****

**Open ROADM MSA 5.0 W-Port Digital Specification**

**(100G-400G)**

**Open ROADM-Draft document**

***July 01, 2021***

www.Open ROADM.org

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**THIS SPECIFICATION IS PROVIDED "AS IS" WITH NO WARRANTIES WHATSOEVER, INCLUDING ANY WARRANTY OF MERCHANTABILITY, NONINFRINGEMENT, FITNESS FOR ANY PARTICULAR PURPOSE, OR ANY WARRANTY OTHERWISE ARISING OUT OF ANY PROPOSAL, SPECIFICATION OR SAMPLE. THE AUTHORS DISCLAIM ALL LIABILITY, INCLUDING LIABILITY FOR INFRINGEMENT OF ANY PROPRIETARY RIGHTS, RELATING TO USE OF INFORMATION IN THIS SPECIFICATION. EACH MEMBER OF THE OPEN ROADM MSA AGREE TO GRANT NON-EXCLUSIVE, FAIR, REASONABLE AND NON-DISCRIMINITORY (FRAND) PATENT LICENSES TO MEMBES OF THE MSA TO ANY CLAIM IT OWNS OR HAS THE ABILTY TO LICENSE THAT ARE STANDAS ESSENTIAL (I.E., MANDATORY, NECESSARY, OR REQUIRED) TO MAKE AND USE A PRODUCT COMPLIANT WITH THESE SPECIFICATIONS UNDER THE FOLLOWING REQUIRED CONDITIONS:**

1. **The patent license only covers preexisting patent claims that are Standards Essential (i.e., mandatory, necessary or required) to make and use a product compliant with THESE Specifications.  The patent license does not include claims directed to Non-Standards Essential design or implementation choices; and**
2. **Any person or entity that intends to claim the benefit of this patent license must: (1) identify in writing which Specification(s) apply to its product(s) before said products are sold; and (2) make a reciprocal commitment in writing to license its own patent claims that are Standards Essential (i.e., mandatory, necessary or required) to make and use a product compliant with theSE Specifications under FRAND licensing terms back to MEMBERS OF THE OPEN ROADM MSA.**

**TITLE: Open ROADM MSA 5.0 W-Port Digital Specification (100G-400G)**

**SOURCE:**

Mike A. Sluyski

Cisco Systems.

3 Mill and Main

Maynard, MA 01754, USA

Phone: +1-978-938-4896 x773

Email: [msluyski@cisco.com](mailto:msluyski@cisco.com)

**ABSTRACT:**  This document describes the Open ROADM W-Port Digital Specification for 100G, 200G, 300G, and 400G transport over coherent DP-QPSK and DP-mQAM optical links. The specification includes the processing functions of adapting a fully standardized FlexO-*x* [7] frame structure with a new open Forward Error Correction (oFEC) encoder and symbol frame format.

The oFEC engine is a block-based encoder and iterative Soft-Decision (SD) decoder. With 3 SD iterations the Net Coding Gain is 11.1 dB @ 10-15 (DP-QPSK) and 11.6 dB @ 10-15 (DP-16QAM), with pre-FEC BER threshold of 2.0×10-2. The combined latency of the encoder and decoder is less than 3 µs.

An Open ROADM transponder/muxponder node may support up to 4 × 100G Client interfaces. The mapping and aggregation of these Client interfaces to a FlexO-x (x=1,2,3,4) frame structure is fully specified by ITU-T G.709/Y.1331 and G.709.1/Y.1331.1. The adaptation functions from the FlexO-x frame structure into the oFEC block and the line-side DSP symbol framing is described in this document and is fully aligned to ITU-T G.709.3/Y.1331.1.

[1 List of Figures 6](#_Toc76028545)

[2 List of Tables 7](#_Toc76028546)

[3 Document Revision History 8](#_Toc76028547)

[4 Definitions 9](#_Toc76028548)

[4.1 Terms defined in this document 9](#_Toc76028549)

[4.2 Abbreviations and Acronyms 9](#_Toc76028550)

[4.3 Conventions 10](#_Toc76028551)

[5 References 11](#_Toc76028552)

[6 Scope and Introduction 12](#_Toc76028553)

[7 FlexO-x frame structure 15](#_Toc76028554)

[7.1 FlexO-1 Frame structure 15](#_Toc76028555)

[7.2 FlexO-2 Frame Structure 17](#_Toc76028556)

[7.3 FlexO-3 Frame Structure 18](#_Toc76028557)

[7.4 FlexO-4 Frame Structure 19](#_Toc76028558)

[8 FlexO-x-DO adaptation 21](#_Toc76028559)

[8.1 Padding Insertion/Removal 22](#_Toc76028560)

[8.2 FlexO-4 adaptation to FOIC4.8-DO 23](#_Toc76028562)

[8.3 FlexO-3 adaptation to FOIC3.6-DO 24](#_Toc76028566)

[8.4 FlexO-2 adaptation to FOIC2.4-DO 25](#_Toc76028571)

[8.5 FlexO-2 adaptation to FOIC2.8-DO 26](#_Toc76028584)

[8.6 FlexO-1 adaptation to FOIC1.4-DO 27](#_Toc76028588)

[8.7 Frame Synchronous Scrambling 28](#_Toc76028589)

[9 Open Forward Error Correction (oFEC) 29](#_Toc76028590)

[9.1 oFEC encoding codec 29](#_Toc76028591)

[9.2 Encoding 33](#_Toc76028592)

[9.2.1 Encoder interface 33](#_Toc76028593)

[9.2.2 Formal encoder definition 35](#_Toc76028594)

[9.3 Decoding 36](#_Toc76028595)

[9.4 oFEC Interleaver 37](#_Toc76028596)

[9.4.1 oFEC Interleaver architecture 37](#_Toc76028597)

[9.4.2 Intra-block interleaving 37](#_Toc76028598)

[9.4.3 Inter-block interleaving 38](#_Toc76028599)

[10 Symbol Mapping and Polarization Distribution 42](#_Toc76028600)

[10.1 Symbol mapping 42](#_Toc76028601)

[10.1.1 DP-16QAM Symbols 42](#_Toc76028602)

[10.1.2 DP-8QAM Symbols 43](#_Toc76028603)

[10.1.3 DP-QPSK Symbols 44](#_Toc76028604)

[11 DSP Framing 45](#_Toc76028605)

[11.1 DSP super-frame 45](#_Toc76028606)

[11.2 DSP sub-frame 46](#_Toc76028607)

[11.3 FAW Sequence 48](#_Toc76028608)

[11.4 Training Sequence 49](#_Toc76028609)

[11.5 Pilot Sequence 50](#_Toc76028610)

[12 Frame Expansion Rate 55](#_Toc76028611)

[13 Appendix A – TRPN/MUXP/SWITCH client interface examples 56](#_Toc76028612)

[13.1 Transponder (TRPN) 57](#_Toc76028613)

[13.2 Muxponder (MUXP) 57](#_Toc76028614)

[13.3 Client interface bit rates 58](#_Toc76028615)

# List of Figures

[Figure 1: Open ROADM Architecture reference 12](#_Toc73018433)

[Figure 2: FlexO-x-DO-*m* functional model 14](#_Toc73018434)

[Figure 3: Digital port functions for a single FlexO-x-DO signal 14](#_Toc73018435)

[Figure 4: FlexO-1 frame structure 15](#_Toc73018436)

[Figure 5: FlexO multi-frame format 16](#_Toc73018437)

[Figure 6: FlexO-2 frame structure 17](#_Toc73018438)

[Figure 7: interleaved FlexO frame to FlexO-2 frame structure 18](#_Toc73018439)

[Figure 8: FlexO-3 frame structure 18](#_Toc73018440)

[Figure 9: Interleaved FlexO frame to FlexO-3 frame structure 19](#_Toc73018441)

[Figure 10: FlexO-4 frame structure 20](#_Toc73018442)

[Figure 11: Interleaved FlexO frame to FlexO-4 frame structure 20](#_Toc73018443)

[Figure 12: n x FlexO to FlexO-x-DO adaption process 21](#_Toc73018444)

[Figure 13: Digital processes of FlexO-x to FlexO-x-DO adaptation. 22](#_Toc73018445)

[Figure 14: FlexO-4 adaptation to an FOIC4.8-D0 line interface 23](#_Toc73018446)

[Figure 15: FlexO-3 adaptation to an FOIC3.6-D0 line interface 24](#_Toc73018447)

[Figure 16: FlexO-2 adaptation to an FOIC2.4-D0 line interface 25](#_Toc73018448)

[Figure 17: FlexO-2 adaptation to an FOIC2.6-D0 line interface 26](#_Toc73018449)

[Figure 18: FlexO-2 adaptation to an FOIC2.8-D0 line interface 27](#_Toc73018450)

[Figure 18: FlexO-1 to oFEC input block adaptation 28](#_Toc73018451)

[Figure 19: Frame synchronous scrambler 29](#_Toc73018452)

[Figure 20: oFEC block encoder and oFEC Interleaver 30](#_Toc73018453)

[Figure 21: Structure of an openFEC Coder 32](#_Toc73018454)

[Figure 22: Sequencing of bits within an input block 35](#_Toc73018455)

[Figure 23: Bit numbering within an output block 36](#_Toc73018456)

[Figure 24: Inter-block interleaving 41](#_Toc73018457)

[Figure 25: DSP symbol mapping and polarization distribution 43](#_Toc73018458)

[Figure 26: DSP Frame generation 46](#_Toc73018459)

[Figure 27: DSP super-frame 47](#_Toc73018460)

[Figure 28: DSP sub-frames 1 to 47 of the DSP super-frame 48](#_Toc73018461)

[Figure 29: Pilot Symbol (DP-16QAM modulation shown) 51](#_Toc73018462)

[Figure 30: Pilot Seed and Sequencing 51](#_Toc73018463)

# List of Tables

[Table 1: Revision History 8](#_Toc73018464)

[Table 2: oFEC adaptation rates 21](#_Toc73018465)

[Table 3: Source positions (row, col) for intra-block interleaving 39](#_Toc73018466)

[Table 4: Interleaver subsets 40](#_Toc73018467)

[Table 5: DP-16QAM symbol amplitude map 44](#_Toc73018468)

[Table 6: 8QAM symbol amplitude map 45](#_Toc73018469)

[Table 7: DP-QPSK symbol amplitude map 45](#_Toc73018470)

[Table 8: FAW/TS/PS pattern 46](#_Toc73018471)

[Table 9: FAW Sequence 49](#_Toc73018472)

[Table 10: Training symbol sequence 50](#_Toc73018473)

[Table 11: Pilot Sequence 51](#_Toc73018474)

[Table 12: Pilot Sequence 55](#_Toc73018475)

[Table 13: FlexO/oFEC expansion rates 56](#_Toc73018476)

[Table 14: Open ROADM 3.0 Transponder (TRPN) functions 58](#_Toc73018477)

[Table 15: Open ROADM 3.0 Muxponder (MUXP) functions 58](#_Toc73018478)

[Table 16 Client interface types and payload bit rates 59](#_Toc73018479)

[Table 17: Line types and frame bit rates 59](#_Toc73018480)

# Document Revision History

Table 1 provides this document revision history.

|  |  |  |
| --- | --- | --- |
| **Document Revision** | **Date** | **Revision Comments** |
| 3.0 | 9/17/18 | Initial Draft – documents client mapping and multiplexing into FlexO container, DSP frame format, and openFEC encoding/decoding functions. |
| 3.01 | 06/06/19 | Maintenance update. Incorporated reviewer’s comments including proposals for distinguishing the essential port function (which has a node-internal digital service interface and a node-external optical transmission interface) and an Open ROADM node specification. |
| 5.0 | 2/26/2020  05/27/21  06/25/21 | * Updated FlexO-x-<fec> naming conventions in Table 14 and Table 15 to align with ITU-T G.709.3 naming conventions. See updated terminology definitions in Clause 4. * Updated the FOICx.k-<fec> naming conventions to align with ITU-T G.709.3 naming conventions. See updated terminology definitions in Clause 4. * Corrected AM/PAD/OH bit length in Figure 15 and Figure 16. * Added FlexO-1-DO and FOIC1.4-DO modes. * Added adaptations of FlexO-2 to FOIC2.6 (8QAM) and FOIC2.8 (16QAM) adaptation modes in Clause 8.5 and Clause 8.6. * Reordered sections 5 and 6 to sections 4 and 5. * Changed document title to Open ROADM MSA 5.0 W-Port Digital Specification (100G-400G). * Fixed typo in Figure 11 and included note on IID ordering when interleaving. * Updates to FlexO-x-D0 signal and FlexO-x-D0 instance definitions in 4.1 |

Table 1: Revision History

# Definitions

This document uses the following terms and definitions.

## Terms defined in this document

* **openFEC (oFEC) -** a block-based encoder and iterative Soft Decision (SD) decoder. With 3 SD iterations the Net Coding Gain (NCG) is 11.1 dB @ when the (DP-QPSK) and 11.6 dB @ 10-15 (DP-16QAM), with pre-FEC BER threshold of 2.0×.
* **FlexO-x-DO[[1]](#footnote-1) -** an information structure consisting of a G.709.1 FlexO-x (x=2,3,4) frame structure protected with oFEC.
* **FlexO-x-DO signal instance -** Refers to an individual FlexO-x-DO member that is part of a FlexO-x-DO-*m* signal group
* **FlexO-x-DO-*m* signal group -** Refers to the group of *m* FlexO-x-DO signal instances.
* **FlexO-x-DO interface instance -** Refers to an individual FlexO-x-DO member that is part of a FlexO-x-DO-*m* interface group
* **FlexO-x-DO-*m* interface group -** Refers to the group of *m* FlexO-x-DO interface instances.
* **FOICx.k-DO[[2]](#footnote-2)** - Refers to a single FlexO-x-DO signal within a FlexO-x-DO-*m* signal group operating over a dual polarization coherent interface with modulation order of k. For QPSK k=4, 8QAM k=8, 16QAM k=8.
* **FOICx.k-DO4** - Refers to a single FlexO-x-DO signal within a FlexO-x-DO-*m* signal group operating over a dual polarization coherent interface with DP-mQAM modulation (m=4,8,16).

## Abbreviations and Acronyms

This document uses the following abbreviations and acronyms.

AM Alignment Marker

BMP Bit-synchronous mapping Procedure

CRC Cyclic Redundancy Check

DSP Digital Signal Processor

DP Dual Polarization

DP-mQAM Coherent interface using DP-mQAM

DWDM Dense Wave Division Multiplexing

FEC Forward Error Correction

FlexO Flexible Optical Transport Network

FOIC FlexO Interface

GE Gigabit Ethernet

LSB Least Significant Bit

MSA Multi-Source Agreement

MSB Most Significant Bit

ODTUG Optical Data Tributary Unit Group

ODTUGk/Cn Optical Data Tributary Unit Group-k/Cn

ODTUjk Optical Data Tributary Unit j into k

ODTUk/Cn.ts Optical Data Tributary Unit k/Cn with ts tributary slots

ODU Optical Data Unit

ODUk/Cn Optical Data Unit-k/Cn

ODUk/Cn.ts Optical Data Unit k/Cn fitting in ts tributary slots

ODUk/CnP Optical Data Unit-k/Cn Path monitoring level

ODUk/CnT Optical Data Unit-k/Cn Tandem connection monitoring level

oFEC

OH Overhead

OTLk.n Group of n Optical Transport Lanes that carry one OTUk

OTLC.n Group of n Optical Transport Lanes that carry one OTUC of an

OTN Optical Transport Network

OTU Optical Transport Unit

OTUk/Cn Optical Transport Unit-k/Cn

PIC Photonic Integrated Circuit

QAM Quadrature Amplitude Multiplexing

QPSK Quadrature Phase Shift Keying

RES Reserved for future international standardization

ROADM Reconfigurable Add Drop Multiplexor

RS Reed-Solomon

WDM Wavelength Division Multiplexing

## Conventions

This document uses the following conventions:

* **Transmission order** - The order of transmission of information in all the diagrams is first from left to right and then from top to bottom unless explicitly called out as different. The most significant bit (bit 1) is illustrated at the left in all the diagrams (e.g., Row 1, Column 1).
* **Reserved bit(s)** - The value of a reserved bit or reserved bit for future standardization shall be set to “0”.
* **Non-Sourced bit(s)** - The value of any non-sourced bit shall read back as “0”.

# References

1. Open ROADM MSA specification version 3.00
2. Open ROADM MSA specification version 2.00
3. Open ROADM Yang Data models - [https://github.com/OpenROADM/Open ROADM\_MSA\_Public](https://github.com/OpenROADM/Open%20ROADM_MSA_Public)
4. IEEE Std. 802.3TM-2018, IEEE Standard for Information Technology – Telecommunications and Information Exchange Between Systems – Local and Metropolitan Area Networks – Specific Requirements Part 3: Carrier Sense Multiple Access with Collision Detection (CSMA/CD) Access Method and Physical Layer Specifications.
5. IEEE P802.3cn Task Force – Future work proposals
6. ITU-T G.709/Y.1331 (06/2018), Interfaces for the optical transport network
7. ITU-T G.709.1/Y1331.1 (06/2018), Flexible OTN short-reach interfaces
8. ITU-T G.709.2/Y.1331.2 (07/2018), OTU4 long-reach interface
9. ITU-T G.709.3/Y.1331.3 (11/2018), Flexible OTN long-reach interfaces
10. ITU-T G.798 (08/2018), Characteristics of transport equipment Description methodology and generic functionality.
11. ITU-T G.870/Y.1352 (2016) Terms and definitions for optical networks
12. Telcordia NEBSTM Requirements: Physical Protection, Telcordia Technologies Generic Requirements, GR-63-CORE Issue 3, March 2006.
13. G.694.1: Spectral grids for WDM applications: DWDM Frequency grid.
14. G.698.2: Amplified multichannel DWDM applications with single channel optical interfaces.

# Scope and Introduction

Open ROADM 3.0 defines specifications [1] for 100-400G single-wavelength optical ports (W-Ports) for muxponder/transponder/switch (MUXP, TRPN, SWITCH) node functions and for optical ports (MW, WR ports) of optical switching (ROADM) equipment (reference Figure 1).



Figure 1: Open ROADM Architecture reference

This specification defines the port function[[3]](#footnote-3) of a 100-400G W-Port, i.e., the protocols, data structures, and algorithms for end-to-end interworking between two (100-400G) W-Ports [1] that are connected via an Open ROADM optical layer network. The intra-system service interface(s) of this port function are fully specified in existing standards (e.g., the mapping and aggregation of client interface signals via OTUCn (n=1,2, 3, or 4) to a FlexO-x (x=1...4) frame structure by ITU-T G.709/Y.331 and G.709.1/Y.1331.1).

*Note: this specification makes no assumption about the allocation of functions on a line card or optical modules (albeit a typical module will provide a muxceiver function for OTN and Ethernet signals beyond 100G)*

Appendix A includes examples of the digital layer signal formats which might be present in a TRPN/MUXP/SWITCH node that is connected, mapped or aggregated for optical transport across the W-Ports. The physical interfaces and any terminated client port protocols identified in Appendix A are for informational purposes only. They are not considered relevant to the W-Port function.

The digital interface type of a 100-400G W-Port is called FlexO-x-DO (x=1,2,3, or 4 for 100G, 200G, 300G, and 400G respectively), and any signal that can be adapted (or mapped) to a FlexO-x information structure can be transported via the FlexO-x-DO port.

Open ROADM node (TRPN/MUXP/SWITCH) can further bond a number *m* of W-Ports into a FlexO-x-DO-*m* interface group to transport signals larger than 400G.

The information processing for a FlexO-x-DO-*m* interface group (as defined in this specification) is represented by the lower part of the functional model shown in Figure 2. One OTUC*n* signal (consisting of *n* OTUC instances) is mapped into the payloads of *n* FlexO frame structures, each FlexO frame structure payload area containing the bits of one OTUC signal. Up to this point, all functions are well defined in existing standards.

The scope of this specification is how the node internal set of *n ×* FlexO signals are mapped into *m* (*m* = [n/x]) FlexO-x-DO signals, each FlexO-x-DO signal containing "x" (frame/multi-frame aligned interleaved) FlexO signals (x ≥ 1). The conceptually serial FlexO-x-DO signal is adapted to a parallel multi-lane distribution (MLD) signal format with k lanes referred to as FOICx.k-DO. Each FlexO-x-DO is transported via one media element (OTSi).



Figure 2: FlexO-x-DO-*m* functional model

The digital formatting and processing done by the FlexO-x-DO port function for the Open ROADM MSA 5.0 100-400G coherent interface is shown in Figure 3.



Figure 3: Digital port functions for a single FlexO-x-DO signal

The mapping processes defined in Section 10 of this document are analogous to Clause 16 of ITU-G.709.3/Y.1331.3, which describes the adaptation of n × FlexO (n=2,4) frame/multi-frame aligned signals adapted to the FlexO-x-DO (x=2,4) frame structure. It should also be noted that the OFEC Frame Block Group OFBGz (z=8,6,4) described in G.709.3/Y.1331.3 Section 16 is structurally consistent with the multi-frame to OFEC adaptation descriptions defined in Clause 8 herein, with the number of FlexO-x rows in an OFBGz being dependent on "x".

# FlexO-x frame structure

The Open ROADM 3.0 FlexO-x frame structure is defined prior to oFEC coder adaptation and oFEC processing. Aspects of the FlexO-x frame structure that are specific to FlexO-x-DO are identified, otherwise the FlexO-x overhead is common to the requirements defined in ITU-T G.709.1.

## FlexO-1 Frame structure

The FlexO (100G) frame structure is defined in G.709.1 with G.709.1/Y.1331(18)\_F8-1 copied as Figure 4 below for reference. FlexO is a block format of 5140-bit columns × 128 rows.



Figure 4: FlexO-1 frame structure

ITU-T G.709.1 further defines an 8-frame multi-frame Flex-O structure shown in Figure 5.



Figure 5: FlexO multi-frame format

The multi-frame contains seven Fixed Stuff locations (FS) in the payload area of the FlexO multi-frames, each containing 1,280 bits. These FS locations located in row 65, columns 1 to 1,280 of the first seven frames within the multi-frame. The last frame within the multi-frame does not contain FS.

The FS bits are filled with all zeros and not checked at the receiver sink function.

The FlexO multi-frame payload, excluding the FS locations, consists of 5,244,160 bits (655,520 bytes) out of the total 5,263,360 bits (657,920 bytes) per FlexO multi-frame.

Alignment Markers (AM), Padding (PAD) and OverHead (OH) are inserted in the first row of each FlexO frame. The FlexO payload and overhead area is fully protected with oFEC, which is defined in Section 9.

## FlexO-2 Frame Structure

The FlexO-2 frame structure is a block format of 10280-bit columns × 128 rows with 960b columns of AM, 960b columns of PAD, and 640b of OH. Parity is added by the oFEC block and interleaver stages downstream of the FlexO-2 frame structure.

The FlexO-2 frame structure is shown in Figure 6.



Figure 6: FlexO-2 frame structure

Two frame/multi-frame aligned 100G FlexO instances are 10-bit interleaved into the FlexO-2 frame structure with Flex-O #A having the lowest IID and Flex-O #B having the highest IID. The interleaving is done in the same way as defined by FlexO-2 (ITU-T G.709.1 clause 8.4).

The interleaving process is shown in Figure 7.



Figure 7: interleaved FlexO frame to FlexO-2 frame structure

The FlexO-3 AM, PAD, and OH fields are the interleaved FlexO instances AM, PAD, and OH.

## FlexO-3 Frame Structure

The FlexO-3 frame structure is 10280b x 192 rows with 1440b columns of AM, 1440b columns of PAD, and 960b of OH. Parity is added by the oFEC block and interleaver stages downstream of the FlexO-3 frame structure.

The FlexO-3 frame structure is shown in Figure 8.



Figure 8: FlexO-3 frame structure

Three frame/multi-frame aligned 100G FlexO instances are 10-bit interleaved into the FlexO-3 with Flex-O #A having the lowest IID and Flex-O #C having the highest IID. The interleaving is done in similar way as defined by ITU-T G.709.1 clause 8.4.

This interleaving process is shown in Figure 9.



Figure 9: Interleaved FlexO frame to FlexO-3 frame structure

The FlexO-3 AM, PAD, and OH fields are the interleaved FlexO instances AM, PAD, and OH.

## FlexO-4 Frame Structure

The FlexO-4 frame structure is 10280b x 256 rows with 1920b columns of AM, 1920b columns of PAD, and 1280b of OH. Parity is added by the oFEC block and interleaver stages downstream of the FlexO-4 frame structure.

The FlexO-4 frame structure is shown in Figure 10.



Figure 10: FlexO-4 frame structure

Four frame/multi-frame aligned 100G FlexO instances are 10-bit interleaved into the FlexO-4 frame structure with Flex-O #A having the lowest IID and Flex-O #D having the highest IID. The interleaving is done in the same way as defined by FlexO-4 in ITU-T G.709.1 clause 8.4).

This process is shown in Figure 11.



The FlexO-4 AM, PAD, and OH fields are the interleaved FlexO instances AM, PAD, and OH.

# FlexO-x-DO adaptation

A logical flow diagram of the FlexO-x to FlexO-x-DO adaptation process is shown in Figure 12.



Figure 12: n x FlexO to FlexO-x-DO adaption process

The FlexO-x frame structure is adapted to the oFEC Coder block by adding padding after every n × 10280-bit rows. The data stream is then scrambled and passed to the oFEC encoder. Table 2 shows the relationships between the oFEC-x Coder Payload and the FlexO-x frame structure. The FlexO-x-DO is aligned and synchronized to the DSP frame, therefore the number of PAD bits, oFEC block and payload bits per DSP frame is modulation dependent.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **OFGBz** | **FlexO-x Rows** | **PAD (bits)** | **oFEC-x coder payload (bits)** | **oFEC Blocks** | **FlexO-x-DO (bits)** | **Modulation Format** | **Symbol Rate (Baud)** |
| **FOIC4.8-D0** | 116 rows | 992 | 1,193,472 | 168 | 1,376,256 | DP-16QAM | 63 139 467 923 |
| **FOIC3.6-D0** | 87 rows | 744 | 895,104 | 126 | 1,032,192 | DP-8QAM | 63 139 467 923 |
| **FOIC2.4-D0** | 58 rows | 496 | 596,736 | 84 | 688,128 | DP-QPSK | 63 139 467 923 |
| **FOIC2.8-D0** | 116 rows | 992 | 1,193,472 | 168 | 1,376,256 | DP-16QAM | 31 569 733 961 |
| **FOIC1.4-D0** | 116 rows | 496 | 596,736 | 84 | 688,128 | DP-QPSK | 31 569 733 961 |

Table 2: oFEC adaptation rates

The digital formatting and processing done by the DSP to adapt the FlexO-x to the oFEC Coder is shown in Figure 13.



Figure 13: Digital processes of FlexO-x to FlexO-x-DO adaptation.

## Padding Insertion/Removal

The oFEC block processing is aligned and synchronized to the DSP super-frame (See Section 11.1). Pad bits are appended to the Flex-O data to enable this alignment. The PAD is removed after the decoder on the receive interface. The PAD is an all-zero field that gets scrambled prior to encoding and removed after decoding and descrambling.

## FlexO-4 adaptation to FOIC4.8-DO

For FlexO-4 adaptation to a FOIC4.8-DO line interface, 116 rows of FlexO-4 information (1,192,480 bits) plus 992 bits of pad (1,193,472 bits total) are scrambled and then bitwise demultiplexed to two oFEC encoders, each of which operate on input blocks of 3,552 bits and produce output blocks of 4096 bits. To process the 1,193,472 bits each encoder operates on 168 input blocks of 7104 bits.

Figure 14 shows the adaptation of the FlexO-4 frame structure to the oFEC input block structure.



Figure 14: FlexO-4 adaptation to an FOIC4.8-D0 line interface

## FlexO-3 adaptation to FOIC3.6-DO

For FlexO-3 adaptation to an FOIC3.6-DO line interface 87 rows of FlexO-3 information (894,360 bits) plus 744 bits of pad (895,104 bits total) are scrambled and then bitwise demultiplexed to two oFEC encoders, each of which operate on input blocks of 3552 bits and produce output blocks of 4,096 bits. To process the 895,104 bits each encoder operates on 126 input blocks of 7104 bits.

Figure 15 shows the adaptation of the FlexO-3 frame structure to the oFEC input block structure.



Figure 15: FlexO-3 adaptation to an FOIC3.6-D0 line interface

## FlexO-2 adaptation to FOIC2.4-DO

For FlexO-2 adaptation to an FOIC2.4-D0 line interface 58 rows of FlexO-2 information (596,240 bits) plus 496 bits of pad (596,736 bits total) are scrambled and then bitwise demultiplexed to two oFEC encoders, each of which operate on input blocks of 3,552 bits and produce output blocks of 4,096 bits. To process the 596,736 bits each encoder operates on 84 input blocks of 7104 bits.

Figure 15 shows the adaptation of the FlexO-2 frame structure to the oFEC input block structure.



Figure 16: FlexO-2 adaptation to an FOIC2.4-D0 line interface

## FlexO-2 adaptation to FOIC2.8-DO

For FlexO-2 adaptation to a FOIC2.8-DO line interface, 116 rows of FlexO-2 information (1,192,480 bits) plus 992 bits of pad (1,193,472 bits total) are scrambled and then bitwise demultiplexed to two oFEC encoders, each of which operate on input blocks of 3,552 bits and produce output blocks of 4096 bits. To process the 1,193,472 bits each encoder operates on 168 input blocks of 7104 bits.

Figure 14 shows the adaptation of the FlexO-2 frame structure to the oFEC input block structure.



Figure 18: FlexO-2 adaptation to an FOIC2.8-D0 line interface

## FlexO-1 adaptation to FOIC1.4-DO

For FlexO-1 116 rows of information (596,240 bits) plus 496 bits of pad (596,736 bits total) are scrambled and then bitwise demultiplexed to two oFEC encoders, each of which operate on input blocks of 3,552 bits and produce output blocks of 4,096 bits. To process the 596,736 bits each encoder operates on 84 input blocks of 7104 bits.

Figure 15 shows the adaptation of the FlexO-1 frame structure to the oFEC input block structure and then to the encoder input block sequence.



Figure 18: FlexO-1 to oFEC input block adaptation

## Frame Synchronous Scrambling

The scrambler/descrambler is located before oFEC encoder on transmit, and after the oFEC decoder on receive. The operation of the scrambler shall be functionally equivalent to that of a frame-synchronous additive scrambler of sequence 65535 and the generating polynomial shall be:

x16+ x12+ x3+ x+1.

The scrambler/descrambler resets to 0xFFFF at the start of each new oFEC input block structure. The scrambler runs continuously over the entire oFEC input block. Figure 17 shows a functional diagram of the frame synchronous scrambler.



Figure 19: Frame synchronous scrambler

# Open Forward Error Correction (oFEC)

NOTE: *This section of the document uses zero-based indexing for mathematical formula convenience.*

The oFEC encoding block shown in Figure 18consists of two FEC encoders/decoders (ENC0 and ENC1) operating in parallel. 7,104 bits are bit de-interleaved/interleaved to/from each encoder/decoder. The encoder expansion ratio is 4096/3552.

The oFEC encoder and oFEC interleaver datapath is shown in Figure 18. The 7104 bits from the scrambler are bit demultiplexed into two parallel 3552/4096 encoder engines. Even numbered bits (0 based) go to encoder 0 (ENC0), and odd numbered bits to encoder 1 (ENC1).



Figure 20: oFEC block encoder and oFEC Interleaver

## oFEC encoding codec

**Note:** *The section below describes a single instance of the oFEC encoder engine, i.e., one of ENC0 or ENC1 in Figure 18.*

For the Open ROADM 5.0 applications two encoding engines operate in parallel, with each engine producing an oFEC codeword. A codeword in oFEC is a semi-infinite set of bits organized in a matrix with semi-infinite number of rows and N columns (N = 128).

It has the property that each bit is part of two “constituent component codewords,” in which each constituent component codeword is a binary vector *x* of length 2N satisfying the parity check constraint *x*H = 0, where H is a (2N, 2N – k) binary parity check matrix, with 2N > k > N. Here k = 239, and each constituent component codeword has (2N – k) = 17 parity bits. The fraction of bits that are parity bits is 17/128, the rate of the code is 111/128 = 0.867, and the overhead is 17/111 = 15.3%.

Specifically, in oFEC, H is a parity check matrix of an extended *BCH*(256, 239) code. This BCH code has a minimum Hamming distance of 6. oFEC uses a BCH textbook encoding with a parity check matrix H that is specified below.

The constituent component codewords are ordered as explained below to allow high speed parallel encoding and decoding. To define what bits are part of a given constituent component codeword the following structure is used:

* The infinite matrix of bits is partitioned in square blocks of B B bits (B = 16), arranged in rows and columns as shown in Figure 19. There are N/B blocks per row (N/B = 8), and each square block is identified by a square block row number, R, and a square block column number, C, where C= 0, 1, …, N/B–1, appearing respectively on the left-hand side and at the top of the figure.
* Each bit inside a square block is identified by its row number, r, where r = 0, 1, …, B – 1, and column number, c, where c = 0, 1, …, B – 1, where bit 0, 0 is at the upper left corner of a block. Overall, each bit in the infinite matrix is identified by a quadruple {R, C, r, c}.
* The number of guard-block rows needs to be even with a value 2G, (e.g., G = 2, or 2G = 4 rows, in Figure 19)



Figure 21: Structure of an openFEC Coder

A row of bits is identified by (R, r), with a square block row number R and a bit row number r within that block, where r = 0, 1, …, B – 1. A constituent component codeword can be identified by the number of the row that contains all bits of the 2nd half of the codeword. The kth bit (k = 0, 1, …, 2N – 1) of constituent component codeword (R, r) is the bit identified with the quadruple:

* If k < N: {(R ^ 1) – 2G − 2 N/B + 2 [k/B], [k/B], (k % B) ^ r, r} (1)
* If k ≥ N: {R, [(k – N)/B], r, (k % B) ^ r} (2)

where,

* [ . ] denotes the floor operator,
* (a % b) denotes the value of a modulo b, and
* (a ^ b) represents the number with a binary representation equal to the bitwise “exclusive or” of the binary representations of the numbers a and b.

These formulas are illustrated in Figure 19. The union of line segments (both vertical and horizontal) of a given color shows the bits forming a constituent component codeword but the ordering in the segments is not the ordering in the codeword.

For example, consider the constituent component codeword (20, 0). The position of its bits in the semi-infinite matrix are indicated by the red line segments. Bits 0 to 15 are in column 0 of block (1, 0), bits 16 to 31 in column 0 of block (3, 1), …, and bits 112 to 127 in column 0 of block (15, 7). The bit indices go up as one descends in the columns.

Bits 128 to 255 are in row 0 of blocks (20, 0) to (20, 7), and their indices go up moving to the right within a row.

Bits 0 to 127 are referred to as the “front” of a constituent component codeword, and bits 128 to 255 as the “back.”

Note that each bit in the oFEC encoder belongs to the front of a constituent component codeword and to the back of another one. Also, if the back of a constituent component codeword is in an odd-numbered row of square blocks (yellow background), then its front is an even-numbered row of square blocks (blue background), and conversely.

The square blocks located below the “front bits” and above the “back bits” of a given constituent component codeword are so called guard blocks, relative to the constituent component codeword of interest.

Continuing the example, the bits of constituent component codeword (20, 15), identified by the orange line segments, are in the same blocks as the segments of constituent component codeword (20, 0). However, because “r” is 15 instead of 0 as in the previous example, the expressions “^ r” in formulas (1) and (2) become significant, and the bits are taken in reverse order in each block. For example, bits 0 to 15 in the front of codeword (20, 15) are bits 15 to 0 in column 15 of block (1, 0).

***Note:*** *The oFEC code is a block-convolutional code, and its performance is characterized by its “error events.” Without the “^ r” permutation, there are about 625,000 possible error events of weight 36 that can start at every decoding of a constituent component codeword. For comparison, a Product Code based on the same constituent component codeword has more than 3.3e13 codewords of weight 36. The presence of the “^ r” permutation can be observed to eliminate error events of weight 36. Consequently, the minimum Hamming distance of the oFEC code is at least 42.*

Encoding

Encoding is done sequentially, in order of increasing row index. At the time when a constituent component codeword (R, r) is being encoded, all constituent component codewords (R’, r’) with R’ < R – 2G must be already be encoded.

To encode a constituent component codeword (R, r), form a vector *x* of length 2N where the front N bits are read from previously encoded bits in the infinite matrix according to formula (1) above. In the back, the first k – N (i.e., 111) bits are fresh information bits. The last 2N – k (i.e., 17) back bits are parity bits that can be calculated to satisfy *x*H = 0. After encoding, the N back bits are placed at their positions in the infinite matrix according to formula (2) above, and bits in those positions are output to an interleaver.

Considering Figure 19, we see that G is large enough to allow the parallel encoding of 2 B (G + 1) = 96 constituent component codewords, assuming the pipeline delay is small. This number is considerably reduced when the pipeline delay increases, which is typically the case in the decoder.

One can also see that at most N/B (N/B + 2 G + 1) = 104 square blocks need to be kept in the encoder memory (excluding the current input). The square blocks that must be kept in memory in order to encode block rows 20 and 21 are surrounded by the dashed line in Figure 19.

A large G allows for longer pipeline delays in the encoding and decoding operations and allows for more parallel execution in the encoder and decoder, at the expense of increased memory.

### Encoder interface

The encoder input consists of rectangular blocks of size (2B) × (2N – k) = 32 × 111 bits. The encoder input blocks are numbered 0, 1, 2, …. The input bits into the encoder are sequenced. The ith input bit is placed in the encoder input block [i / (32 × 111)] at the position indicated by the value i % (32 × 111) in Figure 20. Note that an encoder input block is divided in 16×16-bit blocks, except along the right edge where their size is 16×15.

Bit k = 0, 1, 2… of row p in encoder input block P is placed in position N + k of constituent component codeword (2 P + [p/B], p % B).



Figure 22: Sequencing of bits within an input block

The encoder output consists of rectangular blocks of size (2 B) × N = 32 × 128 bits. The encoder output blocks are numbered 0, 1, 2, …. Bit k = 0, 1, 2, ... of row p in rectangle P is the bit {2P + [p/B], [p/B], k/B, p % B} of the semi-infinite array.

The bits within an output block are sequenced according to Figure 21.



Figure 23: Bit numbering within an output block

### Formal encoder definition

This section directly describes the encoder (ENC0 or ENC1) output bits as a function of the input bits, integrating the diverse elements that have been described in previous sections.

An oFEC encoder is an entity that produces a binary output y(i) from a binary input u(i), where i = 0, 1, 2, ….

The relationship between y and u is expressed through intermediate variables.

In particular, there is a four-dimensional array V(R, C, r, c), where R is an integer; C = 0, 1, …,7; r = 0, 1, …,15; and c = 0, 1, …, 15.

Associated with array V, there are constituent component codeword vectors WR,rw with elements WR,r(i), where R ≥ 0, r = 0, 1. 2….15, and i = 0, 1, …, 255.

V((R ^ 1) − 20 + 2 [k/16], [k/16], (k % 16) ^ r, r) for k < 128

For R ≥ 0, WR,r(k) =

V(R, [(k − 128)/16], r, (k % 16) ^ r) for 128 ≤ k < 256

where,

* [ . ] denotes the floor operator,
* (a % b) denotes the value of a modulo b, and
* (a ^ b) represents the number with a binary representation equal to the bitwise “exclusive or” of the binary representations of the numbers a and b.

The bits in the WR,r satisfy the following equalities:

For R ≥ 0, r = 0, 1, …, 15 and k = 0, 1, …, 110

WR,r(128 + k) = u([R/2] × 32 × 111 + ((R % 2) × 16 + r ) × (16 – [k/96]) + [k/16] × 512 + k % 16)

For R ≥ 20, WR,r H = 0, where H is a parity check matrix of an extended *BCH*(256, 239) code, using a textbook encoding; i.e., if *x* is a vector satisfying *x*H = 0, then

1. *x* has an even parity, and
2. if the first 255 bits of *x* are seen as the binary coefficients of a polynomial x(t) of degree 254 (with bit 0 of *x* being the coefficient of power 254), with t being the indeterminate, then this binary polynomial x(t) is divisible by the binary codeword generator polynomial t16 + t14 + t13 + t11 + t10 + t9 + t8 + t6 +t5 + t +1.

The output y satisfies the relationship

For R ≥ 0; C = 0, 1, ..., 7; r = 0, 1, …, 15; and c = 0, 1, …, 15.

V (R, C, r, c) = y([R/2] × 32 × 128 + (R % 2) × 256 + C × 16 × 32 + r × 16 + c)

It can be observed that 20 × 16 ×17 values are left undefined in WR,r and in V(R, C, r, c) for 0 ≤ R < 20, and thus also in the output y. This is by design; an implementation can choose any convenient values.

However, for test vectors, the output needs to be totally specified. To that end, the following additional constraints are added:

For 0 ≤ R < 20, WR,r H’ = 0, where H’ is a 256 × 17 binary matrix where the first 128 rows are all zero and the last 128 rows are equal to the last 128 rows of H.

## Decoding

Any of the iterative algorithms designed for turbo decoding of Product Codes can easily be adapted to decode oFEC codewords.

For use with iterative decoding, observe that the bits in square block row will all have been decoded as front bits in later constituent component codewords after 2 (N/B + G + 1) rows of blocks have been decoded. Specifically, in Figure 19, bits in square block row R = 0 will all have been decoded as front bits by the time block row 21 has been decoded. It then makes sense to decode the constituent component codewords in block row 0 again.

## oFEC Interleaver

The FEC datapath is shown in Figure 18. After oFEC encoding, mapping the bit stream is interleaved by a block interleaver. The interleaver block size is 172,032 bits (42 encoder output blocks, 21 from ENC0 and 21 from ENC1). The number of interleaver blocks per FlexO-x-DO structure is dependent on the modulation:

* DP-16QAM = (1376256/172032) = 8
* DP-8QAM = (1032192/172032) = 6
* DP-QPSK = (688128/172032) = 4

### oFEC Interleaver architecture

The 172,032 bits in an interleaver block are organized as an (84, 8) array of 16 bit × 16-bit square blocks; see Figure 22 below. Note that the format is similar to the format used by the encoder and decoder. We then apply the two following mechanisms:

1. An intra-block interleaver that reorders the bits in each 16×16 square block to ensure that the bits in each row and column of a square block at the encoder output are remapped almost uniformly in the square block for transmission on the line. That operation can be seen as happening on input to the interleaver.
2. An inter-block interleaver that attempts to have nearby symbols on the line contain bits that are widely separated in the encoder output.

The interleaver is full rate, but it is fed by two half rate encoders, ENC0 and ENC1. Successive rows of square blocks from ENC0 will be written in even block rows of the interleaver buffer (yellowish colors in Figure 22), whereas successive rows of square blocks from ENC1 will be written in odd block rows (pinkish colors). Consequently, the content of an interleaver buffer is the square block row by square block row interleaving of vertical segments of the semi-infinite matrices of encoders ENC0 and ENC1.

### Intra-block interleaving

For the purpose of intra-block interleaving, the interleaver is considered to receive 16 × 16 square blocks of bits from the encoders, and each square block is considered separately.

The intra-block interleaving is specified by the following Table 3, which indicates the row and column of the source bit for each destination bit in the square block. For example, bit (14, 15) [base 0] encoder output block is placed in row 1 of column 0 of the corresponding interleaver square block.



Table 3: Source positions (row, col) for intra-block interleaving

***Note:*** *The left entries of the pairs in this table form a Latin Square. The right entries almost form a Latin square, but they are duplicated in the first and last rows.*

### Inter-block interleaving

The intra-block permutation described in the previous section is applied to each square block in the buffer as it comes in from the encoder.

In addition to partitioning the interleaver buffer as a function of the encoder, ENC0 or ENC1, it is also partitioned in an upper half of 42 block rows (light color tones) and a lower half of 42 block rows (dark color tones). Overall, the buffer is then partitioned in 4 subsets, each containing 21 × 8 square blocks or 336 × 128 bits.

|  |  |
| --- | --- |
| Subset number | Row Blocks |
| 0 | 0, 2, …, 40 |
| 1 | 1, 3, …, 41 |
| 2 | 42, 44, …, 82 |
| 3 | 43, 45, …, 83 |

Table 4: Interleaver subsets

On output, groups of 8 bits are taken in turn from each subset, reading them out of a column of bits before proceeding to the next columns of bits. These output bits form the FlexO-x-DO structure.

Specifically, as shown in Figure 22, the first 8 bits are read from the top of first column of subset 0, then the first 8 bits from the first column of subsets 1, 2, and 3. Those 32 bits are then followed by the taking the next 8 bits in the first column of each of the subsets 0, 1, 2, and 3. After 42 such cycles of 4 × 8 bits each, the first bit column of the interleaver buffer will be completely read out, and the output process continues by reading bit columns 1 to 127.



Figure 24: Inter-block interleaving

***Note:*** *Bits are read by columns, rather than rows because interleaver columns are much longer than rows, so bits in a column are spread over more constituent component codewords than bits in a row, which increases the tolerance to long bursts. The maximum correctable burst length, when used with a hard decoder, is a traditional measure of interleaver quality. In this case it can be shown to be 2,681 bits.*

The bits read out of the interleaver are passed to the modulator where they are used in groups of S = 4, 6 or 8, for QPSK, 8QAM and 16QAM respectively in both the H and V polarizations.

***Note:*** *The output bits with even indexes are used to form symbols for the H polarization, whereas those in odd positions are formed to symbols in the V polarization. This simplifies the line BER estimation in each polarization. The H and V bits will appear at fixed positions in each square block in the decoder independently of the modulation.*

# Symbol Mapping and Polarization Distribution

This section describes the procedure for mapping encoded and interleaved oFEC Blocks to DP-QPSK and DP-mQAM constellation symbols for distribution on each (X/Y) polarizations. This procedure is illustrated in Figure 23 below.



Figure 25: DSP symbol mapping and polarization distribution

## Symbol mapping

Symbol mapping is modulation dependent. Each FlexO-x-DO structure is mapped to 172032 symbols in each polarization.

### DP-16QAM Symbols

The FlexO-x-DO, where x=[2,4] bits denoted by *ck* (k=0…1376255) are mapped to DP-16QAM symbols (*S*).

,

where,

* (maps to the in-phase (I) component of the X-pol of
* (maps to the quadrature-phase (Q) component of the X-pol of
* (maps to the I component of the Y-pol of
* (maps to the Q component of the Y-pol of

In each signaling dimension, the following mapping from binary label to relative symbol amplitude is defined as:

(0,0) → -3,(0,1) →-1,(1,1) → +1,(1,0) → +3

This mapping per polarization is further detailed in Table 5 below.

|  |  |  |
| --- | --- | --- |
|  | I | Q |
| (0,0,0,0) | -3 | -3 |
| (0,0,0,1) | -3 | -1 |
| (0,0,1,0) | -3 | 3 |
| (0,0,1,1) | -3 | 1 |
| (0,1,0,0) | -1 | -3 |
| (0,1,0,1) | -1 | -1 |
| (0,1,1,0) | -1 | 3 |
| (0,1,1,1) | -1 | 1 |
| (1,0,0,0) | 3 | -3 |
| (1,0,0,1) | 3 | -1 |
| (1,0,1,0) | 3 | 3 |
| (1,0,1,1) | 3 | 1 |
| (1,1,0,0) | 1 | -3 |
| (1,1,0,1) | 1 | -1 |
| (1,1,1,0) | 1 | 3 |
| (1,1,1,1) | 1 | 1 |

Table 5: DP-16QAM symbol amplitude map

### DP-8QAM Symbols

The FlexO-3-DO bits denoted by *ck* (k=0…1032191) are mapped to DP-8QAM symbols (*S*),

,

where,

* (maps to the in-phase/quadrature-phase (I/Q) component of the X-pol of
* (maps to the I/Q component of the Y-pol of

In each polarization, we define the following map from binary label to relative symbol amplitudes:

|  |  |  |
| --- | --- | --- |
|  | I | Q |
| (0,0,0) | 0 | -1 |
| (0,0,1) | -1.366 | -1.366 |
| (0,1,0) | -1.366 | 1.366 |
| (0,1,1) | -1 | 0 |
| (1,0,0) | 1.366 | -1.366 |
| (1,0,1) | 1 | 0 |
| (1,1,0) | 0 | 1 |
| (1,1,1) | 1.366 | 1.366 |

Table 6: 8QAM symbol amplitude map

### DP-QPSK Symbols

The FlexO-x-DO, where x=[1,3] bits denoted by *ck* (k=0…688127) are mapped to DP-QPSK symbols (*S*),

,

where,

* (maps to the in-phase (I) component of the X-pol of
* (maps to the quadrature-phase (Q) component of the X-pol of
* (maps to the I component of the Y-pol of
* (maps to the Q component of the Y-pol of

In each polarization, we define the following map from binary label to relative symbol amplitudes:

|  |  |  |
| --- | --- | --- |
|  | I | Q |
| (0,0) | -1 | -1 |
| (0,1) | -1 | 1 |
| (1,0) | 1 | -1 |
| (1,1) | 1 | 1 |

Table 7: DP-QPSK symbol amplitude map

# DSP Framing

This section describes the DSP framing format. The DSP super-frame consists of 48 DSP sub-frames. The DSP frame length expressed in symbols is modulation independent. The frame format is defined for each polarization (X/Y).



Figure 26: DSP Frame generation

## DSP super-frame

A DSP super-frame is defined as a set of 178176 symbols in each X/Y polarization. A DSP sub-frame consists of 3712 symbols. The DSP super-frame thus consists of 48 DSP sub-frames.

Pilot Symbols (PS) are inserted every 32 symbols starting with the first symbol of the first DSP sub-frame. Each DSP sub-frame starts with an 11-symbol training sequence. The first symbol of the training sequence is a Pilot Symbol. The first DSP sub-frame of the super-frame also includes the DSP super-frame Frame Alignment Word (FAW).

As illustrated in Figure 24 above, once the data stream has been mapped into symbols and distributed onto each polarization pilot symbols, training symbols, Frame Alignment Word (FAW), and other overhead are added to create the DSP super-frame/sub-frame structure. Pilot symbols, Training symbols, and FAW symbols are always mapped to the outer constellation points of the optical signal.

|  |  |  |  |
| --- | --- | --- | --- |
| **Parameter** | **DP-16QAM** | **DP-8QAM** | **DP-QPSK** |
| Constellation  Map |  |  |  |
| FAW | 22 Symbols | 22 Symbols | 22 Symbols |
| TS | 11 symbols per DSP sub-frame | 11 symbols per DSP sub-frame | 11 symbols per DSP sub-frame |
| PS | Every 32 symbols | Every 32 symbols | Every 32 symbols |

Table 8: FAW/TS/PS pattern

## DSP sub-frame

Each DSP super-frame is divided into 48 DSP sub-frames and each DSP sub-frame consists of 3,712 symbols.

The first DSP sub-frame of the DSP super-frame includes a 22 symbol Frame Alignment Word (FAW) used for alignment to the oFEC blocks. 74 additional symbols are reserved for future use/innovation.

The first DSP sub-frame includes:

22 symbol super-frame Frame Alignment Word (FAW) used for super-frame delineation and alignment to the oFEC block. 74 additional symbols are reserved for future use/innovation. The FAW sequence is different between X and Y polarizations.

74 symbols are reserved to be used for future proofing and for innovation. These symbols should be randomized to avoid strong tones.

11 symbols available for link training. The first Training Symbol (TS) is shared as a Pilot Symbol (PS) in each DSP sub-frame.

116 Pilot Symbols.

Every subsequent DSP sub-frame (sub-frames 2-48 of a DSP super-frame) include:

11 symbols available for link training. The first Training Symbol (TS) is shared as a Pilot Symbol (PS) in each DSP sub-frame.

116 Pilot Symbols

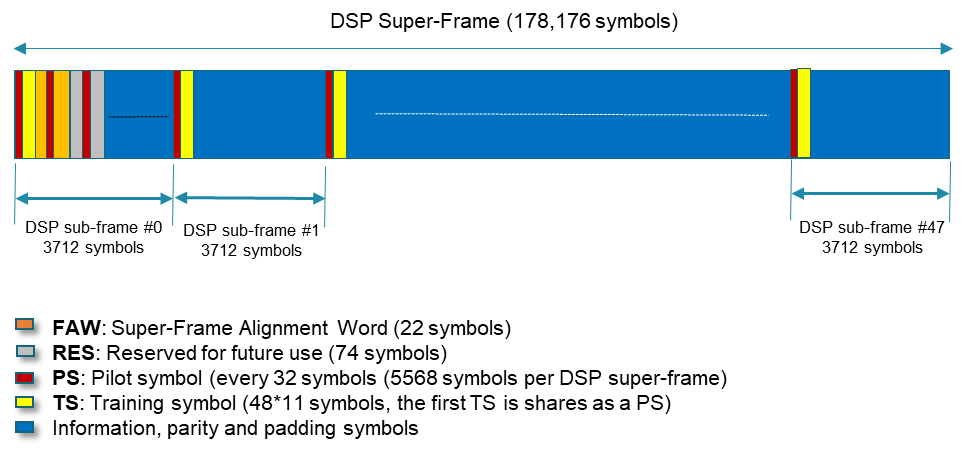


Figure 27: DSP super-frame



Figure 28: DSP sub-frames 1 to 47 of the DSP super-frame

## FAW Sequence

The required sequence and constellation corner relative symbol amplitude for the 16QAM FAW is shown in Table 9.

|  | 16QAM | | 8QAM | | QPSK | |
| --- | --- | --- | --- | --- | --- | --- |
| Index | FAW (X) | FAW (Y) | FAW (X) | FAW (Y) | FAW (X) | FAW (Y) |
| **1** | 3-3j | 3+3j | 1.366-1.366j | 1.366+1.366j | 1-1j | 1+1j |
| **2** | 3+3j | -3+3j | 1.366+1.366j | -1.366+1.366j | 1+1j | -1+1j |
| **3** | 3+3j | -3-3j | 1.366+1.366j | -1.366-1.366j | 1+1j | -1-1j |
| **4** | 3+3j | -3+3j | 1.366+1.366j | -1.366+1.366j | 1+1j | -1+1j |
| **5** | 3-3j | 3-3j | 1.366-1.366j | 1.366-1.366j | 1-1j | 1-1j |
| **6** | 3-3j | 3+3j | 1.366-1.366j | 1.366+1.366j | 1-1j | 1+1j |
| **7** | -3-3j | 3-3j | -1.366-1.366j | 1.366-1.366j | -1-1j | 1-1j |
| **8** | 3+3j | 3-3j | 1.366+1.366j | 1.366-1.366j | 1+1j | 1-1j |
| **9** | -3-3j | -3-3j | -1.366-1.366j | -1.366-1.366j | -1-1j | -1-1j |
| **10** | -3+3j | 3-3j | -1.366+1.366j | 1.366-1.366j | -1+1j | 1-1j |
| **11** | -3+3j | 3+3j | -1.366+1.366j | 1.366+1.366j | -1+1j | 1+1j |
| **12** | 3-3j | -3+3j | 1.366-1.366j | -1.366+1.366j | 1-1j | -1+1j |
| **13** | -3-3j | -3+3j | -1.366-1.366j | -1.366+1.366j | -1-1j | -1+1j |
| **14** | -3-3j | 3+3j | -1.366-1.366j | 1.366+1.366j | -1-1j | 1+1j |
| **15** | -3+3j | -3-3j | -1.366+1.366j | -1.366-1.366j | -1+1j | -1-1j |
| **16** | 3+3j | 3+3j | 1.366+1.366j | 1.366+1.366j | 1+1j | 1+1j |
| **17** | -3-3j | -3-3j | -1.366-1.366j | -1.366-1.366j | -1-1j | -1-1j |
| **18** | 3-3j | -3+3j | 1.366-1.366j | -1.366+1.366j | 1-1j | -1+1j |
| **19** | -3+3j | 3-3j | -1.366+1.366j | 1.366-1.366j | -1+1j | 1-1j |
| **20** | 3+3j | -3-3j | 1.366+1.366j | -1.366-1.366j | 1+1j | -1-1j |
| **21** | -3-3j | 3-3j | -1.366-1.366j | 1.366-1.366j | -1-1j | 1-1j |
| **22** | -3+3j | -3+3j | -1.366+1.366j | -1.366+1.366j | -1+1j | -1+1j |

Table 9: FAW Sequence

## Training Sequence

The required sequence and constellation corner relative symbol amplitude for the 16QAM TS is shown in Table 10. The constellation corner relative symbol amplitude for the 8QAM TS and QPSK TS should be scaled per Table 6 and Table 7 respectively.

|  | 16QAM | | 8QAM | | QPSK | |
| --- | --- | --- | --- | --- | --- | --- |
| Index | Training (X) | Training (Y) | Training (X) | Training (Y) | Training (X) | Training (Y) |
| **1\*** | -3+3j | -3-3j | -1.366+1.366j | -1.366-1.366j | -1+1j | -1-1j |
| **2** | 3+3j | -3-3j | 1.366+1.366j | -1.366-1.366j | 1+1j | -1-1j |
| **3** | -3+3j | 3-3j | -1.366+1.366j | 1.366-1.366j | -1+1j | 1-1j |
| **4** | 3+3j | -3+3j | 1.366+1.366j | -1.366+1.366j | 1+1j | -1+1j |
| **5** | -3-3j | -3+3j | -1.366-1.366j | -1.366+1.366j | -1-1j | -1+1j |
| **6** | 3+3j | 3+3j | 1.366+1.366j | 1.366+1.366j | 1+1j | 1+1j |
| **7** | -3-3j | -3-3j | -1.366-1.366j | -1.366-1.366j | -1-1j | -1-1j |
| **8** | -3-3j | -3+3j | -1.366-1.366j | -1.366+1.366j | -1-1j | -1+1j |
| **9** | 3+3j | 3-3j | 1.366+1.366j | 1.366-1.366j | 1+1j | 1-1j |
| **10** | 3-3j | 3+3j | 1.366-1.366j | 1.366+1.366j | 1-1j | 1+1j |
| **11** | 3-3j | 3-3j | 1.366-1.366j | 1.366-1.366j | 1-1j | 1-1j |

Table 10: Training symbol sequence

\*The first symbol of the Training Sequence is processed as a pilot

## Pilot Sequence

Training symbols and pilot symbols shall be set at the outer 4 points of the DP-mQAM constellation.

The Pilot is a fixed PRBS10 mapped to the QPSK sequence with different seed values for X/Y.

* Seeds are selected so that pilots are DC balanced
* Seeds are selected so that the 1st symbol in the training sequence is also the first symbol in the pilot sequence
* The seed is reset at the head of every DSP super-frame

PRBS10

with seed = 0x19E

PRBS10

with seed = 0x0D0

2x115=230 bits are mapped to outer constellation points

2x115=230 bits are mapped to outer constellation points

Figure 29: Pilot Symbol (DP-16QAM modulation shown)

|  |  |  |
| --- | --- | --- |
| Generator polynomial | Seed X | Seed Y |
|  | 0x19E | 0x0D0 |

Table 11: Pilot Sequence



Figure 30: Pilot Seed and Sequencing

The required sequence and constellation corner relative symbol amplitude for the 16QAM PS is shown in Table 12. The constellation corner relative symbol amplitude for the 8QAM PS and QPSK PS should be scaled per Table 6 and Table 7 respectively.

The complete table is shown below:

|  | 16QAM | | 8QAM | | QPSK | |
| --- | --- | --- | --- | --- | --- | --- |
| **Index** | **Pilot X** | **Pilot Y** | **Pilot X** | **Pilot Y** | **Pilot X** | **Pilot Y** |
| **1** | -3+3i | -3-3i | -1.366+1.366i | -1.366-1.366i | -1+1i | -1-1i |
| **2** | 3+3i | -3-3i | 1.366+1.366i | -1.366-1.366i | 1+1i | -1-1i |
| **3** | 3-3i | 3-3i | 1.366-1.366i | 1.366-1.366i | 1-1i | 1-1i |
| **4** | -3+3i | 3+3i | -1.366+1.366i | 1.366+1.366i | -1+1i | 1+1i |
| **5** | 3-3i | -3-3i | 1.366-1.366i | -1.366-1.366i | 1-1i | -1-1i |
| **6** | 3-3i | 3+3i | 1.366-1.366i | 1.366+1.366i | 1-1i | 1+1i |
| **7** | -3-3i | -3+3i | -1.366-1.366i | -1.366+1.366i | -1-1i | -1+1i |
| **8** | 3+3i | -3+3i | 1.366+1.366i | -1.366+1.366i | 1+1i | -1+1i |
| **9** | -3+3i | -3-3i | -1.366+1.366i | -1.366-1.366i | -1+1i | -1-1i |
| **10** | 3+3i | 3+3i | 1.366+1.366i | 1.366+1.366i | 1+1i | 1+1i |
| **11** | 3+3i | 3+3i | 1.366+1.366i | 1.366+1.366i | 1+1i | 1+1i |
| **12** | -3-3i | -3-3i | -1.366-1.366i | -1.366-1.366i | -1-1i | -1-1i |
| **13** | 3+3i | 3+3i | 1.366+1.366i | 1.366+1.366i | 1+1i | 1+1i |
| **14** | 3-3i | 3+3i | 1.366-1.366i | 1.366+1.366i | 1-1i | 1+1i |
| **15** | 3+3i | 3-3i | 1.366+1.366i | 1.366-1.366i | 1+1i | 1-1i |
| **16** | 3-3i | 3+3i | 1.366-1.366i | 1.366+1.366i | 1-1i | 1+1i |
| **17** | 3+3i | 3+3i | 1.366+1.366i | 1.366+1.366i | 1+1i | 1+1i |
| **18** | 3-3i | -3+3i | 1.366-1.366i | -1.366+1.366i | 1-1i | -1+1i |
| **19** | -3+3i | -3-3i | -1.366+1.366i | -1.366-1.366i | -1+1i | -1-1i |
| **20** | -3-3i | 3-3i | -1.366-1.366i | 1.366-1.366i | -1-1i | 1-1i |
| **21** | 3+3i | 3-3i | 1.366+1.366i | 1.366-1.366i | 1+1i | 1-1i |
| **22** | -3+3i | 3+3i | -1.366+1.366i | 1.366+1.366i | -1+1i | 1+1i |
| **23** | -3+3i | -3+3i | -1.366+1.366i | -1.366+1.366i | -1+1i | -1+1i |
| **24** | 3-3i | 3-3i | 1.366-1.366i | 1.366-1.366i | 1-1i | 1-1i |
| **25** | -3+3i | 3-3i | -1.366+1.366i | 1.366-1.366i | -1+1i | 1-1i |
| **26** | -3+3i | 3+3i | -1.366+1.366i | 1.366+1.366i | -1+1i | 1+1i |
| **27** | -3+3i | -3+3i | -1.366+1.366i | -1.366+1.366i | -1+1i | -1+1i |
| **28** | -3+3i | 3+3i | -1.366+1.366i | 1.366+1.366i | -1+1i | 1+1i |
| **29** | -3-3i | 3+3i | -1.366-1.366i | 1.366+1.366i | -1-1i | 1+1i |
| **30** | 3-3i | 3-3i | 1.366-1.366i | 1.366-1.366i | 1-1i | 1-1i |
| **31** | -3-3i | -3+3i | -1.366-1.366i | -1.366+1.366i | -1-1i | -1+1i |
| **32** | 3+3i | -3-3i | 1.366+1.366i | -1.366-1.366i | 1+1i | -1-1i |
| **33** | -3+3i | 3-3i | -1.366+1.366i | 1.366-1.366i | -1+1i | 1-1i |
| **34** | -3+3i | -3-3i | -1.366+1.366i | -1.366-1.366i | -1+1i | -1-1i |
| **35** | -3+3i | -3-3i | -1.366+1.366i | -1.366-1.366i | -1+1i | -1-1i |
| **36** | 3-3i | 3-3i | 1.366-1.366i | 1.366-1.366i | 1-1i | 1-1i |
| **37** | 3-3i | 3-3i | 1.366-1.366i | 1.366-1.366i | 1-1i | 1-1i |
| **38** | -3-3i | -3-3i | -1.366-1.366i | -1.366-1.366i | -1-1i | -1-1i |
| **39** | -3-3i | 3+3i | -1.366-1.366i | 1.366+1.366i | -1-1i | 1+1i |
| **40** | 3-3i | -3-3i | 1.366-1.366i | -1.366-1.366i | 1-1i | -1-1i |
| **41** | -3-3i | 3-3i | -1.366-1.366i | 1.366-1.366i | -1-1i | 1-1i |
| **42** | 3-3i | 3-3i | 1.366-1.366i | 1.366-1.366i | 1-1i | 1-1i |
| **43** | -3+3i | -3-3i | -1.366+1.366i | -1.366-1.366i | -1+1i | -1-1i |
| **44** | -3+3i | -3-3i | -1.366+1.366i | -1.366-1.366i | -1+1i | -1-1i |
| **45** | -3-3i | 3+3i | -1.366-1.366i | 1.366+1.366i | -1-1i | 1+1i |
| **46** | -3+3i | -3+3i | -1.366+1.366i | -1.366+1.366i | -1+1i | -1+1i |
| **47** | -3-3i | 3+3i | -1.366-1.366i | 1.366+1.366i | -1-1i | 1+1i |
| **48** | 3+3i | -3+3i | 1.366+1.366i | -1.366+1.366i | 1+1i | -1+1i |
| **49** | 3+3i | 3-3i | 1.366+1.366i | 1.366-1.366i | 1+1i | 1-1i |
| **50** | -3+3i | -3+3i | -1.366+1.366i | -1.366+1.366i | -1+1i | -1+1i |
| **51** | 3-3i | 3+3i | 1.366-1.366i | 1.366+1.366i | 1-1i | 1+1i |
| **52** | 3-3i | -3+3i | 1.366-1.366i | -1.366+1.366i | 1-1i | -1+1i |
| **53** | 3-3i | -3+3i | 1.366-1.366i | -1.366+1.366i | 1-1i | -1+1i |
| **54** | -3-3i | 3+3i | -1.366-1.366i | 1.366+1.366i | -1-1i | 1+1i |
| **55** | 3-3i | -3+3i | 1.366-1.366i | -1.366+1.366i | 1-1i | -1+1i |
| **56** | 3+3i | -3+3i | 1.366+1.366i | -1.366+1.366i | 1+1i | -1+1i |
| **57** | -3+3i | -3-3i | -1.366+1.366i | -1.366-1.366i | -1+1i | -1-1i |
| **58** | -3-3i | 3-3i | -1.366-1.366i | 1.366-1.366i | -1-1i | 1-1i |
| **59** | 3-3i | 3-3i | 1.366-1.366i | 1.366-1.366i | 1-1i | 1-1i |
| **60** | 3+3i | -3+3i | 1.366+1.366i | -1.366+1.366i | 1+1i | -1+1i |
| **61** | 3-3i | 3+3i | 1.366-1.366i | 1.366+1.366i | 1-1i | 1+1i |
| **62** | -3-3i | -3-3i | -1.366-1.366i | -1.366-1.366i | -1-1i | -1-1i |
| **63** | 3-3i | 3+3i | 1.366-1.366i | 1.366+1.366i | 1-1i | 1+1i |
| **64** | -3+3i | -3+3i | -1.366+1.366i | -1.366+1.366i | -1+1i | -1+1i |
| **65** | 3-3i | 3-3i | 1.366-1.366i | 1.366-1.366i | 1-1i | 1-1i |
| **66** | 3+3i | 3+3i | 1.366+1.366i | 1.366+1.366i | 1+1i | 1+1i |
| **67** | 3-3i | -3-3i | 1.366-1.366i | -1.366-1.366i | 1-1i | -1-1i |
| **68** | -3+3i | 3-3i | -1.366+1.366i | 1.366-1.366i | -1+1i | 1-1i |
| **69** | 3-3i | -3+3i | 1.366-1.366i | -1.366+1.366i | 1-1i | -1+1i |
| **70** | -3+3i | -3+3i | -1.366+1.366i | -1.366+1.366i | -1+1i | -1+1i |
| **71** | 3+3i | -3+3i | 1.366+1.366i | -1.366+1.366i | 1+1i | -1+1i |
| **72** | -3-3i | -3-3i | -1.366-1.366i | -1.366-1.366i | -1-1i | -1-1i |
| **73** | -3-3i | -3+3i | -1.366-1.366i | -1.366+1.366i | -1-1i | -1+1i |
| **74** | 3-3i | 3+3i | 1.366-1.366i | 1.366+1.366i | 1-1i | 1+1i |
| **75** | -3+3i | -3-3i | -1.366+1.366i | -1.366-1.366i | -1+1i | -1-1i |
| **76** | 3-3i | -3-3i | 1.366-1.366i | -1.366-1.366i | 1-1i | -1-1i |
| **77** | -3+3i | -3-3i | -1.366+1.366i | -1.366-1.366i | -1+1i | -1-1i |
| **78** | -3-3i | 3+3i | -1.366-1.366i | 1.366+1.366i | -1-1i | 1+1i |
| **79** | 3+3i | -3-3i | 1.366+1.366i | -1.366-1.366i | 1+1i | -1-1i |
| **80** | 3+3i | -3-3i | 1.366+1.366i | -1.366-1.366i | 1+1i | -1-1i |
| **81** | 3+3i | 3-3i | 1.366+1.366i | 1.366-1.366i | 1+1i | 1-1i |
| **82** | -3-3i | -3-3i | -1.366-1.366i | -1.366-1.366i | -1-1i | -1-1i |
| **83** | -3-3i | 3+3i | -1.366-1.366i | 1.366+1.366i | -1-1i | 1+1i |
| **84** | 3+3i | -3-3i | 1.366+1.366i | -1.366-1.366i | 1+1i | -1-1i |
| **85** | 3-3i | -3-3i | 1.366-1.366i | -1.366-1.366i | 1-1i | -1-1i |
| **86** | -3+3i | -3-3i | -1.366+1.366i | -1.366-1.366i | -1+1i | -1-1i |
| **87** | 3+3i | 3-3i | 1.366+1.366i | 1.366-1.366i | 1+1i | 1-1i |
| **88** | 3-3i | -3+3i | 1.366-1.366i | -1.366+1.366i | 1-1i | -1+1i |
| **89** | -3-3i | -3+3i | -1.366-1.366i | -1.366+1.366i | -1-1i | -1+1i |
| **90** | 3-3i | 3-3i | 1.366-1.366i | 1.366-1.366i | 1-1i | 1-1i |
| **91** | 3-3i | 3+3i | 1.366-1.366i | 1.366+1.366i | 1-1i | 1+1i |
| **92** | -3+3i | 3-3i | -1.366+1.366i | 1.366-1.366i | -1+1i | 1-1i |
| **93** | -3-3i | 3-3i | -1.366-1.366i | 1.366-1.366i | -1-1i | 1-1i |
| **94** | 3+3i | -3+3i | 1.366+1.366i | -1.366+1.366i | 1+1i | -1+1i |
| **95** | -3-3i | 3-3i | -1.366-1.366i | 1.366-1.366i | -1-1i | 1-1i |
| **96** | -3-3i | 3-3i | -1.366-1.366i | 1.366-1.366i | -1-1i | 1-1i |
| **97** | 3+3i | -3+3i | 1.366+1.366i | -1.366+1.366i | 1+1i | -1+1i |
| **98** | -3+3i | 3-3i | -1.366+1.366i | 1.366-1.366i | -1+1i | 1-1i |
| **99** | 3-3i | -3-3i | 1.366-1.366i | -1.366-1.366i | 1-1i | -1-1i |
| **100** | -3-3i | 3+3i | -1.366-1.366i | 1.366+1.366i | -1-1i | 1+1i |
| **101** | 3+3i | -3-3i | 1.366+1.366i | -1.366-1.366i | 1+1i | -1-1i |
| **102** | -3+3i | -3+3i | -1.366+1.366i | -1.366+1.366i | -1+1i | -1+1i |
| **103** | -3-3i | -3+3i | -1.366-1.366i | -1.366+1.366i | -1-1i | -1+1i |
| **104** | -3-3i | 3+3i | -1.366-1.366i | 1.366+1.366i | -1-1i | 1+1i |
| **105** | 3+3i | -3+3i | 1.366+1.366i | -1.366+1.366i | 1+1i | -1+1i |
| **106** | 3-3i | 3-3i | 1.366-1.366i | 1.366-1.366i | 1-1i | 1-1i |
| **107** | 3+3i | 3+3i | 1.366+1.366i | 1.366+1.366i | 1+1i | 1+1i |
| **108** | -3+3i | -3+3i | -1.366+1.366i | -1.366+1.366i | -1+1i | -1+1i |
| **109** | -3-3i | 3+3i | -1.366-1.366i | 1.366+1.366i | -1-1i | 1+1i |
| **110** | -3+3i | -3-3i | -1.366+1.366i | -1.366-1.366i | -1+1i | -1-1i |
| **111** | -3-3i | -3+3i | -1.366-1.366i | -1.366+1.366i | -1-1i | -1+1i |
| **112** | -3+3i | 3-3i | -1.366+1.366i | 1.366-1.366i | -1+1i | 1-1i |
| **113** | -3+3i | -3+3i | -1.366+1.366i | -1.366+1.366i | -1+1i | -1+1i |
| **114** | 3+3i | 3+3i | 1.366+1.366i | 1.366+1.366i | 1+1i | 1+1i |
| **115** | 3+3i | 3-3i | 1.366+1.366i | 1.366-1.366i | 1+1i | 1-1i |
| **116** | -3-3i | 3-3i | -1.366-1.366i | 1.366-1.366i | -1-1i | 1-1i |

Table 12: Pilot Sequence

# Frame Expansion Rate

The FlexO-2.4-DO (200G DP-QPSK), FlexO-3.6-DO (300G 8QAM), and FlexO-4.8-DO (400G 16 QAM) the optical symbol rate is ~63.139467923 GBaud. For FOIC1.4-DO (100G DP-QPSK) and FOIC2.8 (200G DP-16QAM) the symbol rate is ~31.568733961 GBaud. Table 13 provides detail on expansion for each functional block.

|  |  |
| --- | --- |
| **Parameters** | **Mapping** |
| FEC Payload | FlexO |
| FEC algorithm | oFEC |
| FEC payload size (k) | 3,552 |
| FEC block size (n) | 4,096 |
| The number of FEC blocks in super frame | 168(16QAM)/126(8QAM)/84(QPSK) |
| The total payload size | 1,193,472(16QAM)  895,104(8QAM)  596,736(QPSK) |
| PAD before FEC | 992(16QAM)/744(8QAM)/496(QPSK) |
| The total payload size based on 257b | 1,192,480b (16QAM) 4,640x257b  894,360b (8QAM) 3,480x257b  596,240b (QPSK) 2,320x257b |
| PAD after FEC | 0 |
| The total bits | 1,376,256(16QAM)  1,032,192(8QAM)  688,128(QPSK) |
| Total number of symbols per before DSP frame OH | 172,032 |
| The number of FAW symbols | 22 |
| The number of RES symbols | 74 |
| The number of Training Symbols | 480 |
| The number of Pilot Symbols (PS) | 5,568 |
| The total symbol of super-frame | 178,176 |
| DSP sub-frame symbols | 3,712 |
| The number of DSP sub-frames per super-frame | 48 |
| Modulation format | 16QAM / 8QAM / QPSK |
| Baud rate | ~63.139467923 GBaud\* |
| ~31.568733961 GBaud |

Table 13: FlexO/oFEC expansion rates

\*63.139467923 (GBaud) = 447237897785.775 × (514/544) × (37296/37265) × (4096/3552) × (899/896) × (32/31)/8

# Appendix A – TRPN/MUXP/SWITCH client interface examples

This Appendix A is provided for informational purposes only. There are no limitations or assumptions about the allocation of functions on line cards, or optical modules, however, the most common configuration/implementations for a 100-400G Open ROADM TRPN/MUXP/SWITCH node, are likely to include support n (n=1...4)100G OTN or 100GE Client ports. The Client ports are individually terminated and adapted for transport over the FlexO-x-DO W-Port interface.

The Client interface signaling is expected to conform to existing protocol standards (e.g. G.709[6], G.709.1[7], andIEEE 802.3TM-2018[4]) and operate over standard physical layer interface(s). The termination, mapping and/or aggregation of these payloads into the FlexO-x frame structure is also fully specified in existing standards (e.g., the mapping and aggregation of client interface signals via OTUCn (n=1...4) to a FlexO-x (x=n) frame structure is fully specified by ITU‑T G.709/Y.1331 and G.709.1/Y.1331.1). These functions are not repeated here. What is included in this appendix are examples of several common TRPN/MUXP/SWITCH configurations, for a FlexO-x-DO W-Port.

Open ROADM 5.0 compliant 100-400G Transponder/Muxponder/Switch (TRPN/MUXP/SWITCH) can support multiple Client interface types.

OTN:

* OTUC1/OTU4 (100G)
* OTUC2 (200G)
* OTUC3 (300G)
* OTUC4 (400G)

Ethernet:

* 400GBASE-R
* 200GBASE-R
* 100GBASE-R

A coherent transponder/muxponder/switch (TRPN/MUXP/SWITCH) function may support the following sub functions:

* Termination/generation of the Client interface PMD, PMA, and PCS layers.
* Performance monitoring, error detection and signaling.
* Mapping/de-mapping of Client interface data to/from an OTUCn
* Mapping/de-mapping one OTUCn signal (consisting of n×OTUC instances) into the payload of n FlexO signals.
* Interleave/de-interleave of n×FlexO (frame/multi-frame aligned) signals to a FlexO-x frame structure (x=1...4).
* FlexO-x-DO processing.
* Symbol mapping/de-mapping, polarization distribution, and optical framing.
* Modulation/demodulation of the symbol stream to/from a FOICx-DO signal.

## Transponder (TRPN)

Table 14 shows various Client protocols supported by a W-Port TRPN function by capacity:

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Capacity** | **Mode** | **Client interfaces** | | | | | | | **Line Interface1** |
| **400GBASE-R** | **200GBASE-R** | **100GBASE-R** | **OTUC4**  **(400G)** | **OTUC3**  **(300G)** | **OTUC2**  **(200G)** | **OTUC1** |
| 400G | Transponder | 1 |  |  |  |  |  |  | **FOIC4.8-DO (16 QAM)** |
| 400G | Transponder |  |  |  | 1 |  |  |  | **FOIC4.8-DO (16 QAM)** |
| 300G | Transponder |  |  |  |  | 1 |  |  | **FOIC3.6-DO (8QAM)** |
| 200G | Transponder |  | 1 |  |  |  |  |  | **FOIC2.4-DO (QPSK)**  **FOIC2.8-DO (16QAM)** |
| 200G | Transponder |  |  |  |  |  | 1 |  | **FOIC2.4-DO (QPSK)**  **FOIC2.8-DO (16QAM)** |
| 100G | Transponder |  |  |  |  |  |  | 1 | **FOIC1.4-DO (QPSK)** |
| 100G | Transponder |  |  | 1 |  |  |  | 1 | **FOIC1.4-DO (QPSK)** |

Table 14: Open ROADM 3.0 Transponder (TRPN) functions

1For coherent signaling FOICx.k x= [1...4], k= [4,6,8]. See Section 10 for symbol mapping details of symbol mapping and polarity distribution.

***Note:*** *For a 100G Transponder based on FOIC1.4-SC reference Open ROADM 2.0 [2]*

## Muxponder (MUXP)

An Open ROADM 5.0 100-400G MUXP may support aggregation of *m* Client interfaces up to its specified output bandwidth. The Client interface signals may or may not be synchronous (from a common clock source). Client interfaces are mapped/multiplexed per ITU-T G.709 into an OTUCn (n=1..4), then 10b interleaved to a FlexO-x frame per ITU-T G.709.3. Support of mixed traffic and/or traffic originating from different sources can also be supported. Table 15 shows common examples of MUXP Client interface combinations.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Capacity** | **Mode** | **Client Interfaces** | | | **Line Interface1** |
| **200GBASE-R** | **100GBASE-R** | **OTU4** |
| 400G | Muxponder | 2 |  |  | **FOIC4.8-DO (16 QAM)** |
| 400G | Muxponder |  | 4 |  | **FOIC4.8-DO (16 QAM)** |
| 400G | Muxponder |  |  | 4 | **FOIC4.8-DO (16 QAM)** |
| 300G | Muxponder | 1 | 1 |  | **FOIC3.6-DO (8QAM)** |
| 300G | Muxponder |  | 3 |  | **FOIC3.6-DO (8QAM)** |
| 300G | Muxponder |  |  | 3 | **FOIC3.6-DO (8QAM)** |
| 200G | Muxponder |  | 2 |  | **FOIC2.4-DO (QPSK)** |
| 200G | Muxponder |  |  | 2 | **FOIC2.4-DO (QPSK)** |

Table 15: Open ROADM 3.0 Muxponder (MUXP) functions

1For coherent signaling FOICx.k x= [1...4], k= [4,6,8]. See Section 10 for symbol mapping details of symbol mapping and polarity distribution.

## Client interface bit rates

This section references ITU-T G.709[6],[7] to identify various Client interfaces, and their nominal bit rate. The Client interface can be locally or externally originated. Examples include:

* A single externally originated ODUCn signal coming from a single 100G FlexO-n-RS, or from a group of *m* bonded FlexO-x-RS ports (i.e., from a FlexO-x-RS-m group.
* A single locally originated ODUCn carrying 100G or 400G Ethernet payloads.
* A single locally originated ODU4 signal coming from an OTN framer/mapper.
* 100GE, 200GE, or 400GE signals coming from an Ethernet switch/router interface.

|  |  |  |
| --- | --- | --- |
| Client Interface | Nominal bit rate | tolerance |
| OTUC1 | 239/226 × 99.5328 Gbit/s = 105.258138053 Gbit/s | ± 20 ppm |
| OTUC2 | 2 × 239/226 × 99.5328 Gbit/s = 210.516276106 Gbit/s | ± 20 ppm |
| OTUC3 | 3 × 239/226 × 99.5328 Gbit/s = 315.774414159 Gbit/s | ± 20 ppm |
| OTUC4 | 4 × 239/226 × 99.5328 Gbit/s = 421.032552212 Gbit/s | ± 20 ppm |
| OTU4 | 239/227 × 99.5328 Gbit/s = 104.7944458 Gbit/s | ± 20 ppm |
| 100GE | 100 Gbit/s = 103.125 Gbit/s (66b/66b) | ± 100 ppm |
| 200GE | 2×20479/20480 × 103.125 Gbit/s = 206.239929199 Gbit/s | ± 100 ppm |
| 400GE | 4×20479/20480 × 103.125 Gbit/s = 412.4798583984 Gbit/s | ± 100 ppm |

Table 16 Client interface types and payload bit rates

For Open ROADM, the TRPN/MUXP/SWITCH function client interface(s) may or may not include FEC. When FEC is present the FEC is terminated at the Client interface sink function (e.g., PMD, PMA and lower layer PCS) before being processed to a FlexO-x-DO signal.

Table 17 defines the FlexO-x frame structure rates for the possible FlexO-x-DO signals prior to the addition of any oFEC overhead (reference Section 7 details).

|  |  |  |
| --- | --- | --- |
| FlexO-x frame structure | Nominal bit rate | tolerance |
| FlexO-1 (100G) | 1 × (239/226) × (4112/4097) × 99.5328 Gbit/s = 105.643510782 Gbits/s | ± 20 ppm |
| FlexO-2 (200G) | 2 × (239/226) × (4112/4097) × 99.5328 Gbit/s = 211.287021564 Gbits/s | ± 20 ppm |
| FlexO-3 (300G) | 3 × (239/226) × (4112/4097) × 99.5328 Gbit/s = 316.930532346 Gbits/s | ± 20 ppm |
| FlexO-4 (400G) | 4 × (239/226) × (4112/4097) × 99.5328 Gbit/s = 422.574043128 Gbits/s | ± 20 ppm |

Table 17: Line types and frame bit rates

[End of Document]

1. In addition to a port function and processing view associated with a FlexO-x-DO interface, there is the dual view of a FlexO-x-DO signal or data format. Hence, generally, “FlexO-x-DO is a prefix for objects like “interface,” or “port”, or “signal” or “data”. The same statements apply to other FlexO related terms. [↑](#footnote-ref-1)
2. The general type would be FOICx.k for k lane multilane format. For coherent signaling k=modulation order . [↑](#footnote-ref-2)
3. In OTN terminology this document would somewhat ambiguously be called an interface specification. Here we prefer to call it a port function specification, i.e., a layered description of the processing between an abstract system-internal service interface and a concrete network side system interface (located at a system boundary). [↑](#footnote-ref-3)