Data aggregation architectures for single-chip SDH/SONET framers

R. Clauberg

Single-chip SDH/SONET framer architectures are described that permit data aggregation from several line ports. After presenting an overview of the usual parallel approach and an extension thereof that exploits distributed algorithms, we introduce a novel data-multiplexing architecture that should be suitable for accommodating data from a relatively large number of ports in a single device. In combination with the new virtual concatenation feature of SDH/SONET, this architecture should also allow transport of data from high-bandwidth applications over multiple wavelengths or multiple fibers.

1. Introduction

The synchronous digital hierarchy (SDH)/synchronous optical network (SONET) technology [1] is currently the dominant choice for metropolitan-area networks as well as for accessing wavelength division multiplexing networks in wide-area networks. SDH/SONET networks are generally implemented as double, counter-rotating rings to enable automatic protection switching of data from one ring to the other in case of fiber cuts or other localized ring failures. Large networks are then based on a few very large, high-data-rate rings plus many small, lower-data-rate rings. The rings form a complete network through interconnections from one ring to another, and access to different (i.e., non-SDH/SONET) networks through routers. Each ring operates with a fixed data rate. These data rates follow a rigid hierarchy with values of 155.52 Mb/s (STM-1, OC-3), 622.08 Mb/s (STM-4, OC-12), 2.488 Gb/s (STM-16, OC-48), 9.953 Gb/s (STM-64, OC-192), and 39.813 Gb/s (STM-256, OC-768); the term STM denotes a synchronous transport module, and OC denotes an optical channel. **Figure 1** illustrates such a network.

Within a ring, frames are forwarded that contain an overhead section, designated as the section overhead (SOH) and used primarily for network management, and a payload section. The length of a frame is always 125 μs, whereas the number of bytes in the frame depends on the data rate and hence the clock rate applied at the ring. Each frame is immediately followed by another frame; hence the first bit of the next frame directly follows the last bit of the previous frame. The payload section of the frame is organized into one or more virtual containers (VCs). The VCs in turn are organized into a very short overhead section, the path overhead (POH), plus a payload section. The payload section can contain any kind of payload, e.g., IP packets, ATM cells, Ethernet frames, or time-division multiplexed voice signals, but only one kind per VC. Since the same sequence of VCs is repeated in every frame until a change is configured, assigning a VC to a specific application or user implies assigning the corresponding bandwidth to that application or user. There are three main functional elements in an SDH/SONET network that facilitate data transport from an ingress node to an egress node. The first is an add/drop, mapping payload into one of the VCs belonging to a frame at the source node and extracting the entire payload from the virtual container at the destination node. The second is a multiplexer, which multiplexes VCs from several frames into a higher-data-rate SDH/SONET frame on another ring. The third is a digital cross-connect, which

0018-8646/03/$5.00 © 2003 IBM

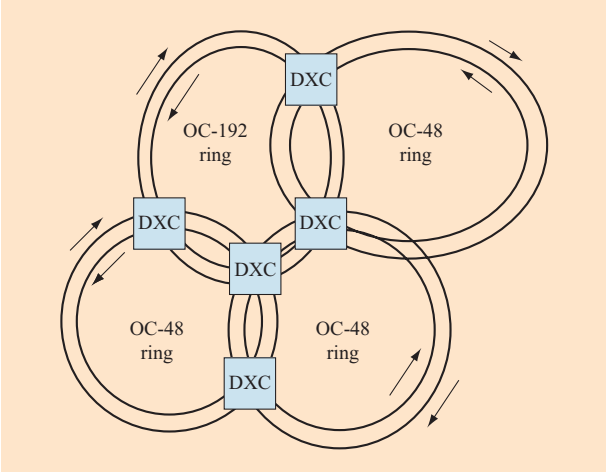

Figure 1

Illustrative SDH/SONET network.

transfers a complete VC from a frame on one ring to a frame on the next ring in its path to the destination node. Only the virtual containers are transferred from one ring to the other; frames are always restricted to a specific ring. The data rate of the frame in the second ring may differ from that in the first ring, but must of course be suitable for the VC. Different combinations of these functional elements can be implemented in a single network element. Most commonly used are add/drop multiplexers (ADMs), which combine the first two functional elements. The transport through the SDH/SONET network is not self-routing, but is based on fixed, configured paths. Hence, the transport is not steered by some specific POH bytes, but exclusively by the sequence number of the VC within a frame. From the payload point of view, only the framers at the source and destination nodes in their add/drop functionality operate directly on the payload; the framers at all intermediate nodes operate exclusively on the frames and the VCs, updating SOH and POH bytes but not analyzing the payload.

Recently a new addition was made to the SDH and SONET standards, extending the definition of virtual concatenation to the range of STS-1 to STS-768 (STM-256) frames. Virtual concatenation in this range now allows bandwidth allocations as integer multiples of 50.1 Mb/s or 150.3 Mb/s. Using virtual concatenation, only the source and destination SDH/SONET nodes need to know that the payload of corresponding VCs must be treated as stored in a concatenated VC. All intermediate nodes need not know this and treat the corresponding VCs as independent. This is in contrast to real concatenation, in which all nodes must treat the corresponding VCs as concatenated. Also in contrast to real concatenation, each single VC still keeps its row of path overhead bytes. Because of the evolution of networks and the need to upgrade all nodes of a ring to a higher data rate simultaneously, many rings are parallel multi-line rings in which a high data capacity is achieved by operating multiple rings in parallel. For example, such a ring may offer four OC-48 rings, each with a data rate of 2.5 Gb/s. The total capacity of the ring is then equivalent to that of an OC-192 ring. For virtual concatenated VCs, this implies that different concatenated VCs may be transmitted over different fibers. At the destination node or a node before this where the VCs converge again, data aggregation is necessary. Virtual concatenation therefore allows sending of data corresponding to a data rate R over connections in which, at least for parts of the connection, R is achieved merely by exploiting several parallel connections each having a data rate smaller than R. Data aggregation in general is a characteristic feature of metropolitan-area networks, where it is often necessary to aggregate data from many lower-speed lines into a higher-speed line. This situation occurs not only in connection with virtual concatenation but also at the edges of a large higher-speed SDH/SONET ring accessed through many lower-speed rings. In general, data aggregation is an important feature of SDH/SONET networks, and the corresponding network components that allow data aggregation in an easy and cost-effective way have a large market potential.

In Section 2, the detailed structure of SDH/SONET frames and their virtual containers is discussed; that highly regular structure is the basis for the data aggregation architectures subsequently described. General data aggregation aspects are discussed in Section 3; data aggregation by multiple parallel units is discussed in Section 4; the novel data-multiplexing architecture is described in Section 5; and concluding remarks are presented in Section 6. Section 2 may be skipped by those sufficiently familiar with SDH/SONET.

2. SDH/SONET frame basics

According to the international SDH standard of the ITU-T, an STM-N frame consists of 9 rows with $9 \times N$ SOH bytes followed by $261 \times N$ payload bytes. The value of N can be $\{1, 4, 16, 64, 256\}$. Transmission of the frame is serial, one row after another. The basic VCs for STM-N frames according to the SDH standard are the VC-4 and the VC-3. The VC-4 is preferred for SDH networks and consists of 261 columns and 9 rows. The first column of a VC-4 is reserved for POH bytes, and the remaining 260 columns are regular payload bytes. The VC-3 has the same structure and the same POH bytes as the VC-4 but contains only 85 columns instead of 261. An STM-N frame contains N byte-interleaved VC-4 "containers." The term

byte-interleaved is used to indicate that for a single row the first byte of the first VC-4 is followed by the first byte of the second VC-4 and so on, until the 261st byte of the $(N - 1)$th VC-4 is followed by the 261st byte of the Nth VC-4. The resulting frame structure is shown in **Figure 2**.

Figure 2 shows that the SOH is divided into three different parts. The first part consists of the first three SOH rows, which are designated as the regenerator section overhead (RSOH). The second part consists of the last five rows, which are designated as the multiplex section overhead (MSOH). The third part is the fourth row of the frame, which contains the administrative unit (AU) pointers. Considering the transmission of the frame in an SDH ring, there are simple signal regenerators to improve signal quality as well as multiplexers where VCs may be transferred from a frame on one ring to one on another ring. A regenerator can update only RSOH bytes; it leaves all other bytes of the frame unchanged. The AU pointers in the fourth row contain the information on where the corresponding VC starts in the frame. Two kinds of AU pointers are relevant for STM-N frames: AU-4 pointers for VC-4s, and AU-3 pointers for VC-3s. A VC-4 plus an AU-4 pointer forms an AU of type 4 (AU-4). A VC-3 extended by a "fixed stuff" column after column 29 and a second fixed stuff column after column 57, plus an AU-3 pointer, forms an AU-3. The AU-4 pointer has the form

| H1 | Y | Y | H2 | 1* | 1* | H3 | H3 | H3 | .

The H1 and H2 bytes encode the pointer value, i.e., the number of bytes between the last H3 byte and the first byte of the VC-4. The Y and 1* bytes have fixed values, and the H3 bytes are reserved for frequency adjustments to correct a small deviation between the system clock rate used for creating the frame and the clock rate with which the VC-4 arrives at this system. If the VC-4 arrives at a slightly higher clock rate than that of the system, the system will, from time to time, write arriving VC bytes into the H3 position to cope with this difference. The H1 and H2 bytes then contain the information that the H3 bytes must be interpreted as VC bytes. Hence, because of the dual role of the H3 bytes, there is no clear separation between SOH and VC bytes for the fourth row, in contrast to what applies for all of the other rows. If the VC-4 arrives at a slightly lower clock rate than that of the system, the first three bytes after the last H3 byte are occasionally filled with empty bytes. Again, the information that these bytes are not real VC bytes is encoded in the H1 and H2 bytes.

The AU-4 pointers of all VCs in the frame are byte-interleaved in exactly the same manner as the VCs. Thus, the fourth row in an STM-N frame with VC-4s starts with N H1 bytes followed by $2 \times N$ Y bytes, $N \times$ H2 bytes, $2 \times N$ 1* bytes, and $3 \times N$ H3 bytes.

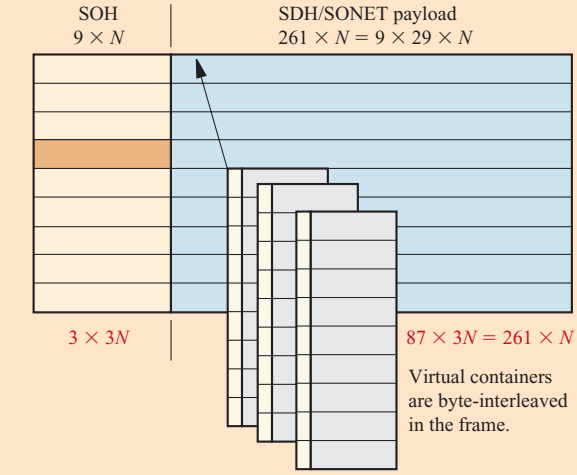

Figure 2

SDH/SONET frame structure.

The North American SONET standard has an identical frame structure for STM-1 and higher-data-rate frames. The notation, however, is STS-3 for STM-1 and STS-3N for STM-N frames, where STS denotes a synchronous transport signal. In addition, the SONET standard offers a smaller VC, the STS-1 synchronous payload envelope (SPE), and the corresponding STS-1 frame. The STS-1 SPE is an 87-column-by-9-row container. The first column of each VC is reserved for path overhead bytes. The STS-1 SPE corresponds to the VC-3 extended by two fixed stuff columns, as described above, except that in the STS-1 SPE they are used for real data. The STS-1 SPE is connected to an AU-3 pointer just as is the VC-3. The AU-3 pointer has the form

| H1 | H2 | H3 |,

where the H1, H2, and H3 bytes have exactly the same meaning as in the AU-4 pointer. A special feature in SDH/SONET is the concatenation of VCs to larger VCs. Three STS-1 SPEs can be concatenated into a single VC-4, and x VC-4s can be concatenated into a single VC-4-xc. Specifically, the AU-4 pointer can be viewed as a pointer related to the concatenation of three STS-1 SPEs with the Y (1*) bytes replacing the H1 (H2) bytes of the second and third STS-1 SPEs. The corresponding H1 and H2 bytes are replaced by the fixed bytes because the positions of these STS-1 SPEs are fixed by the pointer value of the first STS-1 SPE. A VC-4-xc contains one column of path overhead bytes, followed by $(x - 1)$ columns of fixed stuff bytes, followed by $x \times 260$ columns of payload bytes. Concatenation thereby offers a means for allocating a

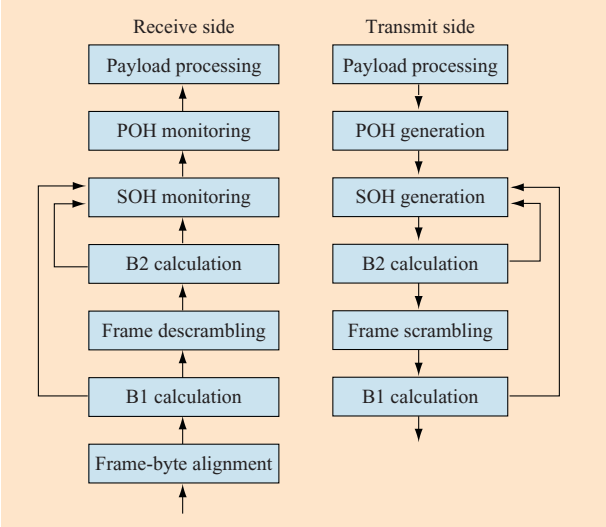

Figure 3

Block diagram of functions in the data path of a simple SDH/SONET framer, and corresponding data flow for *receive* and *transmit* sides of the framer.

larger bandwidth to a specific application or user than that corresponding to a single VC. However, although the standard allows x to be any integer between 1 and N for an STM-N frame, in practice only the values with $x = \{1, 4, 16, 64, 256\}$ are supported. This restriction is due to the fact that all nodes in the path from source to destination of a VC-4-xc must support the specific concatenation. The supported values of x are those that must be supported for clear channel operation for the defined STM-N frames. (In clear channel operation, the entire frame consists of a single VC.) Because of the restrictions in x, the bandwidth allocation is only a coarse one, offering the following data rates (including the path overhead):

STS-1 SPE: 50.1 Mb/s,
VC-4: 150.3 Mb/s,
VC-4-4c: 601.3 Mb/s,
VC-4-16c: 2.405 Gb/s,
VC-4-64c: 9.622 Gb/s,
VC-4-256c: 38.486 Gb/s.

A fine-grained bandwidth is possible only by exploiting the new virtual concatenation feature briefly described in Section 1.

3. General data aggregation aspects

By means of "data aggregation," a corresponding architecture allows the simultaneous processing of data received through several SDH/SONET line ports and the aggregation of the data either into a data stream for link-layer processing, e.g., by a switch fabric, or multiplexing of corresponding data into a higher-data-rate SDH/SONET frame. In the first case, the aggregated data is in the form of IP packets, ATM cells, or similar data. In the second case, the aggregated data is in the form of complete SDH/SONET VCs. Since all known SDH/SONET framers function in a bidirectional mode, these kinds of data aggregation are always accompanied by their inverted function—data distribution onto several SDH/SONET line ports. As a result, the framer must also be able to handle the data rate that corresponds to aggregation of all line ports directly. Hence, framers for data aggregation of, e.g., four STM-1 (STS-3) frames into an STM-4 (STS-12) frame or a corresponding 601.3-Mb/s data stream to the link layer must also be able to process an STM-4 frame directly.

Figure 3 is a block diagram of the functions in the data path of a simple SDH/SONET framer and the corresponding data flow for the receive and transmit sides of the framer. The frame-byte-alignment unit receives data from an SDH/SONET line port. The term *frame-byte alignment* indicates the case in which the incoming bit stream is changed into a multi-byte parallel data stream aligned with the frame structure. The B1 byte is an SOH byte that captures a bit-interleaved parity (BIP-8) calculated over the previous frame after frame scrambling. The B2 bytes are $N \times 3$ SOH bytes that capture a BIP-$N \times 24$ value calculated over parts of the frame before frame scrambling. SOH monitoring checks and interprets all SOH bytes of a received frame; SOH generation creates the corresponding SOH bytes for the current frame on the transmit side of the framer. POH processing is the corresponding operation for the POH bytes. Payload processing covers the mapping of payloads into SDH/SONET frames and the extraction of corresponding payload from the frames. This process usually includes calculating and checking certain checksums over the payload bytes, as well as checking for the packet-delineation features, which separate different payload packets from one another.

4. Data aggregation by multiple parallel units

The simplest way to achieve data aggregation would be to implement the functional blocks of Figure 3 in parallel for each data stream connected to an SDH/SONET line port, and to multiplex these data into a single data stream only after payload processing for data aggregation to a switch fabric or after POH monitoring for multiplexing VCs into a higher-level frame. However, this would still require additional POH, SOH, and B1 and B2 calculation units for the higher-level frame on the transmit side of the framer and an additional payload processor for handling

Table 1 Suitability of basic framer functions for an implementation with parallel units.

Function	*Trivial solution*	*Distributed solution*	*Other solution*
Frame-byte alignment	No	No	Additional units needed
Frame scrambling and descrambling	No	Yes	
SOH monitoring and generation	Yes		
B1 calculation	No	Yes	
B2 calculation	Yes	No	
POH monitoring and processing	Yes, but requires data reordering before POH processing if concatenation exists	No	
B3 calculation	No	Yes	
Payload processing			
ATM payload scrambling	No	Yes	
ATM cell delineation	No	No	Additional unit needed
Header error correction	No	No	Additional unit needed
Frame checksum calculation	No	No	Additional unit needed

higher-data-rate payload. Moreover, this framer would still not be able to receive higher-level frames. To overcome these shortcomings, a scalable, modular architecture based on parallel units with distributed algorithms and data exchange between the parallel units was developed [2]. This architecture exploits the fact that most of the functions of an SDH/SONET framer can be implemented in a distributed mode, which allows M parallel units to handle either M STM-N_1 frames in parallel or a single STM-N frame with $N = M \times N_1$. This architecture is then able to accommodate M STM-N_1 frames independently in parallel, to multiplex single VCs from M STM-N_1 frames into an STM-N frame, and to aggregate IP packets, ATM cells, and similar data from M STM-N_1 frames into a single data stream to a switch fabric. Since both N and N_1 must fulfill the restrictions for the number N in an STM-N frame, the number M is also limited to certain values {1, 3, 4, 16, 64, 256}, of which the number 3 is possible only for units handling STS-1 frames. The challenge for this architecture is the development of distributed algorithms that allow multiple parallel units to accommodate a higher-level frame. Fortunately, only some of the units shown in Figure 3 require distributed algorithms, whereas for other units the structure of the STM-N frame automatically leads to a solution. Because every STM-N frame consists of N byte-interleaved STM-1 frames, the SDH/SONET frame structure automatically leads to a natural decomposition of its payload-independent part, and therefore enables implementation of certain functions by parallel units without the need for interaction between the units. However, there are some units for which no suitable distributed solution seems to exist, which therefore require additional units for handling the higher data rates. A detailed discussion of suitable distributed algorithms is given elsewhere [2], and is not the subject of this paper. **Table 1** lists the functions of Figure 3 and also whether a trivial solution, a distributed solution, or no solution suitable for this architecture exists.

The emphasis is on those functions for which neither a trivial nor a distributed solution exists. In principle, these functions are not suited for an architecture based on parallel processing units. A first implementation example of this architecture is a framer chip designed in cooperation with the TranSwitch Corporation[1] [3, 4]. This framer is sold by Transwitch under the designation PHAST-12 or, since 1999, PHAST-12E. **Figure 4** is a block diagram of the PHAST-12 chip architecture, clearly revealing the parallel units. Also, the necessary exchange of data between the parallel units is shown as arrows connecting them. An important feature not mentioned thus far is the possibility of sending VCs from the receive data path of the framer to its transmit data path. In Figure 4 this is implemented through the central cross-connect stage. This connection enables the SDH/SONET multiplexing of four VC-4 containers received at the receive side into one STM-4 frame generated with these

[1] TranSwitch Corporation, 3 Enterprise Drive, Shelton, CT 06484.

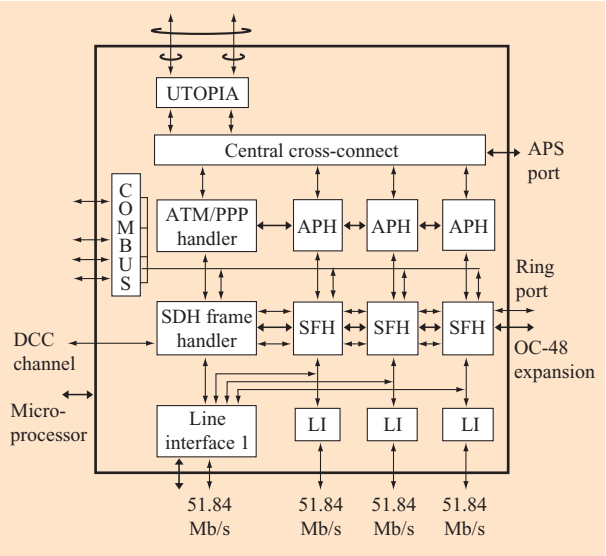

Figure 4

PHAST-12 chip architecture. Adapted from [2] with permission.

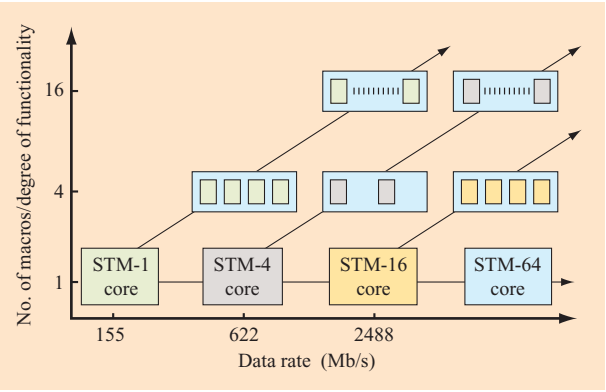

Figure 5

Scalability of parallel architecture.

containers at the transmit side of the framer. In addition to this special kind of data aggregation, the same connection also enables digital cross-connect functionality [2].

Another aspect of this architecture is its scalability. **Figure 5** shows how different framers can be constructed by using different numbers of parallel units as well as different basic data rates for the units. For example, the figure shows how an STM-4 framer (622 Mb/s) can be developed from four STM-1 macros, an STM-16 framer (2.488 Gb/s) from either four STM-4 macros or 16 STM-1 macros, and so on. A practical limit is given by the need to exchange data between all parallel units within a single clock cycle. This limits the time available for data processing by a unit with data exchange to less than $1/M$th of the cycle time for M parallel units. However, the main drawback of this architecture is that there are no suitable distributed solutions for certain necessary functions. A detailed look at Table 1 shows that for all functions in the pure SDH/SONET processing part, except for frame-byte alignment, there are distributed or trivial solutions. For data rates lower than STM-64 (OC-192), frame-byte alignment is integrated into the serializer/deserializer portion of the line port and therefore is of no relevance with respect to the digital framer portion. Hence, the parallel architecture with distributed algorithms is well suited for accommodating the pure SDH/SONET portion of a framer without the payload processor for data rates below STM-64 or 9.96 Gb/s as long as the number of line ports is smaller than the value at which the data exchange between the units creates a timing problem.

In the following, an architecture is presented that entirely avoids the problem of finding suitable distributed algorithms for all required functions and that should be suitable for use with a large number of line ports.

5. Data aggregation by a data-multiplexing architecture

The parallel unit architecture was based on the use of parallel units and the multiplexing of data into a single data stream after a series of parallel processing steps. The data-multiplexing architecture takes the opposite approach. Hence, the very first step on the receive side of a framer in this architecture is multiplexing the data received at multiple SDH/SONET line ports into a single data stream. Accordingly, on the transmit side of the framer, data destined for different line ports are transported in multiplexed form on a single data path. The functional units in Figure 3 then process data from one line port in one clock cycle and data from another line port in the next clock cycle. A snapshot taken at a specific clock cycle would reveal sequential units in the data path working on different frames. The architecture should even allow the multiplexing of data from different line port types, resulting in a separation of data and function. The same data path units will process data from different STM-N (STS-3N) frames at different clock cycles. This even extends to VCs in the sense that units processing VC-specific data do this at the rate of one VC per clock cycle. The basic idea of this architecture is comparable to the replacement of N parallel microprocessors in a corresponding architecture with a single microprocessor running at N times the speed of the original processors and capable of multi-threading with N threads. One consequence of this data multiplexing is that the system clock, which operates on all units in the data path except

at the external interfaces, must be running at a rate greater than the sum of all data rates of the SDH/SONET line ports to ensure that no update of a line port FIFO occurs before the last data from that port was read. On the other hand, a variant of the line port FIFO full signal will be forwarded with the data on the receive data path to mark data read twice from a FIFO. As logical units in the data path of a data-multiplexing architecture switch from one port to another and possibly even from one type of frame to another between successive clock cycles, all intermediate results must be stored whenever the unit switches from one port or VC to another. This requires that fast access memory in the form of register arrays is available that allows the content of working registers to be stored to on-chip memory blocks, working registers to be updated from these memory blocks, and data to be processed within a single clock cycle.

A possible concern here could be that this leads to excessive storage requirements, and a necessary first step must be to analyze the difference in storage requirements between an architecture based on parallel units and a data-multiplexing architecture. The data to be stored when switching ports or VCs are exactly the data that must be conserved from one clock cycle to the next cycle in the processing unit. These data are always stored in registers, regardless of whether the unit works in a data-multiplexing or a parallel architecture. If one compares the case of M line ports handled by M parallel units with that of M line ports handled by a data-multiplexing architecture, the parallel architecture requires M times the register space of a single unit, and the data-multiplexing architecture requires $(M + 1)$ times the corresponding register space. The $M + 1$ value results from the fact that in addition to the M registers needed to store the data from M line ports, there is a working register in the units. This certainly does not point to excessive storage needs for the data-multiplexing architecture. For large values of M, the difference in storage requirements for parallel and data-multiplexing architectures even disappears.

Another question is whether the use of a single unit instead of multiple parallel units would result in a large chip-area saving for the data-multiplexing architecture in comparison to a parallel architecture. The likelihood is that the saving would be large, although not as large as a factor of $M - 1$. The reason is that using the same technology and not being substantially below the speed limits of the technology, the data-multiplexing architecture requires a data path width of approximately M times that of the data path for a single unit in the parallel architecture. However, our investigations with both architectures predict that the chip area required for M parallel units with necessary data exchange between the units should continue to be significantly larger than the corresponding area for a single unit with M times the data path width.

Table 2 Comparisons for STM-1 to STM-64 frames.

Frame type	*SOH row length* (bytes)	*Row length* (bytes)	*Frame length* (bytes)
STM-1	9	$270 = 9 \times 30$	9×270
STM-4	4×9	$4 \times 9 \times 30$	$9 \times 4 \times 270$
STM-16	16×9	$16 \times 9 \times 30$	$9 \times 16 \times 270$
STM-64	64×9	$64 \times 9 \times 30$	$9 \times 64 \times 270$

In the following, a specific example is used to describe the data-multiplexing architecture in greater detail.

Data-multiplexing architecture for an STM-1 to STM-64 framer

For a data-multiplexing architecture framer suitable for processing frames from STM-1 to STM-64 with an aggregate data rate corresponding to STM-64, a data path width should be chosen which facilitates the processing of the entire range of SDH/SONET frames from STM-1 to STM-64. Taking into account the strongly regular structure of SDH/SONET frames, fulfillment of the following requirements in the priority sequence given would be desirable:

1. The data path width should be an integer fraction of the frame length for all frames.
2. The data path width should be an integer fraction of the length of a row for all frames.
3. The data path width should be an integer fraction of the SOH part of a row for all frames.

The first requirement ensures that a new frame always starts with the first byte forwarded on the data path, and is necessary because otherwise the change from one frame to the next would occur somewhere in the middle of the data being forwarded in parallel on the data path. This would require significant additional multiplexing and control logic. The second requirement ensures that every new row starts with the first byte in the data path, and greatly simplifies processing of the row. The third requirement ensures that all bytes forwarded in parallel on the data path are either SOH bytes or bytes of a VC for all rows except the fourth row, which is used for frequency adjustment by redefining specific SOH bytes into data bytes. The simplest data path fulfilling all requirements (the third one even for the fourth row) is a single-byte-wide data path. However, with present CMOS technologies, fabrication of an STM-64 framer with a single-byte-wide data path is not possible. Considering multi-byte-wide data paths, the comparisons in **Table 2** apply. As can be seen from the table, 9 bytes is an integer

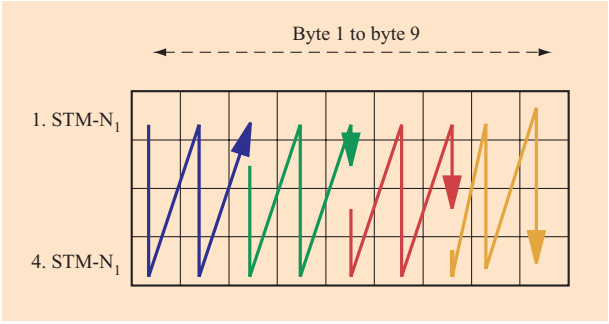

Figure 6

Diagram of procedure for writing STM-4 SOH bytes into reordering buffer.

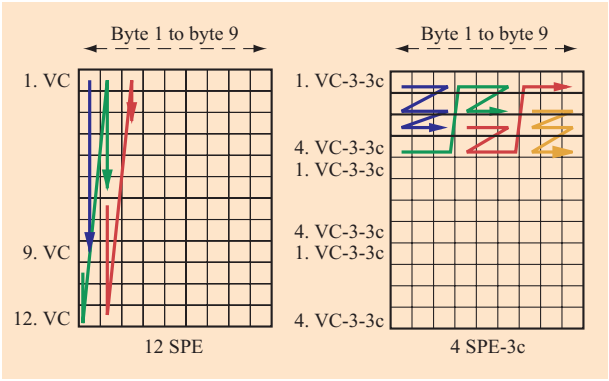

Figure 7

Diagram of procedure for writing STS-12 VC bytes into reordering buffer.

fraction of the frame length, the row length, and the SOH part of the row. Because 9 decomposes as 3×3, 3 of course also fulfills the requirements, so the possible data path widths fulfilling all three requirements are 3 and 9 bytes. A 3-byte-wide data path would require a system clock of at least 415 MHz, which would be problematic for some of the framer functions in CMOS ASIC technology. A 9-byte-wide data path width, on the other hand, would require a system clock of at least 138 MHz, which should be feasible. Even the use of the next higher SDH/SONET standard reference clock of 155.52 MHz should be compatible with current CMOS technology.

Another aspect to consider is the arrangement of data on the data path. Looking at the framer functions shown in Figure 3, one can recognize three main representations of data which are optimum or even necessary for certain framer functions:

1. On the SDH line side, for frame-byte alignment and frame descrambling/frame scrambling, the bits must be presented in the exact sequence of the STM-N frame.
2. For SOH-byte processing (except for the H3 bytes in the fourth row of the frame), the preferred data arrangement is one in which the byte interleaving of the *N* STM-1 subframes within an STM-N frame is removed and all bytes in a single clock cycle on the data path belong to one specific STM-1 subframe. In this representation, the SOH bytes always occur in the same positions for each frame. In the 9-byte data path, this always presents a single SOH row for a STM-1 subframe.
3. For the POH bytes, the H3 bytes (fourth SOH row of the frame), and payload handling, the optimum data arrangement is the one in which the bytes of each row are reordered such that all bytes in a single clock cycle on the data path belong to the same VC-4-*xc* ($x = 1, 4, 16, 64$).

From this list, one immediately recognizes that optimum processing of the data by all units in the data path requires two data reordering operations. The first reorders the SOH bytes of a row, except for the H3 bytes of the fourth row. The second reorders the bytes of a row that belong to VCs (POH and payload bytes), plus the H3 bytes of the fourth row. As mentioned in the Introduction, the H3 SOH bytes require exactly the same reordering as the POH and payload bytes. Since the data bytes forwarded in parallel on the data path are always either SOH or VC bytes for all rows except the fourth, both reordering operations can be performed in a single step and, accordingly, in a single unit (with special consideration for the fourth row) in the transmit and receive data paths, respectively. Of course, the data reordering operations do not involve just the 9 bytes present on the data path at a single clock cycle; the reordering must be done over multiples of 9 bytes, which must be stored in an on-chip buffer for reordering.

In more detail, the reordering of the SOH part of an STM-N frame requires two buffers with the width of the data path and a depth of *N*; two buffers are required because while the first filled buffer is being read, data must be written into a new buffer. **Figure 6** shows how data are written into the buffer. The SOH bytes arriving on the data path are written sequentially into the first column of the buffer, then the second, and so on until the ninth column. When the buffer has been filled, the content is read row by row on the 9-byte-wide data path.

For the VC bytes, the procedure is very similar if there is no concatenation, as shown on the left-hand side in **Figure 7**. Here, the reordering buffer has a width of 9 bytes and a depth equal to the number of VCs in the STM-N (STS-3N) frame. In the case of concatenated VCs,

the first x bytes of a SPE-xc or VC-4-xc are written into the first row, the next into the second row, and so on until as many rows contain the first x bytes as there are SPE-$x$cs or VC-4$x$cs in the frame. Then the next bytes are written into columns $x + 1$ to $2x$ of the first row, and so on. In principle, one could stop after the buffer has been filled from rows 1 to K if there are K VC-4-$x$cs or SPE-$x$cs in the frame; however, to achieve a consistent behavior with frames without concatenation, reading from the buffer begins only after the entire buffer has been filled from rows 1 to N.

Future SDH/SONET framers will support the new virtual concatenation standard. In this case there is a special buffer in the payload processor that assembles all payload bytes of the VCs and performs virtual concatenation of VCs. Real concatenation can then be treated as a special case of virtual concatenation, and the need to reorder data such that real concatenation of VCs is accounted for disappears. In this case, the data reordering simplifies substantially. In principle, the data reordering procedure for SOH bytes can now be applied to all data bytes. It is still necessary to distinguish between SOH and VC bytes, but the procedure to reorder the two types of data is identical.

Considering the SOH byte reordering, there are N STM-1 subframes building an STM-N frame. Thus, it may take N clock cycles with bytes from this specific frame until sufficient bytes have been assembled to fill an entire data path width with bytes belonging to a specific STM-1 subframe of the STM-N frame. As an example, the fifth row of an STM-4 frame before reordering appears as follows:

B21(1) | B21(2) | B21(3) | B21(4) | B22(1) | B22(2) | B22(3) | B22(4) | B23(1) |
B23(2) | B23(3) | B23(4) | K1 | − 5 empty bytes
6 empty bytes | K2 | − 2 empty bytes |
9 empty bytes |
payload bytes in groups of 9 bytes,

with the numbers in parentheses referring to the corresponding STM-1 subframes. After reordering, it appears as follows:

B21(1) | B22(1) | B23(1) | K1 | 2 empty bytes | K2 | 2 empty bytes
B21(2) | B22(2) | B23(2) | 6 empty bytes
B21(3) | B22(3) | B23(3) | 6 empty bytes
B21(4) | B22(4) | B23(4) | 6 empty bytes
payload bytes in groups of 9 bytes, ordered according to nonconcatenated VC-4s or STS-1 SPEs.

As already mentioned, a special case is the fourth row of the frame with the pointer bytes. For an STM-4 (STS-12) frame with AU-3 pointers and corresponding STS-1 SPE containers, this row appears as follows before reordering:

H1(1) | H1(2) | H1(3) | H1(4) | H1(5) | H1(6) | H1(7) | H1(8) | H1(9) |
H1(10) | H1(11) | H1(12) | H2(1) | H2(2) | H2(3) | H2(4) | H2(5) | H2(6) |
H2(7) | H2(8) | H2(9) | H2(10) | H2(11) | H2(12) | H3(1) | H3(2) | H3(3) |
H3(4) | H3(5) | H3(6) | H3(7) | H3(8) | H3(9) | H3(10) | H3(11) | H3(12) |
payload bytes in groups of 9 bytes.

After reordering, the structure is

H1(1) | H2(1) | H3(1) | 6 payload bytes belonging to STS-1 SPE #1 |
H1(2) | H2(2) | H3(2) | 6 payload bytes belonging to STS-1 SPE #2 |
. . . . . .
H1(12) | H2(12) | H3(12) | 6 payload bytes belonging to STS-1 SPE #12 |
payload bytes in groups of 9 bytes, ordered according to nonconcatenated STS-1 SPEs.

Hence, there is a mixture of SOH and payload bytes in the bytes forwarded in parallel on the data path. This is unavoidable because the H3 bytes can be SOH as well as payload bytes, depending on the pointer value encoded in the H1 and H2 bytes.

Figure 8 is the block diagram for an STM-1 (STS-3) to STM-64 (STS-192) framer in the new data-multiplexing architecture. The data-reordering unit in the transmit side is designated as the VC-4 interleaving unit. The figure shows the individual logical units in the data paths connected to multiple memory blocks according to the different frames (ports) and VCs to be processed. The connection from the receive to the transmit side labeled "cross-connect" enables the multiplexing of VCs received on the receive side into a higher-level frame on the transmit side and thereby the SDH/SONET add/drop multiplexing function, the special data aggregation of VCs into a higher-level, higher-data-rate frame. In addition, it also enables the digital cross-connect function of SDH/SONET between the VCs received from different line ports. As a result, it should be possible to use the framer as an SDH/SONET digital cross-connect, a configurable but not self-routing switch.

The framer shown in Figure 8 should be capable of supporting 64 STM-1 ports, 16 STM-4 ports, four STM-16 ports, one STM-64 port, or any combination thereof with an aggregate data rate corresponding to STM-64.

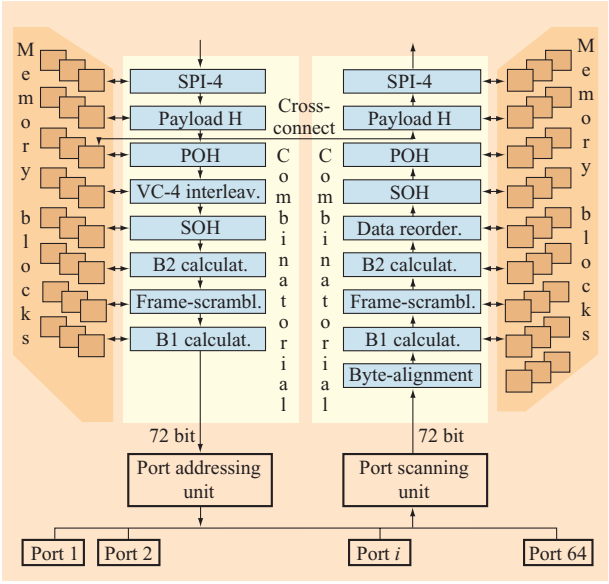

Figure 8

Block diagram of STM-1 to STM-64 SDH/SONET framer with data-multiplexing architecture.

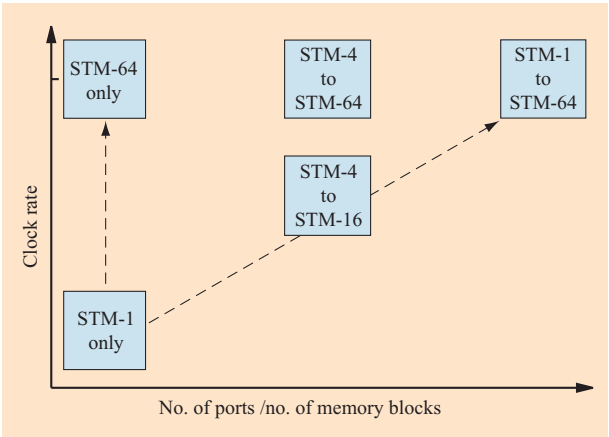

Figure 9

Scalability of data-multiplexing architecture for an STM-1 to STM-64 framer.

The figure pertains to the maximum use of this specific data-multiplexing architecture. It should also be possible to use this architecture for building framers that support only a subrange of the frame types supported in Figure 8. **Figure 9** shows how it may be possible to construct an entire range of framers with this architecture by reducing either the system clock rate from that needed for STM-64 or the number of ports and the related number of memory blocks. Of course, increasing the number of ports always necessitates a corresponding increase in the minimum clock suitable for the framer. Therefore, possible framers exist only in the triangle above the diagonal in the figure. The "STM-1 only" and "STM-64 only" versions are displayed for the sake of completeness only; constructing a single port framer in a data-multiplexing architecture is certainly not efficient, but may be a solution that can be quickly implemented if a corresponding multi-port framer already exists.

In total, the data-multiplexing architecture should be a very efficient architecture for data aggregation devices in SDH/SONET technology that can be used to create single-chip framers for an entire range of data rates. A framer with an aggregate data rate corresponding to STM-256 (OC-768, or 39.81 Gb/s) should be feasible if a data path width of $2 \times 9 = 18$ bytes is chosen for the newest CMOS technology or one of $4 \times 9 = 36$ bytes for a somewhat slower technology. On the basis of the three requirements for the data path width with respect to the frame length, row length, and SOH row length, it should be possible to cover the range from STM-4 to STM-256 with a corresponding architecture.

Although the new architecture for SDH/SONET framers has not yet been implemented, selected timing-critical units for an STM-64 as well as an STM-256 data aggregation framer have been designed in the IBM Cu-11 technology [5]; at least at the design level, these appear to fulfill all timing requirements (including those related to storing and retrieving data from connected memory banks). System-level simulations including the port scanning unit, byte-alignment unit, B1 calculation unit, and frame scrambling unit—all designed in Cu-11 technology—have been successfully performed and demonstrate the principal feasibility of the architecture. Chip size estimates show that the STM-256 framer without virtual concatenation should be feasible with an approximate size of 150 mm^2 in the Cu-11 technology. A chip size of about 220 mm^2 should also support the new virtual concatenation feature with an embedded DRAM for storing the payload of 16 STM-256 frames, as required by the standard.

6. Concluding remarks

Single-chip architectures for data aggregation from multiple SDH/SONET line ports to link layer devices as well as higher-order SDH/SONET frames have been discussed. The architectures range from simple parallel architectures to parallel architectures with distributed algorithms, and finally to data-multiplexing architectures. Simple parallel architectures are suitable for data aggregation to the link layer but require many additional units for data aggregation into higher-level frames. Parallel architectures with distributed algorithms are

suitable for data aggregation into higher-level frames for data rates at which frame-byte alignment is done in the serializer/deserializer part of the framer and up to the number of parallel units at which the necessary data exchange between parallel units causes timing problems. A novel data-multiplexing architecture approach has been described that should be suitable for data aggregation to the link layer as well as into higher-level SDH/SONET frames, but will require fast on-chip access to memory arrays. Since such fast memory access is now available, it is anticipated that data-multiplexing architectures will soon play an important role in data aggregation devices.

Acknowledgments

I thank my colleagues Andreas Herkersdorf and Wolfram Lemppenau (now at FH Gelsenkirchen, Germany) for introducing me to the architecture based on parallel units with distributed algorithms and for a very good collaboration in general. I also thank my colleagues Fabrice Verplanken, David Webb, and Peter Buchmann for many helpful discussions on the data-multiplexing architecture.

References and note

1. M. Sexton and A. Reid, *Transmission Networking: SONET and the Synchronous Digital Hierarchy*, Artech House, Norwood, MA, 1992.
2. R. Clauberg, A. Herkersdorf, W. Lemppenau, and H. Schindler, "A Scalable Modular Architecture for SDH/SONET Technology," *Proceedings of the International Conference on Computer Communications and Networks* (*ICCCN*), Boston, MA, October 1999, pp. 442–446.
3. Product announcement for PHAST-12 by the TranSwitch Corporation, Shelton, CT, February 17, 1999.
4. P. Baechtold, M. Beakes, P. Buchmann, R. Clauberg, J. F. Ewen, J. F. Gilsdorf, P. Hauviller, A. Herkersdorf, J.-C. Le Garrec, W. Lemppenau, Ben Parker, Dale J. Pearson, J. M. Pereira, D. Plassat, S. K. Reynolds, H. R. Schindler, A. Steimle, and D. J. Webb, "Single-Chip 622-Mb/s SDH/SONET Framer, Digital Cross-Connect and Add/Drop Multiplexer Solution," *IEEE J. Solid-State Circuits* **36,** No. 1, 74–80 (2001).
5. The IBM Cu-11 technology was first announced in an IBM press release on April 3, 2000, and is described in detail at *http://www-3.ibm.com/chips/products/asics/products/cu-11.html* and on additional web pages with links provided therein.

Received February 11, 2002; accepted for publication June 6, 2002

Rolf Clauberg *IBM Research Division, Zurich Research Laboratory, Säumerstrasse 4, CH-8803 Rüschlikon, Switzerland (cla@zurich.ibm.com).* Dr. Clauberg is a Research Staff Member at the IBM Zurich Research Laboratory. The work presented here was done while he was Manager of Transport Technology in the Communications Systems Department. He was previously a manager in the Optoelectronics Department, leading projects in device modeling and characterization. Dr. Clauberg holds a Ph.D. degree in physics (Dr. rer. nat.) from the University of Cologne in Germany. He is a member of the German Physical Society, the American Physical Society, and the German Informationstechnische Gesellschaft (ITG/VDE).