Python script for IEEE GEPBud3_1_16a

November 25, 2015

Contents

1 Scope

The purpose of this exercise is to do a deep analysis of the link model embodied in the IEEE GEPBud3_1_16a spreadsheet. The method selected to foster this deep analysis is to recast that model as a Python script. Python is selected, anticipating future link models with expanded capability to incorporate aspects not currently written into the GEPBud3_1_16a model, including:

- multiple lane crosstalk
- PAM4 optical modulation
- VCSEL nonlinearity and unequal rise / fall response times
- pre-emphasis in the optical transmitter
- equalization in the optical receiver

2 List of references

- The spreadhsheet is available at http://www.ieee802.org/3/ae/public/adhoc/serial_pmd/documents/10GEPBud3_1_16a.xls
- http://grouper.ieee.org/802/3/bm/public/ contains various presentations discussing the link budget model.
- ANSI/INCITS TR-60-2014, FC-MSQS-2, Fibre Channel-Methodologies for Signal Quality Specification-2, Annex B "Extending the link budget spreadsheet model"
- D. G. Cunningham and W. G. Lane, Gigabit ethernet networking, MacMillan, ISBN 1-7870-062-0, Chapter 9, the gigabit ethernet optical link model, 1999. This describes the earlier 1GbE link model, but is nevertheless a good reference to the model concepts.

3 Fiber channel model

3.1 Fiber attenuation

The fiber attenuation α is assumed to exhibit the following spectral dependence:

$$\alpha(\lambda_{min}) = c\_atten \cdot \left[1.05 + \frac{1}{(0.00094\lambda_{min})^4}\right] \tag{1}$$

c_atten is a scaling factor for fiber attenuation, depending upon fiber type, defined in cell P4, λ_{min} is the minimum laser wavelength, defined in cell C6, and α is used in column B of the spreadsheet.

3.2 Fiber chromatic dispersion

See FC-MSQS-2 Annex B.6, especially equation B.88, for a detailed discussion of chromatic dispersion in optical fibers. The dispersion is defined by two parameters: S_0 given in cell P8 and the wavelength λ_0 at which the fiber dispersion goes to zero, given in cell P7. Together they define a dispersion which is labeled $D(\lambda)$ in FC-MSQS-2, but which we will label as $D1$ to be consistent with the spreadsheet. $D1$ is given by

$$D1(\lambda) = \frac{S_0\lambda}{4} \cdot \left(1 - \frac{\lambda_0^4}{\lambda^4}\right) \tag{2}$$

This is labeled in the spreadsheet as 'Dispersion D1' and is calculated in cell P9. D1 has units of picoseconds per nanometer per kilometer. In addition, the spreadsheet calculates a second dispersion parameter $D2$ in cell AB4 as

$$D2 = 0.7S_0\Delta\lambda \tag{3}$$

in which $\Delta\lambda$ is the laser spectral width listed in cell C7. I expect this second term to be most critical for 1310 nm links in which the laser wavelength approaches the zero dispersion wavelength of the fiber. It gives negligible contributions for 850 nm links. Together, they define an aggregate dispersion which we will label D, given by

$$D = \sqrt{D1^2 + D2^2} \tag{4}$$

From this dispersion we calculate a chromatic dispersion bandwidth labeled BWcd in the spreadsheet and labeled bw_cd in the Python scripts, calculated in column F by

$$bw\_cd = \frac{0.187 \cdot 10^6}{delta\_lambda \cdot D} L \tag{5}$$

in which $delta\_lambda$ is the laser spectral width in nanometers, and L is the link reach in kilometers.

3.3 Fiber modal dispersion

The modal dispersion bandwidth labeled bw_dm in the Python script listed in section 9.1 and labeled 'effBWm' in the spreadsheet, is calculated in column G as

$$bw\_md = \frac{fib\_lbwp}{L} \tag{6}$$

fib_lbwp (fiber length-bandwidth product) is the name chosen for the parameter in the Python script listed in section 9.1 and which corresponds to cell P12 in the spreadsheet.

4 Response time calculations

4.1 Transmitter 10%-90% response time

The transmitter response is defined by its 20%-80% risetime, cell G2. Assuming a Gaussian impulse response, the corresponding 10%-90% risetime is given by

$$tx\_1090\_rise = 1.518 \cdot tx\_2080\_rise \tag{7}$$

as calculated in cell G3.

4.2 Receiver 10%-90% response time

The receiver response is defined by its bandwidth, cell T5, and dendoted rx_bw in the Python script. Its 10%-90% risetime is calculated in cell T7 as

$$rx\_1090\_rise = \frac{0.329 \cdot 10^6}{rx\_bw} \tag{8}$$

The factor 0.329 relates the risetime-bandwidth product for a single pole response.

4.3 Fiber 10%-90% response times

Given fiber chromatic dispersion bandwidth bw_cd and presumed Gaussian impulse response shape, the corresponding 10%-90% risetime cd_1090_rise is calculated by

$$cd\_1090\_rise = \frac{0.480 \cdot 10^6}{bw\_cd} \tag{9}$$

The risetime-bandwidth product 0.480 for a Gaussian-shaped impulse response is calculated in table B.3 in FC-MSQS-2 Annex B.2. Similarly, gvien the fiber modal bandwidth bw_md and presumed Gaussian impulse response shape, the corresponding 10%-90% risetime md_1090_rise is calculated by

$$md\_1090\_rise = \frac{0.480 \cdot 10^6}{bw\_md} \tag{10}$$

4.4 Composite response times and bandwidths

The eye opening for the link, which includes transmitter, receiver, and fiber channel bandwidth contributions, is dependent upon a composite response time Tc calculated in column I as the square root of the sum of the squares of the constituent response times.

The eye diagrams include a transmitter test set of plots which use a composite response time Tc in which an eye tester receiver bandwidth specified in cell W5 is substituted for the link receiver bandwidth specified in cell T5.

The laser residual noise intensity (RIN) penalty calculation requires an effective link bandwidth in which the transmitter response is excluded. Let's call this $rx\_bw\_rin\_test$. It is calculated in column AK as

$$\frac{1}{rx\_bw\_rin\_test^2} = \frac{1}{bw\_cd^2} + \frac{1}{bw\_md^2} + \frac{0.477}{rx\_bw^2} \tag{11}$$

I am guessing that the factor 0.477 is needed to convert a bandwidth assuming single pole impulse response to a bandwidth assuming Gaussian impulse response.

5 ISI eye closing calculations

5.1 Introductory comments

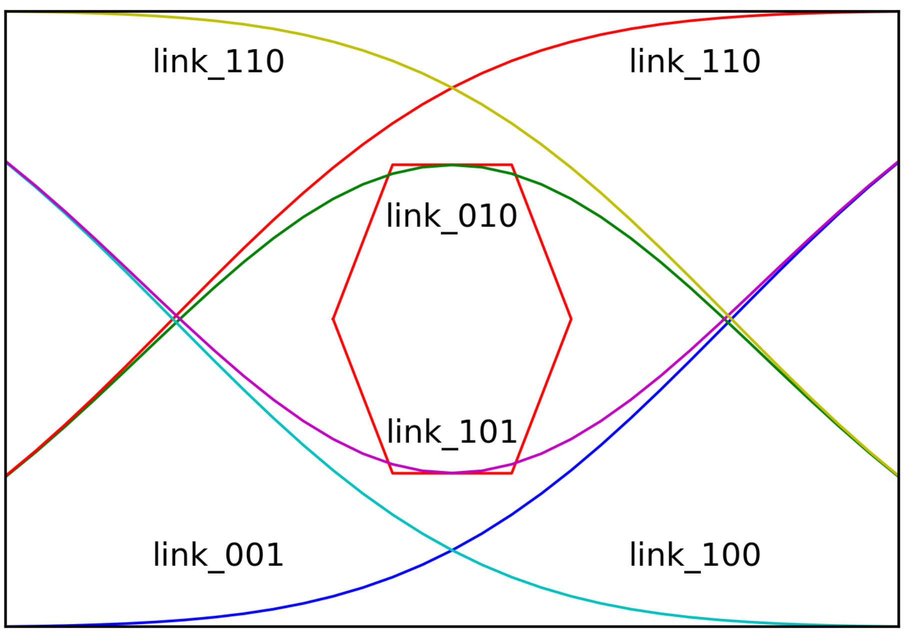

Figure 1: Typical eye predicted by the link model

The mathematical model underlying the isi eye closing calculations is detailed in FC-MSQS-2 Annex B.2. An eye diagram can be calculated from this model. Given any arbitrary data pattern, we find in practice that only 8 distinct lines appear in this eye diagram. The reason is that the composite Gaussian link impulse response only extends at most to immediately adjacent unit intervals, and no farther. We will label these lines as shown in the figure above.

Eye closing calculations and associated power penalties are performed with respect to the innermost eye in this diagram. The upper lid is $link\_010$, which is the unit pulse profile, labeled heye(t) in column Z, and given by equations B.16 and B.17 in FC-MSQS-2. The lower innermost lid is $link\_101$, which is related to $link\_010$ by

$$link\_101 = 1 - link\_010 \tag{12}$$

and hence the isi eye opening (linear units) is given by

$$isi = link\_010 - link\_101 = 2\, link\_010 - 1 \tag{13}$$

Our task in this section is to calculate the unit pulse response.

The first step is to calculate the response to a step function:

$$edge(t) = \frac{1}{2}\left[1 + erf\left(1.81238\frac{t}{Tc}\right)\right] \tag{14}$$

in which Tc is the composite response time of the link and t is time. See "edge response, Version 1", in table B.3 in FC-MSQS2, and equation B.14. Given this edge response, we next calculate the unit pulse profile response, equation B.16:

$$h(t) = \frac{1}{2}erf\left[\frac{1.81238}{Tc}\cdot\left(t + \frac{T}{2}\right)\right] - \frac{1}{2}erf\left[\frac{1.81238}{Tc}\cdot\left(t - \frac{T}{2}\right)\right] \tag{15}$$

This is the unit pulse response in the presence of a Gaussian composite impulse response. Time t is measured from the center of the pulse. T is the baud period (the unit interval), given by $1/br$ in which br is the nominal 'base rate' listed in cell C4.

In the presence of pulse width shrinkage, this gets a slight adjustment, substituting Tb_eff for T, in which

$$Tb\_eff = \frac{T}{speedup} \tag{16}$$

and $speedup$ is calculated in cell Y44 as

$$speedup = \frac{1}{1 - 10^{-6}br \cdot dcd\_dj} \tag{17}$$

The spreadsheet version defines a factor labeled B_1 in cell AB3, equal to 2.563. The arguments in the error functions in the spreadsheet have the form $2.563/\sqrt{8}$ for eye closing analyses. B_1 is calculated in equation B.15 of FC-MSQS2. My preference is to use the equivalent

$$0.90619 = \frac{2.5631}{\sqrt{8}} = erfinv(0.8) \tag{18}$$

For convenience in the Python script I define a scalar factor arg as

$$arg \equiv \frac{B\_1 \cdot Tb\_eff}{\sqrt{8}\,T} = \frac{erfinv(0.8)}{speedup} \tag{19}$$

Thus the unit pulse $link\_010$ has the form

$$\frac{1}{2}\{erf\left[\frac{2\,arg}{Tc}\left(t + \frac{T}{2\,speedup}\right)\right] - \frac{1}{2}erf\left[\frac{2\,arg}{Tc}\left(t - \frac{T}{2\,speedup}\right)\right]\} \tag{20}$$

We will need one more expression before continuing: the error function is anti-symmetric:

$$erf(-x) = -erf(x) \tag{21}$$

5.2 Isi_center

In the absence of any additional impairments, we calculate $link\_010$ at the center of the eye ($t = 0$) to be

$$link\_010 = erf\left(\frac{arg \cdot T}{speedup \cdot Tc}\right) \tag{22}$$

and hence the inner eye opening in linear units isi_center is

$$isi\_center = link\_010 - link\_101 = 2 \cdot erf\left(\frac{arg \cdot T}{speedup \cdot Tc}\right) - 1 \tag{23}$$

as given in equation B.18 in FC-MSQS-2 Annex B.2.

5.3 Isi_dj_center

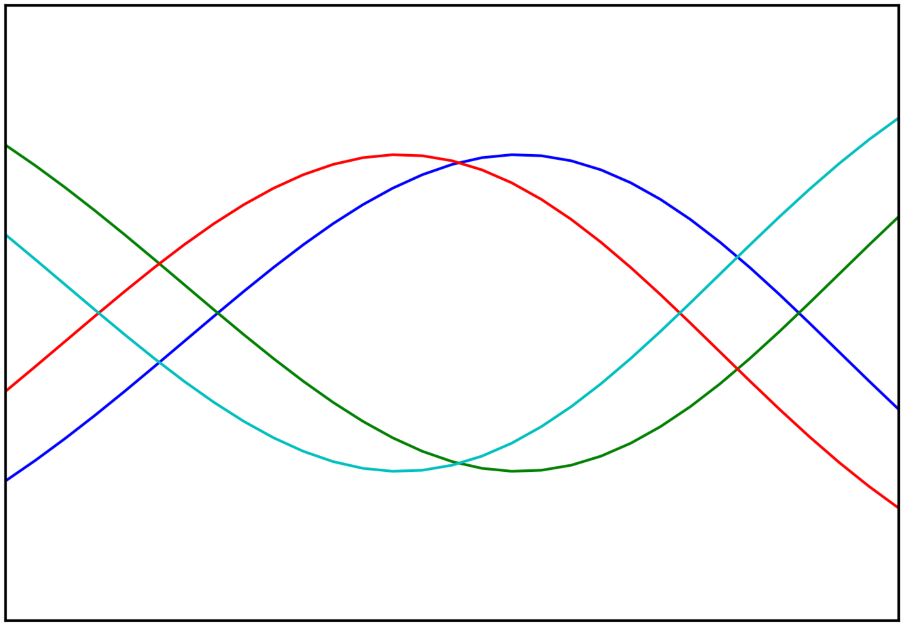

Figure 2: Typical inner eye shifted by deterministic jitter

Let us define "residual" deterministic jitter as the amount of jitter left over when the duty cycle distortion component has been subtracted. This jitter will cause the eye profile to shift to the left and right, as sketched in figure 2. The amount of shift is $dj - dcd\_dj$.

The upper lid $link\_010$ that is shifted to the right becomes

$$\frac{1}{2} erf\left[\frac{2\,arg}{Tc}\left(t + \frac{T}{2\,speedup} - \frac{dj - dcd\_dj}{2}\right)\right] - \frac{1}{2} erf\left[\frac{2\,arg}{Tc}\left(t - \frac{T}{2\,speedup} - \frac{dj - dcd\_dj}{2}\right)\right] \tag{24}$$

At the eye center $t = 0$ this evaluates to

$$link\_010 = \frac{1}{2} erf\left[\frac{arg \cdot T}{speedup \cdot Tc}(1 - dj\_ui)\right] + \frac{1}{2} erf\left[\frac{arg \cdot T}{speedup \cdot Tc}(1 + dj\_ui)\right] \tag{25}$$

in which we define dj_ui as

$$dj\_ui = \frac{speedup\,(dj - dcd\_dj)}{T} \tag{26}$$

From this we can calculate $link\_101$, and hence the eye opening which we will call isi_dj_center:

$$isi\_dj\_center = erf\left[\frac{arg \cdot T}{speedup \cdot Tc}(1 - dj\_ui)\right] + erf\left[\frac{arg \cdot T}{speedup \cdot Tc}(1 + dj\_ui)\right] - 1 \tag{27}$$

This is calculated in column AD.

5.4 Isi_reflection

Another eye-closing process is interferometric effects due to end reflections. This is not an issue for multimode links, but is documented here for completeness.

$$isi\_reflect = 1-2\cdot nf\_refl\cdot 10^{-0.1chil}\cdot refl\_mean\_lin\frac{\sqrt{2er\_lin\left[isi\_dj\_center(er\_lin+1)+er\_lin+1\right]}}{isi\_dj\_center} \tag{28}$$

nf_refl is the noise factor for reflections given in cell L10. $chil$ is the channel insertion loss, column C of the spreadsheet, given by

$$chil = \alpha\cdot L + fiber\_conn\_loss \tag{29}$$

α is the fiber attenuation evaluated at wavelength λ_min, L is the link reach, and $fiber\_conn\_loss$ is the connector loss listed in cell L7. $refl\_mean$ is calculated in cell AB5 as

$$refl\_mean\_lin = 10^{0.05(tx\_refl\_dB+rx\_refl\_dB)} \tag{30}$$

er_lin is calculated in cell AB6 as

$$er\_lin = 10^{0.1er\_dB} \tag{31}$$

isi_dj_center was calculated above.

5.5 Isi_dj_closed

What the spreadsheet labels "ISI, jitter & reflection closed eye", calculated in column AA, we will call in the Python script isi_dj_closed given by the following product:

$$isi\_dj\_closed = isi\_reflection\cdot isi\_dj\_center \tag{32}$$

5.6 Isi_corners

The link budget spreadsheet defines a hexagonal transmitter mask in cells C12-C14. The eye will be more closed at the mask corners, defined by $X2$, than at the center. Hence an important performance metric is the eye opening at the corners $isi\_corners$:

$$isi\_corners = erf\left[\frac{arg\cdot T}{speedup\cdot Tc}(1-eff\_rx\_eye)\right]+erf\left[\frac{arg\cdot T}{speedup\cdot Tc}(1+eff\_rx\_eye)\right]-1 \tag{33}$$

in which we have defined a parameter eff_rx_eye as

$$eff\_rx\_eye = speedup\cdot tx\_mask\_top \tag{34}$$

and tx_mask_top is the top of the hexagonal transmitter mask in UI units:

$$tx\_mask\_top = 2\ (05-X2) \tag{35}$$

5.7 Isi_dj_corners

Next we evaluate the eye closing at the corners of the transmitter mask, with excess dj included:

$$\begin{aligned} isi\_dj\_corners = {} & erf\left[\frac{arg\cdot T}{speedup\cdot Tc}(1-eff\_rx\_eye-dj\_ui)\right] \\ & + erf\left[\frac{arg\cdot T}{speedup\cdot Tc}(1+eff\_rx\_eye+dj\_ui)\right]-1 \end{aligned} \tag{36}$$

6 Link eye profiles

The content of this section follows closely that of the previous section. The major change is a rescaling and a shift in the time axis. Define dimensionless time variable θ, scaled to the unit interval period T, with the zero shifted to the left crossing:

$$\theta \equiv \frac{t + \frac{1}{2}T}{T} \tag{37}$$

This does not incorporate pulse width shrinkage (dcd_dj), so instead for eye profile calculations we use θ_{eff} defined as

$$\theta_{eff} \equiv \frac{t + \frac{1}{2}T_{eff}}{T_{eff}} = \frac{speedup \cdot t + \frac{1}{2}T}{T} \tag{38}$$

We can then derive θ_{eff} as a function of θ:

$$\theta_{eff} = \frac{1}{2} + speedup\left(\theta - \frac{1}{2}\right) \tag{39}$$

In this section we will define two distinct sets of eye profiles. One models the link response, and is given in rows 41-71, columns Z, AA, and AP-AW. The second set models the transmitter test configuration, given in rows 41-71 and columns Z-AI. The two differ only in the composite link response time. The link profiles incorporate the transmitter response, the fiber channel response at target link reach, and the link receiver response. We will call this $Tc(Ltarget)$. The transmitter test response incorporates the transmitter response, the fiber channel response for a 2 meter patch cord, and the eye mask receiver response instead of the link receiver. We will call the composite response time Tc_test.

Consider first the link response profiles. Let us define a convenience parameter Arg_link to be

$$Arg\_link \equiv \frac{B\_1 \cdot Tb\_eff}{\sqrt{2}\,Tc(Ltest)} = \frac{2\,erfinv(0.8)\,T}{speedup \cdot Tc(Ltest)} \tag{40}$$

The following table indicates the relationship between the Python names, the link spreadsheet names, and the corresponding columns in the spreadsheet:

Python name	**Spreadsheet name**	**Column**	**Python equation**
$link\_011$	$erf1$	AR	$\frac{1}{2} + \frac{1}{2}erf\left[Arg\_link \cdot \theta_{eff}\right]$
$link\_110$	$erf2$	AS	$\frac{1}{2} + \frac{1}{2}erf\left[Arg\_link \cdot (1 - \theta_{eff})\right]$
$link\_010$	oio	AT	$link\_011 + link\_110 - 1$
$link\_100$	$erf1'$	AU	$1 - link\_011$
$link\_001$	$erf2'$	AV	$1 - link\_110$
$link\_101$	ioi	AW	$1 - link\_010$

(41)

Next we consider the transmitter test link eye profiles as calculated in columns Z-AI. We define a convenience parameter Arg_test, the same as Arg_link except that we use Tc_test instead of $Tc(Ltarget)$. Our symbols and equations then become

Python name	**Spreadsheet name**	**Column**	**Python equation**
$test\_011$	$erf1$	AD	$\frac{1}{2} + \frac{1}{2}erf\left[Arg\_test \cdot \theta_{eff}\right]$
$test\_110$	$erf2$	AE	$\frac{1}{2} + \frac{1}{2}erf\left[Arg\_test \cdot (1 - \theta_{eff})\right]$
$test\_010$	oio	AF	$test\_011 + test\_110 - 1$
$test\_100$	$erf1'$	AG	$1 - test\_011$
$test\_001$	$erf2'$	AH	$1 - test\_110$
$test\_101$	ioi	AI	$1 - test\_010$

(42)

7 Power penalties

7.1 Power budget

The power budget is calculated in cell L8 as

$$p\_budget = tx\_oma\_min - rx\_sensitivity - fiber\_conn\_loss \tag{43}$$

7.2 Fiber attenuation power budget

The fiber attenuation contribution is calculated in column B as

$$p\_atten = alpha \cdot length \tag{44}$$

7.3 Laser residual noise intensity (RIN) power penalty

Laser rin is calculated by the following steps. First calculate the bandwidth of the link, exclusive of the transmitter response time contribution:

$$\frac{1}{rin\_bw^2} = \frac{1}{bw\_cd^2} + \frac{1}{bw\_md^2} + \frac{0.477}{rx\_bw^2} \tag{45}$$

This is done in column AK which then calculates v_rin:

$$v\_rin = 0.7 \cdot 10^6 \cdot rin\_test\_isi^2 \cdot rin\_bw \cdot 10^{0.1 \cdot rin} \tag{46}$$

Finally, as in column R, calculate rin power penalty \$p_rin \$:

$$p\_rin = -10 \cdot log_{10} \sqrt{1 - \frac{v\_rin \cdot Q^2}{isi\_dj\_refl\_closed}} \tag{47}$$

7.4 Mode partition noise power penalty

Mode partition noise (MPN) power penalty is first calculated by calculating parameter *beta* (column O):

$$beta = 3.14159 \cdot 10^{-6} \cdot speedup \cdot br\_nominal \cdot delta\_lambda \cdot d1 \cdot length \tag{48}$$

$d1$ is the same as $D1(\lambda)$ from the fiber channel model section. Given *beta* we next calculate the mpn noise-to-signal ratio sigma_mpn, as given by equation B.53 in Annex B4.4 of FC-MSQS-2:

$$sigma\_mpn = \frac{k\_mpn}{\sqrt{2}} \left(1 - e^{-beta^2}\right) \tag{49}$$

The factor k_mpn is calculated in equation B.47 as 0.723, simplified to 0.7 in the link budget spreadsheet. Finally, we calculate the mpn power penalty p_mpn (column Q) by

$$p\_mpn = -10 \cdot log_{10} \sqrt{1 - Q^2 \cdot sigma\_mpn^2} \tag{50}$$

7.5 Baseline wander power penalty

The baseline wander power penalty p_blw is calculated in cell T12 as

$$p\_blw = -10 \cdot log_{10} \sqrt{1 - \left(\frac{Q \cdot sigma\_blw}{isi\_tp4\_rx}\right)^2} \tag{51}$$

7.6 Reflection power penalty

The power penalty $p\_reflect$ due to coherent reflections at the link ends is calculated in column N as

$$p\_reflect = -10 log_{10} isi\_reflect \tag{52}$$

7.7 Modal noise variance

Given the assumed power penalty p_mn due to modal noise effects, we can calculate the noise variance v_mn as

$$v\_mn = \frac{1 - 10^{0.2 \cdot p\_mn}}{Q^2} \tag{53}$$

as done in cell AG7. This parameter is needed in the p_cross calculation considered below.

7.8 Power penalty due to eye closing, center of the eye, no deterministic jitter

The power penalty p_isi_center is calculated in column J as

$$p\_isi\_center = -10 \cdot log_{10}\left(isi\_center\right) \tag{54}$$

7.9 Power penalty due to eye closing, transmitter mask edge, no deterministic jitter

The power penalty $p\_isi\_corners$ is calculated in column K as

$$p\_isi\_corners = -10 \cdot log_{10}\left(isi\_corners\right) - p\_isi\_center \tag{55}$$

7.10 Power penalty due to eye closing, center of the eye, with excess deterministic jitter factored

The power penalty $p\_isi\_dj\_center$ is calculated in column L as

$$p\_isi\_dj\_center = -10 \cdot log_{10}\left(isi\_dj\_center\right) - p\_isi\_center \tag{56}$$

7.11 Power penalty due to eye closing, transmitter mask edge, with excess deterministic jitter factored

The power penalty $p\_isi\_dj\_corners$ is calculated in column M as

$$p\_isi\_dj\_corners = -10 \cdot log_{10}\left(isi\_dj\_corners\right) - p\_isi\_center - p\_isi\_corners \tag{57}$$

7.12 Power penalty due to accumulation of noise powers, center of the eye

In the presence of multiple noise penalties, their aggregate power penalty is not simply the sum of the individual noise penalties considered separately. To account for this, the spreadsheet defines a p_cross, which at the center of the eye is calculated in column S as

$$p\_cross\_center = -10 \cdot log_{10}\left[isi\_dj\_close\sqrt{1 - Q^2\left(\frac{sigma\_blw^2 + v\_rin}{isi\_dj\_close} + v\_mn + v\_mpn\right)}\right] \tag{58}$$

$$- p\_blw - p\_isi\_center - p\_dj\_center - p\_mpn - p\_reflect - p\_rin - p\_mn \tag{59}$$

7.13 Total power penalty at the center of the eye

The total power penalty at the center of the eye $p\_total\_central$ is calculated in column T as

$$p\_total\_center == p\_isi\_center + p\_dj\_center + p\_atten + p\_mpn + p\_reflect + p\_rin + p\_cross\_center + p\_mn \tag{60}$$

7.14 Total power penalty at the transmitter mask corners

The final power penalty, evaluated at the corners of the transmitter mask, is $p\_total\_corners$ as calculated in column U:

$$p\_total\_corners = p\_isi\_center + p\_isi\_corners + p\_attem + p\_mpn + p\_reflect + p\_rin + p\_cross\_center + p\_mn + p\_dj\_corners \tag{61}$$

8 Input parameters

The parameter input is assumed to be in an Excel file, located in the same directory as the routine that invokes GbE10_support. The Excel filename is assumed to be '10GbE_inputs.xlsx', and the inputs are assumed to be in worksheet named '850S2000' unless otherwise specified when the GbE10 class is instantiated. The input parameters are:

Description	Cell	Python parameter	Units
Link parameters			
Nominal baud rate	C4	br_nominal	megabaud
Bit error rate target		ber_target	dimensionless
Fiber connector loss	L7	fiber_conn_loss	dB
target link reach	L3	l_target	kilometer
starting link reach	L4	l_start	kilometer
link reach increment	L5	l_inc	kilometer
Deterministic jitter	G7	dj	picoseconds
Duty cycle distortion	G8	dcd_dj	picoseconds
Transmitter performance parameters			
transmitter OMA, minimum	C8	tx_oma_min	dBm
Transmitter extinction ratio, minimum	C9	er_min	dB
transmitter risetime (20%-80%)	T5	tx_2080_rise	picoseconds
laser wavelength, minimum	C6	lambda_min	nanometers
laser spectral width	C7	delta_lambda	nanometers
Transmitter reflectivity	G12	tx_reflection	dB
Residual intensity noise (RIN)	G4	rin	dB per Hertz
Eye opening of the RIN tester	AK7	rin_test_isi	dimensionless
Receiver bandwidth for Tx eye test	W5	txeye_rx_bw	megahertz
transmitter eye mask coordinate X1	C12	X1	UI (dimensionless)
transmitter eye mask coordinate X2	C13	X2	UI (dimensionless)
transmitter eye mask coordinate Y1	C14	Y1	OMA normalized
Receiver performance parameters			
Receiver unstressed sensitivity	T3	rx_sensitivity	dBm
Receiver bandwidth	T5	rx_bw	megahertz
Receiver reflectivity	T4	rx_reflection	dB
Fiber channel performance parameters			
fiber dispersion coefficient	P8	fiber_s0	ps/(nm squared * km)
wavelength of minimum fiber dispersion	P7	fiber_u0	nanometers
fiber length - modal bandwidth product	P12	fib_lbwp	megahertz-kilometer
Noise penalty parameters			
Modal noise power penalty	G13	pmn	dB
noise factor for interferometric reflections	L10	ref_nf	dimensionless
k parameter for mode partition noise	G10	kpmn	dimensionless
Baseline wander noise-to-signal ratio	T10	sigma_blw	dimensionless
starting eye plot time		eye_time_low	UI (dimensionless)
maximum eye plot time		eye_time_high	UI (dimensionless)
eye plot time increment	Y42	eye_time_step	UI (dimensionless)

9 Python script listings

9.1 Python GbE10 class listing

```
In [ ]: #=====================================================================+
        #                                                                     |
        #                     Start of file GbE10_support.py                  |
        #                                                                     |
        #=====================================================================+

        import numpy as np
        import math
        from scipy.special import erf, ndtri, erfinv
        from openpyxl import load_workbook

        class GbE10:

            def __init__(self,worksheet='MMF2000'):

                wb = load_workbook('10GbE_inputs.xlsx')
                ws = wb[worksheet]

                # link parameters
                self.br_nominal      = ws['C4'].value    # nominal baud rate
                self.ber_target      = ws['C5'].value    # allowed bit error rate limit
                self.fiber_conn_loss = ws['C6'].value    # connector loss

                self.l_target        = ws['C8'].value   # desired link reach
                self.l_start         = ws['C9'].value   # minimum link reach
                self.l_inc           = ws['C10'].value   # link reach increment

                # jitter
                self.dj              = ws['C13'].value    # deterministic jitter
                self.dcd_dj          = ws['C14'].value    # duty cycle distortion

                # transmitter performance parameters
                self.tx_oma_min      = ws['C17'].value   # minimum transmitter optical modulation ampli
                self.er_dB_min       = ws['C18'].value    # minimum required extinction ratio
                self.tx_2080_rise    = ws['C19'].value   # transmitter rise / fall time, 20%-80%
                self.lambda_min      = ws['C20'].value   # minimum VCSEL wavelength
                self.delta_lambda    = ws['C21'].value    # VCSEL spectral width
                self.tx_reflection   = ws['C22'].value   # transmitter reflectance
                self.rin             = ws['C23'].value   # laser residual intensity noise (RIN)
                self.rin_test_isi    = ws['C24'].value   # eye opening, RIN tester

                self.txeye_rx_bw     = ws['C26'].value   # receiver bandwidth for Tx eye tester

                self.X1              = ws['C28'].value   # transmitter eye mask, X1
                self.X2              = ws['C29'].value   # transmitter eye mask, X2
                self.Y1              = ws['C30'].value   # transmitter eye mask, Y1

                # receiver performance parameters
                self.rx_sensitivity  = ws['C33'].value   # receiver sensitivity
                self.rx_bw           = ws['C34'].value   # receiver bandwidth
```

```
        self.rx_reflection   = ws['C35'].value   # receiver reflectance

        # fiber channel performance parameters
        self.fiber_s0        = ws['C38'].value   # fiber dispersion
        self.fiber_u0        = ws['C39'].value   # zero dispersion wavelength for fiber
        self.fib_lbwp        = ws['C40'].value   # fiber length-modal bandwidth product

        # noise penalty parameters
        self.pmn             = ws['C43'].value   # power penalty for modal noise
        self.ref_nf          = ws['C44'].value   # noise factor for reflectance
        self.k_mpn           = ws['C45'].value   # k factor for mode partition noise
        self.sigma_blw       = ws['C46'].value   # baseline wander noise standard deviation

        self.eye_time_low    = ws['C48'].value   # minimum eye plot time (UI)
        self.eye_time_high   = ws['C49'].value   # maximum eye plot time
        self.eye_time_step   = ws['C50'].value   # eye plot time increment

        self.preliminary_calc()
        self.fiber_channel_calc()
        self.risetime_calc()
        self.isi_calc()
        self.penalty_calc()
        self.eye_calc()

        # end of GbE10.__init__

#========================================================================+

    def preliminary_calc(self):

        # calculate Q from target BER; see equation B.41 in FC-MSQS-2
        self.Q = -ndtri(self.ber_target)
        # see FC-MSQS-2 equation B.47 in Annex B.4 for the following...
        self.k_rin = math.sqrt(2.0/math.pi)*erfinv(0.8)

        # cell Y44
        self.speedup = 1.0 / (1.0 - 1.0E-6*self.br_nominal*self.dcd_dj)
        self.dj_ui = 1.0E-6*self.speedup*self.br_nominal*(self.dj-self.dcd_dj)    # cell G9

        # define the length vector, column A
        self.l_stop = 2.0*self.l_target - self.l_start
        self.lnum = 1+int(round((self.l_stop-self.l_start)/self.l_inc))
        self.length = np.linspace(self.l_start,self.l_stop,self.lnum)
        # convenient unit vector used in several calculations
        self.l_1 = np.ones(self.lnum)

#========================================================================+

    def fiber_channel_calc(self):

        """Calculates fiber attenuation, modal bandwidth, and
        chromatic dispersion bandwidth"""

        # calculate fiber attenuation at minimum laser wavelength
```

```python
        self.alpha = (1.05+(1.0/(0.00094*self.lambda_min))**4)

        # calculate channel insertion loss                          column C
        self.chil = self.alpha*self.length + self.fiber_conn_loss*self.l_1

        # calculate chromatic dispersion bandwidth
        self.d1 = (0.25*self.fiber_s0*self.lambda_min*(1.0-
                  (self.fiber_u0/self.lambda_min)**4))            # cell P9
        self.d2 = 0.7*self.fiber_s0*self.delta_lambda              # cell AB4
        dtot = math.sqrt(self.d1**2 + self.d2**2)
        self.bw_cd = (0.187E6/(self.delta_lambda*dtot)
                     /self.length)                                 # column F

        # calculate modal dispersion bandwidth for each link length
        self.bw_md = self.fib_lbwp / self.length                   # column G

        # end of GbE10.fiber_channel_calc

#=======================================================================+

    def risetime_calc(self):

        """Calculates the 10%-90% risetimes associated with the transmitter,
        the fiber channel (chromatic disperion, modal dispersion), the
        link receiver, and the reference receiver for transmitter eye
        measurements."""

        # given the transmitter's 20%-80% risetime, and assuming a
        # Gaussian impulse response, calculate the 10%-90% risetime
        # cell G3
        self.tx_1090_rise = 1.518*self.tx_2080_rise

        # calculate the effective risetimes for the fiber channel, given
        # the bandwidths calculated in the previous section, assuming
        # a Gaussian impulse response model
        self.cd_1090_rise = 0.48E6 / self.bw_cd
        self.md_1090_rise = 0.48E6 / self.bw_md

        # calculate the risetime for the link receiver, given its
        # bandwidth and assuming a single pole impulse response
        # Cell T7
        self.rx_1090_rise = 0.329E6/self.rx_bw

        # calculate the risetime for the test receiver used for transmitter
        # eye displays, given its bandwidth and assuming a single pole
        # response
        self.rx_txeye_1090_rise = 0.329E6 / self.txeye_rx_bw

        # calculate Te from column H and Tc from column I
        tr_tx_2 = self.tx_1090_rise**2*self.l_1
        tr_rx_2 = self.rx_1090_rise**2*self.l_1
        tr_cd_2 = np.square(self.cd_1090_rise)
        tr_md_2 = np.square(self.md_1090_rise)
        self.te = np.sqrt(tr_cd_2 + tr_md_2 + tr_tx_2)             # column H
```

```python
        self.tc = np.sqrt(tr_cd_2 + tr_md_2 + tr_tx_2 + tr_rx_2) # column I

        # end of GbE10..risetime_calc

#=======================================================================+

    def isi_calc(self):

        """Calculates the eye opening at the center and at the corners,
        needed for various penalty calculations"""

        arg1 = erfinv(0.8)
        arg2 = arg1/(1.0E-6*self.speedup*self.br_nominal)
        # calculate center eye opening with no additional impairments
        self.isi_center = erf(arg2/self.tc)                       # column Z

        # calculate center eye opening with residual DJ (DJ - DCD)
        # impairment                                              # column AD
        arg3 = erf(arg2*(1.0+self.dj_ui)/self.tc)                 # column AI
        arg4 = erf(arg2*(1.0-self.dj_ui)/self.tc)                 # column AJ
        self.isi_dj_center = arg3 + arg4 - self.l_1               # column AD

        # calculate eye closing induced by interferometric effects from
        # link end reflections
        mean_reflection = math.pow(10.0,0.05*(self.rx_reflection
                        + self.tx_reflection))                    # cell AB5
        er_lin = math.pow(10.0,0.1*self.er_dB_min)                   # cell AB7
        arg5 = np.sqrt(2.0*er_lin*self.isi_dj_center*(er_lin-1.0)
             + (er_lin+1.0)*self.l_1)
        arg6 = np.divide(arg5,self.isi_dj_center)
        arg7 = (2.0*self.ref_nf*np.power(
                10.0,-0.1*self.chil)*mean_reflection)
        self.isi_reflection = self.l_1-np.multiply(arg6,arg7)

        # calculate center eye opening with both residual DJ and reflection
        # degradations included
        self.isi_dj_closed = np.multiply(self.isi_dj_center,
                                         self.isi_reflection)      # column AA

        # calculate eye opening at the corners with no additional impairments
        eff_rx_eye = 2.0*(0.5-self.X2)*self.speedup
        arg8 = erf(arg2*(1.0+eff_rx_eye)/self.tc)                  # column AE
        arg9 = erf(arg2*(1.0-eff_rx_eye)/self.tc)                  # column AF
        self.isi_corners = arg8 + arg9 - self.l_1                  # column AB

        # calculate eye opening at the corners with residual DJ impairment
        arg10 = erf(arg2*(1.0+eff_rx_eye+self.dj_ui)/self.tc)      # column AG
        arg11 = erf(arg2*(1.0-eff_rx_eye-self.dj_ui)/self.tc)      # column AH
        self.isi_dj_corners = arg10 + arg11 - self.l_1                # column AC

        # end of GbE10.isi_calc

#=======================================================================+
```

```python
def penalty_calc(self):

    """Calculates the various link budget penalties"""

    self.p_budget = (self.tx_oma_min
                     - self.rx_sensitivity
                     - self.fiber_conn_loss)*self.l_1

    # fiber attenuation,
    self.p_atten = self.alpha*self.length                     # column B

    # calculate bandwidth for RIN test (exclude transmitter)
    rin_inverse_bw = np.sqrt(
                       np.square(1.0/self.bw_cd)
                     + np.square(1.0/self.bw_md)
                     + (0.477/(self.rx_bw**2))*self.l_1)
    rin_bw = 1.0 / rin_inverse_bw
    # v_rin,
    self.v_rin = (0.7E6*(self.rin_test_isi**2)*
                math.pow(10.0,0.1*self.rin)*rin_bw)          # column AK
    # Prin,
    self.p_rin = -10.0*np.log10(np.sqrt(1.0-np.multiply(self.v_rin,
                  np.square(self.Q/self.isi_dj_closed))))    # column R

    self.beta = (3.14159E-6*self.speedup*self.br_nominal
                *self.delta_lambda*self.d1*self.length)       # column O
    self.sigma_mpn = (self.k_mpn/math.sqrt(2.0)*(self.l_1
                      -np.exp(-np.square(self.beta))))        # column P
    self.p_mpn = (-10.0*np.log10(np.sqrt(1.0-
                  (self.Q**2)*np.square(self.sigma_mpn))))    # column Q

    self.p_blw = (-10.0*math.log10(math.sqrt(1.0-
                 (self.Q*self.sigma_blw)**2))*self.l_1)       # cell T13
    self.p_reflection = -10.0*np.log10(self.isi_reflection)  # column N

    self.v_mn = (((1.0-math.pow(10.0,-0.2*self.pmn))/
                 (self.Q)**2)*self.l_1)                       # cell AG7

    self.p_isi_center = (-10.0*np.log10(2.0*self.isi_center
                        - self.l_1))                          # column J
    self.p_isi_corners = (-10.0*np.log10(self.isi_corners)
                       - self.p_isi_center)                   # column K
    self.p_dj_center = (-10.0*np.log10(self.isi_dj_closed)
                     - self.p_isi_center)                     # column L
    self.p_dj_corners = (-10.0*np.log10(self.isi_dj_corners)
                      -self.p_isi_center
                      -self.p_isi_corners)                    # column M

    # calculate the "cross" penalty contribution, column S
    arg1 = (self.sigma_blw**2 + self.v_rin)/np.square(self.isi_dj_closed)
    arg2 = self.l_1 - (self.Q**2)*(arg1 + self.v_mn
           + np.square(self.sigma_mpn))
    arg3 = -10.0*np.log10(np.multiply(self.isi_dj_closed,np.sqrt(arg2)))
    self.p_cross_center = (                             # column S
```

```
            arg3
          - self.p_blw                                   # cell T13
          - self.p_isi_center                            # column J
          - self.p_dj_center                             # column L
          - self.p_mpn                                   # column Q
          - self.p_reflection                            # column N
          - self.p_rin                                   # column R
          - self.pmn*self.l_1)                           # cell G13

        # calculate the total power budget evaluated at the center of the eye
        self.p_total_center = (                          # column T
            self.p_isi_center                            # columns J
          + self.p_dj_center                             # column L
          + self.p_atten                                 # column B
          + self.p_mpn                                   # column Q
          + self.p_reflection                            # column N
          + self.p_rin                                   # column R
          + self.p_cross_center                          # column S
          + self.pmn*self.l_1)                           # cell G13

        # calculate the total power budget evaluated at the corner of the eye
        self.p_total_corners = (
            self.p_isi_center                            # column J
          + self.p_isi_corners                           # column K
          + self.p_atten                                 # column B
          + self.p_mpn                                   # column Q
          + self.p_reflection                            # column N
          + self.p_rin                                   # column R
          + self.p_cross_center                          # column S
          + self.pmn*self.l_1                            # cell G13
          + self.p_dj_corners)                           # column M

        # end of GbE10.penalty_calc

#=======================================================================+

    def eye_calc(self):

        """Calculates the eye diagrams for the link at target reach
        and for the transmitter at test"""

        # define the eye time vector scaled to UI, dimensionless
        tnum = (1+int(round((self.eye_time_high-self.eye_time_low)
               / self.eye_time_step)))
        self.time = np.linspace(self.eye_time_low,             # column Z
                                self.eye_time_high,
                                tnum)
        # convenience vector
        t_1 = np.ones(tnum)
                                                               # column AA
        self.time_eff = 0.5*t_1 + self.speedup*(self.time - 0.5*t_1)

        arg1 = 2.0*erfinv(0.8)
        arg2 = arg1/(1.0E-6*self.speedup*self.br_nominal)
```

```
        # calculate eye profiles for link at target reach
        T_c_link = self.tc[self.lnum/2]
        arg3 = arg2*self.time_eff/T_c_link                          # column AP
        arg4 = arg2*(1.0 - self.time_eff)/T_c_link                  # column AQ
        self.link_011 = 0.5*t_1 + 0.5*erf(arg3)                     # column AR
        self.link_110 = 0.5*t_1 + 0.5*erf(arg4)                     # column AS
        self.link_010 = self.link_011 + self.link_110 - t_1         # column AT
        self.link_101 = t_1 - self.link_010                         # column AW
        self.link_001 = t_1 - self.link_110                         # column AU
        self.link_100 = t_1 - self.link_011                         # column AV

        # calculate eye profiles for transmitter eye test configuration
        T_c_test = math.sqrt(self.tx_1090_rise**2
                 + (0.329E6/self.txeye_rx_bw)**2)
        arg5 = arg2*self.time_eff/T_c_test
        arg6 = arg2*(t_1 - self.time_eff)/T_c_test
        self.test_011 = 0.5*t_1 + 0.5*erf(arg5)                     # column AR
        self.test_110 = 0.5*t_1 + 0.5*erf(arg6)                     # column AS
        self.test_010 = self.test_011 + self.test_110 - t_1         # column AT
        self.test_101 = t_1 - self.test_010
        self.test_001 = t_1 - self.test_110
        self.test_100 = t_1 - self.test_011

        # end of GbE10.eye_calc

#=======================================================================+
#                                                                       |
#                       End of file Gb10E_support.py                    |
#                                                                       |
#=======================================================================+
```

9.2 Example of power penalty plotting script

```
In [ ]: # start of file Gpenalty_plot.py

        import matplotlib.pyplot as plt

        from GbE10_support import GbE10
        g = GbE10()
        plt.plot (g.length, g.p_budget,label='budget')
        plt.plot (g.length, g.p_total_central,label='P_total (central)')
        plt.plot (g.length, g.p_total_corners,label='P_total(corners)')
        plt.plot (g.length, g.p_isi_central,label='P_isi(central)')
        plt.plot (g.length, g.p_atten,label='P_atten')
        plt.plot (g.length, g.p_cross_central,label='P_cross(central)' )
        plt.legend(loc='upper left')

        plt.show()
```

9.3 Example of eye plotting script

```
In [ ]: # start of file Geye_plot.py
```

```python
import matplotlib.pyplot as plt

from GbE10_support import GbE10
g = GbE10()

plt.plot (g.time, g.link_001)
plt.plot (g.time, g.link_010)
plt.plot (g.time, g.link_011)
plt.plot (g.time, g.link_100)
plt.plot (g.time, g.link_101)
plt.plot (g.time, g.link_110)

plt.plot (g.time, g.test_010)
plt.plot (g.time, g.test_101)

plt.show()
```

10 Alphabetical list of Python GbE10_support object attributes

Python parameter	**Description**	
scalar or vector	**Spreadsheet label**	**Spreadsheet location**
alpha	fiber attenuation	
L vector	Patt	column B
beta	parameter in MPN calculation	
L vector	Beta	column O
bw_cd	bandwidth of chromatic dispersion	
L vector	BWcd	column F
bw_cd	bandwidth of chromatic dispersion	
L vector	BWcd	column F
bw_md	bandwidth of modal dispersion	
L vector	effBWm	column G
cd_1090_rise	risetime for chromatic dispersion	
L vector	—	used in column I
chil	fiber channel insertion loss	
L vector	Ch IL	column C
d1	fiber dispersion	
scalar	Disp. D1	cell P9
d2	fiber dispersion	
scalar	D2	cell AB4
dj_ui	deterministic jitter, with DCD subtracted	
scalar	Effect. DJ	cell G9
isi_center	eye opening at the center	
L vector	heye(0)	column Z
isi_corners	eye opening at the edge of the Tx mask	
L vector	ISI at corners ex jitter	column AB
isi_dj_center	eye opening at the center with DJ degradation	
L vector	ISI and DJ central	column AD
isi_dj_refl_closed	eye opening at the center with DJ and reflection degradation	
L vector	ISI, jitter, & Refl. closed eye	column AA
isi_dj_corners	eye opening at the edge of the Tx mask with dj degradation	
L vector	ISI, jitter & TP4 closed eye	column AC
isi_dj_refl_center	eye opening at the center with DJ and reflection degradations	
L vector	ISI, jitter, and TP4 closed eye	column AC
isi_reflection	eye opening with coherent reflection degradations	
L vector	—	used in column N
isi_tp4_rx	eye opening at mask edge	
L vector	ISI_TP4_Rx	cell AG5
k_rin	laser residual intensity noise (RIN) k factor	
scalar	RIN Coef	cell G6
length	link reach	
L vector	L	column A
link_001	eye profile for the link	
T vector	erf2'	column AV
link_010	eye profile for the link	
T vector	oio	column AT
link_011	eye profile for the link	
T vector	erf1	column AR
link_100	eye profile for the link	
T vector	erf1'	column AU

link_101	eye profile for the link	
T vector	ioi	column AW
link_110	eye profile for the link	
T vector	erf2	column AS
md_1090_rise	risetime (10%-90%) for chromatic dispersion	
L vector		used in column H
p_atten	power penalty due to fiber attenuation	
L vector	Patt	column B
p_blw	power penalty due to baseline wander (blw)	
scalar	P_BLW	cell T13
p_budget	link power budget	
L vector	Pwr.Bud-Conn.Loss	cell K8
p_cross_center	excess noise degradation power penalty	
L vector	Pcross central	column S
p_dj_center	center eye closing power penalty with DJ contribution	
L vector	P_DJ central	column L
p_dj_corners	eye closing at Tx mask edge with DJ contribution	
L vector	P_DJ corners	column M
p_isi_center	eye closing power penalty at the center of the eye	
L vector	Pisi central	column J
p_isi_corners	eye closing power penalty at the Tx mask corners	
L vector	ISI at eye corners ex jitter	column AB
p_mpn	power penalty due to mode partition noise (MPN)	
L vector	Pmnp	column Q
p_reflection	power penalty due to coherent reflections	
L vector	Preflection central	column N
p_rin	power penalty due to laser residual intensity noise (RIN)	
L vector	Prin	column R
p_total_center	sum of the power penalties at the center of the eye	
L vector	Ptotal central	column T
p_total_corners	sum of the power penalties at the Tx mask edge	
L vector	Ptotal corners	column U
Q	derived from target bit error rate (BER)	
scalar	Q	cell C3
rx_1090_rise	risetime (10%-90%) for the link receiver	
scalar	T_rx(10-90)	cell T7
rx_txeye_1090_rise	risetime (10%-90%) for the Tx eye test receiver	
scalar	T_test_rx(10-90)	cell AG8
sigma_mpn	noise-to-signal ratio for mode partition noise (MPN)	
L vector	SDmpn	column P
speedup	factor accounting for pulse width shrinkage	
scalar	speedup	cell Y44
tc	composite link response time (10%-90%)	
L vector	Tc	column I
te	composite link response time (10%-90%) excluding receiver	
L vector	Te	column H
test_001	eye profile for Tx mask test	
T vector	erf2'	column AH
test_010	eye profile for Tx mask test	
T vector	oio	column AF
test_011	eye profile for Tx mask test	
T vector	erf1	column AD

test_100	eye profile for Tx mask test	
T vector	erf1'	column AG
test_101	eye profile for Tx mask test	
T vector	ioi	column AI
test_110	eye profile for Tx mask test	
T vector	erf2	column AE
time	vector of UI intervals for eye profile calculations	
T vector	T	column Z
time_eff	time vector with speedup (pulse width shrinkage) factor	
T vector	Teff	column AA
tx_1090_rise	transmitter risetime (10%-90%)	
scalar	Ts(10-90)	cell F3
v_mn	noise variance for modal noise (MN)	
scalar	Vmn	cell AG7
v_rin	noise variance for laser residual intensity noise (RIN)	
scalar	V_rin(2m test)	cell AG6