**EUROPEAN PATENT SPECIFICATION**

**Date of publication and mention of the grant of the patent:**
01.06.2005 Bulletin 2005/22

**Application number:** 01306819.2

**Date of filing:** 09.08.2001

**Transferring and queueing length and data as one stream in a packet switch**

Übertragung und puffern der Länge und Daten als einen Strom in einer Paketvermittlungseinrichtung

Transférence et mémorisation de la longeur et les données comme un flux dans un commutateur de paquets

**Designated Contracting States:**
DE FR GB IT NL SE

**Priority:** 11.08.2000 US 637049

**Date of publication of application:**
13.02.2002 Bulletin 2002/07

**Proprietor:** Marconi Intellectual Property (Ringfence) Inc.
Warrendale, Pennsylvania 15086 (US)

**Inventors:**
- Zhou, Fan
  Petaluma, CA 94954 (US)
- Reddy, Veera
  Wexford, PA 15090 (US)
- Vogel, Jon
  Mars, PA 16046 (US)

**Representative:**
O’Connell, David Christopher et al
HASELTINE LAKE,
Redcliff Quay
120 Redcliff Street
Bristol BS1 6HU (GB)

**References cited:**
EP-A- 0 569 173
WO-A-98/27660

Note: Within nine months from the publication of the mention of the grant of the European patent, any person may give notice to the European Patent Office of opposition to the European patent granted. Notice of opposition shall be filed in a written reasoned statement. It shall not be deemed to have been filed until the opposition fee has been paid. (Art. 99(1) European Patent Convention).
Description

FIELD OF THE INVENTION

[0001] The present invention is related to packet switching.

BACKGROUND OF THE INVENTION

[0002] In BFS Memory Controller, Wide Cache buffer structure is used where multiple packets are packed within one memory word to optimize buffer access bandwidth. With this, and because BFS can switch packets of different lengths, packet boundary information is lost in the wide cache buffer. If the packet boundary information (i.e. the packet length calculated by Aggregators) were to be sent on a different bus to Separators, (which need this to extract packets and send them to different Port Cards), then the Memory Controllers have to take this on a bus independent of data from Aggregators, Queue it up independent of data, and send it out to Separators on a bus independent of data. Also, within the Memory Controllers data queue link lists, and length information queue link lists have to be synchronized.

[0003] EP 1,009,132 discloses a typical prior art switch for switching packets having variable length. EP 0,569,173 discloses a high-speed packet switch that receives packets simultaneously from a number of serial input links, and provides them simultaneously to multiple serial output links.

SUMMARY OF THE INVENTION

[0004] The present invention pertains to a switch for switching packets. According to the present invention, there is provided a switch for switching packets, each packet having a length, as defined in claim 1.

[0005] According to another aspect of the present invention, there is provided a method for switching packets having a length, as defined in claim 7.

BRIEF DESCRIPTION OF THE DRAWINGS

[0006] In the accompanying drawings, the preferred embodiment of the invention and preferred methods of practicing the invention are illustrated in which:

Figure 1 is a schematic representation of packet striping in the switch of the present invention.

Figure 2 is a schematic representation of an 0C 48 port card.

Figure 3 is a schematic representation of a concatenated network blade.

Figures 4a and 4b are schematic representations regarding the connectivity of the fabric ASICs.

Figure 5 is a schematic representation of sync pulse distribution.

Figure 6 is a schematic representation regarding the relationship between transmit and receive sequence counters for the separator and unstriper, respectively.

Figure 7 is a schematic representation of a switch of the present invention.

Figure 8 is a schematic representation of a packet with a length indicator.

DETAILED DESCRIPTION

[0007] Referring now to the drawings wherein like reference numerals refer to similar or identical parts throughout the several views, and more specifically to figure 7 thereof, there is shown a switch 10 for switching packets 11. Each packet 11 has a length. The switch 10 comprises a port card 12 which receives packets 11 from and sends packets 11 to a network 16. The switch 10 comprises fabrics 14 connected to the port card 12 which switch the packets 11. Each fabric 14 has a memory mechanism 18. Each fabric 14 has a mechanism for determining the length of each packet 11 received by the fabric 14 and placing a length indicator 22 with the packet 11 so when the packet 11 is stored in the memory mechanism 18, the determining mechanism 20 can identify from the length indicator 22 how long the packet 11 is and where the packet 11 ends in the memory mechanism 18.
The present invention pertains to a method for switching packets 11 having a length. The method comprises the steps of receiving a packet 11 at a port card 12 of a switch 10. Then there is the step of sending fragments 26 of the packet 11 to the fabric 14 of the switch 10. Next there is the step of receiving the fragments 26 of the packet 11 at the fabrics 14 of the switch 10. Then there is the step of measuring the length of the packet 11 at each fabric 14 from the fragments 26 of the packet 11 received at each fabric 14. Next there is the step of appending a length indicator 22 to the packet 11. Then there is the step of determining where the packet 11 ends from the length indicator 22 of the packet 11.

In one approach, the packet 11 length can be maintained in a link list independent of the packet 11 data. This would require that another wide cache buffer structure is used, where each word 34 can hold length information 28 for up to N packets, where N is the maximum number packets that could start in one word 34 of the packet data buffer. With wide range of packet sizes that are supported by BFS (from 40 byte packet to 64K byte packet), N has to be computed based on the smallest size packets that can be put together into one word 34 of the wide data buffer. But, on average if the packets were larger than the smallest size packet, then most of the memory 32 in the buffer used to store length information 28 will be wasted. Also, since packet 11 length and packet 11 data has to be sent at the same time from Memory Controllers to Separators, the link list handling length buffer has to be synchronized (kept lock-step) with the link list handling packet 11 data buffer. This approach also requires to have a separate bus for transferring the length information 28 independently across multiple physical devices.
In the operation of the invention, by appending packet 11 length information 28 to the beginning of a packet 11 in the Aggregators, and sending it as one stream through Memory Controllers and to Separators, Memory Controllers doesn't have to handle packet 11 length and packet 11 data separately. This helps Memory controller 30 design, as it doesn't have to maintain separate data and length link lists for same packet 11 stream. Also, this saves lot of memory 32, as because of the wide cache memory approach for data, length memory also has to be similarly implemented. With different packet 11 sizes, on average, lot of this length memory could be wasted.

In a preferred approach, packet 11 length information 28 is queued and transmitted along with the packet 11 data. In this case, packet 11 length information 28 (which is always a fixed number of bits) is attached to the beginning of each packet 11 data. Then this entity containing both packet 11 length and data is queued as one. Even in this approach, multiple packets are packed together in each word 34 of the wide cache buffer. When packets are read from this buffer and sent to Separator ASIC, it will extract the length field for the first packet 11, and based on that decides where the first packet 11 ends in the data stream, as the length field gives the number of bits of packet 11 length and data together. The bit after the end of the first packet 11 data is the start bit of the second packet 11 length field. Then again based on this length value end of the second packet 11 data, and start of third packet 11 length is extracted. This way, all packets can be extracted from the combined stream.

The switch uses RAID techniques to increase overall switch bandwidth while minimizing individual fabric bandwidth. In the switch architecture, all data is distributed evenly across all fabrics so the switch adds bandwidth by adding fabrics and the fabric need not increase its bandwidth capacity as the switch increases bandwidth capacity.

Each fabric provides 40G of switching bandwidth and the system supports 1, 2, 3, 4, 6, or 12 fabrics, exclusive of the redundant/spare fabric. In other words, the switch can be a 40G, 80G, 120G, 160G, 240G, or 480G switch depending on how many fabrics are installed.

A portcard provides 10G of port bandwidth. For every 4 portcards, there needs to be 1 fabric. The switch architecture does not support arbitrary installations of portcards and fabrics.

The fabric ASICs support both cells and packets. As a whole, the switch takes a "receiver make right" approach where the egress path on ATM blades must segment frames to cells and the egress path on frame blades must perform reassembly of cells into packets.

There are currently eight switch ASICs that are used in the switch:

1. Striper - The Striper resides on the portcard and SCP-IM. It formats the data into a 12 bit data stream, appends a checkword, splits the data stream across the N, non-spare fabrics in the system, generates a parity stripe of width equal to the stripes going to the other fabric, and sends the N+1 data streams out to the backplane.

2. Unstriper - The Unstriper is the other portcard ASIC in the switch architecture. It receives data stripes from all the fabrics in the system. It then reconstructs the original data stream using the checkword and parity stripe to perform error detection and correction.

3. Aggregator - The Aggregator takes the data streams and routewords from the Stripers and multiplexes them into a single input stream to the Memory Controller.

4. Memory Controller - The Memory controller implements the queueing and dequeueing mechanisms of the switch. This includes the proprietary wide memory interface to achieve the simultaneous en-/de-queueing of multiple cells of data per clock cycle. The dequeueing side of the Memory Controller runs at 80Gbps compared to 40Gbps in order to make the bulk of the queueing and shaping of connections occur on the portcards.

5. Separator - The Separator implements the inverse operation of the Aggregator. The data stream from the Memory Controller is demultiplexed into multiple streams of data and forwarded to the appropriate Unstriper ASIC. Included in the interface to the Unstriper is a queue and flow control handshaking.

There are 3 different views one can take of the connections between the fabric: physical, logical, and "active."

Physically, the connections between the portcards and the fabrics are all gigabit speed differential pair serial links. This is strictly an implementation issue to reduce the number of signals going over the backplane. The "active" perspective looks at a single switch configuration, or it may be thought of as a snapshot of how data is being processed at a given moment. The interface between the fabric ASIC on the portcards and the fabrics is effectively 12 bits wide. Those 12 bits are evenly distributed ("striped") across 1, 2, 3, 4, 6, or 12 fabrics based on how the fabric ASICs are configured. The "active" perspective refers to the number of bits being processed by each fabric in the current configuration which is exactly 12 divided by the number of fabrics.

The logical perspective can be viewed as the union or max function of all the possible active configurations. Fabric slot #1 can, depending on configuration, be processing 12, 6, 4, 3, 2, or 1 bits of the data from a single Striper.
and is therefore drawn with a 12 bit bus. In contrast, fabric slot #3 can only be used to process 4, 3, 2, or 1 bits from a single Striper and is therefore drawn with a 4 bit bus.

[0026] Unlike previous switches, the switch really doesn’t have a concept of a software controllable fabric redundancy mode. The fabric ASICs implement N+1 redundancy without any intervention as long as the spare fabric is installed.

[0027] As far as what does it provide; N+1 redundancy means that the hardware will automatically detect and correct a single failure without the loss of any data.

[0028] The way the redundancy works is fairly simple, but to make it even simpler to understand a specific case of a 120G switch is used which has 3 fabrics (A, B, and C) plus a spare (S). The Striper takes the 12 bit bus and first generates a checkword which gets appended to the data unit (cell or frame). The data unit and checkword are then split into a 4-bit-per-clock-cycle data stripe for each of the A, B, and C fabrics (A₃ A₂ A₁ A₀, B₃ B₂ B₁ B₀, and C₃ C₂ C₁ C₀). These stripes are then used to produce the stripe for the spare fabric S₃ S₂ S₁ S₀ where Sₙ = Aₙ XOR Bₙ XOR Cₙ and these 4 stripes are sent to their corresponding fabrics. On the other side of the fabrics, the Unstriper receives 4 4-bit stripes from A, B, C, and S. All possible combinations of 3 fabrics (ABC, ABS, ASC, and SBC) are then used to reconstruct a "tentative" 12-bit data stream. A checkword is then calculated for each of the 4 tentative streams and the calculated checkword compared to the checkword at the end of the data unit. If no error occurred in transit, then all 4 streams will have checkword matches and the ABC stream will be forwarded to the Unstriper output. If a (single) error occurred, only one checkword match will exist and the stream with the match will be forwarded off chip and the Unstriper will identify the faulty fabric stripe.

[0029] For different switch configurations, i.e. 1, 2, 4, 6, or 12 fabrics, the algorithm is the same but the stripe width changes.

[0030] If 2 fabrics fail, all data running through the switch will almost certainly be corrupted.

[0031] The fabric slots are numbered and must be populated in ascending order. Also, the spare fabric is a specific slot so populating fabric slots 1, 2, 3, and 4 is different than populating fabric slots 1, 2, 3, and the spare. The former is a 160G switch without redundancy and the latter is 120G with redundancy.

[0032] Firstly, the ASICs are constructed and the backplane connected such that the use of a certain portcard slots requires there to be at least a certain minimum number of fabrics installed, not including the spare. This relationship is shown in Table 0.

[0033] In addition, the APS redundancy within the switch is limited to specifically paired portcards. Portcards 1 and 2 are paired, 3 and 4 are paired, and so on through portcards 47 and 48. This means that if APS redundancy is required, the paired slots must be populated together.

[0034] To give a simple example, take a configuration with 2 portcards and only 1 fabric. If the user does not want to use APS redundancy, then the 2 portcards can be installed in any two of portcard slots 1 through 4. If APS redundancy is desired, then the two portcards must be installed either in slots 1 and 2 or slots 3 and 4.

[0035] To add capacity, add the new fabric(s), wait for the switch to recognize the change and reconfigure the system to stripe across the new number of fabrics. Install the new portcards.

[0036] Note that it is not technically necessary to have the full 4 portcards per fabric. The switch will work properly with 3 fabrics installed and a single portcard in slot 12. This isn’t cost efficient, but it will work.

[0037] To remove capacity, reverse the adding capacity procedure.

[0038] If the switch is oversubscribed, i.e. install 8 portcards and only one fabric.

[0039] It should only come about as the result of improperly upgrading the switch or a system failure of some sort. The reality is that one of two things will occur, depending on how this situation arises. If the switch is configured as a 40G switch and the portcards are added before the fabric, then the 5th through 8th portcards will be dead. If the switch is configured as 80G non-redundant switch and the second fabric fails or is removed then all data through the switch.
The Striper ASIC accepts data from the ingress port via the Input Bus (BIB) (also known as DIN_ST_bl_ch bus). The Striper ASIC sends data to the unstriper ASIC via the Output Bus (BOB) between the unstriper ASIC and the egress blade ASIC such as Vortex and the Output Bus (BOB) between the unstriper ASIC and the egress blade ASIC such as Trident. The Striper ASIC accepts data from the ingress port via the Input Bus (BIB) (also known as DIN_ST_bl_ch bus).
[0057] This bus can either operate as 4 separate 32 bit input busses (4xOC48c) or a single 128 bit wide data bus with a common set of control lines to all stripers. This bus supports either cells or packets based on software configuration of the stripper chip.

[0058] The unstriper ASIC sends data to the egress port via Output Bus (BOB) (also known as DOUT_UN_BI_ch bus), which is a 64 (or 256) bit data bus that can support either cell or packet. It consists of the following signals:

[0059] This bus can either operate as 4 separate 32 bit input busses (4xOC48c) or a single 128 bit wide data bus with a common set of control lines from all Unstripers. This bus supports either cells or packets based on software configuration of the unstriper chip.

[0060] The Synchronizer has two main purposes. The first purpose is to maintain logical cell/packet or datagram ordering across all fabrics. On the fabric ingress interface, datagrams arriving at more than one fabric from one port card's channels need to be processed in the same order across all fabrics. The Synchronizer's second purpose is to have a port card's egress channel re-assemble all segments or stripes of a datagram that belong together even though the datagram segments are being sent from more than one fabric and can arrive at the blade's egress inputs at different times. This mechanism needs to be maintained in a system that will have different net delays and varying amounts of clock drift between blades and fabrics.

[0061] The switch uses a system of a synchronized windows where start information is transmit around the system. Each transmitter and receiver can look at relative clock counts from the last resynch indication to synchronize data from multiple sources. The receiver will delay the receipt of data which is the first clock cycle of data in a synch period until a programmable delay after it receives the global synch indication. At this point, all data is considered to have been received simultaneously and fixed ordering is applied. Even though the delays for packet 0 and cell 0 caused them to be seen at the receivers in different orders due to delays through the box, the resulting ordering of both streams at receive time = 1 is the same. Packet 0, Cell 0 based on the physical bus from which they were received.

[0062] Multiple cells or packets can be sent in one counter tick. All destinations will order all cells from the first interface before moving onto the second interface and so on. This cell synchronization technique is used on all cell interfaces. Differing resolutions are required on some interfaces.

[0063] The Synchronizer consists of two main blocks, mainly, the transmitter and receiver. The transmitter block will reside in the Stripper and Separator ASICs and the receiver block will reside in the Aggregator and Unstriper ASICs. The receiver in the Aggregator will handle up to 24(6 port cards x 4 channels) input lanes. The receiver in the Unstriper will handle up to 13 (12 fabrics + 1 parity fabric) input lanes.

[0064] When a sync pulse is received, the transmitter first calculates the number of clock cycles it is fast (denoted as N clocks).

[0065] The transmit synchronizer will interrupt the output stream and transmit N K characters indicating it is locking down. At the end of the lockdown sequence, the transmitter transmits a K character indicating that valid data will start on the next clock cycle. This next cycle valid indication is used by the receivers to synchronize traffic from all sources.

[0066] At the next end of transfer, the transmitter will then insert at least one idle on the interface. These idles allow the 10 bit decoders to correctly resynchronize to the 10 bit serial code window if they fall out of sync.

[0067] The receive synchronizer receives the global synch pulse and delays the synch pulse by a programmed number (which is programmed based on the maximum amount of transport delay a physical box can have). After delaying the synch pulse, the receiver will then consider the clock cycle immediately after the synch character to be eligible to be received. Data is then received every clock cycle until the next synch character is seen on the input stream. This data is not considered to be eligible for receipt until the delayed global synch pulse is seen.

[0068] Since transmitters and receivers will be on different physical boards and clocked by different oscillators, clock speed differences will exist between them. To bound the number of clock cycles between different transmitters and receivers, a global synch pulse is used at the system level to resynchronize all sequence counters. Each chip is programmed to ensure that under all valid clock skews, each transmitter and receiver will think that it is fast by at least one clock cycle. Each chip then waits for the appropriate number of clock cycles they are into their current sync_pulse_window. This ensure that all sources run N*sync_pulse_window valid clock cycles between synch pulses.

[0069] As an example, the synch pulse window could be programmed to 100 clocks, and the synch pulses sent out at a nominal rate of a synch pulse every 10,000 clocks. Based on a worst case drifts for both the synch pulse transmitter clocks and the synch pulse receiver clocks, there may actually be 9,995 to 10,005 clocks at the receiver for 10,000 clocks on the synch pulse transmitter. In this case, the synch pulse transmitter would be programmed to send out synch pulses every 10,006 clock cycles. The 10,006 clocks guarantees that all receivers must be in their next window. A receiver with a fast clock may have actually seen 10,012 clocks if the synch pulse transmitter has a slow clock. Since the synch pulse was received 12 clock cycles into the synch pulse window, the chip would delay for 12 clock cycles.

[0070] When a port card or fabric is not present or has just been inserted and either of them is supposed to be driving the inputs of a receive synchronizer, the writing of data to the particular input FIFO will be inhibited since the input
clock will not be present or unstable and the status of the data lines will be unknown. When the port card or fabric is inserted, software must come in and enable the input to the byte lane to allow data from that source to be enabled. Writes to the input FIFO will be enabled. It is assumed that, the enable signal will be asserted after the data, routeword and clock from the port card or fabric are stable.

[0071] At a system level, there will be a primary and secondary sync pulse transmitter residing on two separate fabrics. There will also be a sync pulse receiver on each fabric and blade. This can be seen in Figure 5. A primary sync pulse transmitters will be a free-running sync pulse generator and a secondary sync pulse transmitter will synchronize its sync pulse to the primary. The sync pulse receivers will receive both primary and secondary sync pulses and based on an error checking algorithm, will select the correct sync pulse to forward on to the ASICs residing on that board.

The sync pulse receiver will guarantee that a sync pulse is only forwarded to the rest of the board if the sync pulse from the sync pulse transmitters falls within its own sequence "0" count. For example, the sync pulse receiver and an Unstripper ASIC will both reside on the same Blade. The sync pulse receiver and the receive synchronizer in the Unstripper will be clocked from the same crystal oscillator, so no clock drift should be present between the clocks used to increment the internal sequence counters. The receive synchronizer will require that the sync pulse it receives will always reside in the "0" count window.

[0072] If the sync pulse receiver determines that the primary sync pulse transmitter is out of sync, it will switch over to the secondary sync pulse transmitter source. The secondary sync pulse transmitter will also determine that the primary sync pulse transmitter is out of sync and will start generating its own sync pulse independently of the primary sync pulse transmitter. This is the secondary sync pulse transmitter's primary mode of operation. If the sync pulse receiver determines that the primary sync pulse transmitter has become in sync once again, it will switch to the primary side. The secondary sync pulse transmitter will also determine that the primary sync pulse transmitter has become in sync once again and will switch back to a secondary mode. In the secondary mode, it will sync up its own sync pulse to the primary sync pulse. The sync pulse receiver will have less tolerance in its sync pulse filtering mechanism than the secondary sync pulse transmitter. The sync pulse receiver will switch over more quickly than the secondary sync pulse transmitter. This is done to ensure that all receiver synchronizers will have switched over to using the secondary sync pulse transmitter source before the secondary sync pulse transmitter switches over to a primary mode.

[0073] Figure 5 shows sync pulse distribution.

[0074] In order to lockdown the backplane transmission from a fabric by the number of clock cycles indicated in the sync calculation, the entire fabric must effectively freeze for that many clock cycles to ensure that the same enqueueing and dequeueing decisions stay in sync. This requires support in each of the fabric ASICs. Lockdown stops all functionality, including special functions like queue resynch.

[0075] The sync signal from the sync pulse receiver is distributed to all ASICs. Each fabric ASIC contains a counter in the core clock domain that counts clock cycles between global sync pulses. After the sync pulse is received, each ASIC calculates the number of clock cycles it is fast. Because the global sync is not transferred with its own clock, the calculated lockdown cycle value may not be the same for all ASICs on the same fabric. This difference is accounted for by keeping all interface FIFOs at a depth where they can tolerate the maximum skew of lockdown counts.

[0076] Lockdown cycles on all chips are always inserted at the same logical point relative to the beginning of the last sequence of "useful" (non-lockdown) cycles. That is, every chip will always execute the same number of "useful" cycles between lockdown events, even though the number of lockdown cycles varies.

[0077] Lockdown may occur at different times on different chips. All fabric input FIFOs are initially set up such that lockdown can occur on either side of the FIFO first without the FIFO running dry or overflowing. On each chip-chip interface, there is a sync FIFO to account for lockdown cycles (as well as board trace lengths and clock skews). The transmitter signals lockdown while it is locked down. The receiver does not push during indicated cycles, and does not pop during its own lockdown. The FIFO depth will vary depending on which chip locks down first, but the variation is bounded by the maximum number of lockdown cycles. The number of lockdown cycles a particular chip sees during one global sync period may vary, but they will all have the same number of useful cycles. The total number of lockdown cycles each chip on a particular fabric sees will be the same, within a bounded tolerance.

[0078] The Aggregator core clock domain completely stops for the lockdown duration - all flops and memory hold their state. Input FIFOs are allowed to build up. Lockdown bus cycles are inserted in the output queues. Exactly when the core lockdown is executed is dictated by when DOUT_AG bus protocol allows lockdown cycles to be inserted. DOUT_AG lockdown cycles are indicated on the DestID bus.

[0079] The memory controller must lockdown all flops for the appropriate number of cycles. To reduce impact to the silicon area in the memory controller, a technique called propagated lockdown is used.

[0080] The on-fabric chip-to-chip synchronization is executed at every sync pulse. While some sync error detecting capability may exist in some of the ASICs, it is the Unstriper's job to detect fabric synchronization errors and to remove the offending fabric. The chip-to-chip synchronization is a cascaded function that is done before any packet flow is enabled on the fabric. The synchronization flows from the Aggregator to the Memory Controller, to the Separator, and back to the Memory Controller. After the system reset, the Aggregators wait for the first global sync signal. When
received, each Aggregator transmits a local sync command (value 0x2) on the DestID bus to each Memory Controller.

[0081] The striping function assigns bits from incoming data streams to individual fabrics. Two items were optimized in deriving the striping assignment:

1. Backplane efficiency should be optimized for OC48 and OC192.
2. Backplane interconnection should not be significantly altered for OC192 operation.

[0082] These were traded off against additional muxing legs for the striper and unstriper ASICs. Irregardless of the optimization, the switch must have the same data format in the memory controller for both OC48 and OC192.

[0083] Backplane efficiency requires that minimal padding be added when forming the backplane busses. Given the 12 bit backplane bus for OC48 and the 48 bit backplane bus for OC192, an optimal assignment requires that the number of unused bits for a transfer to be equal to (number_of_bytes *8)/bus_width where "/" is integer division. For OC48, the bus can have 0, 4 or 8 unutilized bits. For OC192 the bus can have 0, 8, 16, 24, 32, or 40 unutilized bits.

[0084] This means that no bit can shift between 12 bit boundaries or else OC48 padding will not be optimal for certain packet lengths.

[0085] For OC192c, maximum bandwidth utilization means that each striper must receive the same number of bits (which implies bit interleaving into the striper). When combined with the same backplane interconnection, this implies that in OC192c, each stripe must have exactly the correct number of bits come from each striper which has 1/4 of the bits.

[0086] For the purpose of assigning data bits to fabrics, a 48 bit frame is used. Inside the striper is a FIFO which is written 32 bits wide at 80-100 MHz and read 24 bits wide at 125 MHz. Three 32 bit words will yield four 24 bit words. Each pair of 24 bit words is treated as a 48 bit frame. The assignments between bits and fabrics depends on the number of fabrics.
### TABLE 11: Bit striping function

<table>
<thead>
<tr>
<th></th>
<th>Fab 0</th>
<th>Fab 1</th>
<th>Fab 2</th>
<th>Fab 3</th>
<th>Fab 4</th>
<th>Fab 5</th>
<th>Fab 6</th>
<th>Fab 7</th>
<th>Fab 8</th>
<th>Fab 9</th>
<th>Fab 10</th>
<th>Fab 11</th>
</tr>
</thead>
<tbody>
<tr>
<td>0:11</td>
<td>0:11</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1 fab</td>
<td>12:23</td>
<td>12:23</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>24:35</td>
<td>24:35</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>36:47</td>
<td>36:47</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0:11</td>
<td>0, 2, 5, 7, 8, 10</td>
<td>1, 3, 4, 6, 9, 11</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2 fab</td>
<td>12:23</td>
<td>13, 15, 16, 18, 21</td>
<td>12, 14, 17, 19, 20, 22</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>24:35</td>
<td>+24 to 0:11</td>
<td>+24 to 0:11</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>36:47</td>
<td>+24 to 12:23</td>
<td>+24 to 12:23</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0:11</td>
<td>0, 3, 5, 10</td>
<td>2, 4, 7, 9</td>
<td>1, 6, 8, 11</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3 fab</td>
<td>12:23</td>
<td>15, 17, 22, 13</td>
<td>14, 16, 19, 21</td>
<td>13, 18, 20, 23</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>24:35</td>
<td>+24 to 0:11</td>
<td>+24 to 0:11</td>
<td>+24 to 0:11</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>36:47</td>
<td>+24 to 12:23</td>
<td>+24 to 12:23</td>
<td>+24 to 12:23</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0:11</td>
<td>0, 5, 10</td>
<td>3, 4, 9</td>
<td>2, 7, 8</td>
<td>1, 6, 11</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4 fab</td>
<td>12:23</td>
<td>15, 16, 21</td>
<td>14, 19, 20</td>
<td>13, 18, 23</td>
<td>12, 17, 22</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>24:35</td>
<td>26, 31, 32</td>
<td>25, 30, 35</td>
<td>24, 29, 34</td>
<td>27, 28, 33</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>36:47</td>
<td>37, 42, 47</td>
<td>36, 41, 46</td>
<td>39, 40, 43</td>
<td>38, 43, 44</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0:11</td>
<td>0, 11</td>
<td>1, 4</td>
<td>5, 8</td>
<td>2, 9</td>
<td>3, 6</td>
<td>7, 10</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6 fab</td>
<td>12:23</td>
<td>14, 21</td>
<td>15, 18</td>
<td>19, 22</td>
<td>12, 23</td>
<td>13, 16</td>
<td>17, 20</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>24:35</td>
<td>+24 to 0:11</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>36:47</td>
<td>+24 to 12:23</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0:11</td>
<td>0</td>
<td>4</td>
<td>8</td>
<td>1</td>
<td>5</td>
<td>9</td>
<td>2</td>
<td>6</td>
<td>10</td>
<td>3</td>
<td>7</td>
<td>11</td>
</tr>
<tr>
<td>12 fab</td>
<td>12:23</td>
<td>15</td>
<td>19</td>
<td>23</td>
<td>12</td>
<td>16</td>
<td>20</td>
<td>13</td>
<td>17</td>
<td>21</td>
<td>14</td>
<td>18</td>
</tr>
<tr>
<td>24:35</td>
<td>26</td>
<td>30</td>
<td>34</td>
<td>27</td>
<td>31</td>
<td>35</td>
<td>24</td>
<td>28</td>
<td>32</td>
<td>25</td>
<td>29</td>
<td>33</td>
</tr>
<tr>
<td>36:47</td>
<td>37</td>
<td>41</td>
<td>45</td>
<td>38</td>
<td>42</td>
<td>46</td>
<td>39</td>
<td>43</td>
<td>47</td>
<td>37</td>
<td>40</td>
<td>44</td>
</tr>
</tbody>
</table>
The following tables give the byte lanes which are read first in the aggregator and written to first in the separator. The four channels are notated A,B,C,D. The different fabrics have different read/write order of the channels to allow for all busses to be fully utilized.

One fabric-40G

The next table gives the interface read order for the aggregator.

<table>
<thead>
<tr>
<th>Fabric</th>
<th>1st</th>
<th>2nd</th>
<th>3rd</th>
<th>4th</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>A</td>
<td>B</td>
<td>C</td>
<td>D</td>
</tr>
<tr>
<td>Par</td>
<td>A</td>
<td>B</td>
<td>C</td>
<td>D</td>
</tr>
</tbody>
</table>

Two fabric-80G

<table>
<thead>
<tr>
<th>Fabric</th>
<th>1st</th>
<th>2nd</th>
<th>3rd</th>
<th>4th</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>A</td>
<td>C</td>
<td>B</td>
<td>D</td>
</tr>
<tr>
<td>Par</td>
<td>A</td>
<td>C</td>
<td>B</td>
<td>D</td>
</tr>
<tr>
<td>1</td>
<td>B</td>
<td>D</td>
<td>A</td>
<td>C</td>
</tr>
</tbody>
</table>

120G

<table>
<thead>
<tr>
<th>Fabric</th>
<th>1st</th>
<th>2nd</th>
<th>3rd</th>
<th>4th</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>A</td>
<td>D</td>
<td>B</td>
<td>C</td>
</tr>
<tr>
<td>Par</td>
<td>A</td>
<td>D</td>
<td>B</td>
<td>C</td>
</tr>
<tr>
<td>1</td>
<td>C</td>
<td>A</td>
<td>D</td>
<td>B</td>
</tr>
<tr>
<td>2</td>
<td>B</td>
<td>C</td>
<td>A</td>
<td>D</td>
</tr>
</tbody>
</table>

Four fabric-160G

<table>
<thead>
<tr>
<th>Fabric</th>
<th>1st</th>
<th>2nd</th>
<th>3rd</th>
<th>4th</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>A</td>
<td>B</td>
<td>C</td>
<td>D</td>
</tr>
<tr>
<td>Par</td>
<td>A</td>
<td>B</td>
<td>C</td>
<td>D</td>
</tr>
<tr>
<td>1</td>
<td>D</td>
<td>A</td>
<td>B</td>
<td>C</td>
</tr>
<tr>
<td>2</td>
<td>C</td>
<td>D</td>
<td>A</td>
<td>B</td>
</tr>
<tr>
<td>3</td>
<td>B</td>
<td>C</td>
<td>D</td>
<td>A</td>
</tr>
</tbody>
</table>

Six fabric-240 G

<table>
<thead>
<tr>
<th>Fabric</th>
<th>1st</th>
<th>2nd</th>
<th>3rd</th>
<th>4th</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>A</td>
<td>D</td>
<td>C</td>
<td>B</td>
</tr>
<tr>
<td>1</td>
<td>B</td>
<td>A</td>
<td>D</td>
<td>C</td>
</tr>
</tbody>
</table>
Interfaces to the gigabit transceivers will utilize the transceiver bus as a split bus with two separate routeword and data busses. The routeword bus will be a fixed size (2 bits for OC48 ingress, 4 bits for OC48 egress, 8 bits for OC192 ingress and 16 bits for OC192 egress), the data bus is a variable sized bus. The transmit order will always have routeword bits at fixed locations. Every striping configuration has one transceiver that is used to talk to a destination in all valid configurations. That transceiver will be used to send both routeword busses and to start sending the data.

The backplane interface is physically implemented using interfaces to the backplane transceivers. The bus for both ingress and egress is viewed as being composed of two halves, each with routeword data. The two bus halves may have information on separate packets if the first bus half ends a packet.

For example, an OC48 interface going to the fabrics locally speaking has 24 data bits and 2 routeword bits. This bus will be utilized acting as if it has 2x (12 bit data bus + 1 bit routeword bus). The two bus halves are referred to as A and B. Bus A is the first data, followed by bus B. A packet can start on either bus A or B and end on either bus A or B.

In mapping data bits and routeword bits to transceiver bits, the bus bits are interleaved. This ensures that all transceivers should have the same valid/invalid status, even if the striping amount changes. Routewords should be interpreted with bus A appearing before bus B.

The bus A/Bus B concept closely corresponds to having interfaces between chips.

All backplane busses support fragmentation of data. The protocol used marks the last transfer (via the final segment bit in the routeword). All transfers which are not final segment need to utilize the entire bus width, even if that is not an even number of bytes. Any given packet must be striped to the same number of fabrics for all transfers of that packet. If the striping amount is updated in the striper during transmission of a packet, it will only update the striping at the beginning of the next packet.

Each transmitter on the ASICs will have the following I/O for each channel:
- 8 bit data bus, 1 bit clock, 1 bit control.

On the receive side, for channel the ASIC receives:
- a receive clock, 8 bit data bus, 3 bit status bus.

The switch optimizes the transceivers by mapping a transmitter to between 1 and 3 backplane pairs and each receiver with between 1 and 3 backplane pairs. This allows only enough transmitters to support traffic needed in a configuration to be populated on the board while maintaining a complete set of backplane nets. The motivation for this optimization was to reduce the number of transceivers needed.

The optimization was done while still requiring that at any time, two different striping amounts must be supported in the gigabit transceivers. This allows traffic to be enqueued from a striping data to one fabric and a striper striping data to two fabrics at the same time.

Depending on the bus configuration, multiple channels may need to be concatenated together to form one
larger bandwidth pipe (any time there is more than one transceiver in a logical connection). Although quad gbit transceivers can tie 4 channels together, this functionality is not used. Instead the receiving ASIC is responsible for synchronizing between the channels from one source. This is done in the same context as the generic synchronization algorithm.

[0105] The 8b/10b encoding/decoding in the gigabit transceivers allow a number of control events to be sent over the channel. The notation for these control events are K characters and they are numbered based on the encoded 10 bit value. Several of these K characters are used in the chipset. The K characters used and their functions are given in the table below.

**TABLE 12:**

<table>
<thead>
<tr>
<th>K character</th>
<th>Function</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>28.0</td>
<td>Sync indication</td>
<td>Transmitted after lockdown cycles, treated as the prime synchronization event at the receivers</td>
</tr>
<tr>
<td>28.1</td>
<td>Lockdown</td>
<td>Transmitted during lockdown cycles on the backplane</td>
</tr>
<tr>
<td>28.2</td>
<td>Packet Abort</td>
<td>Transmitted to indicate the card is unable to finish the current packet. Current use is limited to a port card being pulled while transmitting traffic</td>
</tr>
<tr>
<td>28.3'</td>
<td>Resynch window</td>
<td>Transmitted by the striper at the start of a synch window if a resynch will be contained in the current sync window</td>
</tr>
<tr>
<td>28.4</td>
<td>BP set</td>
<td>Transmitted by the striper if the bus is currently idle and the value of the bp bit must be set.</td>
</tr>
<tr>
<td>28.5</td>
<td>Idle</td>
<td>Indicates idle condition</td>
</tr>
<tr>
<td>28.6</td>
<td>BP cir</td>
<td>Transmitted by the striper if the bus is currently idle and the bp bit must be cleared.</td>
</tr>
</tbody>
</table>

[0106] The switch has a variable number of data bits supported to each backplane channel depending on the striping configuration for a packet. Within a set of transceivers, data is filled in the following order:

F[fabric]_[oc192 port number] [oc48 port designation (a,b,c,d)] [transceiver_number]

[0107] The chipset implements certain functions which are described here. Most of the functions mentioned here have support in multiple ASICs, so documenting them on an ASIC by ASIC basis does not give a clear understanding of the full scope of the functions required.

[0108] The switch chipset is architected to work with packets up to 64K + 6 bytes long. On the ingress side of the switch, there are busses which are shared between multiple ports. For most packets, they are transmitted without any break from the start of packet to end of packet. However, this approach can lead to large delay variations for delay sensitive traffic. To allow delay sensitive traffic and long traffic to coexist on the same switch fabric, the concept of long packets is introduced. Basically long packets allow chunks of data to be sent to the queueing location, built up at the queueing location on a source basis and then added into the queue all at once when the end of the long packet is transferred. The definition of a long packet is based on the number of bits on each fabric.

[0109] If the switch is running in an environment where Ethernet MTU is maintained throughout the network, long packets will not be seen in a switch greater than 40G in size.

[0110] A wide cache-line shared memory technique is used to store cells/packets in the port/priority queues. The shared memory stores cells/packets continuously so that there is virtually no fragmentation and bandwidth waste in the shared memory.

[0111] There exists multiple queues in the shared memory. They are per-destination and priority based. All cells/packets which have the same output priority and blade/channel ID are stored in the same queue. Cells are always dequeued from the head of the list and enqueued into the tail of the queue. Each cell/packet consists of a portion of the egress route word, a packet length, and variable-length packet data. Cell and packets are stored continuously, i.e., the memory controller itself does not recognize the boundaries of cells/packets for the unicast connections. The packet length is stored for MC packets.

[0112] The multicast port mask memory 64Kx16-bit is used to store the destination port mask for the multicast connections, one entry (or multiple entries) per multicast VC. The port masks of the head multicast connections indicated by the multicast DestID FIFOs are stored internally for the scheduling reference. The port mask memory is retrieved when the port mask of head connection is cleaned and a new head connection is provided.

[0113] APS stands for a Automatic Protection Switching, which is a SONET redundancy standard. To support APS feature in the switch, two output ports on two different port cards send roughly the same traffic. The memory controllers
To support data duplication in the memory controller ASIC, each one of multiple unicast queues has a programmable APS bit. If the APS bit is set to one, a packet is dequeued to both output ports. If the APS bit is set to zero for a port, the unicast queue operates at the normal mode. If a port is configured as an APS slave, then it will read from the queues of the APS master port. For OC48 ports, the APS port is always on the same OC48 port on the adjacent port card.

The shared memory queues in the memory controllers among the fabrics might be out of sync (i.e., same queues among different memory controller ASICs have different depths) due to clock drifts or a newly inserted fabric. It is important to bring the fabric queues to the valid and sync states from any arbitrary states. It is also desirable not to drop cells for any recovery mechanism.

A resynch cell is broadcast to all fabrics (new and existing) to enter the resynch state. Fabrics will attempt to drain all of the traffic received before the resynch cell before queue resynch ends, but no traffic received after the resynch cell is drained until queue resynch ends. A queue resynch ends when one of two events happens:

1. A timer expires.
2. The amount of new traffic (traffic received after the resynch cell) exceeds a threshold.

At the end of queue resynch, all memory controllers will flush any left-over old traffic (traffic received before the queue resynch cell). The freeing operation is fast enough to guarantee that all memory controllers can fill all of memory no matter when the resynch state was entered.

Queue resynch impacts all 3 fabric ASICs. The aggregators must ensure that the FIFOs drain identically after a queue resynch cell. The memory controllers implement the queueing and dropping. The separators need to handle memory controllers dropping traffic and resetting the length parsing state machines when this happens. For details on support of queue resynch in individual ASICs, refer to the chip ADSs.

For the dequeue side, multicast connections have independent 32 tokens per port, each worth up 50-bit data or a complete packet. The head connection and its port mask of a higher priority queue is read out from the connection FIFO and the port mask memory every cycle. A complete packet is isolated from the multicast cache line based on the length field of the head connection. The head packet is sent to all its destination ports. The 8 queue drainers transmit the packet to the separators when there are non-zero multicast tokens are available for the ports. Next head connection will be processed only when the current head packet is sent out to all its ports.

Queue structure can be changed on fly through the fabric resynch cell where the number of priority per port field is used to indicate how many priority queues each port has. The stripper ASIC resides on the network blade.

The following words have reasonably specific meanings in the vocabulary of the switch. Many are mentioned elsewhere, but this is an attempt to bring them together in one place with definitions.

<table>
<thead>
<tr>
<th>Word</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>APS</td>
<td>Automatic Protection Switching. A sonet/sdh standard for implementing redundancy on physical links. For the switch, APS is used to also recover from any detected port card failures.</td>
</tr>
<tr>
<td>Backplane</td>
<td>A generic term referring either to the general process the switch boards use to account for varying transport delays between boards and clock drift or to the logic which implements the TX/RX functionality required for the switch ASICs to account for varying transport delays and clock drifts.</td>
</tr>
<tr>
<td>synch</td>
<td></td>
</tr>
<tr>
<td>BIB</td>
<td>The switch input bus. The bus which is used to pass data to the stripper(s). See also BOB.</td>
</tr>
<tr>
<td>Blade</td>
<td>Another term used for a port card. References to blades should have been eliminated from this document, but some may persist.</td>
</tr>
<tr>
<td>BOB</td>
<td>The switch output bus. The output bus from the stripper which connects to the egress memory controller. See also BIB.</td>
</tr>
<tr>
<td>Egress</td>
<td>This is the routeword which is supplied to the chip after the unstriper. From an internal chipset perspective, the egress routeword is treated as data. See also fabric routeword.</td>
</tr>
<tr>
<td>Routeword</td>
<td>Routeword used by the fabric to determine the output queue. This routeword is not passed outside the unstriper. A significant portion of this routeword is blown away in the fabrics.</td>
</tr>
<tr>
<td>Freeze</td>
<td>Having logic maintain its values during lock-down cycles.</td>
</tr>
</tbody>
</table>
The relationship between the transmit and receive counters can be seen in figure 6.

Although the invention has been described in detail in the foregoing embodiments for the purpose of illustration, it is to be understood that such detail is solely for that purpose and that variations can be made therein by those skilled in the art without departing from the scope of the invention except as it may be described by the following claims.

Claims

1. A switch (10) for switching packets (11), each packet (11) having a length, the switch comprising:

   a port card (12) which receives packets (11) from and sends packets (11) to a network (16); and

   fabrics (14) connected to the port card (12) which switch the packets (11), each fabric (14) having a memory mechanism (18), the switch characterized in that:

   the port card (12) sends stripes of corresponding fragments of each packet to the fabrics, each fabric (14) having a mechanism for determining the length of each packet (11) from the stripes of fragments (26) of the packet (11) received by the fabric (14) and placing a length indicator (22) with the packet (11) so when the packet (11) is stored in the memory mechanism (18), the determining mechanism (20) can identify from the length indicator (22) how long the packet (11) is and where the packet (11) ends in the memory mechanism (18).

2. A switch (10) as described in Claim 1 characterized by the fact that the determining mechanism (20) includes an aggregator (24) which receives the stripes of the packet (11) fragments (26) from the port card (12), determines the packet (11) length and appends packet (11) length information (28) to the beginning of the packet (11) in the length indicator (22).

3. A switch (10) as described in Claim 2 characterized by the fact that the memory mechanism (18) includes a memory controller (30), the aggregator (24) sending the packet (11) with the packet (11) length information (28) to the memory controller (30) which stores the packet (11) with the packet (11) length information (28).

4. A switch (10) as described in Claim 3 characterized by the fact that the memory controller (30) has a memory (32) which has a wide cache buffer structure in which multiple packets (11) are put into one word (34).

5. A switch (10) as described in Claim 4 characterized by the fact that the fabric (14) includes a separator (36) which reads the stripes of fragments (26) of packets (11) from the memory controller (30) and extracts the packet (11)
length information (28) from each stripe of fragments (26) of packet (11) to determine when each packet (11) ends, and sends fragments (26) of the packet (11) to the port card (12), the port card assembling the packet (11) from the stripes of the fragments of the packet the port card (12) receives from the fabrics (14).

6. A switch (10) as described in Claim 5 characterized by the fact that the separator (36) removes the packet (11) length information (28) from each packet (11) before sending any stripes of fragments (26) of each packet (11) to an unstriper (38) of the port card (12).

7. A method for switching packets (11) having a length comprising the steps of:
   
   receiving a packet (11) at a port card (12) of a switch (10);

   sending corresponding fragments (26) of the packet (11) as stripes to fabrics (14) of the switch (10) from the port card;

   receiving the stripes of corresponding fragments (26) of the packet (11) at the fabrics (14) of the switch (10);

   measuring the length of the packet (11) at each fabric (14) from the stripes of fragments (26) of the packet (11) received at each fabric (14);

   appending a length indicator (22) to the packet (11);

   storing the packet (11) with the length indicator (22) in a memory mechanism (18) of the fabric (14);

   reading the packet (11) from the memory mechanism (18); and

   determining where the packet (11) ends from the length indicator (22) of the packet (11).

8. A method as described in Claim 7 characterized by the fact that the step of receiving the stripes of corresponding fragments includes the step of receiving the fragments (26) at an aggregator (24) of each of the fabric (14).

9. A method as described in Claim 8 characterized by the fact that the measuring step includes the step of measuring the length of the packet (11) with the aggregator (24).

10. A method as described in Claim 9 characterized by the fact that the appending step includes the step of appending the length indicator (22) to the packet (11) with the aggregator (24).

11. A method as described in Claim 10 characterized by the fact that the storing step includes the step of storing the packet (11) with the length indicator (22) in a memory controller (30) of the memory mechanism (18).

12. A method as described in Claim 11 characterized by the fact that the reading step includes the step of reading the packet (11) from the memory controller (30) with a separator (36) of the fabric (14).

13. A method as described in Claim 12 characterized by the fact that the determining step includes the step of determining where a packet (11) ends from the length indicator (22) with the separator (36).

14. A method as described and Claim 13 characterized by the fact that after the determining step, there is the step of removing the packet (11) length information (28) with the separator (36).

15. A method as described in Claim 14 characterized by the fact that after the removing step, there is the step of sending stripes of fragments (26) of the packet (11) from the separator (36) to the port card (12).

16. A method as described in Claim 15 characterized by the fact that the step of sending fragments (26) from the separator (36) to the port card (12) includes the step of sending fragments (26) of the packet (11) to the port card (12) in a same logical time with corresponding fragments (26) of the packet (11) from other fabrics (14) to the port card (12).

17. A method as described in Claim 16 characterized by the fact that the storing step includes the step of storing the
fragments (26) of the packet (11) in a memory (32) of the memory controller (30) which has a wide cache buffer structure in which multiple packets (11) are put into one word (34).

18. A method as described in Claim 17 characterized by the fact that after the reading step, there is the step of extracting the packet (11) length information (28) from the packet (11) with the separator (36).

19. A method as described in Claim 18 characterized by the fact that there is a step of receiving the stripes of fragments (26) of the packet (11) from the fabrics (14) with an unstriper (38) of the port card (12).

20. A method as described in Claim 19 characterized by the fact that the sending stripes of fragments (26) of the packet to the fabric (14) step includes the step of sending with a striper (40) of the port card (12) to the aggregator (24) of each fabric (14) the stripes of fragments (26) of the packet (11).

21. A method as described in Claim 20 characterized by the fact that the step of sending fragments (26) to the port card (12) includes the step of sending fragments (26) from the separator (36) to an unstriper (38) of the port card (12).

Patentansprüche

1. Eine Vermittlung (10) zum Vermitteln von Paketen (11), wobei jedes Paket (11) eine Länge aufweist, und wobei die Vermittlung umfasst:

   eine Anschlusskarte (12), die Pakete (11) von einem Netz (16) empfängt und Pakete (11) dorthin sendet; und

   Netzeinrichtungen (14), die mit der Anschlusskarte (12) verbunden sind, die die Pakete (11) vermitteln, wobei jede Netzeinrichtung (14) einen Speichermechanismus (18) aufweist, und die Vermittlung dadurch gekennzeichnet ist, dass:

   die Anschlusskarte (12) Streifen (Stripes) entsprechernder Fragmente eines jeden Pakets zu den Netzeinrichtungen sendet, wobei jede Netzeinrichtung (14) einen Mechanismus aufweist, zum Bestimmen der Länge eines jeden Pakets (11) aus den Streifen von Fragmenten (26) des durch die Netzeinrichtung (14) empfangenen Pakets (11), und zum Anordnen eines Längenindikators (22) mit dem Paket (11), sodass dann, wenn das Paket (11) in dem Speichermechanismus (18) gespeichert wird, der Bestimmungsmechanismus (20) aus dem Längenindikator (22) bestimmen kann, wie lang das Paket (11) ist, und wo das Paket (11) in dem Speichermechanismus (18) endet.


4. Eine Vermittlung (10) nach Anspruch 3, dadurch gekennzeichnet, dass der Speichercontroller (30) einen Speicher (32) aufweist, der eine Breitcachepufferstruktur aufweist, in der mehrere Pakete (11) in ein Wort (34) gefügt sind.


6. Eine Vermittlung (10) nach Anspruch 5, dadurch gekennzeichnet, dass der Separator (36) die Paket- (11) Län-
geninformation 28 von jedem Paket (11) entfernt, bevor er irgendwelche Streifen (Stripes) von Fragmenten (26) jedes Pakets (11) zu einem Unstriper (38) der Anschlusskarte (12) sendet.

7. Ein Verfahren zum Vermitteln von Paketen (11) mit einer Länge, die Schritte umfassend:
Empfangen eines Pakets (11) an einer Anschlusskarte (12) einer Vermittlung (10);
Senden entsprechender Fragmente (26) des Pakets (11) als Streifen (Stripes) von der Anschlusskarte zu Netzeinrichtungen (14) der Vermittlung (10);
Empfangen der Streifen entsprechender Fragmente (26) des Pakets (11) an den Netzeinrichtungen (14) der Vermittlung (10);
Messen der Länge des Pakets (11) an jeder Netzeinrichtung (14) aus den an jeder Netzeinrichtung (14) empfangenen Streifen von Fragmenten (26) des Pakets (11);
Anfügen eines Längenindikators (22) an das Paket (11);
Speichern des Pakets (11) mit dem Längenindikator (22) in einem Speichermechanismus (18) der Netzeinrichtung (14);
Lesen des Pakets (11) von dem Speichermechanismus (18); und
Bestimmen, aus dem Längenindikator (22) des Pakets (11), wo das Paket (11) endet.

8. Ein Verfahren nach Anspruch 7, dadurch gekennzeichnet, dass der Schritt zum Empfangen der Streifen entsprechender Fragmente den Schritt enthält, die Fragmente (26) an einem Aggregator (24) einer jeden Netzeinrichtung (14) zu empfangen.


10. Ein Verfahren nach Anspruch 9, dadurch gekennzeichnet, dass der Anfügeschritt den Schritt umfasst, den Längenindikator (22) an das Paket (11) mit dem Aggregator (24) anzufügen.

11. Ein Verfahren nach Anspruch 10, dadurch gekennzeichnet, dass der Speicherschritt den Schritt umfasst, das Paket (11) mit dem Längenindikator (22) in einem Speichercontroller (30) des Speichermechanismus (18) zu speichern.


17. Ein Verfahren nach Anspruch 16, dadurch gekennzeichnet, dass der Speicherschritt den Schritt umfasst, die
Fragmente (26) des Pakets (11) in einem Speicher (32) des Speichercontrollers (30) zu speichern, welcher eine Breitcachepufferstruktur aufweist, bei der mehrere Pakete (11) in ein Wort (34) gelegt werden.


**Revendications**

1. Commutateur (10) destiné à commuter des paquets (11), chaque paquet (11) ayant une certaine longueur, le commutateur comprenant :

   une carte (12) de port qui reçoit des paquets (11) à partir de, et qui envoie des paquets (11) à un réseau (16) ; et des dispositifs (14) reliés à la carte (12) de port et qui commutent les paquets (11), chaque dispositif (14) ayant un mécanisme (18) de mémoire, le commutateur étant **caractérisé en ce que** :

   la carte (12) de port envoie des bandes de fragments correspondants de chaque paquet aux dispositifs, chaque dispositif (14) ayant un mécanisme pour déterminer la longueur de chaque paquet (11) à partir des bandes de fragments (26) du paquet (11) reçu par le dispositif (14) et pour placer un indicateur (22) de longueur avec le paquet (11) de sorte que, lorsque le paquet (11) est stocké dans le mécanisme (18) de mémoire, le mécanisme (20) de détermination puisse identifier, à partir de l'indicateur (22) de longueur, la longueur du paquet (11) ainsi que l'endroit où le paquet (11) aboutit dans le mécanisme (18) de mémoire.

2. Commutateur (10) selon la description de la revendication 1 **caractérisé par le fait que** le mécanisme (20) de détermination comporte un agrégateur (24) qui reçoit les bandes des fragments (26) de paquet (11) à partir de la carte (12) de port, détermine la longueur du paquet (11) et ajoute des informations (28) de longueur de paquet (11) au début du paquet (11) dans l'indicatrice (22) de longueur.

3. Commutateur (10) selon la description de la revendication 2 **caractérisé par le fait que** le mécanisme (18) de mémoire comporte un contrôleur (30) de mémoire, l'agréageur (24) envoyant le paquet (11) avec les informations (28) de longueur de paquet (11) au contrôleur (30) de mémoire qui stocke le paquet (11) avec les informations (28) de longueur de paquet (11).

4. Commutateur (10) selon la description de la revendication 3 **caractérisé par le fait que** le contrôleur (30) de mémoire possède une mémoire (32) qui possède une structure de tampon d'antémémoire importante dans laquelle des paquets (11) multiples sont disposés dans un seul mot (34).

5. Commutateur (10) selon la description de la revendication 4 **caractérisé par le fait que** le dispositif (14) comporte un séparateur (36) qui lit les bandes de fragments (26) de paquets (11) à partir du contrôleur (30) de mémoire et extrait les informations (28) de longueur de paquet (11) à partir de chaque bande de fragments (26) de paquet (11) pour déterminer le moment où chaque paquet (11) aboutit, et envoie des fragments (26) du paquet (11) à la carte (12) de port, la carte de port assemblant le paquet (11) à partir des bandes des fragments du paquet que la carte (12) de port reçoit à partir des dispositifs (14).

6. Commutateur (10) selon la description de la revendication 5 **caractérisé par le fait que** le séparateur (36) supprime
7. Procédé destiné à commuter des paquets (11) ayant une longueur comprenant les étapes de réception d'un paquet (11) à une carte (12) de port d'un commutateur (10) ; d'envoi de fragments correspondants (26) du paquet (11) en tant que bandes aux dispositifs (14) du commutateur (10) ; de réception des bandes de fragments correspondants (26) du paquet (11) aux dispositifs (14) du commutateur (10) ; de mesure de la longueur du paquet (11) à chaque dispositif (14) à partir des bandes de fragments (26) du paquet (11) reçus à chaque dispositif (14) ; d'ajout d'un indicateur (22) de longueur au paquet (11) ; de stockage du paquet (11) avec l'indicateur (22) de longueur dans un mécanisme (18) de mémoire du dispositif (14) ; de lecture du paquet (11) à partir du mécanisme (18) de mémoire ; et de détermination de l'endroit où le paquet (11) aboutit à partir de l'indicateur (22) de longueur (11) du paquet.

8. Procédé selon la description de la revendication 7 caractérisé par le fait que l'étape de réception des bandes de fragments correspondants comporte l'étape de recevoir des fragments (26) à un agrégateur (24) de chacun des dispositifs (14).

9. Procédé selon la description de la revendication 8 caractérisé par le fait que l'étape de mesure comporte l'étape de mesurer la longueur du paquet (11) avec l'agrégeateur (24) de longueur (11) avec l'agrégeateur (24).

10. Procédé selon la description de la revendication 9 caractérisé par le fait que l'étape d'ajout comporte l'étape d'ajouter l'indicateur (22) de longueur au paquet (11) avec l'agrégeateur (24).

11. Procédé selon la description de la revendication 10 caractérisé par le fait que l'étape de stockage comporte l'étape de stocker le paquet (11) avec l'indicateur (22) de longueur dans un contrôleur (30) de mémoire du mécanisme (18) de mémoire.

12. Procédé selon la description de la revendication 11 caractérisé par le fait que l'étape de lecture comporte l'étape de lire le paquet (11) à partir du contrôleur (30) de mémoire avec un séparateur (36) du dispositif (14).

13. Procédé selon la description de la revendication 12 caractérisé par le fait que l'étape de détermination comporte l'étape de déterminer l'endroit où un paquet (11) aboutit à partir de l'indicateur (22) de longueur avec le séparateur (36) de longueur (11) du paquet.

14. Procédé selon la description de la revendication 13 caractérisé par le fait qu'après l'étape de détermination, il y a l'étape de suppression des informations (28) de longueur de paquet (11) avec le séparateur (36).

15. Procédé selon la description de la revendication 14 caractérisé par le fait qu'après l'étape de suppression, il y a l'étape d'envoi de bandes de fragments (26) du paquet (11) à partir du séparateur (36) à la carte (12) de port.

16. Procédé selon la description de la revendication 15 caractérisé par le fait que l'étape d'envoi de fragments (26) à partir du séparateur (36) à la carte (12) de port comporte l'étape d'envoyer des fragments (26) du paquet (11) à la carte (12) de port dans un même temps logique avec des fragments correspondants (26) du paquet (11) à partir d'autres dispositifs (14) à la carte (12) de port.

17. Procédé selon la description de la revendication 16 caractérisé par le fait que l'étape de stockage comporte l'étape de stocker les fragments (26) du paquet (11) dans une mémoire (32) du contrôleur (30) de mémoire qui possède une structure de tampon d'antémémoire importante dans laquelle des paquets (11) multiples sont disposés dans un seul mot (34).

18. Procédé selon la description de la revendication 17 caractérisé par le fait qu'après l'étape de lecture, il y a l'étape d'extraction des informations (28) de longueur de paquet (11) à partir du paquet (11) avec le séparateur (36).

19. Procédé selon la description de la revendication 18 caractérisé par le fait qu'il y a une étape de réception des informations (28) de longueur de paquet (11) à partir de chaque paquet (11) avant d'envoyer toutes bandes de fragments (26) de chaque paquet (11) à un élément d'enlèvement de bandes (38) de la carte (12) de port.
bandes de fragments (26) du paquet (11) à partir des dispositifs (14) avec un élément d'enlèvement de bandes (38) de la carte (12) de port.

20. Procédé selon la description de la revendication 19 caractérisé par le fait que l'étape d'envoi de bandes de fragments (26) du paquet au dispositif (14) comporte l'étape d'envoyer avec un élément de mise de bandes (40) de la carte (12) de port à l'agrégateur (24) de chaque dispositif (14) les bandes de fragments (26) du paquet (11).

21. Procédé selon la description de la revendication 20 caractérisé par le fait que l'étape d'envoi de fragments (26) à la carte (12) de port comporte l'étape d'envoyer des fragments (26) à partir du séparateur (36) à un élément d'enlèvement de bandes (38) de la carte (12) de port.
FIG. 2
FIG. 3
FIG. 5
NO VALID OR IDLE DATA IS SENT. ONLY LOCK-DOWN CYCLES ARE SENT

SYNC PULSE

SYNC PULSE PERIOD = 22.00

TX K CHAR

LOGICAL CYCLES TRANSMITTED = 22.000

RX CNT

D = SYNC PULSE RECEIVE DELAY = 150
X = COUNTER TICK LENGTH = 250
C = LOCK-DOWN AMOUNT

FIG. 6
FIG. 7

FIG. 8