

**SONET/SDH Deframer Core
STM0/OC1, STM1/OC3, STM4/OC12**

Single and Multi-Channel

Revision History

Version	Date	Details
Version 1.4	12/8/03	Creation of Document, based on existing Sonic STM1/4 Deframer documentation

Contents

1. Features	4
2. Functional Description	4
2.1. Frame Alignment	4
2.2. Frame Synchronisation	4
2.3. Regenerator Section Overhead	5
2.4. Multiplex Section Overhead	5
2.5. Pointer Processing	5
2.6. VC Processing	5
2.7. Higher Order Path Overhead	5
3. Signal Description	6
3.1. Data Input Interface	6
3.2. Data Output Interface	7
3.3. Status and Configuration Interface	8
3.3.1. Frame Synchronisation	8
3.3.2. Regenerator Section Overhead	9
3.3.3. Multiplex Section Overhead	10
3.3.4. AU Pointers	11
3.3.5. VC Configuration	12
3.3.6. Higher Order Path Overhead	13
4. Implementation Details	14
4.1. Resource Utilisation	14
4.2. Connection to Data Source	15
4.2.1. Data sourced by CDR	15
4.2.2. Data sourced by CDR (no frame alignment)	15
4.2.3. Data sourced by internal logic	15

1. Features

The Aliathon STM0/1/4 Deframer core provides a flexible, resource-efficient, programmable-logic based solution for SDH interfacing. It caters for both concatenated payloads, such as VC4-4c over STM4, and channelised applications, such as multiple VC3s over STM1. The core...

- Detects and aligns to the SDH framing pattern.
- Performs frame synchronisation for STM0, STM1 and STM4, and generates LOS, OOF and LOF alarms. The core may be dynamically switched between SDH rates.
- Descrambles the SDH frame, extracts Regenerator Section Overhead, and detects B1 errors.
- Extracts Multiplex Section Overhead and detects B2 errors.
- Processes all AU pointers.
- Extracts VC3, VC4 and VC4-4c, both channelised and single-channel. All legal combinations of VCs are supported. The VC settings may be dynamically reconfigured, and made on a per VC basis.
- Detects B3 errors for all VCs and extracts Higher Order Path Overhead.

2. Functional Description

Figure 2 illustrates the major functional blocks within the SDH Deframer Core.

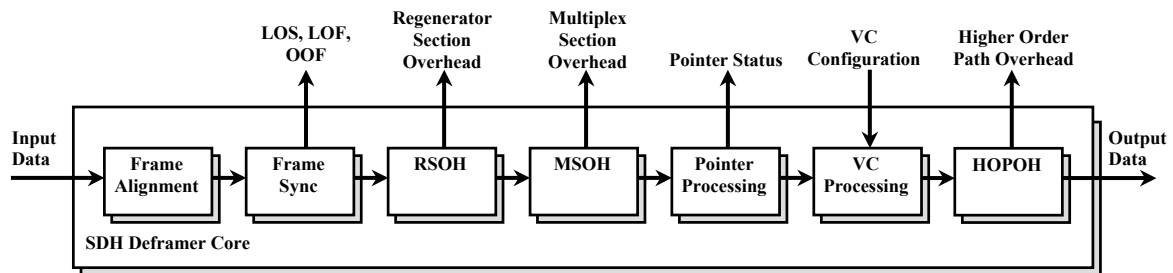


Figure 2 – SDH Deframer Core Functional Blocks

2.1. Frame Alignment

The Frame alignment block detects the SDH framing pattern at any bit offset within the input data stream, and aligns to it. Some STM1/STM4 LIU devices provide this functionality, and thus the frame alignment may optionally be disabled to lower resource utilisation.

2.2. Frame Synchronisation

The core performs frame synchronisation as per ITU-T G.707. It generates Loss-of-Signal (LOS), Loss-of-Frame (LOF) and Out-of-Frame (OOF) alarms.

2.3. Regenerator Section Overhead

The RSOH block descrambles the incoming SDH frame (descrambling may be enabled or disabled). It verifies the incoming B1 value and indicates the presence and number of any B1 errors. It also extracts the Regenerator Section Overhead and outputs it from the core.

2.4. Multiplex Section Overhead

The Multiplex Section Overhead block verifies the incoming B2 value and indicates the presence and number of any B2 errors. It also extracts the Multiplex Section Overhead and outputs it from the core.

2.5. Pointer Processing

The Pointer Processing block processes all the incoming AU pointers (up to 12 for VC3 over STM4), and determines the state of each – Loss-of-Pointer (LOP), Alarm-Indication-Signal (AIS), Concatenation Indication (CI) or Normal.

2.6. VC Processing

The VC (Virtual Container) block extracts all legal combinations of VCs. In “simple” configuration mode the core extracts one or more of the same VC type from the SDH frame (for example, 1 VC4 over STM1, or 12 VC3s over STM4). In “advanced” configuration mode a different VC type can be specified for each potential VC location, allowing for mixes of VC sizes to be extracted. Figure 2 illustrates the SDH multiplex structure and shows how different VC sizes may be combined. Over STM0 VC3 is the only possibility. An STM1 may carry 1 VC4 or 3 VC3s. For STM4 the possibilities are more complex – it may carry 1 VC4-4c, or 4 AUG4s (which can each consist of 3 VC3s or 1 VC4), and thus combinations such as 2 VC4s and 6 VC3s, or 3 VC4s and 3 VC3s are possible. The core can extract all such combinations.

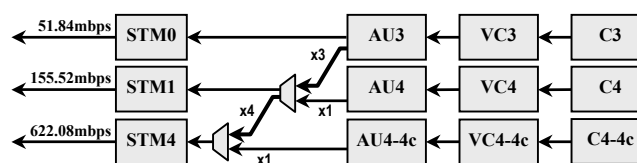


Figure 2 – SDH Multiplex Structure

2.7. Higher Order Path Overhead

The HOPOH block extracts Higher Order Path Overhead for all the configured VCs and passes it out on a specific HOPOH bus. It also calculates and verifies the B3 value for all VCs and indicates the presence and number of B3 errors.

3. Signal Description

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

- Data input
- Processed data output
- Status and configuration

3.1. Data Input Interface

This interface provides the input clock and data to the deframer core. Typically an external CDR device would drive these signals, but the core also provides internal frame alignment and a generic chip-enable input that allows other data sources to drive the core.

Name	Type	Description
reset	I	Asynchronous reset input.
sdh_clk	I	sdh_clk is typically supplied by the external LIU. All inputs and outputs from the core are synchronous to this clock unless otherwise noted.
sdh_data (7..0)	I	sdh_data is the byte wide SDH data input. When sdh_vld is asserted it carries VC data for the VC indicated by vc_id . Note that bit 7 is the most significant (first received).
sdh_vld	I	When asserted this indicates that there is valid data on sdh_data . If an external CDR device is used then this input would typically be tied high, as CDR devices provide data on every clock cycle. This input is useful for driving the core with data from another clock domain, via an asynchronous fifo, for example.
sdh_fp	I	When asserted this indicates that the byte on sdh_data is the third A2 byte location. This is typically used with an external CDR device that provides frame-alignment functionality (such as Vitesse VSC8117 and AMCC S3032). This input is only relevant when sdh_align is deasserted.
sdh_oof	O	When an external CDR device is used to provide frame-alignment functionality, this output indicates that the core is "out-of-frame" and thus the CDR should attempt to align to an SDH framing pattern. If the external CDR does not provide frame alignment, or data is supplied by some other means, then this output should be left unconnected.
sdh_align	I	This input configures the core to perform SDH framing alignment, and should typically be hard-wired high or low. When low, the core assumes that the input data on pyld_data is already byte-aligned and that the pyld_fp input will indicate the frame position. When high, the core performs frame detection and alignment itself.

3.2. Data Output Interface

This interface provides the data output from the deframer core. Each output byte is associated with a particular VC and identified as being VC payload or overhead.

Name	Type	Description
frm_vld	○	When asserted this indicates that the following outputs are valid. If the sdh_vld signal on the Data Input Interface is tied high then this signal will always be asserted.
frm_sof	○	When asserted this indicates that the byte on the frm_data output is the first A1 byte of the SDH frame.
frm_vc_vld	○	The Data Output Interface carries the complete SDH frame, including all overhead and payload data. This signal, when asserted, indicates that the data byte on the frm_data output belongs to a particular VC (indicated by the frm_vc_id output) and not to SDH frame overhead.
frm_pyld_vld	○	This output is similar to frm_vc_vld but it does not include VC path overhead and stuffing, and only indicates VC payload data. This output is only ever active when frm_vc_vld is asserted.
frm_j1	○	This output indicates that the data byte on the frm_data output is the J1 byte for the VC indicated by frm_vc_id . It is only ever asserted when frm_vc_vld is also asserted.
frm_data (7..0)	○	This output carries the byte aligned SDH frame data. Note that bit 7 is the most significant (first received).
frm_vc_id (3..0)	○	This output identifies which VC the byte on frm_data belongs to. It is only valid when frm_vc_vld is asserted. The VC configuration of the core (see VC Configuration Interface) affects the behaviour of this output. If the core is set to STM4 and the VC type to VC4-4c then there is only one VC, and thus this output will always be 0000. If the VC type was set to VC3, then there are 12 VCs and this output would range from 0 (0000) to 11 (1011).
frm_vc_size (1..0)	○	This outputs indicates the type of VC that the byte on frm_data is from. Defined values are... 00 : VC3 01 : VC4 10 : VC4-4c

3.3. Status and Configuration Interface

This interface allows the core to be dynamically configured, and provides status outputs. To lower resource utilisation, any unused outputs should be left open, and configuration inputs should be hard-wired to the required value if they do not need to change. The following configuration and status signals are grouped by function.

3.3.1. Frame Synchronisation

Name	Type	Description
fs_rate (1..0)	I	This configuration inputs sets the SDH rate of the core. Defined values are... 00 : STM0 01 : STM1 10 : STM4
fs_oof	O	This output indicates that the SDH Deframer core is not synchronised to the SDH frame structure (out-of-frame).
fs_lof	O	When asserted this output indicates that the fs_oof state has persisted for 3ms or more (loss-of-frame).
fs_los	O	When asserted this output indicates that the input data stream has been all 0s for 20µs or more (loss-of-signal).

3.3.2. Regenerator Section Overhead

Name	Type	Description
rsoh_scram_off	I	When this input is driven high SDH descrambling is disabled. It should typically be driven low to enable descrambling.
rsoh_b1_err	O	When asserted high this output indicates the occurrence of a B1 error.
rsoh_num_b1 (3..0)	O	This output indicates the number of B1 errors. When rsoh_b1_err is asserted this output will range from 1 (0001) to 8 (1000).
rsoh_vld	O	When this output is asserted the following Regenerator Section Overhead outputs are valid.
rsoh_out (7..0)	O	This output carries the byte wide Regenerator Section Overhead.
rsoh_id (3..0)	O	This output identifies what kind of RSOH byte is on rsoh_out . Defined values are... 0000 : A1 0001 : A2 0010 : J0 0011 : B1 0100 : E1 0101 : F1 0110 : D1 0111 : D2 1000 : D3
rsoh_col (3..0)	O	For STM1 and STM4 there is more than one byte for each RSOH field (although typically only the first is of interest). This output indicates which "column" the byte on rsoh_out is from. For STM1 this output ranges from 0 (0000) to 2 (0010), and for STM4 it ranges from 0 (0000) to 11 (1011).
rsoh_first_col	O	This output indicates that rsoh_col is 0. As most applications only use the first RSOH column, rsoh_vld may be left unconnected and this output used to qualify RSOH data.

3.3.3. Multiplex Section Overhead

Name	Type	Description
msoh_b2_err	○	When asserted high this output indicates the occurrence of a B2 error.
msoh_num_b2 (3..0)	○	This output indicates the number of B2 errors. When msoh_b2_err is asserted this output will range from 1 (0001) to 8 (1000).
msoh_vld	○	When this output is asserted the following Multiplex Section Overhead outputs are valid.
msoh_out (7..0)	○	This output carries the byte wide Multiplex Section Overhead.
msoh_id (3..0)	○	This output identifies what kind of MSOH byte is on msoh_out . Defined values are... 0000 : B2 0001 : K1 0010 : K2 0011 : D4 0100 : D5 0101 : D6 0110 : D7 0111 : D8 1000 : D9 1001 : D10 1010 : D11 1011 : D12 1100 : S1 1101 : M1 1110 : E2
msoh_col (3..0)	○	For STM1 and STM4 there is more than one byte for each MSOH field (although typically only the first is of interest). This output indicates which "column" the byte on msoh_out is from. For STM1 this output ranges from 0 (0000) to 2 (0010), and for STM4 it ranges from 0 (0000) to 11 (1011).
msoh_first_col	○	This output indicates that msoh_col is 0. As most applications only use the first MSOH column, msoh_vld may be left unconnected and this output used to qualify MSOH data.

3.3.4. AU Pointers

Name	Type	Description
auptr_id (3..0)	0	This output indicates which AU pointer the following status outputs are for. It will always be 0 (0000) for STM0, range from 0 (0000) to 2 (0010) for STM1, and range from 0 (0000) to 11 (1011) for STM4.
auptr_state (1..0)	0	This indicates the state of the pointer processing state machine, for the particular pointer location indicated by auptr_id . Defined values are... 00 : LOP – Loss of Pointer 01 : AIS – Alarm Indication Signal 10 : Normal 11 : Concatenation Indication
auptr_val (9..0)	0	This output carries the currently accepted pointer value for the AU indicated by auptr_id . It ranges between 0 (0000000000) and 782 (1100001110).
auptr_inc	0	When asserted this indicates a pointer increment, for the AC indicated by auptr_id .
auptr_dec	0	When asserted this indicates a pointer decrement, for the AC indicated by auptr_id .
auptr_ndf	0	When asserted this indicates that a new pointer value has been accepted as a result of an NDF being received, for the AC indicated by auptr_id .
auptr_new	0	When asserted this indicates that a new pointer value has been accepted as a result of 3 consecutive, identical pointers, for the AC indicated by auptr_id .

3.3.5. VC Configuration

Name	Type	Description																
vcfg_global	I	<p>When this input is asserted the core extracts the VC type as indicated by the vcfg_size input. Thus, given the SDH rate, the core will process the following number of VCs...</p> <table border="1" data-bbox="536 568 1436 714"> <thead> <tr> <th></th> <th>VC3</th> <th>VC4</th> <th>VC4-4c</th> </tr> </thead> <tbody> <tr> <td>STM0</td> <td>1</td> <td>3</td> <td>12</td> </tr> <tr> <td>STM1</td> <td></td> <td>1</td> <td>4</td> </tr> <tr> <td>STM4</td> <td></td> <td></td> <td>1</td> </tr> </tbody> </table> <p>When this input is not asserted the core is configured on a per-VC basis by the vcfg_vld and vcfg_loc inputs. This allows any legal mix of VC sizes to be processed (for example, 1 VC4 and 9 VC3s over STM4).</p>		VC3	VC4	VC4-4c	STM0	1	3	12	STM1		1	4	STM4			1
	VC3	VC4	VC4-4c															
STM0	1	3	12															
STM1		1	4															
STM4			1															
vcfg_size (1..0)	I	<p>This indicates the type of VC to extract. When vcfg_global is asserted this is applied uniformly across the SDH frame. Otherwise it is applied to the VC location indicated by vcfg_loc when vcfg_vld is asserted. Defined values are...</p> <p>00 : VC3 01 : VC4 10 : VC4-4c</p>																
vcfg_clk	I	<p>The following configuration inputs should be synchronous to this clock. This allows the VC configuration interface to be driven directly by a microprocessor interface. This clock may be the same as sdh_clk.</p>																
vcfg_vld	I	<p>When asserted the value on the vcfg_size input is applied to the VC indicated by vcfg_loc. If vcfg_global is tied high then this input should be tied low.</p>																
vcfg_loc (3..0)	I	<p>This indicates the VC to apply the configuration on vcfg_size to when vcfg_vld is asserted. It corresponds to H1 columns in the SDH frame (1 for STM0, 3 for STM1, 12 for STM4) and VC interleaving must be considered to determine the correct value.</p>																

3.3.6. Higher Order Path Overhead

Name	Type	Description
hpoh_vc_id (3..0)	○	As the core can process up to 12 different VCs, there are up to 12 different Higher Order Path Overheads. This output indicates which of the VCs the following outputs are for. If the core is processing a single VC (VC4 over STM1 for example) then this output will always be 0 (0000). If it is processing 12 VCs (VC3 over STM4) then it range between 0 (0000) and 11 (1011).
hpoh_b3_err	○	When asserted high this output indicates the occurrence of a B3 error, for the VC indicated by hpoh_vc_id .
hpoh_num_b3 (3..0)	○	This output indicates the number of B3 errors. When hpoh_b3_err is asserted this output will range from 1 (0001) to 8 (1000).
hpoh_vld	○	When this output is asserted the following Higher Order Path Overhead outputs are valid, for the VC indicated by hpoh_vc_id .
hpoh_out (7..0)	○	This output carries the byte wide Higher Order Path Overhead.
hpoh_id (3..0)	○	This output identifies what kind of HPOH byte is on hpoh_out . Defined values are... 0000 : J1 0001 : B3 0010 : C2 0011 : G1 0100 : F2 0101 : H4 0110 : F3 0111 : K3 1000 : N1

4. Implementation Details

4.1. Resource Utilisation

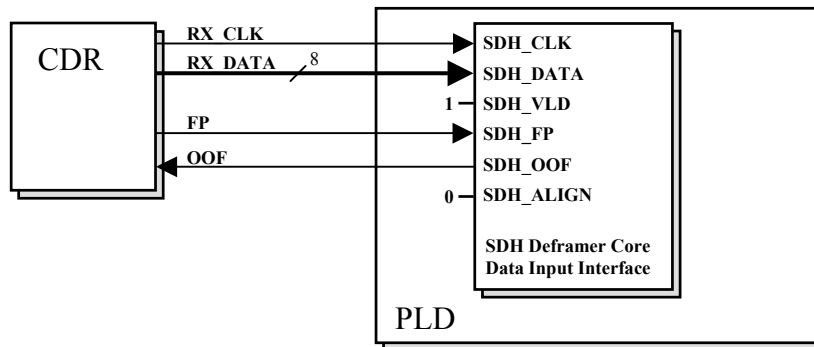
The following figures are calculated assuming that all core IOs are routed off-chip. This results in a worst-case resource utilisation figure, and for any given application the resource utilisation is likely to be lower. The example parts are the slowest (and hence cheapest) offered speed-grade, and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

	Stratix Family Eg : EP1S10F484C7			Stratix GX Family Eg : EP1SGX10CF672C7			Cyclone Family Eg : EP1C6F256C8		
	Used by Core	In example Part	Percentage used	Used by Core	In example Part	Percentage used	Used by Core	In example Part	Percentage used
Logic Elements (LEs)	1216	10570	11.5%	1216	10570	11.5%	1216	5980	20%
M512 RAM Blocks	7	94	7%	7	94	7%		n/a	
M4k RAM Blocks	1	60	1%	1	60	1%	8	20	40%
M-RAM Blocks	0	1	0%	0	1	0%		n/a	
Fmax	100MHz			100MHz			90MHz		

4.2. Connection to Data Source

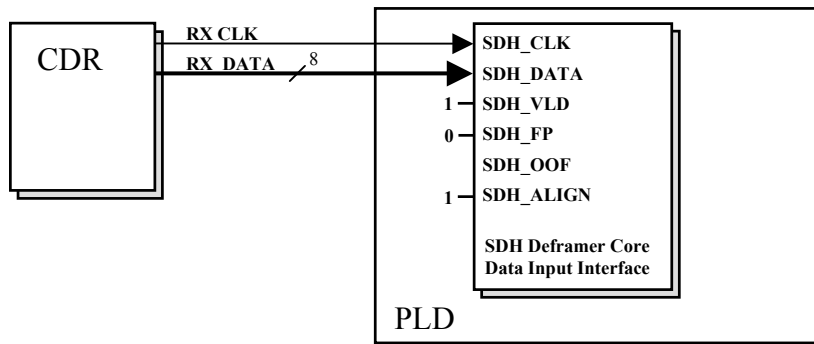
4.2.1. Data sourced by CDR

The following diagram illustrates how to drive the cores data input interface when an external CDR device, providing frame alignment functionality, is used.



4.2.2. Data sourced by CDR (no frame alignment)

The following diagram illustrates how to drive the cores data input interface when an external CDR device, which does not provide frame alignment functionality, is used.



4.2.3. Data sourced by internal logic

The following diagram illustrates how to drive the core from internal logic.

