Designing a Multiprocessor C54x Platform for Voice-Over-Network Applications

Erich Vogel
Technical Staff, San Jose FSO

Wireline Communications

Abstract

The transmission of voice over data networks has long been dismissed as unfeasible due to the incompatibility of voice traffic and data traffic resulting in unacceptable voice quality. The market for such systems, however, is witnessing surprising growth due to its potential advantages in system cost and infrastructure and the rise in alternative networking options, such as ATM and frame relay.

Market requirements for practical and cost-effective solutions are driving Voice-over-Network (VoN) systems to process increasing numbers of voice channels on smaller platforms. This paper discusses the hardware and software issues involved in designing such a system using the Texas Instruments (TI™) TMS320C54x digital signal processor (DSP). Specifically, the discussion centers on system and architectural issues of a multiprocessor-based approach to process multiple voice streams. Topics include the voice and network interfaces to the DSP, common software algorithms, system control, and hardware architectures.

Contents

Figures

Introduction

VoN products have recently witnessed unforeseen growth in the form of single-user Internet-based telephony to larger scale concentrators and PBX equipment routing telephone calls over private network trunks. This growth is attributed primarily to the economic and practical advantages of combining voice and data services over a common medium.

In voice-over-network systems, the DSP bears the responsibility of converting continuous voice streams into forms suitable for reliably transmitting over a packet- or cell-based network. The tasks of the DSP in this process include

- Packetization
- Voice processing and compression
- Network delay concealment
- Line echo cancellation
- Efficient bandwidth utilization

Creating a cost-effective solution, a major goal of most designs, often requires that boards process as many channels as possible. The solution to this problem is twofold:

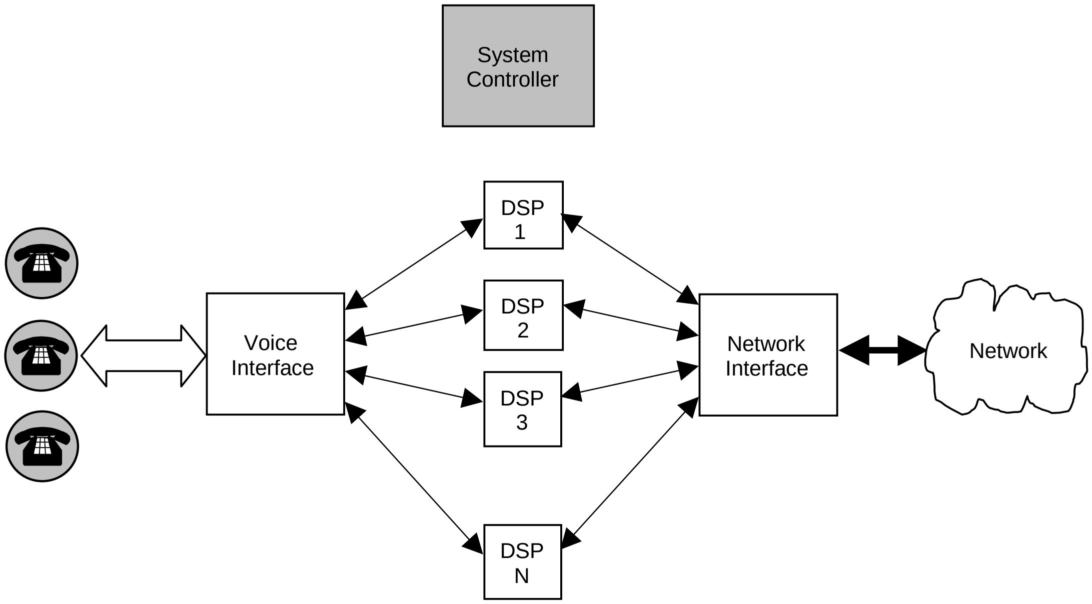
- Increasing the number of channels processed by each DSP
- Increasing the number of DSPs on the board

Each solution presents individual design challenges. The number of voice channels that each DSP can manage is a function of its algorithms, the speed of the DSP, and its available memory. Packing multiple DSPs on a board involves other issues, such as board area constraints, communication bandwidth, routing, and power dissipation.

This paper introduces the system-level design considerations of such a voice-over-network solution based on a multiple TMS320C54x platform. The C54x DSP is well suited for such applications due to its software availability, low power dissipation, and appropriate mix of peripherals and memory. Topics covered in this discussion include hardware architectures, system control options, and application software requirements in the design and integration of a voice-over-network system. However, this report is not intended to provide a complete reference design, but rather to outline the framework on which to base such a design.

The block diagram in Figure 1 provides a general system overview required of a voice-over-network solution. The block diagram can be divided into the following areas: the voice interface, the DSP array (including software), system control, and the packet (network) interface. Each will be discussed individually with an emphasis centered on the DSPs and their contribution.

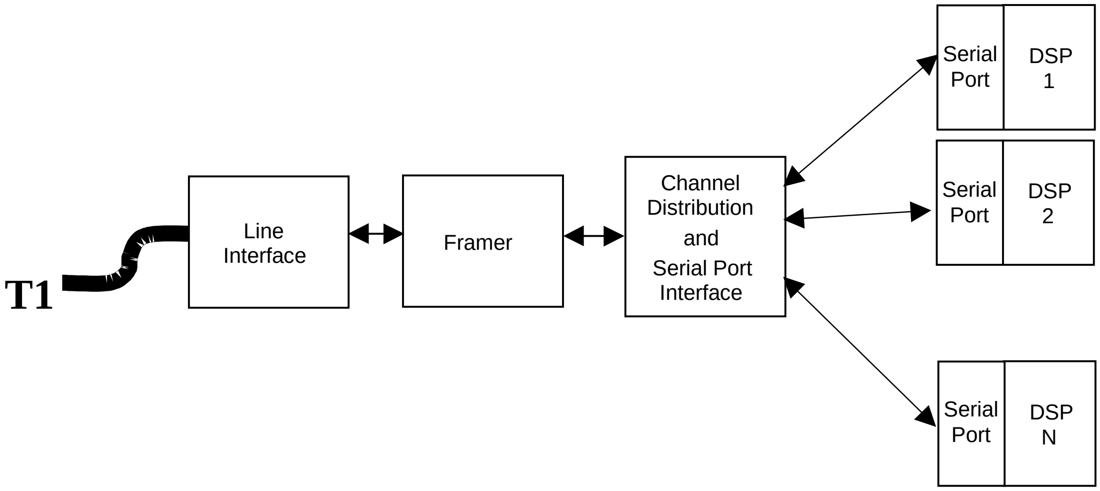
Figure 1. System Block Diagram



Voice Interface

The voice interface provides the mechanism to transmit and receive digital voice between the telephone system and the DSP. The details of this process depend on the source for the voice data. A simple single-channel case could consist of a digital telephone connected via a serial link to the DSP. This discussion, however, will center on aggregate voice streams such as those from a PBX, requiring service by multiple processors. The most common mediums for transmitting such multi-channel digital voice traffic are T1 (US) and E1 (Europe) lines, employing a time-division access scheme. Figure 2 shows a block diagram of the major components involved in such an interface including the line interface, framer, channel distribution, and serial interface.

Figure 2. Voice Interface



Line Interface and Framer

The front-end element in interfacing to a T1/E1 line is the transceiver, which contains several components, including a line interface, a framer, and often a buffer. The line interface serves as the physical connection to the twisted pair for transmitting and receiving analog signal waveforms and recovering clocking information used in synchronization. Connecting the transceiver to the line also requires several additional components, including transformers and line filters. Next, the framer and buffer detect the frame, multiframe, and channel boundaries in the data stream, monitoring for error conditions and alarms, and buffer blocks of data for connection to the processor.

Multiple channels of digital data are transmitted over T1 or E1 lines in a time-division multiplexed (TDM) fashion. Each timeslot, or DS0, consists of signaling information and eight bits of data. Timeslots are further grouped into elements called frames, or DS1s. Both the T1 and E1 standards adopt different channel attributes, such as the timeslots per frame and the base clock rate at which the bits are transmitted. A T1 stream employs 24 channels per frame and operates at a bit rate of 1.544 Mbps (1.536 of which is available for data). E1 data is transmitted at 2.048 Mbps and contains 30 channels. In both systems, channels contain eight bits of data transmitted at 64 kbps. Each channel also contains associated signaling consisting of information about the state of the call, whether it be on-hook, off-hook, or otherwise.

When transmitting voice over the channels, the voice is usually digitized and transmitted as a companded PCM voice stream. Companding is a perceptual compression technique in which linear data is converted to a non-linear logarithmic form. Two common standards are u-law and a-law.

Channel Distribution and Serial Interface

Channel distribution refers to relaying each channel in the frame to the appropriate DSP responsible for its processing. Framers generally provide this data as a TDM serial stream that is transmitted to the serial port peripheral of the C54x. This interface may require a small amount of glue logic to provide the appropriate clocking and interfacing signals.

Interfacing the framer chip to the DSP is facilitated by using the Multi-Channel Buffered Serial Port (McBSP) peripheral available on the TMS320C5410 DSP. This peripheral allows the DSP to extract a given channel (or channels) from a TDM frame by programming the peripheral with the appropriate timeslot and framing information. This allows all channels to be broadcast to all DSPs and offers the advantage of not requiring any channel distribution logic or selective routing.

The McBSP also allows this bus broadcast approach to be used in the DSP serial transmit direction. During transmit, the McBSP only transmits data during its responsible channels; otherwise, it remains in a high impedance state. This solves the bus collision problem of multiple DSPs writing to the bus at the same time, assuming each DSP is programmed appropriately with non-overlapping channels. An alternative solution to channel distribution is to use more complex logic to distribute the channels among the DSPs using dedicated serial lines. This is a more complex and less flexible alternative.

Using the aforementioned approach, the host processor manages the channel allocation task by informing each individual DSP of its responsible timeslots at initialization time. Channels can also be allocated dynamically by appropriately informing each DSP of its assigned channels during operation. This could be employed to circumvent problems such as a nonfunctioning DSP or to reallocate channels among DSPs for optimal resource management.

The framer provides signaling data for each channel as a separate data stream. To manage this signaling data, several methods can be suggested. Since the DSPs generally have no direct need to process signaling data, the signaling lines from/to the framer can be directly routed to the host serial port, alleviating a burden from the DSP. Another solution is to route signaling to the DSPs in the same manner as the data. Since each C54x has two McBSPs, the other serial port can be allocated for signaling data. The details depend on the method used by the framer chip.

Physically connecting the C54x serial port to the framer requires little overhead in terms of external connections. Three lines are required by both the transmit and receive operations:

❒ Bit-rate clock signal

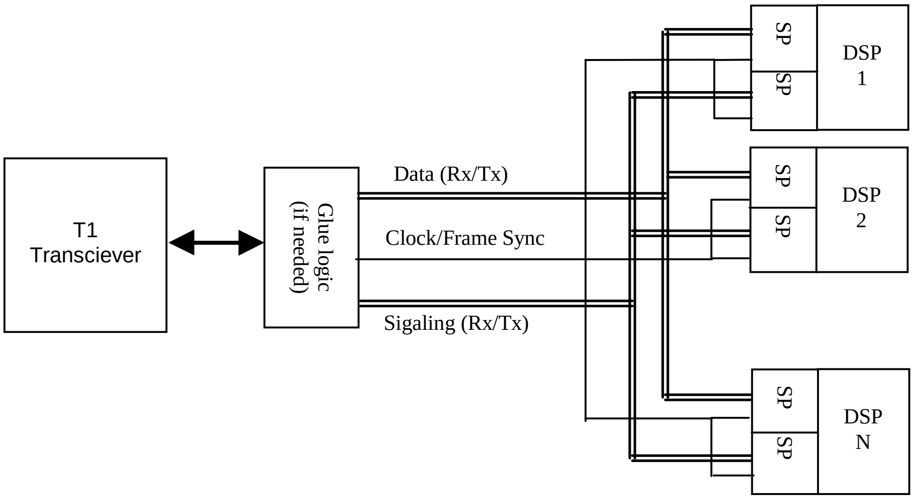
❒ Frame sync

❒ Serial data line

The clock and frame sync must be synchronized with the T1/E1 line clock and can generally be extracted from the transceiver chip. Assuming adequate current drive, both the transmit and receive lines can be distributed amongst all chips in the bus. For further details on interfacing to the C54x serial ports, see the *TMS320C54x Reference Set.*

A diagram of the transceiver/DSP serial interface is shown below in Figure 3.

Figure 3. Transceiver Serial Port Interface



System Control

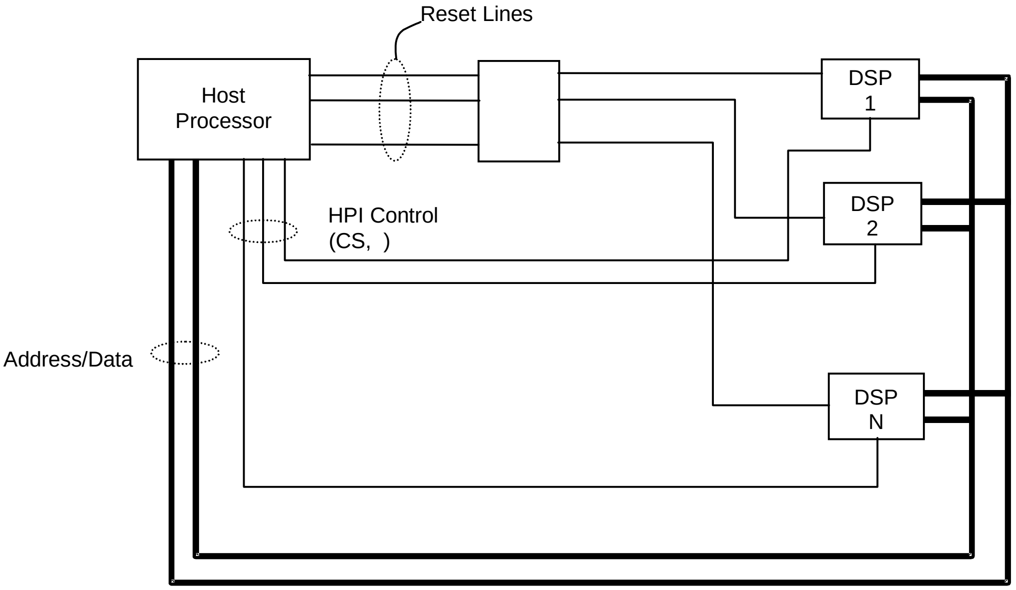
DSP-based systems generally function so that a host processor (such as a microcontroller) provides control over the subordinate or slave DSPs. With respect to DSP control, the host's responsibilities can be divided into two tasks:

- First, the host manages system initialization, in which it downloads code, distributes parameters, and allocates resources to each processor.
- Second, during normal operation, the host maintains the proper functioning of the system by regularly monitoring the performance of each DSP.

During system initialization, the host processor boots up first, then prepares to initialize the DSPs. To control the DSPs for either initialization or steady-state system monitoring, the host requires a mechanism to communicate with each DSP individually. A peripheral on the DSP called the Host Port Interface (HPI) facilitates this communication link. The HPI allows the host processor access to a block of internal memory for read and write accesses. For details on the operation of the Host Port Interface, see the *TMS320C54x Reference Set.*

Another requirement of the host is that it control the reset operation of each DSP. This allows the host to initiate the boot process of each DSP and gives the host a mechanism to restart a non-functioning DSP. An overview of the required host/DSP interconnections is shown in Figure 4.

Figure 4. System Control



Several possible methods can be used to efficiently boot and initialize multiple DSPs, each with advantages and disadvantages. One method uses the host to transfer code via the HPI to each DSP individually. This allows the host complete control of the code distributed to each DSP, thus increasing the design flexibility. The code would reside on an EPROM to which only the host would require access.

Another solution provides either an EPROM to each DSP or a central EPROM to which each DSP has access. The host would thus initiate and control the boot process of each DSP from the appropriate EPROM.

To boot a DSP via the HPI, the host must first download code via the HPI. This is done while the DSP is in the reset state. On some 54x processors, the host only has access via the HPI to a 2K block of memory in the DSP. Since almost all VoN applications are larger than 2K, a boot manager is first downloaded. Once the DSP is removed from the reset state, it runs from the factory-installed boot ROM, which subsequently branches to the HPI accessible memory (0x1000). The host must then transfer the application code to the boot manager via the HPI. This HPI accessibility limitation will not exist in future versions of the peripheral.

During both initialization and steady state operation, a mechanism must be provided to allow message transfer between the host and DSPs. The HPI also serves this need well. An important requirement for message transfer is deciding on efficient and well suited protocols, handshaking, and message formats. Since the HPI is used both to transmit data packets and message traffic, overhead information in the packet allows the DSP and packet processor to differentiate packets intended for data transfer and those intended for messaging. Details of such protocols are often proprietary, but many are based on common data transfer protocols such as HDLC.

A handshaking mechanism must also be provided to indicate packet transfers and acknowledgements. The C54x has several interrupt mechanism for this purpose. An HPI interrupt initiated by the host to the DSP indicates that a new packet has been written to the HPI memory block. Once the DSP has read or processed the packet, an acknowledgement must be sent indicating that another packet may be transmitted. The DSP must also have a mechanism to request a data or message packet transfer to the host. An interrupt or a host polling mechanism can be used as such a service request.

Once each DSP has been successfully started, the role of the host switches to that of monitoring the proper operation of each DSP. This is easily accomplished by requiring each DSP to regularly provide updates of system performance to the host. A system problem can be indicated by the DSP either ceasing to transmit status messages or transmitting a message indicating poor performance. At this point, the host can either reset the DSP and restart it, or leave it down and reallocate resources to other DSPs.

DSP Interface and Application Software

The primary role of the DSP in a voice-over-network system is to bidirectionally convert voice data between a continuous data stream and a packetized data stream. Associated functions of this process are

❒ Data compression

❒ Echo cancellation

❒ Network packet delay concealment

The most optimal solution to this system also requires that each DSP in a multiple-DSP environment also process multiple streams of voice. The number of streams is determined primarily by the speed of the signal processing algorithms and associated overhead, and by the speed of the processor. For example, given that the compression algorithm, echo cancellation, and overhead processing only require 20 MIPS of processing bandwidth, a 100-MHz C5410 DSP could process four channels of voice with 20 MIPS left for overhead, error margin, and future expansion.

Operating System

A crucial element of a system processing multiple data streams is the proper coordination and management of resources to service the data. This task management is best suited for a Real Time Operating System (RTOS), such as Spectron/TI DSP BIOS. The Spectron/TI DSP BIOS provides a firmware kernel for task management, analysis, and trace functions with very little memory overhead (< 1K words).

The first step in adapting a software application to use the DSP BIOS is to structure the application into various modules or threads. For example, the echo canceller and vocoder should be allocated to different threads. In fact, the vocoder can be further split into coding and decoding functions. Depending on the processing time required by a block, further partitioning might be necessary for the task scheduler to optimally manage the tasks. BIOS manages the execution of these threads by prioritized, pre-emptive scheduling. This requires that the programmer assign each thread an appropriate priority such that if a task is executing and a task of higher priority is scheduled for execution, the lower priority task may be interrupted or preempted.

The application interfaces to the BIOS kernel functions via the DSP BIOS API, which provides a standard environment for development and integration. The following is a list of additional features offered by the DSP BIOS environment

- Periodic Functions—BIOS also allows tasks to be scheduled based on the system clock. This allows functions such as watchdog tasks to be executed periodically.
- I/O Stream Manager—DSP BIOS provides two means for data transfer, data pipes and host transfers. Data pipes manage I/O for data transfers by providing buffering, notification, and software data structures. Transfers can occur between threads or as a mechanism for transferring data on/off chip via a peripheral. The host transfer is a form of a pipe in which one end of the pipe is managed by the host. This can be used to send and receive data streams from the host computer. It also aids in simulations of external (peripheral) data transfers during development by sending data via the host.
- Instrumentation Manager—The instrumentation provided by DSP BIOS allows real-time monitoring of system functions with minimal intrusion on the application. The available instrumentation includes logging, statistics accumulation, and event monitoring. Logging allows capturing information about events and provides a means for programs to send messages to the host. The statistics accumulator captures information (count, max, total) for variables. The event monitor allows the programmer to trace the execution of threads during system run-time. This provides an easy means to find real-time bugs, such as priority problems, and MIPS issues.
- Configuration Tool—The Configuration tool provides a visual editor for creating system objects such as signals, I/O streams, event logs, etc. It also allows the programmer to set up system properties.

The operating system was also designed to use minimum system resources such as processing time and memory. For instance, the kernel only uses less than 1K words of program space, the API is modularized to only link in libraries required by the applications, and all instrumentation data is formatted on the host rather than the DSP.

Signal Processing Algorithms

The major goals of the DSP application software in a voice-over-network system are threefold:

- ❒ To compress the data travelling across the network to minimize bandwidth and maximize channels
- ❒ To enhance voice quality
- ❒ To transform the data from a continuous stream to packets and vice versa

Other possible tasks may include those associated with handling data in addition to voice traffic, such as modem and fax transmissions. For data transmissions, the DSP must switch off the voice coding and echo cancellation since these are perceptual voice algorithms and would corrupt the data. A key element in this process is the detection of fax traffic by the DSP. Fax traffic is delineated from a voice stream by specific tones and protocols injected in the voice stream. Consequently, the DSP must constantly probe for these signals.

A block diagram of the elements involved in the voice processing is shown in Figure 5 and Figure 6.

Figure 5. Voice Encoder Block Diagram (Ingress)

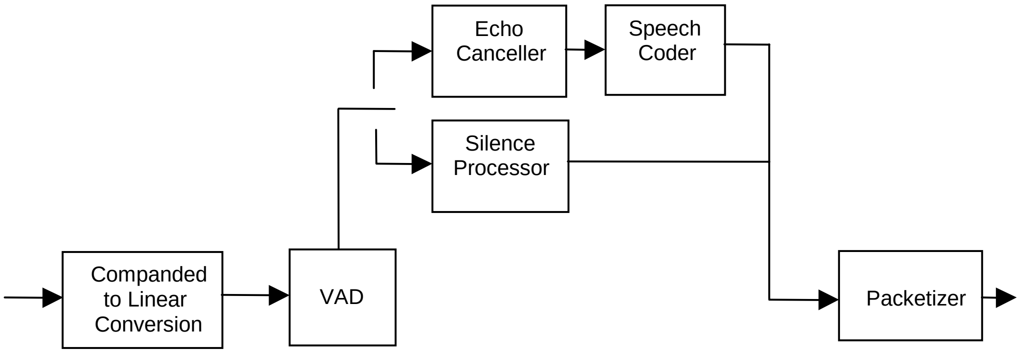
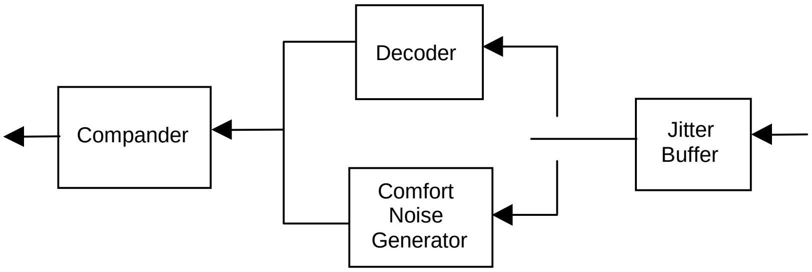


Figure 6. Voice Decoder Block Diagram (Egress)



Voice Coding

Voice coding, known as *vocoding*, provides a means to increase the call or channel density on a system by compressing the voice data. Many vocoders and vocoder implementations exist. Choosing the appropriate one depends on trade-offs between compression, processing requirements, and resultant voice quality. Table 1 lists common vocoders used for voice-over-network systems and their characteristics.

Table 1. Vocoder Formats

Vocoder Standard	Bit Rates (kbps)	Notes
G.726 ADPCM	40, 32, 24, 16	MIPS efficient, but voice quality degrades rapidly for lower bit rates
G.728 LD-CELP	40, 16, 12.8, 9.6	Better voice quality at lower bit rates and higher compress, but less MIPS efficient
G.729 CS-ACELP		Best compression, but least efficient

Another bandwidth optimization technique commonly used is voice activity detection (VAD). This technique determines whether the channel is currently transmitting voice or silence based on the energy of the input signal. Since speech, or voice activity, only comprises approximately 40% of the average voice call, the remaining 60% wastes bandwidth since no useful information is transferred. Consequently, when speech inactivity, or silence, is detected, the algorithm only sends an indication that the current frame is silent instead of the actual coded waveform representation. The receiver subsequently inserts an approximation of the background noise called *comfort noise* during these periods.

Echo Cancellation

Echoes in a voice network are caused by signal reflections, usually from several sources but primarily from boundaries where hybrid circuits exist. These hybrid circuits convert from four-wire dual half-duplex systems to two-wire full duplex, thus minimizing wire costs. The resultant reflections take the form of audible copies of the speaker’s voice delayed by a perceivable time. Echoes always exist in voice networks—it is only the echoes delayed by a large amount of time that become annoying. In fact, most people find the total absence of echo somewhat annoying and associate a small amount of echo with an active telephone line.

Echo cancellation is the process by which these delayed echoes are removed. The most common technique used is adaptive filtering, in which a processor determines the characteristics of a waveform or voice on one line, then uses this information to configure or adapt a filter to cancel the same voice in the opposite direction (the echo). The metric used to compare these algorithms is the amount of echo cancelled and the length of the filter, which impacts the maximum echo delay that the algorithm can cancel.

Jitter Buffers

Network congestion can cause variations in delay between packets or cells as information is transmitted over a network. This delay variance is known as *jitter*. Since voice is very delay sensitive, even small amounts of jitter interrupt a voice conversation. For this reason, voice traffic has historically never been accepted as suitable for transmission across highly congested connectionless data networks.

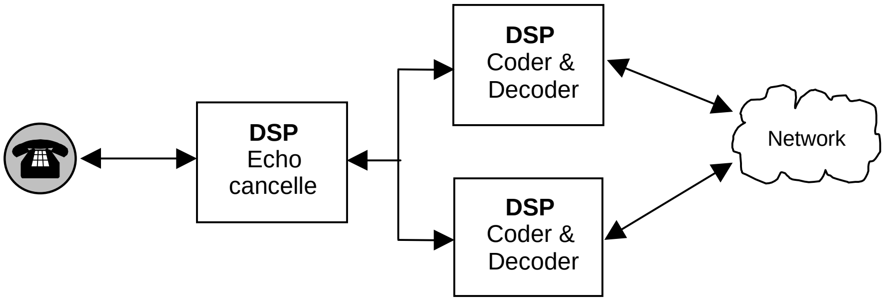
A common solution to this variability is to create a buffer that is first primed by a certain amount of data depending on the amount of variation expected. The voice rate dictates the timely removal of the data from the buffer, but the data input rate follows the variable rate of the network. The key is that the data stored in the buffer absorbs the variability of the input. However, the tradeoff is the delay incurred by buffering a certain amount of voice data. The variables, such as the jitter buffer size, must be determined by statistical monitoring of the network channel. Monitoring of the buffer overrun and underrun can be used to dynamically reconfigure these parameters based on current network delays.

Task Allocation

The discussion has assumed thus far that each DSP in the system performs the same operations and are essentially interchangeable in their roles. This solution has both advantages and disadvantages. The primary advantage is system design and control. Since each DSP performs identical roles, each can be loaded with the same software and can be interconnected with the same configuration. The system controller does not need to differentiate between different tasks and different DSPs. One major disadvantage, however, is software size. Since each DSP must perform all system tasks including voice coding, voice decoding, and echo cancellation, the software for each must reside in memory. This may not be an optimal solution if the software does not fit in internal memory, resulting in extra board space for external RAM and time consuming off-chip accesses.

In such a distributed system, different DSPs would be responsible for different tasks in the data path. For example, tasks could be divided between dedicated echo canceller DSPs and voice coding/decoding DSPs. A purely dedicated approach would be to allocate processors for echo cancellation, voice coding, and voice decoding. This type of configuration offers the advantage of allowing software to be reused for each channel within a processor. However, since the processing of a stream of data involves several DSPs, data must be transferred between them. Consequently, this process requires a longer delay due to the additional overhead required of interprocessor communication. Figure 7 shows a possible configuration for such a distributed approach.

Figure 7. Distributed Software Configuration



To efficiently optimize a distributed system, the processing blocks may be divided unevenly, as shown in Figure 7. Since an echo canceller algorithm generally requires less MIPS or real-time processing than a vocoder, an echo canceller DSP can process more channels in a given time than a vocoder. As a result, the distribution of the tasks may vary depending on algorithm implementation speeds.

Network Interface

The network interface block performs the tasks of sending packetized voice/data from the DSP to the network physical layer and receiving data from the network and delivering it to the DSP. The details of this process depend on the type of network interface used. Two of the most common networks used for such systems are

- ❒ ATM
- ❒ Frame Relay

ATM

Much work has been done on the marriage of voice traffic and ATM. Several forums have been created, most notably VTOA (Voice and Telephony Over ATM), to address implementation and interoperability requirements for such systems. The structure of ATM is well suited for the transmission of voice traffic due to its small data packets, known as *cells*. In fact, ATM was designed to transmit voice data. Small and uniformly sized cells provide the advantage of smaller, more predictable delays, which is essential to the transmission of voice traffic. Cell-based traffic also meshes well with the bursty nature of voice. However, the advantages that small cells provide to voice traffic result in disadvantages to data traffic, since smaller cells subsequently result in a higher overhead to data ratio.

The layers of the OSI network model applicable to this discussion on ATM are the Adaptation layer and the Physical layer. The ATM Adaptation Layers (AAL) provide for the efficient transmission of packets of various types. AAL Type 2 is specifically intended for use in packet voice and video applications and is designed to efficiently support low-rate, short, and variable length packets in delay sensitive applications. The AAL layer is also subdivided into two parts:

- ❒ Common Part Sublayer (CPS)
- ❒ Service Specific Convergence Sublayer (SSCS)

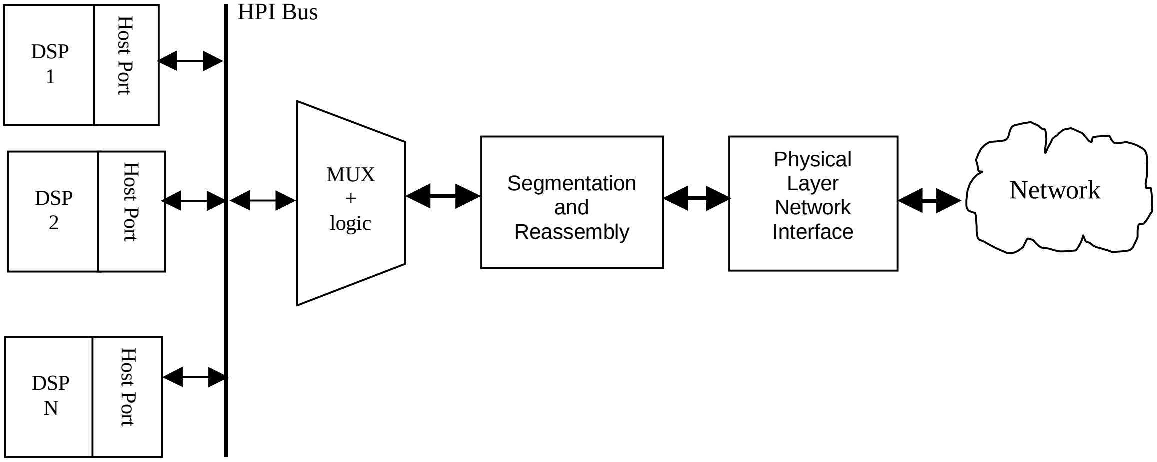
The SSCS layer specifies packet formats and procedures to encode various types of information streams for efficient transport. In other words, this layer is cognizant of the type of information being communicated. In fact, in a voice/data system, it can be viewed as being a part of the signal-processing block. The CPS Layer is closer to the physical layer and thus has the responsibility of transmitting the packet versus the packet information.

The implementation of the ATM interface includes the following tasks:

- ❒ Packetization and depacketization of voice data into properly formatted AAL data units (PDUs) by the DSP
- ❒ Multiplexing of these PDUs to an Segmentation and Reassembly processor
- ❒ Physical layer ATM cell interface

These tasks are also shown in Figure 8.

Figure 8. ATM Interface Block Diagram



Frame Relay

Frame relay is another popular network switching technology used to transmit voice and data. Frame relay is based on the X.25 protocol with several modifications resulting in higher performance and greater efficiency. For example, frame relay does not employ time-consuming data-correction techniques, as does X.25, leaving this to higher layers.

For voice communications, the two main differences between frame relay and ATM-based transmissions are data formatting and service quality. ATM communication is based on small, fixed length packets known as cells for data transmission, whereas frame relay utilizes variable length packets. Since small packets provide several advantages when transmitting voice, frame relay systems generally fragment voice packets into small cell-like packets. Small packets statistically yield better network service by reducing network jitter and delay. Also, the perceptual consequences of dropping small packets are smaller than larger packets.

Another issue encountered in transmitting voice over frame relay systems is the variable nature of data packets sharing the network. Since data transmissions generally utilize larger packet lengths for efficiency, the delays incurred by voice packets can vary widely. This impacts the required real-time operation of voice traffic. Consequently, the ATM protocol was created to solve network issues related to the incompatibility of frame relay and various traffic types such as voice and video.

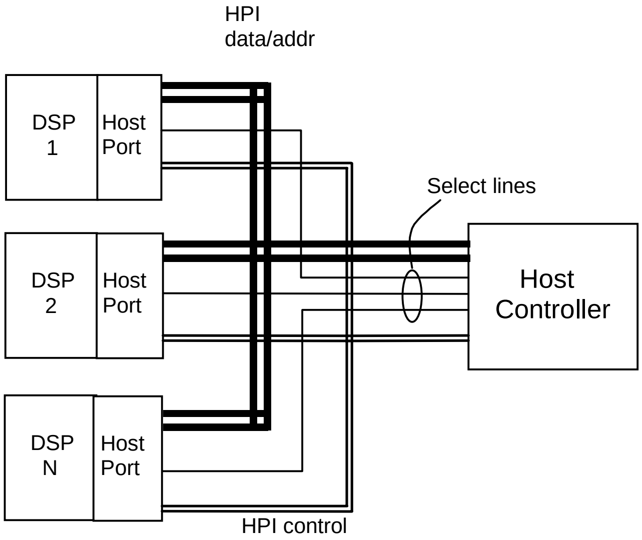
HPI Bus

The next task in specifying a multiprocessor VoN system is to create a mechanism by which the DSPs can share access to a common logic device, such as a microprocessor or ASIC. A bus provides this shared communication link by granting all systems connected to it access based on either a bus master or a set of mutually understood access rules.

Each DSP connects to the bus and ultimately to the controller device via the Host Port Interface (HPI). Essentially, the host buffers and multiplexes the data streams to and from each DSP and provides the interface to the network controller. It may also format the data packets or affix additional header information for routing or error control.

Several issues must be addressed when creating the bus, including device configuration issues, addressing or arbitration, service requests, and circuit loading issues. Figure 9 shows an overview of a general-purpose implementation of such a bus using the HPI.

Figure 9. Host/HPI Interface



For a voice-over-packet system, the most appropriate configuration assigns the DSPs as the bus slaves and the controller device as the bus master. Since all systems are connected to a common physical line, each must transmit at appropriate times to avoid bus contentions. Efficiently distributing the bus accesses among the DSPs is the role of the bus master.

First, the host must employ either a mechanism for device addressing or the means by which the host controller selects and communicates with each individual device on the bus. The fastest solution connects the controller with each DSP via an individual select line. The HPI Chip Select line (HCS) serves this purpose. This allows specific DSPs to be accessed by the controller by simply enabling the HCS line for the appropriate DSP. For host data writes, this method allows the controller to target individual DSPs or to broadcast to multiple DSPs. Such an approach requires that the host have access to each DSP select line. For small numbers of DSPs, either connecting the select lines to GPIO pins or to a memory mapped register works well.

If the number of DSPs becomes large, the number of select lines may render this approach unfeasible, in which case address decode logic may be required. This method offers the advantage of low software overhead required for the DSP to manage the bus. A disadvantage, however small, is the additional overhead of physically supplying and routing the chip selects to each DSP.

Another solution for device addressing is to leave the address decodes to the DSP. For the host to select a DSP for communication, it must first send an address over the bus to select the appropriate DSP. Each DSP subsequently decodes the address and determines if the upcoming data is intended for it, then proceeds as appropriate. The disadvantage of this method is its software complexity. Each DSP must service these bus interrupts to decode the address on the bus for every access, thus wasting MIPS.

This method offers the advantages of versatility and physical simplicity in that none of the DSPs need to have individual connections—all connections can be distributed. Adding additional DSPs to the bus in this solution becomes trivial since no hardware configuration is required. If quick and versatile expandability is a requirement, this may be the best solution.

The next issue arises when the DSP requests delivery of a packet to the host. Given that each DSP is expected to regularly transmit packets with equal frequency as would be expected of a busy, evenly distributed VoN system, an appropriate solution is for the host to process the DSPs in a round robin polling fashion. Each DSP would indicate a service request by setting a flag accessible by the host via the HPI, which the host would routinely poll. If a packet exists to transmit, the host processes it; if not, it continues to the next processor.

Another solution is for the DSPs to actively request service by the host controller, requiring a dedicated interrupt line for each DSP. This method is more appropriate in a situation in which only a subset of the DSPs generally are expected to require service, thus not requiring the host to waste time polling DSPs that normally are idle.

Bus loading is another issue of note. Each device that connects to a bus creates a capacitive and current load. If many devices are connected, this load requires more time to charge, thus causing significant delays in bus transmissions. Each DSP also has limits on current drive, limiting this charging time. Consequently, as devices are added to the bus, drivers or transceivers may be required depending on individual loading characteristics and the number of devices present on the bus. This presents a trade-off decision since additional logic such as drivers and transceivers also create delays.

Conclusion

Designing a multiprocessor platform presents many challenges including, but not limited to, system control, data path architectures, and software design. With the large software base, tools, and flexible peripheral options of the TI C54x family, these challenges become tractable and ultimately superior solutions.

TI Contact Numbers

INTERNET

TI Semiconductor Home Page
www.ti.com/sc

TI Distributors
www.ti.com/sc/docs/distmenu.htm

PRODUCT INFORMATION CENTERS

Americas

Phone	+1(972) 644-5580
Fax	+1(972) 480-7800
Email	sc-infomaster@ti.com

Europe, Middle East, and Africa

Phone	
Deutsch	+49-(0) 8161 80 3311
English	+44-(0) 1604 66 3399
Español	+34-(0) 90 23 54 0 28
Francais	+33-(0) 1-30 70 11 64
Italiano	+33-(0) 1-30 70 11 67
Fax	+44-(0) 1604 66 33 34
Email	epic@ti.com

Japan

Phone	
International	+81-3-3457-0972
Domestic	0120-81-0026
Fax	
International	+81-3-3457-1259
Domestic	0120-81-0036
Email	pic-japan@ti.com

Asia

Phone	
International	+886-2-23786800
Domestic	
Australia	1-800-881-011
TI Number	-800-800-1450
China	10810
TI Number	-800-800-1450
Hong Kong	800-96-1111
TI Number	-800-800-1450
India	000-117
TI Number	-800-800-1450
Indonesia	001-801-10
TI Number	-800-800-1450
Korea	080-551-2804
Malaysia	1-800-800-011
TI Number	-800-800-1450
New Zealand	000-911
TI Number	-800-800-1450
Philippines	105-11
TI Number	-800-800-1450
Singapore	800-0111-111
TI Number	-800-800-1450
Taiwan	080-006800
Thailand	0019-991-1111
TI Number	-800-800-1450
Fax	886-2-2378-6808
Email	tiasia@ti.com

TI is a trademark of Texas Instruments Incorporated.

Other brands and names are the property of their respective owners.

IMPORTANT NOTICE

Texas Instruments and its subsidiaries (TI) reserve the right to make changes to their products or to discontinue any product or service without notice, and advise customers to obtain the latest version of relevant information to verify, before placing orders, that information being relied on is current and complete. All products are sold subject to the terms and conditions of sale supplied at the time of order acknowledgement, including those pertaining to warranty, patent infringement, and limitation of liability.

TI warrants performance of its semiconductor products to the specifications applicable at the time of sale in accordance with TI's standard warranty. Testing and other quality control techniques are utilized to the extent TI deems necessary to support this warranty. Specific testing of all parameters of each device is not necessarily performed, except those mandated by government requirements.

CERTAIN APPLICATIONS USING SEMICONDUCTOR PRODUCTS MAY INVOLVE POTENTIAL RISKS OF DEATH, PERSONAL INJURY, OR SEVERE PROPERTY OR ENVIRONMENTAL DAMAGE (“CRITICAL APPLICATIONS"). TI SEMICONDUCTOR PRODUCTS ARE NOT DESIGNED, AUTHORIZED, OR WARRANTED TO BE SUITABLE FOR USE IN LIFE-SUPPORT DEVICES OR SYSTEMS OR OTHER CRITICAL APPLICATIONS. INCLUSION OF TI PRODUCTS IN SUCH APPLICATIONS IS UNDERSTOOD TO BE FULLY AT THE CUSTOMER'S RISK.

In order to minimize risks associated with the customer's applications, adequate design and operating safeguards must be provided by the customer to minimize inherent or procedural hazards.

TI assumes no liability for applications assistance or customer product design. TI does not warrant or represent that any license, either express or implied, is granted under any patent right, copyright, mask work right, or other intellectual property right of TI covering or relating to any combination, machine, or process in which such semiconductor products or services might be or are used. TI's publication of information regarding any third party's products or services does not constitute TI's approval, warranty, or endorsement thereof.