case), it will not result in a unique pointer to the Primary Search Table entry.

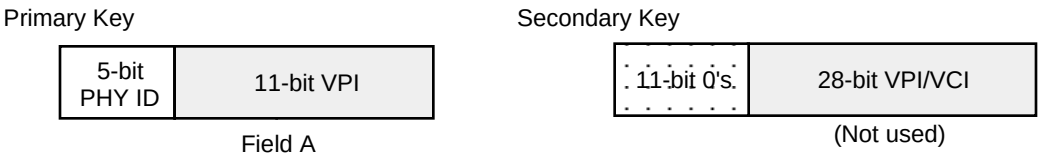
   For each PHY device, which is uniquely identified by a PHYID, there should only be one type of connection, either UNI or NNI.

Examples

Here are some examples of Primary and Secondary Search Key construction:

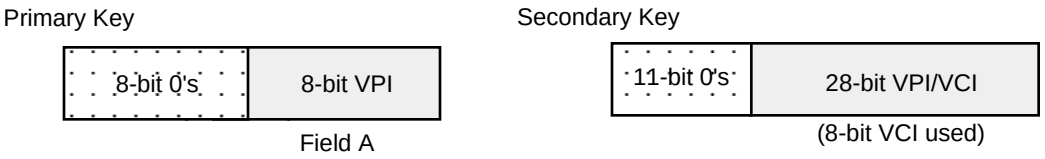
Note: The 28-bit VPI/VCI address always occupy the least significant bits of the Secondary Key. It is the *Select* field (see Section 3) that specifies which bit is actually used in the binary search.

1. 32 PHY's, 2K VPC's with 11-bit VPI, no appended bytes



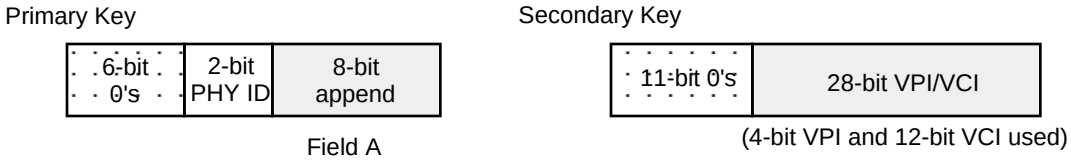
5-bit PHY ID needed for 32 PHY devices, and all of the 11-bit VPI in the Primary Key. In this case, there is no binary search, only the direct lookup. Note that there is still a Secondary Search Table entry for each VC Table Record, even though there is only one VC Table Record in the "root" pointed to by the Primary Search Table entry (see case 1 in Section 4.2 Adding VC's).

2. Single PHY, 64K VCC's with 8-bit VPI and 8-bit VCI, no appended bytes



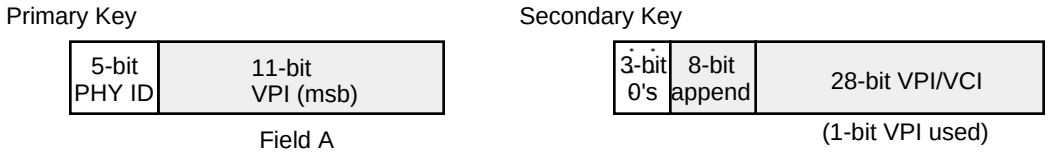
8-bit VPI in the Primary Key, and due to the gap between VPI and VCI, 8-bit VCI in the Secondary Key.

3. 4 PHY's, 64K VCC's and VPC's with 4-bit VPI and 12-bit VCI, 1 appended byte (eg. traffic class indicator)



2-bit PHY ID for 4 PHY devices and only the 8-bit appended byte in the Primary Key. 4-bit VPI and 12-bit VCI in the Secondary Key, as chosen by the *Select* field. The VPI is not chosen to be in the Primary Key due to the gap between the appended byte and the 4-bit VPI address (the 4 bits are the least significant bits).

4. 32 PHY's, 4K VPC's with 12-bit VPI, 1 appended byte.



The most that can be fitted into the Primary Key is 11 bits of the VPI, since there are 5 bits of PHY ID. The remaining VPI bit has to go into the Secondary Key, along with the appended byte. An alternative is the put the appended byte into the Primary Key instead of the VPI, but this will not be maximizing the number of bits in the Primary Key.

6. FREQUENTLY ASKED QUESTIONS

1) Is the NNI bit in the VC Table Record the same as the NNI bit found in Register 0x0C, Bit 5?

No.

The NNI bit in Register 0x0C is a global NNI bit that selects whether the first four bits of the ATM cell header are used when determining whether a cell should be discarded because it is Unassigned or one which is reserved for the Physical Layer. When set to logic 1 (default), the NNI format is used and first four bits of the ATM cell header must be zeros for a cell to be indentified as an Unassigned or Physical Layer cell. When set to logic 0, the UNI format is used and first four bits of the ATM cell header are ignored.

If a mixture of UNI and NNI cells pass through the Input Cell Interface, this register bit should be set to a logic 1. Any Physical Layer UNI cells which contain non-zero GFC fields shall be passed through the input FIFO and subsequently rejected by the VC Identification algorithm. This results in a increment of the Invalid Cell Count instead of the Physical Layer Cell Count.

2) What is the Max VC Table Index in Register 0x26?

The MAX[15:0] bits represent the current maximum VC Table index (SA[15:0]). It is used by the one second servicing algorithm as the first VC Table Record serviced; the index is decremented with each subsequent connection serviced. An accurate value in this location maximizes the efficiency of the RCMP. Fixing this register to all ones guarantees all connections will be serviced each second. Whenever a connection is provisioned or deleted, the microprocessor should evaluate whether to modify the state of MAX[15:0].

Setting MAX[15:0] to all zeros effectively disables the generation of AIS, RDI and Continuity Check cells and disables the clearing of AIS, RDI and Continuity Check alarms.

3) Where does the PHY ID come from?

The PHY ID is the same 5-bit identifier that the input PHY interface uses to select and poll the various PHY devices from which a cell will be transferred. The RCMP can support a maximum of 32 PHY devices. Since VPI/VCI address is valid only within one physical link, VC's from different PHY devices can have the same VPI/VCI address, and therefore, the RCMP and the switch need the PHY ID to distinguish the VC's.

7. ABBREVIATIONS

ABR	Available Bit Rate service
ATM	Asynchronous Transfer Mode
NNI	Network Network Interface
PHY	Physical Layer
RCMP	Routing Control, Monitoring and Policing
SRAM	Static Random Access Memory
UDF	User Defined Field
UNI	User Network Interface
VC	Virtual Connection. Can be either a VPC or a VCC
VCC	Virtual Channel Connection (uses both VPI and VCI addresses)
VCI	Virtual Channel Identifier
VPC	Virtual Path Connection (uses only VPI address)
VPI	Virtual Path Identifier

8. REFERENCE

1) PMC-940904, "Routing Control, Monitoring and Policing Standard Product Datasheet", October 3, 1995, Issue 3, PMC-Sierra, Inc.

2) "ATM Theory and Application", David McDysan and Darren Spohn, McGraw-Hill, 1995, New York, USA.

3) ITU-T Recommendation I.361 - "B-ISDN ATM Layer Specification", March 1993.

4) ITU-T Recommendation I.150 - "B-ISDN Asynchronous Transfer Mode Functional Characteristics", March 1993.

NOTES

Contact us for applications support:

FAX: (604) 668-7301

PHONE: (604) 668-7300

Email: apps@pmc-sierra.bc.ca

PMC-960161P1 Issue date: January 30, 1996

Printed in Canada