

# SONET/SDH Demapper Core

## Multi-Channel

16 Input Streams (VCs)

512 Output Streams (TUs)

## 1. Features

The Aliathon Multi-channel SDH Demapper core provides a flexible, resource-efficient, programmable-logic based solution for demapping lower order PDH signals from SDH/SONET. The core is ideally suited for STM0, STM1 and STM4 inputs, and the core may be replicated in parallel to interface to STM16 and beyond. The core...

- Accepts up to 16 input VCs. These VCs can be a mix of VC3 and VC4.
- Demaps TUG3 structures from VC4 and TUG2 structures from VC3 and TUG3.
- Demaps TUs (TU11, TU12 and TU2) from within TUG2. The TU type can be dynamically configured on a per TUG2 basis. TU3 within TUG3 is also catered for, as is VC11 over TU12.
- Processes all TU pointers, including the TU3 pointer for TU3 over TUG3.
- Extracts all Lower-Order Path Overhead, and verifies BIP-2 calculations (B3 in the case of TU3 over TUG3).
- Demaps PDH signals from VC and TU payloads. All asynchronous, bit-synchronous and byte-synchronous mappings for TU11, TU12, TU2, VC3 and VC4 are implemented. The demapping type may be dynamically configured on a per-TU/VC basis.
- Provides a multi-channel output of up to 336 TUs (TU11 over STM4).
- Plugs directly into Aliathon's SDH Deframer cores for a complete SDH solution, and into Aliathon's PDH deframer cores (E1/T1 and E3/T3).

## 2. Functional Description

Figure 1 illustrates the major functional blocks within the SDH Demapper Core.

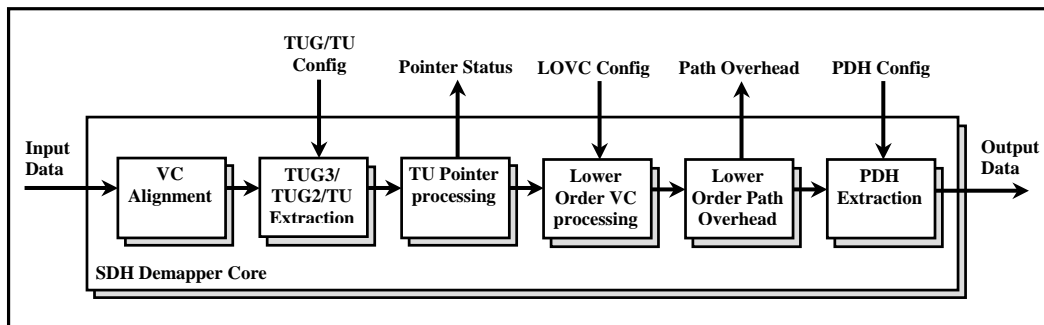


Figure 1 – SDH Demapper Core Functional Blocks

### 2.1. VC Alignment

The VC alignment block synchronises to the VC structure being passed into the core by the upstream logic.

## 2.2. TUG3/TUG2/TU Extraction

The TUG3/TUG2/TU Extraction block processes structured VC payloads and extracts TUs. All possible paths through TUG3 and TUG2 structures are supported (see figure 2), including TU3 over TUG3. The type of structure may be dynamically configured on a per VC basis, and hence the core can simultaneously extract TU3s from one VC4, TU12s from another VC4 and TU11s from VC3s (as an example).

## 2.3. TU Pointer Processing

The TU Pointer Processing block process all TU pointers (up to 336 TU11 pointers over STM4) and extracts aligned lower-order VCs, allowing for pointer movement. This includes VC3 over TU3.

## 2.4. Lower Order VC Processing

The Lower-Order VC block handles the VC11 in TU12 mapping, which may be dynamically configured on a per TU basis.

## 2.5. Lower Order Path Overhead

The Lower Order Path Overhead block calculates and verifies the BIP-2 value(B3 in the case of VC3 over TU3) for each lower-order VC. It also extracts the path overhead for all of the VCs.

## 2.6. PDH Extraction

The PDH extraction block demaps PDH signals from TU payloads. All defined asynchronous, bit synchronous and byte synchronous mappings are supported for all VCs. The demapping type may be dynamically con

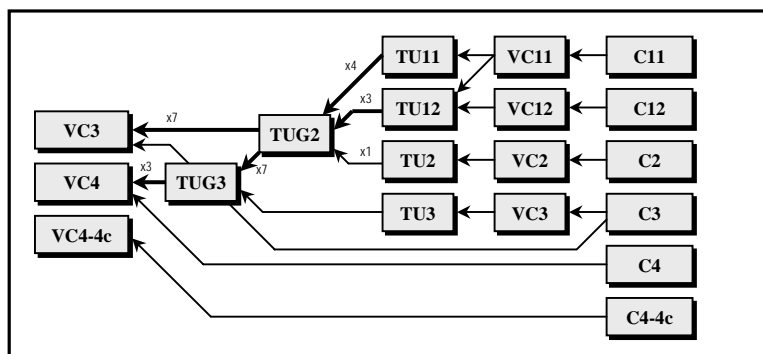


Figure 2 – TU to VC Multiplex Structure

### 3. Signal Description

The input and output signals are grouped by function into the following interfaces.

- Downstream Timing/Data input
- Upstream Timing/Data output
- Status and configuration

#### 3.1. Downstream Timing/Data Input

This interface will typically be driven by a downstream device generating SDH and VC timing and data signals, such as Aliathon's SDH deframer core.

Name	Type	Description
<b>reset</b>	I	Asynchronous reset input. Active high.
<b>frm_clk</b>	I	<b>frm_clk</b> is the main clock for the core. All inputs and outputs are synchronous to this clock unless otherwise noted.
<b>dti_sdh_vld</b>	I	When asserted this indicates that the following inputs are valid. In many applications this input can be tied high.
<b>dti_sdh_sof</b>	I	When asserted this indicates that clock cycle is the first A1 byte of the SDH frame. This input is not essential for the operation of the core and may be tied low.
<b>dti_sdh_colcnt (3..0)</b>	I	This input indicates the SDH sub-column valid in this cycle. It will always be 0 for STM0, range from 0 (0000) to 2 (0010) for STM1 and from 0 (0000) to 11 (1011) for STM4.
<b>dti_sdh_toh</b>	I	This input indicates that this clock cycle is fixed transport overhead (ie: Regenerator Section Overhead, Multiplex Section Overhead and AU pointers).
<b>dti_sdh_size (1..0)</b>	I	This input indicates the type of SDH frame being generated. Valid values are... 00 : STM0 01 : STM1 10 : STM4
<b>dti_vc_id (3..0)</b>	I	This input identifies which VC is currently valid. If the input data is all from a single VC (for example, VC3 over STM0 or VC4 over STM1) then this input may be tied to 0 (0000). If the input is from a number of VCs (for example, VC3 or VC4 over STM4) then this input may range from 0 (0000) to 11 (1011).
<b>dti_vc_size (1..0)</b>	I	This input indicates the type of VC that is currently valid. Defined values are... 00 : VC3 01 : VC4 10 : VC4-4c

Name	Type	Description
<b>dti_vc_vld</b>	l	This input indicates that the VC indicated by <b>dti_vc_id</b> is valid. This input is deasserted for transport overhead.
<b>dti_vc_pldvld</b>	l	When asserted this input indicates valid VC payload. This excludes VC Path Overhead and stuff columns.
<b>dti_vc_j1</b>	l	When asserted this indicates that the J1 byte for the VC indicated by <b>dti_vc_id</b> is currently valid.
<b>dti_vc_poh</b>	l	When asserted this indicates that the VC Path Overhead for the VC indicated by <b>dti_vc_id</b> is currently valid.
<b>dti_vc_h4cnt</b> <b>(1..0)</b>	l	This counter indicates the H4 multiframe count for structured VC payloads. It ranges between 0 (00) and 3 (11).
<b>dti_vc_ptr</b> <b>(2..0)</b>	l	This input indicates AU pointer actions for the current VC. It is encoded as... 000 : No action 001 : Pointer Increment 010 : Pointer Decrement 100 : New pointer (no NDF) 101 : New pointer (valid NDF)
<b>dti_data</b> <b>(7..0)</b>	l	Byte wide input data, aligned to the above timing inputs.

### 3.2. Downstream Timing/Data Output

This interface will typically drive a downstream device with SDH, VC and TU timing signals and data, such as Aliathon's PDH deframer cores (DS1, T1, DS3, T3, etc).

Name	Type	Description
<b>dto_sdh_vld</b>	O	When asserted this indicates that the following outputs are valid.
<b>dto_sdh_sof</b>	O	When asserted this indicates that this clock cycle is the first A1 byte of the SDH frame.
<b>dto_sdh_colcnt (3..0)</b>	O	This output indicates the SDH sub-column valid in this cycle. It will always be 0 for STM0, range from 0 (0000) to 2 (0010) for STM1 and from 0 (0000) to 11 (1011) for STM4.
<b>dto_sdh_toh</b>	O	This output indicates that this clock cycle is fixed transport overhead (ie: Regenerator Section Overhead, Multiplex Section Overhead and AU pointers).
<b>dto_sdh_size (1..0)</b>	O	This output indicates the type of SDH frame being generated. Valid values are... 00 : STM0 01 : STM1 10 : STM4
<b>dto_vc_id (3..0)</b>	O	This output identifies which VC is currently valid. If the input data is all from a single VC (for example, VC3 over STM0 or VC4 over STM1) then this input may be tied to 0 (0000). If the input is from a number of VCs (for example, VC3 or VC4 over STM4) then this input may range from 0 (0000) to 11 (1011).
<b>dto_vc_size (1..0)</b>	O	This output indicates the type of VC that is currently valid. Defined values are... 00 : VC3 01 : VC4 10 : VC4-4c
<b>dto_vc_vld</b>	O	This output indicates that the VC indicated by <b>dto_vc_id</b> is valid. This output is deasserted for transport overhead.
<b>dto_vc_pldvid</b>	O	When asserted this output indicates valid VC payload. This excludes VC Path Overhead and stuff columns.
<b>dto_vc_j1</b>	O	When asserted this indicates that the J1 byte for the VC indicated by <b>dto_vc_id</b> is currently on.
<b>dto_vc_poh</b>	O	When asserted this indicates that the VC Path Overhead for the VC indicated by <b>dto_vc_id</b> is currently valid.
<b>dto_vc_h4cnt (1..0)</b>	O	This counter indicates the H4 multiframe count for structured VC payloads. It ranges between 0 (00) and 3 (11).
<b>dto_vc_ptr (2..0)</b>	O	This output indicates AU pointer actions for the current VC. It is encoded as... 000 : No action 001 : Pointer Increment 010 : Pointer Decrement 100 : New pointer (no NDF) 101 : New pointer (valid NDF)

Name	Type	Description
<b>dto_lovc_</b> <b>enb</b>	○	This output is asserted when the current VC (indicated by <b>dto_vc_id</b> ) is carrying a structured payload (ie: carrying Lower Order VCs).
<b>dto_lovc_id</b> <b>(8..0)</b>	○	This output indicates the current Lower Order VC. See section 4.2 for a format description.
<b>dto_lovc_</b> <b>size (2..0)</b>	○	This indicates the type of lower-order VC. Defined values are... 000 : VC11 001 : VC12 010 : VC2 011 : VC3 100 : VC11 (via TU12)
<b>dto_lovc_</b> <b>vld</b>	○	When asserted this indicates that a Lower Order VC is currently valid.
<b>dto_lovc_</b> <b>pldvld</b>	○	When asserted this indicates that Lower Order VC payload is currently valid.
<b>dto_lovc_v5</b>	○	When asserted the V5 byte (J1 in the case of Lower Order VC3) for the VC indicated by <b>dto_lovc_id</b> is currently valid.
<b>dto_lovc_</b> <b>poh</b>	○	When asserted Lower Order Path Overhead for the VC indicated by <b>dto_lovc_id</b> is currently valid.
<b>dto_lovc_</b> <b>ptr (2..0)</b>	○	This output indicates TU pointer actions for the current Lower Order VC. It is encoded as... 000 : No action 001 : Pointer Increment 010 : Pointer Decrement 100 : New pointer (no NDF) 101 : New pointer (valid NDF)
<b>dto_pdh_</b> <b>enb</b>	○	When asserted this indicates that the current VC (Higher or Lower Order) is carrying a PDH mapping.
<b>dto_pdh_</b> <b>map (2..0)</b>	○	This output indicates the type of PDH mapping used. Defined as... 000 : Demapping disabled. 001 : Asynchronous Mapping (type 1) 010 : Asynchronous Mapping (type 2) 011 : Bit synchronous 100 : Byte synchronous (type 1) 101 : Byte synchronous (type 2)  Refer to section 4.3 for further details...
<b>dto_pdh_</b> <b>vld</b>	○	This output is asserted high when <b>dto_pdh_bits</b> is not 0 (0000).
<b>dto_pdh_</b> <b>bits (3..0)</b>	○	This output indicates the number of valid PDH payload bits in this clock cycle. It ranges between 0 (0000) and 8 (1000).

Name	Type	Description
<b>dto_pdh_ohid (2..0)</b>	○	The format of this output is currently undocumented.
<b>dto_pdh_stuff (1..0)</b>	○	This output indicates the state of PDH stuffing for the VC indicated by <b>dto_lovc_id</b> . It is defined as... 00 : No stuffing 01 : Positive stuffing – resulting PDH rate is marginally slower 10 : Negative stuffing – resulting PDH rate is marginally faster
<b>dto_data (7..0)</b>	○	Output data, aligned with the above timing outputs.

### 3.3. Status and Configuration Interface

#### 3.3.1. TUG3/TUG2/TU Extraction

This interface configures how TUG3s, TUG2s and TUs are extracted from the input VCs.

Name	Type	Description
<b>tu_vc_id (3..0)</b>	O	This output indicates what VC is requesting TUG3 configuration information.
<b>tu_tug3</b>	I	This input configures whether TUG3 is extracted from input VCs or not (set high to extract TUG3). It should be valid for the VC indicated by <b>tu_vc_id</b> in the preceding clock cycle. If all VCs have the same setting then this input may be hardwired to a set value. If different settings are to be applied to different VCs then this input may be driven by the output of a RAM, with <b>tu_vc_id</b> as the address input.
<b>tu_vctug3_id (3..0)</b>	O	This output indicates what VC/TUG3 is requesting TUG2 configuration information.
<b>tu_tug2</b>	I	This input configures whether TUG2 is extracted from input VCs/TUG3s or not (set high to extract TUG2). It should be valid for the VC/TUG3 indicated by <b>tu_vctug3_id</b> in the preceding clock cycle. If all VC/TUG3s have the same setting then this input may be hardwired to a set value. If different settings are to be applied to different VC/TUG3s then this input may be driven by the output of a RAM, with <b>tu_vctug3_id</b> as the address input.
<b>tu_tu_id (6..0)</b>	O	This output indicates what TUG is requesting TU configuration information. The most significant 4 bits indicate the VC/TUG number, whilst the least significant 3 bits indicate the TUG2 number.
<b>tu_tu_type (1..0)</b>	I	This input configures what type of TU is extracted from TUG2s and TUG3s. It should be valid for the TUG3/TUG2 indicated by <b>tu_tu_id</b> in the preceding clock cycle. If all TUG3/TUG2s have the same setting then this input may be hardwired to a set value. If different settings are to be applied to different TUG3/TUG2s then this input may be driven by the output of a RAM, with <b>tu_tu_id</b> as the address input. Defined values are... 00 : TU11 01 : TU12 10 : TU2 11 : TU3

### 3.3.2. TU Pointer Status

This output provides TU pointer status.

Name	Type	Description
<b>tptr_id</b> (8..0)	O	Indicates which TU the following status outputs are for. Refer to section 4.2 for a description of the structure of this output.
<b>tptr_byte_vld</b>	O	Indicates that a TU pointer byte is available on the <b>tptr_byte</b> output, for the TU indicated by <b>tptr_trib</b> .
<b>tptr_byte</b> (7..0)	O	The TU pointer byte.
<b>tptr_byte_type</b> (1..0)	O	This output indicates the type of TU byte on the <b>tptr_byte</b> output. Defined values are... 00 : V1 01 : V2 10 : V3 11 : V4
<b>tptr_status</b> (1..0)	O	The status of the TU pointer processing state machine for the TU indicated by <b>tptr_trib</b> . Defined values are 00 : Loss of Pointer 01 : AIS (Alarm Indication Signal) 10 : Normal 11 : CI (Concatenation Indication)
<b>tptr_inc</b>	O	Indicates that there has been a pointer increment for the TU indicated by <b>tptr_trib</b> .
<b>tptr_dec</b>	O	Indicates that there has been a pointer decrement for the TU indicated by <b>tptr_trib</b> .
<b>tptr_new</b>	O	Indicates that a new pointer value has been accepted for the TU indicated by <b>tptr_trib</b> , based on 3 identical, consecutive pointers being received.
<b>tptr_ndf</b>	O	Indicates that a new pointer value has been accepted for the TU indicated by <b>tptr_trib</b> , based on an NDF (New Data Flag) being received.
<b>tptr_val</b> (9..0)	O	The currently accepted TU pointer value for the TU indicated by the <b>tptr_trib</b> output.

### 3.3.3. Lower Order VC Configuration

This interface configures how Lower Order VCs are extracted from the input TUs.

Name	Type	Description
<b>lvc_cfg_id (8..0)</b>	O	This output indicates what VC/TU is requesting Lower Order VC configuration information. Refer to section 4.2 for a description of the structure of this input.
<b>lvc_11t12</b>	I	This input configures whether VC11 over TU12 is extracted from input TUs or not (set high to enable VC11 over TU12). It should be valid for the TU indicated by <b>lvc_cfg_id</b> in the preceding clock cycle. If all VCs/TUs have the same setting then this input may be hardwired to a set value. If different settings are to be applied to different VCs/TUs then this input may be driven by the output of a RAM, with <b>lvc_cfg_id</b> as the address input.

### 3.3.4. Lower Order Path Overhead

This interface implements a generic Lower Order Path Overhead output.

Name	Type	Description
<b>lpoh_id (8..0)</b>	O	Indicates which TU the following status outputs are for. Refer to section 4.2 for a description of the structure of this output.
<b>lpoh_bip_err</b>	O	When asserted high this indicates that a BIP-2 (or B3 in the case of VC3 over TU3) error has been detected for the TU indicated by <b>lpoh_trib</b> .
<b>lpoh_num_bip (3..0)</b>	O	This reports the number of BIP-2 (or B3) errors detected. For TU11, TU12 and TU2 this will range between 0 (0000) and 2 (0010), and for TU3, 0 (0000) and 8 (10000).
<b>lpoh_vld</b>	O	When asserted this indicates that the following Lower Order Path Overhead outputs are valid.
<b>lpoh_data (7..0)</b>	O	This output carries the byte wide Lower Order Path Overhead.
<b>lpoh_type (3..0)</b>	O	<p>This output indicates what type of Lower Order Path Overhead is available on the <b>lpoh_data</b> output. For TU11, TU12 and TU2 defined values are.</p> <p>0000 : V5  0001 : J2  0010 : N2  0011 : K4</p> <p>If the TU is a TU3 then it is encoded as...</p> <p>0000 : J1  0001 : B3  0010 : C2  0011 : G1  0100 : F2  0101 : H4  0110 : F3  0111 : K3  1000 : N1</p>

### 3.3.5. PDH Demapping

This interface implements PDH demapping status and configuration.

Name	Type	Description
<b>pdh_cfg_id</b> (8..0)	O	This output indicates what VC/TU is requesting PDH demapping configuration information. Refer to section 4.3 for a description of the structure of this output.
<b>pdh_type</b> (2..0)	I	<p>This input configures what kind of PDH mapping is applied to VCs/TUs. It should be valid for the TU indicated by <b>pdh_cfg_id</b> in the preceding clock cycle. If all VCs/TUs have the same PDH mapping then this input may be hardwired to a set value. If different mappings are to be applied to different VCs/TUs then this input may be driven by the output of a RAM, with <b>pdh_cfg_id</b> as the address input.</p> <p>Defined values are...</p> <p>000 : Demapping disabled.            001 : Asynchronous Mapping (type 1)            010 : Asynchronous Mapping (type 2)            011 : Bit synchronous            100 : Byte synchronous (type 1)            101 : Byte synchronous (type 2)</p> <p>See section 4.3 for further details.</p>
<b>pdh_stf_vld</b>	O	When asserted this output indicates that the <b>pdh_stf_bits</b> output is valid for the VC/TU indicated by <b>pdh_stf_id</b> .
<b>pdh_stf_bits</b> (1..0)	O	This output can be used to measure the rate of the PDH signal mapped into the VC/TU payload. For every stuff opportunity, this output indicates the number of valid data bits. As some PDH mapping types have two potential stuff opportunities in a single TU payload byte this output ranges between 0 (00) and 2 (10).
<b>pdh_stf_id</b> (8..0)	O	This output indicates which VC/TU the <b>pdh_stf_bits</b> output is for. Refer to section 4.2 for a description of the structure of this output.

## 4. Implementation Details

### 4.1. Resource Utilisation

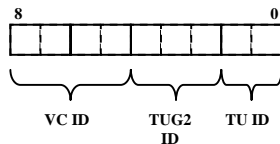
The following figures are calculated assuming that all core IOs are routed off-chip. This results in a worst-case resource utilisation figure, and for any given application the resource utilisation is likely to be lower. The example parts are the slowest (and hence cheapest) offered speed-grade, and the core exceeds performance requirements (77.76MHz for STM4) in these devices.

	Stratix Family Eg : EP1S10F484C7			Stratix GX Family Eg : EP1SGX10CF672C7			Cyclone Family Eg : EP1C12F324C8		
	Used by core	In example part	Percentage used	Used by core	In example part	Percentage used	Used by core	In example part	Percentage used
<b>Logic Elements (LEs)</b>	2007	10570	19%	2007	10570	19%	1987	12060	16%
<b>M512 RAM Blocks</b>	10	94	10.5%	10	94	10.5%		n/a	
<b>M4k RAM Blocks</b>	11	60	18.5%	11	60	18.5%	21	52	38%
<b>M-RAM Blocks</b>	0	1	0%	0	1	0%		n/a	
<b>Fmax</b>	95MHz			95MHz			88MHz		

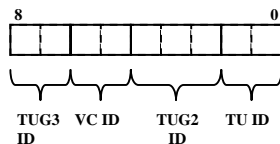
	Stratix2 Family Eg : EP2S15F484C5		
	Used by Core	In example Part	Percentage used
<b>ALMs</b>	1049	6240	17%
<b>M512 RAM Blocks</b>	5	104	4%
<b>M4k RAM Blocks</b>	10	78	12%
<b>M-RAM Blocks</b>	0	1	0%
<b>Fmax</b>	100MHz		

## 4.2. VC/TUG2/TU Identification

Throughout the interfaces a 9 bit pointer is used to identify specific TUGs and TUs. There may be up to 336 TU11s in an STM4, hence the 9 bits of the pointer. For VC3, the resulting 9 bit pointer format is...



For a VC4 an extra field is required for the TUG3 information, but there are fewer possible VCs, so the resulting format is...



## 4.3. PDH Mapping

The core implements all valid mappings of PDH signals from SDH containers. The following table relates the core configuration with container size and mapping type. Asynchronous mappings have no fixed phase or frequency relationship between the SDH container and the PDH signal – bit stuffing is performed to accommodate frequency differences. Bit Synchronous mappings have no fixed phase relationship between the SDH container and the PDH signal, but no bit stuffing is performed and thus the resulting PDH frequency is fixed. Byte Synchronous mappings have both fixed phase and frequency.

		Container Size				
		C11	C12	C2	C3	C4
000	Disabled					
001	Async. (Type 1)	DS1 :1.544mbps	E1 :2.048mbps	DS2 : 6.312mbps	DS3 : 44.736mbps	E4 : 139.264mbps
010	Async. (Type 2)				E3 : 34.368mbps	
011	Bit Sync.	DS1 :1.544mbps		DS2 : 6.312mbps		
100	Byte Sync. (Type 1)	DS1 :1.544mbps	E1 :2.048mbps			
101	Byte Sync. (Type 2)	4 x 384kbps	31 x 64kbps			

#### **4.4. Ordering Information**

For technical enquiries and ordering please contact Aliathon Ltd at

[enquires@aliathon.com](mailto:enquires@aliathon.com)

Aliathon Ltd  
Evans Business Centre  
Pitreavie Court, Dunfermline  
United Kingdom  
KY11 5UU

Phone +44 (0)1383 737 736

Fax +44 (0)1383 749 501

<http://www.aliathon.com/>