

# SONET/SDH Mapper Core

## Multi-Channel

16 Output Streams (VCs)

512 Input Streams (TUs)

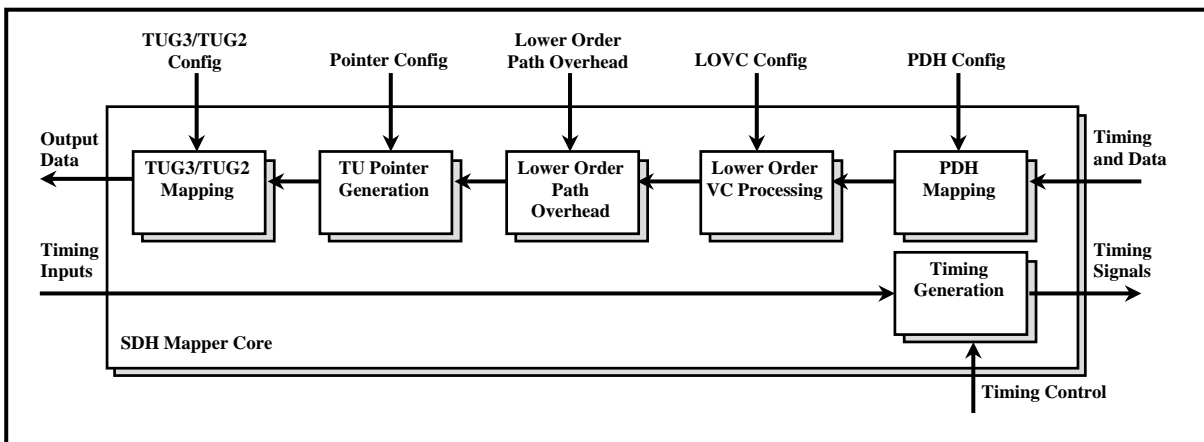
## 1. Features

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

- Generates up to 16 VC payloads. These VCs can be a mix of VC3 and VC4.
- Maps TUG3 structures into VC4 and TUG3 structures into VC3 and TUG3.
- Maps TUs (TU11, TU12 and TU2) into 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.
- Generates all TU pointers, including the TU3 pointer for TU3 over TUG3.
- Inserts all Lower-Order Path Overhead, and calculates BIP-2 (B3 in the case of TU3 over TUG3).
- Maps PDH signals into VC and TU payloads. All asynchronous, bit-synchronous and byte-synchronous mappings for TU11, TU12, TU2, VC3 and VC4 are implemented. The mapping type may be dynamically configured on a per-TU/VC basis.
- Accepts a multi-channel input of up to 336 TU payloads (TU11 over STM4).
- Plugs directly into Aliathon's SDH Framers cores for a complete SDH solution, and into Aliathon's PDH framer cores (E1/T1 and E3/T3).

## 2. Functional Description

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



**Figure 1 – SDH Mapper Core Functional Blocks**

### 2.1. TUG3/TUG2 Mapping

The TUG3/TUG2 Mapping block packages TUs into VC payloads via TUG3 and TUG2 multiplex structures. Every VC may be assigned a different TUG mapping type, every TUG3 may be configured to carry TUG2 or TU3, and each TUG2 may be assigned a different TU type (TU11,

TU12 or TU2). Configuration may be changed dynamically. Figure 2 shows the supported multiplex structure.

## 2.2. TU Pointer Processing

The TU Pointer Processor block generates TU pointers for all configured TUs. Pointer movements and new pointer values may optionally be inserted.

## 2.3. Lower Order Path Overhead

This block generates the BIP-2 value (or B3, in the case of Lower Order VC3) for all TUs and inserts it into the Path Overhead. It also provides a generic interface enabling external logic to insert Lower Order Path Overhead and to error Path Overhead for test purposes.

## 2.4. Lower Order VC Processing

The Lower Order VC Processing block implements VC11 over TU12, if required.

## 2.5. PDH Mapping

The PDH mapping block implements PDH signal mapping into VC/TU payloads. All asynchronous, bit synchronous and byte synchronous mappings are implemented, for DS1, E1, DS2, DS3, E3 and E4 PDH signals.

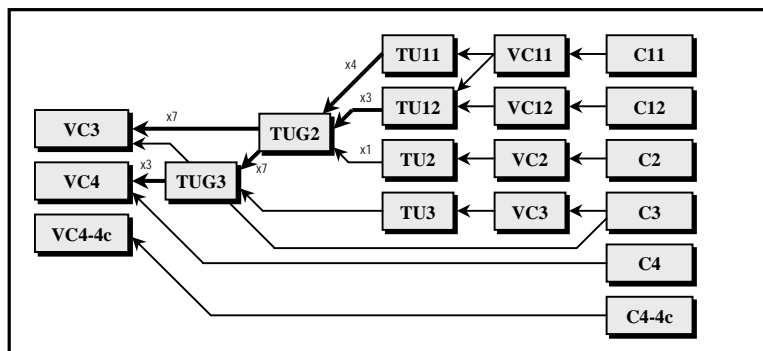


Figure 2 – TU to VC Multiplex Structure

### 3. Signal Description

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

- Downstream timing input (eg: from SDH framer)
- Downstream timing/data output (eg: to SDH upframer)
- Upstream timing output (eg: to PDH framers)
- Upstream timing/data input (eg: from PDH framers)
- Status and configuration

#### 3.1. Downstream Timing Input

This interface will typically connect to a downstream device generating SDH and VC timing signals, such as Aliathon's SDH framer core.

| Name                         | Type | Description  |
|------------------------------|------|--|
| <b>reset</b>                 | I    | Asynchronous reset input. Active high.   |
| <b>tx_clk</b>                | I    | <b>tx_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...<br>00 : STM0<br>01 : STM1<br>10 : STM4   |
| <b>dti_vc_id</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</b>           | I    | This input indicates the type of VC that is currently valid. Defined values are...<br>00 : VC3<br>01 : VC4<br>10 : VC4-4c  |

| Name                       | Type | Description   |
|----------------------------|------|---|
| <b>dti_vc_vld</b>          | I    | 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_pldvd</b>        | I    | When asserted this input indicates valid VC payload. This excludes VC Path Overhead and stuff columns.  |
| <b>dti_vc_j1</b>           | I    | 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>          | I    | 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 (1..0)</b> | I    | This counter indicates the H4 multiframe count for structured VC payloads. It ranges between 0 (00) and 3 (11).   |
| <b>dti_vc_ptr (2..0)</b>   | I    | This input indicates AU pointer actions for the current VC. It is encoded as...<br>000 : No action<br>001 : Pointer Increment<br>010 : Pointer Decrement<br>100 : New pointer (no NDF)<br>101 : New pointer (valid NDF) |

### 3.2. Downstream Timing/Data Output

This interface will typically connect to a downstream device accepting SDH and VC timing signals and frame data, such as Aliathon's SDH framer core.

| 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 the data on <b>dto_data</b> 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 the data on <b>dto_data</b> 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...<br>00 : STM0<br>01 : STM1<br>10 : STM4   |
| <b>dto_vc_id</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</b>           | O    | This output indicates the type of VC that is currently valid. Defined values are...<br>00 : VC3<br>01 : VC4<br>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_data</b> .  |
| <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).   |

| Name                         | Type | Description  |
|------------------------------|------|--|
| <b>dto_vc_ptr<br/>(2..0)</b> | O    | This output indicates AU pointer actions for the current VC. It is encoded as...<br>000 : No action<br>001 : Pointer Increment<br>010 : Pointer Decrement<br>100 : New pointer (no NDF)<br>101 : New pointer (valid NDF)   |
| <b>dto_vc_increq</b>         | O    | This is an out-of-band request for an AU pointer increment, and should be used for rate-adaptation (for example, if the VC data is provided via an asynchronous fifo that is becoming full). If it is asserted the downstream Timing Generator will cause an AU pointer increment for the current VC ( <b>dto_vc_id</b> ) in the next frame. |
| <b>dto_vc_decreq</b>         | O    | This operates the same as <b>dto_vc_increq</b> but for pointer decrements.   |
| <b>dto_data</b>              | O    | VC payload data for transmission.  |

### 3.3. Upstream Timing Output

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

| Name                         | Type | Description   |
|------------------------------|------|---|
| <b>uto_sdh_vld</b>           | O    | When asserted this indicates that the following outputs are valid.  |
| <b>uto_sdh_sof</b>           | O    | When asserted this indicates that this clock cycle is the first A1 byte of the SDH frame.   |
| <b>uto_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>uto_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>uto_sdh_size (1..0)</b>   | O    | This output indicates the type of SDH frame being generated. Valid values are...<br>00 : STM0<br>01 : STM1<br>10 : STM4   |
| <b>uto_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>uto_vc_size (1..0)</b>    | O    | This output indicates the type of VC that is currently valid. Defined values are...<br>00 : VC3<br>01 : VC4<br>10 : VC4-4c  |
| <b>uto_vc_vld</b>            | O    | This output indicates that the VC indicated by <b>uto_vc_id</b> is valid. This output is deasserted for transport overhead.   |
| <b>uto_vc_pldvid</b>         | O    | When asserted this output indicates valid VC payload. This excludes VC Path Overhead and stuff columns.   |
| <b>uto_vc_j1</b>             | O    | When asserted this indicates that the J1 byte for the VC indicated by <b>uto_vc_id</b> is currently on.   |
| <b>uto_vc_poh</b>            | O    | When asserted this indicates that the VC Path Overhead for the VC indicated by <b>uto_vc_id</b> is currently valid.   |
| <b>uto_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>uto_vc_ptr (2..0)</b>     | O    | This output indicates AU pointer actions for the current VC. It is encoded as...<br>000 : No action<br>001 : Pointer Increment<br>010 : Pointer Decrement<br>100 : New pointer (no NDF)<br>101 : New pointer (valid NDF)  |

| Name                        | Type | Description  |
|-----------------------------|------|--|
| <b>uto_lovc_enb</b>         | ○    | This output is asserted when the current VC (indicated by <b>uto_vc_id</b> ) is carrying a structured payload (ie: carrying Lower Order VCs).  |
| <b>uto_lovc_id (8..0)</b>   | ○    | This output indicates the current Lower Order VC. See section 4.2 for a format description.  |
| <b>uto_lovc_size (2..0)</b> | ○    | This indicates the type of lower-order VC. Defined values are...<br>000 : VC11<br>001 : VC12<br>010 : VC2<br>011 : VC3<br>100 : VC11 (via TU12)  |
| <b>uto_lovc_vld</b>         | ○    | When asserted this indicates that a Lower Order VC is currently valid.   |
| <b>uto_lovc_pldvld</b>      | ○    | When asserted this indicates that Lower Order VC payload is currently valid.   |
| <b>uto_lovc_v5</b>          | ○    | When asserted the V5 byte (J1 in the case of Lower Order VC3) for the VC indicated by <b>uto_lovc_id</b> is currently valid.   |
| <b>uto_lovc_poh</b>         | ○    | When asserted Lower Order Path Overhead for the VC indicated by <b>uto_lovc_id</b> is currently valid.   |
| <b>uto_lovc_ptr (2..0)</b>  | ○    | This output indicates TU pointer actions for the current Lower Order VC. It is encoded as...<br>000 : No action<br>001 : Pointer Increment<br>010 : Pointer Decrement<br>100 : New pointer (no NDF)<br>101 : New pointer (valid NDF)   |
| <b>uto_pdh_enb</b>          | ○    | When asserted this indicates that the current VC (Higher or Lower Order) is carrying a PDH mapping.  |
| <b>uto_pdh_map (2..0)</b>   | ○    | This output indicates the type of PDH mapping used. Defined as...<br>000 : Demapping disabled.<br>001 : Asynchronous Mapping (type 1)<br>010 : Asynchronous Mapping (type 2)<br>011 : Bit synchronous<br>100 : Byte synchronous (type 1)<br>101 : Byte synchronous (type 2)<br><br>Refer to section 4.3 for further details... |
| <b>uto_pdh_vld</b>          | ○    | This output is asserted high when <b>uto_pdh_bits</b> is not 0 (0000).   |
| <b>uto_pdh_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>uto_pdh_ohid (1..0)</b>  | ○    | The format of this output is currently undocumented.   |
| <b>uto_pdh_stuff (1..0)</b> | ○    | This output indicates the state of PDH stuffing for the VC indicated by <b>uto_lovc_id</b> . It is defined as...<br>00 : No stuffing<br>01 : Positive stuffing – resulting PDH rate is marginally slower<br>10 : Negative stuffing – resulting PDH rate is marginally faster |

### 3.4. Upstream Timing/Data Input

This interface will typically be driven by an upstream device providing SDH, VC and TU timing signals and data, such as Aliathon's PDH framer cores (E1, DS1, E3, DS3, etc).

| Name                         | Type | Description  |
|------------------------------|------|--|
| <b>uti_sdh_vld</b>           | I    | When asserted this indicates that the following outputs are valid.   |
| <b>uti_sdh_sof</b>           | I    | When asserted this indicates that this clock cycle is the first A1 byte of the SDH frame.  |
| <b>uti_sdh_colcnt (3..0)</b> | I    | This input indicates the SDH sub-column valid in this cycle. It should always be 0 for STM0, range from 0 (0000) to 2 (0010) for STM1 and from 0 (0000) to 11 (1011) for STM4.   |
| <b>uti_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>uti_sdh_size (1..0)</b>   | I    | This input indicates the type of SDH frame being generated. Valid values are...<br>00 : STM0<br>01 : STM1<br>10 : STM4   |
| <b>uti_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>uti_vc_size (1..0)</b>    | I    | This input indicates the type of VC that is currently valid. Defined values are...<br>00 : VC3<br>01 : VC4<br>10 : VC4-4c  |
| <b>uti_vc_vld</b>            | I    | This input indicates that the VC indicated by <b>uti_vc_id</b> is valid. This input should be deasserted for transport overhead.   |
| <b>uti_vc_pldvid</b>         | I    | When asserted this input indicates valid VC payload. This excludes VC Path Overhead and stuff columns.   |
| <b>uti_vc_j1</b>             | I    | When asserted this indicates that the J1 byte for the VC indicated by <b>uti_vc_id</b> is currently on.  |
| <b>uti_vc_poh</b>            | I    | When asserted this indicates that the VC Path Overhead for the VC indicated by <b>uti_vc_id</b> is currently valid.  |
| <b>uti_vc_h4cnt (1..0)</b>   | I    | This counter indicates the H4 multiframe count for structured VC payloads. It ranges between 0 (00) and 3 (11).  |
| <b>uti_vc_ptr (2..0)</b>     | I    | This input indicates AU pointer actions for the current VC. It is encoded as...<br>000 : No action<br>001 : Pointer Increment<br>010 : Pointer Decrement<br>100 : New pointer (no NDF)<br>101 : New pointer (valid NDF)  |

| Name                        | Type | Description  |
|-----------------------------|------|--|
| <b>uti_vc_increq</b>        | I    | This is an out-of-band request for an AU pointer increment, and should be used for rate-adaptation (for example, if the VC data is provided via an asynchronous fifo that is becoming full). If it is asserted the downstream Timing Generator will cause an AU pointer increment for the current VC ( <b>uti_vc_id</b> ) in the next frame. |
| <b>Uti_vc_decreq</b>        | I    | This operates the same as <b>uti_vc_increq</b> but for pointer decrements.   |
| <b>uti_lovc_enb</b>         | I    | This input is asserted when the current VC (indicated by <b>uti_vc_id</b> ) is carrying a structured payload (ie: carrying Lower Order VCs).   |
| <b>uti_lovc_id (8..0)</b>   | I    | This input indicates the current Lower Order VC. See section 4.2 for a format description.   |
| <b>uti_lovc_size (2..0)</b> | I    | This indicates the type of lower-order VC. Defined values are...<br>000 : VC11<br>001 : VC12<br>010 : VC2<br>011 : VC3<br>100 : VC11 (via TU12)  |
| <b>uti_lovc_vld</b>         | I    | When asserted this indicates that a Lower Order VC is currently valid.   |
| <b>uti_lovc_pldvld</b>      | I    | When asserted this indicates that Lower Order VC payload is currently valid.   |
| <b>uti_lovc_v5</b>          | I    | When asserted the V5 byte (J1 in the case of Lower Order VC3) for the VC indicated by <b>uti_lovc_id</b> is currently valid.   |
| <b>uti_lovc_poh</b>         | I    | When asserted Lower Order Path Overhead for the VC indicated by <b>uti_lovc_id</b> is currently valid.   |
| <b>uti_lovc_ptr (2..0)</b>  | I    | This output indicates TU pointer actions for the current Lower Order VC. It is encoded as...<br>000 : No action<br>001 : Pointer Increment<br>010 : Pointer Decrement<br>100 : New pointer (no NDF)<br>101 : New pointer (valid NDF)   |
| <b>uti_lovc_increq</b>      | I    | This is an out-of-band request for a TU pointer increment, and should be used for rate-adaptation (for example, if the TU data is provided via an asynchronous fifo that is becoming full). If it is asserted the downstream Timing Generator will cause a TU pointer increment for the current TU ( <b>uti_lovc_id</b> ) in the next frame. |
| <b>Uti_lovc_decreq</b>      | I    | This operates the same as <b>uti_lovc_increq</b> but for pointer decrements.   |

| Name                        | Type | Description  |
|-----------------------------|------|--|
| <b>uti_pdh_enb</b>          | l    | When asserted this indicates that the current VC (Higher or Lower Order) is carrying a PDH mapping.  |
| <b>uti_pdh_map (2..0)</b>   | l    | This input indicates the type of PDH mapping used. Defined as...<br>000 : Demapping disabled.<br>001 : Asynchronous Mapping (type 1)<br>010 : Asynchronous Mapping (type 2)<br>011 : Bit synchronous<br>100 : Byte synchronous (type 1)<br>101 : Byte synchronous (type 2)<br><br>Refer to section 4.3 for further details...  |
| <b>uti_pdh_vld</b>          | l    | This input is asserted high when <b>uti_pdh_bits</b> is not 0 (0000).  |
| <b>uti_pdh_bits (3..0)</b>  | l    | This input indicates the number of valid PDH payload bits in this clock cycle. It ranges between 0 (0000) and 8 (1000).  |
| <b>uti_pdh_ohid (1..0)</b>  | l    | The format of this input is currently undocumented.  |
| <b>uti_pdh_stuff (1..0)</b> | l    | This input indicates the state of PDH stuffing for the VC indicated by <b>uti_lovc_id</b> . It is defined as...<br>00 : No stuffing<br>01 : Positive stuffing – resulting PDH rate is marginally slower<br>10 : Negative stuffing – resulting PDH rate is marginally faster  |
| <b>uti_data (7..0)</b>      | l    | This is the frame data input associated with the above timing inputs.<br><br>If PDH mapping is enabled (ie: <b>uti_pdh_vld</b> is asserted) then this is treated as a variable width input – the number of valid bits is indicated by the <b>uti_pdh_bits</b> input, and this number of bits is taken from the most significant bits of this byte (ie: if <b>uti_pdh_bits</b> is 2 (0010) then bits (7..6) from this byte are transmitted.<br><br>If PDH mapping is not enabled, then this input functions as a regular byte-wide payload input. |

### 3.5. Status and Configuration Interface

#### 3.5.1. Timing Generation TUG3/TUG2/TU Mapping

This interface configures how TUG3s, TUG2s and TUs are generated for the output VCs.

| Name                           | Type | Description   |
|--------------------------------|------|---|
| <b>tu_vc_id<br/>(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 generated for output VCs or not (set high to insert 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<br/>(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 generated for output VCs/TUG3s or not (set high to insert 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<br/>(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<br/>(1..0)</b>   | I    | This input configures what type of TU is inserted into 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...<br>00 : TU11<br>01 : TU12<br>10 : TU2<br>11 : TU3 |

### 3.5.2. Timing Generation TU Pointer Control

This interface gives external logic control over TU pointer generation, and may be used to insert pointer movements and new pointer values.

| Name                    | Type | Description   |
|-------------------------|------|---|
| <b>tuptr_id (8..0)</b>  | 0    | Indicates which TU the following outputs are for. Refer to section 4.2 for a description of the structure of this output.   |
| <b>tuptr_vld</b>        | 0    | Indicates that a the opportunity exists to influence the TU pointer for the TU indicated by <b>tuptr_id</b> . External logic should respond by driving the following inputs in the following clock cycle.                                       |
| <b>tuptr_inc</b>        | 1    | When asserted high, in the clock cycle following <b>tuptr_vld</b> , a TU pointer increment is performed for the TU indicated by <b>tuptr_id</b> in the preceding clock cycle.   |
| <b>tuptr_dec</b>        | 1    | When asserted high, in the clock cycle following <b>tuptr_vld</b> , a TU pointer decrement is performed for the TU indicated by <b>tuptr_id</b> in the preceding clock cycle.   |
| <b>tuptr_ndf</b>        | 1    | When asserted high, in the clock cycle following <b>tuptr_vld</b> , a new TU pointer value is used for the TU indicated by <b>tuptr_id</b> in the preceding clock cycle. An NDF (New Data Flag) event is indicated in the TU pointer bytes.     |
| <b>tuptr_new</b>        | 1    | When asserted high, in the clock cycle following <b>tuptr_vld</b> , a new TU pointer value is used for the TU indicated by <b>tuptr_id</b> in the preceding clock cycle. An NDF (New Data Flag) event is not indicated in the TU pointer bytes. |
| <b>tuptr_val (9..0)</b> | 1    | The pointer value used when a new pointer value is inserted using <b>tuptr_ndf</b> and <b>tuptr_new</b> . This input should be valid at the same time as these signals.   |

### 3.5.3. Timing Generation Lower Order VC Configuration

This interface configures how Lower Order VCs are inserted into the output TUs.

| Name                         | Type | Description   |
|------------------------------|------|---|
| <b>lvc_cfg_id<br/>(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 inserted into output 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. |

### 3.5.4. Timing Generation PDH Mapping Configuration

This interface provides PDH mapping configuration access, and stuff rate control.

| Name                          | Type | Description  |
|-------------------------------|------|--|
| <b>pdh_cfg_id</b><br>(8..0)   | O    | This output indicates what VC/TU is requesting PDH mapping configuration information. Refer to section 4.3 for a description of the structure of this output.  |
| <b>pdh_type</b><br>(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.<br/>           001 : Asynchronous Mapping (type 1)<br/>           010 : Asynchronous Mapping (type 2)<br/>           011 : Bit synchronous<br/>           100 : Byte synchronous (type 1)<br/>           101 : Byte synchronous (type 2)</p> <p>See section 4.3 for further details.</p> |
| <b>pdh_stf_vld</b>            | O    | This output indicates the opportunity to influence PDH stuff rate, for the TU indicated by <b>pdh_cfg_id</b> . The <b>pdh_stf_type</b> input should be driven in the following clock cycle in response.  |
| <b>pdh_stf_type</b><br>(1..0) | I    | <p>This input is encoded as...</p> <p>00 : No stuffing<br/>           01 : Positive stuffing – resulting PDH rate is marginally slower<br/>           10 : Negative stuffing – resulting PDH rate is marginally faster<br/>           11 : Automatic. The core generates the correct rate.</p> <p>Note that for DS3 over VC3 and E4 over VC4 negative stuffing is not valid – only positive stuffing or no stuffing is allowed.</p>  |

### 3.5.5. Lower Order Path Overhead

The Lower Order Path Overhead interface gives external logic access to Lower Order Path Overhead, allowing it to insert overhead data and errors.

| Name                     | Type | Description   |
|--------------------------|------|---|
| <b>lpoh_vld</b>          | O    | When asserted this indicates that the following Lower Order Path Overhead outputs are valid.  |
| <b>lpoh_vc_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_id (3..0)</b>    | O    | <p>This output indicates what type of Lower Order Path Overhead is available for manipulation by external logic. For TU11, TU12 and TU2 defined values are.</p> <p>0000 : V5<br/>           0001 : J2<br/>           0010 : N2<br/>           0011 : K4</p> <p>If the TU is a TU3 then it is encoded as...</p> <p>0000 : J1<br/>           0001 : B3<br/>           0010 : C2<br/>           0011 : G1<br/>           0100 : F2<br/>           0101 : H4<br/>           0110 : F3<br/>           0111 : K3<br/>           1000 : N1</p> |
| <b>lpoh_ins_bip</b>      | I    | When asserted the core inserts its internally generated BIP-2 value (B3 in the case of Lower Order VC3) into the Lower Order Path Overhead.   |
| <b>lpoh_ins</b>          | I    | When asserted (in response to <b>lpoh_vld</b> ) the <b>lpoh_data</b> and <b>lpoh_mask</b> inputs are used for Lower Order Path Overhead injection   |
| <b>lpoh_data (7..0)</b>  | I    | This data byte is qualified with <b>lpoh_mask</b> and transmitted as LOPOH overhead if <b>lpoh_ins</b> is asserted.   |
| <b>lpoh_mask (7..0)</b>  | I    | This mask is qualified with <b>lpoh_data</b> for LOPOH insertion. A bit set to 1 causes the corresponding bit in <b>lpoh_data</b> to be sent as overhead, otherwise the data supplied at the Data Input Interface is used.  |
| <b>lpoh_err</b>          | I    | When asserted (in response to <b>lpoh_vld</b> ) the <b>lpoh_err_mask</b> and <b>lpoh_err_val</b> inputs are used inject errors into the LOPOH.  |

| Name                                   | Type | Description  |
|--|------|--|
| <b>lpoh_err_mask</b><br><b>(15..0)</b> | l    | This input is used to error LOPOH bytes (after they have been inserted via the above inputs). Every pair of bits corresponds to a bit of the RSOH byte ((1..0) to bit 0, (3..2) to bit 1, etc), and is encoded as...<br>00 : No change<br>01 : Invert LOPOH bit<br>10 : Replace with associated bit from <b>lpoh_err_val</b> .<br>11 : Undefined |
| <b>lpoh_err_val</b><br><b>(7..0)</b>   | l    | If the <b>lpoh_err_mask</b> input overwrites bits in the LOPOH overhead, the bits used are taken from this input.  |

## 4. Implementation Details

### 4.1. Resource Utilisation

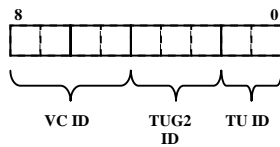
The following figures are calculated assuming that all core IOs are routed off-chip. This results in a worst-case resource utilisation figure, and for any given application the resource utilisation is likely to be lower. The example parts are the slowest (and hence cheapest) offered speed-grade, and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

|                             | Stratix Family<br>Eg : EP1S10F484C7 |                 |                 | Stratix GX Family<br>Eg : EP1SGX10CF672C7 |                 |                 | Cyclone Family<br>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> | 1952                                | 10570           | 18.5%           | 1952                                      | 10570           | 18.5%           | 2140                                | 12060           | 18%             |
| <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                               |                 |                 |

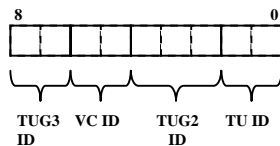
|                        | Stratix2 Family<br>Eg : EP2S15F484C5 |                 |                 |
|------------------------|--------------------------------------|-----------------|-----------------|
|                        | Used by Core                         | In example Part | Percentage used |
| <b>ALMs</b>            | 1114                                 | 6240            | 18%             |
| <b>M512 RAM Blocks</b> | 8                                    | 104             | 8%              |
| <b>M4k RAM Blocks</b>  | 12                                   | 78              | 15%             |
| <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 into 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/>