digital

Generic Flow Control (GFC) Design Considerations

Rev 1.0 3/26/96

Contact: Bob Simcoe
Digital Equipment Corporation
simcoe@school.enet.dec.com

Contents

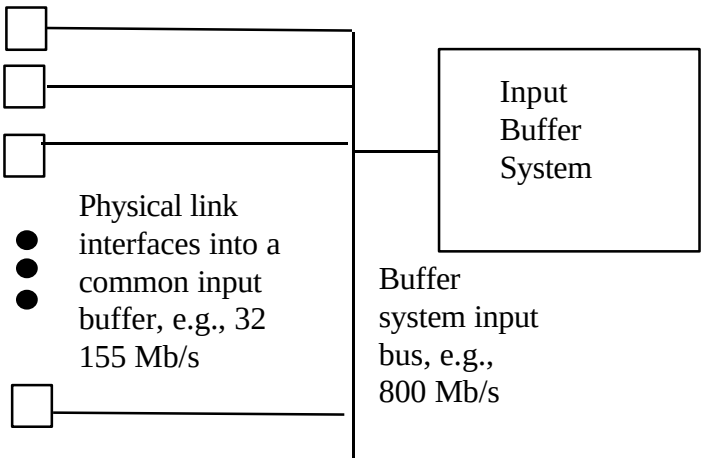
Generic Flow Control

Generic Flow Control (GFC) is a one-way, link-level flow control scheme. It controls the flow of data into an ATM network. GFC is defined in ITU-T SG 13 Revised version documents I.361-TD41 and I.150-TD42 November 1994 Geneva . The essential parts of those documents are also contained in the ATM Forum contribution 95-0181.

This document provides a rationale for the use of GFC, and presents design considerations so that those who chose to implement this function have some guidance toward a common service model and parameters for interoperability. The economic advantages that acrue with the use of GFC should lead to its wide-scale deployment in ATM networks.

1. Introduction

Generic flow control is a one-way, lossless control mechanism which allows the network equipment to control the output from an end station for the class of traffic that is defined as controlled. This mechanism does not allow the end station to exert any control on traffic from the network.



GFC Gains

The usefulness of this mechanism is that it allows overbooking of the bandwidth of the input side of the switch (or other network equipment) buffers. This allows a much higher degree of lossless multiplexing than is otherwise possible. This sharing allows the network side costs of connections to be significantly reduced. By overbooking the input bandwidth of the switch buffers, a high degree of lossless sharing is possible and the buffer system can be utilized by more end nodes than the otherwise normal requirement of full bandwidth input would allow. GFC is used to coordinate access to the shared input bandwidth when temporary overbooking conflicts occur. GFC in this application provides a link level, short term, flow control mechanism that only works on the link from the end station to the first piece of network equipment. GFC works in conjunction with the larger switch buffer system that is organized generally either by service category or by VC. GFC is a link level, not a network level, flow control mechanism and is expected to be active only for the short intervals of time needed to sequence access to a shared buffer input bandwidth resource. GFC does not replace or obviate the need for network level flow control mechanisms.

The figure “GFC Gains” shows the kinds of gains that can be realized with GFC. A buffer system with 800-Mb/sec. input bandwidth would only be able to support about five 155-Mb/sec links. With GFC, the

number can be increased substantially. For instance, with 32 155-Mb/sec links, each link would have the same bandwidth available as a 25-Mb/sec link under heavy loading conditions, but would still have the 155-Mb/sec bandwidth accessible under normal conditions. This allows the cost of the input buffer system to be shared by six times as many end stations and reduces the cost of that part of the system to 1/6 of what it would be otherwise.

As defined by ITU-T, GFC has three categories of traffic: uncontrolled, class A, and class B. This document will focus on the minimal set of functionality needed for the Local Area Network (LAN) environment. For this application area, only the uncontrolled and class A channels will be used. This gives an uncontrolled channel for real-time, delay-sensitive service categories which are controlled by higher level coordination of network access, and a single, controlled channel for service categories which can be subject to control at the data link level according to network congestion conditions or other factors. Controlled traffic is for applications that can tolerate a wider range of delay in data delivery before they are deemed not useful.

In the example above, the network call admission control would need to coordinate the sum of uncontrolled bandwidth on each link to be sure that it did not exceed the bandwidth bottleneck and that some bandwidth was available for the controlled service.

Practical implementations of GFC need to address certain potential problems and parameter decisions. This document is intended to give guidance in those decisions.

2. GFC field definitions

GFC is a one way flow control scheme where the end station is the controlled equipment, and the connected switch is the controlling equipment.

In the GFC protocol, the ATM connections are divided into two categories: controlled and uncontrolled. In the end station, for instance, all connections for CBR and rtVBR service categories are uncontrolled, and the rest of the connections, (ABR, UBR, nrtVBR service categories) are controlled.

This document will discuss only the default mode (one queue mode) in the GFC specification. It is expected that this will be the interoperable implementation that most end stations and network equipment will implement. This document does not define any use for the SET_B controlled channel, nor consider any implementation issues other than not precluding its definition or use in the future.

The GFC field in ATM cell header contains 4 bits. The bits have a distinct meaning for each direction of the link. The bits transmitted and received by the end station (controlled equipment) are shown in Table 1 and Table 2. In the direction toward the end station, the combinations of values of the GFC field are shown in Table 3. All other combinations are ignored by the end station. In the direction from end station to the switch, the possible GFC field settings by the end station are shown in Table 4.

Table 1
GFC Field Definition from Controlling Equipment to Controlled Equipment for the default mode

Bit	Name	Value	Functions
0 (MSB)	HALT	0 - no halt 1 - halt	Halt stops the end stations transmission towards the network of cells for all ATM connections.
1	SET_A	0 - not set 1 - set	This field applies to controlled ATM connections.. When SET_A = 1 is received, the GO_Counter is set to GO_Value (GO_Value default value = 1)
2	SET_B	0	The default mode does not use SET_B and therefore it is always 0
3 (LSB)	unused	0	Always 0.

Table 2
GFC Field Definition from Controlled Equipment to Controlling Equipment for the default mode

Bit	Name	Value	Functions
0 (MSB)	unused	0	Always 0.
1	CTRL_CONN	0 - uncontrolled 1 - controlled	This field indicates the cell belongs to a controlled connection (1) or an uncontrolled connection (0).
2	unused	0	Always 0 in the default mode.
3 (LSB)	GFC_Control enabled	0 - uncontrolled 1 - controlled	This field indicates the equipment is GFC controlled (1) or not GFC controlled (0).

Table 3
Valid combinations of bits in the GFC Field towards the Controlled Equipment for the default mode

Combinations (bits 0 1 2 3)	Functions
0 0 0 0	no halt, not set
1 0 0 0	halt, not set
0 1 0 0	no halt, set
1 1 0 0	halt, set

Table 4
Valid combinations of bits in the GFC Field from the Controlled Equipment for the default mode

Combinations (bits 0 1 2 3)	Functions
0 0 0 0	End station is uncontrolled.
0 0 0 1	End station is controlled. Cell on an uncontrolled ATM connection.
0 1 0 1	End station is controlled. Cell is on a controlled ATM connection.

3. Mapping ATM traffic services categories to GFC Classes

There are only two classes of service for GFC, but the ATM Forum has identified 5 major service categories, each with several variations. There is a necessity to map the ATM Forum-defined categories into the GFC classes. This document recommends a mapping that identifies the services that are real-time and delay sensitive (CBR and rt VBR), and assigns them to the uncontrolled category of GFC. The services that are not real-time and can tolerate the delay associated with control and sharing (nrt VBR, ABR, and UBR), are recommended to be mapped to the controlled category of GFC. Other mappings are possible. For instance ATM Forum rate ABR might be mapped into the uncontrolled class, but the impact on the buffers needed at the switch for GFC would be substantial and interoperability may not be achieved.

4. Implementation Issues

The following implementation issues arise:

4.1 Start up

The ITU-T defined way for the end station to recognize that the network device is capable of GFC and desires to control with that mechanism is to observe that either the Halt or the Set_A or Set_B bits are set. Network devices that do not do GFC control will always have these bits set to zero. The ITU-T documents recommend a 5 second decision period.

4.1.1 The Problem

If the line were to be monitored at all times for occurrences of Halt or Set_A or Set_B bits set, the end station will likely go into the GFC controlled mode by being falsely triggered by datalink errors that are not caught by the header error check. The header error check is not a strong check, since it only is an eight bit quantity. For many error conditions, there will be a 1/256 chance that the header error check will have the value that indicates that the errored data is correct. Since the GFC will most likely be used on the links that are from the end station to the first network device, these links will often be copper wire which is subject to more noise and errors than optical fibers. This makes it very likely that links which have been up for a while will see errored bits that indicate that the Halt or Set_A or B bits are set. This makes it dangerous to have simple logic that continuously monitors these bits to put the end station into the GFC active mode.

If the end station goes into the controlled mode, when the network side of the link is not really capable of GFC and is transmitting all of the GFC bits as zero, the link will stop because the end station will believe that the network is controlling it off. This is because the defined polarities of the signals, coming from a network device that is in the uncontrolled mode to an end station that has put itself into the controlled mode, are such that they will cause the end station to stop sending all controlled traffic

The 5 second decision period cannot be easily built into hardware as perhaps envisioned by the ITU-T specification writers. The reason for this is that most implementations for setting the GFC bits are software or firmware driven. The time it takes to start the software for each system will vary. While the low level

hardware may be up and running immediately, the GFC bits may not be set in any deterministic way. The 5 second test time should not be started until it is certain that the software is initialized and running.

4.1.2. Solutions

The recognition process for using GFC control on the link should be strongly filtered so that false recognition is reduced to an extremely small probability. This can be done by making it possible recognize the signals for going into the GFC mode only for a small time period at start up and by filtering the signal so that it would require multiple error events to trigger the undesired action. After that the GFC bits should either be utilized for control or ignored. In order to deal with the cases when one end of the link does not implement GFC, both sides of the link need to implement a timer. This can be done in firmware or software. These activities can only be reliably accomplished after the software is initialized and running since most implementations of the physical layer hardware requires that the GFC bits be supplied by the upper level software.

4.1.3 Startup condition proposal

4.1.3.1 Network side

The network side software or firmware shall, as soon as it is properly initialized and running, turn off the transmitter output to force the line to resync.

After resync, the network side shall set the Halt bit = 0 (or alternatively the Halt bit can be set and reset in a cyclic manner) and the Set_A = 1 in all cells and start a 5 second time-out period. (Note: This is a "safe" setting in that it allows data to flow during the 5 second time the GFC mode capability is being determined, but it is an uncontrolled mode that does not protect the network equipment during this period). During the 5 second time out period the network side shall monitor the return cell stream.

If the return cell stream has the GFC_Enabled bit set on at least three cells (from an unspecified number of samples) during the time period, the network side shall go into GFC_enabled mode and the Set_A bit can now be used to control the flow of controlled data. Once in this mode the network device shall remain in this mode until the link is reset.

If a time-out period expires and at least three GFC_Enabled bits have not been set on received cells, the network device shall go into the non GFC control mode and shall reset both the Halt and the Set_A bits. Once in this mode the network device shall remain in this mode and send all GFC bits as zeros until the link is reset.

4.1.3.2 End Station side

The end station side software or firmware shall, as soon as it is properly initialized and running, turn off the transmitter output and force the line to resync.

The end station shall start by setting the GFC_Enabled bit = 0 in all cells and start a 5 second time out period.

If the Set_A bit is received as set in at least three cells (from an unspecified number of samples), the end station shall go into GFC controlled mode. The end station can start sending uncontrolled cells when the Halt bit is received as reset and can start sending controlled cells when both the Halt bit is reset and the Set_A bits is set on received cells.

If a time-out period expires and at least three cells (from an unspecified number of samples) have not been received with the GFC bits set, then the end station shall set itself into the non GFC controlled mode and transmit the GFC bits as all zeros. A more formal description of the proposed start up procedure follows:

Proposed GFC Init Procedures

Event	Current State	Next State	Output
All lower-level initialization complete	PreGFC	NewLinkInit	Turn TX power off for >500 ms
TX power is turned back on and RX ATM cell framing achieved	NewLinkInit	GFCquery-Complete	Start "GFC_Enable" test
GFC_Enable test complete	GFCquery-Complete	LinkEnabled	Set GFC_ENABLED = GFC_Enable test result
Receive power > 200 ms	NewLinkInit, GFCquery-Complete, LinkEnabled	NewLinkInit	Clear GFC_ENABLED bit

Other requirements:

1. After turning off TX power to start the transition to the "NewLinkInit" state, each receiver must achieve valid RX ATM cell framing and start the GFC_desired test within 2 seconds after a valid cell stream is present at that receiver.

2. The GFC_Enable test can terminate at any time with a positive result; otherwise, the test times out after 5 seconds (ITU-T spec) with a negative result.

GFC_Enable test at **controlling** equipment (Network side):

1. The SET_A bit shall be **asserted** on all transmitted cells including unassigned cells during the entire test.

2. If at least three received cells(from an unspecified number of samples) contain GFC_ENABLED = 1, then return a positive result.

GFC_desired test at **controlled** equipment (End station side)

1. The GFC_Enabled bit shall be **deasserted** on all transmitted cells including unassigned cells until the GFC_Enable test returns a positive result

2. If ((SET_A) = 1) in at least three received cells (from an unspecified number of samples), then the GFC_Enabled bit shall be **asserted** on all subsequent transmitted cells including unassigned cells and return a positive result.

4.2 SONET Overhead

4.2.1 The Problem

The definition of GFC with a go counter default value of 1 assumes that the data stream in the two directions on the link are running on the same time base, and that decisions about forwarding cells is done in the same time frame. Following this assumption, each cell received with the Set_A bit set represents a credit to send one cell in the reverse direction (or a number of credits if go count values other than 1 are implemented). A simple implementation of this would have a single bit of state that is set when Set_A is received, and is reset when the next cell is transmitted. Unfortunately, because of SONET overhead processing, even if the two directions on the link are running at the same precise speed, there will be times when the phase relationship between the two streams suddenly shifts and there will be two transmit opportunities in the time that there was only one receive opportunity. With a simple implementation of GFC state for the default go counter, this will result in lost send opportunities, and link utilization will be affected adversely. Also it may be desirable in some implementations to make transmit decisions early so that a pipeline of data can be established. These decisions may even be made on a much faster time scale than the received data cells carrying Set_A signals.

4.2.2 Solutions

To compensate for the SONET overhead, there are several options. One is to have a go counter of 2 even if the default is one. Another is to treat the Set_A signal more like an Xon/Xoff signal. That is, instead of being a credit to send one cell, it is permission to send until that permission is revoked. Another is to have a default Go count increment of one with a limit on the Go count accumulation of default + 1. Another is to use the speed differential of the Utopia interface to hide the SONET overhead delay. These solutions are adequate for implementations that are not heavily pipelined. See appendix A for a more detailed description of a solution for heavily pipelined implementations.

Option 1 Not recommended for implementation

The first option, setting the go counter value to 2 is not recommended. This option would require network management intervention or compatibility restrictions.

The second method is almost correct but has a subtle problem when the GFC is running at near link bandwidth.

The third and fourth methods for dealing with SONET overhead stutter are recommended and provide the same system behavior that the GFC specification seems to intend.

Option 2 Not recommended for implementation

To bridge the SONET gap there can be a bit of state *A* that holds the Set_A bit from the last received cell. (This is used to bridge the SONET overhead time.) Another bit of state (go_count) also samples at each receive cell time. The input to the go_count state set it to a one if *A* is one or go count is one. Go_count can be asynchronously reset if *A* is zero and a controlled cell is sent.

The go_count becomes a credit to send a single cell. On transmitting a cell, the credit is decremented only if A is zero. If A is one, then a new credit is available and there is no need to decrement the go_count.

The cell used to implement the go_counter should allow the strobed input to dominate the reset if the two events happen at the same time.

The effect of above described actions should be to implement a go_counter of one with the property that if no new cell fully arrives before the next transmit opportunity, the transmitter can use the last received information again for the new cell to be transmitted (even though it used that same information for the previous cell). Also there will be a memory of only a single unused cell transmit credit carried forward in time.

This implementation has the property that it does not always follow the pattern of Set_A that is received. A single Set_A of zero may not result in no cell being sent because the mechanism to bridge SONET overhead may also bridge a single Set_A of zero. This implementation therefore does not actually follow the spirit of the GFC control stream.

Option 3 A recommended implementation

To bridge the SONET gap, the GO counter may be incremented to one count above the default GO_VALUE = 1(e.g. 2, but note this is not the same as having 2 be a default value), but if that occurs then any Set_A = 0 received reduces the GO counter back to the default GO_VALUE. Pseudocode detailing such an implementation is presented in Appendix B.

Any similar implementation must be sure to never send more controlled cells than allowed by the number of received SET_A commands, and must never send more than ' default GO_VALUE' controlled cell(s) after the received commands become a steady stream of Set_A = 0.

Option 4 Another recommended implementation

Another implementation, similar in properties, relies on the Utopia interface being faster than the link. This allows the transmit decision to be made somewhat later in time than simple pacing would allow since the cell can be delivered across the Utopia interface faster than it can be delivered to the link. This allows a delayed decision at the transmitter Utopia interface if no received cell is available with GFC guidance. The speed difference of Utopia can be used to hide the SONET overhead stutter that occurs on the link. If no cell has been received at the receive interface, the transmitter can wait up to about a quarter of a cell time (dependent on the speed difference of Utopia interface and the line rate) for the receive cell to come in before making the transmit decision.

4.3 Defaults for the number of cells buffered

Implementators need to know a buffer budget so that they have some guidance on how much buffering is needed and how to apportion that buffering to cover logic processing and link propagation delays. These numbers may well vary by link speed since logic, system, and propagation delays tend to increase with link speed. Two link speeds where GFC is most likely to be needed are OC-3 and OC-12. ATM links at 25 Mb are less likely to need GFC because the input bandwidth of typical buffer system can support large numbers of these links without any need to overbook the input bandwidth.

The typical client links for OC-3 STP and OC-12 Multimode fiber are limited to 100 meters and 500 meters, respectively. The propagation delays for these distances are on the order or 1 cell for OC-3 and 8 cells for OC-12. (A 10us per km round trip propagation delay for fiber translates to 1us RTT for a 100 meter link and 5 us for a 500 meter link. At 2.83us/cell at OC-3 rates and 0.71us/cell at OC-12 rates this translates to 1 cell for the OC-3 link and 8 cells for the OC-12 link (after rounding up).

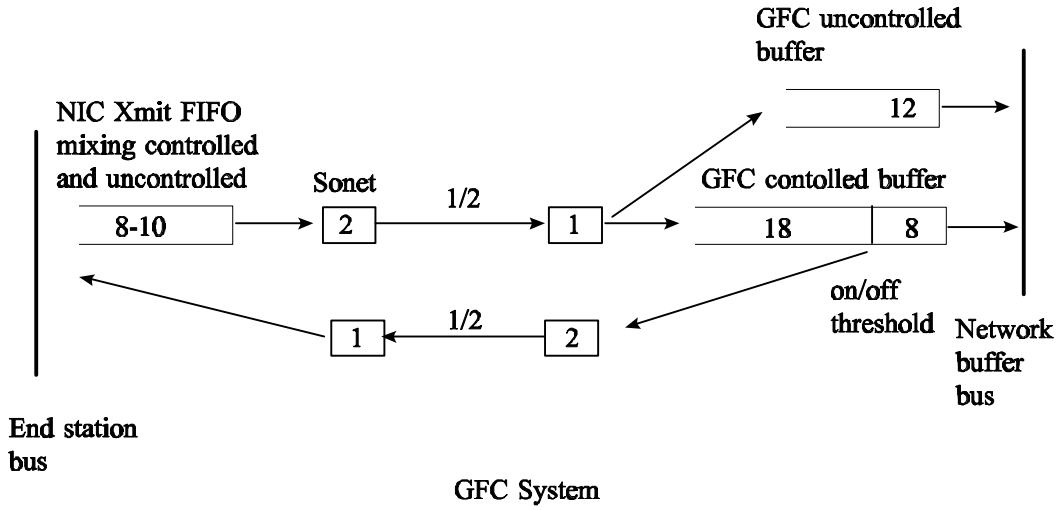
The Sonet framer chips used for 155 Mb/s generally add 6 cells over a round trip (2 for transmit and 1 for receive on each end) and at 622 Mb/s the number is more like 12 cells (three at each interface).

The number of cells of buffering for each GFC controlled link can be designed to allow a simple implementation (even though we don't recommend this as the best alternative) in the end station. By simple implementation we mean, for instance, that the decision to transmit controlled and uncontrolled traffic can occur at the PCI bus interface. This means that a single fifo can be used for pipelining both kinds of data across the bus, but it also means that the fifo is part of the link round trip time. A better implemetation would maintain separate fifos for each data type(controlled and uncontrolled), the decisions to send controlled or uncontrolled traffic can be made closer to the physical link and the fifos will not be part of the link delay.

For the network side our recommendation for 155 Mb/s links is 38 cells (or 2k bytes) of buffering per link. The recommended budget for distributing this buffering is:

- 6 cells for physical layer and logic delays at the network side
- 1 cell for link propagation delays
- 19 cells for total end node delay(physical layer delays, fifo delays, and the Go_value of 1) for the controlled traffic class
- 12 cells for speed-matching uncontrolled class data cells onto a higher speed bus in the network equipment for a small number of VCs (e.g., 8).

Assuming an input buffer bandwidth of at least 800 Mb/sec and no more than 32 155 links statistically sharing that bandwidth, this gives each link at least 25 Mb/sec if all links are active simultaneously and all get an equal share. At the network equipment side, the buffer would be divided into two sections. The



GFC System

uncontrolled section would consist of 12 cells and the controlled section would consist of 26 cells. The threshold for the controlled buffer could be placed at the 8th cell point, giving 18 cells of storage from the time sending a cell with Set_A = 0. As the fifo empties, Set_A is asserted if less than 8 cells remain in the buffer. This allows end nodes with up to 10 cells of fifo buffer between the GFC decision point and the PHY to work without having separate fifo buffers for the different GFC classes.

There is no mechanism to control the burstiness of uncontrolled cells from multiple VCs from an end node. Call admission control must deal with the allocation of bandwidth of the bottleneck resource. For a switch that uses GFC to deal with buffer system input bandwidth limitations, the CAC must insure that the input bandwidth is not overbooked with uncontrolled traffic, and should probably leave a margin of bandwidth to drain buffers of uncontrolled traffic.

This still leaves open the issue of sizing the buffers for the uncontrolled traffic. If CAC has limited the bandwidth, then the buffers should not build up for uncontrolled traffic. Two or three cells per input port should be all that is needed, provided that data was all paced and never clumped. However arbitration for the input bus and other system factors may not give perfect performance. Even with perfect pacing, data from independent VCs can arrive in a burst due to alignments of scheduling, so a generous margin for error is appropriate. We recommend that 3-4 cells be allocated at the network equipment to allow holding

uncontrolled data until it can be synchronized with its bandwidth allocation and placed on the buffer bus, and that one cell per active VC in the uncontrolled category be reserved so that a burst of cells arriving on different VCs can be handled. Since the GFC is intended for client end nodes, we recommend that the number of active real time VCs planned for be eight. This makes twelve the total buffer recommendation for the uncontrolled service class.

Appendix A: Transmission With GO_VALUE Set to One

A.1 Background

ITU Recommendation I.361, Annex B, specifies a power-up default of “one” for GO_VALUE (in the one-queue model of GFC). The value could be increased through management procedures. Maintaining full transmit performance in spite of clock differences, and of GFC signals lost by link errors, is one reason to increase GO_VALUE. We believe that the loss of GFC signals on the link has a negligible impact on performance—less impact, in fact, than the loss of data cells due to link errors. And by careful design, GFC implementations with a GO_VALUE of one can sustain full performance despite the asynchronous arrival and departure of cells.

Another reason to increase GO_VALUE would be to allow a SAR chip to make multiple GFC transmit decisions an unbounded amount of time before the controlled cells are emitted by the terminal. For example, the SAR could build of *a queue of* committed transmissions in the form of DMA requests for the host memory bus. Unless DMA latency were tightly bounded, such a SAR chip might emit a burst of cells long after the most recent GFC Set_A command. Besides requiring an increased GO_VALUE, this SAR structure seems to add unnecessary complexity.

We begin with one design solution for the common case of GFC and PHY logic spanning three clock domains, and SONET-like framing of ATM cells. That is, we show GFC logic to be embedded in a SAR chip that uses a typical Utopia PHY chip. GFC logic could also be added to the PHY transmit clock domain for greater simplicity, if a Utopia interface is not required.

A.2 Handling SONET and Utopia Jitter

We are given a PHY layer with nominally equal rates for received and transmitted cells. Every received cell—unassigned or occupied—will be passed to the ATM layer, for interpretation of GFC commands. Only cells with bad HEC are ignored. For the design presented here, bad cells may be discarded at either layer. PHY Idle cells are also ignored by the GFC logic, but a GFC-controlling switch would be sending unassigned cells instead. (Unassigned cells carry GFC signals, whereas Idle cells do not.)

Our problem is that at certain points of reference, the receiver’s and transmitter’s cell periods aren’t matched, for short spans of time. Even within the SONET PHY layer, one byte stream falls 10 bytes behind the other where SONET overhead is inserted. The mismatch is worse at typical points of implementing the GFC procedures.

A GFC command can be interpreted a few clock cycles after the start-of-cell byte crosses the Utopia RxData[n:0] bus. If the Utopia interface bandwidth is much higher than the PHY bit rate, then the SAR chip might jitter the start-of-cell transfer by a cell time or more with respect to PHY cell arrivals. Similarly,

if GFC-based transmissions are scheduled by a high-speed controller which is separated from the PHY layer by fifos, transmit decisions might burst several times faster than the PHY cell rate. Figure 1: Timing of Transmissions versus Received GFC Signals depicts mild receive and transmit bursting, as if the Utopia busses and the transmit scheduler ran only 25% faster than the PHY layer.

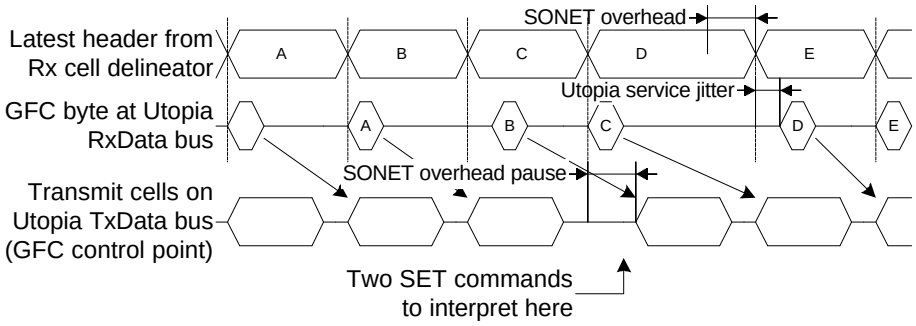


Figure 1: Timing of Transmissions versus Received GFC Signals

As noted in the figure, GFC logic separated from the PHY layer by a typical Utopia implementation will have to interpret a history of two SET commands in order never to miss a transmit opportunity. (We presented suitable logic to interpret a pair of SET commands earlier in this paper.) We recommend this simple approach, with GFC implemented near the Utopia transmit bus, for most SAR chip designs. This approach works well if the SAR chip has one transmit fifo per GFC group (e.g., *uncontrolled* and *group-A* fifos), or if the SAR chip confines its active connections to a single group. A SAR chip with two transmit fifos is depicted below.

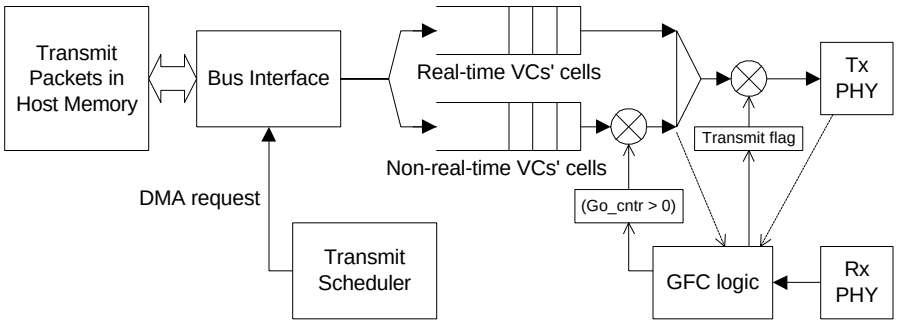


Figure 2: SAR Structure with a Transmit Fifo per GFC Group

If the SAR chip cannot be equipped with a transmit cell fifo per GFC group, then the Go_cntr decision would have to be made prior to adding a cell to the (only) transmit fifo. As a result, the transfer delay of the transmit fifo becomes part of the round-trip delay of the generic flow control loop—call it RTT_{GFC}. Since the transmit fifo is elastic, meaning that it has a variable delay, the Go_cntr decision could jitter by several cell times with respect to the PHY-layer stream of arriving SET commands. To maintain full transmit performance when Go_cntr decisions may fall *N* cell times behind the GFC SET arrivals, we need to interpret a history of *N* SET commands. Suitable logic for this case is presented below.

A.3 Handling Transmit Fifo Jitter in the GFC Loop

Consider a SAR chip that makes transmit decisions at the host-bus side of an 8-cell fifo. Each Go_cntr test permits the transmit scheduler to commit one cell for transmission, by posting a DMA read request to fetch the cell from the host memory. Suppose the peak transmit scheduling rate is twice the PHY's cell rate. When the host bus is available, the SAR chip tries to fill its transmit fifo, in case of high latencies for future bus transfers. Any cell added to the fifo must not be blocked by downstream GFC logic, because GFC "uncontrolled" cells (e.g., for CBR service) flow through the same fifo. (This comment does not apply to GFC HALT commands, which an ATM switch negates with guaranteed frequency.)

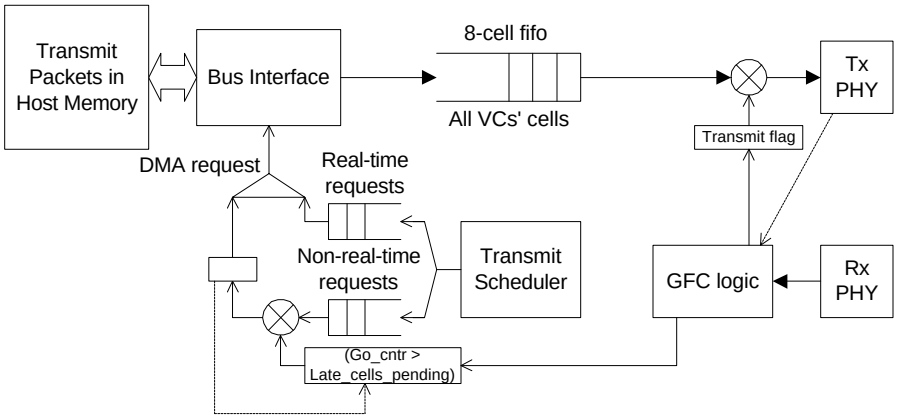


Figure 3: SAR Structure with a Single Transmit Fifo

In the worst case, we need to burst 16 cells into an empty 8-cell fifo at twice the PHY's cell rate. During this burst, the first 8 cells will be passed on to the PHY layer, and 7 to 9 new GFC SET commands will be received (ignoring link errors). The last GFC SET command typically arrives after the 16th transmit-cell decision, so count on 6 to 8 SET commands to replenish Go_cntr during the transmit decisions.

If only 6 new SET commands are received during the burst of transmit scheduling, then we must exploit a history of 10 prior SET commands to avoid possible lost transmit opportunities. We need a burst-oriented interpretation of SET commands that complies with Recommendation I.361, and that fits the design parameters of the controlling switch port and the desired cable length. (Refer to the earlier section which presents a recommended GFC latency budget.)

The following GFC algorithm is suitable for the problem posed above. Our goal is to convert sequential SET commands into a batch of transmit decisions. In the example algorithm, each SET command is "double counted", but then Go_cntr is decremented once per PHY cell time to remove GFC permissions in excess of GO_VALUE. As before, we speak of a SAR chip connected to the PHY layer by the Utopia interface. All valid received cells are passed to the GFC logic with a small delay. We use cell transmission events at the Utopia interface as our measure of the PHY cell rate. The SAR chip must send unassigned cells to the PHY layer to keep its transmitter busy. Posting a DMA request commits a cell for transmission, regardless of the DMA latency.

GFC ALGORITHM:

```
Constant GO_VALUE = 1;
Constant ELASTICITY = 9;  // Sufficient SET history for the example SAR design.
VAR GFC_enable := FALSE;
VAR Transmit := TRUE;
VAR Go_cntr := 0;              // GFC permissions for group A.
VAR Late_cells_pending := 0;          // Group-A permissions used for pending DMA requests.
VAR Real_time_pending := 0;          // Reserve fifo space for uncontrolled VCs' cells too.
VAR transmit_fifo := make_tx_fifo (<8 cells>);


// Represent synchronous handling of GFC events.
PROCEDURE tick ()
BEGIN
   IF (phy_rx_cell_available) THEN start_receive_cell ()
   ELSE IF (phy_tx_cell_needed) THEN start_transmit_cell ()
   ELSE IF (tx_DMA_complete) THEN finish_tx_cell_DMA ()
   ELSE IF (not_full (transmit_fifo)) THEN BEGIN
        IF (<any uncontrolled VC due to transmit>) THEN
            start_tx_cell_DMA (<the VC>, uncontrolled)
        ELSE IF (<any controlled VC due to transmit>) THEN BEGIN
                IF ((Go_cntr > Late_cells_pending)
                   AND (Late_cells_pending < GO_VALUE)) THEN
                        start_tx_cell_DMA (<the VC>, group_A)
                ELSE IF (NOT GFC_enable) THEN
                        start_tx_cell_DMA (<the VC>, group_A);
        END;
   END;
END;


FUNCTION not_full (tx_fifo : Tx_cell_fifo)
BEGIN
   RETURN (free_cell_spaces (tx_fifo) > (Real_time_pending + Late_cells_pending));
END;


PROCEDURE start_receive_cell ()
BEGIN
   VAR rcell : ATM_cell;

   rcell := get_phy_rx_cell_header ();
   IF (bad_HEC (rcell)) THEN RETURN;
   detect_GFC_mode (rcell);  // Monitor GFC bits to enter GFC mode as specified.
   IF (GFC_enable) THEN BEGIN
        Transmit := NOT rcell.GFC.halt;
        IF (rcell.GFC.set_A) THEN BEGIN
```

```
            // The excess step in Go_cntr is intentional; see start_transmit_cell().
            IF (Go_cntr < GO_VALUE) THEN
                Go_cntr := GO_VALUE + 1
            ELSE IF (Go_cntr < (GO_VALUE + ELASTICITY)) THEN
                Go_cntr := Go_cntr + 2;
        END;
    END;
END;


PROCEDURE start_transmit_cell ()
BEGIN
    VAR tcell : ATM_cell;

    ASSERT (Late_cells_pending <= GO_VALUE);
    IF (Go_cntr > GO_VALUE) THEN Go_cntr := Go_cntr - 1;
    IF (Transmit AND not_empty (transmit_fifo)) THEN BEGIN
        tcell := read_cell (transmit_fifo);
        // tcell.GFC.group_A is true if the cell came from a “controlled” VC.
    END ELSE tcell := make_unassigned_cell (GFC.group_A := FALSE);
    tcell.GFC.controlled_mode := GFC_enable;
    start_tx_utopia_cell (tcell);
END;


PROCEDURE finish_tx_cell_DMA ()
BEGIN
    VAR tcell : ATM_cell;
    VAR txVC : Tx_VC_index;

    txVC := get_DMA_VC_index ();

    tcell := format_cell (get_DMA_read_data (), txVC);
    tcell.GFC.group_A := is_controlled_VC (txVC);
    // Allow next DMA request. Note that Late_cells_pending captures the state of
    // GFC_enable in start_tx_cell_DMA(). Use it here too, for synchronization.
    // At all times, (Go_cntr >= Late_cells_pending)
    // and (Late_cells_pending <= GO_VALUE).
    IF (tcell.GFC.group_A AND (Late_cells_pending > 0)) THEN BEGIN
        Go_cntr := Go_cntr - 1;
        Late_cells_pending := Late_cells_pending - 1;
    END ELSE Real_time_pending := Real_time_pending - 1;
    IF NOT (GFC_enable) THEN tcell.GFC.group_A := FALSE;
    write_cell (tcell, transmit_fifo);
END;
```

```
PROCEDURE start_tx_cell_DMA (txVC : Tx_VC_index, group : GFC_group)
BEGIN
   IF (group == uncontrolled) THEN BEGIN
        request_DMA_read (txVC);
        Real_time_pending := Real_time_pending + 1;
   END ELSE BEGIN                                  // txVC is in group_A.
        request_DMA_read (txVC);
        IF (GFC_enable) THEN Late_cells_pending := Late_cells_pending + 1
        ELSE Real_time_pending := Real_time_pending + 1;
   END;
END;
```

Figure 4: Pseudocode for GFC Decision in Single-fifo SAR Chip

This pseudocode was written to meet requirements of the GFC specification, and to address some aspects of a SAR design, in the following sense.

1. Some time RTT_{GFC} after a final SET command was sent by the controlling ATM switch, at most GO_VALUE additional cells will arrive at the switch for controlled connections. This RTT_{GFC} covers the link's round-trip time, PHY-layer delays at each end, the switch's ATM-layer GFC logic delay, and the terminal's worst-case transfer delay for fifo(s) that affect committed transmissions.
2. Define T_{GFC}(term) to be worst-case delay from stopping the SET commands at the terminal's input to seeing no more controlled cells at its output, aside from a final GO_VALUE cells that may be transmitted at any time. We allow a terminal to collect recent SET commands, and to apply them in any bursty pattern, providing that no GFC SET command allows a transmission later than the stated worst-case response time, T_{GFC}(term).

   If you subtract from T_{GFC}(term) any fixed delays within the terminal's GFC logic path, the difference is an upper bound on the history of SET commands that a terminal might apply to rapid GFC decisions.
3. In a modular computer, the worst-case DMA latency (times the number of queued DMA requests) may exceed the budget for RTT_{GFC} imposed by the ATM switch. If so, then another controlled cell may be committed for transmission only if (Go_cntr > Late_cells_pending), as opposed to (Go_cntr > 0). To meet condition (1) above for any DMA latency, at most GO_VALUE DMA requests can be posted at a time, for all controlled VCs.
4. For uncontrolled VCs, the code is written to allow a queue of committed transmissions to be formed as DMA requests. This queuing may not be useful. The code merely shows that arbitrarily delayed cells are not prohibited by GFC on uncontrolled connections, except by the GFC HALT mechanism.

For some RTT_{GFC}, item (1) is a necessary condition for ITU GFC compliance. Condition (2) is suggested as the constraint on aggressive interpretations of SET commands in the terminal. In any workable configuration, the controlling ATM switch supports a total GFC loop delay of at least the prevailing RTT_{GFC}, and this budget covers the terminal's worst-case response time, T_{GFC}(term). Condition (2) allows a terminal to clump its GFC-controlled transmissions in any pattern, providing that no cell's GFC decision violates the T_{GFC}(term) portion of RTT_{GFC}, the configuration's maximum loop delay.

A.4 Conclusion

We presented a GFC algorithm for a hypothetical SAR chip which has only a single transmit cell fifo. Since the terminal's worst-case GFC response time must fit within the overall GFC loop budget, the algorithm bounds the control delay for committed transmissions in excess of GO_VALUE. Subject to that constraint, the algorithm allows some flexibility in the timing of GFC decisions.

We find that the resulting GFC algorithm is too complex for general use. Whenever possible, SAR chips should be designed with one transmit cell fifo per GFC group.

Appendix B

The model below allows a transmitter to 'ride through' a RX SONET overhead gap. Rather than a pure "go counter," the count can be incremented above the GO_VALUE, but any cell with SET_A not set brings the count back down to GO_VALUE. This model will never send more controlled cells than the number of received SET_A commands, and will not send more than GO_VALUE controlled cells after the received cells no longer carry SET_A commands.

This code allows the gap size to be larger than one cell - such is the case for OC-12 and faster SONET links. But the GO_VALUE is required to be equal to or greater than the gap size.

```
Constant GO_VALUE = 1;
Constant GAP_SIZE = 1;       // Required to be less than or equal to GO_VALUE
VAR GFC_enable := TRUE;  // Assume we already entered the controlled mode.
VAR Transmit := TRUE;
VAR Go_cntr := 0;

PROCEDURE receive_cell (rcell : ATM_cell)
BEGIN
  IF (bad_HEC (rcell)) THEN RETURN;
  detect_GFC_mode (rcell);  // Monitor GFC bits to enter GFC mode as specified.
  IF (GFC_enable) THEN BEGIN
    Transmit := NOT rcell.GFC.halt;
    IF (rcell.GFC.set_A AND (Go_cntr < GO_VALUE) THEN Go_cntr := GO_VALUE ;
    ELSE IF (rcell.GFC.set_A AND (Go_cntr < GO_VALUE + GAP_SIZE)) THEN
      Go_cntr := Go_cntr + 1;
    ELSE IF ((Go_cntr > GO_VALUE) AND NOT rcell.GFC.set_A) THEN
      Go_cntr := GO_VALUE;
    END;
END;

WHILE (<PHY transmitter needs another cell to keep busy>) DO
BEGIN
  VAR tcell : ATM_cell;
  IF (Transmit AND <any uncontrolled VC due to transmit>) THEN BEGIN
    tcell := next_cell (next_uncontrolled_VC ());
```

```
    tcell.GFC.group_A := FALSE;
    END
  ELSE IF (Transmit AND <any controlled VC due to transmit> AND (Go_cntr > 0)) THEN BEGIN
    tcell := next_cell (next_controlled_VC ());
    tcell.GFC.group_A := TRUE;
    Go_cntr := Go_cntr - 1;
    END
  ELSE tcell := make_unassigned_cell (GFC.group_A := FALSE);
  tcell.GFC.controlled_mode := GFC_enable;
  send_Utopia_cell (tcell);
END
```