

	Document #	Date effective October 23, 2006
	Author(s) Ryan Herbst	Supersedes
	Draft Revision 0.02 – January 12, 2007	
Document Title Pretty Good Protocol - Design Specification		

Hard copies of this document are for REFERENCE ONLY and should not be considered the latest revision.

Pretty Good Protocol - Design Specification

CHANGE HISTORY LOG

Revision	Effective Date	Description of Changes
0.01	10/23/2006	Initial Version
0.02	01/12/2007	Removed electrical IDLE to allow fiber use, added PIB state output signals.

Table of Contents

1. Introduction.....	5
2. External Interfaces	7
2.1 Clock & Reset.....	7
2.2 Configuration Signals	7
2.3 Frame Transmit Interface.....	7
2.4 Frame Receive Interface	10
2.5 Transmit CRC Interface	11
2.6 Receive CRC Interface	11
2.7 Physical Interface Signals.....	12
2.8 Event Counter & Status Signals.....	14
3. Operation.....	16
3.1 Physical Interface.....	17
3.1.1 Ordered Sets.....	18
3.1.2 Link Initialization.....	20
3.2 Cell Transmission	24
3.3 Cell Reception.....	26
3.4 Transmission Scheduler	26
3.5 ACK/NACK Timer Logic.....	27
3.6 Receive Tracking Block.....	28
3.7 ACK/NACK Transmit Logic.....	29

Pretty Good Protocol - Design Specification

3.8 Random Data Generator 29

3.9 Error Count 29

4. Known Limitations 30

1.

INTRODUCTION

The Pretty Good Protocol (PGP) is a VHDL module which facilitates the bi-directional transmission of frame based messages over a two-wire physical link with a data rate of 2.5Gbs. The following features are supported by the PGP module:

- Multiple physical layer (PHY) device support
 - PCI Express compliant PIPE interface
 - Xilinx Multi Gigabit Transceiver (MGT) support
- Physical Layer Training
 - Automatic receive signal inversion
 - Endpoint version detection
- 4 individual virtual channels (VC)
 - Equal allocation of link bandwidth
 - Per VC indication of remote FIFO full & FIFO almost full status
- Flexible frame transmission
 - Unlimited frame size
 - 16-bit data interface supporting odd number of bytes in transferred frame
 - Frame transmission can be paused by user logic
- Acknowledgement of remote frame reception
 - 256 in flight frames supported
 - 32-bit context ID for local user logic tracking of frame acknowledgment
 - Up to 524uS timeout to detect missing acknowledgement or frame corruption

The basic operation of the PGP is to transport frames from one end of a link to the other. Frames are

provided by external user logic over a 16-bit FIFO like interface. Frames with an odd number of bytes are handled by allowing the user to mask the unused portion of the 16-bit value passed at the end of the frame. The user logic is allowed to pass frames of unlimited size and can pause frame transmission at any point for an arbitrary amount of time.

Frames received from the user interface are then broken into individual cells with a max payload size of 512Bytes. Information within the cell will identify the location of the start and end of the frame transmitted. Transmission of cells from the four virtual channels will be interleaved in order to ensure equal allocation of bandwidth between virtual channels.

In addition to transporting frames across a link the PGP also transports two buffer threshold bits for each of the four virtual channels supported. The use of these two bits is entirely up to the user logic.

Frames transported across the link are tracked using 256 unique sequence numbers. When the user logic passes a frame to the PGP it will pass a 32-bit context value which can be any arbitrary value which is useful the use logic for frame tracking. Each newly transmitted frame is assigned a sequence number which is then associated with the context value passed from the user logic.

The remote end of the link will acknowledge the successful completion of frames by sending ACK/NACK updates to the transmitter logic. When an acknowledgement is received from the remote side of the link the stored context ID will be passed back to the user. If the frame is corrupted for any reason and/or the acknowledgement is lost the local logic will timeout waiting for an acknowledgement and indicate this timeout condition to the user logic.

2. EXTERNAL INTERFACES

2.1 Clock & Reset

The following table describes the clock and reset signals used by the PGP.

Signal	Width	Direction	Description
pgpClock	1	Input	Master 125Mhz clock. This clock is used by all internal logic and also serves as the clock to which all interface signals are synchronized.
pgpReset	1	Input	Master PGP reset. This reset shall be synchronized to the pgpClock and asserted for at least 2 clock periods. Forces all internal logic to be reset and the physical interface to be re-linked.
pibReLink	1	Input	Asserted for a single clock period to force the physical interface logic to attempt to re-link to the remote end. If the link is currently healthy the link will be terminated and the training sequence will be started.

Clock & Reset Signals

2.2 Configuration Signals

The following signals define the operation of the PGP. These signals are sampled by the core at reset.

Signal	Width	Direction	Description
pibMode	1	Input	Sets the PHY mode of the PGP core. '0' = PIPE, '1' = MGT.
ackTimeout[15:0]	1	Input	Determines the number of micro seconds to wait before declaring a timeout on the reception of an ACK/NACK status updated from the remote end of the link.

Configuration Signals

2.3 Frame Transmit Interface

The following table identifies the signals which are used by the user logic to pass transmit frames to the PGP. Each of the four virtual channels has its own set of signals. The N value in the following list is replaced with the virtual channel number, 0-3.

Pretty Good Protocol - Design Specification

Signal	Width	Direction	Description
vcNFrameTxValid	1	Input	Signal from user logic indicating data is ready on the input signals. Data is transferred when this signal is asserted at the same time as vcNFrameTxReady.
vcNFrameTxReady	1	Output	Signal from PGP indicating it is ready to accept data from the user logic. Data is transferred when this signal is asserted at the same time as vcNFrameTxValid.
vcNFrameTxSOF	1	Input	Signal asserted by user logic to indicate the passed data is the first of a frame.
vcNFrameTxWidth	1	Input	Signal asserted by the user logic to indicate the passed data is 16-bits wide. Must always be '1' except for when EOF asserted to indicate the last data in the frame.
vcNFrameTxEOF	1	Input	Signal asserted by user logic to indicate the passed data is the last of the frame.
vcNFrameTxEOFE	1	Input	Signal asserted with EOF when the user logic wishes to indicate that an error exists within the passed frame data.
vcNFrameTxData[15:0]	16	Input	Frame data passed by the user logic for transmission.
vcNFrameTxCid[31:0]	32	Input	Arbitrary 32-bit context data associated with the transmitted frame. This value is stored by the PGP and passed back to the user logic when the frame is ACKed or NACKed. The CID values must be held on this bus for the entire transfer of the frame.
vcNFrameTxAckCid[31:0]	32	Output	Arbitrary 32-bit context data associated with frame being ACKed or NACKed.
vcNFrameTxAckEn	1	Output	Signal asserted to indicate the ACK/NACK of a transmitted frame. User logic should sample AckCid[31:0] & TxAck when this signal is asserted.
vcNFrameTxAck	1	Output	Asserted high with AckEn to indicate frame ACK, asserted low with AckEn to indicate frame NACK.
vcNRemBuffAFull	1	Output	Remote receive buffer almost full buffer status.
vcNRemBuffFull	1	Output	Remote receive buffer full status indication.

Frame Transmit Signals

The transmit interface between the user logic and the PGP is frame based with the user logic passing framed data for transmission. This interface consists of a 16-bit wide data bus with signals to indicate start of frame (vcNTxFrameSOF), end of frame (vcNTxFrameEOF) and end of frame with error (vcTxFrameEOFE). The user can transmit an odd number of bytes in a frame by using the vcNTxFrameWidth signal to indicate that the last word transmitted only contains a single byte in bits 7-0. In all cases the byte contained in bits 7-0 of the interface bus is transmitted first.

Handshaking between the user logic and the PGP consists of two signals, `vcNTxFrameValid` asserted by the user logic & the `vcNTxFrameReady` signal asserted by the PGP. The valid signal is asserted by the user logic when it is ready to transfer a 16-bit word to the PGP. The ready signal is asserted by the PGP logic when it is ready to accept a 16-bit word from the user logic. Frame data is transferred between the two blocks when valid & ready are asserted at the same time. The user logic must be able to deal with the PGP de-asserting ready at any point and for arbitrary periods of time. Similarly the PGP must accept that the user logic can de-assert valid at any time.

In order to ensure the proper delivery of frames to the remote side the user logic must be aware of the status of the receive buffers at the remote end. Two buffer status signals are transferred between the two link endpoints, allowing two buffer threshold values to be communicated. The `vcNRemBuffAFull` almost full signal indicates that the lower of two buffer threshold values has been exceeded on the remote end while the `vcNRemBuffFull` signal indicates that the upper of the two buffer threshold levels has been exceeded. The PGP logic simply transfers this status between the two ends of the link allowing the user logic to decide how it will react to the remote buffer status. It can pause the current transfer, end the frame early or continue the current transfer. The PGP will accommodate the user logic regardless of the current state of the remote buffer.

As each frame is successfully transferred to the remote end of the link its successful transmission will be indicated by the PGP through the use of an ACK/NACK mechanism. As each frame is accepted from the user logic the PGP will also accept a 32-bit context id (CID) associated with the transmitted frame. This value is stored by the PGP for use when indicating an ACK or NACK to the user logic. The PGP will indicate the ACK/NACK state of the transmitted frame by asserting the `vcNTxAckEn` signal high for one clock. While this signal is asserted the user logic will be passed the CID associated with the frame on the `vcNFrameTxAckCID` bus. If the frame is being ACKed then the `vcNFrameTxAck` signal will be asserted high. This same signal will be held low if the frame is being NACKed. Frames which are passed to the PGP with an EOFE assertion will always be NACKed. More information about the ACK/NACK mechanism of the PGP is included later.

2.4 Frame Receive Interface

The following table identifies the signals which are used by the user logic to receive frames from the PGP. Similar to the transmit interface, each of the four virtual channels has its own set of signals. The N value in the following list is replaced with the virtual channel number, 0-3.

Signal	Width	Direction	Description
vcNFrameRxValid	1	Input	Signal to user logic indicating data is ready on the output signals. The user logic has no mechanism to pause the PGP.
vcNFrameRxSOF	1	Input	Signal asserted by the PGP to indicate the passed data is the first of a frame.
vcNFrameRxWidth	1	Input	Signal asserted by the PGP to indicate the passed data is 16-bits wide. Will always be '1' except for when EOF asserted to indicate the last data in the frame.
vcNFrameRxEOF	1	Input	Signal asserted by the PGP to indicate the passed data is the last of the frame.
vcNFrameRxEOFE	1	Input	Signal asserted with EOF when a frame has been received with a known error. This known error is either detected within the PGP or indicated by the user logic on the remote end of the link.
vcNFrameRxData[15:0]	16	Input	Receive frame data passed by the PGP.
vcNLocBuffAFull	1	Output	Local receive buffer almost full buffer status.
vcNLocBuffFull	1	Output	Local receive buffer full status indication.

Frame Receive Signals

Similar to the transmit interface, the receive interface between the user logic and the PGP is frame based. This interface consists of a 16-bit wide data bus with signals to indicate start of frame (vcNRxFrameSOF), end of frame (vcNRxFrameEOF) and end of frame with error (vcTxFrameEOFE). Since the PGP allows an odd number of bytes to be transferred, the vcNRxFrameWidth signal to indicate that the last word transmitted only contains a single byte in bits 7-0. In all cases the byte contained in bits 7-0 of the interface has been received first.

Unlike the transmit direction where either the PGP or the user logic can pause frame transfer, the user logic must accept all data received by the PGP. The vcNRxFrameValid signal is asserted by the PGP whenever data is being transferred from the PGP to the user logic. The user logic must be able to handle pauses in the stream of receive data from the PGP.

The only mechanism the user logic has to keep it's receive buffers from overflowing are the two buffer status signals transferred across the link to the transmitting logic. The vcNLocBuffAFull almost full signal indicates that the lower of two buffer threshold values has been exceeded while the vcNLocBuffFull signal indicates that the upper of the two buffer threshold levels has been exceeded. The PGP has no mechanism to pause data transfer based upon the status of either of these signals. The signal states are simply passed on to the user logic on the transmitting end of the link.

2.5 Transmit CRC Interface

The first of two external CRC engine required by the PGP is the transmit CRC engine. This engine is used by the PGP to add a CRC to the cells being transferred to the remote end of the link. An external CRC engine is used to allow the design to take advantage of cores which may exist in the implementation target.

The following signals make up the Transmit CRC Interface.

Signal	Width	Direction	Description
crcTxIn[15:0]	16	Output	Data being passed to the CRC engine by the PGP.
crcTxInit	1	Output	Signal to indicate the data passed is the first set of data in a CRC envelope.
crcTxValid	1	Output	Signal to indicate to the CRC engine that it should update the CRC on this clock.
crcTxWidth	1	Output	Signal to indicate which portion of the passed data should be used for CRC generation. Set to '1' to indicate 16-bits, set to '0' to indicate 8-bits (bits 7-0).
crcTxOut[31:0]	32	Input	32-bit CRC value generated by the CRC logic. This value is sampled 4 clocks after the last data value is passed to the CRC engine.

Transmit CRC Interface

2.6 Receive CRC Interface

The second of two external CRC engine required by the PGP is the receive CRC engine. This engine is used by the PGP to check the CRC value of cells received from the remote end of the link. An external CRC engine is used to allow the design to take advantage of cores which may exist in the implementation target.

The following signals make up the Receive CRC Interface.

Signal	Width	Direction	Description
crcRxIn[15:0]	16	Output	Data being passed to the CRC engine by the PGP.
crcRxInit	1	Output	Signal to indicate the data passed is the first set of data in a CRC envelope.
crcRxValid	1	Output	Signal to indicate to the CRC engine that it should update the CRC on this clock.
crcRxWidth	1	Output	Signal to indicate which portion of the passed data should be used for CRC generation. Set to '1' to indicate 16-bits, set to '0' to indicate 8-bits (bits 7-0).
crcRxOut[31:0]	32	Input	32-bit CRC value generated by the CRC logic. This value is sampled 4 clocks after the last data value is passed to the CRC engine.

Receive CRC Interface

2.7 Physical Interface Signals

In order to allow the PGP to be instantiated in multiple FPGA types, the core will support two different physical interface devices. When used in a Spartan3 FPGA device an external PHY which supports the PIPE interface standard will be used. When used in a more advanced FPGA, such as a Virtex4, the built in Multi Gigabit Transceiver will be used.

Although the two PHY interfaces are similar, both supporting 16-bit transmit and receive data, some signals related to control & status monitoring are unique to the individual device. The following table defines the signals used to connect to the external PHY. Common signals which are shared by the two PHY devices start with the prefix 'phy' while signals specific to a particular PHY start with either the prefix 'mgt' or 'pipe'. Input signals in the group not being used are ignored by the PGP logic and will be tied to '0'.

Signal	Direction	Description
phyTxDispNeg	Output	Asserted high by the PGP to set the running disparity for the two characters being transmitted to negative.
phyTxIdle	Output	Asserted high by the PGP to force electrical idle to be transmitted on the external TXP & TXN lines.
phyRxIdle	Input	Asserted high by the PHY to indicated electrical idle is being received on the external RXP

Pretty Good Protocol - Design Specification

		& RXN lines.
phyRxPolarity	Output	Asserted high by the PGP to invert the data received on the external RXP & RXN lines.
phyRxData[15:0]	Input	Parallel data or K character received by the PHY.
phyRxDataK[1:0]	Input	Indicates that the received data is a K character. Bit 1 corresponds to phyRxData[15:8].
phyTxData[15:0]	Output	Parallel data or K character to be transmitted by the PHY.
phyTxDataK[1:0]	Output	Indicates that the data to be transmitted is a K character. Bit 1 corresponds to phyTxData[15:8].
pipeResetL	Output	Asserted low by the PGP to reset the PIPE PHY.
pipeTxStatus[2:0]	Input	Receive status indication from the PIPE PHY. Decoded as: 000 = Received data OK 001 = 1 SKP Added 010 = 1 SKP Removed 011 = Receiver detected 100 = 8B/10B decode error 101 = Elastic buffer overflow 110 = Elastic buffer underflow 111 = Receiver disparity error
pipeStatus	Input	Handshake signal from the PIPE PHY indicating that requested functions are complete. Used during reset and receiver detection by the PGP.
pipeRxValid	Input	Asserted high by the PIPE PHY to indicate valid data received on the RXD & RXN pins.
pipeTxDet	Output	Asserted high to request receiver detection by the PIPE PHY.
pipePd[1:0]	Output	Used to control the power state of the PIPE PHY.
mgtRxLock	Input	Indication from the MGT PHY that the receiver has locked to its reference clock and is ready to receive data.
mgtTxLock	Input	Indication from the MGT PHY that the transmitter has locked to its reference clock and is ready to transmit data.
mgtRxPmaReset	Output	Asserted by the PGP to reset the receiver PMA logic of the MGT PHY.
mgtTxPmaReset	Output	Asserted by the PGP to reset the transmitter PMA logic of the MGT PHY.
mgtRxReset	Output	Asserted by the PGP to reset the receiver logic of the MGT PHY.
mgtTxReset	Output	Asserted by the PGP to reset the transmitter logic of the MGT PHY.
mgtRxBuffError	Input	Asserted by the MGT PHY to indicate that a receiver buffer under run or overrun has occurred.

Hard copies of this document are for REFERENCE ONLY and should not be considered the latest revision.

mgtTxBuffError	Input	Asserted by the MGT PHY to indicate that a transmitter buffer under run or overrun has occurred.
mgtRxDispErr[1:0]	Input	Asserted by the MGT PHY to indicate a receiver running disparity error has occurred. Bit 1 corresponds to data bits 15:8.
mgtRxDecErr[1:0]	Input	Asserted by the MGT PHY to indicate an 8B/10B decode error in received data. Bit 1 corresponds to data bits 15:8.

Physical Interface Device Signals

2.8 Event Counter & Status Signals

No internal event counters are maintained within the PGP. It is left up to the user logic to determine which values it wishes to count, the amount of events it will count and how the event counters will be reset. The PGP will simply provide count increment signals which will be asserted for a single clock pulse to indicate that the associated counter should be incremented.

In addition to the event counter signals the PGP will provide a number of status signals which can be used to track the state of the core. The following table describes the event counter & status signals provided by the PGP.

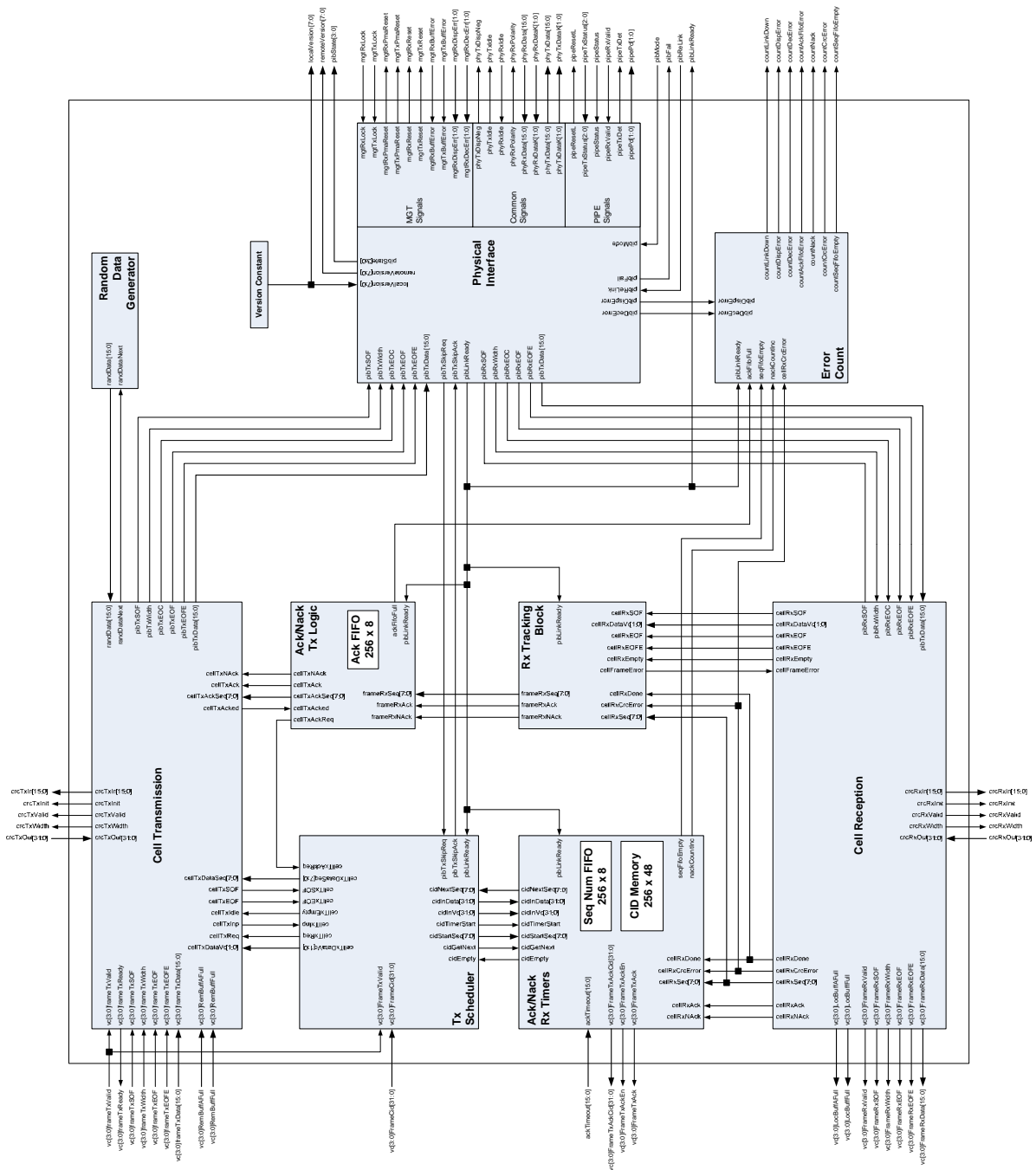
Signal	Width	Direction	Description
localVersion[7:0]	8	Output	8-bit version ID of the local PGP core.
remoteVersion[7:0]	8	Output	8-bit version ID reported by the remote PGP core.
pibState[3:0]	4	Output	4-Bit PIB state. See 3.1.2.3.
pibFail	1	Output	Signal to indicate that the physical interface logic was unable to create a link to the remote end of the link.
pibLinkReady	1	Output	Signal to indicate that the link to remote PGP core is healthy and ready to transfer data.
countLinkDown	1	Output	Signal asserted for a single clock to indicate that a link down event has occurred. This can be caused by the local user logic requesting a re-link, the remote end of the link requesting a re-link or a loss of the link to the remote end.
countDispError	1	Output	Asserted for a single clock to indicate that a 8B/10B running disparity error has been detected by the receiver.
countDecError	1	Output	Asserted for a single clock to indicate that a 8B/10B decode error has been detected

Pretty Good Protocol - Design Specification

			by the receiver.
countAckFifoError	1	Output	Asserted for a single clock to indicate that the local ACK/NACK transmit FIFO is full possibly resulting in the failure to transmit of ACK/NACK symbols.
countNack	1	Output	Asserted for a single clock to indicate that a NACK has been passed to the user logic, either due to a NACK received from the remote end or a lock ACK timeout.
countCrcError	1	Output	Asserted for a single clock to indicate that a CRC error has been detected by the cell receiver logic.
countSeqFifoEmpty	1	Output	Asserted for a single clock to indicate that the sequence number FIFO is empty and frame transmission has been halted.

Counter / Status Signals

3. OPERATION



PGP Block Diagram

Hard copies of this document are for REFERENCE ONLY and should not be considered the latest revision.

The above diagram shows the interconnection of the PGP functional blocks which are described in the following text.

3.1 Physical Interface

The physical interface logic acts as an interface between the physical device and the rest of the PGP logic. It is responsible for initialization of the physical interface device as well as initialization the link to the remote PGP.

Before going into details of the operation of the physical interface block it is important to understand the low level format of data being sent across the link. All bytes transferred between the two ends of the link are converted from 8-byte values into 10-byte values with special properties. This 8B/10B has a few functions:

- Adding special alignment characters which allows the physical interface to find bytes in a serial stream of bits
- Add additional characters for link alignment and maintenance overhead
- Enforce a balanced DC level on the link by maximizing the number of 0-1 & 1-0 transitions on the link

The pool of 8B/10B characters available is organized into two sets. The first set consists of K characters which have special properties and are used for special purposes. These characters can be decoded even if the 1's and 0's received on the physical interface are inverted. The remaining characters are the data characters which make up the 255 values which the user logic will transfer.

The following table described the K characters which are used by the PGP logic:

Symbol	Encoding	Description
COM	K28.5	Command character. Used to align the serial stream into bytes as well as identify the start of an ordered set.
SKP	K28.0	Special character which is added and removed from a Clock Compensation ordered set to adjust for differences between the transmit and receiver clock rates.

LTS	K28.1	Special character which is used in a link training ordered set.
IDL	K28.3	Character sent during an electrical IDLE state.
VTS	K23.7	Character used to identify a version handshake ordered set.
SOC	K27.7	Character used to identify the start of a cell on the link.
EOC	K28.2	Character used to identify the end of a cell on the link.
EOF	K29.7	Character used to identify the end of a cell on the link where the cell is also the last cell of a frame.
EOFE	K30.7	Character used to identify the end of a cell on the link where the cell is also the last cell of an errored frame.

K Characters

The symbols described in the above table are used to form ordered sets as well as cells which are transmitted across the link.

3.1.1 Ordered Sets

Ordered sets are special arrangements of K characters and data characters which are used to special purposes. These ordered sets are used for IDLE state, link training, version handshaking and clock compensation.

IDLE Ordered Set

Comma, K28.5
IDL, K28.3
IDL, K28.3
IDL, K28.3

IDLE Ordered Set

The IDLE ordered set is sent by the physical interface block when it is coming out of reset. It is also used to force the remote end of the link to re-initialize its physical interface.

Comma, K28.5	Comma, K28.5
LTS, K28.1	LTS, K28.1
D10.2	D21.5
LTS, K28.1	LTS, K28.1

Non-Inverted Inverted

Link Training Ordered Set

The link training ordered set shown above is transmitted during link initialization to detect if the POS & NEG signals connected to the receiver are inverted. Since the K characters are the same when inverted they will always be seen regardless of the state of the receive signals. The data character, D10.2, transmitted in this ordered set will be received as expected when the link is non-inverted or as a D21.5 character when the link is inverted.

The clock compensation ordered set is transmitted periodically by the physical interface block. This ordered set is used by the physical interface device to compensate for clock differences between the transmitter and the receiver. In cases where the receiver's clock is slower than the transmitter's clock the SKP K characters in this ordered set will be removed as necessary. In the opposite case SKP characters will be added as needed. Due to the addition and removal of SKP characters the receiver logic must be able to deal with variations in the length of this ordered set. The clock compensation ordered set is shown below.

Comma, K28.5
SKP, K28.0
SKP, K28.0
SKP, K28.0

Clock Compensation Ordered Set

The final ordered set used is the version training ordered set. This ordered set is used to ensure that the two ends of the link are operating the same version of the PGP. If the two end versions do not match the link initialization will fail.

Comma, K28.5
VTS, K23.7
Version ID
VTS, K23.7

Version Training Ordered Set

3.1.2 Link Initialization

The PGP physical interface logic is responsible to initializing the phy device when the PGP logic is reset either at power up or when a link init is requested by external logic. The reset operation performed depends on the state of the phyMode configuration signal sampled at reset.

3.1.2.1 PIPE Interface Initialization

The PIPE physical layer device has a very straightforward initialization sequence. In order to reset the device the pipeResetL signal is held low to reset all of its internal logic. While resetting the device the PGP logic will drive additional signals to ensure that the PIPE device comes up in the proper state. The following sequence will be followed:

1. Force reset by driving appropriate signals:
 - Assert and hold pipeResetL low
 - Assert and hold phyTxIdle high forcing the physical interface into electrical idle.
 - Set the devices power state to P1 by setting pipePd[1:0] to '10'.
2. Wait for the physical device to acknowledge reset by driving pipeStatus high
3. De-assert the pipeResetL signal.
4. Wait for the physical device to complete its initialization. When complete the device will drive pipeStatus low.
5. Transition to state P0 by setting pipePd[1:0] to '00'.

Upon completion of this operation the physical device is now in power state P0, its normal operating mode, and it's transmit port is in electrical idle state. The device is now ready to complete its initialization by handshaking with the remote transmitter.

3.1.2.2 MGT Interface Initialization

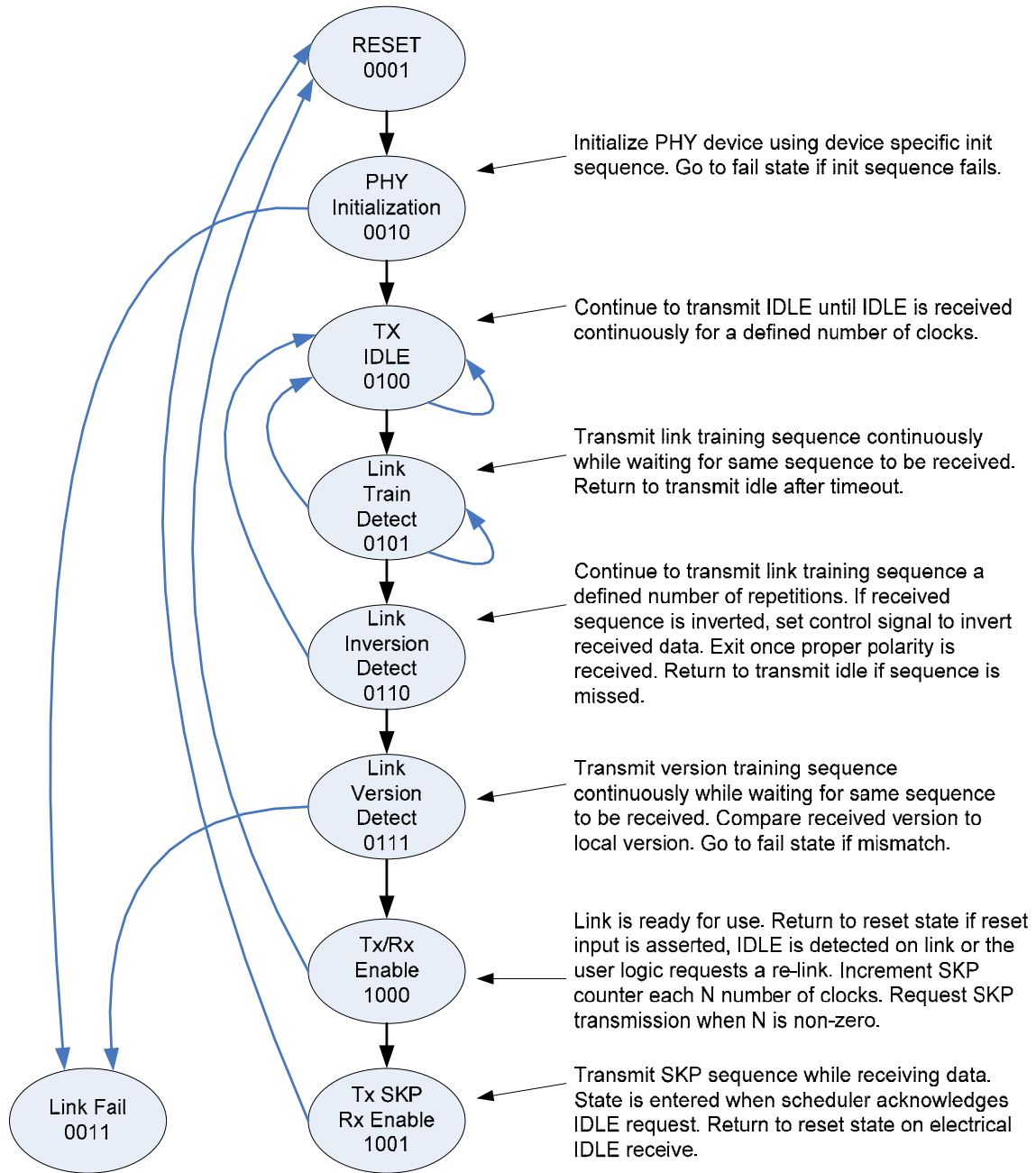
In order to initialize the MGT phy device a specific pattern of signal assertion and de-assertion must be followed. The signals driven during this process include mgtRxPmaReset, mgtTxPmaReset, mgtRxReset and mgtTxReset. The following sequence is followed in order to reset the MGT phy device:

1. Assert mgtTxPmaReset & mgtRxPmaReset high.
2. Wait 4 clock cycles.
3. De-Assert mgtTxPmaReset & mgtRxPmaReset low.
4. Wait for mgtTxLock & mgtRxLock to be asserted high.
5. Assert mgtTxReset & mgtRxReset high.
6. Wait 4 clock cycles.
7. De-Assert mgtTxReset & mgtRxReset low.
8. Wait 8 clock cycles.
9. Assert phyTxIdle high.

Upon completion of this operation the physical device is now in its normal operating mode, and it's transmit port is in electrical idle state. The device is now ready to complete its initialization by handshaking with the remote transmitter.

3.1.2.3 Link State Machine

The following diagram describes the operation of the physical interface logic.



Link State Machine

The state machine begins in the reset state. This state is entered during power up, when external logic requests an re-link or when the remote end of the link is requesting a re-link by transmitting

IDLE. Upon exiting reset the physical interface device will be re-initialized using the sequence specific to the attached device. If the initialization sequence fails then the state machine will go to the link fail state. The physical interface logic will then transmit until it receives the same sequence for a defined number of clocks. SKP ordered sets will be transmitted in this state as required.

Once the exit condition has been met the physical interface logic will then go into the next state where it will transmit the link training sequence. Once it has received the same sequence it will enter the next state where it will determine if the link is inverted. If the link training sequence has not been received for a defined number of clocks the state machine will return to the IDLE state and start to transmit IDLE again. SKP ordered sets will be transmitted in this state as required.

In the inversion detection state the state machine will determine if the receive signals on the phy device are receiving an inverted signal. If the link is inverted the receive signals will be swapped using a control signal to the phy device. The link training sequence will be transmitted a defined number of times in this state to ensure that the remote end has received enough sequences to properly detect an inverted signal. The state machine will return to IDLE if a sequence is received other than the link training or version handshaking sequences are detected. SKP ordered sets will be transmitted in this state as required.

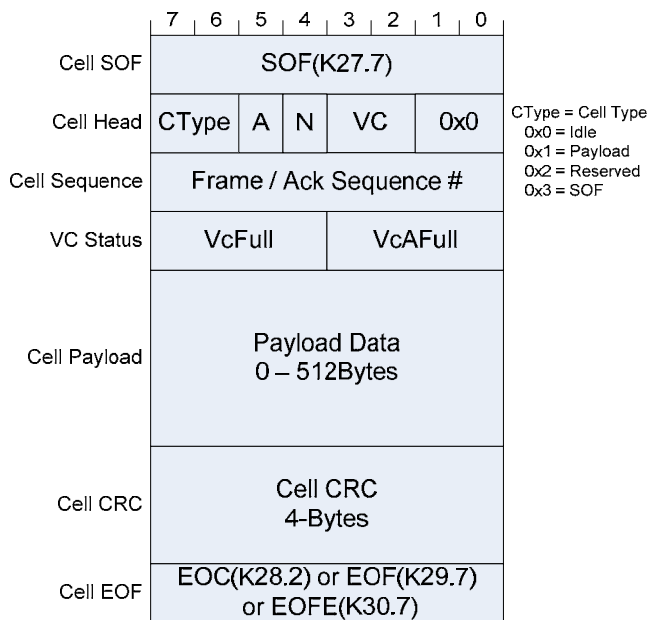
In the next state the link state machine will transmit a version handshaking sequence to the remote side of the link. This sequence will be transmitted for a defined number or repetitions. If the received version does not match the local version then the state machine will enter the link fail state. Upon receiving the proper version and having transmitted the defined number of sequences the state machine will then enter its normal state. SKP ordered sets will be transmitted in this state as required.

In the normal TX/RX state the physical interface logic will transmit cells from the cell transmission logic and pass received cells to the cell reception logic. During this operation the state machine will keep track of the number of clock periods that have passed since the last SKP sequence had been sent. Once a defined number of clocks have been passed the state machine will increment the SKP transmit counter to keep track of the number of SKP sequences that should be transmitted. Once this

SKP transmit counter is non-zero the link state machine will assert a SKP request to the TX scheduler logic. The scheduler will acknowledge this request once the current cell has completed its transmission. The link state machine will then enter the SKP TX state where it will send the required number of SKP sequences to the remote end. Once it is done transmitting its sequences the request line will be de-asserted allowing cell transmission to continue. During the SKP transmission state the physical interface block will continue to receive cells.

3.2 Cell Transmission

The cell transmission block is responsible for receiving framed data from the user logic and segmenting the data into cells for transmission on the physical link. The structure of the cell transmitted on the physical link is described in the following diagram.



Cell Structure

The cell is started with a special SOF character which identifies the start of the cell. The next byte contains information about the data which is contained in the cell including the CType field which defines the type of cell. Supported values for this field are Idle, Payload & SOF. The ‘A’ & ‘N’ fields are used for ACK & NACK updates to the remote end of the link. The VC field defines which

VC this data contained within the cell is associated with. The Frame / Ack Sequence number contains either the sequence number of the current frame or the sequence number which is being ACKed or NACKed. When the cell being transmitted is an SOF type this field contains the sequence number of the frame being transmitted, otherwise it contains the sequence number being ACKed or NACKed. The 'VcFull' and 'VCAFull' fields contain buffer status updates. The following data is between 0 & 512 bytes of payload data being transmitted within the cell. The size of the payload is determined by the location of the EOC, EOF, or EOF character. The 4-bytes before this character is the 32-bit CRC value for all data in the cell between the SOF and EOC, EOF or EOF character.

The following table defines the types of cells and allowed field combinations for the cells.

CType Field	A Field	N Field	VC Field	Seq #	Description
0x0 = IDLE	0	0	0x0	0x00	IDLE frame no ACK / NACK update.
0x0 = IDLE	1	0	0x0	ACK	IDLE frame with ACK update.
0x0 = IDLE	0	1	0x0	NACK	IDLE frame with NACK update.
0x1 = Payload	0	0	Data VC	0x00	Payload with no ACK / NACK update.
0x1 = Payload	1	0	Data VC	ACK	Payload with ACK update.
0x1 = Payload	0	1	Data VC	NACK	Payload with NACK update.
0x3 = SOF	0	0	Data VC	Frame	Start of frame cell.

Allowed CELL Field Combinations

The cell transmission logic is directed what to do by the transmission scheduler. The scheduler will direct the transmission logic as to whether to transmit an IDLE cell or a payload cell for a particular VC. The cell transmission logic will detect if the payload cell should be converted to an SOF and/or an EOF cell based upon the SOF/EOF signals from the user logic. For payload or IDLE cells the cell transmission logic will look at the state of the ACK/NACK signals to determine if an ACK/NACK update should be sent, indicating the result of its decision to the ACK/NACK logic. When an SOF cell is being transmitted the cell transmission logic will ignore the state of the signals from the ACK/NACK transmission logic. When transmitting IDLE cells, the payload data for the cell will be taken from a random number generation block which will generate a scrambled sequence of data to

transmit. This data will be ignored by the logic on the remote end.

3.3 Cell Reception

The cell reception block is responsible for receiving cells from the physical interface and passing the payload data to the user logic in a framed format. As each cell is received its headers are examined to determine if the data is associated with a VC or if the cell is IDLE. If the data is associated with a VC it will be passed on to the user logic accordingly.

As each cell is received information about the cell is passed to a receiver tracking block. This block tracks the state of each VC based upon the SOF, EOC, EOF & EOFE signals contained within the cell. Similarly if the cell contains an ACK or NACK update the appropriate information is passed to the ACK/NACK timer logic.

As each cell is received its CRC is calculated and compared to the value contained at the end of the cell. If there is an error the cell is dropped and all data contained within the cell is ignored.

3.4 Transmission Scheduler

The role of the transmission scheduler is to determine what data is being transmitted at any given time. The transmission scheduler receives the valid signal from each of the four VC interfaces, a SKP request signal from the physical interface logic and a request signal from the ACK/NACK Transmission Logic. The scheduler uses these signals to determine what should be transmitted by the cell transmission logic.

During normal transmission the scheduler will sample the incoming valid signals indicating that the user logic wishes to transmit data to the remote end of the link on a particular VC. The scheduler will select one of these VC's in a round-robin arbitration mechanism and direct the cell transmission logic to take data from the selected VC. The round-robin arbitration described in the following table ensures that the four VCs get equal access to the physical link.

Last TX VC	Priority 1 VC	Priority 2 VC	Priority 3 VC	Priority 4 VC
VC 0	VC 1	VC 2	VC 3	VC 0

VC 1	VC 2	VC 3	VC 0	VC 1
VC 2	VC 3	VC 0	VC 1	VC 2
VC 3	VC 0	VC 1	VC 2	VC 3

Round-Robin Arbitration

In anticipation that the transmitted VC may be the start of a new frame, the scheduler logic will have already requested a new sequence number from the ACK/NACK Timer Logic. This sequence number will be presented to the cell transmission logic as it is told which VC to transmit data from. If the transmitted cell contains the start of the new frame this cell transmission logic will indicate this to the scheduler. The scheduler will then sample the context ID from the user logic and pass this context ID to the ACK/NACK Timer logic. The scheduler will then mark the VCs state as in frame and store the sequence number for later use. Eventually when the EOF from the given VC has been transmitted as indicated by the cell transmission logic the scheduler will pass the sequence number to the ACK/NACK Timer logic telling it to start the timer for the given sequence number. This ensures that the timer is not started until the last piece of the frame has been transmitted.

While arbitrating between VC's the scheduler is also monitoring the state of the request line from the ACK/NACK transmit logic. It will also monitor whether the cell transmission logic has transmitted an ACK/NACK update in the last cell. If the last cell did not contain an ACK/NACK update (SOF was transmitted) and the ACK/NACK logic is requesting an ACK/NACK update the scheduler will tell the cell transmission logic to transmit an empty cell, ensuring that an ACK/NACK update will occur.

Lastly the scheduler also accepts SKP requests from the physical interface logic. Having received this request the scheduler will allow the current cell to finish transmitting and then acknowledge the request from the physical interface logic. Transmission of SKP symbols has the highest priority in the system.

3.5 ACK/NACK Timer Logic

The ACK/NACK Timer Logic block is responsible for tracking the sequence numbers for frames

which are transmitted to the remote end of the link. The sequence numbers are 256 unique IDs which are assigned to each frame that is transmitted across the link. As each frame starts transmission the user logic will pass a 32-bit context ID associated with the transmitted frame. Working with the TX Scheduler logic the ACK/NACK Timer Logic will assign a sequence number from a FIFO containing free sequence numbers and store this context ID in a memory addresses by the sequence number assigned. When the frame has completed its transmission a timer for the sequence number will be started.

As the remote end of the link transmits ACK/NACK updates the cell reception logic will pass those updates to the ACK/NACK timer logic. The sequence number will be used to retrieve the associated context ID which will be passed back to the user logic. The sequence number will be marked as IDLE and added to the FIFO containing free sequence numbers.

Local timer logic will continuously monitor each location in the context memory once every 2uS comparing the current timer counter to the timeout values in each location in the context memory. If any of the timers are found to be expired the given context will be passed back to the user logic with an error indication and the sequence number will be returned to the sequence number FIFO.

If an ACK/NACK is received from the remote end of the link after a timer has expired the received ACK/NACK indication will be ignored.

3.6 Receive Tracking Block

The Receive Tracking block receives updates from the cell reception block as each cell is received. This information is used to track the current state of each VC. As the head of each frame is received the sequence number of the frame will be passed to the tracking logic and the state of the VC will be marked as 'in_frame'. When the end of that frame is eventually received the stored sequence number will be passed to the ACK/NACK transmission block. If for some reason the EOF cell is missed and a new frame is started an error indication will be passed with the sequence number to the ACK/NACK transmission block.

The Receive Tracking block will also detect when there are sequence errors on a VC. Examples of

sequence errors are when payload frames are received when the VC is IDLE or when a SOF cell is received and the EOF cell for the previous frame has not been received. Any errors detected by the receive tracking logic will result in the cell receiver asserting EOFE at the end of the frame.

3.7 ACK/NACK Transmit Logic

As ACK/NACK updates are received from the Receive Tracking Block the ACK/NACK Transmit Logic will queue those ACK/NACK updates into a FIFO. If the FIFO is not empty a request signal will be passed to the Transmit Scheduler. At the same time the ACK/NACK information will be passed to the Cell Transmission block. When the Cell Transmission block transmits a cell with ACK/NACK information a signal will be passed back to the ACK/NACK Transmit Logic indicating the next ACK/NACK update should be read from the FIFO if available.

3.8 Random Data Generator

The Random Data Generator generates a random 8-bit value which is used by the Cell Transmission block as filler data between cells and as payload data in IDLE cells. At this time the formula used to generate a random number has not been defined.

3.9 Error Count

The Error Count block of logic is a very simple block which performs edge detection on all error signals used to increment counters. As an error signal is asserted the Error Count block will generate a single-clock wide pulse which can be used by external logic to increment a counter.

4. KNOWN LIMITATIONS

This section of the document identifies the known limitations of the PGP design. Knowledge of these limitations is important when designed logic which uses the PGP to transport frames.

Lost cells within a frame are undetected.

The PGP does not have a mechanism to detect that a cell within a frame has been dropped. When receiving a cell with a CRC error, information about the VC associated with the cell can not be trusted. The result of this limitation is frame data will continue to be passed to the user logic without detecting the missing frame data. The user logic must have a mechanism to detect that the frame data it has received has been truncated.

Two subsequent frames can be concatenated

It is possible for two frames to become concatenated if the EOF cell for one frame and the SOF cell for the next frame are lost. This will result in the acknowledgement of the successful reception of the first frame and the ACK timeout of the second frame. A seemingly intact frame will then be passed to the user logic. The user logic must have a mechanism to detect this type of frame concatenation.

ACK timeout can occur on successful frame transmission.

An ACK timeout in the PGP can be the result of a frame transmission error or the loss of an ACK update from the remote side. In the case where the ACK update is lost a timeout will be passed to the user logic even though the frame has made it to the remote end of the link. If the user logic wishes to re-transmit the frame the user logic on the receive side of the link may receive two copies of the same frame.