BROADCOM®

Data Sheet

BCM16K

Knowledge-Based Processor

General Description

The BCM16K Knowledge-Based Processor (KBP) performs high-speed operations on large-rule databases for a wide range of telecommunications applications, including data center switches and Enterprise routers. It provides network awareness and enables real-time modifications and updates to the routing configuration, making it ideal for packet classification, policy enforcement, and forwarding.

This family of processors addresses next-generation classification requirements through high-performance parallel decisions and improved entry storage capabilities. Parallel operations allow the device to reach decision speeds of multiple billion decisions per second (BDPS). Embedded error correction circuitry (ECC) improves system testability and operational reliability. The key processing unit (KPU) and the context buffer (CB) enable efficient interface transfers with flexible search key construction.

This device seamlessly connects to Jercicho 2C+ BCM88850, Jericho 2C BCM88800, Qumran 2C, Jericho 2 BCM88690, Jericho+ BCM88680, and Jericho BCM88670.

Features

- Dual host enables two host devices to connect to one device.
- Database records 40b: 2048k/1024k.
- Table width configurable as 80/160/320/480/640 bits.
- User Data Array for associated data, width configurable as 32/64/128/256 bits.
- Context Buffer for storing master search keys.
- Up to sixteen parallel searches.
- Simultaneous Multithreading (SMT) operation up to four threads.
- NetRoute forwarding solution for Longest Prefix Match (LPM).
- NetACL solution for Access Control Lists (ACL).
- Logical Tables provide support for intelligent database management (LTR).
- Key Processing Unit (KPU) for flexible search key construction.
- Statistics and counters.
- Result Buffer provides programmability for flexible routing of search results.
- Range Matching for efficient storage utilization.
- ECC on User Data and Database Array. Parity protection on all embedded memories.
- Background ECC scan for database entries with provision for 2-bit anywhere and 4-bit continuous error detection.
- Forward Error Correction (FEC) when using PAM-4 signaling.
- PCI Express (PCIe) lane.

Figure 1: Functional Block Diagram

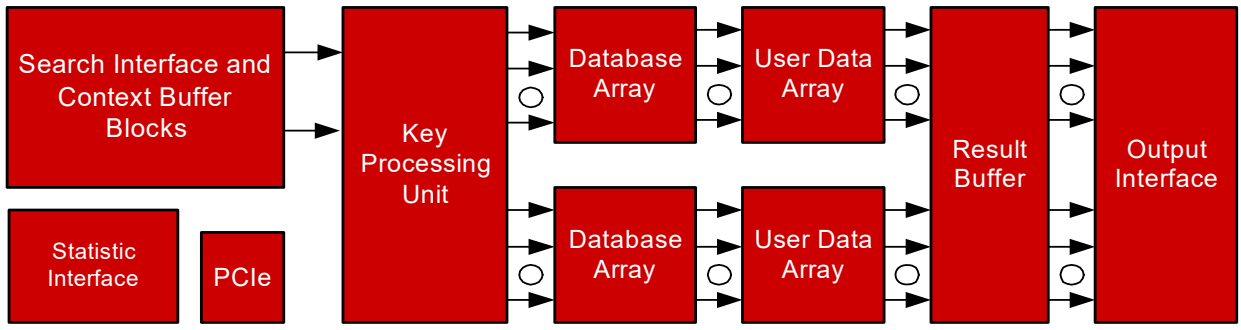


Table of Contents

Chapter 1: About This Document

1.1 Purpose and Audience

The processor performs high-speed operations on large-rule databases for a wide range of telecommunications applications including WAN, MAN, and Enterprise switches and routers. It provides network awareness and enables real-time modifications and updates to the routing configuration, making it ideal for packet classification, policy enforcement, and forwarding.

This document is intended for software, hardware, and application engineers.

1.2 Acronyms and Abbreviations

In most cases, acronyms and abbreviations are defined on first use.

Acronyms and abbreviations in this document are also defined in Appendix A: “Acronyms and Abbreviations,” on page 152.

For a comprehensive list of acronyms and other terms used in Broadcom documents, go to: http://www.broadcom.com/press/glossary.php.

1.3 References

The references in this section may be used in conjunction with this document.

For Broadcom documents, replace the “xx” in the document number with the largest number available in the repository to ensure that you have the most current version of the document.

Document (or Item) Name	Number	Source
Broadcom Items		
[1] KBP SDK Reference Manual	KBP-SDK-SWUM1xx-R	Broadcom CSP
[2] BCM16K Board Design Guidelines	16000-DG1Xx	Broadcom CSP

Chapter 2: Introduction

2.1 Overview

This family of processors addresses next-generation classification needs through high-performance parallel decisions and improved entry storage capabilities. Parallel operations allow the device to reach decision speeds of multiple Billion Decisions Per Second (BDPS). Embedded Error Correction Circuitry (ECC) improves system testability and operational reliability. The key processing unit (KPU) and the context buffer (CB) enable efficient interface transfers with flexible search key construction.

This knowledge-based processor seamlessly connects to Jercicho 2C+ BCM88850, Jericho 2C BCM88800, Qumran 2C, Jericho 2 BCM88690, Jericho+ BCM88680, and Jericho BCM88670.

2.2 Features

2.2.1 General Features

- Dual host enables two host devices to connect to one device.
- Database records 40b: 2048k/1024k.
- Tables width configurable as 80/160/320/480/640 bits.
- User Data Array for associated data, width configurable as 32/64/128/256 bits.
- Dual core DBA/UDA architecture.
- Context Buffer for storing master search keys.
- Up to sixteen parallel searches.
- Simultaneous Multithreading (SMT) operation up to four threads (one per port) for search applications.
- Implements the NetRoute forwarding solution for Longest Prefix Match (LPM).
- NetACL solution for access control list.
- Logical Tables provide support for intelligent database management.
- Key Processing Unit (KPU) for flexible search key construction.
- Result Buffer provides programmable interconnect for flexible routing of search results.
- Range Matching for efficient storage utilization.
- ECC protection for User Data and Database Array. Parity protection on all embedded memories.
- Background ECC scan for database entries.
- Forward Error Correction (FEC) when using PAM-4 signaling.

2.2.2 Electrical Features

- Core clock operating frequency 1000 MHz.
- 32 RX and TX Serial Lanes up to 56.25 Gb/s (PAM-4) for search:
  - In single-host single-port mode, a maximum 16 lanes for search.
  - In single-host dual-port mode, a maximum 16 lanes per port for search.
  - In dual-host quad-port mode, a maximum of 8 lanes per port for search.
  - All search ports must operate at the same SerDes rate
- 16 RX and TX Serial Lanes up to 53.125 Gb/s (PAM-4) for statistics:
  - Four lanes per port for statistics.
  - All statistics ports must operate at the same SerDes rate.
- Miscellaneous CMOS I/Os at 1.8V.
- Core Supply 0.850V controlled through AVS.
- Package 37.5 mm x 37.5 mm FCBGA.

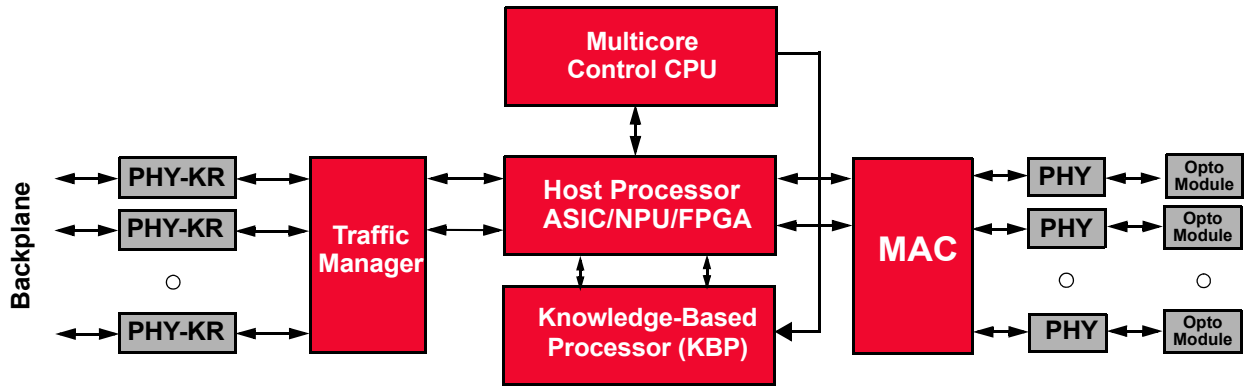
2.3 Applications

The processor can be used in many applications, including;

- IPv4 and IPv6 Packet Classification
- Access Control Lists
- Policy-based Routing and QoS
- IPv4 and IPv6 Longest Prefix Match
- Flow-based Access Control Lists

Figure 2 shows a typical line-card application.

Figure 2: Line Card



2.4 Device Architecture Overview

Figure 3 and Figure 4 shows a block diagram of the processor. The main blocks are described in the following subsections.

Figure 3: Functional Block Diagram (with ILA)

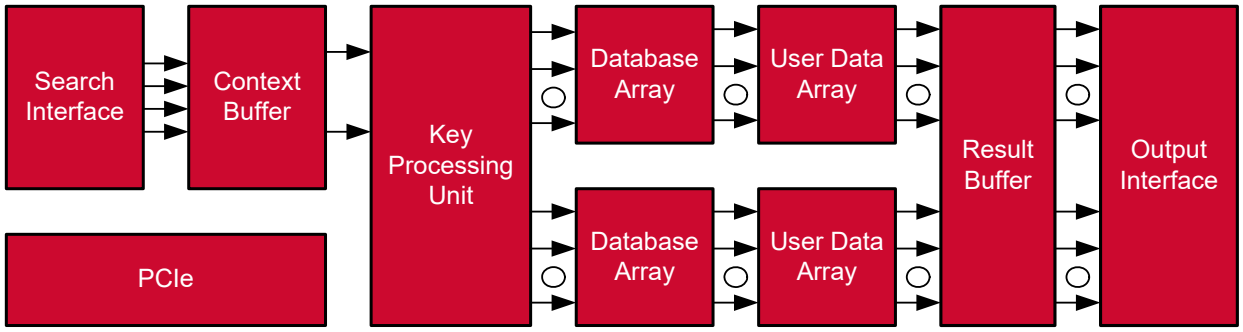
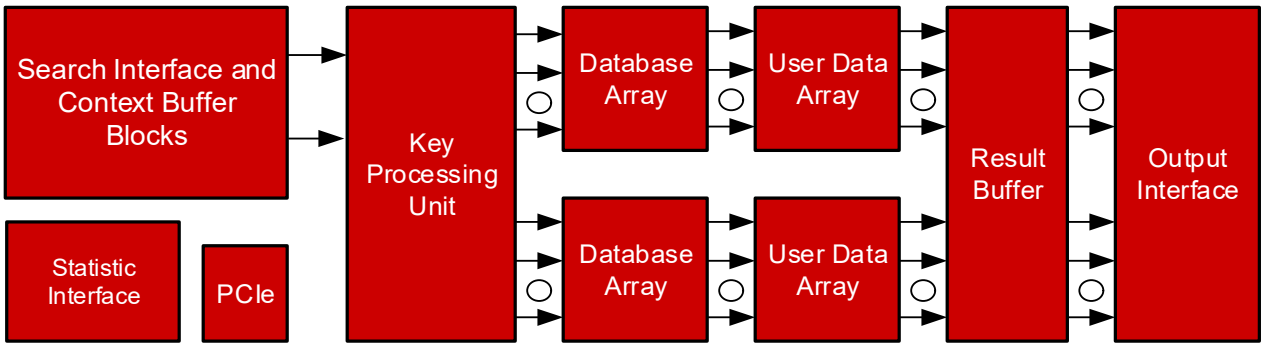


Figure 4: Functional Block Diagram (with Broadcom Switch)



2.4.1 Context Buffer

The context buffer is for loading and storing master search keys from the host processor. All data written from the host processor is stored in the context buffer until overwritten.

See Chapter 3, Context Buffer for additional information.

2.4.2 Key Processing Unit

The key processing unit provides user flexibility in generating search keys for the various search profiles (logical tables) through key manipulation and preprocessing. The key processing unit offers the ability for users to create unique keys for each search operation issued per a compare instruction.

See Chapter 4, Key Processing Unit for additional information.

2.4.3 Database Array

Each database record is composed of 80 X-Y values, one X-Y pair for each bit in the entry. When writing to the processor, users can choose to write in Data Word/Mask Word format or in the X-Y format.

See Chapter 6, Database Architecture Overview for additional information.

2.4.4 User Data Array

The User Data Array (UDA) is a RAM subsystem. The UDA is a shared storage resource with a capacity of 512 Mb per core. The resource consists of 2 x 256 blocks capable of multiple core parallel accesses per clock cycle. ECC protection is provided with 1-bit correct and 2-bit detect.

See Chapter 7, User Data Array for additional information.

2.4.5 Result Buffer

The highest priority match response is sent to the Output Interface for transmission. Result Buffer also allows for post processing of AD (truncating, shifting and padding).

2.4.6 PCIe Interface

The processor has a PCIe 2 port configured for one lane enabling direct connection to control plane. All device functions are supported including device initialization, PIO read/write, table updates, and searches. Multi-channel TX DMA exports statistics and counter data. Multi-channel exports the RX DMA to the control plane processor for updates.

See Chapter 19, Device Configuration for additional details.

2.4.7 Search Data Flow

The processor is designed to offer maximum flexibility for the user to create and apply search keys, resolve search results from database blocks, access user data memory, and output the results.

The processor provides a context buffer for loading and storing a “master search key.” The context buffer is followed by the key processing units, which can generate up to eight per core unique keys, up to 640b in width, from the master search key. The KPU output can be further modified by inserting up to two range information. The range hardware is programmable. The programmable interconnect maps the keys generated by the KPUs to superblocks containing the database entries.

On the result side, a programmable interconnect is available to provide flexible routing and priority encoding of search results. All blocks directed toward the same result output (same parallel search) are used in the determination of the highest priority match (HPM) for that output. The programmable result interconnect allows for the results from the database blocks to be mapped to any one of eight result outputs. This results in a mesh structure between block results to eight per core output slots. These eight per core results can then be sent to the result buffer to be output through the interface or can be used to index the user data memory. The search results and user data can then be sent to the result buffer for output.

Chapter 3: Context Buffer

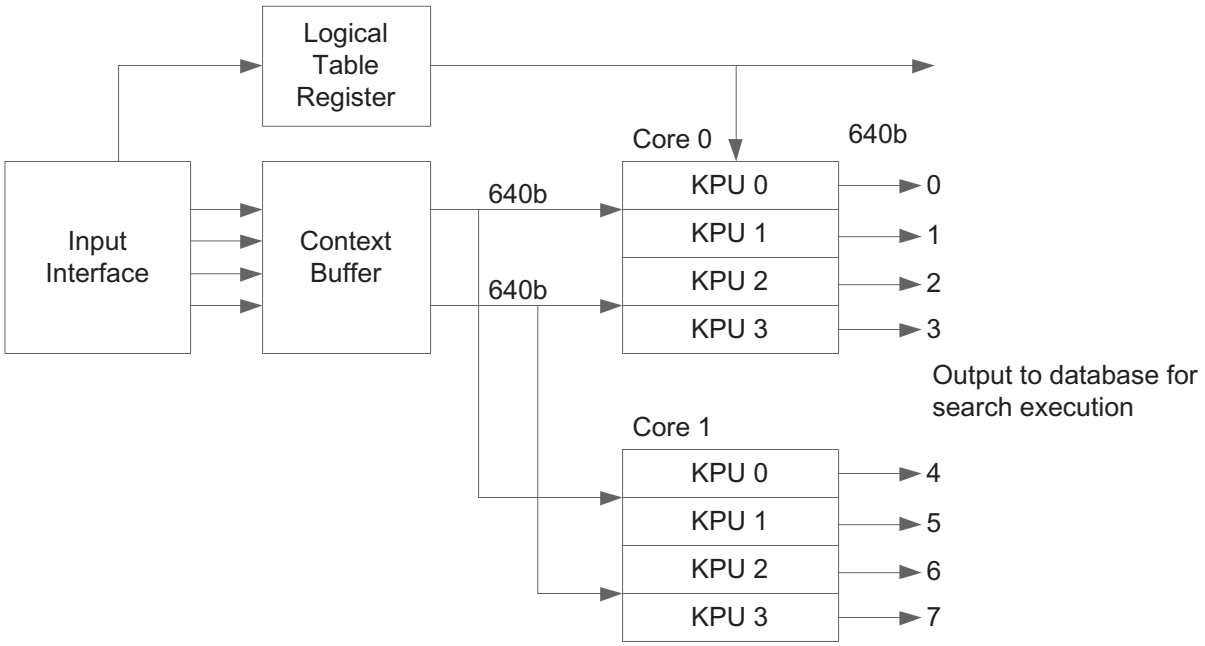
3.1 Overview

The context buffer is for loading and storing master search keys from the host processor. All data written from the host processor is stored in the context buffer until overwritten.

The context buffer is used to store search keys from current or previous instructions for later modification and reissue for flexible multithreaded search key management. During a compare operation, the processor sends 640 bits of search key from the context buffer to the Key Processing Unit (KPU) to be transformed into the search keys to be presented to the database. The 640-bit context is selected by the context buffer address provided in the Interlaken Look-Aside control word during the compare operations.

The context buffer is designed for a multithreaded environment with key persistence. The user can write to any context buffer location at any time. For example, the user can write to address location 20 in one cycle and address location 9 in the next cycle.

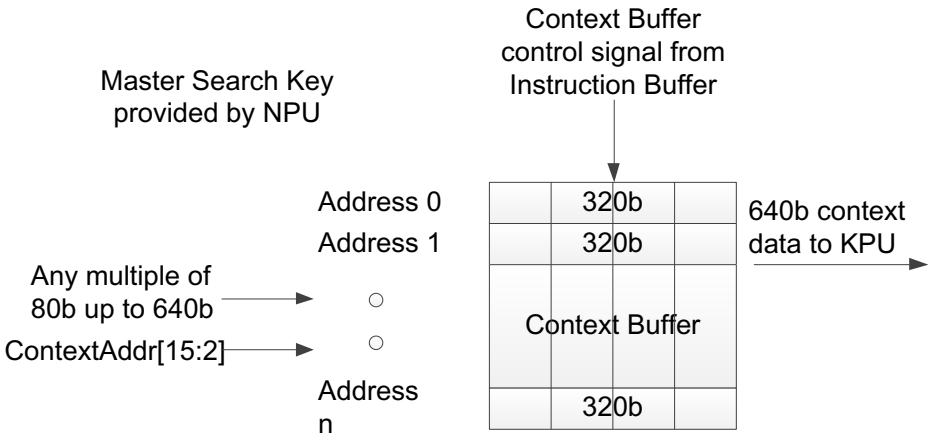
Figure 5: Context Buffer and Key Processing Unit Block Diagram



3.2 Context Buffer Key Transfer to KPU

The Context Buffer address (ContextAddr[15:0]) provided in the control word dictates the storage location of the data inside the context buffer. Each address points to an 80b entry in the context buffer, as shown in Figure 6.

Figure 6: Context Buffer Key Address Format



The context buffer address decoder ignores the least significant three bits of the context buffer address when selecting the 640b of data to be transferred to the key processing unit. For example, if a user executes a compare instruction at Context Address location 11, then the search key data transferred to the KPU come from Context Address locations 8, 9, 10, 11, 12, 13, 14, and 15. Address location 8 will have LSB 80 bits of the 640-bit key. In 320b context configuration, the context buffer address decoder ignores the least significant two bits of the context buffer address when selecting the 320b of data to be transferred to the key processing unit (on the [319:0] bits).

3.3 Context Buffer Addressing

The context buffer is written in multiples of 80b. The number of 80b context buffer locations that are written is determined by the width of the new search key transferred during either compare operations or context buffer write operations. The relationship is as follows:

$$n = \left\lceil \frac{s}{(80)} \right\rceil, n = \text{number of 80b context buffer locations written, } s = \text{ width of the new search key in bits}$$

For example:

- If 80b of the new search key is transferred, then one 80b context buffer entry is updated.
- If 240b of the new search key is transferred, then three 80b context buffer entries are updated.
- If 384b of the new search key is transferred, then five 80b context buffer entries are updated.

If the width of the new search key is not divisible by 80, as shown in the last example, the remaining bits are filled with “don’t care” values. In this scenario, the placement of the search “don't care” values in the context buffer is determined by whether the device is configured for network byte ordering (NBO) or non-network byte ordering (NNBO). If the device is in NBO mode, the “don't care” values are written in the LSB. If the device is in NNBO mode, the “don't care” values are written in the MSB. See “Data Word to Context Buffer Mapping—Non-Network Byte Ordering” and “Data Word to Context Buffer Mapping—Network Byte Ordering” for details.

These random bits must be properly masked using Block Mask Registers or local masks in each database entry.

The context buffer is protected by parity. The instruction “Context Buffer No Write and Compare” will not overwrite existing data in the context buffer (key persistence) enabling the CPU to send 64b junk data. The parity is checked for the targeted 640b context during a compare operation. The compare operation will not be executed if a context buffer error is detected and the response for the compare operation will be flagged with an Error Status bit being set. See “Context Buffer Soft Error Detection” for details.

The context buffer address dictates the storage location of the least significant 80b of the new search key. If the new search key crosses a 640-bit context buffer boundary, only the 640-bit context location that is referred to by the context buffer address is updated. The portion of the new search key that falls outside of the addressed 640-bit context buffer location is lost and the original data retained.

For Table 1, Data A [x] is new data written into the context buffer and Data B [x] and Data C [x] are existing data in the context buffer.

Table 1: 160b Context Buffer Write Operation

Context Buffer Address from Command ContextAddr[15:0]	Context Buffer Address from Command			
	If Context Buffer Address = 0x0	If Context Buffer Address = 0x2	If Context Buffer Address = 0x5	If Context Buffer Address = 0x7
0x0	A [0] 79:0	B [0] 79:0	B [0] 79:0	B [0] 79:0
0x1	A [1] 159:80	B [1] 159:80	B [1] 159:80	B [1] 159:80
0x2	B [2] 239:160	A [0] 79:0	B [2] 239:160	B [2] 239:160
0x3	B [3] 319:240	A [1] 159:80	B [3] 319:240	B [3] 319:240
0x4	B [4] 399:320	B [4] 399:320	B [4] 399:320	B [4] 399:320
0x5	B [5] 479:400	B [5] 479:400	A [0] 79:0	B [5] 479:400
0x6	B [6] 559:480	B [6] 559:480	A [1] 159:80	B [6] 559:480
0x7	B [7] 639:560	B [7] 639:560	B [7] 639:560	A [0] 79:0
0x8	C [0] 79:0	C [0] 79:0	C [0] 79:0	C [0] 79:0
Data written to context buffer and Data transferred to key processing unit	Data A [1:0] (160b) written to context buffer and Data A [1:0] (160b) and Data B [7:2] (480b) transferred to key processing unit	Data A [1:0] (160b) written to context buffer and Data B [1:0] (160b), Data A [1:0] (160b), and Data B [7:4] (320b) transferred to key processing unit	Data A [1:0] (160b) written to context buffer and Data B [4:0] (400b), Data A [1:0] (160b), and Data B [7] (80b) transferred to key processing unit	Data A [0] (80b) written to context buffer and Data A [1] (80b) NOT written to context buffer and Data B [6:0] (560b) and Data A [0] (80b) transferred to key processing unit

3.4 Context Buffer Update

The Context Buffer address specifies the storage location of the least significant 80b of the new search key. If the new search key crosses a 640-bit Context Buffer boundary, only the 640-bit context location, which is referred to by the Context Buffer address, is updated. The portion of the new search key, which falls outside of the addressed 640-bit Context Buffer location, is lost. In the last example in Table 2, the upper most 80 bits of data, ‘S,’ is not written to Context Buffer address 0x8.

Figure 7: Context Buffer Update Example

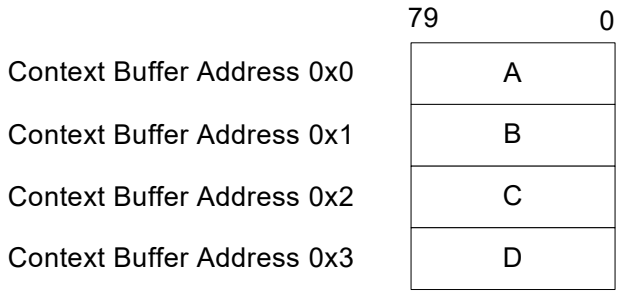


Table 2: Context Buffer Update Example

Context Address	New Search Key	Existing Data Within the Addressed 640-bit Context	New 640-bit Data Within the Addressed 640-bit Context
0x0	QP	HGFEDCBA	HGFEDCQP
0x1	QP	HGFEDCBA	HGFEDQPA
0x3	SRQP	HGFEDCBA	HSRQPCBA
0x5	SRQP	HGFEDCBA	RQPEDCBA

Chapter 4: Key Processing Unit

4.1 Overview

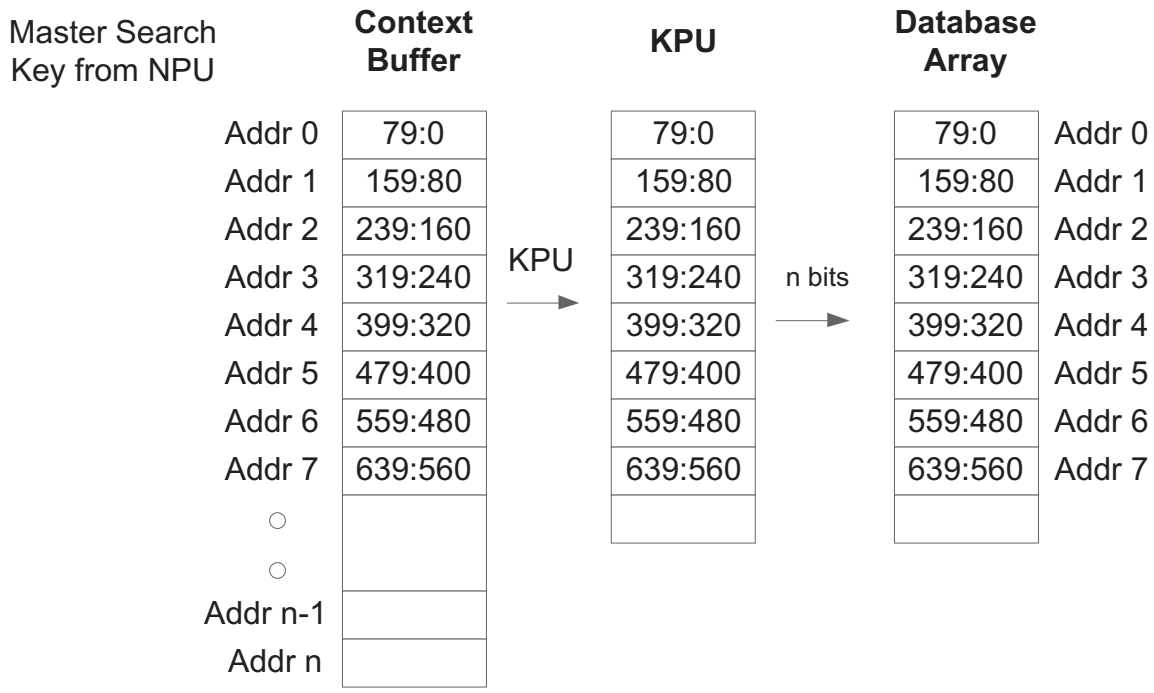
The key processing unit provides user flexibility in generating search keys for the various search profiles (logical tables) through key manipulation and preprocessing. The key processing unit offers the ability for users to create unique keys for each search operation issued per compare instruction. Through register control, the key processing unit assembles eight unique keys up to 640b or four unique key pairs, with each key pair sharing, in unit of 160b, up to 640b of data from the NPU. Byte-level parsing is used across the entire 640b of data.

Up to 15 segments can be used to generate search keys for compare operations. Each KPU gets the 640b master key coming from the NPU in single thread mode, and one of the four 640b master keys coming from the NPU in SMT mode. On the master key, a given KPU can produce one key of up to 640b length and one additional key of up to 320b length. However, as stated above, the combined width of the two keys must not exceed 640b.

The KPU has a Fill-with-Zero feature which, when set, will fill all bytes in a segment with zeros. Enabling Fill-with-Zero for all unused key segments will reduce power consumption.

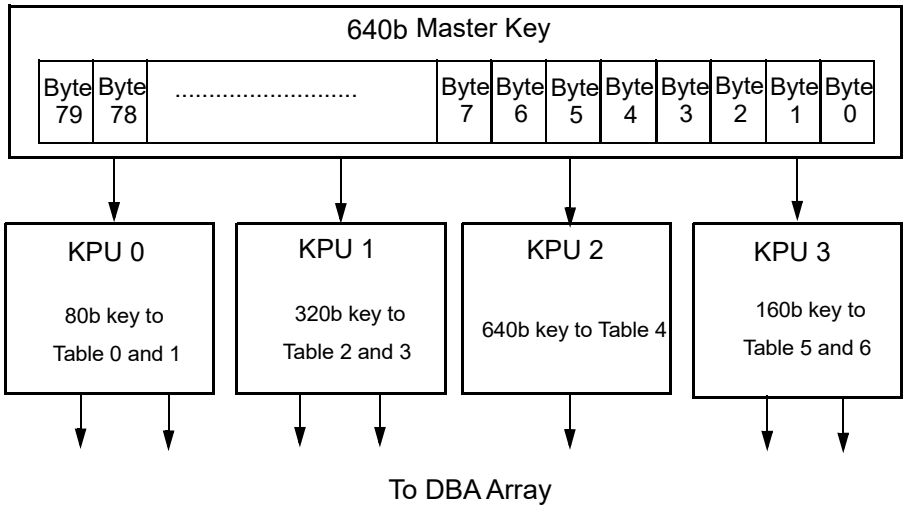
KPU programming is controlled by the KBP SDK. Contact Broadcom application engineering for further details.

Figure 8: Master Key



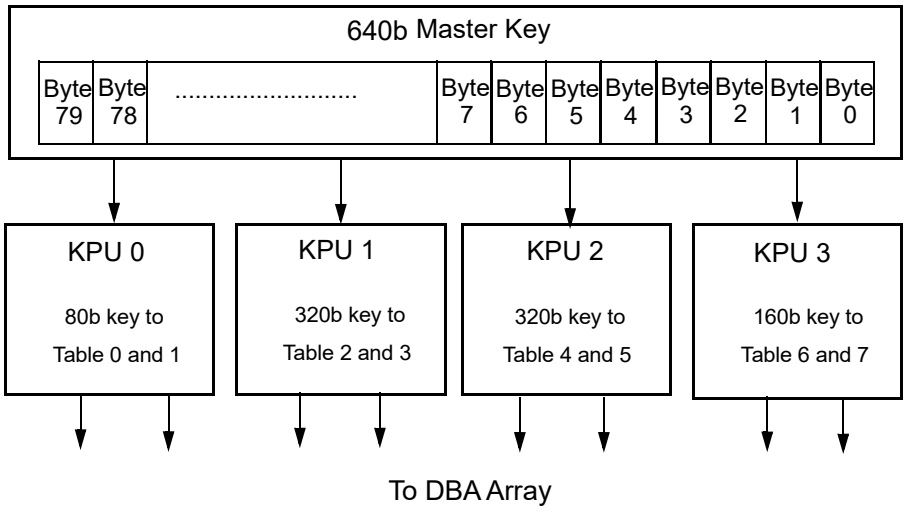
The key processing functionality of the key processing unit is preconfigured using registers, and then activated during compare operations with the referenced logical table register. Figure 9 depicts an example of each key processing unit constructing a unique search key from one distinct context.

Figure 9: Key Processing Unit Diagram—Example 1



Valid Start_Byte values range from 0 to 0x4F. If the Start_Byte fields of all segments are programmed to 0x7F then the search key data is passed through to the database for search operation. Length_Byte values range from one to 16 with 0 = 1 byte, 1 = 2 bytes, … 15 = 16 bytes. If the 15 search key segments are greater than 640b, then only the first 640b is used, and the balance of bits of Most Significant bits greater than 640b is ignored. If all 15 segments do not complete 640b in length, then the remaining balance up to 640b is copied from the master search key.

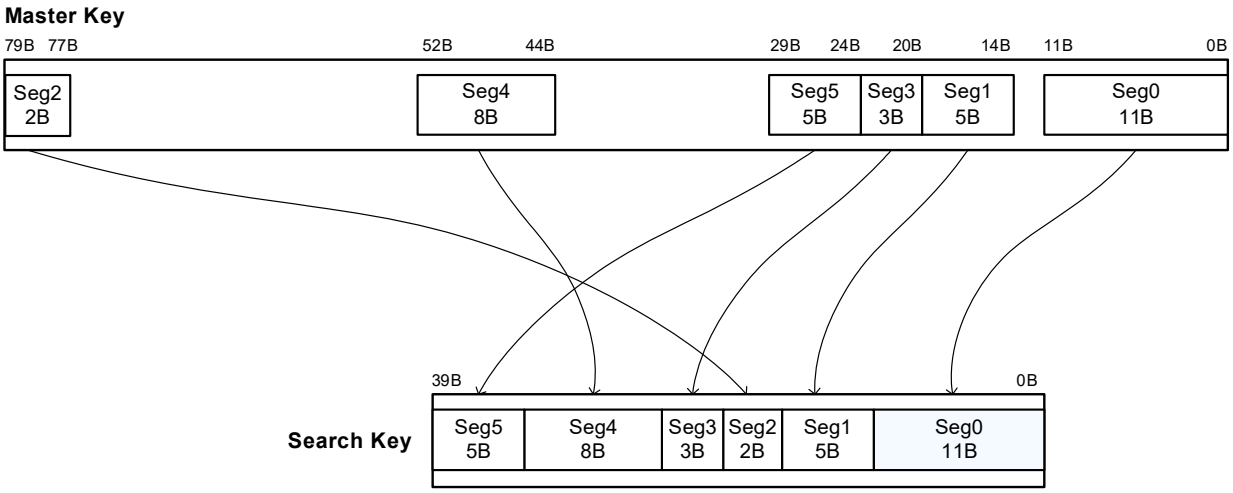
Figure 10: Key Processing Unit Diagram for Eight Search Keys



4.2 Key Processing Unit Byte Parsing

The key processing unit parses search key data from the master key. The byte-level parsing is defined by register settings in the logical table register. The user specifies the Start_Byte and the Length_Byte of each segment. Up to 15 segments can be concatenated to form a search key.

Figure 11: KPU Byte Parsing



Chapter 5: Range Matching

5.1 Overview

Source and Destination port ranges are often used in Access Control List entries (ACLs) to better identify and process network traffic. However, when the port ranges are translated to ternary values, they often require multiple entries, leading to inefficient use of the database and a reduction in the number of user ACLs that can be stored. The worst case number of entries required to store an n-bit range with prefix encoding is (2^n – 2). This inefficiency is compounded when both source and destination port fields include ranges.

Range matching programming is controlled by the KBP SDK. Contact Broadcom application engineering for further details.

5.2 Range Matching Implementation

The processor implements range encoding by utilizing unused bits within each database entry to minimize the expansion in the number of database entries required. This effectively doubles the number of user ACEs with port ranges. The encoding algorithm in the processor is flexible and selects a combination of encoding schemes, with the control plane software selecting the optimal encoding for each range set.

In the example shown in Table 3, six entries are required to store a single ACL containing the range 1 to 14; with only the range field changing between entries. The same ACL port range 1 to 14, using range expansion requires only three entries as shown in Table 4. The same data is now contained in half the number of entries, effectively doubling the storage density, with part of the unused width being used for range encoding.

Table 3: Port Expansion Without Range Matching for Ports 1 to 14

Destination Address	Source Address	16b Port	Ports
A	B	16'b0000_0000_0000_0001	Port 1
A	B	16'b0000_0000_0000_001*	Ports 2 and 3
A	B	16'b0000_0000_0000_01**	Ports 4 to 7
A	B	16'b0000_0000_0000_10**	Ports 8 to 11
A	B	16'b0000_0000_0000_110*	Ports 12 and 13
A	B	16'b0000_0000_0000_1110	Port 14

Table 4: Port Expansion with Range Matching for Ports 1 to 14 with 24b Encoding

Destination Address	Source Address	Used by Range Matching Algorithm	Ports
A	B	32'b0000_0000_0000_0000_0000_0**1_****_****	Ports 1, 2, and 3
A	B	32'b0000_0000_0000_0000_000*_1***_****_****	Ports 4 to 11
A	B	32'b0000_0000_0000_0000_0011_10**_****_****	Ports 12 to 14

The processor has range encoding implemented in hardware. Range encoding requires both hardware and software support. The software operates both in the control plane and the data plane to implement range operations. Various internal registers are used by the software to implement the range operations.

The range encoding can be dynamically applied per LTR. Up to two range fields per search key can be independently selected for range matching, providing efficient range rule compaction. The search profile in the Logical Table register includes range matching registers. See range registers for further details on insertion locations of encoded range fields into parallel search keys and extraction locations of range fields from the master search key.

5.3 Naming Convention

Throughout this document, the following naming conventions are used for range matching.

Table 5: Naming Conventions

Name	Description
Range Encoding	A form of arithmetic coding; a data compression method
Range Field	16b of the port address requiring range encoding
Encoded Range Field	16b, 24b, or 32b output from the range encoding engine

5.4 Considerations

- Range encoding is supported for up to two 16b range fields within master key.
  - As shown in Encoded Range Fields, the 16b range fields within master key must start and end on byte boundaries. The device uses all 16-bits of the range field regardless of the number of bits used to classify the port. Potential unused bits in the range field cannot be allocated for other purposes.
  - As shown in Encoded Range Fields, encoded range fields can be inserted at any byte location within the constructed key (output of KPU).
- Range insertion per constructed key basis:
  - The user specifies the number of bytes of the encoded range field to insert into the constructed key. Two, three, or four byte insertions are allowed. Optimal compression is achieved when all 32b of the encoded range field is inserted. Even with two-byte insertion, the device performs some level of compression.
  - The range function is implemented in parallel with the key processing unit (KPU) function. The user must ensure that there is enough room in the constructed search key to insert the encoded range field.
  - There can be, at maximum, two ranges (maximum of 32b each) inserted in any constructed key (if 640b contains a key pair, then each key would be able to have two ranges).
  - The location of range insertion is programmable. If there are two ranges to be inserted, the second range is inserted after the first one, and the second one occupies higher bit positions than the first one.
  - To accommodate the additional bits associated with the encoded range field, the constructed key must provide spare bits, or remove fields from master key, or increase block size.
  - The encoded range field insertion location is defined by the LTR with possible locations being 160b, 320b, and/or 640b boundary.

5.5 Range Encoding Data Flow

Range Encoding requires both hardware and software support. The software operates both in the control plane and the data plane to implement range operations. Various internal registers are used by the software to implement the range operations.

In the control plane, the Broadcom KBP SDK programs appropriate range values. The KBP SDK has several function calls, including a function call that encodes the 16b Range Field into a 16b, 24b, or 32b encoded range field. The user passes the port ranges to this function, which determines the encoding for most efficient storage. The function returns the range encoded fields to the user to be included in the user's databases.

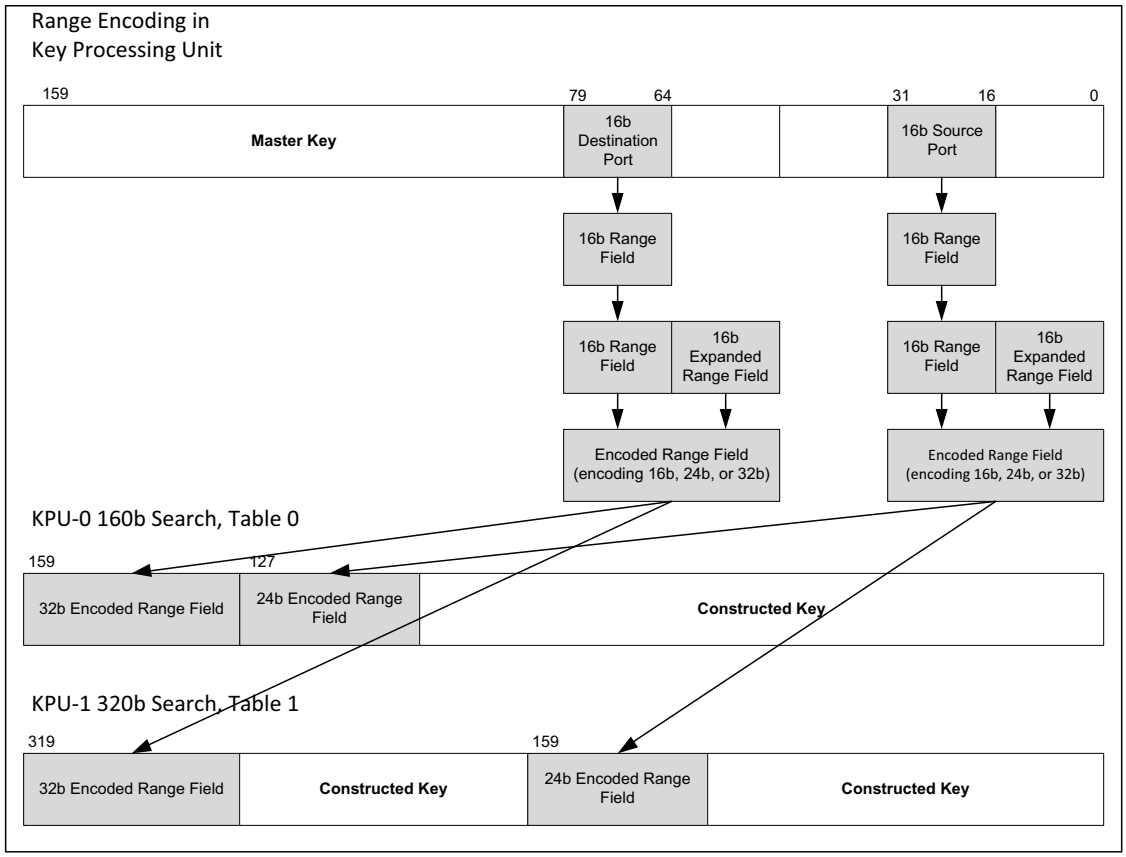
In the data plane, the user writes the input data, (that is, unencoded master search key) to the buffer. The buffer transfers the data to the KPU and range function. The KPU and range function operate in parallel. The device, using the range function, encodes the selected range field(s) and inserts up to 32b of the encoded range field back into the constructed key.

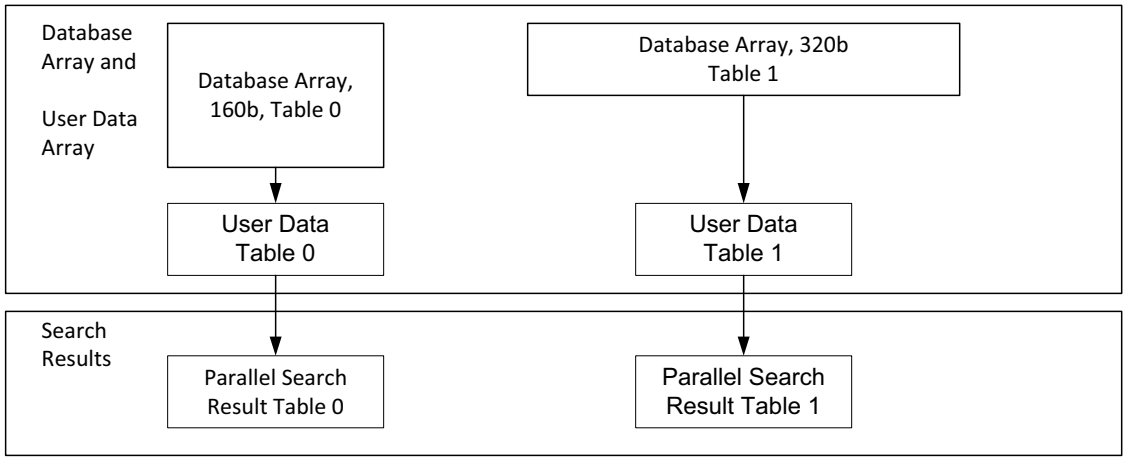
Figure 12 on page 25 shows a typical sequence where an instruction writes 160b of input data into the context buffer. An instruction such as Write Context Buffer and Compare1 transfers 640b of data from the context buffer to the KPU and range function. In this example, only the least significant 160b of the transfer data is used, the most significant 160b of transfer data is not used, and the database blocks targeted by this instruction must be configured to 160b.

5.6 Encoded Range Fields

Figure 12 shows the format of the encoded range fields.

Figure 12: Encoded Range Fields per Key





Chapter 6: Database Architecture Overview

6.1 Database Entry

Each database record is composed of 80 X-Y values, one X-Y pair for each bit in the entry. When writing to the processor, users can choose to write in the traditional Data Word/Mask Word format or in the X-Y format. Internally, the processor always stores in the encoded, X-Y format. Consequently, database read operations return data in the X-Y format only (a single read operation returns an ‘X’ value or a ‘Y’ value, as dictated by the instruction encoding). Table 6 shows the two encoding formats:

Table 6: Data/Mask Encoding

Data	Mask	Data/Mask Description
0	0	Bit matches a ‘0’ only.
0	1	Always hot: bit matches either a ‘0’ or a ‘1’. Represents an ‘X’ (don’t care).
1	0	Bit matches a ‘1’ only.
1	1	Always hot: bit matches either a ‘0’ or a ‘1’. Represents an ‘X’ (don’t care).

Table 7: X-Y Encoding

X	Y	X-Y Description
0	0	Always hot: bit matches either a ‘0’ or a ‘1’. Represents an ‘X’ (don’t care).
0	1	Bit matches a ‘0’ only.
1	0	Bit matches a ‘1’ only.
1	1	Always miss: Forces a mismatch for the compare, unless BMR masks off corresponding bit.
Y = D_b AND M_b		
X = D AND M_b		

ECC scan will detect corruption of up to four consecutive bits and two bits anywhere in the 80b word and correct up to 1b error at the time of the ECC scan. The ECC code is extended to 11 bits, by adding three parity bits to the 8b regular ECC code. The three parity bits are defined as follows:

- P0 (Code[8]) = XOR {D0, D3,...,D78}
- P1 (code[9]) = XOR {D1,D4,..., D79}
- P2 (Code[10]) = XOR {D2, D5,..., D80}

Note that D80 is a valid bit, VBIT.

6.2 Block Description

The database is divided into a number of internal blocks, dependent on the processor capacity. In case of 80b, 160b, and 320b configurations, each array block works independently. In case of wider widths of 480b and 640b, two array blocks are paired. The valid 480b and 640b block pairs are 0 and 4, 1 and 5, 2 and 6, and 3 and 7. Each of these blocks is configurable into the following organizations: 4k x 80b, 2k x 160b, 1k x 320b, and 512 x 640b.

Two neighboring blocks in a super block pair can be combined to create two blocks of one of the following combinations:

- One block 1k x 480b and one block of 1k x 0b,
- One block 1k x 480b and one block 1k x 80b
- One block 1k x 480b and one block 2k x 80b
- One block 1k x 480b and one block 1k x 160b
- Two blocks of 512 x 640b

Additionally, the blocks can be configured to store other widths (example: 8k x 40b) through the use of block masks.

To program a 640b entry into an Array Block (AB), the 320 LS bits are programmed in the first AB of the block pair and then 320 Most Significant bits are programmed in the second AB of the block pair. Array Block numbers are generated in the final result for 640b compares. For entries 0 through 511, the AB number in the result is the same as the first AB in the block pair. For entries 512 to 1023, the AB number in the result is the same as the second AB in the block pair.

When database ECC scan is enabled and the block is enabled, then all database records in that block take part in the scanning process irrespective of their VBIT status, valid or empty.

Table 8: Block Organization

Block	Block Mask Register [319:0]	MSB	Block Mask Register 0			LSB
		MSB	Block Mask Register 1			LSB
		MSB	Block Mask Register 2			LSB
		MSB	Block Mask Register 3			LSB
	Database Block	12’h000	MSB	X	LSB	VBIT
			MSB	Y	LSB	
		12’h001	MSB	X	LSB	VBIT
			MSB	Y	LSB	
			⋮			
		12’hFFE	MSB	X	LSB	VBIT
			MSB	Y	LSB	
		12’hFFF	MSB	X	LSB	VBIT
			MSB	Y	LSB	

Each 80-bit database record has a Validity Bit, VBIT. The status of the VBIT, valid, ‘1’ or invalid, ‘0,’ determines if the record takes part in compare operations. When a VBIT is set to valid ‘1,’ the database record takes part in a compare operation. When the VBIT is cleared to ‘0,’ that record does not take part in a compare operation.

To invalidate or delete a database entry, clear the VBIT to ‘0.’ Only one VBIT needs to be cleared to ‘0’ for 320b, 480b, and 640b records.

Table 9: Invalidate or Delete Database Record

Database Entry	A to H Represent 80-bit Segments A = [79:0], B = [159:80], C = [239:160], to H = [639:560]
80-bit Entry	Must clear VBIT of {A} to ‘0’
160-bit Entry	Must clear one location of VBIT in set {A, B} to ‘0’
320-bit Entry	Must clear one location of VBIT in set {A, B, C, D} to ‘0’
480-bit Entry	Must clear one location of VBIT in set {A, B, C,..., F} to ‘0’
640-bit Entry	Must clear one location of VBIT in set {A, B, C,..., H} to ‘0’

6.2.1 Superblock

The blocks are grouped into superblocks, each superblock is composed of a set of blocks. For each compare cycle, each superblock can offer one search key to its member blocks. All blocks in a single superblock must share the same search key on a given search operation. However, individual blocks within a superblock can have different block masks. Table 10 details the superblock per core assignments for a processor with 2048k 40b or 1024k 80b records.

Table 10: Superblock and Block Assignments per Core

Superblock	Blocks
0	0, 1, 2, and 3
1	4, 5, 6, and 7
:	:
30	120, 121, 122, and 123
31	124, 125, 126, and 127

6.2.2 Block and Superblock Relationship

For each device the associated blocks and superblocks are listed in Table 11.

Table 11: Device Block and Superblock per Core Details

40b Records	Number of Blocks	Block Configuration	Number of Superblocks
1024k	128	4k x 80b, configurable	32
512k	64	4k x 80b, configurable	16

6.2.2.1 Number of Records per Width

For each device, the associated number of records based on block width is listed in Table 12. The native record widths are 80b, 160b, 320b, 480b, or 640b. Other sized records may be stored in a native record width and accessed by the use of block masks. For example, two 40b records may be stored in one 80b record.

Table 12: Record Details per Core

40-bit Records	80-bit Records	160-bit Records	320-bit Records	480-bit Records	640-bit Records
1024k	512k	256k	128k	64k	64k
512k	256k	128k	64k	32k	32k

Table 13: Configuration of 480b Record (Assumes Native 640b Record)

	Option #1	Option #2		Option #3	
Record Widths	480b	480b	80b	480b	160b
No. of Entries	64k	64k	128k	64k	64k

6.3 Error Detection and Correction

This section describes the following types of error detection for the device.

- Database array soft error detection and correction
- Context Buffer soft error detection
- User data array soft error detection and correction
- Logical table register soft error detection

6.3.1 Database Soft Error Detection and Correction

This device implements ECC protection for database soft errors. ECC is calculated and stored automatically by the processor during write operations to database locations. Optionally, Device error GIO_L can be asserted when any database soft error is detected. Additionally, Database Soft Error bit in the Error Status Register is asserted.

Table 14 outlines all of the different combinations of Database Soft Error Scan and ECC Scan that are possible, and how to select each configuration using Device Configuration Register (DCR) options.

Table 14: Database Scan Options

Database Scan Type	DCR[6] Soft Error Scan Enable	DCR[7] Invalidation Enable	DCR[8] Auto-Correction Enable
Database Scan Enabled	1	0: Detect only	0: No correction
		1: Detect and invalidate	1: Correction
Database Scan Disabled	0	X	X

6.3.2 Context Buffer Soft Error Detection

The context buffer is parity protected. The parity for an entry in the context buffer is checked during accesses to that particular entry, except for writes. Context buffer logic automatically initializes all context buffer location after power up.

The Context Buffer parity is checked only during Read, Context Buffer Write and Compare, and Context Buffer No Write and Compare instructions. Context Buffer parity is not checked for Context Buffer Write and Context Buffer Skip and Compare instructions. Since the data in the context buffer is dynamic, there is no corrective action for a memory error.

Context Buffer parity errors can be monitored by observing GIO_L signal. Context Buffer Parity Error field (bit TBD) in the Error Status Register is asserted if the corresponding Error Status Mask register bit is enabled (by setting it to 1'b1). Context Buffer is protected by parity (1b per 16b of data). The error checking happens each master key read and if the error is found, ‘PktErr’ is asserted and the packet error vector bit ‘CB Parity Error during compare’ is set in the response. In addition to this, the error is captured in an error status register which can be enabled to create interrupt on GIO_L pin.

When GIO_L is asserted or an error reply is generated, the Error Status Register should be read to determine the cause of the interrupt. GIO_L is asserted for database and UDA parity error.

Chapter 7: User Data Array

7.1 Overview

The User Data Array (UDA) is a random access memory resource within the processor that can be used for associated data storage as well as other application such as statistics, counters, and algorithmic search. The size of the UDA is device dependent and can be up to 512 Mbits per core. Associated data storage for database search results is the application envisioned for the UDA. The UDA supports flexible access widths, concurrent accesses from multiple request engines, and single-cycle accesses up to 1024 bits. If the search results in a miss, then 0s are returned in the specified AD width. In AD-Only mode, this is the only indication that the search missed. As such, it is not recommended to have all 0s as valid associated data.

The User Data Array also supports a second level of associated data indirection. Contact your local application engineer for additional details.

7.2 Features

The UDA provides the following features:

- Total capacity: 512 Mb/256 Mb per core, with option to power-off any segment of memory in unit of 8 Mb.
- UDA blocks: 256/128 per core.
- Block width configurable: 32b, 64b, 128b, 256b.
  - Each block is independently configurable.
- Parallelism: Up to eight per core associated data per clock.
  - Each associated data can be a different size.
- ECC protection.
- ECC scheme: 1-bit detect and correct, 2-bit detect.

Table 15: Database Array Capacity per Core

Database Array per Core	Records
40 Mb	1024k 40b database records
20 Mb	512k 40b database records

Table 16: User Data Array Capacity Per Core

User Data Array (Mb)	32b Associate Data Records
512 Mb	16M
256 Mb	8M

7.3 UDA Read/Write Accesses

Each address location in the UDA memory is accessed as a 32b value. As a result, all write and read operations are done on 32-bit locations. The read/write address range of the device is given below.

The UDA also supports 64b read/write access. However, addressing is still at 32b word level. In other words, while doing 64b read or write, the LSb of the UDA address is expected to be 0.

Table 17: UDA Address Range Based on Device Size per Core

Device Size	Address Range for 32b Access
512 Mbit	0x0 to 0xFF_FFFF
256 Mbit	0x0 to 0x7F_FFFF

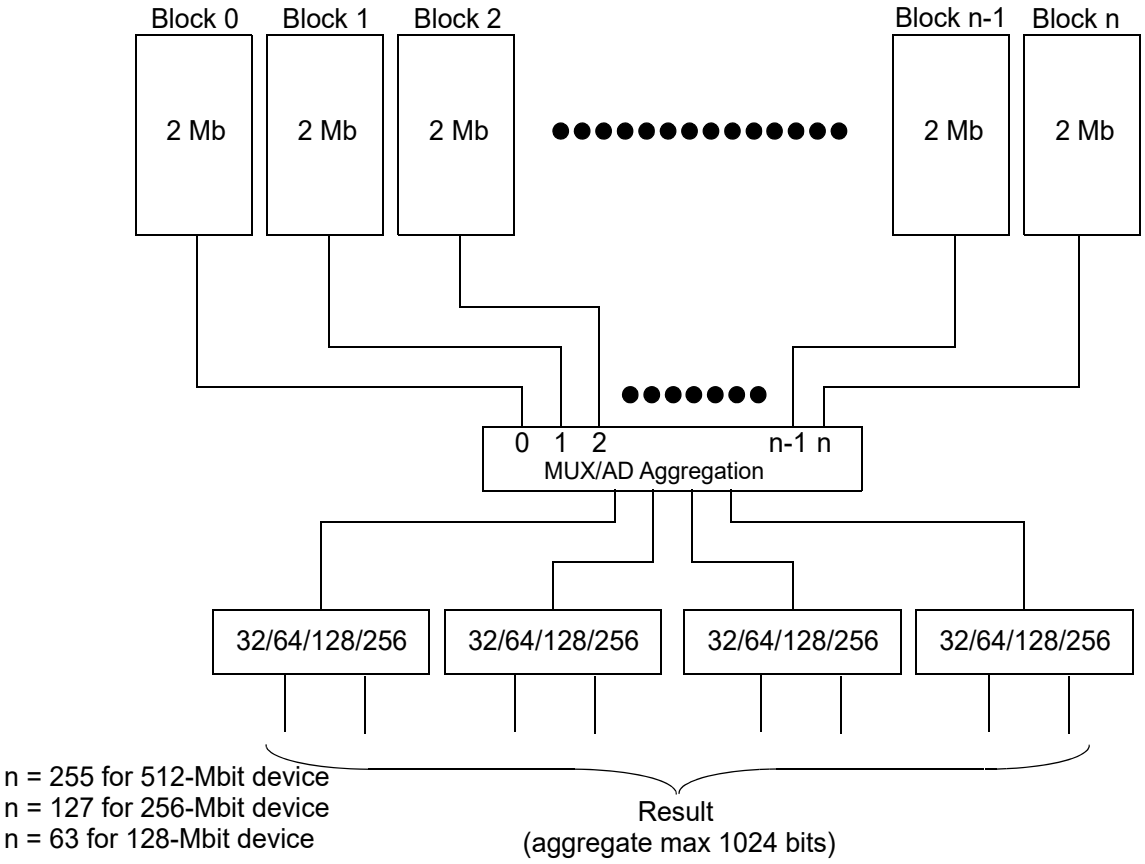
7.4 UDA Organization for Associated Data

The user data array is organized into 2 x 256 blocks, 2Mb each. Each block has 2 Mb of capacity and can be configured in widths of 32, 64, 128, or 256 bits. Memory update accesses are always 32 or 64 bits. Each block is read-accessible independent of the other blocks. A total of eight accesses are allowed at the same time. Only two parallel accesses are allowed per 32 Mb and the two parallel access cannot be the same 2 Mb block. Each parallel access can have a different width, as long as the aggregate data size does not exceed 1024 bits.

For power savings, UDA blocks may be powered down in units of 4 consecutive blocks. See UDA configurations registers for further details.

If there are more than four associated data accesses, then the max AD size for result 0 and 4 each is 128b or less. The same rule applies for result pairs 1 and 5, 2 and 6, and 3 and 7.

Figure 13: UDA Organization per Core for Associated Data Lookup



Each UDA block can be dynamically accessed in multiple bit widths ranging from 32b to 256b. This allows the user to assign associated data of various widths on a per database array block basis. This gives flexibility to the table management software to reassign blocks to different tables as the tables grow or shrink in real-time. As a result, logical tables may be mapped onto noncontiguous blocks within the UDA blocks.

7.5 Database to UDA Referencing

The UDA address generation is controlled by the application using registers, where the base addresses and shift parameters are configured on a per Database Array block basis.

The translation from search index results to UDA memory is done on a per database block basis. In particular, each Database Array block has a user-configured base address (BA) as well as shift value and shift direction, as shown below. The BA represents the UDA address corresponding to the first entry in the searched Database Array block. The shift represents the binary division of the DBA address so as to normalize it into continuously increasing 32b AD words. The correspondence between database array block widths and UDA data-widths is shown below, along with the shift parameters. The shift index direction and shift index value must be programmed by the user.

In the examples below, the 15b base address (BA) represents the UDA address where the logical table to which the current database block belongs is referenced from. One base address is available for each database array block. The total number of 32b locations in a 512-Mbit UDA is 16M, which requires a 24b address. The total number of 640b address locations in a database array block is 512, thereby, requiring a 9b address. Thus, the BA bits needed for this case are 15b (24b – 9b).

The address translation is done for each search result after the hit index is derived. Both the index and the derived UDA data are output to support different applications that may require either or both outputs. Additional indirection is available to dynamically move associated data block from one UDA block to another.

7.5.1 UDA Base Address Translation

The following are the equations to derive the base address:

BA (base address) for AB (as defined in equation): (FLTA >> 9) >> n, where (2^n = (ADwidth/32) (1)

BA for the next AB: (ADwidthPrevBlock/ADwidthNextBlock) × (BA for previous AB+ (640/ABwidth)) (2)

FLTA (first level translated address)—physical UDA address derived from the DBA index.

A representation of the base address, in terms of FLTA is given below:

Base address (BA) calculation:

FLTA = ({BA, 9'b0} + ShiftIndex) << 0, if AD = 32b
FLTA = ({BA, 9'b0} + ShiftIndex) << 1, if AD = 64b
FLTA = ({BA, 9'b0} + ShiftIndex) << 2, if AD = 128b
FLTA = ({BA, 9'b0} + ShiftIndex) << 3, if AD = 256b

The following subsection provides an example of the base address calculation.

The User Data Array also supports a second level of associated data indirection. Contact your local application engineer for additional details.

Two database array blocks are configured to be 320b width. Assume these blocks are called 0x15 and 0x16. The database address (in unit of 80b words) range of the entries in these blocks is 0x15000 to 0x16FFF. The total number of database entries in the database for these two blocks is 2K 320-bit entries. Thus, 2K entries are required in the UDA to correspond to the 2K database entries. Assume the associated data for these database entries is 64b. The translated address per block is generated by the following:

Since the DBA width is 320, there are 1024 entries per DBA block.

In this example, we are considering that the first DBA entry corresponds to UDM8 (memory is at 64Mb), this means that the address of the UDA entry in unit of 32b words is 0x200000. The table represents the address of the UDA entry in terms of UDA size.

From Table 18 on page 35, FLTA = 0x200000, FLTA[21] = 1b1, BA[11] = 1b1 or BA = 0x800.

- Pre-shift Index of the first entry in 320b database block 0x15 = 0x000.
- Pre-shift Index of the last entry in 320b database block 0x15 = 0xFFC.
- Shifted Index for the first entry in database block 0x15 = 0x000.
- Shifted Index for the last entry in database block 0x15 = 0x3FF (shifted right by 2, from table) – DB[11:2].
- FLTA, start = 0x200000 in terms of AD size.
- FLTA, end = 0x2007FE in terms of AD size.

For the next DB, since contiguous addressing is considered, the base address is 0x802.

Use equation 2: (64/64) × (0x800 + 640/320) = 0x802 or incrementing BA[1]/FLTA[11] derives the same result:

- Pre-shift Index for the first entry in 320b database block 0x16 = 0x000.
- Pre-shift Index for the last entry in 320b database block 0x16 = 0xFFC.
- Shifted Index for the first entry in database block 0x16 = 0x000.
- Shifted Index for the last entry in database block 0x16 = 0x3FF (shifted right by 2, from table) – DB[11:2].
- FLTA, start = 0x200800 in terms of AD size.
- FLTA, end = 0x200FFE in terms of AD size.

Table 18: Database Address to First Level Translation Address (FLTA)

AB Width	AD Size	AB shift dir	FTA23	FTA22	FTA21	FTA20	FTA19	FTA18	FTA17	FTA16	FTA15	FTA14	FTA13	FTA12	FTA11	FTA10	FTA9	FTA8	FTA7	FTA6	FTA5	FTA4	FTA3	FTA2	FTA1	FTA0
640	32	R3	BA14	BA13	BA12	BA11	BA10	BA9	BA8	BA7	BA6	BA5	BA4	BA3	BA2	BA1	BA0	DB11	DB10	DB9	DB8	DB7	DB6	DB5	DB4	DB3
320	32	R2	BA14	BA13	BA12	BA11	BA10	BA9	BA8	BA7	BA6	BA5	BA4	BA3	BA2	BA1	DB11	DB10	DB9	DB8	DB7	DB6	DB5	DB4	DB3	DB2
160	32	R1	BA14	BA13	BA12	BA11	BA10	BA9	BA8	BA7	BA6	BA5	BA4	BA3	BA2	DB11	DB10	DB9	DB8	DB7	DB6	DB5	DB4	DB3	DB2	DB1
80	32	R0	BA14	BA13	BA12	BA11	BA10	BA9	BA8	BA7	BA6	BA5	BA4	BA3	DB11	DB10	DB9	DB8	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
640	64	R3	BA13	BA12	BA11	BA10	BA9	BA8	BA7	BA6	BA5	BA4	BA3	BA2	BA1	BA0	DB11	DB10	DB9	DB8	DB7	DB6	DB5	DB4	DB3	1'b0
320	64	R2	BA13	BA12	BA11	BA10	BA9	BA8	BA7	BA6	BA5	BA4	BA3	BA2	BA1	DB11	DB10	DB9	DB8	DB7	DB6	DB5	DB4	DB3	DB2	1'b0
160	64	R1	BA13	BA12	BA11	BA10	BA9	BA8	BA7	BA6	BA5	BA4	BA3	BA2	DB11	DB10	DB9	DB8	DB7	DB6	DB5	DB4	DB3	DB2	DB1	1'b0
80	64	R0	BA13	BA12	BA11	BA10	BA9	BA8	BA7	BA6	BA5	BA4	BA3	DB11	DB10	DB9	DB8	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0	1'b0
640	128	R3	BA12	BA11	BA10	BA9	BA8	BA7	BA6	BA5	BA4	BA3	BA2	BA1	BA0	DB11	DB10	DB9	DB8	DB7	DB6	DB5	DB4	DB3	1'b0	1'b0
320	128	R2	BA12	BA11	BA10	BA9	BA8	BA7	BA6	BA5	BA4	BA3	BA2	BA1	DB11	DB10	DB9	DB8	DB7	DB6	DB5	DB4	DB3	DB2	1'b0	1'b0
160	128	R1	BA12	BA11	BA10	BA9	BA8	BA7	BA6	BA5	BA4	BA3	BA2	DB11	DB10	DB9	DB8	DB7	DB6	DB5	DB4	DB3	DB2	DB1	1'b0	1'b0
80	128	R0	BA12	BA11	BA10	BA9	BA8	BA7	BA6	BA5	BA4	BA3	DB11	DB10	DB9	DB8	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0	1'b0	1'b0
640	256	R3	BA11	BA10	BA9	BA8	BA7	BA6	BA5	BA4	BA3	BA2	BA1	BA0	DB11	DB10	DB9	DB8	DB7	DB6	DB5	DB4	DB3	1'b0	1'b0	1'b0
320	256	R2	BA11	BA10	BA9	BA8	BA7	BA6	BA5	BA4	BA3	BA2	BA1	DB11	DB10	DB9	DB8	DB7	DB6	DB5	DB4	DB3	DB2	1'b0	1'b0	1'b0
160	256	R1	BA11	BA10	BA9	BA8	BA7	BA6	BA5	BA4	BA3	BA2	DB11	DB10	DB9	DB8	DB7	DB6	DB5	DB4	DB3	DB2	DB1	1'b0	1'b0	1'b0
80	256	R0	BA11	BA10	BA9	BA8	BA7	BA6	BA5	BA4	BA3	DB11	DB10	DB9	DB8	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0	1'b0	1'b0	1'b0

7.5.2 Associated Data Response Formats

Associated Data is returned in one of two formats. In default operation, the Index and the Associated Data are returned for each search. If the search result is a miss, then zeroes are returned in the specified AD width instead.

Alternatively, the device can be placed in ADorIndex mode by setting bits 47:40 of AD (Associated Data) Control Logical Table Register. Bit 40 works for Result0 and Bit 47 works for Result7. If the bit is set to 1'b1, either index or AD but not both are sent as part of result. In other words, if AD is enabled for a result, index (and match flags) will not be sent as part of that result but if AD is not enabled, index (and match flags) would be sent. Also, if only AD is being returned and result is a miss, then zeroes in the specified AD width are retuned instead. In this case, all zero should not be a valid AD value since all 0 would be indication of a miss.

7.5.2.1 Example HPM Index and Associated Data

This scenario has the following parameters:

- HPM index + associated data
- Four results:
  - Result 0 has 32b AD
  - Result 1 has 64b AD
  - Result 2 has 128b AD
  - Result 3 has 256b AD

Figure 14: Example HPM Index and Associated Data

<table>
<tr><td colspan="2">Control Word</td></tr>
<tr><td>Index 0</td><td>Assoc Data 0</td></tr>
<tr><td>Index 1</td><td>Assoc Data 1</td></tr>
<tr><td>Assoc Data 1</td><td>Index 2</td></tr>
<tr><td colspan="2">Assoc Data 2</td></tr>
<tr><td colspan="2">Assoc Data 2</td></tr>
<tr><td>Index 3</td><td>Assoc Data 3</td></tr>
<tr><td colspan="2">Assoc Data 3</td></tr>
<tr><td colspan="2">Assoc Data 3</td></tr>
<tr><td colspan="2">Assoc Data 3</td></tr>
<tr><td>Assoc Data 3</td><td>0</td></tr>
</table>

7.5.2.2 Example Associated Data Only

This example shows four results are programmed in ADorIndex mode.

- Associated data only
- Four results:
  - Result 0 has 32b AD
  - Result 1 has 64b AD
  - Result 2 has 128b AD
  - Result 3 has 256b AD

Figure 15: Example Associated Data Only

<table>
<tr><td colspan="2">Control Word</td></tr>
<tr><td>Assoc Data 0</td><td>Assoc Data 1</td></tr>
<tr><td>Assoc Data 1</td><td>Assoc Data 2</td></tr>
<tr><td colspan="2">Assoc Data 2</td></tr>
<tr><td>Assoc Data 2</td><td>Assoc Data 3</td></tr>
<tr><td colspan="2">Assoc Data 3</td></tr>
<tr><td colspan="2">Assoc Data 3</td></tr>
<tr><td colspan="2">Assoc Data 3</td></tr>
<tr><td>Assoc Data 3</td><td>0</td></tr>
</table>

7.5.2.3 Example HPM Index and Associated Data

This example has the following parameters:

- HPM index + associated data
- Two results:
  - Result 0 has 32b AD
  - Result 1 does not have AD

Figure 16: HPM Index and Associated Data

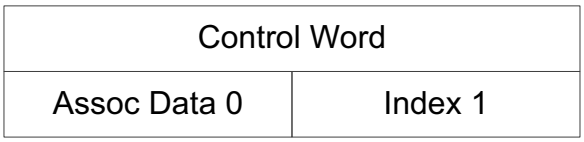
<table>
<tr><td colspan="2">Control Word</td></tr>
<tr><td>Index 0</td><td>Assoc Data 0</td></tr>
<tr><td>Index 1</td><td>0</td></tr>
</table>

7.5.3 Example HPM Index or Associated Data

This is the same as Example 2a with the exception that both results are programmed in ADorIndex mode.

- HPM index or associated data
- Two results:
  - Result 0 has 32b AD
  - Result 1 does not have AD

Figure 17: Example HPM Index or Associated Data



NOTE: If Burst Short is 16 bytes, then an extra data word is included in the response packet.

Chapter 8: Simultaneous Multithreading (SMT)

8.1 Overview

The device, by default, is enabled for simultaneous multithreading (SMT) as long as compares use LTR from four groups (that is, 0 to 31, 32 to 63, 64 to 95, and 96 to 127). SMT allows for higher rates of compare instructions as compared to single-thread. The device, on reset, gets configured to accept SMT instructions belonging to LTR groups.

This section discusses the various aspects of single-thread and multithread modes.

8.2 Simultaneous Multithread Mode

In SMT mode, the device allows for the concurrent processing of four master keys. The device selects the four master keys based on resource conflicts between the four selected top-of-queue instructions.

The device implements four queues per port. These four queues are based on top two bits of LTR index ([6:5]). Thus, the instructions are selected out of four (one-port usage) or eight (two-port usage) instructions. The various resources (LTR, KPU, PCM, Database Array, and UDA) as shown in Figure 18 are shared in non-symmetric ways. For example, when two instructions are issued, KPUs could be divided as two KPUs per thread, PCM could be divided on 1/3 to 2/3 basis and DBA/UDA could be divided on 1/4 to 3/4 basis. The four partitions are called Banks.

Figure 18: Single and Dual Port with Single Host

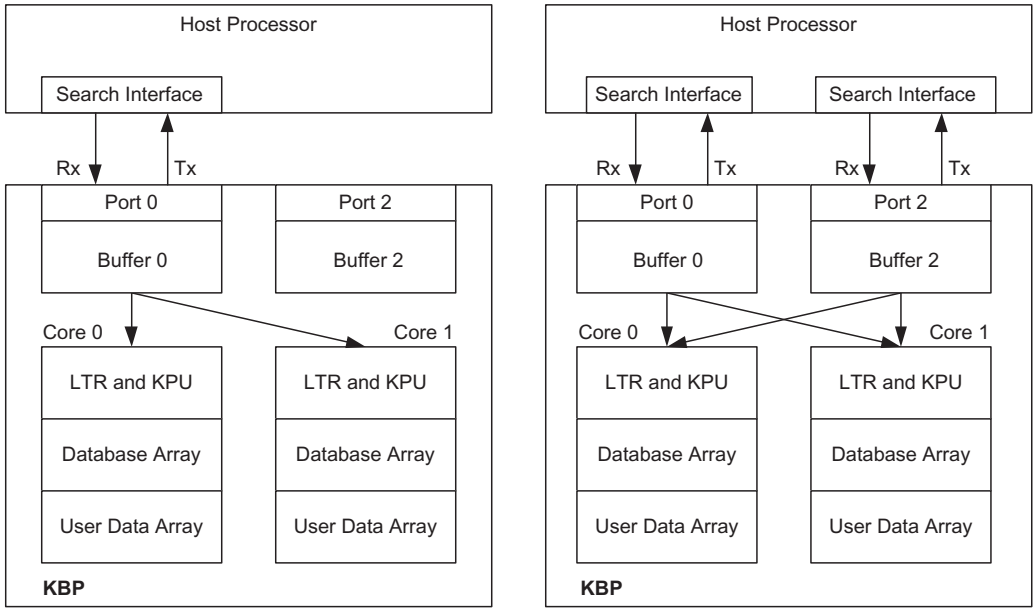
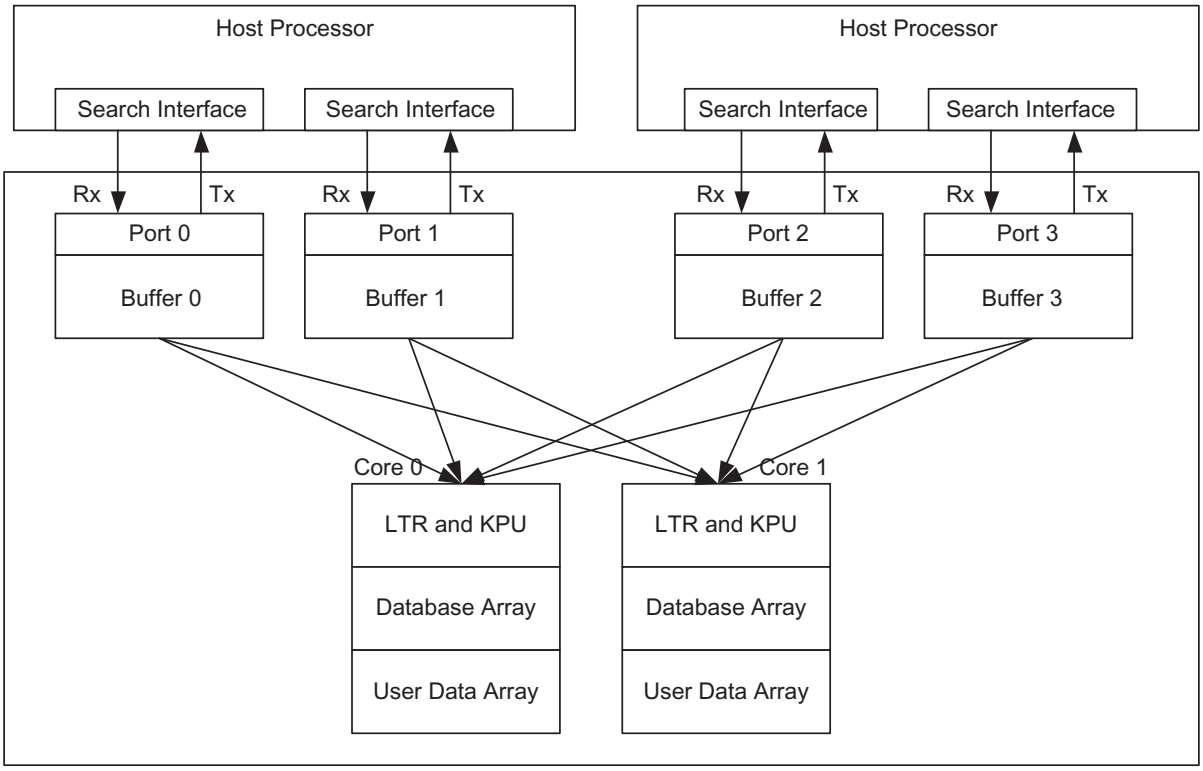


Figure 19: Quad Port with Dual Host



8.2.1 Assignment of Resources in SMT Mode

In SMT mode device resources can be assigned to one of the two threads as shown in Table 19. Table 19 also shows resource assignments for the BCM15K and not the BCM16K. Contact your Broadcom FAE and AE for additional information.

Table 19: SMT Resource Assignment for BCM15K

Resource	Thread	Min. No.	Max. No.	Unit	Total Range
LTR	0	32	96	32	LTR [0:95]
	1	32	96	32	LTR [32:127]
KPU	0	1	3	1	KPU [0:2]
	1	1	3	1	KPU [1:3]
DBA Superblock (SB)	0	1	32	1	SB [0:15]
	1	1	32	1	SB [16:31]
UDA Superblock	0	1	255	1	UDA [0:96]
	1	1	255	1	UDA [95:255]

NOTE: Contact Your Broadcom AE for additional information.

8.2.2 Database Array Bank Boundaries

In SMT mode, the user sets the bank boundary by defining a set of super-blocks that can only be searched by a given thread. However, for keys that are 480b and wider, pairs of super blocks must be assigned to the same bank. Read/Write access is not restricted.

When accessing database array banks:

- Search operations in each bank are issued simultaneously
- Search operations must not cross database array bank boundary
- Search operations may enable any number of database array blocks as in normal operation

8.2.3 UDA Bank Boundaries

In SMT mode, the user sets the user data array bank boundary by defining a set of User Data Super Block where each 2 Mb can only be accessed by a given thread. Read/Write access is not restricted.

When accessing UDA banks:

- Access operations in each bank are issued simultaneously.
- Access operations that specify UDA Super Block must not cross bank boundary.
- Access operations may enable any number of UDA Super Block as in normal operation.

Chapter 9: Pin Descriptions

9.1 Overview

Providing power to the ESD diode can potentially damage the chip. To prevent turning on the ESD diode, do not apply voltage to all signal pins before the power supplies have ramped.

The following tables list the processor pins and their definitions.

- Table 20, Clock, Configuration, and Reset Pin Descriptions
- Table 21, Interface Transmit, Receive, Clock, and Control Pin Descriptions
- Table 22, JTAG Pin Descriptions
- Table 23, Miscellaneous Pin Descriptions
- Table 24, Test, Power, and Ground Pin Descriptions

For all static low and high inputs, an external 4.7 kΩ pull-up or pull-down resistor is recommended. Do not directly connect any inputs to ground or VDDQ.

9.2 Pin Descriptions

Table 20: Clock, Configuration, and Reset Pin Descriptions

Parameter	Symbol	Type	No. Pins	Description
Core Logic Reset	CRST_L	I	1	CRST_L is an active low asynchronous input that resets the core logic of the processor. The reset operation initializes the control logic. Connection to this input pin is required. Refer to the Board Design Guideline for recommendations. This pin does not have ODT.
Management Data I/O Clock	MDC	I	1	The Management Data I/O clock input is the clock for the MDIO port, compatible with IEEE 802.3ae, only Clause 45, not Clause 22. Connection to this input pin is required. This pin does not have ODT.
Management Data Input Output	MDIO	IO	1	The Management Data Input Output pin is compatible with IEEE 802.3ae, Clause 45. This pin is open drain. Externally pull-up this pin to VDD18 via a 1 kΩ resistor. Connection to this input/output pin is required. This pin does not have ODT.
Management Port ID	MPID[4:0]	I	5	The Management Port ID input pins set the management port address for the device. The Management Port ID must match the 5-bit management port address to select this device. Valid MPIDs: 0 to 23. Do not use MPID 24 to 31. For the selected MPID N, this device uses MPID numbers N through N + 8. Connection to these input pins is required. These pins do not have ODT.
PCIe Reset	PERST_L	I	1	The PCIe Reset is an asynchronous active low input that provides the hardware reset used to initialize the PCIe port. Connection to this input pin is required. This pin does not have ODT.
I^2C Data Line	SDA	Open Drain	1	I^2C interface pin. One external 1 kΩ resistor to VDD18 is required for this signal.
I^2C Clock Line	SCL	I	1	I^2C interface pin. One external 1 kΩ resistor to VDD18 is required for this signal.
SerDes Ref Select	SREF	I	1	SerDes reference clock select. 0 = Reserved and 1 = 156.25 MHz. This pin must be pulled high using 1 kΩ resistor to VDD18 on the board. Connection to this input pin is required. This pin does not have ODT.
System Power On Reset	SRST_L	I	1	The System Power on Reset is an asynchronous active low input that provides the hardware reset used to initialize the processor. Connection to this input pin is required. Refer to the Board Design Guideline for recommendations. This pin does not have ODT.

Table 21: Interface Transmit, Receive, Clock, and Control Pin Descriptions

Parameter	Symbol	Type	No. Pins	Description
SerDes Reference Clock	SREFCLK0_P SREFCLK0_N SREFCLK1_P SREFCLK1_N	I	4	The SerDes Reference Clock differential pair is used to drive the RX and TX serial lanes. SREFCLK_P and SREFCLK_N must be driven by a 156.25 MHz clock generator. These pins contain 100Ω differential on-die termination (ODT) to generate its own bias voltage. These pins require AC coupling. Connection to these input pins is required.

Table 21: Interface Transmit, Receive, Clock, and Control Pin Descriptions (Continued.)

Parameter	Symbol	Type	No. Pins	Description
PCIe Reference Clock	PCREFCLK_P PCREFCLK_N	I	2	The PCIe Reference Clock operates at 100 MHz and are typically driven by a clock generator. PCIe Reference Clock requires external AC coupling, termination and bias circuit. Refer to the Board Design Guideline for details.
Core Reference Clock	CREFCLK_P CREFCLK_N	I	2	The Core Reference Clock differential pair is used as a reference for the database core. CREFCLK_P and CREFCLK_N must be driven by 156.25 MHz clock generator. These pins contain 100Ω differential on-die termination to generate its own bias voltage. These pins require AC coupling. The maximum frequency for this clock is 450 MHz. Connection to these input pins is required.
PCIe Receive Data	PCRXDATA_P PCRXDATA_N	I/O	2	P/N differential pairs for the PCIe interface. Each receiver pair has a 100Ω differential ODT.
PCIe Transmit Data	PCTXDATA_P PCTXDATA_N	I/O	2	P/N differential pairs for the PCIe interface.
Receive Data	RXDATA_P [47:0] RXDATA_N [47:0]	I	96	Primary Receive Data are P/N differential pairs for the input serial receivers. Unused signal pairs must be left unconnected and configured to be turned off. RXDATA_P [31:0] for search RXDATA_N [31:0] for search RXDATA_P [47:32] for statistics RXDATA_N [47:32] for statistics See recommended RX/TX Data lane assignments Jericho 2C+ (Table 44 and Table 45), Jericho+ BCM88680 (Table 50), Jericho 2 BCM88690 (Table 51), and Jericho 2C BCM88800 (Table 52). RXDATA_P and RXDATA_N must be connected to the host processor's transmitters. Each receiver pair has a 100Ω differential ODT.
Transmit Data	TXDATA_P [47:0] TXDATA_N [47:0]	O	96	Primary Transmit Data are P/N differential pairs for the output serial transmitters. Unused signal pairs must be left unconnected and configured to be turned off. TXDATA_P [31:0] for search, read, and write TXDATA_N [31:0] for search, read, and write TXDATA_P [47:32] for statistics. TXDATA_N [47:32] for statistics, if unused, leave disconnected and powered down. See recommended RX/TX Data lane assignments for Jericho 2C+ (Table 44 and Table 45), Jericho+ BCM88680 (Table 50), Jericho 2 BCM88690 (Table 51), and Jericho 2C BCM88800 (Table 52).

Table 22: JTAG Pin Descriptions

Parameter	Symbol	Type	No. Pins	Description
Test Clock Input	TCK	I	1	The JTAG test clock TCK pin is input only and provides the clock for all JTAG operations. The TCK pin is independent of the processor and interface clocks. TCK has no internal bias; when not active this pin must be externally biased to VDD18 or VSS to prevent floating inputs. If JTAG is not active, provide bias to VDD18 or ground through a resistor. This pin does not have ODT.
Test Data Input	TDI	I	1	The Test Data Input TDI signal is input only and is sampled on the rising edge of the Test Clock. Data and test instructions are received serially by the test logic through TDI. Leave unconnected if not used. Contains a weak pull-up. This pin does not have ODT.
Test Data Output	TDO	O	1	The Test Data Output TDO signal is output only and available on the falling edge of the Test Clock. The Test Data Output driver always drives even when not passing test data.
Test Mode Select	TMS	I	1	The Test Mode Select TMS signal is input only and sampled on the rising edge of the Test Clock. The Test Mode Select is the command input for the TAP controller. Leave unconnected if not used. Contains a weak pull-up. This pin does not have ODT.
Test Reset	TRST_L	I	1	The Test Reset Input TRST_L is an active low reset which provides for asynchronous initialization and drives the TAP controller into the Test-Logic-Reset state. TRST_L must be held low during normal operation. This signal contains a weak internal pull-up. Connection to this input pin is required. This pin does not have ODT.

Table 23: Miscellaneous Pin Descriptions

Parameter	Symbol	Type	No. Pins	Description
General Interrupt Output	GIO_L	O	1	General Interrupt Output is asynchronous, active-low, open-drain output that flags the occurrence of error conditions, which includes database parity errors. This signal should be routed back to the host device as an interrupt signal. GIO_L is asserted when Packet Errors, Interface Errors or Device Errors are detected. When GIO_L is asserted, the host device should read the 3Error Status Register to determine the cause of the error. GIO_L remains asserted until the corresponding error bit is cleared in the Error Status Register. This pin is open drain. Externally pull-up this pin to VDD18 via a 1 kΩ resistor.
Mode Select	MSEL[1:0]	I	2	Mode Select are static signals which are used to select device operation modes MSEL[1:0] = 2'b00 = standard mode of operation MSEL[1:0] = 2'b01 = reserved MSEL[1:0] = 2'b10 = reserved MSEL[1:0] = 2'b11 = reserved These pins must be pulled low with a 1 kΩ resistors to VSS on the board. Connection to these input pins is required. These pins do not have ODT.
Protocol Select	PRTCL	I	1	Protocol select. 0 = External Look-up (ELK) SerDes protocol. 1 = Interlaken Look-Aside (ILA) SerDes protocol. This pin does not have ODT.
Resistor Calibration 0	RESCAL0	I	1	External Bias Resistor 0 One external 4.53 kΩ resistor must be connected between this pin and VSS. Resistor accuracy less than 1%.
Resistor Calibration 1	RESCAL1	I	1	External Bias Resistor 1 One external 4.53 kΩ resistor must be connected between this pin and VSS. Resistor accuracy less than 1%.

Table 23: Miscellaneous Pin Descriptions (Continued.)

Parameter	Symbol	Type	No. Pins	Description
AVS Output Voltage	AVSA	Sig or other	1	Automatic Voltage Scaling A Pin. AVS connection and operation are required for proper device operation. Pin is used for automatic voltage scaling. See the Board Design Guidelines for proper pin connection.
AVS Output Voltage	AVSD	PWR	1	Analog Voltage Scaling D Pin. AVS connection and operation are required for proper device operation. Pin is used for automatic voltage scaling. See the Board Design Guidelines for proper pin connection.

Table 24: Test, Power, and Ground Pin Descriptions

Parameter	Symbol	Type	No. Pins	Description
Do Not Connect	DNC	I/O	80	Do not connect.
Do Not Populate	DNP	–	4	Do not populate.
Core Power	VDD	PWR	190	Programmable 0.850V nominal core voltage supply.
GP 1.8V Power	VDD18	PWR	8	1.8V General purpose voltage supply.
Memory Power	VDDM	PWR	20	0.85V Memory voltage supply.
SerDes 1.8V PLL Power	VDDS18PLL	PWR	6	SerDes 1.8V PLL power rail. Connect to PLL power supply filter.
SerDes PLL Power	VDDS08PLL1 to VDDS08PLL6	PWR	6	0.8V SerDes PLL Power1 to 6. Connect to PLL power supply filter.
PCIe PLL Power	VDDPCPLL	PWR	1	0.8V PCIe PLL supply. Connect to PLL power supply filter.
Core PLL Power	VDDCPLL	PWR	1	1.8V Core PLL supply. Connect to PLL power supply filter.
Voltage Sense Negative	VSENSEN	GND	1	Negative Voltage Sense. This pin is shorted to the on-die VSS plane. It is to be used for regulator sensing and cannot carry large current. If unused, this pin must be shorted to VSS.
Voltage Sense Positive	VSENSEP	PWR	1	Positive Voltage Sense. This pin is shorted to the on-die VDD plane. It is to be used for regulator sensing and cannot carry large current. If unused, this pin must be shorted to VDD.
Core Ground	VSS	GND	649	Core Ground.
Core PLL Ground	VSSCPLL	GND	1	Connect to PLL power supply filter GND for VDDCPLL.
PCIe Power	VDDPC	PWR	2	0.8V PCIe power supply.
SerDes Power	VDDS1 to 6	PWR	72	0.8V SerDes power supply.
SerDes Logic Power	VDDSL	PWR	12	0.90V SerDes logic supply.
SerDes TX Termination Power	VDDS12	PWR	12	1.2V SerDes transmit termination power supply.

Chapter 10: Statistics and Counters

10.1 Overview

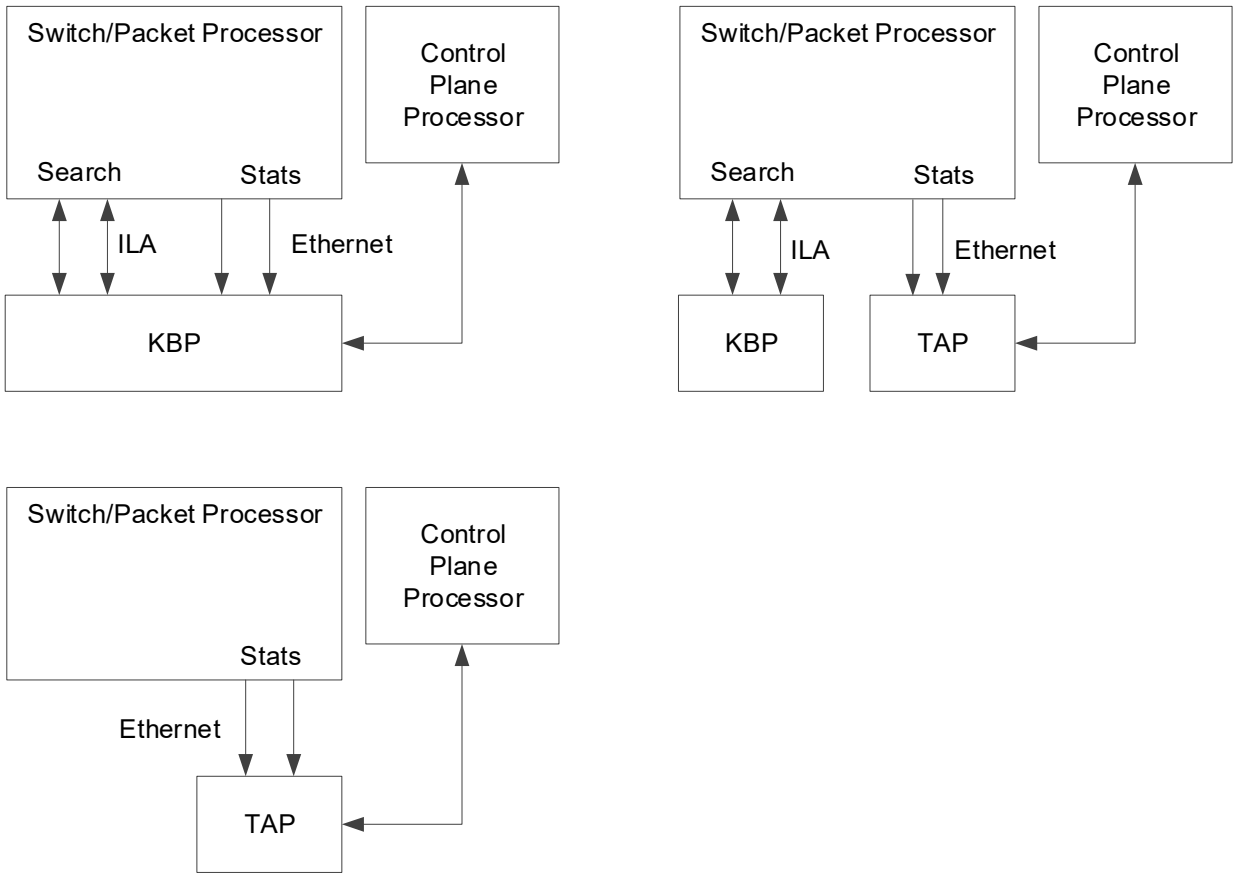
The statistics and counter engines enable the host and control plane to easily and effectively track both on-chip search hits for ACLs and routes (search-and-count) as well as independent statistics generated from the host, which are unrelated to on-chip search functions.

For Broadcom switches, there are 16 receive and transmits lanes arranged at 2 x (2 x 200G) ports which are sent via Ethernet interface. Up to 16M byte-packet pairs are supported. Dynamic reporting of counters to host memory is also supported. Two lanes per port (for example: 100G per port) is also supported.

Figure 20 shows the connectivity options with switch/packet processor in context of statistics and counter functionality.

The Broadcom BCM5235 Telemetry Analytic Processor (TAP) performs collection and analysis of network status and packet data. Network and packet information is collected at line rate and processed in real time through high-speed parallel operations. Any network data may be collected; examples include billing, port access, flow ID, queue size, IP route, traffic disposition, security events, and OAM stats.

Figure 20: Switch/Packet Processor Connectivity Options



10.1.1 General Features

The following general features are supported:

- 32 Statistic Engines per User Data Array core
- Total 16M x 64b counters/statistics or 32M x 32b statistics/counters using 2-to-1 compression
- Up to 64 statistics/counter tables
- Counter management logic that guarantees no overflow by DMA to host memory via PCIe

10.1.2 Statistics/Counters Tables Specifications

The following statistics and counter table specifications are supported:

- Single-value statistics as well as counter pair supported.
- Counter pair has two single-value counters: packets and bytes.
- Each table consists of counter sets or counter groups. One table can be configured to represent as small as 2 Mb of counter memory and as large as 512 Mb of counter memory.

10.1.3 Independent Statistics Records Specifications for Broadcom Switches

The following independent statistics and record specifications are supported for Broadcom switches:

- Statistics records, each having one of the four fixed lengths: 64b, 72b, 80b, or 96b.
- In a given application, only two sizes are expected-one for ingress and other for egress.
- A statistics record has three regions: Type, OPCODE, and UnStructured Region (USR).
- The Opcode field (2b size) indicates if the record is ingress, egress, or NULL.
- Each record holds up to four statistic update commands.
- Each statistics command has own statistics pointer address, set index attribute, and value located inside the USR.
- It is possible for multiple commands to share fields. For example each command shares same packet byte count.

Table 25: Supported Counter Widths with Core and Full Device Cap

Counter Widths	Capacity	
	Core	Device
32b	16M	32M
64b	8M	16M
128b	4M	8M

10.1.4 Statistic Interface

The processor interface receives statistic records generated from the switch/packet processor over Ethernet. Each record is flexible and may contain user-defined flow, port, or traffic-class counter data. There are 16 receive lanes arranged as 2 × (2 × 200G) ports for a total of four receive ports. No transmit ports are available. The user can initiate a DMA of counter data via PCIe Gen2 port to host control plane memory space.

Figure 22 and Figure 23 show possible switch/packet processor connections to the processor.

Table 26: Statistic Interface Port Assignment

TAP Port	Port 0	Port 1	Port 2	Port 3
SerDes	35:32 Lanes	39:36 Lanes	43:40 Lanes	47:44 Lanes

Figure 21: Single-Host Configurations

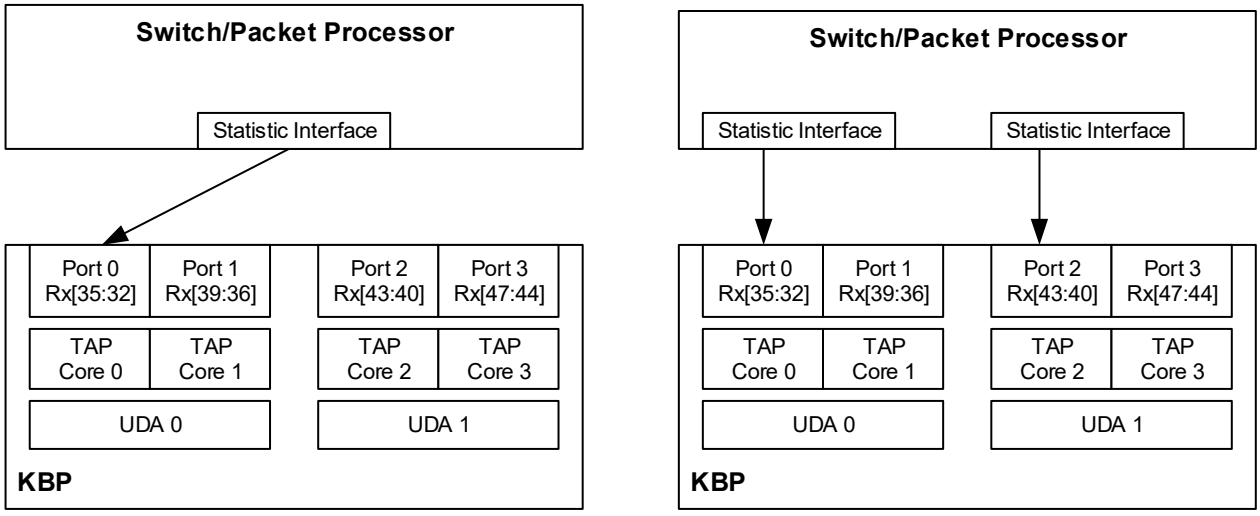
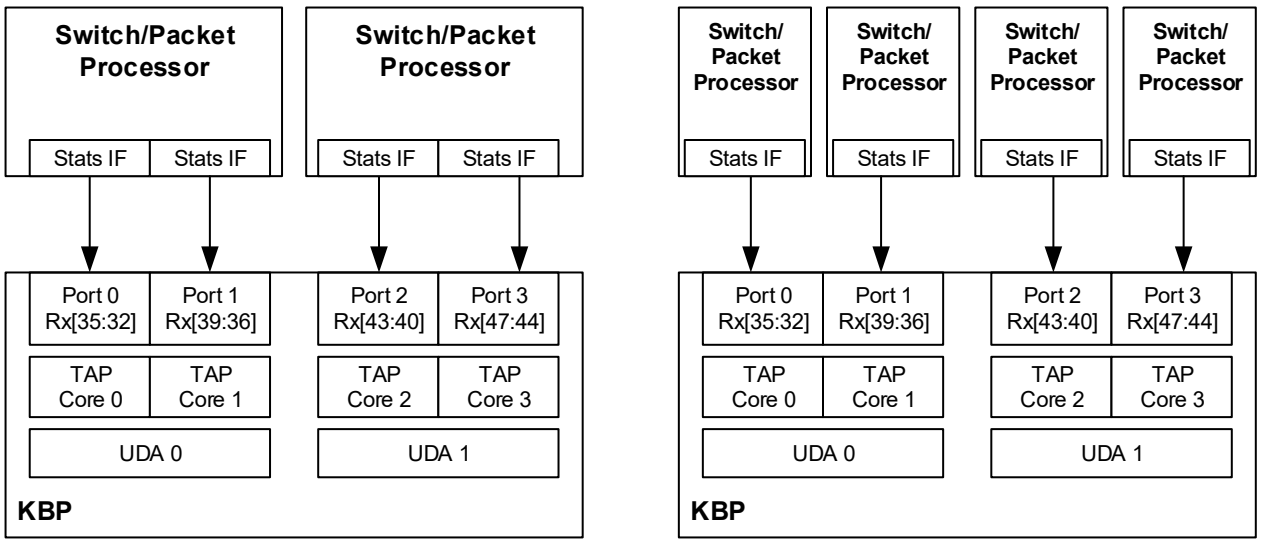


Figure 22: Dual and Quad-Host Configurations

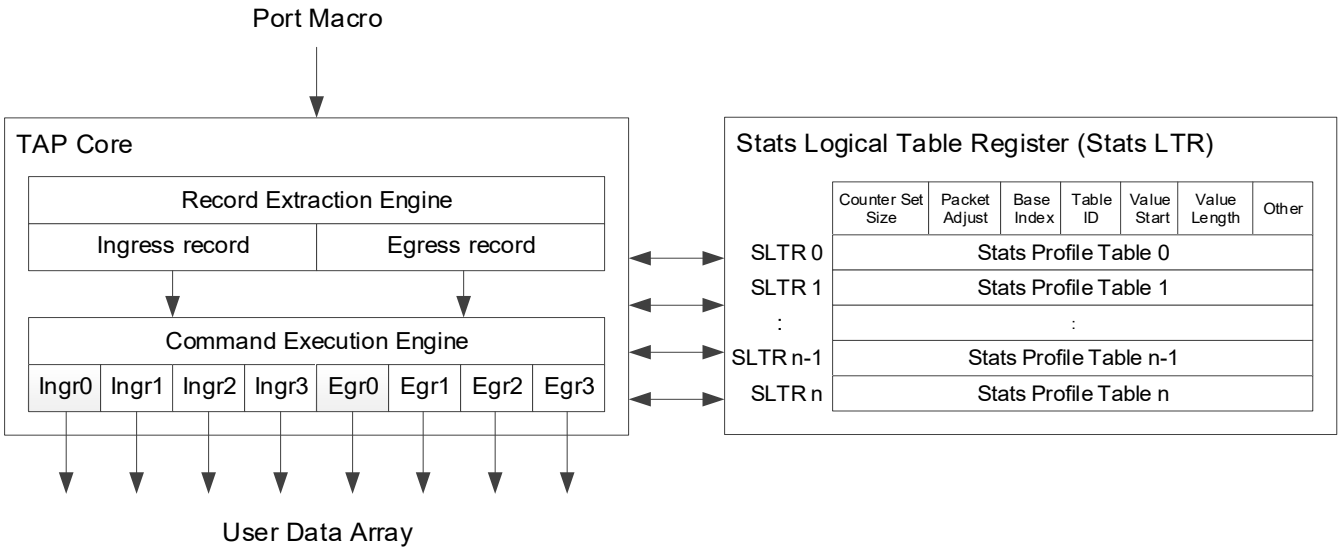


10.1.5 TAP Core and Stats Logical Table Register (Stats LTR)

Packet records exit the statistics port macro and are received into the TAP core. There are four TAP cores. Each TAP core consists of a record extraction engine and command execution engine. The record extraction engine extracts one ingress record and one egress record and transfers the instruction and data to the command execution engine. The command execution engine processes up to four instructions for ingress and four instructions for egress for a total of eight instructions. Each instruction accesses the stats profile register to determine the action to take. The appropriate counter tables in the user data array (UDA) are updated.

The Stats LTR stores the counter table profiles programmed by the user and works as a map to parse the incoming stats records to extract the stats command data. Each stats command updates one StatsObject. Each SLTR has four identical sub-registers (called groups). Each group is describing one StatsObject. SLTR fields include counter set size, packet adjustment, stats table base index, stats table ID, value start, value length select, stats pointer max, stats pointer min, invalid-if-zero, set index attribute start (0 to 7), set index attribute valid (0 to 7), stats pointer start (0 to 3), and stats pointer length (0 to 3).

Figure 23: TAP Core and Stats LTR Block Diagram

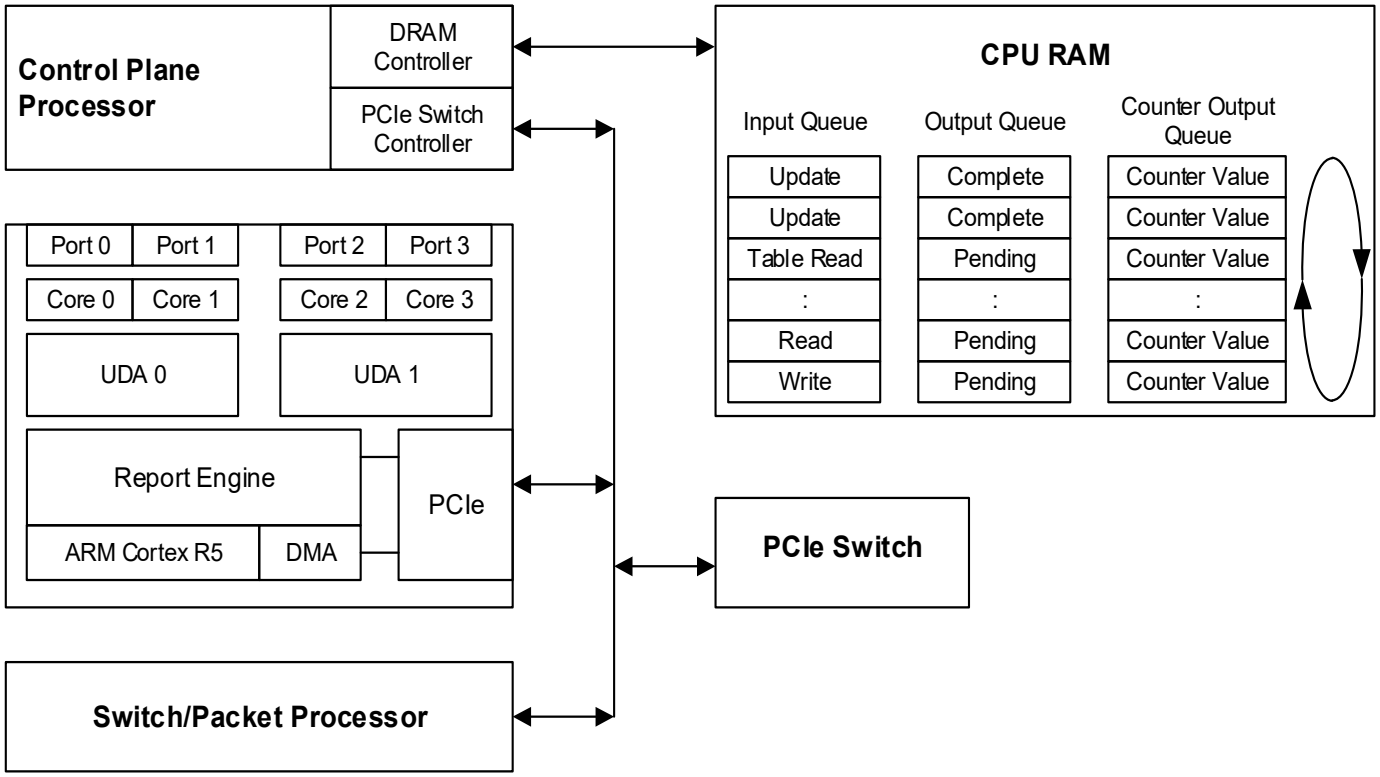


10.2 Report Engine

The device uses a Broadcom proprietary reporting algorithm, also referred to as dynamic eviction, for monitoring and reporting counters. Counter reporting is evaluated at each value increment. The counter report rate will remain constant and may happen at any time.

The processor is a master DMA that dynamically pushes 128b event messages to local host processor memory. An accelerator counter scan command is available when system software or control plane API needs a fresh image of counter values. In response to such a message, device DMAs out a set of Counter Update messages followed by Counter Update Completion message.

Figure 24: Report Engine through PCIe to Control Plane Processor



For counter table management, the processor writes reported and read counter values into the output queue stored in the circular RAM buffer without the involvement of the control plane processor. At a later time, the control plane processor processes counter values.

The control plane processor is expected to operate asynchronously from the processor by enqueuing a command, going away, and then checking later as the command is completed. All control plane processor work is completed in its own RAM with processor read and writes to the same RAM through PCIe. The bandwidth required for 100k updates/sec requires several transactions for each update and uses less than 100 Mb/s of bandwidth. The control plane processor does not need to wait for updates to be complete.

As an example, to read a table with 128k counters with each counter represented by 128b in the counter output queue, requires 128k × 128b = 16 Mb of data. If the allocated control plane processor bandwidth is 0.5% for counter reads (~200 clocks between each counter read), then 128b in 200 ns is ~640 Mb/s of PCIe write bandwidth. Total time to read all counters ~25 ms (millisecond).

Table 27: Example Read Bandwidth for 32K Counters

No. Counters	Counter Width (bits)	Minimum Read	No. Counter per Read	No. Read Ops	Total Data (128b = counter message size)		Bandwidth Gb/s	PCIe Gen2 (20% overhead, 4 Gb/s)
32,000	32	64	2.0	16,000	2,048,000		0.0020	0.05%
32,000	64	64	1.0	32,000	4,096,000		0.0041	0.10%
32,000	128	64	0.5	64,000	8,192,000		0.0082	0.20%

Table 28: Example Read Time Given Data Plane Bandwidth Allocation

Clock Frequency (MHz)	Data Plane BW Allocation	No. Operations per Sec		No. Read Operations	Time to Complete Read Operations (ms)
1,000	0.01%	100,000		16,000	160
1,000	0.05%	500,000		16,000	32
1,000	0.10%	1,000,000		16,000	16
1,000	0.50%	5,000,000		16,000	3
1,000	1.00%	10,000,000		16,000	2
1,000	0.01%	100,000		32,000	320
1,000	0.05%	500,000		32,000	64
1,000	0.10%	1,000,000		32,000	32
1,000	0.50%	5,000,000		32,000	6
1,000	1.00%	10,000,000		32,000	3
1,000	0.01%	100,000		64,000	640
1,000	0.05%	500,000		64,000	128
1,000	0.10%	1,000,000		64,000	64
1,000	0.50%	5,000,000		64,000	13
1,000	1.00%	10,000,000		64,000	6

10.3 Data Structure in PCI Control Plane Memory

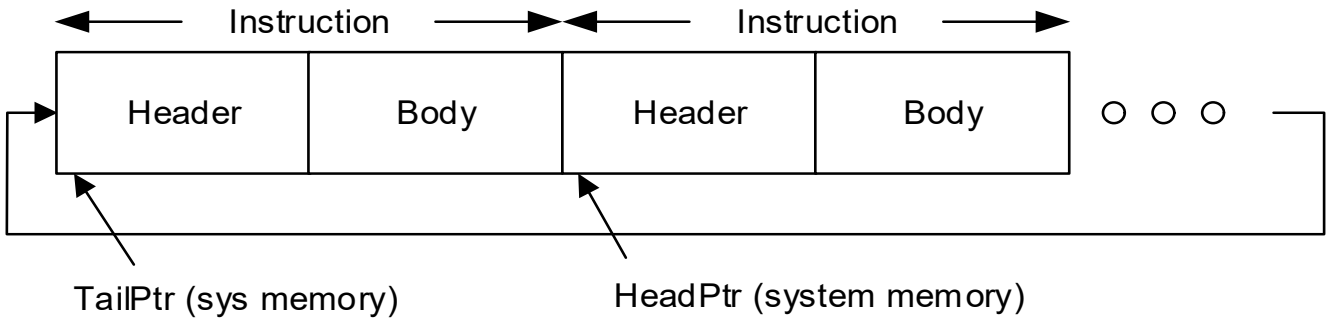
Efficient communication between the PCI Control Plane and the TAP device is through a set of circular buffers. TAP allows for a single circular buffer to retrieve the requests from the PCI control plane to the device. For responses to the requests sent by device back to the control plane, there are two circular buffers that can be configured by the software. In addition to this, there are unsolicited messages that are sent by device to PCI control plane. The counter event messages are such messages. It is possible for such counter event messages to be sent by the device in two independent circular buffers (one for dynamic counter eviction and one for the counter scan related ones) or in single buffer. The single buffer option is selected by default. The request and response messages follow {Body, Header} format (header located at lower address and body located at higher address) and counter event messages follow 128b data format.

For every type of circular buffer, there are two pointers–head and tail. The head is maintained and updated by the source of the message. So in the case of a request, the head pointer is maintained and updated by the PCI control plane and in the case of responses or counter events, the same is maintained and updated by the device. There is a pre-defined area in the PCI control plane memory where the two pointers (per buffer) are maintained and updated. At the time of device initialization, the PCI control plane configures the circular buffer size, its address, and location of the related pointers. The device polls for the pointers to determine if there is any request to be retrieved from the PCI control plane. Also, if the pointers for a response indicate that there is no memory available for the device to write the response into the PCI control plane, it polls for the tail pointer value for each of the enabled circular buffers (also referred to as DMA Transmit Channels) to find if the space is available. The polling of the head pointer for a request circular buffer happens only if a device does not have any more requests to process and likewise polling for tail pointer for various response circular buffers happens only if there is no space found in the respective buffers. The polling interval is configurable and it should be set such that the polling does not add any significant overhead to the PCI bus and at the same time, it does not create undue latency.

10.3.1 Request Data Structure

Figure 25 shows the structure of the request circular buffer in the PCI host memory. The software needs to poll for the TailPtr only if it does not have any more space for submitting request messages. By that time, it is very likely that most of the request messages have already been picked by the device (as would be indicated by updated TailPtr value). The header (HDR) size is fixed (64b), but the body size is dependent on the type of request instruction. For example, the PIOWR instruction takes 3x64b as the size of the body, but PIORDX/PIORDY take 2x64b as the body size.

Figure 25: Structure of the Request Circular Buffer



10.3.2 Response Data Structure

Response (solicited type) data structure is similar to the request data structure. The header is 64b and the body is dependent upon the type of request that the response belongs to. The typical size of the body is 2x64b.

10.3.3 Counter Event Data Structure

A Counter Eviction Event is an event in which a counter (or a group of counters) is being read out through DMA channels to the host processor memory while its local copy (copies) are reset to zero. There are two main counter event types. The first type is a bulk read of a counter or a group of counters and the second type is a dynamic eviction of a counter.

Counter data structure does not have a {Body, Hdr} type of response. Instead, it can be seen as a set of 128b data structures, each data structure reflecting one counter event message. The following provides further detail of the 128b data structure. The message can have counter data in it or it can carry a notification. There are two types of notification messages, one that indicates the occurrence of error is also referred to as Error notification message. In this capacity, the message is an alternative to a regular counter data message. The other type of notification message is a Completion notification message. This message is applicable to the Counter Scan command only and it indicates that all counters that were requested by the command have been uploaded to the PCI control plane memory.

10.4 DMA Message Format

There are two DMA message formats (dynamic eviction and counter read scan) available for sending counter values through DMA to the control plane memory circular buffer. Both share the same main message format except that the counter read scan has a *Done* message.

The main DMA message format is 2 x 64b words. The first word is the value and the second word is the address plus control information. The two DMA message formats, Normal and Error, are listed in Table 29 and Table 30 respectively. The DMA Done and Counter Scan Completion message format are listed in Table 31 and Table 32 respectively.

Table 29: DMA Message Format—Normal

Field	Location	Description
Counter Value	63:0	Counter Value. CtrValue.
Counter Address	63:3	Counter Address in System Space (host memory). CtrAddrInSysSpace.
Notification	2	Notification Done: 1'b0
Counter Type	1:0	Counter Type. CtrType. 2'b00: 64b single counter 2'b01: 64b compressed counter pair 2'b10: 32b counters (two counters) 2'b11: 128b uncompressed counter pair (software will know which pair)

Table 30: DMA Message Format—Error

Field	Location	Description
Reserved	63:60	Reserved: 4'h0
Error Value	59:56	Error: 4'h4
Reserved	55:0	Reserved: 56'h0
Counter Address	63:3	Counter Address in System Space (host memory). CtrAddrInSysSpace.
Notification Error	2	Notification Error: 1'b1
Counter Type	1:0	Counter Type. CtrType. 2’b00 = 64b single counter 2’b01 = 64b compressed counter pair 2’b10 = 32b counters (two counters) 2’b11 = 128b uncompressed counter pair (software will know which pair)

Table 31: DMA Done Message Format

Field	Location	Description
Done	127:126	Done 2'b00 = Reserved 2'b01 = Reserved 2'b10 = Done, no error 2'b11 = Done, error during transaction
Reserved	125:116	Reserved: 10'h021
Marker Nibble	115:112	Marker Nibble returned from the Counter Read Scan Request
Time Out Indicator	111:108	TimeOut Indicator (if 0xF, then indicates Scan terminated due to too long of a delay in scan read)
Reserved	107:64	Reserved: 44'h0
Host Counter Address	63:3	Host Counter Address
Notification Done	2	Notification Done: 1'b1
Counter Type	1:0	Counter Type. CtrType. 2’b00 = 64b single counter 2’b01 = 64b compressed counter pair 2’b10 = 32b counters (two counters) 2’b11 = 128b uncompressed counter pair (software will know which pair)

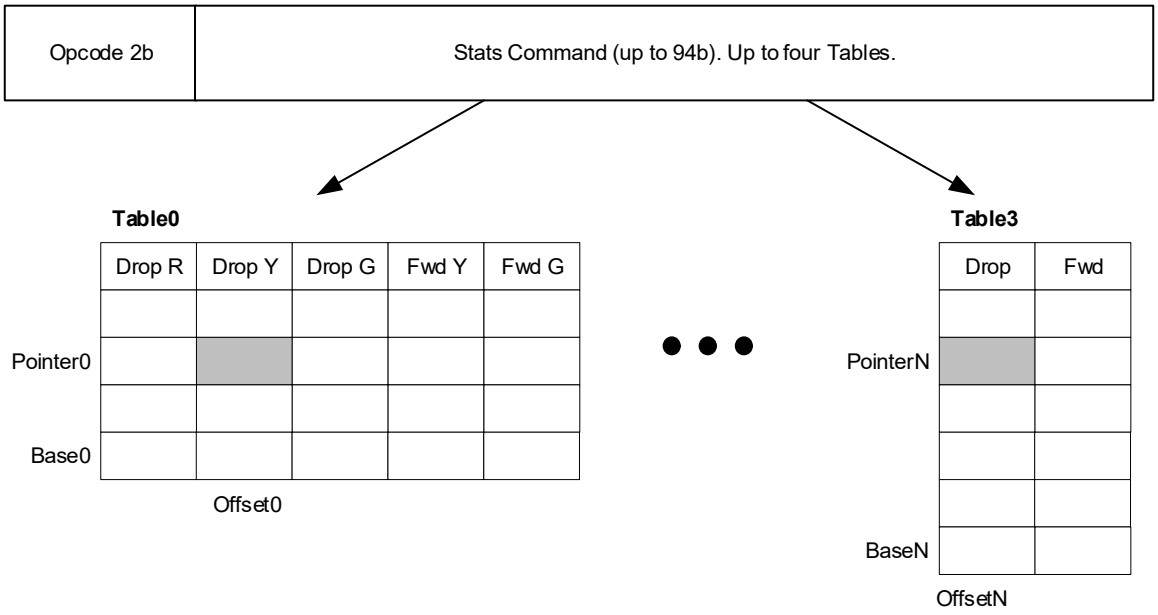
Table 32: DMA Counter Scan Completion Message Format

Field	Location	Description
Error Indicator	127:124	No Error 4'b1000 Error 4'b1100
Reserved	123:120	Reserved. 4'b0010
Reserved	119:116	Reserved. 4'b0001
Completion Signature	115:108	Completion Signature is generated from the original counter scan request and carries additional error indications.
Reserved	107:64	Reserved. 44'b0
Reserved	63:4	Reserved. 61'b0
Reserved	3:2	Reserved. 1'b1
Reserved	1:0	Reserved. 2'b00

Chapter 11: Statistics Record

A statistics command consists of 2b Opcode and up to 94b of statistics command data. Opcode determines if ingress or egress record and points to opcode translation register. Each record can update up to four stats counter objects (StatsObject), in what is referred to as stats commands.

Figure 26: Stats Record and Stats Table



11.1 Statistics Command

Opcode 2b	Stats Command (up to 94b). Up to four StatsObjects.

Opcode Description: processor parses the first two bits (opcode) of the record to determine record type.

Opcode	Description
2'b00	NULL opcode
2'b01	Invalid opcode
2'b10	Ingress record
2'b11	Egress record

Opcode Translation Register – Ingress and Egress records have lookup registers to define the 8b opcode extension inside the record.

Table 33: Opcode Translation Description

Opcode	Description
Record Size	Record Size 2'b00 = 64b record 2'b01 = 72b record 2'b10 = 80b record 2'b11 = 96b record
Opcode Extension	Organized as four ports, two per core. Each port has 1 x 16 ingress and 1 x 16 egress SLTRs. Opcode Extension (SLTR Index): 0 to 8b, MSB bit is reserved. 7'b000_0000 = Stats LTR 0 Pointer 7'b000_0001 = Stats LTR 1 Pointer : 7'b111_1111= Stats LTR 127 Pointer

11.2 Stats Record Structure

The opcode and opcode extension are merged to form a pointer for a Look-Up Table (LUT). LUT information references the Stats Logical Table Register (Stats LTR or SLTR). The Stats LTR is a programmable register storing statistics and counter table profiles. The SLTR profile, packet processor and traffic manager attributes are sent to the Stats Command Extractor (SCE) and it creates the necessary StatsObject data, which are sent to the counter engine for processing. Counter engine will process and increment the appropriate counter.

The Stats LTR works as a map to parse the incoming stats records to extract the stats commands. Each stats command will update one StatsObject. Each SLTR has four identical sub-registers (called groups). Each group describes one StatsObject. SLTR fields include Counter Set Size, Packet Adjustment, Stats Table Base Index, Stats Table ID, Value Start, Value Length Select, Stats Pointer Max, Stats Pointer Min, Set Index Attribute Start: 0 to 7, Set Index Attribute Valid (0 to 7), Stats Pointer Start (0 to 3), and Stats Pointer Length (0 to 3). The device hardware logic is flexible such that the byte count may be positioned in another stats record location.

Table 34: Stats Record Format Details

Opcode 2b	Byte Count 14b	Opcode Extension 0 to 8b	PP and TM Attributes 0 to 24b	StatsObject3 0 to 20b	StatsObject2 0 to 20b	StatsObject1 0 to 20b	StatsObject0 0 to 20b

Table 35: Stats Record Definition

Field Name	Description
Opcode	Opcode: 2b 2’b00 = NULL record 2’b01 = Invalid record 2’b10 = Ingress record 2’b11 = Egress record
Byte Count (Packet Size)	Byte Count Value: 10b to 14b 10b = Packet Length in bytes = 1024-byte max. 11b = Packet Length in bytes = 2048-byte max. 12b = Packet Length in bytes = 4096-byte max. 13b = Packet Length in bytes = 8128-byte max. 14b = Packet Length in bytes = 16256-byte max.
Opcode Extension	Organized as four ports, two per core. Each port has 1 x 16 ingress and 1 x 16 egress SLTRs. Opcode Extension (SLTR Index): 0 to 8b, MSB bit is reserved. 7'b000_0000 = Stats LTR 0 Pointer 7'b000_0001 = Stats LTR 1 Pointer : 7'b111_1111= Stats LTR 127 Pointer

Table 35: Stats Record Definition (Continued.)

Field Name	Description
Packet Processor and Traffic Manager Attributes	Packet Processor and Traffic Manager attributes. Attribute data used for determining stats counter offset value (more about this in counter-set definition). Multiple attributes include but not limited to: Drop Precedence = 2b Packet Color (traffic class) = 3b Packet Disposition = 1b Drop Reason = 3b In-PP-Port = 8b Out-PP-Port = 8b Egress-MC = 1b System MC = 1b See the Broadcom Jericho2 BCM88690 Counters, Meters, and Statistics Interface Application Note for additional information.
Stats Object 3	Stats Object 3 descriptor. Descriptor represents counter address inside stats table/database. Stats Object consists of up to four variable size fields (VSF) up to 21b total. Counter addressed formed by concatenation of VSFs. VSF locations defined in Stat LTR.
Stats Object 2	Stats Object 2 descriptor. Descriptor represents counter address inside stats table/database. Stats Object consists of up to four variable size fields (VSF) up to 21b total. Counter addressed formed by concatenation of VSFs. VSF locations defined in Stat LTR.
Stats Object 1	Stats Object 1 descriptor. Descriptor represents counter address inside stats table/database. Stats Object consists of up to four variable size fields (VSF) up to 21b total. Counter addressed formed by concatenation of VSFs. VSF locations defined in Stat LTR.
Stats Object 0	Stats Object 0 descriptor. Descriptor represents counter address inside stats table/database. Stats Object consists of up to four variable size fields (VSF) up to 21b total. Counter addressed formed by concatenation of VSFs. VSF locations defined in Stat LTR.

11.3 Stats Record Format Examples

The billing example shown in Table 36 updates a 64b stats record with two counters representing a 3b packet color with a 1b packet disposition. The second billing example shown in Table 37 updates a 96b stats record with three counters.

Table 36: Billing 64b Stats Record with Two Counters

Field Name and Bits Positions	Description
Opcode [63:62]	Record Type: Ingress/Egress
Byte Count Value [61:48]	Between 64B and up to 16 KB
Opcode Extension [47:40]	Stats LTR Index
Ptr0 [39:22] Address	First counter pointer (one fragment)
SIA3 [21]	Fourth Set Index Attribute – Packet Color
SIA2 [20]	Third Set Index Attribute – Packet Color
SIA1 [19]	Second Set Index Attribute – Packet Color
SIA0 [18]	First Set Index Attribute – Packet Disposition
Ptr1 [17:0] Address	Second counter pointer (one fragment)

Table 37: Billing 96b Stats Record with Three Counters

Field Name and Bits Positions	Description
Opcode [95:94]	Record Type: Ingress/Egress
Opcode Extension [93:86]	SLTR Index
Byte Count Value [85:72]	Between 64B and up to 16 KB
Ptr0 [71:51] Address	First counter pointer (one fragment)
SIA0 [50]	First Set Index Attribute – Drop Precedence
SIA1 [49]	Second Set Index Attribute – Drop Precedence
Ptr1 [48:28] Address	Second counter pointer (one fragment)
SIA2 [28]	Third Set Index Attribute (overlapped) – Drop Disposition
Ptr2_2 [27:18] Address	Third pointer (second fragment)
SIA4 [17]	Fifth Set Index Attribute – In-PP-Port
SIA3 [16]	Fourth Set Index Attribute (overlapped) – In-PP-Port
Ptr2_1 [16:3] Address	Third counter pointer (first fragment)
SIA5[2]	Sixth Set Index Attribute – Packet Color
SIA6[1]	Seventh Set Index Attribute – Packet Color
SIA7[0]	Eighth Set Index Attribute – Packet Color

Chapter 12: Ethernet Packet Format

The processor interface prescribes to the industry standard Ethernet packet format with a preamble and start-of-frame delimiter. The stats port macro packet consists of multiple 512-bit words (64 bytes = 1 x 512-bit word). If the packet has more than 64 bytes, this results in multiple 512-bit words.

12.1 Standard and Stats Processor Ethernet Packet Format

Table 38: Standard Ethernet Packet Format

Preamble	Start of Frame Delimiter	MAC Destination	MAC Source	IEEE 801.1Qtag (Optional)	Ethernet Type	Payload	Frame Check CRC	Inter Packet Gap
7 bytes	1 byte	6 bytes	6 bytes	4 bytes	2 bytes	46 to 1500 bytes	4 bytes	12 bytes

Table 39: Processor Ethernet Packet Format

Preamble	Start-of-Frame Delimiter	Payload	Frame Check CRC	Inter-Packet Gap
7 bytes	1 byte	64 to 1024 bytes	4 bytes	12 bytes

The preamble and start-of-frame delimiter totals 8 bytes:

7	6	5	4	3	2	1	0
0x55	0x55	0x55	0x55	0x55	0x55	0x55	0xD5

The Inter Packet Gap has n bytes of idle characters:

0	1	2	3	….	n-1
IDLE	IDLE	IDLE	IDLE	…	IDLE

Table 40 and Table 42 are examples of 11 records with 64b, 72b, 80b, and 96b length packed into an Ethernet packet, which is shown in Table 42.

Table 40: Ethernet Packet Example Key

Record	Length
Record_64	[63:0]
Record_72	[71:0]
Record_80	[79:0]
Record_96	[95:0]

Table 41: Ethernet Packet Record Byte Count

Record	Length	Byte Count
Record0	Record_72	9
Record1	Record_80	10
Record2	Record_80	10
Record3	Record_64	8
Record4	Record_80	10
Record5	Record_96	12
Record6	Record_80	10
Record7	Record_72	9
Record8	Record_72	9
Record9	Record_96	12
Record10	Record_80	10
Total		109

Table 42: Ethernet Packet Example

7	6	5	4	3	2	1	0
0x55	0x55	0x55	0x55	0x55	0x55	0x55	0xD5
Record0 71:64	Record0 63:56	Record0 55:48	Record0 47:40	Record0 39:32	Record0 31:24	Record0 23:16	Record0 15:8
Record0 7:0	Record1 79:72	Record1 71:64	Record1 63:56	Record1 55:48	Record1 47:40	Record1 39:32	Record1 31:24
Record1 23:16	Record1 15:8	Record1 7:0	Record2 79:72	Record2 71:64	Record2 63:56	Record2 55:48	Record2 47:40
Record2 39:32	Record2 31:24	Record2 23:16	Record2 15:8	Record2 7:0	Record3 63:56	Record3 55:48	Record3 47:40
Record3 39:32	Record3 31:24	Record3 23:16	Record3 15:8	Record3 7:0	Record4 79:72	Record4 71:64	Record4 63:56
Record4 55:48	Record4 47:40	Record4 39:32	Record4 31:24	Record4 23:16	Record4 15:8	Record4 7:0	Record5 95:88
Record5 87:80	Record5 79:72	Record5 71:64	Record5 63:56	Record5 55:48	Record5 47:40	Record5 39:32	Record5 31:24
Record5 23:16	Record5 15:8	Record5 7:0	Record6 79:72	Record6 71:64	Record6 63:56	Record6 55:48	Record6 47:40
Record6 39:32	Record6 31:24	Record6 23:16	Record6 15:8	Record6 7:0	Record7 71:64	Record7 63:56	Record7 55:48
Record7 47:40	Record7 39:32	Record7 31:24	Record7 23:16	Record7 15:8	Record7 7:0	Record8 71:64	Record8 63:56
Record8 55:48	Record8 47:40	Record8 39:32	Record8 31:24	Record8 23:16	Record8 15:8	Record8 7:0	Record9 95:88
Record9 87:80	Record9 79:72	Record9 71:64	Record9 63:56	Record9 55:48	Record9 47:40	Record9 39:32	Record9 31:24
Record9 23:16	Record9 15:8	Record9 7:0	Record10 79:72	Record10 71:64	Record10 63:56	Record10 55:48	Record10 47:40
Record10 39:32	Record10 31:24	Record10 23:16	Record10 15:8	Record10 7:0	CRC	CRC	CRC
CRC	IDLE	IDLE	IDLE	IDLE	IDLE	IDLE	IDLE
IDLE	IDLE	IDLE	IDLE	IDLE			

Chapter 13: NetRoute and NetACL

13.1 Overview

This device is a “heterogeneous” device featuring the convergence of massively parallel circuit technology with the superior low-power efficiency and flexibility of NetACL and NetRoute technologies. The processor enables multiple parallel searches and assemble all outputs into a single result based on a user-defined priority.

The processor employs advanced processing engines and intelligent load balancing to provide a dramatic increase in forwarding and access control lists (ACL) capacity, reduced power consumption, a single device with a fixed latency, and the ability to dynamically configure application tables for forwarding and ACL based on user scenarios.

The processor has fully deterministic longest-prefix match (LPM), ACL, and exact match functionality, and serves a broad range of forwarding applications including IPv4 unicast/multicast, IPv6 unicast/multicast, and MPLS-VPN.

NetRoute has the capability to perform two LPM in parallel and select one of the results, which are referred to as public and private. If private matches, then we select private; otherwise public is selected. An additional feature for default routes enables the ability to support private, private or public, and public.

The processor provides the largest database in the industry, supporting up to 20M IPv4 routes or 11M IPv6 routes and 4M 160-bit ACL entries on a single chip. The processor operates at line rate with no impact to the maximum number of decisions per second with a low fixed latency for all key sizes.

13.2 Features

- Support for 20M IPv4 route entries with 24b AD pointer
- Support for 11M IPv6 route entries with 24b AD pointer
- Support for 4M x 160b ACLs
- Capacity scales approximately linearly with entry width
- Support for variable size associated data and indirection tables
- 8x per core parallel searching without restrictions
- Up to 200k table updates per second
- Low latency for both forwarding and ACL applications
- Flexible database configuration to support both traditional searches, NetRoute, and NetACL applications

Chapter 14: Configuration Examples: Jericho 2, Jericho 2C, and Jericho 2C+

This processor seamlessly connects to Jericho 2C+ BCM88850, Jericho 2C BCM88800, Qumran 2C, Jericho 2 BCM88690, Jericho+ BCM88680, and Jericho BCM88670. See Table 43 through Table 53 for configuration examples.

Figure 27: BCM52311 OP and BCM16K OP2 Configurations with Jericho Processors

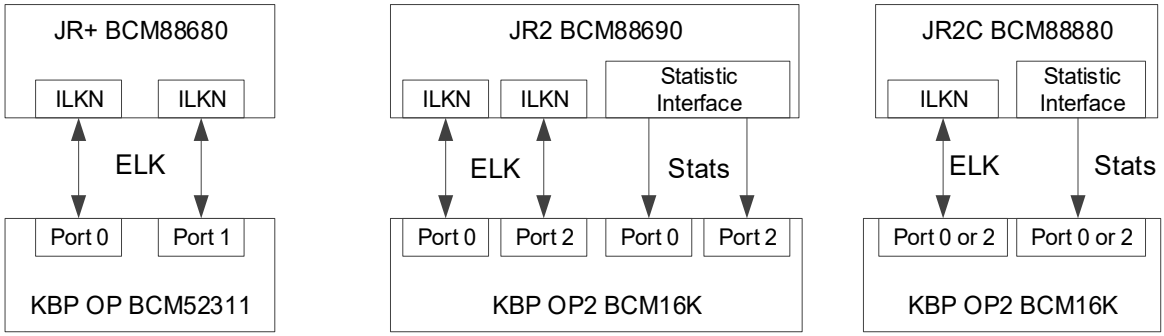


Table 43: BCM52311 OP and BCM16K OP2 Configurations with Jericho Processors

Feature	BCM88680 JR+ + BCM52311 OP	BCM88690 JR2 + BCM16K OP2	BCM88800 JR2C + BCM16K OP2
Packet Rate	835 MPPS	2000 MPPS	1000 MPPS
Parallelism	8	8	8
SerDes Rate supported	27.3 Gb/s	53.125 Gb/s	53.125 Gb/s
Max IPv4 FWD only	8M	15M	20M
Max IPv4 FWD and RPF	6M	7.5M	15M
Max IPv6 FWD only	5M	7.5M	10M
Max IPv6 FWD and RPF	3.5M	3.8M	7.5M
Max IPv4 ACLs	3M × 160b	3M × 160b	3M × 160b
Max IPv6 ACLs	2M × 320b	2M × 320b	2M × 320b
Update Rates	100 k/sec	200 k/sec	200 k/sec
Statistics	–	16M byte-packet pairs	16M byte-packet pairs
Table-entry counters	–	For each table entry	For each table entry

NOTE:

1. Capacity assumes 24b of associated data.
2. The capacity scale varies based on the database entry distribution.
3. Contact you Broadcom FAE and AE for specific database configurations.
4. Single dimension table.

Figure 28: BCM16K OP2 Configurations with Jerico 2C+ Processors

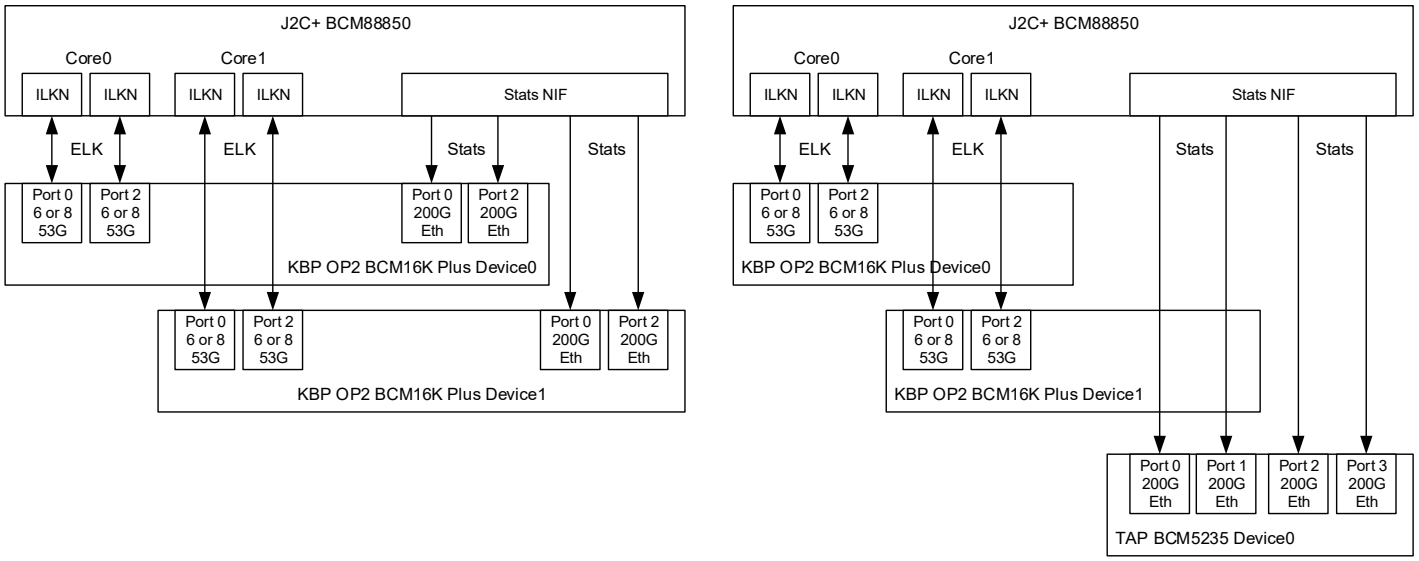


Table 44: 1x J2C+ with 2x BCM16K

RX/TX Data per Port	BCM16K Device0		BCM16K Device1		BCM16K Device0		BCM16K Device1	
	Search	Search	Search	Search	Stats	Stats	Stats	Stats
	Port 0	Port 2	Port 0	Port 2	Port 0	Port 2	Port 0	Port 2
4 Lanes	–	–	–	–	35:32	43:40	35:32	43:40
6 Lanes	5:0	21:16	5:0	21:16	–	–	–	–
8 Lanes	7:0	23:16	7:0	23:16	–	–	–	–

Table 45: 1x J2C+ with 2x BCM16K + 1x TAP BCM5325

RX/TX Data per Port	BCM16K Device0		BCM16K Device1		TAP BCM5235 Device0			
	Search	Search	Search	Search	Stats	Stats	Stats	Stats
	Port 0	Port 2	Port 0	Port 2	Port 0	Port 1	Port 2	Port 3
4 Lanes	–	–	–	–	35:32	43:40	35:32	43:40
6 Lanes	5:0	21:16	5:0	21:16	–	–	–	–
8 Lanes	7:0	23:16	7:0	23:16	–	–	–	–

14.1 ELK SerDes Connectivity Options for J2C+ BCM88850

Table 46 summarizes the ELK interface connectivity options for core 0 and Table 47 summarizes the ELK interface connectivity options for core 1.

Table 46: Core 0 ELK SerDes Connectivity Options

Number of Lanes	Interface	SerDes Number	Comments
8 lanes per interface	ELK 0	0 to 7	FAB octet 0
	ELK 1	8 to 15	FAB octet 1
6 lanes per interface	ELK 0	0 to 5	FAB octet 0
	ELK 1	6 to 11	FAB octet 0 and 1

Table 47: Core 1 ELK SerDes Connectivity Options

Number of Lanes	Interface	SerDes Number	Comments
8 lanes per interface	ELK 2	93 to 103	FAB octet 12
	ELK 3	104 to 111	FAB octet 13
6 lanes per interface	ELK 2	96 to 101	FAB octet 12
	ELK 3	102 to 107	FAB octet 12 and 13

14.2 Excerpt from J2C+ Data Sheet on Statistics

The external statistics processor can be Broadcom KBP or TAP devices. When connecting to the Broadcom KBP or TAP as an external statistics processor, the statistics interface should be configured to split mode.

Split mode – Each core has two statistics interfaces, and each interface is a 200GbE port (for a total of four 200GbE ports per BCM88850 device). The BCM88850 splits the egress and ingress statistics reports between the two ports.

This is a dedicated mode to connect directly to the statistics interface, which uses Ethernet ports from the NIF. Only NIF ports PM50-0 and PM50-9 can be used as statistics interfaces. The port usage per mode is as follows:

- Up to four Ethernet ports are used in quad mode (it is expected that in quad mode, the ports will be 100GbE).
- Up to two Ethernet ports are used in dual mode (it is expected that in dual mode, the ports will be 200GbE).
- Two Ethernet ports are used in split mode for each device core (four 200GbE ports total, including two on PM50-0 andtwo on PM50-9).

Table 48: TAP Configurations with J2C+ BCM88850

Attribute	Drawing 1 – TAP
Switch	
No. Switches	1 Switch
Cores	2 Core
Ports	4 Port
SerDes Lanes	4x 4 Lanes, 53 Gb/s
TAP	
No. TAP	1 TAP
Streams	Ingress and Egress, FDX
Ports	4 Port
SerDes Lanes	4x 4 Lanes at 53 Gb/s
Bandwidth	4x 200 Gb/s
Core Frequency	1 GHz
Packet Rate	2.7 BPPS FDX
Stats Objects	5.4B Stats objects/sec
No. UDA Cores	2 UDA
Statistics	4x 4 byte-packet pairs

Table 49: BCM16K OP2 Configurations with Jericho 2C+ Processors

Feature	BCM88850 J2C+ 2x BCM16K Plus
Packet Rate	2800 MPPS
Parallelism	8
SerDes Rate supported	53.125 Gb/s
Max IPv4 FWD only	15M
Max IPv4 FWD and RPF	7.5M
Max IPv6 FWD only	7.5M
Max IPv6 FWD and RPF	3.8M
Max IPv4 ACLs	3M × 160b
Max IPv6 ACLs	2M × 320b
Update Rates	200 k/sec
Statistics	16M byte-packet pairs
Table-entry counters	For each table entry

NOTE: The number of Search and Stats lanes are independent.

Table 50: DNX 1xJR+ with BCM16K

RX/TX Data per Port	JR+ BCM88680 Device 0			
	Search		Stats	
	Port 0	Port 2	Port 0	Port 2
2 Lanes	1:0	17:16	–	–
4 Lanes	3:0	19:16	35:32	–
4 Lanes	3:0	19:16	35:32	43:40
6 Lanes	5:0	21:16	–	–
8 Lanes	7:0	23:16	–	–
10 Lanes	9:0	25:16	–	–
12 Lanes	11:0	27:16	–	–
16 Lanes	15:0	31:16	–	–

NOTE: The number of Search and Stats lanes are independent.

Table 51: DNX 1xJR2 with BCM16K

RX/TX Data per Port	JR2 BCM88690 Device 0			
	Search		Stats	
	Port 0	Port 2	Port 0	Port 2
2 Lanes	1:0	17:16	33:32	41:40
4 Lanes	3:0	19:16	35:32	43:40
6 Lanes	5:0	21:16	–	–
8 Lanes	7:0	23:16	–	–
10 Lanes	9:0	25:16	–	–
12 Lanes	11:0	27:16	–	–
16 Lanes	15:0	31:16	–	–

NOTE: The number of Search and Stats lanes are independent.

Table 52: DNX 1xJR2C with BCM16K

RX/TX Data per Port	JR2C BCM88880 Device 0		
	Search	Stats	
	Port 0	Port 0	Port 2
2 Lanes	1:0	33:32	–
2 Lanes	1:0	33:32	41:40
4 Lanes	3:0	35:32	–
4 Lanes	3:0	35:32	43:40
6 Lanes	5:0	–	–
8 Lanes	7:0	–	–

Table 52: DNX 1xJR2C with BCM16K (Continued.)

RX/TX Data per Port	JR2C BCM88880 Device 0		
	Search	Stats	
	Port 0	Port 0	Port 2
10 Lanes	9:0	–	–
12 Lanes	11:0	–	–
16 Lanes	15:0	–	–

Table 53: BCM88690 JR2 + BCM16K OP2 Connection Example with 53G SerDes

	Search Port 0 NIF or Fabric	Search Port 1 NIF or Fabric	Stats Port 0 NIF	Stats Port 1 NIF	Total NIF Used for OP2 Connect	Total Fabric Used for OP2 Connect	Comment
OP2 attach-option 1	8xNIF	8xNIF	4xNIF	4xNIF	24	0	Can use up to 12 53G lanes for Search Port 0 and 1. 10 53G lanes sufficient for 480b key.
OP2 attach-option 2	8xNIF	8xFabric	4xNIF	4xNIF	16	8	
OP2 attach-option 3	8xFabric	8xFabric	4xNIF	4xNIF	8	16	
OP2 attach-option 4	8xNIF	8xNIF	4xNIF	0	20	0	
OP2 attach-option 5	8xNIF	8xFabric	4xNIF	0	12	8	
OP2 attach-option 6	8xFabric	8xFabric	4xNIF	0	4	16	

Chapter 15: Power Control

This section highlights the Power Control (PC) options.

15.1 Overview

This feature limits the maximum dynamic power consumed by the device while maintaining the maximum number of database records. This is accomplished by limiting the maximum number of blocks that are simultaneously accessed during a search operation.

15.2 Usage

The Enhanced Power Control (EPC) searches are enabled in the LTR on a per-result basis. As such, it is possible to perform EPC searches in parallel with non-EPC searches. The enhanced power control level offers further power savings with one restriction, a maximum of 256 array blocks of 160b or greater width in the DBA can be enabled for enhanced power control compares. The SDK is required in order to use this feature.

The searches done at the regular power control level behave exactly like the non-EPC searches. There are no additional restrictions.

Compared to the previous generation of devices, this generation of power control provides up to four times improvement in enhanced power control.

For specific details on how to enable this feature using the SDK, contact your local Broadcom representative.

Chapter 16: Interlaken-LA Instruction Descriptions

16.1 Overview

This chapter contains the Interlaken Look-Aside instruction descriptions for both non-network byte ordering and network byte ordering.

16.2 Interlaken Look-Aside Descriptor Formats

In a packet, Control Words are transmitted first, followed by optional burst of Data Words. All bursts of Data Words must be transmitted in the order of DW0 first, and the number of Data Words is variable per the Interlaken-LA specifications with a minimum of two Data Word per packet.

The Control Word for each instruction contains fields defined by the Interlaken-LA specification. These includes bits 66:57, bits 40:39, and bit 32. For more information about these bits, see Table 54, Interlaken-LA Protocol Control Word Format for Request and Response Packets.

16.2.1 Example of Interlaken-LA Instruction

16.2.1.1 Control Word

66	65 64	63 0
INV	Framing	Control Word

16.2.1.2 Data Words

66	65 64	63 32	31 0
Data Word 0			
INV	Framing	Valid Data (LSB)	
Data Word 1			
INV	Framing	Valid Data	
Data Word 2			
INV	Framing	Valid Data (MSB)	32’b0

16.2.2 Interlaken-LA Control Word Format for Request and Response Packets

Interlaken Look-Aside (Interlaken-LA) Control Word for device request and response packets is described in this section. Interlaken LA specific fields are described in detail in Table 54. Device specific fields are described in detail in the following sections for each instruction.

Table 54: Interlaken-LA Protocol Control Word Format for Request and Response Packets

Bit	Field Name	Description
66	Inversion	Interlaken Look-Aside specified field. Used to indicate whether bits [63:0] have been inverted to limit the running disparity. 1'b1 = Inverted. 1'b0 = Not inverted.
65:64	Framing	Interlaken Look-Aside specified field. 64b/67b mechanism to distinguish control and data words. As per ILA spec, must be written with 10 for control words.
63	Control	Interlaken Look-Aside specified field. 1'b1 = This is an idle or burst control word. 1'b0 = This is a framing layer control word.
62	Type	Interlaken Look-Aside specified field. 1'b1 = The channel number and SOP fields are valid and a data burst follows this control word (a 'Burst Control Word'). 1'b0 = The channel number field and SOP fields are invalid and no data follows this control word (an 'Idle Control Word').
61	SOP	Interlaken Look-Aside specified field. Start of Packet. 1'b1 = The data burst following this control word represents the start of a data packet. 1'b0 = A data burst that follows this control word is either the middle or end of a packet.
60:57	EOP[3:0]	Interlaken Look-Aside specified field. This field refers to the data burst preceding this control word. 1xxx - End-of-Packet, with bits [59:57] defining the number of valid bytes in the last 8-byte word in the burst. Bits [59:57] are encoded such that 3'b000 means 8 bytes valid, 3'b001 means 1 byte valid, etc., with 3'b111 meaning 7 bytes valid. The valid bytes start with bit position [63:56]. 1'b0000 = No End-of-Packet, no ERR. 1'b0001 = Error and End-of-Packet. All other combinations are undefined.
56:41	ContextAddr[15:0]	Context Buffer Address – The address within the context buffer. For read and write instructions (except Context Buffer Write) this field is ignored. Contents of this field are passed from request to response control word unchanged. User may ignore this field or use as a tag.
40	Flow Control	Interlaken Look-Aside specified field. Flow Control - Flow control indication. 1'b1 = Flow control ON. 1'b0 = Flow control OFF.
39	Interlaken Look-Aside/ Protocol Type	Interlaken Look-Aside specified field. Must be written with 1 to indicate Interlaken Look-Aside protocol. If this bit is 0, this packet is ignored.
38:33	OpCode[5:0]	The OpCode[5:0] field works in conjunction with the OpCode[8:6] to encode the type of instruction to be executed. Refer to Table 56 on page 77 for a complete list of instructions and their respective encoding.
32	Channel Number	Interlaken Look-Aside specified field, channel number. Reserved, must be written with 0.

Table 54: Interlaken-LA Protocol Control Word Format for Request and Response Packets (Continued.)

Bit	Field Name	Description
31:30	Error Status	Error Status.
29:28	Reserved	Reserved, must write 0.
29	ContextAddr[15]	Context Buffer Address. The address within the context buffer. For read and write instructions (except Context Buffer Write), this field is ignored. Contents of this field are passed from request to response control word, unchanged. User may ignore this field or use as a tag.
28	Reserved	Reserved, must write 0.
27	LTR[6]	LTR[6] is the MSB of the Logical Table Register. Must be treated as OpCode[9] for non-compare instructions, and must be set to 0.
26:24	OpCode[8:6]	The OpCode[8:6] field works in conjunction with the OpCode[5:0] to encode the type of instruction to be executed. See Table 56 on page 77 for a complete list of instructions and their respective encoding.
23:0	CRC24	Interlaken Look-Aside specified field. A CRC error check that covers the previous data burst, if any, and this control word.

16.2.3 Interlaken-LA Data Word Format for Request and Response Packets

The Interlaken Look-Aside (Interlaken-LA) data word for request and response packets is described in this section. The Interlaken-LA specific fields are described in detail in Table 55. Device specific fields are described in detail in the following sections for each instruction.

Table 55: Interlaken-LA Data Word Format for Request and Response Packets

Bit	Field Name	Description
66	Inversion	Interlaken Look-Aside specified field. Used to indicate whether bits [63:0] have been inverted to limit the running disparity. 1'b1 = Inverted 1'b0 = Not inverted
65:64	Control/Data	Interlaken Look-Aside specified field. 64b/67b mechanism to distinguish between control and data words. This field must be set with 01 for data words.
63:0	Data/Reserved	Refer to each instruction section for details.

16.3 OpCode Definitions

Table 56: Instruction Set OpCodes and Definitions

OpCode [8:6]	LTR[6], OpCode [5:0]	Instruction	Description
3'b100	7'b0000000	Context Buffer Write	Performs write operation to Context Buffer location specified by context address.
3'b001	LTR[6:0]	Context Buffer Write and Compare	Writes data to Context Buffer location and executes a compare operation up to 640b wide.
3'b110	LTR[6:0]	Context Buffer No Write and Compare	Only uses existing data in Context Buffer location and executes a compare operation up to 640b wide. Incoming new search key is ignored and not written to the Context Buffer. Legacy opcodes [8:6] 3'b010, 3'b101 are supported for same operation. May be discontinued in future generations.
3'b111	LTR[6:0]	Context Buffer Skip and Compare	Only uses incoming search key to execute a compare operation up to 640b wide. Does not modify any Context Buffer Contents.
3'b000	0x6 ~ 0x7F	Management Instructions	Management instructions are used by Broadcom SDK.

16.4 Non-Network Byte Ordering Within Data Words

Table 57 describes the non-network data byte ordering within data words for the following instructions: *Context Buffer Write and Context Buffer Write and Compare*. For each of these instructions, the data can be any length in byte increments. The length of the data transfer is determined by the number of Data Words, and Control Word SOP and EOP fields up to 10 data words. The contents of each Data Word are transmitted in the order of bit 66 down to bit 0.

Table 57: Data Byte Ordering within Data Words

Data Word	66:64	63:56	55:48	47:40	39:32	31:24	23:16	15:8	7:0
DW 0	I-LA Sync bits	Data [7:0]	Data [15:8]	Data [23:16]	Data [31:24]	Data [39:32]	Data [47:40]	Data [55:48]	Data [63:56]
DW 1	I-LA Sync bits	Data [71:64]	Data [79:72]	Data [87:80]	Data [95:88]	Data [103:96]	Data [111:104]	Data [119:112]	Data [127:120]
DW 2	I-LA Sync bits	Data [135:128]	Data [143:136]	Data [151:144]	Data [159:152]	Data [167:160]	Data [175:168]	Data [183:176]	Data [191:184]
DW 3	I-LA Sync bits	Data [199:192]	Data [207:200]	Data [215:208]	Data [223:216]	Data [231:224]	Data [239:232]	Data [247:240]	Data [255:248]
DW 4	I-LA Sync bits	Data [263:256]	Data [271:264]	Data [279:272]	Data [287:280]	Data [295:288]	Data [303:296]	Data [311:304]	Data [319:312]
DW 5	I-LA Sync bits	Data [327:320]	Data [335:328]	Data [343:336]	Data [351:344]	Data [359:352]	Data [367:360]	Data [375:368]	Data [383:376]
DW 6	I-LA Sync bits	Data [391:384]	Data [399:392]	Data [407:400]	Data [415:408]	Data [423:416]	Data [431:424]	Data [439:432]	Data [447:440]
DW 7	I-LA Sync bits	Data [455:448]	Data [463:456]	Data [471:464]	Data [479:472]	Data [487:480]	Data [495:488]	Data [503:496]	Data [511:504]
DW 8	I-LA Sync bits	Data [519:512]	Data [527:520]	Data [535:528]	Data [543:536]	Data [551:544]	Data [559:552]	Data [567:560]	Data [575:568]
DW 9	I-LA Sync bits	Data [583:576]	Data [591:584]	Data [599:592]	Data [607:600]	Data [615:608]	Data [623:616]	Data [631:624]	Data [639:632]

16.5 Data Word to Context Buffer Mapping—Non-Network Byte Ordering

This section details the data word mapping over the Interlaken-LA interface to the Context Buffer for 80b, 160b, 240b, 320b, 400b, 480b, 560b, and 640b transfers.

Table 58: 640b Data Word to Context Buffer Mapping

Data Word	**63:56**	**55:48**	**47:40**	**39:32**	**31:24**	**23:16**	**15:8**	**7:0**
Data Word 0	7:0	15:8	23:16	31:24	39:32	47:40	55:48	63:56
Data Word 1	71:64	79:72	87:80	95:88	103:96	111:104	119:112	127:120
Data Word 2	135:128	143:136	151:144	159:152	167:160	175:168	183:176	191:184
Data Word 3	199:192	207:200	215:208	223:216	231:224	239:232	247:240	255:248
Data Word 4	263:256	271:264	279:272	287:280	295:288	303:296	311:304	319:312
Data Word 5	327:320	335:328	343:336	351:344	359:352	367:360	375:368	383:376
Data Word 6	391:384	399:392	407:400	415:408	423:416	431:424	439:432	447:440
Data Word 7	455:448	463:456	471:464	479:472	487:480	495:488	503:496	511:504
Data Word 8	519:512	527:520	535:528	543:536	551:544	559:552	567:560	575:568
Data Word 9	583:576	591:584	599:592	607:600	615:608	623:616	631:624	639:632

Context Buffer	**79:72**	**71:64**	**63:56**	**55:48**	**47:40**	**39:32**	**31:24**	**23:16**	**15:8**	**7:0**
Address 0	79:72	71:64	63:56	55:48	47:40	39:32	31:24	23:16	15:8	7:0
Address 1	159:152	151:144	143:136	135:128	127:120	119:112	111:104	103:96	95:88	87:80
Address 2	239:232	231:224	223:216	215:208	207:200	199:192	191:184	183:176	175:168	167:160
Address 3	319:312	311:304	303:296	295:288	287:280	279:272	271:264	263:256	255:248	247:240
Address 4	399:392	391:384	383:376	375:368	367:360	359:352	351:344	343:336	335:328	327:320
Address 5	479:472	471:464	463:456	455:448	447:440	439:432	431:424	423:416	415:408	407:400
Address 6	559:552	551:544	543:536	535:528	527:520	519:512	511:504	503:496	495:488	487:480
Address 7	639:632	631:624	623:616	615:608	607:600	599:592	591:584	583:576	575:568	567:560

Table 59: 560b Data Word to Context Buffer Mapping

Data Word	**63:56**	**55:48**	**47:40**	**39:32**	**31:24**	**23:16**	**15:8**	**7:0**
Data Word 0	7:0	15:8	23:16	31:24	39:32	47:40	55:48	63:56
Data Word 1	71:64	79:72	87:80	95:88	103:96	111:104	119:112	127:120
Data Word 2	135:128	143:136	151:144	159:152	167:160	175:168	183:176	191:184
Data Word 3	199:192	207:200	215:208	223:216	231:224	239:232	247:240	255:248
Data Word 4	263:256	271:264	279:272	287:280	295:288	303:296	311:304	319:312
Data Word 5	327:320	335:328	343:336	351:344	359:352	367:360	375:368	383:376
Data Word 6	391:384	399:392	407:400	415:408	423:416	431:424	439:432	447:440
Data Word 7	455:448	463:456	471:464	479:472	487:480	495:488	503:496	511:504
Data Word 8	519:512	527:520	535:528	543:536	551:544	559:552	0s	0s

Context Buffer	**79:72**	**71:64**	**63:56**	**55:48**	**47:40**	**39:32**	**31:24**	**23:16**	**15:8**	**7:0**
Address 0	79:72	71:64	63:56	55:48	47:40	39:32	31:24	23:16	15:8	7:0
Address 1	159:152	151:144	143:136	135:128	127:120	119:112	111:104	103:96	95:88	87:80
Address 2	239:232	231:224	223:216	215:208	207:200	199:192	191:184	183:176	175:168	167:160
Address 3	319:312	311:304	303:296	295:288	287:280	279:272	271:264	263:256	255:248	247:240
Address 4	399:392	391:384	383:376	375:368	367:360	359:352	351:344	343:336	335:328	327:320
Address 5	479:472	471:464	463:456	455:448	447:440	439:432	431:424	423:416	415:408	407:400
Address 6	559:552	551:544	543:536	535:528	527:520	519:512	511:504	503:496	495:488	487:480

Table 60: 480b Data Word to Context Buffer Mapping

Data Word	**63:56**	**55:48**	**47:40**	**39:32**	**31:24**	**23:16**	**15:8**	**7:0**
Data Word 0	7:0	15:8	23:16	31:24	39:32	47:40	55:48	63:56
Data Word 1	71:64	79:72	87:80	95:88	103:96	111:104	119:112	127:120
Data Word 2	135:128	143:136	151:144	159:152	167:160	175:168	183:176	191:184
Data Word 3	199:192	207:200	215:208	223:216	231:224	239:232	247:240	255:248
Data Word 4	263:256	271:264	279:272	287:280	295:288	303:296	311:304	319:312
Data Word 5	327:320	335:328	343:336	351:344	359:352	367:360	375:368	383:376
Data Word 6	391:384	399:392	407:400	415:408	423:416	431:424	439:432	447:440
Data Word 7	455:448	463:456	471:464	479:472	0s	0s	0s	0s

Context Buffer	**79:72**	**71:64**	**63:56**	**55:48**	**47:40**	**39:32**	**31:24**	**23:16**	**15:8**	**7:0**
Address 0	79:72	71:64	63:56	55:48	47:40	39:32	31:24	23:16	15:8	7:0
Address 1	159:152	151:144	143:136	135:128	127:120	119:112	111:104	103:96	95:88	87:80
Address 2	239:232	231:224	223:216	215:208	207:200	199:192	191:184	183:176	175:168	167:160
Address 3	319:312	311:304	303:296	295:288	287:280	279:272	271:264	263:256	255:248	247:240
Address 4	399:392	391:384	383:376	375:368	367:360	359:352	351:344	343:336	335:328	327:320
Address 5	479:472	471:464	463:456	455:448	447:440	439:432	431:424	423:416	415:408	407:400

Table 61: 400b Data Word to Context Buffer Mapping

Data Word	**63:56**	**55:48**	**47:40**	**39:32**	**31:24**	**23:16**	**15:8**	**7:0**		
Data Word 0	7:0	15:8	23:16	31:24	39:32	47:40	55:48	63:56		
Data Word 1	71:64	79:72	87:80	95:88	103:96	111:104	119:112	127:120		
Data Word 2	135:128	143:136	151:144	159:152	167:160	175:168	183:176	191:184		
Data Word 3	199:192	207:200	215:208	223:216	231:224	239:232	247:240	255:248		
Data Word 4	263:256	271:264	279:272	287:280	295:288	303:296	311:304	319:312		
Data Word 5	327:320	335:328	343:336	351:344	359:352	367:360	375:368	383:376		
Data Word 6	391:384	399:392	0s	0s	0s	0s	0s	0s		
Context Buffer	**79:72**	**71:64**	**63:56**	**55:48**	**47:40**	**39:32**	**31:24**	**23:16**	**15:8**	**7:0**
Address 0	79:72	71:64	63:56	55:48	47:40	39:32	31:24	23:16	15:8	7:0
Address 1	159:152	151:144	143:136	135:128	127:120	119:112	111:104	103:96	95:88	87:80
Address 2	239:232	231:224	223:216	215:208	207:200	199:192	191:184	183:176	175:168	167:160
Address 3	319:312	311:304	303:296	295:288	287:280	279:272	271:264	263:256	255:248	247:240
Address 4	399:392	391:384	383:376	375:368	367:360	359:352	351:344	343:336	335:328	327:320

Table 62: 320b Data Word to Context Buffer Mapping

Data Word	**63:56**	**55:48**	**47:40**	**39:32**	**31:24**	**23:16**	**15:8**	**7:0**		
Data Word 0	7:0	15:8	23:16	31:24	39:32	47:40	55:48	63:56		
Data Word 1	71:64	79:72	87:80	95:88	103:96	111:104	119:112	127:120		
Data Word 2	135:128	143:136	151:144	159:152	167:160	175:168	183:176	191:184		
Data Word 3	199:192	207:200	215:208	223:216	231:224	239:232	247:240	255:248		
Data Word 4	263:256	271:264	279:272	287:280	295:288	303:296	311:304	319:312		
Context Buffer	**79:72**	**71:64**	**63:56**	**55:48**	**47:40**	**39:32**	**31:24**	**23:16**	**15:8**	**7:0**
Address 0	79:72	71:64	63:56	55:48	47:40	39:32	31:24	23:16	15:8	7:0
Address 1	159:152	151:144	143:136	135:128	127:120	119:112	111:104	103:96	95:88	87:80
Address 2	239:232	231:224	223:216	215:208	207:200	199:192	191:184	183:176	175:168	167:160
Address 3	319:312	311:304	303:296	295:288	287:280	279:272	271:264	263:256	255:248	247:240

Table 63: 240b Data Word to Context Buffer Mapping

Data Word	**63:56**	**55:48**	**47:40**	**39:32**	**31:24**	**23:16**	**15:8**	**7:0**		
Data Word 0	7:0	15:8	23:16	31:24	39:32	47:40	55:48	63:56		
Data Word 1	71:64	79:72	87:80	95:88	103:96	111:104	119:112	127:120		
Data Word 2	135:128	143:136	151:144	159:152	167:160	175:168	183:176	191:184		
Data Word 3	199:192	207:200	215:208	223:216	231:224	239:232	0s	0s		
Context Buffer	**79:72**	**71:64**	**63:56**	**55:48**	**47:40**	**39:32**	**31:24**	**23:16**	**15:8**	**7:0**
Address 0	79:72	71:64	63:56	55:48	47:40	39:32	31:24	23:16	15:8	7:0
Address 1	159:152	151:144	143:136	135:128	127:120	119:112	111:104	103:96	95:88	87:80
Address 2	239:232	231:224	223:216	215:208	207:200	199:192	191:184	183:176	175:168	167:160

Table 64: 160b Data Word to Context Buffer Mapping

Data Word	**63:56**	**55:48**	**47:40**	**39:32**	**31:24**	**23:16**	**15:8**	**7:0**		
Data Word 0	7:0	15:8	23:16	31:24	39:32	47:40	55:48	63:56		
Data Word 1	71:64	79:72	87:80	95:88	103:96	111:104	119:112	127:120		
Data Word 2	135:128	143:136	151:144	159:152	0s	0s	0s	0s		
Context Buffer	**79:72**	**71:64**	**63:56**	**55:48**	**47:40**	**39:32**	**31:24**	**23:16**	**15:8**	**7:0**
Address 0	79:72	71:64	63:56	55:48	47:40	39:32	31:24	23:16	15:8	7:0
Address 1	159:152	151:144	143:136	135:128	127:120	119:112	111:104	103:96	95:88	87:80

Table 65: 80b Data Word to Context Buffer Mapping

Data Word	**63:56**	**55:48**	**47:40**	**39:32**	**31:24**	**23:16**	**15:8**	**7:0**		
Data Word 0	7:0	15:8	23:16	31:24	39:32	47:40	55:48	63:56		
Data Word 1	71:64	79:72	0s	0s	0s	0s	0s	0s		
Context Buffer	**79:72**	**71:64**	**63:56**	**55:48**	**47:40**	**39:32**	**31:24**	**23:16**	**15:8**	**7:0**
Address 0	79:72	71:64	63:56	55:48	47:40	39:32	31:24	23:16	15:8	7:0

If 64-bit context buffer write, then context buffer address 0 is updated as follows.

Table 66: Not a Multiple of 80-Bit and Less Than 80-Bit Data Word to Context Buffer Mapping for NNBO

Data Word	**63:56**	**55:48**	**47:40**	**39:32**	**31:24**	**23:16**	**15:8**	**7:0**		
Data Word 0	7:0	15:8	23:16	31:24	39:32	47:40	55:48	63:56		
Context Buffer	**79:72**	**71:64**	**63:56**	**55:48**	**47:40**	**39:32**	**31:24**	**23:16**	**15:8]**	**7:0**
Address 0	Random Data		63:56	55:48	47:40	39:32	31:24	23:16	15:8	7:0

If 96-bit context buffer write, then context buffer address 0 and 1 are updated as follows.

Table 67: Not a Multiple of 80-Bit and More Than 80-Bit Data Word to Context Buffer Mapping for NNBO

Data Word	**63:56**	**55:48**	**47:40**	**39:32**	**31:24**	**23:16**	**15:8**	**7:0**		
Data Word 0	7:0	15:8	23:16	31:24	39:32	47:40	55:48	63:56		
Data Word 1	71:64	79:72	87:80	95:88	0s	0s	0s	0s		
Context Buffer	**79:72**	**71:64**	**63:56**	**55:48**	**47:40**	**39:32**	**31:24**	**23:16**	**15:8]**	**7:0**
Address 0	[79:72]	[71:64]	[63:56]	[55:48]	[47:40]	[39:32]	[31:24]	[23:16]	[15:8]	[7:0]
Address 1	Random Data								[95:88]	[87:80]

16.6 Network Byte Ordering Within Data Words

Table 68 describes the network byte ordering within data words for the following instructions: *Context Buffer Write and Context Buffer Write and Compare*. For each of these instructions, the data can be any length in byte increments. The length of the data transfer is determined by the number of Data Words, and Control Word SOP and EOP fields. The contents of each Data Word is transmitted in the order of bit 66 down to bit 0.

For network byte ordering (NBO), data is always aligned to the MSB (closest 80b) of the Data Word. If only 64b of data is written, then the data is placed at data [79:16], data [15:0] is invalid and will be ignored.

Table 68: Network Byte Ordering Within Data Words

Data Word	66:64	63:56	55:48	47:40	39:32	31:24	23:16	15:8	7:0
DW 0	I-LA Sync bits	Data [639:632]	Data [631:624]	Data [623:616]	Data [615:608]	Data [607:600]	Data [599:592]	Data [591:584]	Data [583:576]
DW 1	I-LA Sync bits	Data [575:568]	Data [567:560]	Data [559:552]	Data [551:544]	Data [543:536]	Data [535:528]	Data [527:520]	Data [519:512]
DW 2	I-LA Sync bits	Data [511:504]	Data [503:496]	Data [495:488]	Data [487:480]	Data [479:472]	Data [471:464]	Data [463:456]	Data [455:448]
DW 3	I-LA Sync bits	Data [447:440]	Data [439:432]	Data [431:424]	Data [423:416]	Data [415:408]	Data [407:400]	Data [399:392]	Data [391:384]
DW 4	I-LA Sync bits	Data [383:376]	Data [375:368]	Data [367:360]	Data [359:352]	Data [351:344]	Data [343:336]	Data [335:328]	Data [327:320]
DW 5	I-LA Sync bits	Data [319:312]	Data [311:304]	Data [303:296]	Data [295:288]	Data [287:280]	Data [279:272]	Data [271:264]	Data [263:256]
DW 6	I-LA Sync bits	Data [255:248]	Data [247:240]	Data [239:232]	Data [231:224]	Data [223:216]	Data [215:208]	Data [207:200]	Data [199:192]
DW 7	I-LA Sync bits	Data [191:184]	Data [183:176]	Data [175:168]	Data [167:160]	Data [159:152]	Data [151:144]	Data [143:136]	Data [135:128]
DW 8	I-LA Sync bits	Data [127:120]	Data [119:112]	Data [111:104]	Data [103:96]	Data [95:88]	Data [87:80]	Data [79:72]	Data [71:64]
DW 9	I-LA Sync bits	Data [63:56]	Data [55:48]	Data [47:40]	Data [39:32]	Data [31:24]	Data [23:16]	Data [15:8]	Data [7:0]

16.7 Data Word to Context Buffer Mapping—Network Byte Ordering

This section details the data word mapping over the Interlaken-LA interface to the Context Buffer for 80b, 160b, 240b, 320b, 400b, 480b, 560b, and 640b transfers. The following data word to context buffer mappings are used in network byte ordering.

Table 69: 640b Data Word to Context Buffer Mapping

Data Word	63:56	55:48	47:40	39:32	31:24	23:16	15:8	7:0
Data Word 0	639:632	631:624	623:616	615:608	607:600	599:592	591:584	583:576
Data Word 1	575:568	567:560	559:552	551:544	543:536	535:528	527:520	519:512
Data Word 2	511:504	503:496	495:488	487:480	479:472	471:464	463:456	455:448
Data Word 3	447:440	439:432	431:424	423:416	415:408	407:400	399:392	391:384
Data Word 4	383:376	375:368	367:360	359:352	351:344	343:336	335:328	327:320
Data Word 5	319:312	311:304	303:296	295:288	287:280	279:272	271:264	263:256
Data Word 6	255:248	247:240	239:232	231:224	223:216	215:208	207:200	199:192
Data Word 7	191:184	183:176	175:168	167:160	159:152	151:144	143:136	135:128
Data Word 8	127:120	119:112	111:104	103:96	95:88	87:80	79:72	71:64
Data Word 9	63:56	55:48	47:40	39:32	31:24	23:16	15:8	7:0

Context Buffer	79:72	71:64	63:56	55:48	47:40	39:32	31:24	23:16	15:8	7:0
Address 0	79:72	71:64	63:56	55:48	47:40	39:32	31:24	23:16	15:8	7:0
Address 1	159:152	151:144	143:136	135:128	127:120	119:112	111:104	103:96	95:88	87:80
Address 2	239:232	231:224	223:216	215:208	207:200	199:192	191:184	183:176	175:168	167:160
Address 3	319:312	311:304	303:296	295:288	287:280	279:272	271:264	263:256	255:248	247:240
Address 4	399:392	391:384	383:376	375:368	367:360	359:352	351:344	343:336	335:328	327:320
Address 5	479:472	471:464	463:456	455:448	447:440	439:432	431:424	423:416	415:408	407:400
Address 6	559:552	551:544	543:536	535:528	527:520	519:512	511:504	503:496	495:488	487:480
Address 7	639:632	631:624	623:616	615:608	607:600	599:592	591:584	583:576	575:568	567:560

Table 70: 560b Data Word to Context Buffer Mapping

Data Word	63:56	55:48	47:40	39:32	31:24	23:16	15:8	7:0		
Data Word 0	559:552	551:544	543:536	535:528	527:520	519:512	511:504	503:496		
Data Word 1	495:488	487:480	479:472	471:464	463:456	455:448	447:440	439:432		
Data Word 2	431:424	423:416	415:408	407:400	399:392	391:384	383:376	375:368		
Data Word 3	367:360	359:352	351:344	343:336	335:328	327:320	319:312	311:304		
Data Word 4	303:296	295:288	287:280	279:272	271:264	263:256	255:248	247:240		
Data Word 5	239:232	231:224	223:216	215:208	207:200	199:192	191:184	183:176		
Data Word 6	175:168	167:160	159:152	151:144	143:136	135:128	127:120	119:112		
Data Word 7	111:104	103:96	95:88	87:80	79:72	71:64	63:56	55:48		
Data Word 8	47:40	39:32	31:24	23:16	15:8	7:0	0s	0s		
Context Buffer	**79:72**	**71:64**	**63:56**	**55:48**	**47:40**	**39:32**	**31:24**	**23:16**	**15:8**	**7:0**
Address 0	79:72	71:64	63:56	55:48	47:40	39:32	31:24	23:16	15:8	7:0
Address 1	159:152	151:144	143:136	135:128	127:120	119:112	111:104	103:96	95:88	87:80
Address 2	239:232	231:224	223:216	215:208	207:200	199:192	191:184	183:176	175:168	167:160
Address 3	319:312	311:304	303:296	295:288	287:280	279:272	271:264	263:256	255:248	247:240
Address 4	399:392	391:384	383:376	375:368	367:360	359:352	351:344	343:336	335:328	327:320
Address 5	479:472	471:464	463:456	455:448	447:440	439:432	431:424	423:416	415:408	407:400
Address 6	559:552	551:544	543:536	535:528	527:520	519:512	511:504	503:496	495:488	487:480

Table 71: 400b Data Word to Context Buffer Mapping

Data Word	63:56	55:48	47:40	39:32	31:24	23:16	15:8	7:0		
Data Word 0	399:392	391:384	383:376	375:368	367:360	359:352	351:344	343:336		
Data Word 1	335:328	327:320	319:312	311:304	303:296	295:288	287:280	279:272		
Data Word 2	271:264	263:256	255:248	247:240	239:232	231:224	223:216	215:208		
Data Word 3	207:200	199:192	191:184	183:176	175:168	167:160	159:152	151:144		
Data Word 4	143:136	135:128	127:120	119:112	111:104	103:96	95:88	87:80		
Data Word 5	79:72	71:64	63:56	55:48	47:40	39:32	31:24	23:16		
Data Word 6	15:8	7:0	0s	0s	0s	0s	0s	0s		
Context Buffer	**79:72**	**71:64**	**63:56**	**55:48**	**47:40**	**39:32**	**31:24**	**23:16**	**15:8**	**7:0**
Address 0	79:72	71:64	63:56	55:48	47:40	39:32	31:24	23:16	15:8	7:0
Address 1	159:152	151:144	143:136	135:128	127:120	119:112	111:104	103:96	95:88	87:80
Address 2	239:232	231:224	223:216	215:208	207:200	199:192	191:184	183:176	175:168	167:160
Address 3	319:312	311:304	303:296	295:288	287:280	279:272	271:264	263:256	255:248	247:240
Address 4	399:392	391:384	383:376	375:368	367:360	359:352	351:344	343:336	335:328	327:320

Table 72: 320b Data Word to Context Buffer Mapping

Data Word	63:56	55:48	47:40	39:32	31:24	23:16	15:8	7:0
Data Word 0	319:312	311:304	303:296	295:288	287:280	279:272	271:264	263:256
Data Word 1	255:248	247:240	239:232	231:224	223:216	215:208	207:200	199:192
Data Word 2	191:184	183:176	175:168	167:160	159:152	151:144	143:136	135:128

Table 72: 320b Data Word to Context Buffer Mapping (Continued.)

Data Word	**63:56**	**55:48**	**47:40**	**39:32**	**31:24**	**23:16**	**15:8**	**7:0**		
Data Word 3	127:120	119:112	111:104	103:96	95:88	87:80	79:72	71:64		
Data Word 4	63:56	55:48	47:40	39:32	31:24	23:16	15:8	7:0		
Context Buffer	**79:72**	**71:64**	**63:56**	**55:48**	**47:40**	**39:32**	**31:24**	**23:16**	**15:8**	**7:0**
Address 0	79:72	71:64	63:56	55:48	47:40	39:32	31:24	23:16	15:8	7:0
Address 1	159:152	151:144	143:136	135:128	127:120	119:112	111:104	103:96	95:88	87:80
Address 2	239:232	231:224	223:216	215:208	207:200	199:192	191:184	183:176	175:168	167:160
Address 3	319:312	311:304	303:296	295:288	287:280	279:272	271:264	263:256	255:248	247:240

Table 73: 240b Data Word to Context Buffer Mapping

Data Word	**63:56**	**55:48**	**47:40**	**39:32**	**31:24**	**23:16**	**15:8**	**7:0**		
Data Word 0	239:232	231:224	223:216	215:208	207:200	199:192	191:184	183:176		
Data Word 1	175:168	167:160	159:152	151:144	143:136	135:128	127:120	119:112		
Data Word 2	111:104	103:96	95:88	87:80	79:72	71:64	63:56	55:48		
Data Word 3	47:40	39:32	31:24	23:16	15:8	7:0	0s	0s		
Context Buffer	**79:72**	**71:64**	**63:56**	**55:48**	**47:40**	**39:32**	**31:24**	**23:16**	**15:8**	**7:0**
Address 0	79:72	71:64	63:56	55:48	47:40	39:32	31:24	23:16	15:8	7:0
Address 1	159:152	151:144	143:136	135:128	127:120	119:112	111:104	103:96	95:88	87:80
Address 2	239:232	231:224	223:216	215:208	207:200	199:192	191:184	183:176	175:168	167:160

Table 74: 160b Data Word to Context Buffer Mapping

Data Word	**63:56**	**55:48**	**47:40**	**39:32**	**31:24**	**23:16**	**15:8**	**7:0**		
Data Word 0	159:152	151:144	143:136	135:128	127:120	119:112	111:104	103:96		
Data Word 1	95:88	87:80	79:72	71:64	63:56	55:48	47:40	39:32		
Data Word 2	31:24	23:16	15:8	7:0	0s	0s	0s	0s		
Context Buffer	**79:72**	**71:64**	**63:56**	**55:48**	**47:40**	**39:32**	**31:24**	**23:16**	**15:8**	**7:0**
Address 0	79:72	71:64	63:56	55:48	47:40	39:32	31:24	23:16	15:8	7:0
Address 1	159:152	151:144	143:136	135:128	127:120	119:112	111:104	103:96	95:88	87:80

Table 75: 80b Data Word to Context Buffer Mapping

Data Word	**63:56**	**55:48**	**47:40**	**39:32**	**31:24**	**23:16**	**15:8**	**7:0**		
Data Word 0	79:72	71:64	63:56	55:48	47:40	39:32	31:24	23:16		
Data Word 1	15:8	7:0	0s	0s	0s	0s	0s	0s		
Context Buffer	**79:72**	**71:64**	**63:56**	**55:48**	**47:40**	**39:32**	**31:24**	**23:16**	**15:8**	**7:0**
Address 0	79:72	71:64	63:56	55:48	47:40	39:32	31:24	23:16	15:8	7:0

If 64-bit context buffer write, then context buffer address 0 is updated as follows.

Table 76: Not a Multiple of 80-Bit and Less than 80-Bit Data Word to Context Buffer Mapping for NBO

Data Word	**63:56**	**55:48**	**47:40**	**39:32**	**31:24**	**23:16**	**15:8**	**7:0**		
Data Word 0	63:56	55:48	47:40	39:32	31:24	23:16	15:8	7:0		
Context Buffer	**79:72**	**71:64**	**63:56**	**55:48**	**47:40**	**39:32**	**31:24**	**23:16**	**15:8**	**7:0**
Address 0	63:56	55:48	47:40	39:32	31:24	23:16	15:8	7:0	Random Data	

If 96-bit context buffer write, then context buffer addresses 0 and 1 are updated as follows.

Table 77: Not a Multiple of 80-Bit and More than 80-Bit Data Word to Context Buffer Mapping for NBO

Data Word	**63:56**	**55:48**	**47:40**	**39:32**	**31:24**	**23:16**	**15:8**	**7:0**		
Data Word 0	95:88	87:80	79:72	71:64	63:56	55:48	47:40	39:32		
Data Word 1	31:24	23:16	15:8	7:0	0s	0s	0s	0s		
Context Buffer	**79:72**	**71:64**	**63:56**	**55:48**	**47:40**	**39:32**	**31:24**	**23:16**	**15:8**	**7:0**
Address 0	15:8	7:0	Random Data							
Address 1	95:88	87:80	79:72	71:64	63:56	55:48	47:40	39:32	31:24	23:16

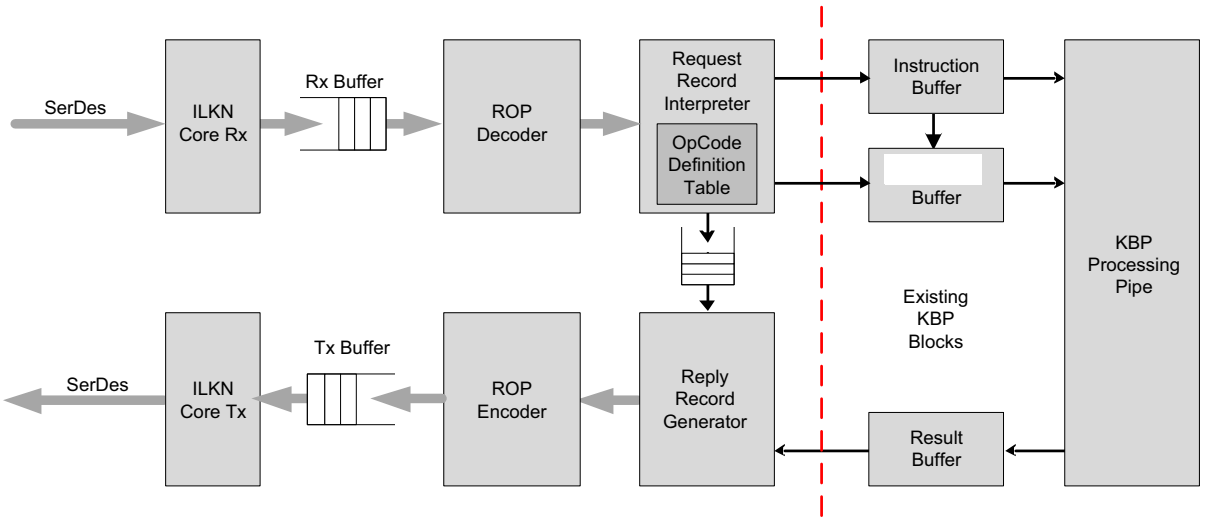
Chapter 17: ROP Layer and Interlaken Controller

17.1 General Overview

The Interlaken controller block diagram describes, at a high level, the Records-Over-Packet (ROP) layer implementation with the Interlaken PCS layer.

The Interlaken (ILKN) Core RX block with the ROP Decoder block extracts the OpCode, sequence number, and packet data. The Request Record Interpreter determines what operations to perform on each packet. After leaving the Request Record Interpreter, the packet is passed to the core KBP blocks for read, write, or search operations. Upon leaving the KBP core Result Buffer, the Reply Record Generator and ROP Encoder blocks correctly format packets for transmission via the Interlaken (ILKN) Core TX block.

Figure 29: Interlaken Controller/ROP Functional Diagram



17.2 Interlaken Controller Overview

The Interlaken controller features and functions are described in this section.

17.2.1 Interlaken Controller Feature Set

The interlaken controller features are as follows:

- The interlaken device supports two host processors with two ports per host
- Lane configuration options single port:
  - 4x RX/TX, lanes 3:0 only
  - 8x RX/TX, lanes 7:0 only
  - 12x RX/TX, lanes 11:0 only
  - 16x RX/TX, lanes 15:0 only
- Lane configuration options dual port
  - 4x RX/TX per port, lanes 21:18 for port 1 and lanes 3:0 for port 0
  - 8x RX/TX per port, lanes 25:18 for port 1 and lanes 7:0 for port 0
  - 12x RX/TX per port, lanes 29:18 for port 1 and lanes 11:0 for port 0
  - 16x RX/TX per port, lanes 33:18 for port 1 and lanes 15:0 for port 0

  Port 0 and Port 1 lane configuration can be different
- 4 lane Increments
- RX and TX lanes can be configured independently
- Lane reversal and polarity inversion supported
- Traffic sent only on Channel 0

SerDes Rate

- Minimum SerDes rate is 12.5 Gb/s
- Maximum SerDes rate is 56.25 Gb/s
- In dual port mode, both ports must operate at same SerDes rate.

Packet

- BurstShort = 32B per Interlaken specification.
- BurstMax = 256B per Interlaken specification.
  RX Interlaken core discards any packet received that violates the BurstMax value
- All packets must be sent in a single Burst
- RX Interlaken checks that EOP and SOP fields of a burst are both asserted, and if not, the packet is considered in error (contents are discarded even if the CRC-24 is correct)
- TX Interlaken guarantees the SOP or EOP or both SOP and EOP fields in the burst control word are set wherever applicable
- Every packet is a single burst, BurstMin applies that is, 32B

Channel and Backpressure

- Single Channel supported
  For example, channel 0 represented in Burst Control Word by Channel Number field set to all zeros. Any Burst Control Word received with Channel Number set to nonzero value is discarded by the receiver.
- TX Interlaken guarantees the SOP or EOP or both SOP and EOP fields in the burst control word are set wherever applicable
- TX asserts reset calendar bit in all transmitted control word
- TX asserts Xoff for all other channels (Channel 1 to 15)

- TX controller assert Xoff to host when RX FIFO full condition or after power-up when the core is not ready
- KBP receiver, channel 0 Xoff bit is detected when “reset calendar bit is set”
- Multiple-use field must not be used to extend either Flow Control or Channel Number information and thus channel number field and in-band flow control field are both 8-bit values
- KBP receiver, Xoff is extracted from bit 55 of IL Control Word. On Transmit side, Flow Control information (or Xoff) is sent on IL Control Word bit 55. Xoff and channel information sent on bits [54:40] will be ignored at the RX side.

Metaframe

- Maximum Metaframe Length = 8K words
- Default Metaframe Length is 2K words and identical for both Receive and Transmit
- TX Interlaken only send a single SKIP word
- RX Interlaken supports the reception of additional SKIP word and silently discards them. The minimum gap between Skip Words within a MetaFrame must be ≥ 16 Data Words.

Clocking, PPM, and Skew

- Interlaken supports the independent clocking scheme between Host and KBP with the PPM limit specified in Interlaken Specification
- RX Interlaken controller supports up to 214 UI, value recommended by Interlaken interop specification for 25.78125 Gb/s speed (CEI-28G-SR/MR)
- TX Interlaken controller skew between lanes is guaranteed to be lower than 40 UI (lower than 67 UI budget for PMA TX)

  The total UI for the KBP chip output is 160. (The budget is 214 UI for speed > 5G). The internal UI distribution is shown in Table 78.

Table 78: UI Skew

SerDes	QHMF	CLK to QHMF	Final UI
1 lane	1 cycle	2.45 ns	
40 UI	40 UI	80 UI	160 UI

Features Not Supported

- TX Interlaken does not implement the Rate Limiter feature
- Interlaken controller does not implement retransmit
- KBP does not implement sequential instructions
- KBP does not implement re-entrant instructions
- Interlaken controller does not implement EEI (Energy Efficient Interlaken)
- Multiple-use field of Interlaken Burst Control Word is ignored on RX
- The diagnostic Word status bits[33:32] is driven to all 1s by default on the transmit side
- Diagnostic word status bits are as per IL specification. Lane and Link health are transmitted.
- No padding between records within a ROP packet. As soon as a pad is seen, the record extraction will stop and the remaining ROP packet will be discarded.
- No padding at the start of the packet. If so, that ROP packet will be discarded.
- Padding is only allowed towards the end of the ROP packet.
- No Support for INFO records.
- No support for Copy Data.
- No support for Key_cfg.
- Two modes are supported.
  - Mode 1: With 16b ContextId in the payload.
  - Mode 0: Without ContextId.

Link-up Sequence

- Interlaken Controller does not power up automatically and requires control plane software accessing the KBP interface controller through MDIO/PCIe to configure the SerDes as well as the Interlaken Controller before enabling the Interlaken training.
- The Interlaken controller implements a complete register space accessible through the MDIO/PCIe registers for link-up, debug, statistics, and more.

17.2.1.1 Interlaken Controller Performance

This section highlights the Interlaken-ROP layer performance, latency, and potential limitations.

On the receive side, the Interlaken ROP is able to sustain the full bandwidth of Interlaken protocol without stalling the Host (assuming that the KBP Core logic does not stall the Interface receiver). We support one record extraction per core clock per port and are able to sustain single cycle extraction at 900 MPPS.

17.2.1.2 Interlaken Descriptor Formats

Within a packet, Control Words are transmitted first followed by an optional burst of Data Words (DW). All bursts of Data Words must be transmitted in order with DW0 first, and the number of Data Words is variable per the Interlaken specifications with a minimum of one Data Word on RX and four Data Words on TX per packet.

The Control Word for each instruction contains fields defined by the Interlaken specification. These include bits 66:57, bits 40:39, and bit 32. For more information about these bits see Table 79 on page 93.

17.2.1.2.1 Example of Interlaken Instruction

17.2.1.2.2 Control Word

66	65:64	63	0
INV	Framing	Control Word	

17.2.1.2.3 Data Words

66	65:64	63	32	31	0
Data Word 0					
INV	Framing	Valid Data (LSB)			
Data Word 1					
INV	Framing	Valid Data			
Data Word 2					
INV	Framing	Valid Data (MSB)		32’b0	

17.2.1.2.4 Interlaken Control Word Format for Request and Response Packets

The Interlaken Control Word for device request and response packets is described in this section. Interlaken-specific fields are described in detail in Table 79 on page 93. Device specific fields are described in detail in the following sections for each instruction.

Table 79: Interlaken Control Word Format for Request and Response Packets

Bit	Field Name	Description
66	Inversion	Interlaken-specified field Used to indicate whether bits [63:0] have been inverted to limit the running disparity 1’b1 = Inverted 1’b0 = Not inverted

Table 79: Interlaken Control Word Format for Request and Response Packets (Continued.)

Bit	Field Name	Description
65:64	Framing	Interlaken-specified field 64b/67b mechanism to distinguish control and data words Must be written with 10 for control words
63	Control	Interlaken-specified field 1’b1 = This is an idle or burst control word 1’b0 = This is a framing layer control word
62	Type	Interlaken-specified field 1’b1 = The channel number and SOP fields are valid and a data burst follows this control word (a ‘Burst Control Word’) 1’b0 = The channel number field and SOP fields are invalid and no data follows this control word (an ‘Idle Control Word’)
61	SOP	Interlaken-specified field Start of Packet 1’b1 = The data burst following this control word represents the start of a data packet 1’b0 = A data burst that follows this control word is either the middle or end of a packet
60:57	EOP[3:0]	Interlaken-specified field This field refers to the data burst preceding this control word 1xxx - End-of-Packet, with bits [59:57] defining the number of valid bytes in the last 8-byte word in the burst. Bits [59:57] are encoded such that 3’b000 means 8 bytes valid, 3’b001 means 1 byte valid, etc., with 3’b111 meaning 7 bytes valid. The valid bytes start with bit position [63:56]. 1’b0000 = No End-of-Packet, no ERR 1’b0001 = Error and End-of-Packet All other combinations are undefined
56	Reset Calender	Reset Calender
55	Xon/Xoff	Xon/Xoff indication
54:24	Reserved	Reserved
23:0	CRC24	Interlaken-specified field. A CRC error check that covers the previous data burst, if any, and this control word.

17.2.1.2.5 Interlaken Data Word Format for Request and Response Packets

The Interlaken data word for request and response packets is described in this section. Device specific fields are described in detail in the following sections for each instruction.

Table 80: Interlaken Data Word Format for Request and Response Packets

Bit	Field Name	Description
66	Inversion	Interlaken-specified field. Used to indicate whether bits [63:0] have been inverted to limit the running disparity. 1'b1 = Inverted 1'b0 = Not inverted
65:64	Control/Data	Interlaken-specified field. 64b/67b mechanism to distinguish between control and data words. This field must be written with 01 for data words.
63:0	Data/Reserved	Refer to each instruction section for details

17.3 ROP Layer Overview

The ROP protocol is used to facilitate efficient exchange of small records over relatively large packets, synchronize replies to requests, and protect against packet/record loss.

8-bit OpCode

An 8-bit OpCode points to a Lookup Table (LUT) descriptor that contains all the information a sender or recipient requires about the record. The information includes the record type and the size of the data portion. OpCodes are synchronized between both sides of RX and TX. OpCode '0' is reserved to designate an empty record having no sequence number and no data. Thus, inserting '0' padding before or between a record within a ROP packet is illegal (called padding). Inserting '0' padding after all records in a packet is legal.

In reply records, the OpCode is copied from the corresponding Request record.

The ROP layer Lookup Table (LUT) contains a 19-bit LUT Descriptor field.

16-bit Sequence Number

The 16-bit Sequence Number is a field that is present following the OpCode of the first Request record and the first Reply record within a given packet, which is used to protect against record loss.

All Request records are assigned an incrementing sequence number. This number is explicitly inserted in packets for the first Request record.

All Reply records have a sequence number that is equal to the corresponding Request record's number. Replies within one packet have an incrementing sequence number, thus only the first Reply record's sequence number is explicitly inserted. In the event of a lost Request/Reply, a packet may have to be truncated (and possibly padded) to insure that all records within a packet have incrementing Sequence-Number.

- A field following the OpCode of the first Request record and the first Reply record within a given packet.
- Used to protect against records loss.
- All Request records are assigned an incrementing sequence number. This number is inserted in packets for the first Request record (the remainder can be deduced).
- All Reply records have a sequence number that is equal to the corresponding Request record's number. Replies within one packet have an incrementing sequence number, thus only the first Reply record's sequence number is explicitly inserted. In the event of a lost Request/Reply, a packet may have to be truncated (and possibly padded) to ensure that all records within a packet have incrementing Sequence-Number.
- If a record is lost, there is a discontinuity in the Sequence-Number. If the discontinuity falls in the middle of a packet, the packet must be truncated and possibly padded, and a new packet started. The new packet has the sequence number of the record after the lost record.

Data

Data is variable in size, and contains additional information according to the OpCode. For example, in a Request, the data field contains the lookup keys information. While in a Reply, the data field contains the lookup result.

Lookup Reply Record Data is flexibly constructed from:

- Match status
- Match Index Associated Data

Table 81 is an example format of a ROP Request Packet with Various Search Key Sizes. All padding has a value of '0', consistent with the interpretation of the '0' (NULL) OpCode. The above example shows the padding between the Blue record and the Green record and at the end of packet.

Table 81: ROP Packet Format Example

	66:64	63:56	55:48	47:40	39:32	31:24	23:16	15:8	7:0
CW	IL Synch	Interlaken: Control, Type, SOP, EOP, CRC24							
DW0	IL Synch	OpCode	Seq 15:8	Seq 7:0	Data 159:152	Data 151:144	Data 143:136	Data 135:128	Data 127:120
DW1	IL Synch	Data 119:112	Data 111:104	Data 103:96	Data 95:88	Data 87:80	Data 79:72	Data 71:64	Data 63:56
DW2	IL Synch	Data 55:48	Data 47:40	Data 39:32	Data 31:24	Data 23:16	Data 15:8	Data 7:0	OpCode
DW3	IL Synch	Data 79:72	Data 71:64	Data 63:56	Data 55:48	Data 47:40	Data 39:32	Data 31:24	Data 23:16
DW4	IL Synch	Data 15:8	Data 7:0	OpCode	Data 639:632	Data 631:624	Data 623:616	Data 615:608	Data 607:600
DW5	IL Synch	Data 599:592	Data 591:584	Data 583:576	Data 575:568	Data 567:560	Data 559:552	Data 551:544	Data 543:536
DW6	IL Synch	Data 535:528	Data 527:520	Data 519:512	Data 511:404	Data 503:496	Data 495:488	Data 487:480	Data 479:472
DW7	IL Synch	Data 471:464	Data 463:456	Data 455:448	Data 447:440	Data 439:432	Data 431:424	Data 423:416	Data 415:408
DW8	IL Synch	Data 407:400	Data 399:392	Data 391:384	Data 383:376	Data 375:368	Data 367:360	Data 359:352	Data 351:344
DW9	IL Synch	Data 343:336	Data 335:328	Data 327:320	Data 319:312	Data 311:304	Data 303:296	Data 295:288	Data 287:280
DW10	IL Synch	Data 279:272	Data 271:264	Data 263:256	Data 255:248	Data 247:240	Data 239:232	Data 231:224	Data 223:216
DW11	IL Synch	Data 215:208	Data 207:200	Data 199:192	Data 191:184	Data 183:176	Data 175:168	Data 167:160	Data 159:152
DW12	IL Synch	Data 151:144	Data 143:136	Data 135:128	Data 127:120	Data 119:112	Data 111:104	Data 103:96	Data 95:88
DW13	IL Synch	Data 87:80	Data 79:72	Data 71:64	Data 63:56	Data 55:48	Data 47:40	Data 39:32	Data 31:24
DW14	IL Synch	Data 23:16	Data 15:8	Data 7:0	OpCode	Data 79:72	Data 71:64	Data 63:56	Data 55:48
...									
DW30	IL Synch	Data 95:88	Data 87:80	Data 79:72	Data 71:64	Data 63:56	Data 55:48	Data 47:40	Data 39:32
DW31	IL Synch	Data 31:24	Data 23:16	Data 15:8	Data 7:0	Pad 0s	Pad 0s	Pad 0s	Pad 0s

17.3.1 ROP Layer Performance

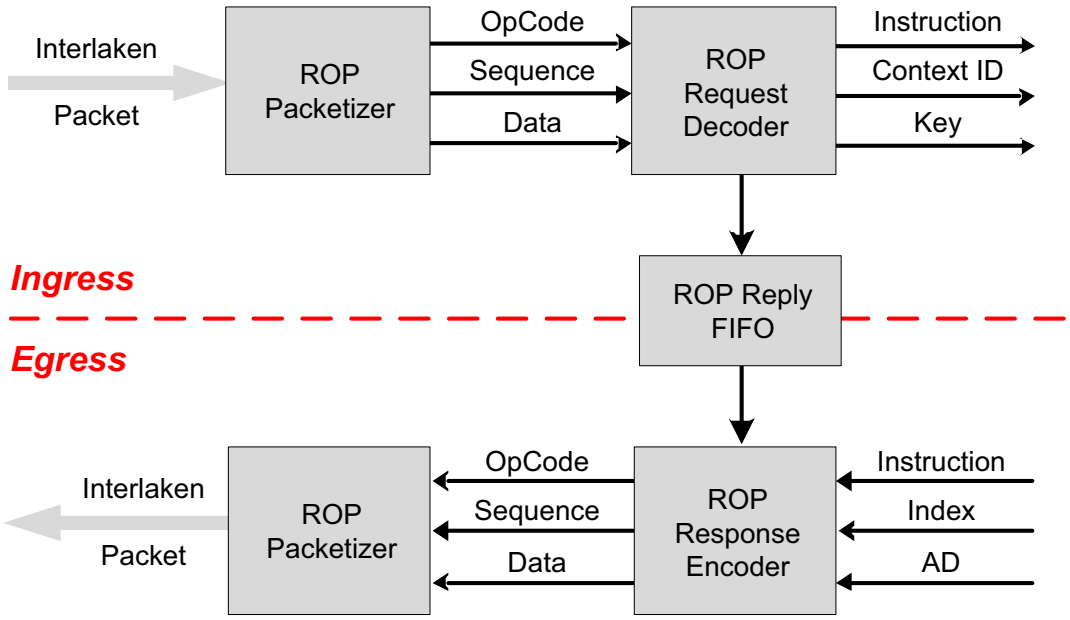
The ROP Extractor layer can extract one record every clock cycle at Core clock up to 640 bits from the Interlaken RX layer

The following restrictions apply.

- ROP Layer LUT is programmed using LUT WR instruction. The Host must wait for LUT WR Response before start using the newly programmed entry.
- The search key is always MSB aligned on 80b boundary if the key is not a multiple of 80b.

17.3.2 ROP Ingress and Egress Flow

Figure 30: ROP Ingress and Egress Flow



For ROP ingress, the ROP Packetizer extracts (unpacks) records received from the Interlaken RX and provides the OpCode, sequence number, and data to the ROP Request Decoder, that passes the instruction context ID and search key to the KBP core.

For ROP egress, the KBP core passes the instruction, index, and/or associated data to the ROP Response Encoder. LTR programming determines if match index or associated data is sent to the host. LTR programming also determines how responses are packed.

When a response comes from the KBP core, the SeqNum from the core and the SeqNum stored in the ROP reply FIFO are checked. If the ROP Reply FIFO contains no parity error and the SeqNums match, replies are constructed according to the core information. If the sequence numbers mismatch, the current ROP packet accumulation is closed and sent to the host. If parity error is detected, an ERROR record is returned to the host.

17.3.2.1 ROP Request Decoder Lookup Table (LUT)

The ROP Request Decoder LUT gives the size of the record (S) and record valid state. The ROP Request Decoder LUT also has a 1b Mode field with the Instruction. In Mode 0, the 10b Instruction is taken directly from the LUT entry. The 15b ContextId is fixed. The Record Data (RecData) contains up to 640b of Search Key data and must be byte aligned. Mode field is programmed in LUT Table.

Up to 640b Search Key is constructed based on the LUT programming Record Data[S-1:0] where S = maximum size of a record.

17.3.2.2 ROP OpCodes

This section describes the OpCodes that are supported by ROP Layer:

Table 82: Complete Instruction List Supported for ROP

Fields	Request Size	Response Size
Hard-Coded OpCodes		
NOP	OpCode: 1B Record Payload: 16B of 0	OpCode: 1B Record Payload: 1B of data
ERROR (250)	N/A	OpCode: 1B Record Payload: 8B of error status
Variable OpCodes		
Compare, Up To Eight Results	OpCode: 1B Record Payload: Up to 80B of key + 2B of ContextId	OpCode: 1B Record Payload: 2B (for 1 Result, No Index, 1B AD) to 128B (for 8 results and 128b of AD each)

Table 83: ROP LUT Entry

Fields	Range	Definition
RecSize	6:0	Record Size, up to 640b data.
RecValid	7	Record is Valid. 1’b1 = Valid 1’b0 = Invalid
Mode	8	Reserved
Instruction	18:9	Instruction bits.

17.3.2.3 Instruction Set Definitions

The following table contains the Instruction Set definitions.

Table 84: Instruction Set Definitions

Instr[9:0] {Instr[9], Instr[8:6], Instr[5:0]}			Instruction	Description
0	3’b000	6’b000000	NOP	Not supported. Defaults to 0.
LTR[6]	3’b010	LTR[5:0]	Compare	Executes a compare up to 640b wide

17.4 Instruction Formats

This section provides request and response formatting for each compare instruction.

17.4.1 CMP Instruction

The format of the CMP instruction is as follows:

NOTE: The maximum Key size (Data) is 80B.

The following example shows a 10B Instruction:

66:64	63:56	55:48	47:40	39:32	31:24	23:16	15:8	7:0
IL-Sync Bits	OpCode [7:0]	SeqNum [15:8]	SeqNum [7:0]	Data [79:72]	Data [71:64]	Data [63:56]	Data [55:48]	Data [47:40]
IL-Sync Bits	Data [39:32]	Data [31:24]	Data [23:16]	Data [15:8]	Data [7:0]	NxtRec OpCode	–	–

Table 85: CMP Request Fields

Field	Description
IL-Sync	Interlaken sync bits
OpCode[7:0]	Compare OpCode (variable) as set up in the LUT table
SeqNum[15:0]	Request sequence number
Data[79:0]	Data to be compared (10B in the example – max 80B)
NxtRec OpCode	Start of the next record OpCode

17.4.2 CMP Response Format

The format of the CMP Response is as follows:

The response size can be up to 113B for eight results in a search. The following example shows a reply with seven results. R0, R2, R4, R6 are configured to send 24b Index and R1, R3, R5 are configured to send 128b AD.

	66:64	63:56	55:48	47:40	39:32	31:24	23:16	15:8	7:0
DW0	IL-Sync	OpCode [7:0]	SeqNum [15:8]	SeqNum [7:0]	MF Status [7:0] =11111110	IDX0 [23:16]	IDX0 [15:8]	IDX0 [7:0]	AD1 [127:120]
DW1	IL-Sync	AD1 [119:112]	AD1 [111:104]	AD1 [103:96]	AD1 [95:88]	AD1 [87:80]	AD1 [79:72]	AD1 [71:64]	AD1 [63:56]
DW2	IL-Sync	AD1 [55:48]	AD1 [47:40]	AD1 [39:32]	AD1 [31:24]	AD1 [23:16]	AD1 [15:8]	AD1 [7:0]	IDX2 [23:16]
DW3	IL-Sync	IDX2 [15:8]	IDX2 [7:0]	AD3 [127:120]	AD3 [119:112]	AD3 [111:104]	AD3 [103:96]	AD3 [95:88]	AD3 [87:80]
DW4	IL-Sync	AD3 [79:72]	AD3 [71:64]	AD3 [63:56]	AD3 [55:48]	AD3 [47:40]	AD3 [39:32]	AD3 [31:24]	AD3 [23:16]
DW5	IL-Sync	AD3 [15:8]	AD3 [7:0]	IDX4 [23:16]	IDX4 [15:8]	IDX4 [7:0]	AD5 [127:120]	AD5 [119:112]	AD5 [111:104]
DW6	IL-Sync	AD5 [103:96]	AD5 [95:88]	AD5 [87:80]	AD5 [79:72]	AD5 [71:64]	AD5 [63:56]	AD5 [55:48]	AD5 [47:40]
DW7	IL-Sync	AD5 [39:32]	AD5 [31:24]	AD5 [23:16]	AD5 [15:8]	AD5 [7:0]	IDX6 [23:16]	IDX6 [15:8]	IDX6 [7:0]
DW8	IL-Sync	NxtRec OpCode	–	–	–	–	–	–	–

Table 86: CMP Response Fields

Response Field	Definition
IL-Sync	Interlaken sync bits
Opcode[7:0]	Opcode of Request Record
SeqNum[15:0]	Response sequence number that matches the corresponding Request sequence number
MF Status[7:0]	Match Flag 0= Match was not found 1= Match was found and Index and AD are Valid
IDX[23:0]	Match Index is valid only if MF bit is ‘1’
AD[nn:0]	Associated data is valid only if MF is ‘1’. Size of AD depends on Result size requested.
NxtRec OpCode	Start of the next response OpCode

17.4.3 ERROR Record

The record size is 10B.

The Error Record is sent if eight_rslt_en is set to 1 and any compare instruction encounters an error.

Whenever an error is encountered in the ROP layer during the processing of a ROP record OR if the record encounters any kind of error in the KBP Core, the following ERROR record (fixed opcode = 250) is sent back to the Host. The response payload portion would contain the error status information. The request opcode that encountered this error will be sent as part of the Error response payload.

Table 87: When ERROR Record is the Only Record in a Packet

	66:64	63:56	55:48	47:40	39:32	31:24	23:16	15:8	7:0
DW0	IL-Sync	OpCode = 250	SeqNum [15:8]	SeqNum [7:0]	InSeqNum [15:8]	InSeqNum [7:0]	InOpCode [7:0]	Err Sts [55:48]	Err Sts [47:40]
DW1	IL-Sync	Err Sts [39:32]	Err Sts [31:24]	Err Sts [23:16]	Err Sts [15:8]	Err Sts [7:0]	NxtRec OpCode	–	–

Table 88: When ERROR Record is the Last Record in a Packet

	66:64	63:56	55:48	47:40	39:32	31:24	23:16	15:8	7:0
DW0	IL-Sync	OpCode = 250	InSeqNum [15:8]	InSeqNum [7:0]	InOpCode [7:0]	Err Sts [55:48]	Err Sts [47:40]	Err Sts [39:32]	Err Sts [31:24]
DW1	IL-Sync	Err Sts [23:16]	Err Sts [15:8]	Err Sts [7:0]	NxtRec OpCode	–	–	–	–

Table 89: ERROR Response Fields

Response Field	Definition
IL-Sync	Interlaken sync bits
OpCode[7:0]	250
SeqNum[15:0]	Response sequence number.
InSeqNum[15:0]	Incoming packet sequence number that generated the error.
InOpCode[7:0]	Incoming OpCode that generated the error.
Err Sts[55:48]	Bits[7:1]: Always 0 Bit[0]: Reply FIFO Error
Err Sts[47:0]	This bits can also be retrieved from Error Status Register. A summary on the Interface related errors is provided in Table 90. For additional details, refer to the Registers document or contact Broadcom application team.
NxtRec OpCode	Start of the next response OpCode.

Table 90: Summary on Interface related errors (ERROR Response)

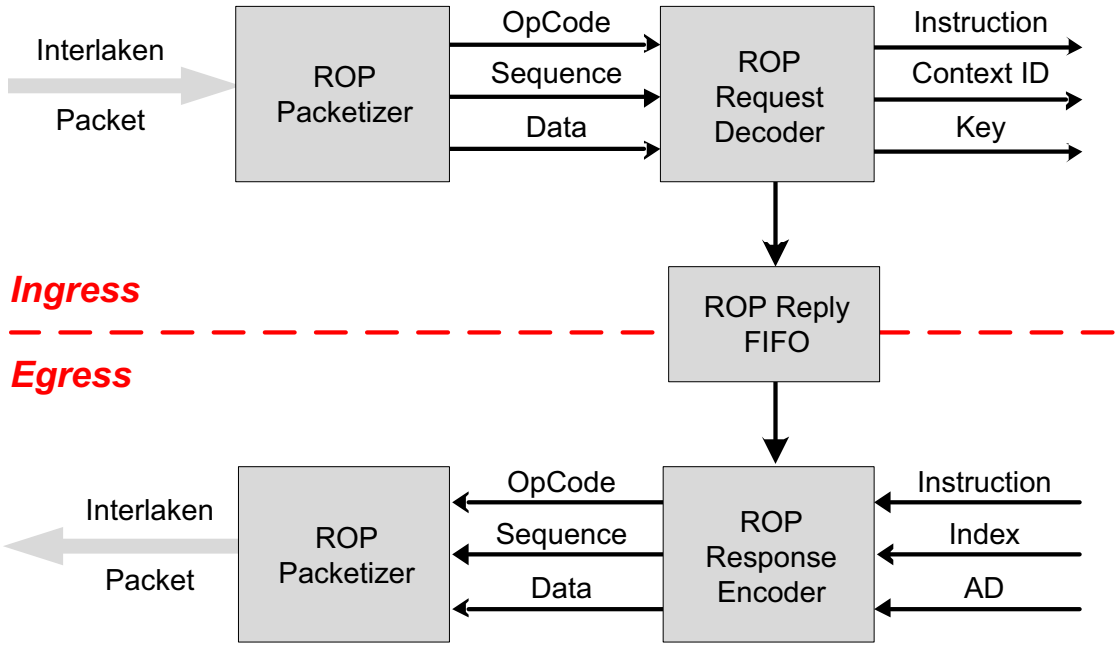
Err Sts[17]	Host Transmit Interface Error
Err Sts[16]	Host Receive Interface Error
Err Sts[15:12]	Reserved
Err Sts[11]	Instruction Length Error on Receive Interface
Err Sts[10]	Illegal Instruction Error on Receive Interface.

Table 90: Summary on Interface related errors (ERROR Response) (Continued.)

Err Sts[09]	Illegal End Of Packet (EOP) setting in the Control Word Error on Receive Interface.
Err Sts[08]	Burst Max Error on Receive Interface.
Err Sts[07]	Burst Short Error on Receive Interface.
Err Sts[06]	Protocol Error on Receive Interface.
Err Sts[05]	Missing Start of Packet (SOP) in Burst Control Word Error on Receive Interface.
Err Sts[04]	MAC FIFO Parity Error on Receive Interface.
Err Sts[03]	Missing End of Packet (EOP) in Burst Control Word Error on Receive Interface.
Err Sts[02]	Framing (ILA-Sync) Error on Receive Interface.
Err Sts[01]	CRC24 Error on Receive Interface.
Err Sts[00]	Reserved

17.4.4 ROP Egress Flow

Figure 31: ROP Egress Flow



On egress, the KBP core passes the instruction, index, and/or associated data to the ROP Response Encoder. LTR programming determines if match index and/or associated data are sent to the host. LTR programming also determines how responses are packed.

Chapter 18: Single and Dual-Host Mode

18.1 Overview

In a dual-host implementation, the processor device is connected to two independent hosts. These ports can be either two separate host devices, or a single host device with two ports. Both ports must operate at same SerDes rate.

In dual-host mode, the external SerDes channels are divided into two groups. The number of SerDes channels in each group depends on the system configuration.

18.2 Dual-Port Mode

Figure 32 and Figure 33 show a possible dual-host configuration. In this mode, two independent hosts are connected to a single device. Host Port 0 and Port 2 have access to both Core 0 and Core 1.

Figure 32: Dual Port with Single Host

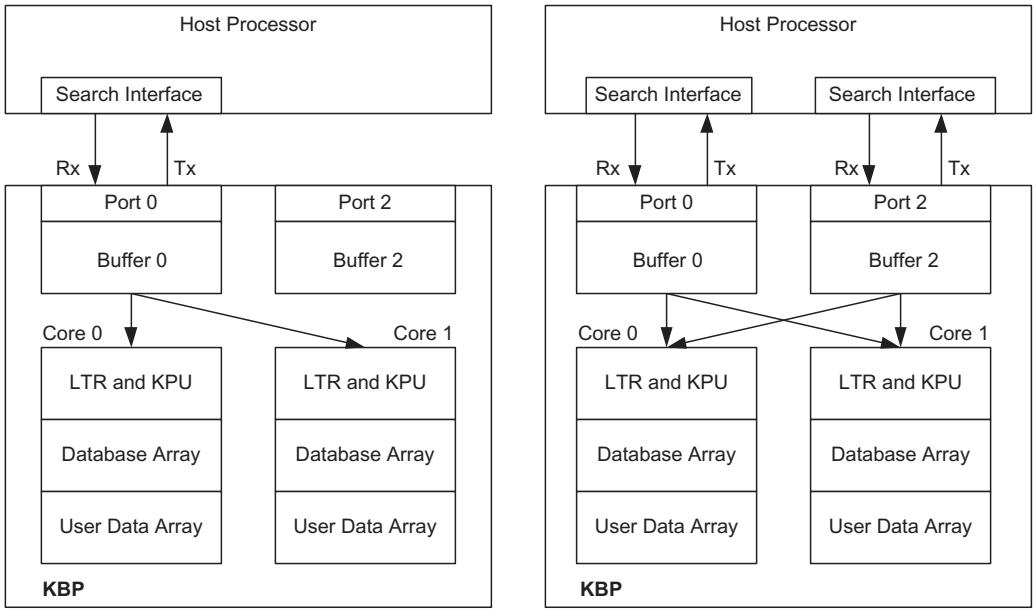
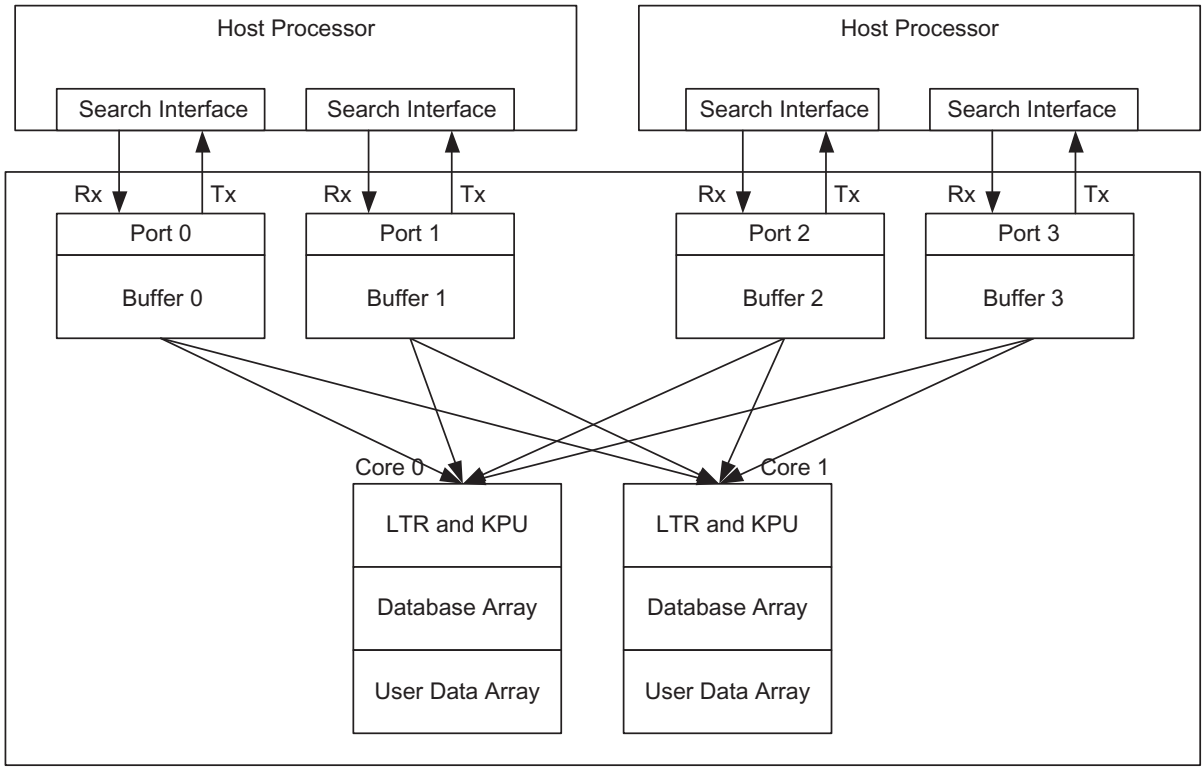


Figure 33: Dual Port with Dual Host



18.3 Search Lane Assignments in Single-Host Mode

This section describes lane configurations for host device connections. Each port can be configured to different lane configuration.

Table 91: Single Host Single Port

	Host 0
Lanes Per Port	**Port 0**
4 lanes	3:0
6 lanes	5:0
8 lanes	7:0
10 lanes	9:0
12 lanes	11:0
16 lanes	15:0

Table 92: Single Host Dual Port

Lanes Per Port	Host 0	
	Port 0	Port 1
4 lanes	3:0	19:16
6 lanes	5:0	21:16
8 lanes	7:0	23:16
10 lanes	9:0	25:16
12 lanes	11:0	27:16
16 lanes	15:0	31:16

Table 93: Search Lane Assignments in Dual-Host Mode

Lanes Per Host	Host 0		Host 1	
	Port 0	Port 1	Port 0	Port 1
8 lanes	3:0	11:8	19:16	27:24
10 lanes	4:0	12:8	20:16	28:24
12 lanes	5:0	13:8	21:16	29:24
16 lanes	7:0	15:8	23:16	31:24

18.4 Interface Feature Summary

Table 94: Interface Feature Summary

Features	Description
Number of Lanes	*Single Port:* Maximum of 16 lanes; [15:0] *Dual Port*: Maximum of 16 lanes per port. *Quad Port:* Maximum of 8 lanes per port.
Lane Swapping	M and N values are a subset of the number of lane configurations mentioned. Single Port (16 lanes maximum) Lane [N] becomes lane [15 – N], where N = 0 to 15. Dual Host (16 lanes each maximum) Port0: Lane [N] becomes lane [15 – N], where N = 0 to 15 Port1: Lane [M + 16] becomes lane [31 – M], where M = 0 to 15 Quad Port Port0: Lane [N] becomes lane [15 – M], where M = 0 to 7 Port1: Lane [M + 8] becomes lane [15 – M], where M = 0 to 7 Port2: Lane [P + 16] becomes lane [23 – P], where P = 0 to 7 Port3: Lane [S + 24] becomes lane [31 – S], where S = 0 to 7 The TX and RX swapping have independent control.
Skew (Lanes)	Transmit skew between all enabled lanes is guaranteed to be lower than 174 UI. RX skew between all enabled lanes is tolerable to maximum of 174 UI (standard requirement is 428 UI for 56G).
Speeds	53.125 and 56.25 Gb/s
Protocol	Interlaken, IL EOP byte count transmitted in the IL control word has byte granularity. Allows for packetization of multiple response into one packet. 1 skip word sent on TX. N skip words accepted on RX per meta frame. Programmable MetaFrameLength up to 8K words. References: ■ Interlaken Protocol Definition (v1.2) ■ Interlaken Clarification (v1.1)
Response Related	Zeroes out the 24b Index and 1B to 31B AD for a Valid Miss Result. The rest of the fields in the Index Status Byte remain untouched (if only IndexOrAD is used, the AD/Index field is still zeroed out).
Ordering	All responses are in strict order irrespective of interface mode.
Ports	Both ports must have same speed.
Configurability	SerDes lanes, see Number of Lane feature above.
	Metaframe A programmable MetaFrame to guarantee lane alignment and scrambler synchronization. Programmable MetaFrameLength. (up to 8K IL/ILA words, each word being 67b without FEC, with FEC enabled 65b).
	MDIO/PCIe interface for configuration and status reporting.

Table 94: Interface Feature Summary (Continued.)

Features	Description
Robustness	64b/67b data decoding and descrambling Forward Error Correction (FEC) when using PAM-4 signaling CRC32 check for lane integrity respectively Transmit and Receive MAC reply FIFO are Parity protected
Flow Control	In-band flow control

18.5 Interlaken FEC

The processor supports an optional RS-FEC on the Interlaken Interface for search. The FEC is compliant to the FEC extension of the Interlaken Reed-Solomon Forward Error Correction Protocol Definition from the Interlaken Alliance. The processors Forward Error Correction (FEC) is expected give acceptable BER when using PAM-4 signaling at high speeds. The FEC is turned on via software for PAM-4 signaling.

The following FEC usage guidelines apply to the search interface:

- For the search interface using a SerDes rate of up to and including 28.125 Gb/s in NRZ mode, FEC is not supported.
- For the search interface using a SerDes rate above 28.125 Gb/s in PAM-4 mode, FEC is mandatory.

18.6 Context Buffer Organization in Dual Host Mode

The device has two context buffers each 5120 x 640b. Port 0 and 1 access context buffer 0 and port 2 and 3 access context buffer 1. By default, an access to context 0 by the host connected to port 0 and port 2 accesses entry 0 of the context buffer.

When the host device connected to port 2 and 3 sends an instruction on context 0, the location in the context buffer that is accessed depends on the programming, see icf_crb_dir_DEVICE_CONFIG_REG - DEVICE_CONFIG_REG in the *KBP 16 nm Gen1 Registers Map* (see References).

The default for this 3-bit field is 3’b100, which configures the context buffer as follows:

- Host device 0 accesses context buffer entries 5120 x 640b
- Host device 1 accesses context buffer entries 5120 x 640b

Note that host 0 always has access to the entire 5120 x 640b entries in the context buffer. However, if a block of context buffer entries are dedicated to another host or port, there is no need for host 0 to require access to these entries.

Therefore, when programmed with the default value, if host 1 sends a packet on context 0, location 0 of the context buffer is accessed.

The following formula can be used to determine the starting address of the context buffer for port 1.

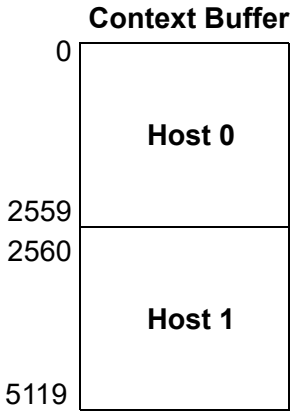
Context Buffer starting address for port 1 = 640 x FieldValue

For example, if this field is programmed with a value of 3'b100, the starting address for port 1 in the context buffer would be:

Context Buffer starting address for port 1 = 640 x 4 = 2560

In this configuration, the entries in the context buffer would be assigned as shown in Figure 34.

Figure 34: Context Buffer Entry Assignments—Port 1 Context ID = 3’b100

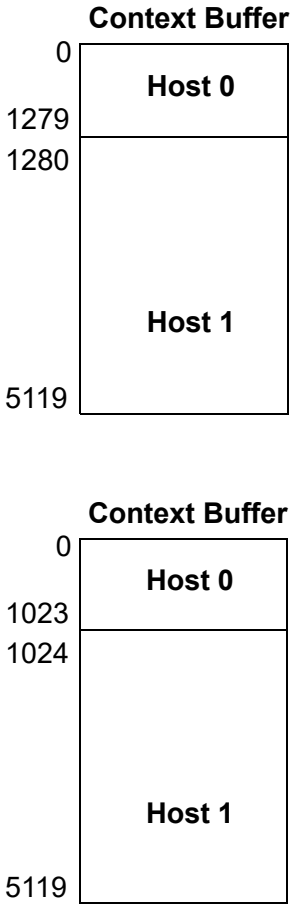


To increase the number of context buffer entries allocated to host 1, the value programmed can be decreased from the default value of 3’b100. For example, if this field is programmed with a value of 3'b100, for more information, see icf_crb_dir_DEVICE_CONFIG_REG - DEVICE_CONFIG_REG in the *KBP 16 nm Gen1 Registers Map* (see References), the context buffer entries for host 1 would be calculated as follows:

Context Buffer starting address for port 1 = 640 x 2 = 1280

In this configuration, the entries in the context buffer would be assigned as shown in Figure 35. Host 0 is assigned 1279:0 context buffer entries, and host 1 is assigned 5119:1280 entries.

Figure 35: Context Buffer Entry Assignments—Port 1 Context ID = 3’b010



Note that if the value programmed is 3’b000, the starting address for port 1 in the context buffer would be:

Context Buffer starting address for port 1 = 640 x 0 = 0

If this occurs, the possibility of overlapping contexts exists.

18.7 Overlapping Contexts in Dual-Port Mode

When the device is configured in dual-port mode, the possibility exists that both host devices may access the same context buffer entry at the same time. As stated above, this can occur when bits 50:48 of the *Device Configuration* register are set to 0 and a transfer on context 0 is initiated by both host devices in the same clock cycle. It is up to the application software to ensure that this scenario never occurs. However, should this condition occur, the device will issue an error packet that will be broadcast onto both ports simultaneously. Note that assigning both host devices the same context buffer entries does not result in the generation of an error packet as this is a valid configuration. In addition, both hosts may want to access the same context buffer entry on different clock cycles. This also does not result in the generation of an error packet and can be considered a valid operation depending on the overall system application. It is only when both devices access the same context buffer entry in the same clock cycle that an error will be generated.

Chapter 19: Device Configuration

19.1 Overview

The device uses the MDIO interface for communication with the host via a 2-wire MDIO port. This section provides a brief overview of the MDIO physical interface and also provides configuration information, registers definitions, and register programming.

19.2 MDIO Physical Interface

The Management Data Input Output (MDIO) port can be used to configure the device to exercise certain test modes and to check the status of the device. This interface is consistent with IEEE 802.3ae Clause 45. For additional information on the MDIO interface, refer to IEEE 802.3ae, Clause 45.

The MDIO interface has two signals: Management Data Clock (MDC) and Management Data Input/Output (MDIO). MDIO has specific terminology to define various devices on the bus. The device driving the MDIO bus is identified as the Station Management Entity (STA). The target devices that are managed by the MDC are referred to as MDIO Manageable Devices (MMD).

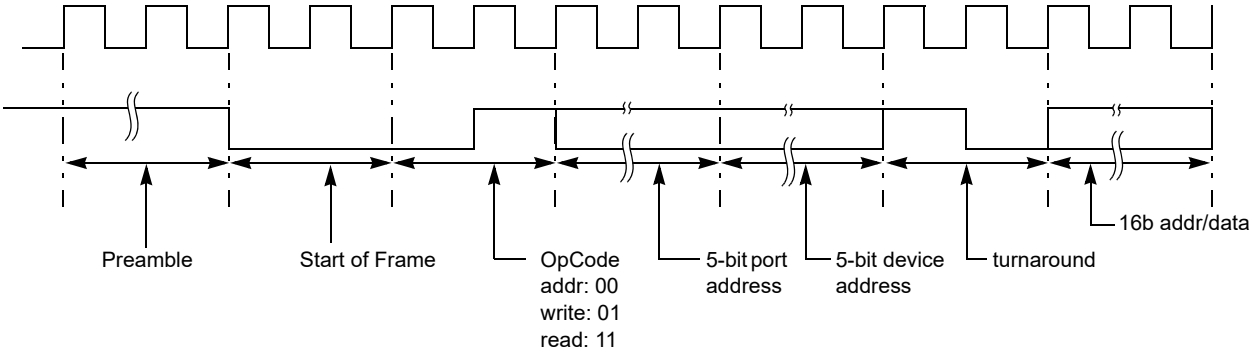
The STA initiates all communication in MDIO and is responsible for driving the clock on MDC. MDIO is specified to have a frequency of up to 2.5 MHz.

Three transactions are supported:

- Address
- Write Data
- Read Data

Figure 36 shows a representative MDIO transaction. Each transaction is 64 MDC clock cycles long and contains 32 cycles preamble, 2 cycles start-of-frame, 2 cycles OpCode, 5 cycles port address, 5 cycles device address, 2 cycles turnaround, and 16 cycles of address/data.

Figure 36: MDIO Transaction Format



19.2.1 MDIO Address Transaction

During every address transaction the register address is stored along with OpCode, port address, and device address. The MDIO address is set equal to the value on the incoming MDIO pin. The device supports MDIO device addresses 1 to 11, 16, 17, 18, and 20. The port address is determined by the MPID pins. Valid MPID pin settings are 0 to 23. Do not set the MPID to 24 to 31. For the selected MPID N, this device uses MPID numbers N through N + 8. If sharing MDIO pins with other devices, ensure that those devices avoid MPIDs N through N + 8.

19.2.2 MDIO Write Transaction

During a write MDIO transaction, as soon as the write data is received and stored into the 16 bits of the status register, a one-shot transaction follows where the data is written to the address specified in the address register.

19.2.3 MDIO Read Transaction

During a read MDIO transaction, after the port address is received, the MDIO logic does a one-shot transaction where it reads the data from the location specified in the address register. This falls into the Hi-Z part of the turnaround cycles. After the turnaround is complete, the MDIO logic drives the MDIO bus with data from the status register.

Chapter 20: Device Usage Restrictions

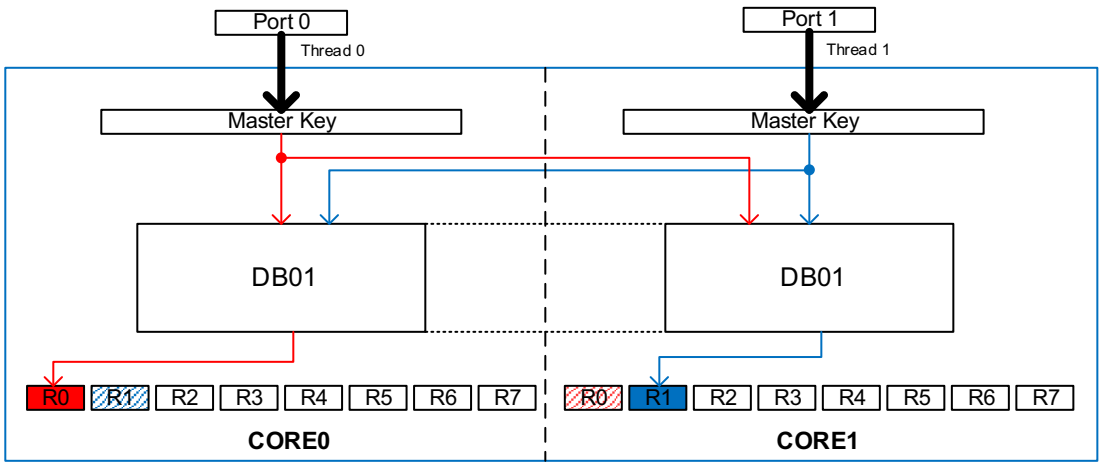
20.1 Overview

In the configuration example shown in Figure 27, Jericho 2 (BCM88690) connects to OP2 over two ports with an effective total packet rate throughput at 2000 MPPS. In this configuration, the two cores of OP2 are used in parallel to connect the two input ports ports to the output result ports. Concurrent searches over two threads are performed over the two cores to achieve 2000 MPPS performance target. As mentioned in Table 43, each of these threads support a parallelism of 8 (in other words, one can have 8 parallel searches per thread over two cores producing 8 results per port at the output). A typical allocation technique uses database replication in the cores to meet the performance target. Under such a condition, all result ports are available for the threads making maximum usage of parallel searches possible. This chapter deals with the situations where a number of parallel searches are restricted to fewer than 8 per thread because of database distribution as well as usage of counters (Search 'N Count). Note that the restrictions being discussed in this chapter are not applicable to Jericho 2C or Jericho+ connectivity with BCM16K where a single thread is in use. Also note that the diagrams in this chapter are meant to show the high-level relationship during the search and do not necessarily show the hardware paths available within silicon. For example, the arrows connecting Master Key to a specific database show the database being searched per the instruction received.

20.2 Single Database Spanning Across the Cores

Depending on the size (for example, scale or capacity) of a particular database required, an extra-large database may be allocated across the cores to better optimize the overall capacity available on the device. Under such a condition, a restriction kicks in with regards to the availability of result ports which translates to a reduction in the number of searches possible per core. Figure 37 provides a simplistic view of a case where a single database that spans across the cores is being searched from two threads concurrently for the same Master Key provided through the two input ports.

Figure 37: Single Extra Large Database Spanning Across Cores

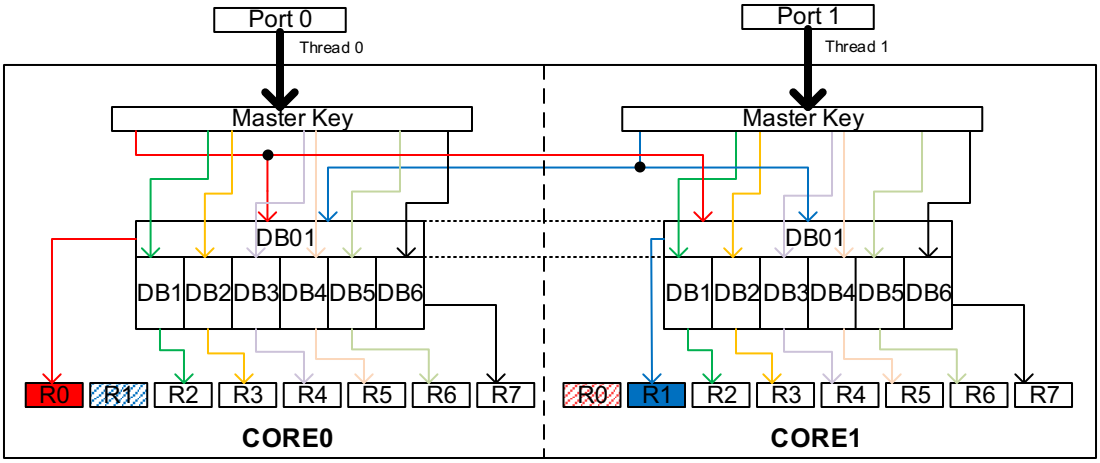


Single Extra Large Database Spanning Across Cores
Example: DB01=FWD

Since the database is shared between the cores, Master Key from one thread must be checked against the database instances on both cores. In the figure above, red and blue arrows show that relationship. The result for Red arrow search must be provided to the result port of the core where the thread was being executed which, in this case, is R0 of CORE0. The same is true for the blue arrow search where the final result would show up on result port R1 of CORE1. As the final result is required in the core where a particular thread is running, silicon hardware requires the same result port number to be mapped on the 2nd core, as the temporary data holding port before the search result from the 2nd core goes back to the 1st port as final output. This requirement is shown in the color-specific shaded result ports in Figure 37.

In summary, when a database is spread over two cores, each search of that database requires two result ports per thread instead of one. The second result port in a particular core is required to hold the temporary search result for a search initiated by the 2nd core before being transferred to its final destination result port. Due to this temporary hold requirement, the number of available result ports decreases while the number of parallel searches possible decreases. Figure 38 below shows the same example used earlier in conjunction with additional databases to be searched. When a single database spans over two cores, the total number of searches possible per thread (also per port) becomes 7 instead of 8. This figure, just like Figure 37, shows the condition where result port R1 on CORE0 and R0 on CORE1 are being used to hold temporary results while all other database searches require one result port each. In other words, the number of parallel searches possible through a Master Key is a function of database allocation and overall scale requirements.

Figure 38: Single Extra Large Database Spanning Across Cores – Total 7 Databases

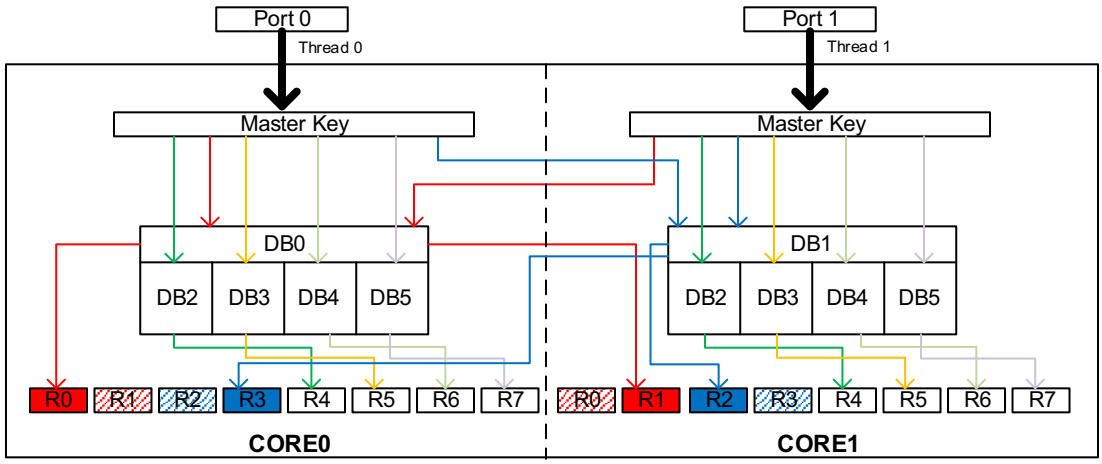


Single Extra Large Database Spanning Across Cores – Total 7 Databases
Example: DB01=FWD, DB1=RPF, DB2-DB6=ACL

20.3 Single Database Residing On One Core, But Accessed From Two Parallel Threads

While Single Database Spanning Across the Cores specifically addresses a database spanning across the cores, it is also possible to have two different databases allocated to one core each – again for overall scale requirement. Figure 39 shows such a situation. In this case, the same restriction, described above, kicks in on the core where a particular database exists. In Figure 39, if two databases DB0 (in CORE0) and DB1 (in CORE1) are searched simultaneously, then the Master Keys being presented to the two cores, are limited to 6 searches as each core uses two result ports for temporary result storage. In Figure 39, red and blue arrows show the search and result relationship. Red arrow shows DB0 searched from both Thread0 and Thread1 and the final result for the search operation going to CORE0-R0 and CORE1-R1 respectively with CORE0-R1 and CORE1-R0 being used as temporary result holding ports. Similarly, Blue arrow shows DB1 searched from both Thread0 and Thread1 and the final result for the search operation going to CORE0-R3 and CORE1-R2, respectively, with CORE0-R2 and CORE1-R3 being used as temporary result holding ports.

Figure 39: One Database in Each Core Shared with Second Core – Total 6 Databases



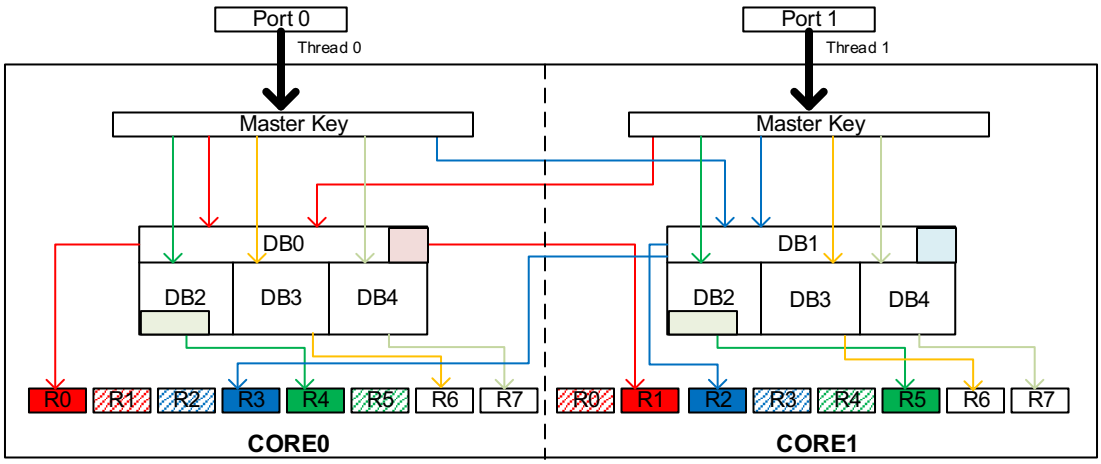
One Database in Each Core shared with 2nd core – Total 6 Databases
Example: DB0=FWD, DB1=RPF, DB2-DB5=ACL

The example shown in Figure 39 provides a general rule – the number of searches possible per Master Key decreases by the number of shared databases.

20.4 Effect of Search 'N Count on Number of Possible Searches When Any Database is Shared Between the Threads

Single Database Spanning Across the Cores and Single Database Residing On One Core, But Accessed From Two Parallel Threads addressed restrictions related to searches only. When counters are present on the searched database, another restriction is possible. Figure 40 shows such a condition. When a single database is shared between two cores, all databases that have counters associated with them require an additional result port for temporary search result storage. As shown in Figure 40, DB0, DB1, and DB2 have counters associated with them (represented by a colored box within the database). Because of the rule mentioned in Single Database Residing On One Core, But Accessed From Two Parallel Threads, the number of parallel searches would have been limited to 6 in this case. However, with a counter associated with it, DB2, mirrored in both cores, now requires an additional temporary result port on both cores. As a result, in the example case below, the total number of searches possible through a Master Key becomes 5. In the worst case with all databases having counter, this restriction limits the total number of parallel searches to 4. Refer to Table 95 to determine the number of searches to be embedded in a master key.

Figure 40: One Database with Counter in Each Core Shared with Second Core



One Database with counter in each core shared with 2nd core
3rd Database with counter in each core (Mirrored) – Total 5 Databases
Example: DB0=FWD, DB1=RPF, DB2=ACL1, DB3=ACL2, DB4=ACL3

Table 95: Number of Searches Imbedded in a Master Key

No. of Databases with Counters	No. of Databases without Counters	No. of Parallel Searches
0	7 (1 shared database)	7
1 (shared database)	6	7
2	4	6
3	2	5
4	0	4

20.5 Result Port Allocation Restriction when a Shared Database is Present

The restriction mentioned in Effect of Search 'N Count on Number of Possible Searches When Any Database is Shared Between the Threads can be extended to another restriction when it comes to result port allocation. Irrespective of the presence of counters, result output for the shared databases must be allocated to lower-numbered result ports as depicted in Figure 37 through Figure 40. All mirrored databases with counters require their result ports to be mapped right after the shared databases (see Figure 40). In other words, mirrored databases without counters can take up the higher-numbered available result ports only after the allocation of result ports for the shared databases and mirrored databases with counters is done.

20.6 STATS Interface Restriction

There are several restrictions applicable to the Statistics Interface when it comes to the packet size that must be transferred over the interface.

- Max Packet Size cannot exceed 448 bytes.
- Use a Min Packet Size of 256 bytes. If the required packet size is smaller than 256, use NULL to fill up the packet to MinPktSize.
- Device buffer can accumulate up to a total of four packets for further processing within the device. Hence, oversubscription over STATS interface should be avoided.

Chapter 21: PCIe Interface

21.1 Introduction

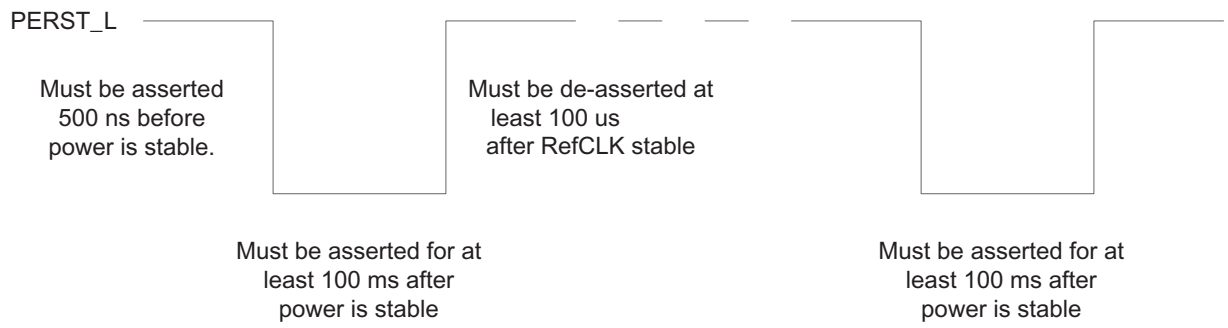
This section describes the functional description, relevant interface capabilities, and programming requirements for the PCIe controller implementation, which uses one controller to support one individual PCI Express (PCIe) lane that can be organized into x1 links.

A PCIe interface is required. Device bring-up cannot be completed without PCIe.

21.2 PCIe Features

- Compliant to PCI Express Base Specification revision 2.0
- 5 Gigabits/second/direction of raw bandwidth
- Type-0 configuration space in Endpoint (EP) mode
- Maximum payload size of 256 bytes
- Maximum request size of 256 bytes
- Polarity inversion on receive
- Supports MSI

Figure 41: PCIe Reset



21.3 Theory of Operation

The PCIe interface connects internally through the DMA controller(s).

The primary transaction type is a DMA read operation from the Host system memory initiated by the device as a DMA Master and a DMA write operation to the Host system memory. Individual writes and reads to and from all registers and memories are also supported. During device configuration, PCIe is in slave mode.

The PCIe external side interface complies with PCI Local Bus Specification Revision 2.2 for PCI-compatible operation, and PCI Express 2.0 Base specification with regard to signals and behavior.

21.3.1 PCIe Configuration Space Access

The PCIe Controller can be set up to use the original configuration space or the extended configuration space.

Upon access to this memory address space, a 40-bit address (translated from 64 bits by the MMU) is presented to the PCIe controller, which in turn is translated to the PCIe configuration address as 39:24 Base, 23:16 Bus Number, 15:11 Device Number, 10:8 Function Number, and 7:0 Register Number.

Extended register configuration space access is as below:

39:28	27:20	19:15	14:12	11:2
Base	Bus Number	Device Number	Function Number	Register Number

Providing the Bus Number and Device Number that identifies the PCIe Controller's own configuration space results in an internal configuration register access. The required read or write command effectively reads or writes the on-chip configuration registers.

21.3.2 DMA Descriptor Structures

The Request DMA Descriptors are set-up by the Host-resident software in a system memory request queue. These descriptors are up to 512B in length and 64b granular. A descriptor is comprised of a 64b header followed by payload. The header contains the same user-data fields found in the ILA Control Word as well as a few control bits and the payload length.

The device's Request DMA Controller pulls out (Master DMA over the PCIe interface) each Descriptor on a demand basis and transfers it to the targeted subsystem for execution. Similarly, responses are derived by the subsystem that executed the request instructions within the Request Descriptors and are handed over to the Response DMA Controller.

The Response DMA Controller formats the Response Descriptors of the same structure as the Request Descriptors described above. It then places each Response Descriptor into the system memory response queue by Master DMA'ing over the PCIe interface. These request and response transactions over the PCIe interface occur simultaneously for different instructions. This in effect allows the pipelining of the request-execution-response phases. The DMA Controllers have their own FIFO's in each direction in order to decouple the rate differences in the three phases.

21.4 PCIe Interrupts

The PCIe controller reports interrupts through the "Virtual Wire" concept of the PCIe, that is in-band messages carry Interrupt information.

21.4.1 Interrupt Setup

The PCIe Controller detects multiple sources of interrupts and reports them to the Programmable Interrupt Controller (PIC).

The MSI interrupt supports one entry.

21.4.2 Configuring PCIe

The following must be configured and setup for startup and initialization:

- PCI Configuration Header registers as needed.
- PCIe Interface Capability Registers as needed.
- PCIe Interface's Extended Capability Registers as needed.

Chapter 22: Device Initialization and Power-Up/Down Sequence

22.1 Device Initialization

Refer to the *BCM16K Board Design Guidelines* (References).

22.2 Power-Up Sequence

Refer to the *BCM16K Board Design Guidelines* (References).

22.3 Power-Down Sequence

Refer to the *BCM16K Board Design Guidelines* (References).

Chapter 23: LMax

23.1 Overview

The device issues Xoff (transmit off) to the source device as an indication that the input FIFOs have reached a threshold. LMax is defined as the maximum response time, from the time the input FIFO threshold has been detected until the source of data (NPU/ASIC) stops sending data to the response to that Xoff is observed on the data path. The Xoff LMax can also be defined as how much additional data can be received after input FIFO threshold has been reached without data loss.

The device input FIFO is used to buffer packets received. When the packets in the FIFO reach a threshold value (Xoff threshold), the device sends an Xoff in all outgoing Interlaken-LA Control Words of that port. Upon receiving the Xoff, the ASIC/NPU stops packet transmission to this device. After the packets stored in the FIFO fall below a threshold (Xon threshold), the device sends an Xon in all outgoing Interlaken-LA Control Words, and upon receiving this information, the ASIC or NPU device resumes packet transmission of that port.

23.2 Input FIFO Architecture

Incoming instruction packets generate information that is stored in the input FIFO. The device issues Xoff when the FIFO reaches its Xoff threshold, and reissues Xon when the Xon threshold is reached.

Xoff is declared when the FIFO threshold ≥ the programmed value in MAC_REGS_RXMAC_THRESHOLD_PT*_HI.

Xon is declared when the FIFO threshold ≤ the programmed value in MAC_REGS_RXMAC_THRESHOLD_PT*_LO.

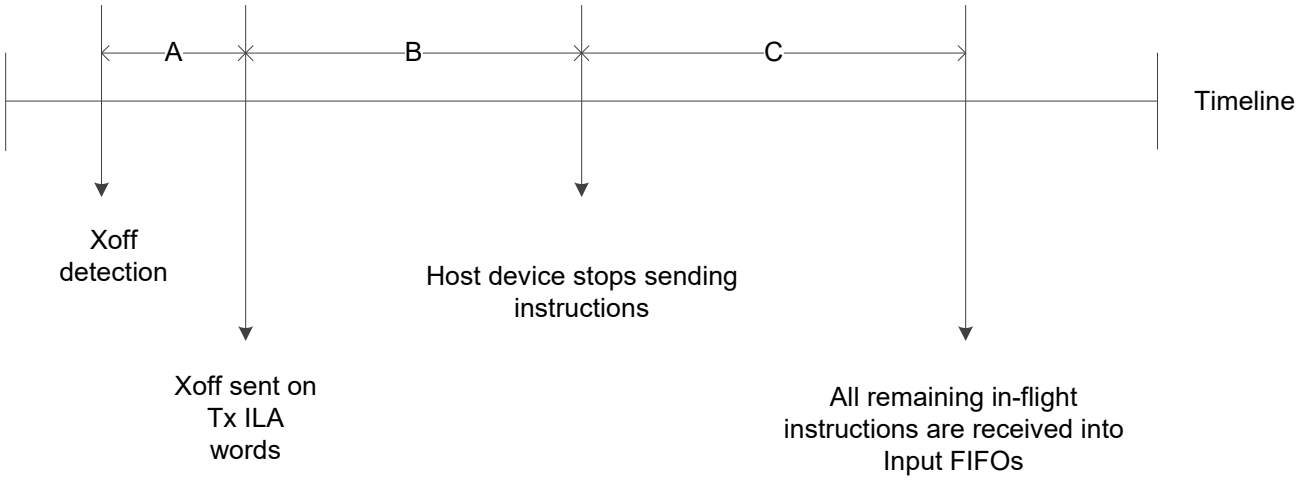
The input FIFO total depth is 1024 entries per lane, where the entry width is 8B, that is, 64b. For each instruction packet, one input FIFO entry is used to store the relevant information from the ILA control word and additional input FIFO entries are used to store the information from the ILA data words. For example, a Compare one search with 80b search key has 1 ILA control word and two ILA data words; therefore, this instruction will consume three input FIFO entries.

Bank configuration does not affect the FIFO depth.

23.3 Calculating System LMax Requirements

Xoff threshold value must be carefully selected in order to avoid input FIFO overflow. Figure 42 shows the timeline of events that occur when Xoff is detected in the input FIFO.

Figure 42: System Latency



When Xoff is detected, there are three delays until there are no more instructions being inserted into the input FIFO.

- Delay A: This delay occurs from Xoff threshold detection until Xoff transmission on transmit ILA control words.
- Delay B: This delay occurs from the transmission of Xoff until the host depletes all in-flight instructions to the pin of the host device. This time includes the time it takes for host to detect the Xoff through the ILA controller, to stop the transmission of instruction packets, and to deplete the in-flight instructions within the host device's ILA output pipeline.
- Delay C: This delay occurs while depleting the remaining in-flight instructions within the input pipeline into the input FIFO.

The host controller designer must select the input FIFO threshold such that when Xoff threshold is detected, the remaining buffer in the FIFO can absorb the maximum number of instructions that will continue to be issued during the three delays described above.

The available entries indicate the buffer available for host controller delay to Xoff event (delay B). All the data in the tables has been rounded up to the nearest entry. LMAX value for 12.5G SerDes with NRZ mode is X, Y, and Z.

Table 96: LMAX Values

Mode	LMAX (Byte)
NRZ	100 KB
PAM-4	80 KB

Chapter 24: Latency

The processor uses a fully pipelined architecture for all operations. The latency of the device is defined as the time from when the device detects the request on the RX bus pins to the time the response is sent on the TX bus.

Table 97: Latency Table

SerDes Speed (Gb/s)	900 MHz		1000 MHz		Mode
	Min. (ns)	Max. (ns)	Min. (ns)	Max. (ns)	
56.25	463	473	438	448	PAM-4
53.125	476	486	451	461	PAM-4
51.5625	482	493	458	468	PAM-4
28.125	327	347	302	322	NRZ
27.34375	329	350	304	325	NRZ
12.5G	424	469	400	444	NRZ

Chapter 25: JTAG Boundary Scan

25.1 JTAG Boundary Scan Overview

The JTAG Port operates in a manner consistent with IEEE Standard 1149.1-1900, a serial boundary scan interface, but does not implement all of the functions required for 1149.1 compliance. Some functions have been modified or eliminated. The processor supports 1149.1-1900 TAP (Test Access Port) controller architecture and operates in a manner that does not conflict with IEEE Standard 1149.1 compliant devices.

25.1.1 JTAG Boundary Scan Uses 1.8V CMOS I/O Signaling

All JTAG boundary scan signals use 1.8V CMOS I/O signaling levels. TDO is a point-to-point connection without any pull-up/down.

25.1.2 Disabling JTAG

If the TAP port is not required the TCK signal must be pulled down to GND to prevent floating inputs. All other JTAG signals may be left unconnected.

25.1.3 Resetting the TAP Controller

The JTAG TAP Controller must be reset by asserting TMS for five consecutive TCK cycles or asserting TRST_L.

25.1.4 JTAG Boundary Scan Registers

25.1.4.1 Instruction Register

The Instruction Register provides the address and control signals required to access a specific data register in the scan path. The instructions to be executed by the TAP Controller are contained in the Instruction Register. Upon performing a TAP reset, the Instruction Register is preloaded with the IDCode instruction.

Table 98: JTAG Boundary Scan Instructions

Instruction Name	OpCode	Instruction Explanation
Reserved	0x000	Reserved
ExTest	0x001	Latches the I/Os into the Boundary Scan Register (BSR) and connects this register between TDI and TDO.
IDCode	0x002	Connects the Identification Register between TDI and TDO, enabling the Identification Register contents to be read out.
Sample/Preload	0x003	Latches the I/O into the Boundary Scan Register and connects this register between TDI and TDO. However, the device is left in its normal functional mode.
HighZ	0x004	This instruction is similar to a Clamp, but it leaves the processor output pins in a high-impedance state rather than drive fixed logic 1 or logic 0 values.
Clamp	0x005	This instruction is similar to a High-Z, the processor output pins drive fixed logic 1 or logic 0 values.
Bypass	0x00F	Enables the bypass mode by connecting the Bypass Register between TDI and TDO.

25.1.4.2 Boundary Scan Register

The Boundary Scan Register allows access to device I/Os for testing purposes. The TAP Controller latches the state of the I/Os into the Boundary Scan Register during the CAPTURE-DR state after the Instruction Register is loaded with the ExTest and Sample/Preload OpCodes. This data is then shifted out through the TDO pins in the SHIFT-DR state.

25.1.4.3 Identification Register

The Identification Register is a 32-bit register that contains the manufacturer, device and die revision ID codes for the device. The TAP controller shifts the contents of this register onto TDO whenever the Instruction Register is loaded with the IDCode OpCode.

Table 99: TAP Identification Register Table

Bits	Name	R/W	Initial Value	Description
31:28	Revnum	R	Version Dependent	Revision number A1: 0x0 A2: 0x2 A3: 0x4 B0: 0x0 B1: 0x2
27:12	JTAG DevID	R	Version Dependent	JTAG DevID field This value will vary depending on the product. Please see the Table 100 for a list of possible values.
11:1	MFRID	R	0xBF	Manufacturer Identification Number
0	IDReg	R	0x1	ID Register presence indicator. Indicate the presence of the TAP Identification Register.

Table 100: JTAG DevID

Capacity	JTAG ID
X = 80/1000 Mb	0x0069A17F
M = 40/512 Mb	0x0068C17F

25.1.5 JTAG Timing Diagram and AC-Timing Parameters

Figure 43 shows the timing diagram for the JTAG signals.

Figure 43: JTAG Timing Diagram

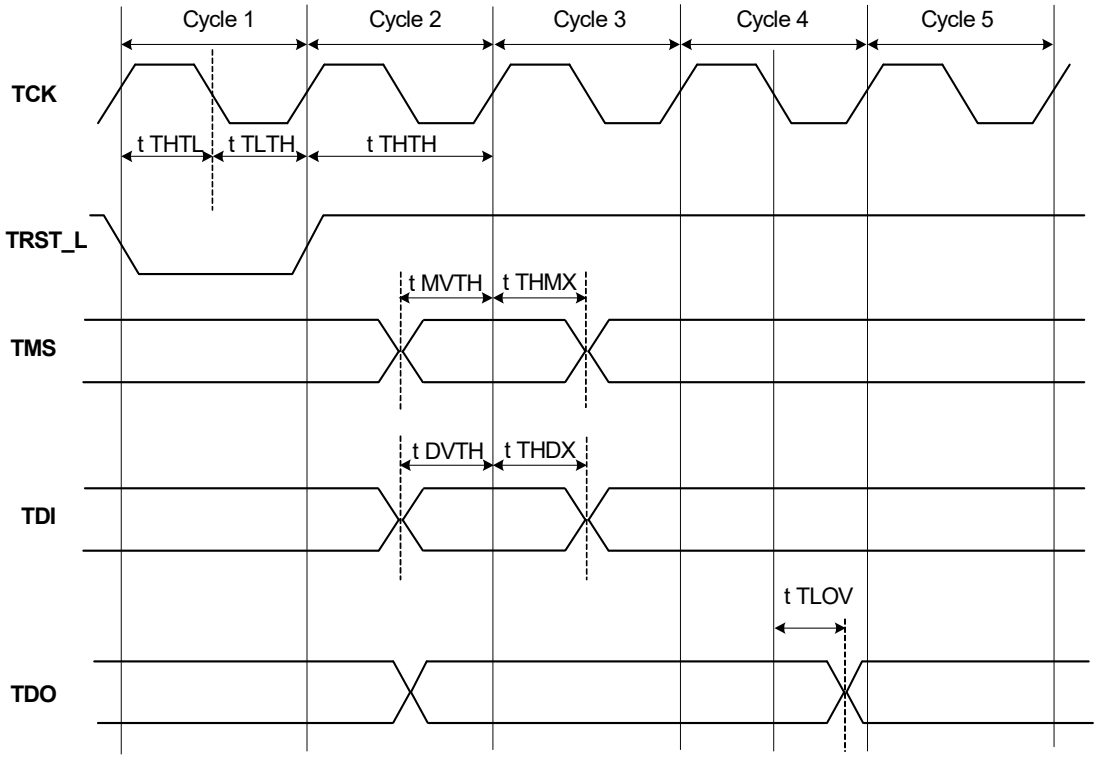


Table 101: JTAG AC-Timing Parameters

Parameter	Description	Min.	Max.	Unit
tTHTH	TCK Cycle Time	100	–	ns
tTHTL	TCK High Pulse Width	40	–	ns
tTLTH	TCK Low Pulse Width	40	–	ns
tMVTH	TMS Setup	10	–	ns
tTHMX	TMS Hold	10	–	ns
tDVTH	TDI Setup	10	–	ns
tTHDX	TDI Hold	10	–	ns
tTLOV	TCK Low to Data Valid	–	25	ns

Chapter 26: Electrical Specifications

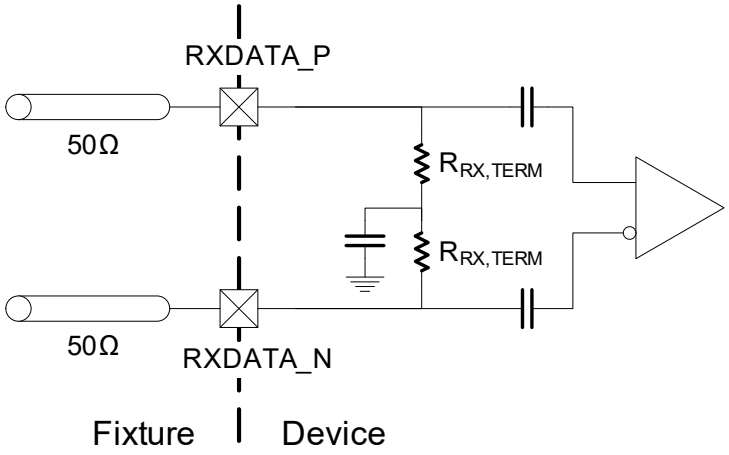
26.1 Overview

This section describes the electrical specifications.

26.1.1 Receiver

The RXDATA inputs' Equivalent circuit is shown in Figure 44. RXDATA_P and RXDATA_N are terminated differentially to one another on-chip, with a nominal differential resistance of 100Ω. AC-coupling capacitors are implemented past the termination. The data must be DC-balanced.

Figure 44: Equivalent Input Circuit for the SerDes Receiver

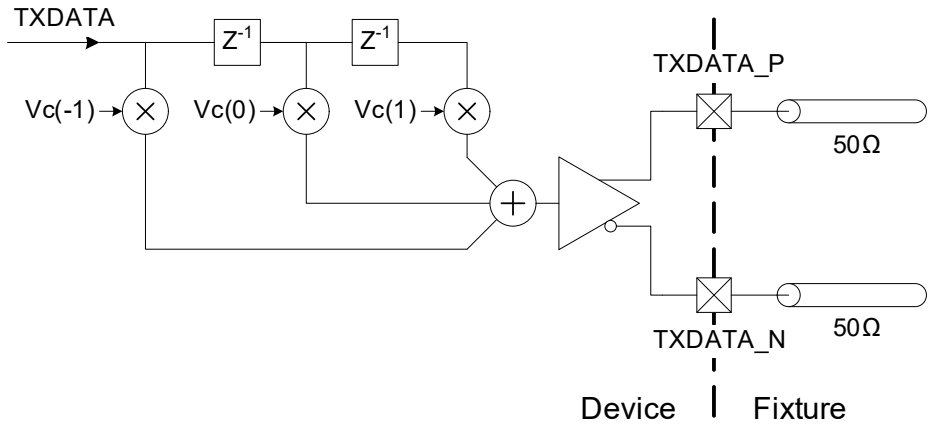


The far-end transmitter devices' total output jitter and the jitter induced by the channel cannot exceed the processor's receiver jitter tolerance, $J_{RX,TJ}$. In addition, the deterministic jitter $J_{RX,DJ}$ specification must be met.

26.1.2 Transmitter

The processor provides parallel data on the TXDATA pins. A simplified version of the output circuit and test fixture is shown in Figure 45.

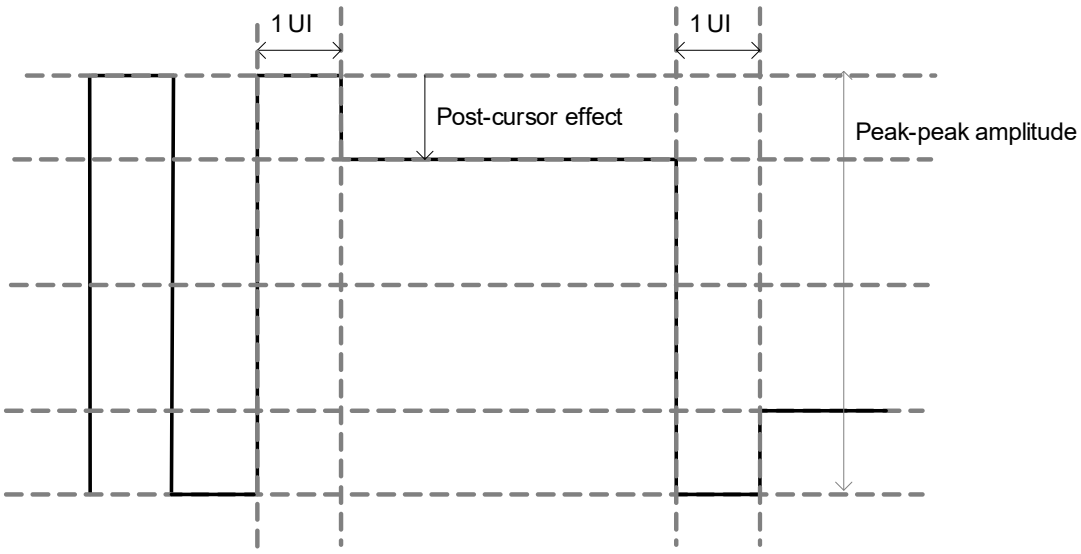
Figure 45: SerDes Equivalent Output Circuit



The drive current flowing in the output circuit has two different components: half the current is inversely proportional to the sheet resistance, and the other half has a constant value.

For Interlaken-LA applications, the processor includes a three-tap feed-forward equalizer that is useful in driving longer printed circuit traces. This equalizer preemphasizes the output signal whenever there is a data transition. The amount of preemphasis can vary between 0 and 50% and is programmed using an internal register. Figure 46 shows the effect of different register settings. Note that preemphasis does not increase the overall swing, but instead reduces the output amplitude when there is no transition. In Figure 45, Vc(–1), Vc(0), and Vc(+1) refer to pre-, main, and post-cursor, respectively.

Figure 46: Effects of Transmit Preemphasis



26.2 Absolute Maximum Ratings

Stresses and conditions greater than those rated below will cause permanent damage to the processor, resulting in failures.

Table 102: Absolute Maximum Ratings

Parameter	Range Min.	Range Max.
VDD	–0.5V	+1.05V
VDD18	–0.5V	+1.98V
VDDS12	–0.5V	+1.32V
VDDM	–0.5V	+0.935V
VDDSL	–0.5V	+1.05V
VDDS1–6	–0.5V	+0.88V
VDDS08PLL1–6	–0.5V	+0.88V
VDDS18PLL	–0.5V	+1.98V
VDDPC	–0.5V	+0.88V
VDDPCPLL	–0.5V	+0.88V
VDDCPLL	–0.5V	+1.98V
Storage Temperature	–55°C	+150°C

Table 103: Absolute Maximum Ratings—ESD Protection

Parameter	Min.	Nom.	Max.	Revision
ESD Protection (non-SerDes pins)				
Human Body Model (HBM) per EIA/JESD22-A114-E	±1000V	–	–	Ax
Charge Device Model (CDM) per EIA/JESD22-C101C-E	±250V	–	–	Ax
ESD Protection (SerDes pins)				
Human Body Model (HBM) per EIA/JESD22-A114-E	±1000V	–	–	Ax
Charge Device Model (CDM) per EIA/JESD22-C101C-E	±200V	–	–	Ax

26.3 Recommended Operating Conditions

Table 104: Recommended Operating Conditions

Parameter	Condition	Description
Operating Junction Temperature – T_J	Commercial	0°C to 105°C
Operating Junction Temperature – T_J	Extended Temperature	–40°C to +105°C

26.4 Electrical Specifications

Table 105: Performance Specifications

Parameter	Min.	Typical	Max.	Unit	Notes
Core Clock	–	850	1000	MHz	Frequency select via SDK
Baud, Symbol Rate	27.34375	–	56.25	Gbd	–
Unit Interval	35.33	–	38.79	ps	–
SerDes Reference Clock Frequency	–	156.25	–	MHz	–
Core Reference Clock Frequency	–	156.25	–	MHz	–
PCIe Reference Clock	–	100	–	MHz	–
BER	–	–	10^{-12}	S^{-1}	NRZ at 28.125 Gb/s under worst case conditions
RX Receiver					
Input Voltage: Differential	85	–	1600	mV	–
Input Voltage: Common Mode	–	750	900	mV	With respect to SerDes ground VSSS
Input Impedance	80	100	120	Ω	DC, differential, integrated on-chip
Jitter Tolerance: Total	–	–	0.65	UI	Peak-peak
Jitter Tolerance: Deterministic	–	–	0.37	UI	Peak-peak
TX Transmitter					
Output Voltage: Differential	0	1000	1050	mV	Given VDDS12 = 1.2V
Output Voltage: Common Mode	–	450	–	mV	When terminated with 100Ω differential impedance
Output Impedance	–	100	–	Ω	DC, differential, integrated on-chip
Output Differential Skew	–	–	2	ps	50% rising/falling vs. 50% falling/rising
Output Jitter: Random	–	8	10	mUI	Wideband, RMS value
Output Jitter: Deterministic	–	0.05	–	UI	Peak-peak
Output Jitter: total	–	0.15	0.28	UI	Peak-peak

Table 106: SerDes Data Rates

Refclk (MHz)	SerDes Rate (Gb/s)
156.25	56.25
156.25	53.125
156.25	51.5625
156.25	28.125
156.25	27.34375
156.25	25.78125
156.25	12.5

26.5 DC Electrical Specifications

Table 107: DC Electrical Specifications

Parameter	Symbol	Test Conditions	Min.	Nominal	Max.	Unit
Core Supply Voltage[a]	VDD	–	X[b]	0.850	X[b]	V
I/O Supply Voltage	VDD18	–	1.710	1.800	1.890	V
SerDes Termination	VDDS12	–	1.165	1.200	1.235	V
SRAM Supply Voltage	VDDM	–	0.825	0.850	0.875	V
Digital SerDes	VDDSL	–	0.870	0.900	0.930	V
Analog SerDes	VDDS1-6	–	0.775	0.800	0.825	V
SerDes PLL	VDDS08PLL1-6	–	0.775	0.800	0.825	V
SerDes 1.8V PLL	VDDS18PLL	–	1.710	1.800	1.890	V
Analog PCIe	VDDPC	–	0.775	0.800	0.825	V
PCIe PLL	VDDPCPLL	–	0.775	0.800	0.825	V
Core PLL Supply	VDDCPLL	–	1.745	1.800	1.855	V
Input HIGH Voltage[c]	VIH(DC)	Logic HIGH for slow-speed input signals	1.145	1.800	1.890	V
Input LOW Voltage[c]	VIL(DC)	Logic LOW for slow-speed input signals	–0.090	0	+0.745	V

a. The VDD spec must be met at the balls of the device. The regulator used to power the VDD rail must have the sense pins connected to the VDD and VSS balls of the device.

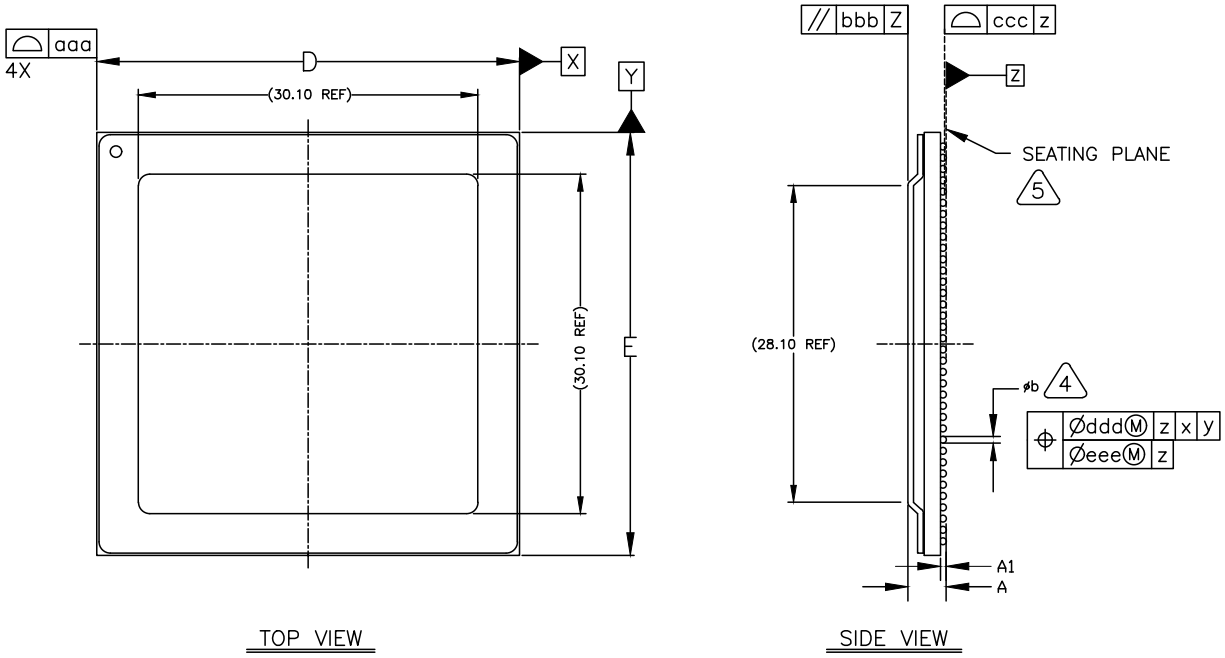
b. Managed by the AVS system. AVS connection and operation are required for proper device operation. For details of the regulator design, see the Board Design Guidelines.

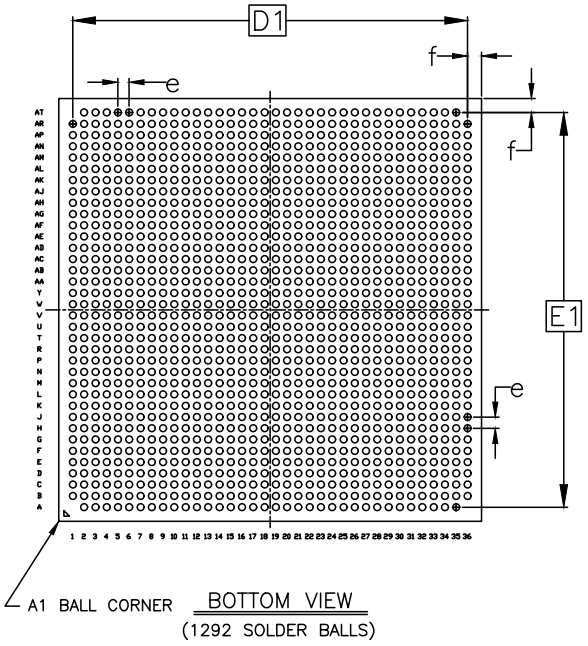
c. VIH and VIL spec also applies to MDIO and JTAG signals.

Chapter 27: Mechanical Specifications

27.1 Package Specifications

Figure 47: Package Drawing





DIMENSIONAL REFERENCES (mm)			
REF.	MIN	NOM	MAX
A	3.201	3.388	3.575
A1	0.4	0.5	0.6
D	37.40	37.50	37.60
D1		35.00 BSC	
E	37.40	37.50	37.60
E1		35.00 BSC	
b	0.50	0.60	0.70
e		1.0 BSC	
f	–	1.25	–
aaa	–	–	0.10
bbb	–	–	0.25
ccc	–	–	0.20
ddd	–	–	0.25
eee	–	–	0.10

Filename: BCM52320A0KFSBG_MOD_001

6. PCB LAND PATTERN RECOMMENDATION: LAND PATTERN KEY E REFER TO BROADCOM PACKAGING APPLICATION NOTE "PRINTED CIRCUIT BOARD LAND PATTERN RECOMMENDATIONS FOR BALL GRID ARRAY PACKAGES."

5 PRIMARY DATUM Z AND SEATING PLANE ARE DEFINED BY THE SPHERICAL CROWNS OF THE SOLDER BALLS.

4 DIMENSION IS MEASURED AT THE MAXIMUM SOLDER BALL DIAMETER, PARALLEL TO PRIMARY DATUM Z.

3. THE BASIC SOLDER BALL GRID PITCH IS 1.0mm

2. THIS PACKAGE CONFORMS TO THE JEDEC REGISTERED OUTLINE MS-034B.

1. ALL DIMENSIONS AND TOLERANCES CONFORM TO ASME Y14.5M-1994.

NOTES: UNLESS OTHERWISE SPECIFIED

27.1.1 Package Notes

The following notes are in reference to Figure 47:

- Controlling dimension: millimeter.
- Primary datum Z and seating plane are defined by the spherical crowns of the solder balls.
- Dimension 4 is measured at the maximum solder ball diameter, parallel to primary datum Z.
- The fiducial marker (circle in upper-left corner of the Top View) is for reference only.
- The maximum allowed force during heat sink attachment at room temperature is 45 lb/20.4 kgf for a maximum of 30 seconds.
- The maximum allowed sustained force during device lifespan is 27.8 lb/12.6 kgf.
- With heat sink attached, force should be uniformly applied on device surface. Concentrated force on device surface can cause die or package damage.
- An external heat sink is not allowed to tilt when in contact with the device top surface.

27.2 Package Thermal Characteristics

Customer system designs, materials and processes used for the assembly of our products (for example, selection of thermal interface materials, methods of applying heat sinks, and so on), power dissipation levels of adjacent components on the customer boards, and various environmental and customer board manufacture variables can greatly affect the selection of thermal solution required. Since these factors are unlikely within the customer's knowledge and control, it is essential that the customers evaluate our products and their associated solutions for example, recommended heat sinks) to determine whether they are fit for a particular purpose and suitable for the customer's method of application while maintaining our product's junction temperature below our maximum allowed specification.

Please contact a Broadcom FAE and application engineering for further information such as thermal model.

Chapter 28: Ball Assignment

28.1 Ball Assignment by Location

Ball Name	Signal Name
A1	DNP
A2	VSS
A3	TXDATA_N[27]
A4	TXDATA_P[27]
A5	VSS
A6	VSS
A7	TXDATA_P[24]
A8	TXDATA_N[24]
A9	VSS
A10	TXDATA_P[29]
A11	TXDATA_N[29]
A12	VSS
A13	VSS
A14	TXDATA_P[31]
A15	TXDATA_N[31]
A16	VSS
A17	PCRXDATA_N
A18	PCRXDATA_P
A19	VSS
A20	PCREFCLK_N
A21	VSS
A22	TXDATA_N[15]
A23	TXDATA_P[15]
A24	VSS
A25	VSS
A26	TXDATA_N[13]
A27	TXDATA_P[13]
A28	VSS
A29	TXDATA_N[8]
A30	TXDATA_P[8]
A31	VSS
A32	VSS
A33	TXDATA_P[11]
A34	TXDATA_N[11]
A35	VSS
A36	DNP
B1	VDDS5
B2	VDDS12
B3	VSS
B4	VSS
B5	TXDATA_P[26]
B6	TXDATA_N[26]
B7	VSS
B8	VSS
B9	VSS
B10	VSS
B11	VSS
B12	TXDATA_N[30]
B13	TXDATA_P[30]
B14	VSS
B15	VSS
B16	VSS
B17	VDDPC
B18	VDDPC
B19	VSS
B20	PCREFCLK_P
B21	VSS
B22	VSS
B23	VSS
B24	TXDATA_P[14]
B25	TXDATA_N[14]
B26	VSS
B27	VSS
B28	VSS
B29	VSS
B30	VSS
B31	TXDATA_N[10]
B32	TXDATA_P[10]
B33	VSS
B34	VSS
B35	VDDS12
B36	VDDS2
C1	VDDS5
C2	VSS
C3	VSS
C4	VSS
C5	VSS
C6	VSS
C7	TXDATA_N[25]
C8	TXDATA_P[25]
C9	VSS
C10	TXDATA_N[28]
C11	TXDATA_P[28]
C12	VSS
C13	VSS
C14	VSS
C15	PCTXDATA_N
C16	PCTXDATA_P
C17	VSS
C18	VDDPCPLL
C19	DNC
C20	VSS
C21	DNC
C22	VSS
C23	VDD18
C24	VSS
C25	VSS
C26	TXDATA_P[12]
C27	TXDATA_N[12]
C28	VSS
C29	TXDATA_P[9]
C30	TXDATA_N[9]
C31	VSS
C32	VSS
C33	VDD18
C34	VSS
C35	VSS
C36	VDDS2
D1	VSS
D2	TXDATA_N[23]
D3	VSS
D4	RXDATA_P[22]
D5	RXDATA_N[22]
D6	VSS
D7	VSS
D8	VSS
D9	VSS
D10	VSS
D11	VSS
D12	VSS
D13	VSS
D14	VSS
D15	VSS
D16	VSS
D17	VSS
D18	VSS
D19	VSS
D20	VSS
D21	VSS
D22	VSS
D23	VSS
D24	VSS
D25	VSS
D26	VSS
D27	VSS
D28	VSS
D29	VSS
D30	VSS
D31	VSS
D32	VSS
D33	VSS
D34	VSS
D35	TXDATA_N[7]
D36	VSS
E1	VSS
E2	TXDATA_P[23]
E3	VSS
E4	VDDS5
E5	VDDS5
E6	VSS
E7	RXDATA_N[27]
E8	VDDS4
E9	RXDATA_P[26]
E10	VSS
E11	VDDS4
E12	VDDS12

Ball Name	Signal Name
E13	VDDS4
E14	VSS
E15	RXDATA_P[28]
E16	VDDS4
E17	RXDATA_P[30]
E18	VSS
E19	RXDATA_P[14]
E20	VDDS3
E21	RXDATA_P[12]
E22	VSS
E23	RXDATA_P[9]
E24	VDDS3
E25	RXDATA_P[11]
E26	VSS
E27	VDDS3
E28	VDDS12
E29	VDDS3
E30	VSS
E31	RXDATA_N[6]
E32	VDDS2
E33	RXDATA_N[4]
E34	VSS
E35	TXDATA_P[7]
E36	VSS
F1	TXDATA_P[22]
F2	VSS
F3	VSS
F4	RXDATA_N[21]
F5	RXDATA_P[21]
F6	VSS
F7	RXDATA_P[27]
F8	VDDS4
F9	RXDATA_N[26]
F10	VSS
F11	VDDS4
F12	VDDS12
F13	VDDS4
F14	VSS
F15	RXDATA_N[28]
F16	VDDS4
F17	RXDATA_N[30]
F18	VSS
F19	RXDATA_N[14]
F20	VDDS3

Ball Name	Signal Name
F21	RXDATA_N[12]
F22	VSS
F23	RXDATA_N[9]
F24	VDDS3
F25	RXDATA_N[11]
F26	VSS
F27	VDDS3
F28	VDDS12
F29	VDDS3
F30	VSS
F31	RXDATA_P[6]
F32	VDDS2
F33	RXDATA_P[4]
F34	VSS
F35	VSS
F36	TXDATA_P[6]
G1	TXDATA_N[22]
G2	VSS
G3	VSS
G4	VSS
G5	VSS
G6	VSS
G7	VSS
G8	VSS
G9	VSS
G10	VSS
G11	VSS
G12	VSS
G13	VSS
G14	VSS
G15	VSS
G16	VSS
G17	VSS
G18	VSS
G19	VSS
G20	VSS
G21	VSS
G22	VSS
G23	VSS
G24	VSS
G25	VSS
G26	VSS
G27	VSS
G28	VSS

Ball Name	Signal Name
G29	VSS
G30	VSS
G31	VSS
G32	VSS
G33	VSS
G34	VSS
G35	VSS
G36	TXDATA_N[6]
H1	VSS
H2	TXDATA_N[21]
H3	VSS
H4	RXDATA_N[16]
H5	RXDATA_P[16]
H6	VSS
H7	RXDATA_N[20]
H8	VDDS5
H9	RXDATA_P[23]
H10	VSS
H11	RXDATA_N[25]
H12	VDDS4
H13	RXDATA_N[24]
H14	VSS
H15	RXDATA_N[29]
H16	VDDS4
H17	RXDATA_N[31]
H18	VSS
H19	RXDATA_N[15]
H20	VDDS3
H21	RXDATA_N[13]
H22	VSS
H23	RXDATA_N[8]
H24	VDDS3
H25	RXDATA_N[10]
H26	VSS
H27	RXDATA_N[7]
H28	VDDS2
H29	RXDATA_P[5]
H30	VSS
H31	VSS
H32	RXDATA_P[0]
H33	RXDATA_N[0]
H34	VSS
H35	TXDATA_N[5]
H36	VSS

Ball Name	Signal Name
J1	VSS
J2	TXDATA_P[21]
J3	VSS
J4	VDDS5
J5	VDDS5
J6	VSS
J7	RXDATA_P[20]
J8	VDDS5
J9	RXDATA_N[23]
J10	VSS
J11	RXDATA_P[25]
J12	VDDS4
J13	RXDATA_P[24]
J14	VSS
J15	RXDATA_P[29]
J16	VDDS4
J17	RXDATA_P[31]
J18	VSS
J19	RXDATA_P[15]
J20	VDDS3
J21	RXDATA_P[13]
J22	VSS
J23	RXDATA_P[8]
J24	VDDS3
J25	RXDATA_P[10]
J26	VSS
J27	RXDATA_P[7]
J28	VDDS2
J29	RXDATA_N[5]
J30	VSS
J31	VSS
J32	VDDS2
J33	VDDS2
J34	VSS
J35	TXDATA_P[5]
J36	VSS
K1	TXDATA_P[20]
K2	VSS
K3	VSS
K4	RXDATA_N[17]
K5	RXDATA_P[17]
K6	VSS
K7	VSS
K8	VSS

Ball Name	Signal Name
K9	VSS
K10	VSS
K11	VSS
K12	VSS
K13	VSS
K14	VSS
K15	VSS
K16	VSS
K17	VSS
K18	VSS
K19	VSS
K20	VSS
K21	VSS
K22	VSS
K23	VSS
K24	VSS
K25	VSS
K26	VSS
K27	VSS
K28	VSS
K29	VSS
K30	VSS
K31	VSS
K32	RXDATA_P[1]
K33	RXDATA_N[1]
K34	VSS
K35	VSS
K36	TXDATA_P[4]
L1	TXDATA_N[20]
L2	VSS
L3	VSS
L4	VSS
L5	VSS
L6	VSS
L7	VDDS08PLL6
L8	VDDS18PLL
L9	VDDS08PLL5
L10	VDDS18PLL
L11	VDDS08PLL4
L12	VDDS18PLL
L13	VSS
L14	SREFCLK1_P
L15	SREFCLK1_N
L16	VSS

Ball Name	Signal Name
L17	CREFCLK_N
L18	CREFCLK_P
L19	VSSCPLL
L20	VDDCPLL
L21	VSS
L22	SREFCLK0_N
L23	SREFCLK0_P
L24	VSS
L25	VDDS18PLL
L26	VDDS08PLL3
L27	VDDS18PLL
L28	VDDS08PLL2
L29	VDDS18PLL
L30	VDDS08PLL1
L31	VSS
L32	VSS
L33	VSS
L34	VSS
L35	VSS
L36	TXDATA_N[4]
M1	VSS
M2	VSS
M3	VSS
M4	RXDATA_P[18]
M5	RXDATA_N[18]
M6	VSS
M7	VSS
M8	VSS
M9	VSS
M10	VSS
M11	VSS
M12	VSS
M13	DNC
M14	DNC
M15	DNC
M16	DNC
M17	DNC
M18	DNC
M19	DNC
M20	DNC
M21	DNC
M22	DNC
M23	DNC
M24	DNC

Ball Name	Signal Name
M25	VSS
M26	VSS
M27	VSS
M28	VSS
M29	VSS
M30	VSS
M31	VSS
M32	RXDATA_N[2]
M33	RXDATA_P[2]
M34	VSS
M35	VSS
M36	VSS
N1	VDDS5
N2	VDDS12
N3	VSS
N4	VDDS5
N5	VDDS5
N6	VSS
N7	VSS
N8	VDDSL
N9	VSS
N10	VDDSL
N11	VSS
N12	VDDSL
N13	VSS
N14	VDDSL
N15	VSS
N16	VDD
N17	VSS
N18	VDD
N19	VSS
N20	VDD
N21	VSS
N22	VDD
N23	VSS
N24	VDDSL
N25	VSS
N26	VDDSL
N27	VSS
N28	VDDSL
N29	VSS
N30	VDDSL
N31	VSS
N32	VDDS2

Ball Name	Signal Name
N33	VDDS2
N34	VSS
N35	VDDS12
N36	VDDS2
P1	VDDS5
P2	VSS
P3	VSS
P4	RXDATA_N[19]
P5	RXDATA_P[19]
P6	VSS
P7	VDD
P8	VSS
P9	VDD
P10	VSS
P11	VDD
P12	VSS
P13	VDD
P14	VSS
P15	VDD
P16	VSS
P17	VDD
P18	VSS
P19	VDD
P20	VSS
P21	VDD
P22	VSS
P23	VDD
P24	VSS
P25	VDD
P26	VSS
P27	VDD
P28	VSS
P29	VDD
P30	VSS
P31	VSS
P32	RXDATA_P[3]
P33	RXDATA_N[3]
P34	VSS
P35	VSS
P36	VDDS2
R1	VSS
R2	TXDATA_P[17]
R3	VSS
R4	VSS

Ball Name	Signal Name
R5	VSS
R6	VSS
R7	VSS
R8	VDD
R9	VSS
R10	VDD
R11	VSS
R12	VDD
R13	VSS
R14	VDD
R15	VSS
R16	VDD
R17	VSS
R18	VDD
R19	VSS
R20	VDD
R21	VSS
R22	VDD
R23	VSS
R24	VDD
R25	VSS
R26	VDD
R27	VSS
R28	VDD
R29	VSS
R30	VDD
R31	VSS
R32	VSS
R33	VSS
R34	VSS
R35	TXDATA_P[1]
R36	VSS
T1	VSS
T2	TXDATA_N[17]
T3	VSS
T4	RXDATA_N[47]
T5	RXDATA_P[47]
T6	VSS
T7	VDD
T8	VSS
T9	VDD
T10	VSS
T11	VDD
T12	VSS

Ball Name	Signal Name
T13	VDD
T14	VSS
T15	VDD
T16	VSS
T17	VDD
T18	VSS
T19	VDD
T20	VSS
T21	VDD
T22	VSS
T23	VDD
T24	VSS
T25	VDD
T26	VSS
T27	VDD
T28	VSS
T29	VDD
T30	VSS
T31	VSS
T32	RXDATA_P[37]
T33	RXDATA_N[37]
T34	VSS
T35	TXDATA_N[1]
T36	VSS
U1	TXDATA_N[16]
U2	VSS
U3	VSS
U4	VDDS6
U5	VDDS6
U6	VSS
U7	VSS
U8	VDD
U9	VSS
U10	VDD
U11	VSS
U12	VDD
U13	VSS
U14	VDD
U15	VSS
U16	VDD
U17	VSS
U18	VDD
U19	VSS
U20	VDD

Ball Name	Signal Name
U21	VSS
U22	VDD
U23	VSS
U24	VDD
U25	VSS
U26	VDD
U27	VSS
U28	VDD
U29	VSS
U30	VDD
U31	VSS
U32	VDDS1
U33	VDDS1
U34	VSS
U35	VSS
U36	TXDATA_N[0]
V1	TXDATA_P[16]
V2	VSS
V3	VSS
V4	RXDATA_P[46]
V5	RXDATA_N[46]
V6	VSS
V7	VDD
V8	VSS
V9	VDD
V10	VSS
V11	VDD
V12	VSS
V13	VDD
V14	VSS
V15	VDD
V16	VSS
V17	VDD
V18	VSS
V19	VDD
V20	VSS
V21	VDD
V22	VSS
V23	VDD
V24	VSS
V25	VDD
V26	VSS
V27	VDD
V28	VSS

Ball Name	Signal Name
V29	VDD
V30	VSS
V31	VSS
V32	RXDATA_N[38]
V33	RXDATA_P[38]
V34	VSS
V35	VSS
V36	TXDATA_P[0]
W1	VSS
W2	TXDATA_N[18]
W3	VSS
W4	VSS
W5	VSS
W6	VSS
W7	VSS
W8	VDD
W9	VSS
W10	VDDSL
W11	VSS
W12	VDD
W13	VSS
W14	VDD
W15	VSS
W16	VDD
W17	VSS
W18	VDD
W19	VSS
W20	VDD
W21	VSS
W22	VDD
W23	VSS
W24	VDD
W25	VSS
W26	VDDSL
W27	VSS
W28	VDD
W29	VSS
W30	VDD
W31	VSS
W32	VSS
W33	VSS
W34	VSS
W35	TXDATA_N[2]
W36	VSS

Ball Name	Signal Name
Y1	VSS
Y2	TXDATA_P[18]
Y3	VSS
Y4	RXDATA_P[40]
Y5	RXDATA_N[40]
Y6	VSS
Y7	VDD
Y8	VSS
Y9	VDD
Y10	VSS
Y11	VDDSL
Y12	VSS
Y13	VDD
Y14	VSS
Y15	VDD
Y16	VSS
Y17	VDD
Y18	VSS
Y19	VDD
Y20	VSS
Y21	VDD
Y22	VSS
Y23	VDD
Y24	VSS
Y25	VDD
Y26	VSS
Y27	VDDSL
Y28	VSS
Y29	VDD
Y30	VSS
Y31	VSS
Y32	RXDATA_N[39]
Y33	RXDATA_P[39]
Y34	VSS
Y35	TXDATA_P[2]
Y36	VSS
AA1	TXDATA_P[19]
AA2	VSS
AA3	VSS
AA4	VDDS6
AA5	VDDS6
AA6	VSS
AA7	VSS
AA8	VDD

Ball Name	Signal Name
AA9	VSS
AA10	VDD
AA11	VSS
AA12	VDD
AA13	VSS
AA14	VDD
AA15	VSS
AA16	VDD
AA17	VSS
AA18	VDD
AA19	VSS
AA20	VDD
AA21	VSS
AA22	VDD
AA23	VSS
AA24	VDD
AA25	VSS
AA26	VDD
AA27	VSS
AA28	VDD
AA29	VSS
AA30	VDD
AA31	VSS
AA32	VDDS1
AA33	VDDS1
AA34	VSS
AA35	VSS
AA36	TXDATA_P[3]
AB1	TXDATA_N[19]
AB2	VSS
AB3	VSS
AB4	RXDATA_N[41]
AB5	RXDATA_P[41]
AB6	VSS
AB7	VDD
AB8	VSS
AB9	VDD
AB10	VSS
AB11	VDD
AB12	VSS
AB13	VDD
AB14	VSS
AB15	VDD
AB16	VSS

Ball Name	Signal Name
AB17	VDD
AB18	VSS
AB19	VDD
AB20	VSS
AB21	VDD
AB22	VSS
AB23	VDD
AB24	VSS
AB25	VDD
AB26	VSS
AB27	VDD
AB28	VSS
AB29	VDD
AB30	VSS
AB31	VSS
AB32	RXDATA_P[33]
AB33	RXDATA_N[33]
AB34	VSS
AB35	VSS
AB36	TXDATA_N[3]
AC1	VSS
AC2	VSS
AC3	VSS
AC4	VSS
AC5	VSS
AC6	VSS
AC7	VSS
AC8	VDD
AC9	VSS
AC10	VDD
AC11	VSS
AC12	VDD
AC13	VSS
AC14	VDD
AC15	VSS
AC16	VDD
AC17	VSS
AC18	VDD
AC19	VSS
AC20	VDD
AC21	VSS
AC22	VDD
AC23	VSS
AC24	VDD

Ball Name	Signal Name
AC25	VSS
AC26	VDD
AC27	VSS
AC28	VDD
AC29	VSS
AC30	VDD
AC31	VSS
AC32	VSS
AC33	VSS
AC34	VSS
AC35	VSS
AC36	VSS
AD1	VDDS6
AD2	VDDS12
AD3	VSS
AD4	RXDATA_N[44]
AD5	RXDATA_P[44]
AD6	VSS
AD7	VDD
AD8	VSS
AD9	VDD
AD10	VSS
AD11	VDD
AD12	VSS
AD13	VDD
AD14	VSS
AD15	VDD
AD16	VSS
AD17	VDD
AD18	VSS
AD19	VDD
AD20	VSS
AD21	VDD
AD22	VSS
AD23	VDD
AD24	VSS
AD25	VDD
AD26	VSS
AD27	VDD
AD28	VSS
AD29	VDD
AD30	VSS
AD31	VSS
AD32	RXDATA_N[32]

Ball Name	Signal Name
AD33	RXDATA_P[32]
AD34	VSS
AD35	VDDS12
AD36	VDDS1
AE1	VDDS6
AE2	VSS
AE3	VSS
AE4	VDDS6
AE5	VDDS6
AE6	VSS
AE7	VSS
AE8	VDD
AE9	VSS
AE10	VDD
AE11	VSS
AE12	VDD
AE13	VSS
AE14	VDD
AE15	VSS
AE16	VDD
AE17	VSS
AE18	VDD
AE19	VSS
AE20	VDD
AE21	VSS
AE22	VDD
AE23	VSS
AE24	VDD
AE25	VSS
AE26	VDD
AE27	VSS
AE28	VDD
AE29	VSS
AE30	VDD
AE31	VSS
AE32	VDDS1
AE33	VDDS1
AE34	VSS
AE35	VSS
AE36	VDDS1
AF1	VSS
AF2	TXDATA_N[47]
AF3	VSS
AF4	RXDATA_P[45]

Ball Name	Signal Name
AF5	RXDATA_N[45]
AF6	VSS
AF7	VDD
AF8	VSS
AF9	VDD
AF10	VSS
AF11	VDD
AF12	VSS
AF13	VDD
AF14	VSS
AF15	VDD
AF16	VSS
AF17	VDD
AF18	VSS
AF19	VDD
AF20	VSS
AF21	VDD
AF22	VSS
AF23	VDD
AF24	VSS
AF25	VDD
AF26	VSS
AF27	VDD
AF28	VSS
AF29	VDD
AF30	VSS
AF31	VSS
AF32	RXDATA_P[36]
AF33	RXDATA_N[36]
AF34	VSS
AF35	TXDATA_N[39]
AF36	VSS
AG1	VSS
AG2	TXDATA_P[47]
AG3	VSS
AG4	VSS
AG5	VSS
AG6	VSS
AG7	VSS
AG8	VDD
AG9	VSS
AG10	VDD
AG11	VSS
AG12	VDD

Ball Name	Signal Name
AG13	VSS
AG14	VDD
AG15	VSS
AG16	VDD
AG17	VSS
AG18	VDD
AG19	VSS
AG20	VDD
AG21	VSS
AG22	VDD
AG23	VSS
AG24	VDD
AG25	VSS
AG26	VDD
AG27	VSS
AG28	VDD
AG29	VSS
AG30	VDD
AG31	VSS
AG32	VSS
AG33	VSS
AG34	VSS
AG35	TXDATA_P[39]
AG36	VSS
AH1	TXDATA_P[46]
AH2	VSS
AH3	VSS
AH4	RXDATA_N[43]
AH5	RXDATA_P[43]
AH6	VSS
AH7	VSENSEP
AH8	VSS
AH9	VDD
AH10	VSS
AH11	VDD
AH12	VSS
AH13	VDD
AH14	VSS
AH15	VDD
AH16	VSS
AH17	VDD
AH18	VSS
AH19	VDD
AH20	VSS

Ball Name	Signal Name
AH21	VDD
AH22	VSS
AH23	VDD
AH24	VSS
AH25	VDD
AH26	VSS
AH27	VDD
AH28	VSS
AH29	VDD
AH30	VSS
AH31	VSS
AH32	RXDATA_P[35]
AH33	RXDATA_N[35]
AH34	VSS
AH35	VSS
AH36	TXDATA_P[38]
AJ1	TXDATA_N[46]
AJ2	VSS
AJ3	VSS
AJ4	VDDS6
AJ5	VDDS6
AJ6	VSS
AJ7	VSENSEN
AJ8	VDD
AJ9	VSS
AJ10	VDD
AJ11	VSS
AJ12	VDD
AJ13	VSS
AJ14	VDD
AJ15	VSS
AJ16	VDD
AJ17	VSS
AJ18	VDD
AJ19	VSS
AJ20	VDD
AJ21	VSS
AJ22	VDD
AJ23	VSS
AJ24	VDD
AJ25	VSS
AJ26	VDD
AJ27	VSS
AJ28	VDD

Ball Name	Signal Name
AJ29	TMS
AJ30	TDI
AJ31	VSS
AJ32	VDDS1
AJ33	VDDS1
AJ34	VSS
AJ35	VSS
AJ36	TXDATA_N[38]
AK1	VSS
AK2	TXDATA_N[45]
AK3	VSS
AK4	RXDATA_P[42]
AK5	RXDATA_N[42]
AK6	VSS
AK7	MDC
AK8	TRST_L
AK9	MPID[1]
AK10	VDDM
AK11	VDDM
AK12	VSS
AK13	VDDM
AK14	VSS
AK15	VDDM
AK16	VSS
AK17	VDDM
AK18	VSS
AK19	VDDM
AK20	VSS
AK21	VDDM
AK22	VSS
AK23	VDDM
AK24	VSS
AK25	VDDM
AK26	VSS
AK27	VDDM
AK28	VSS
AK29	VDDM
AK30	TDO
AK31	VSS
AK32	RXDATA_N[34]
AK33	RXDATA_P[34]
AK34	VSS
AK35	TXDATA_N[37]
AK36	VSS

Ball Name	Signal Name
AL1	VSS
AL2	TXDATA_P[45]
AL3	VSS
AL4	VSS
AL5	VSS
AL6	VSS
AL7	RESCAL1
AL8	MPID[0]
AL9	MPID[2]
AL10	PERST_L
AL11	DNC
AL12	VDDM
AL13	DNC
AL14	VDDM
AL15	DNC
AL16	VDDM
AL17	DNC
AL18	VDDM
AL19	DNC
AL20	VDDM
AL21	DNC
AL22	VDDM
AL23	DNC
AL24	VDDM
AL25	DNC
AL26	VDDM
AL27	DNC
AL28	VDDM
AL29	VSS
AL30	VDD18
AL31	VSS
AL32	VSS
AL33	VSS
AL34	VSS
AL35	TXDATA_P[37]
AL36	VSS
AM1	TXDATA_P[44]
AM2	VSS
AM3	VSS
AM4	VDDS12
AM5	VSS
AM6	VSS
AM7	SCL
AM8	SRST_L

Ball Name	Signal Name
AM9	VSS
AM10	VDD18
AM11	VSS
AM12	DNC
AM13	PRTCL
AM14	DNC
AM15	VSS
AM16	DNC
AM17	VSS
AM18	DNC
AM19	VSS
AM20	DNC
AM21	VSS
AM22	DNC
AM23	VSS
AM24	DNC
AM25	VSS
AM26	DNC
AM27	VSS
AM28	DNC
AM29	VDD18
AM30	DNC
AM31	VSS
AM32	VSS
AM33	VDDS12
AM34	VSS
AM35	VSS
AM36	TXDATA_P[36]
AN1	TXDATA_N[44]
AN2	VSS
AN3	TXDATA_P[41]
AN4	VSS
AN5	TXDATA_N[40]
AN6	VSS
AN7	VDD18
AN8	GIO_L
AN9	SDA
AN10	MPID[3]
AN11	DNC
AN12	VSS
AN13	DNC
AN14	VSS

Ball Name	Signal Name
AN15	DNC
AN16	VSS
AN17	DNC
AN18	VSS
AN19	DNC
AN20	VSS
AN21	DNC
AN22	VSS
AN23	DNC
AN24	VSS
AN25	DNC
AN26	VSS
AN27	DNC
AN28	VSS
AN29	DNC
AN30	DNC
AN31	VSS
AN32	TXDATA_N[32]
AN33	VSS
AN34	TXDATA_P[33]
AN35	VSS
AN36	TXDATA_N[36]
AP1	VSS
AP2	VSS
AP3	TXDATA_N[41]
AP4	VSS
AP5	TXDATA_P[40]
AP6	VSS
AP7	VSS
AP8	CRST_L
AP9	MDIO
AP10	MSEL[1]
AP11	DNC
AP12	DNC
AP13	DNC
AP14	DNC
AP15	DNC
AP16	DNC
AP17	DNC
AP18	DNC
AP19	DNC
AP20	DNC
AP21	DNC
AP22	DNC

Ball Name	Signal Name
AP23	DNC
AP24	DNC
AP25	SREF
AP26	DNC
AP27	DNC
AP28	DNC
AP29	DNC
AP30	VSS
AP31	VSS
AP32	TXDATA_P[32]
AP33	VSS
AP34	TXDATA_N[33]
AP35	VSS
AP36	VSS
AR1	VSS
AR2	VDDS6
AR3	VSS
AR4	TXDATA_N[42]
AR5	VSS
AR6	TXDATA_P[43]
AR7	VSS
AR8	AVSD
AR9	AVSA
AR10	VDD18
AR11	VSS
AR12	DNC
AR13	VSS
AR14	DNC
AR15	VSS
AR16	DNC
AR17	VSS
AR18	DNC
AR19	VSS
AR20	DNC
AR21	VSS
AR22	DNC
AR23	VSS
AR24	DNC
AR25	VSS
AR26	DNC
AR27	VSS
AR28	DNC
AR29	VDD18
AR30	VSS

Ball Name	Signal Name
AR31	TXDATA_N[35]
AR32	VSS
AR33	TXDATA_N[34]
AR34	VSS
AR35	VDDS1
AR36	VSS
AT1	DNP
AT2	VDDS6
AT3	VSS
AT4	TXDATA_P[42]
AT5	VSS
AT6	TXDATA_N[43]
AT7	VSS
AT8	TCK
AT9	MPID[4]
AT10	MSEL[0]
AT11	DNC
AT12	VSS
AT13	DNC
AT14	VSS
AT15	DNC
AT16	VSS
AT17	DNC
AT18	VSS
AT19	DNC
AT20	VSS
AT21	DNC
AT22	VSS
AT23	DNC
AT24	VSS
AT25	DNC
AT26	VSS
AT27	DNC
AT28	VSS
AT29	RESCAL0
AT30	VSS
AT31	TXDATA_P[35]
AT32	VSS
AT33	TXDATA_P[34]
AT34	VSS
AT35	VDDS1
AT36	DNP

Chapter 29: Summary of BCM15K/BCM52311 and BCM16K Differences

29.1 Summary of Differences

This section summarizes the differences between this device and the previous generation.

Table 108: Summary of Differences

BCM15K and BCM52311	BCM16K
Max core frequency: 900 MHz.	Max. core frequency: 1000 MHz.
Context Buffer 4096 x 640b.	Context Buffer 5120 x 640b.
Four KPUs where each KPU generates one key or a key pair.	Four KPUs per core where each KPU generates one key or a key pair.
Maximum 8x parallel search per cycle.	Maximum 8x per core parallel search per cycle.
In SMT mode, LTRs, and KPUs can be assigned asymmetrically between two threads.	In SMT mode, LTRs, and KPUs can be assigned asymmetrically between two threads per core.
512 Mb User Data Array.	512 Mb per core user data array.
Maximum 1024b User Data Array accessible per cycle with no restrictions.	Maximum 1024b per core User Data Array accessible per cycle with no restrictions.
Serial Lanes: 36 RX/TX lanes up to 28.125G. Max 16 lanes for single host.	Serial lanes for search: 32 RX/TX up to 56.25 Gb/s. Serial lanes for statistics for Broadcom switches: 16 RX/TX up to 53.125 Gb/s.
PCIe interface highly recommended.	PCIe required for device bring-up.
SMT two threads supported.	SMT four threads supported.
Statistic not supported.	Statistic supported.

Chapter 30: Errata and Device Status

30.1 KBP SDK for B0

For B0-silicon support, SDK version 1.5.7 or newer is required.

30.2 Switching MDIO Access from Pins to PCIe

Issue	When MDIO access is switched from external MDIO pins to MDIO through PCIe, an error may occur. If the last access through external MDIO pins is a write, any subsequent register access through PCIe times out and is unsuccessful.
Workaround	Always close the MDIO transaction through external pins with an MDIO read. Any register can be read.
Silicon Revisions Affected	All
Resolution	There is no plan to fix this issue in silicon. Use SDK 1.5.3 or later.

30.3 A0 Silicon 56G Link-Training and Tracking is Sub-Optimal across Full Temperature Range

Issue	Link Training on some links can exhibit run-to-run variation in BER under the same initial conditions. Link BER can be unstable over up and down temperature ramps.
Workaround	For A0 silicon, use the latest KBPSDK for the most robust firmware and programming available. For full production qualification across the entire range of processes, voltage and temperature, use B0 silicon.
Silicon Revisions Affected	A0
Resolution	This issue will be fixed in B0.

30.4 PCI Configuration Space STATUS Register Access

Issue	PCIe DMA may stop working if the STATUS register of PCI Configuration space is written using a byte (8b) or word (16b) write operation.
Workaround	Always use 32b Read-modify-Write operation for COMMAND/STATUS registers.
Silicon Revisions Affected	All
Resolution	There is no plan to fix this issue in silicon. Use KBP SDK 1.5.5 or later.

Chapter 31: Ordering Information

31.1 Part Number Decoding

Table 109 shows how the ordering part number is decoded. Contact your sales representative for device availability prior to ordering.

Table 109: Ordering Part Number Constituents

Parameter	Constituent
Part Numbers	BCM16K
Revision	A0/B0
Temperature (ambient)	C = Commercial 0°C to 105°C E = Extended –40°C to +105°C
RoHS	H = RoHS 6
Package	0 = 37.5 mm
Mode	S = Heterogeneous with Statistics
Capacity	X = 80/1000 Mb M = 40/512 Mb S = 20/512 Mb
SerDes	5 = 53.125 and 56.25 Gb/s 7 = 28.125, 27.34375, and 25.78125 Gb/s
Speed Grade	1 = 1000 MHz 9 = 900 MHz
Protocol	R = ROP Record-Over-Packet and ILA Interlaken Look-Aside
Switch/Packet Processors	S = Single D = Dual
Connects to JR2C+	Blank = Does not connect to JR2C+ P = Plus = Does connect to JR2C+

Table 110: Ordering Part Numbers

Part Number	Rev	Temperature	RoHS	Package	Mode	Capacity	SerDes	Speed Grade	Protocol	Switch/PP	J2C+
BCM16KB0CH0SX79RS	B0	C	H	0	S	X	7	9	R	S	–
BCM16KB0CH0SS51RS	B0	C	H	0	S	S	5	1	R	S	–
BCM16KB0CH0SM59RSP	B0	C	H	0	S	M	5	9	R	S	P
BCM16KB0CH0SX59RSP	B0	C	H	0	S	X	5	9	R	S	P
BCM16KB0CH0SM51RS	B0	C	H	0	S	M	5	1	R	S	–
BCM16KB0CH0SX51RS	B0	C	H	0	S	X	5	1	R	S	–
BCM16KB0CH0SM51RD	B0	C	H	0	S	M	5	1	R	D	–
BCM16KB0CH0SX51RD	B0	C	H	0	S	X	5	1	R	D	–
BCM16KB0CH0SS51LS	B0	C	H	0	S	S	5	1	L	S	–
BCM16KB0CH0SM51LS	B0	C	H	0	S	M	5	1	L	S	–
BCM16KB0CH0SX51LS	B0	C	H	0	S	X	5	1	L	S	–
BCM16KB0CH0SM51LD	B0	C	H	0	S	M	5	1	L	D	–
BCM16KB0CH0SX51LD	B0	C	H	0	S	X	5	1	L	D	–

Table 111: Ordering Part Numbers

Part Number	Rev	Temperature	RoHS	Package	Mode	Capacity	SerDes	Speed Grade	Protocol	Switch/PP
BCM16KA0CH0SX79RS	A0	C	H	0	S	X	7	9	R	S
BCM16KA0CH0SS51RS	A0	C	H	0	S	S	5	1	R	S
BCM16KA0CH0SM51RS	A0	C	H	0	S	M	5	1	R	S
BCM16KA0CH0SX51RS	A0	C	H	0	S	X	5	1	R	S
BCM16KA0CH0SM51RD	A0	C	H	0	S	M	5	1	R	D
BCM16KA0CH0SX51RD	A0	C	H	0	S	X	5	1	R	D
BCM16KA0CH0SS51LS	A0	C	H	0	S	S	5	1	L	S
BCM16KA0CH0SM51LS	A0	C	H	0	S	M	5	1	L	S
BCM16KA0CH0SX51LS	A0	C	H	0	S	X	5	1	L	S
BCM16KA0CH0SM51LD	A0	C	H	0	S	M	5	1	L	D
BCM16KA0CH0SX51LD	A0	C	H	0	S	X	5	1	L	D

Appendix A: Acronyms and Abbreviations

Table 112 lists the acronyms and abbreviations used in this document.

For a more complete list of acronyms and other terms used in Broadcom documents, go to: http://www.broadcom.com/press/glossary.php.

Table 112: Acronyms and Abbreviations

Term	Description
ACE	Access Control Entry
ACL	Access Control List
BA	Base Address
BCR	Block Configuration Register
BDPS	Billion Decisions Per Second
BMR	Block Mask Register
CB	Context Buffer
DCR	Device Configuration Register
ECC	Embedded Error Correction Circuitry
EEI	Energy Efficient Interlaken
HPM	Highest Priority Match
ILKN	Interlaken
KPU	Key Processing Unit
LP	Low Power Mode
LPM	Longest Prefix Match
LTR	Logical Table Register
LUT	Lookup Table
MDC	Management Data Clock
MMD	MDIO Manageable Devices
MSB	Most Significant Bit
ODT	On Die Termination
RAM	Random Access Memory
ROP	Records-Over-Packet
STA	Station Management Entity
UDA	User Data Array
UI	Unit Interval

Revision History

16000-DS122; April 20, 2020

Added:

- ELK SerDes Connectivity Options for J2C+ BCM88850

16000-DS121; April 1, 2020

Updated:

- General Description – Updated device connection information.
- Overview – Updated device connection information.
- Ordering Part Number Constituents – Updated information.
- Ordering Part Numbers – Added two new part numbers (BCM16KB0CH0SM51RSP and BCM16KB0CH0SX51RSP).

16000-DS120; May 17, 2019

Updated:

- Ordering Information

16000-DS119; May 8, 2019

Added:

- Device Usage Restrictions
- PCI Configuration Space STATUS Register Access

16000-DS118; January 16, 2019

Updated:

- Table 24, Test, Power, and Ground Pin Descriptions
- Table 98, Absolute Maximum Ratings
- Table 103, DC Electrical Specifications

16000-DS117; September 6, 2018

Updated:

- Ordering Information

16000-DS116; August 17, 2018

Updated:

- Table 24, Test, Power, and Ground Pin Descriptions
- Table 98, Absolute Maximum Ratings
- Table 103, DC Electrical Specifications

16000-DS115; August 10, 2018

Updated:

- Table 20, Clock, Configuration, and Reset Pin Descriptions
- Table 21, Interface Transmit, Receive, Clock, and Control Pin Descriptions

- Table 101, Performance Specifications
- Table 102, SerDes Data Rates

16000-DS114; June 21, 2018

Updated:

- Table 43, BCM52311 OP and BCM16K OP2 Configurations with Jericho Processors
- Table 50, BCM88690 JR2 + BCM16K OP2 Connection Example with 53G SerDes.
- Table 91, Dual Host Feature Summary
- Figure 42, Package Drawing

Added:

- Device Initialization and Power-Up/Down Sequence

16000-DS113; May 25, 2018

Updated:

- Clock, Configuration, and Reset Pin Descriptions

16000-DS112; May 18, 2018

Updated:

- Table 80, Latency Table
- Table 89, SerDes Data Rates
- Table 92, Ordering Part Number Constituents
- Table 93, Ordering Part Numbers

Added:

- Errata and Device Status

16000-DS111; February 27, 2018

Updated:

- Figure 2, Line Card
- Figure 4, Functional Block Diagram (with Broadcom Switch)
- Figure 5, Context Buffer and Key Processing Unit Block Diagram
- “PCIe Interface” on page 11
- Figure 21, Single-Host Configurations
- Figure 22, Dual and Quad-Host Configurations
- Figure 24, Report Engine through PCIe to Control Plane Processor
- Figure 27, BCM52311 OP and BCM16K OP2 Configurations with Jericho Processors
- “Search Lane Assignments in Single-Host Mode” on page 87
- Table 85, Absolute Maximum Ratings
- Table 88, Performance Specifications
- Table 90, DC Electrical Specifications

Added:

- Data Structure in PCI Control Plane Memory

16000-DS110; January 5, 2018

Updated:

- Figure 40, Package Drawing

  Package updates:

  – The current dimensions for the D and E package width/lengths vary from 37.4 mm to 37.6 mm with 37.5 mm being typical. The previous dimensions provided only the typical value.

  – The current dimension for the b ball size vary from 0.5 mm to 0.7 mm with 0.6 mm being typical. The previous dimensions provided only the typical value.

  – The current dimensions aaa, bbb, ccc, ddd, and eee give a maximum number. The previous dimensions provided only one number. This is referred to as x, y, z in the current drawing and A, B, C in the previous.

16000-DS109; December 8, 2017

Updated:

- Report Engine

Added:

- DMA Message Format
- Chapter 10, Statistics Record
- Chapter 11, Ethernet Packet Format

16000-DS108; November 14, 2017

Updated:

- Ball Assignment

Signal Name	Previous Ball Number	Updated Ball Number
VDDS5	E8	H8
VDDS5	F8	J8
VDDS4	H8	E8
VDDS4	J8	F8

16000-DS107; November 9, 2017

Updated:

- Ball Assignment

Table 113: Ball Number Changes

Signal Name	Previous Ball Number	Updated Ball Number
PCRXDATA_P	B17	A18
VDDPC	A18	B17
PCREFCLK_N	A19	A20
PCREFCLK_P	B19	B20
VSS	A20	A19
VSS	B20	B19
PCTESTN	C20	C19
VSS	C19	C20
RXDATA_P[33]	AF32	AB32

Table 113: Ball Number Changes

Signal Name	Previous Ball Number	Updated Ball Number
RXDATA_N[33]	AF33	AB33
RXDATA_P[36]	AB32	AF32
RXDATA_N[36]	AB33	AF33
RXDATA_P[44]	Y5	AD5
RXDATA_N[44]	Y4	AD4
RXDATA_P[40]	AD4	Y4
RXDATA_N[40]	AD5	Y5

16000-DS106; October 20, 2017

Updated:

- Minor updates and reorganization.

16000-DS105; August 21, 2017

Updated:

- “Electrical Features” on page 8.
- Table 54, Test, Power, and Ground Pin Descriptions
- “DC Electrical Specifications” on page 91

16000-DS104; August 08, 2017

Updated:

- Figure 36, Package Drawing

16000-DS103-R; August 04, 2017

Updated:

- Table 50, Clock, Configuration, and Reset Pin Descriptions
- Table 52, JTAG Pin Descriptions
- Table 53, Miscellaneous Pin Descriptions
- Table 54, Test, Power, and Ground Pin Descriptions
- Table 56, Latency Table
- Table 61, Performance Specifications
- Table 62, SerDes Data Rates

Added:

- Chapter 19, JTAG Boundary Scan

16000-DS102-R; May 31, 2017

Updated:

- Section 25: “Ordering Information,” on page 174

16000-DS101-R; December 15, 2016

Updated:

16000-DS100-R; October 27, 2016

Initial release.