NSC's Digital Answering Machines Solution

Ohad Falik, Gideon Intrater,
National Semiconductor (I.C.) Ltd.
P.O.Box 3007, Herzelia B. 46104, ISRAEL
Email : cohfta@taux01.nsc.com

Abstract

Digital Answering Machines (DAMs) store messages in solid state memories instead of on magnetic tapes, thereby improving their reliability, and increasing the number of features they can support. A digital signal processing capability that enables data compression, and the use of cost effective memories are fundamental to DAMs. National Semiconductor developed a three chip cluster that is tuned to the DAM application. This DAM cluster based on the NS32AM160, offers a solution to both the required processing power, and system integration. This paper describes the requirements and structure of the DAM application. It identifies tasks in that application, and distributes them among the elements of a DAM system. Finally, it describes the integration of those elements in the NSC DAM solution, and describes NSC's DAM cluster in detail.

Keywords: Consumer Electronics, Embedded Processors, DSP, Digital Answering Machines.

1.0 Introduction

Traditional Answering Machines (AMs) use two magnetic tape recorders, one for incoming messages and the other for an outgoing message. These machines are unreliable, complicated and relatively expensive. A newer generation of AMs uses digital speech storage devices for the outgoing message. However, these AMs still have most of the reliability problems caused by magnetic tape wear and mechanical malfunctions because they still require one tape recorder [1]. Typically, these AMs are limited to a few simple features such as play, erase, save all messages, remote control operations and memo recording.

Digital Answering Machines (DAMs) use digital solid state storage devices to hold both incoming and outgoing messages. The main issue when solid state memories are used is the volume of the data. The standard bit rate for voice representation on digital telephone lines is 64 Kbits/sec. Therefore, a one minute recording requires almost 4 Mbits of storage memory. The amount of memory required to store a few minutes of recorded raw data makes DAMs very expensive. First generation DAMs used improved sampling techniques like Adaptive Delta Modulation, which improved the bit rate (22 Kbits/sec) but reduced voice quality. Newer DAMs use voice compression algorithms to improve voice quality, and compress the data even further. This paper describes this type of DAMs and National Semiconductor's solution for efficient implementation of the DAM application.

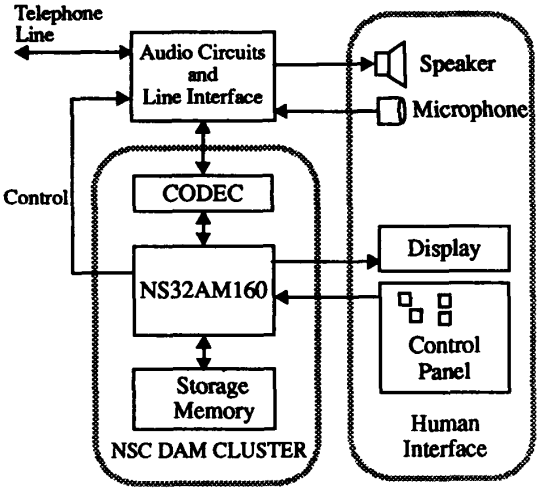

Figure 1: Digital Answering Machine System Block Diagram

Voice compression algorithms are Digital Signal Processing (DSP) based methods for processing voice data and reducing its bit rate. Sophisticated algorithms may reduce the bit rate by a factor of 5 to 8, while preserving the quality of the voice. Voice compression algorithms and powerful DSP processors are fundamental to DAM systems. More processing power in the system makes it possible to add various features to the DAM. For example, it is very useful to know the time of arrival of a message. Voice Synthesis (VS) algorithms may be used to announce the time of arrival of a message before message playback. Storing messages in random access memory makes it possible to add features such as selective erasing and playback of mes-

sages. Remote control of DAM functions is possible using touch-tone dialing (DTMF). A voice synthesized menu can guide the user to the desired control option.

The National Semiconductor (NSC) DAM solution is implemented by a three chip cluster that uses the NS32AM160 processor as its base. It also includes a COder/DECoder (CODEC) and a DRAM. NSC's NS32AM160 is an embedded processor that is tuned especially for the DAM application, but may be used for other applications as well. The processor handles system control, voice compression/decompression/synthesis, DTMF and memory management functions; the CODEC performs A/D and D/A conversion; and the DRAM stores the messages.

Section 2.0 of this paper describes the structure of a typical DAM application. Section 3.0 analyzes NSC's approach to a DAM solution. Section 4.0 describes the structure of the chip that resulted from that analysis, and Section 5.0 describes the NS32AM160 DSP engine.

2.0 DAM System Structure

A block diagram of a DAM system is shown in Figure 1. The COder/DECoder (CODEC) connects the analog and digital parts of the DAM. It converts analog voice signals to digital data, and vice versa. The CODEC is controlled by the NS32AM160 processor. An incoming message, arriving through the telephone line, is connected to the CODEC input by the audio circuits and line interface. The incoming message may also be concurrently directed to the speaker, so that it can be heard while it is recorded. Another path is used for the outgoing message, which is generated by the processor. After D/A conversion in the CODEC, the outgoing message is passed to the telephone line via the audio circuits and line interface. Other audio paths are also possible, for example from the microphone to the CODEC input to record outgoing messages.

Analog circuits switch the audio signal among the different audio sources and destinations. The processor controls the path of the signal using analog switches activated by the output ports of the processor. In addition, the analog circuits perform other tasks such as ring detection and gain control.

Recorded messages are stored in memory. The volume of data that must be stored depends on the recording time required and the compression rate. For example, the amount of memory required for a seven minute recording, at a bit rate of 10 Kbits/sec, is about 4 Mbits. To store this volume of data economically on solid state memory devices, DRAMs are used. Error tolerant voice compression algorithms enable the use of memory devices that lose some of the stored data. Audio quality DRAMs (ARAMs) are DRAMs that contain some defective memory locations.

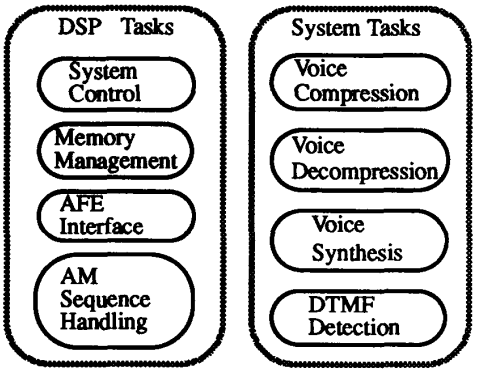

Figure 2: DAM software functions.

By using ARAMs with lower access speeds, the cost of memory is significantly reduced.

Interfacing with the user is another important task in the DAM application. DAM functions and operation modes are controlled locally via push-buttons and switches. Remote control of the DAM is performed via DTMF signaling through the telephone line. The status of the DAM and other information such as the number of recorded messages, are displayed on an LCD or LED display. Some information is presented orally using Voice Synthesis (VS), either for remote users or to simplify the local display.

The DAM application also includes software functions. Figure 2 shows two groups of tasks that are performed in the DAM application. DSP tasks include voice compression and decompression, VS and DTMF detection. These tasks usually involve massive computations, and their requirements should be carefully analyzed. System tasks include system control, memory management, Analog Front End (AFE) interface and AM sequence handling. System control includes handling of the audio circuits, telephone line interface and human interface. The memory management task arranges messages in memory. The AFE interface handles the CODEC read and write buffers and initialization. AM sequence handling accepts input such as line and key status, and activates the other functions accordingly.

For example, if the DAM is waiting for an incoming call, and the ring detector identifies a ring, the AM sequence handler will start the following sequence of events. A line connect command will be issued, and the outgoing message will be played. Simultaneously, DTMF detection will be activated to identify remote commands. After the outgoing message is finished playing, if no remote command was received, a tone generation command will be issued by the sequence handler to produce a beep. Then, the sequence handler will instruct the system control block to configure

the system to record a message, and will activate the voice compression algorithm. When message recording is completed, the sequence handler will disconnect the phone line, and return to the "waiting for an incoming call" state.

We analyzed this typical DAM system to design a highly integrated DAM solution that would include system control, voice compression/decompression, message storage, VS and human interface.

3.0 Defining the NSC DAM Cluster

The three chips in NSC's DAM cluster were defined based on the application requirements described above, and using a special methodology for developing an application specific processor that is based on a general purpose microprocessor core. NSC's three chip DAM cluster is illustrated in Figure 3. A complete DAM solution also requires the definition of system integration logic, system features, and the application development environment, which are described in the next section.

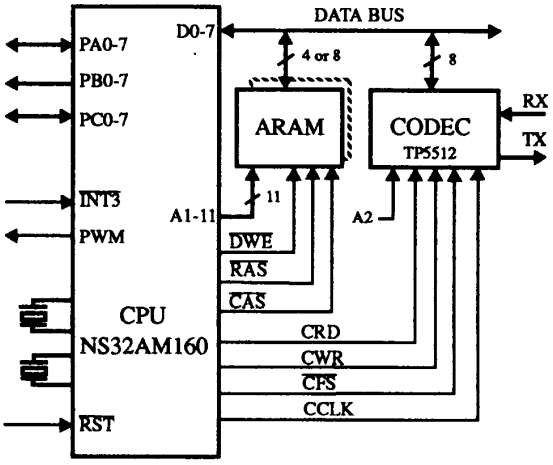

Figure 3: National Semiconductor's Digital Answering Machine Cluster

Methodology

An existing general purpose microprocessor (CPU core) is tailored to a specific application in two steps [2]. In the first, Performance Acceleration Modules (PAMs) are defined, in conjunction with the CPU core. PAMs are hardware modules designed to execute a set of specific functions faster than the CPU core. Integration of PAMs enhances the processing power of the CPU. This allows real-time execution of the application.

In the second step, System Integration Modules (SIMs) are defined. SIMs are on-chip modules designed to replace

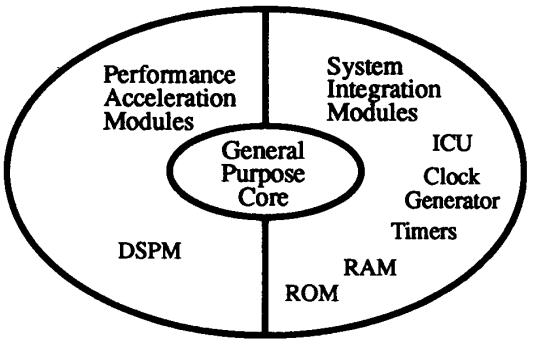

Figure 4: Structure of an Application Specific Processor.

external hardware such as timers, DRAM controller and Interrupt Control Unit. SIMs are traditionally part of the application's board and are integrated on-chip to minimize the total system cost. Figure 4 illustrates the distribution of functions on an application specific processor. In the DAM application, SIM definition included the distribution of functions among the three chips in the cluster.

The CPU Core and the PAM

First we analyzed the DAM application and its time constraints. In the DAM application the performance critical tasks are voice compression, voice decompression, voice generation and DTMF functions. It is important to note that none of these voice functions is performed concurrently, but DTMF detection is concurrent with them all.

A voice compression/decompression algorithm was selected that compresses data that was sampled at 64 Kbits/sec to about 13 Kbits/sec, and has high error immunity. This algorithm uses fixed-point real arithmetic in its DSP functions. A DTMF algorithm was selected that uses fixed-point complex arithmetic. VS computation requirements are relatively low when compared with the compression algorithm, and involve integer and fixed-point calculations.

In selecting a CPU core as the base for the new application specific processor, the NS32AM160, we looked for a low cost CPU core with medium range performance that would support the DSP and system tasks. We chose the CG core [2,3], which consists of a 32-bit execution unit, an instruction prefetch queue and a bus interface unit. All these units operate concurrently to achieve an average of 1-2.5 MIPS at 10-25 Mhz. Floating-point instructions not used by the application were removed from the core, and interrupt latency was reduced by using direct exception [4].

Because of the vectorial nature of the DSP algorithms and the typical bus load of the CG core, we decided to use the DSP Module (DSPM) as the PAM [4]. The DSPM is capable of autonomous operation in parallel with CPU core operation. The DSPM executes programs stored in an inter-

nal on-chip RAM, and manipulates data stored either in the internal RAM or in an external off-chip memory. To maximize utilization of hardware resources, the DSPM contains a pipelined DSP oriented datapath, and a control logic that implements a set of DSP vector commands. The DSPM is described in detail in Section 5.0.

Memory

The data types used in the DAM are program code, program working area, fixed data and voice data. The program code and fixed data are placed in an on-chip ROM. The program working area is located in an on-chip RAM, while the voice data is placed in an ARAM device.

ARAM specifications were defined according to the algorithm's error immunity, and the ARAM's access density. This density depends on the voice compression ratio and the ARAM word size. Using the selected compression algorithm, 4-bit words can be accessed at 3.3 K access/sec. This allows use of slow ARAM devices without adversely affecting performance. The 4-bit wide bus makes it possible to have 4 Mbits of memory using only one device. The system may be configured to have 1 to 32 Mbits of ARAM, using 1 to 8-bit memory words, and ARAM devices with address space of 1 to 4-Mbit addresses.

CODEC

For the CODEC function, we needed a low cost single chip device that meets telephone line standards. We selected the TP5512, which consists of an 8-bit μ-law coded monolithic PCM COMBO that performs both A/D and D/A conversion, and utilizes a parallel I/O microprocessor interface[5]. It includes integrated filters for both input and output that meet telephone line standards. A CODEC conversion rate of 8 KHz covers the telephone line audio spectrum (300 Hz to 3400 Hz).

4.0 NS32AM160 System Module Integration

After we determined the distribution of system functions among the system components, we defined the NS32AM160 SIM modules. To achieve maximal system integration, the SIMs should contain as much of the system interface as possible. The NS32AM160 SIMs are the on-chip memories, the Interrupt Control Unit (ICU), the clock generator, the WATCHDOG (WD) timer, the Pulse Width Modulator (PWM), the I/O ports and a Bus Interface Unit (BIU) tailored for the CODEC and ARAM (DRAMC) interfaces. A block diagram of the NS32AM160 is shown in Figure 5 [6].

The NS32AM160 includes both RAM and ROM on-chip. The 2.1 Kbyte on-chip RAM has two parts, one used by the CG core, and the other by the DSPM. The 25 Kbyte of

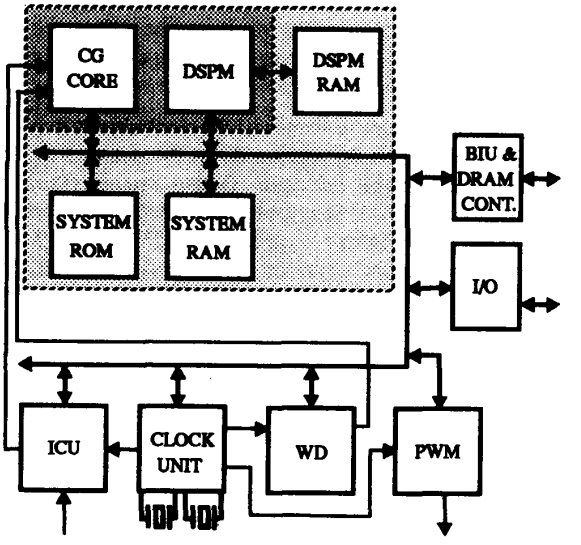

Figure 5: NS32AM160 Block Diagram

ROM holds both code and constant data, such as the Voice Synthesis vocabulary. Locating both RAM and ROM internally eliminates the need for external memories other than the ARAM.

The NS32AM160 supports four vectored interrupts. The ICU uses four interrupt input pins to handle the non-vectored interrupts according to a fixed priority scheme. The interrupts are assigned to CODEC interface (highest priority), external source, DSPM or 500 Hz system clock (lowest priority). Each interrupt source may be masked independently, this makes it possible to disable the external interrupt source in case it is not required, and enables support of interrupt nesting. The DSPM interrupt enables efficient parallel operation of the CG core and the DSPM. System real-time functions and timekeeping are performed in the system click interrupt handler. The 500 Hz pace is determined by scaling down the CPU main clock.

Three types of Non-Maskable Interrupts (NMIs) are used for DSPM exception handling and the WATCHDOG mechanism. The DSPM uses NMI for debugging, for breakpoints, and for detecting illegal access attempts by the core during DSPM operation. The WATCHDOG mechanism can be used to prevent the program from losing control, and to reactivate the core during power down mode.

One of the most important tasks of the NS32AM160 is system control. 24 I/O pins are used for this purpose. 16 of these pins may be individually configured as input or output, and 8 are dedicated output signals. The I/O signals are defined by the system configuration to perform such tasks as scanning push buttons, driving the display and controlling the analog switch.

Another useful feature available in the NS32AM160 is a Pulse Width Modulator (PWM). It may be utilized for low speed A/D and D/A conversions using an external resistor-capacitor network.

The clock generator generates all the clock signals required by the application, by scaling down the output from the 40.96 MHz crystal oscillator. During normal operation mode, the NS32AM160 operates on a 20.48 MHz clock. In order to allow battery backup, the NS32AM160 has a power down operation mode. To support this mode, the NS32AM160 has low frequency oscillator input (455 KHz), in addition to the normal operation clock. The NS32AM160 is switched to power down mode by a software command. In power down mode the core clock is slowed, and some of the modules are stopped. The DRAM can not be accessed during power down mode, but its content is preserved by continuing the refresh. Power consumption can be further reduced, by executing the processor WAIT command. This causes the core to stop executing, and to wait for an interrupt, which is issued every 0.1 second by the WATCHDOG unit.

The DRAM controller block manages ARAM read, write and refresh bus transactions. Bus read and write transactions that require four cycles in the CG core are translated by the DRAM controller to ARAM bus cycles. Contention between core access and DRAM refresh cycles is resolved by stalling the core using its $\overline{RDY}$ input. The DRAM controller also decodes addresses. DRAM address mapping is fixed, and DRAM is accessed like a memory-mapped peripheral. Since ARAM holds only data, the data does not have to be in consecutive addresses. As a result, the external memory bus width can be configured without changing the DRAM controller.

The CODEC interface translates the CG bus cycles to TP5512 transactions. It generates the special read and write signals required after accessing the CODEC address. The TP5512 requires two clock signals for the conversion circuits. These two clock signals are generated in the NS32AM160 by scaling its main clock down, without any external component. One of these clocks ($\overline{CFS}$) defines the initiation and completion of the conversions. It is used internally to interrupt the ICU, enabling efficient interface to the CODEC.

The NS32AM160 has three configuration modes: internal ROM, external ROM and development. The active configuration is selected during reset, by pull-up/pull-down resistors. In internal ROM mode, maximal system integration is achieved, by the three chip cluster described above in which both ROM and RAM are located on-chip. In external ROM mode an off-chip ROM device can be used to enable small production lines, and to provide more ROM than is available on-chip. In development mode, the NS32AM160

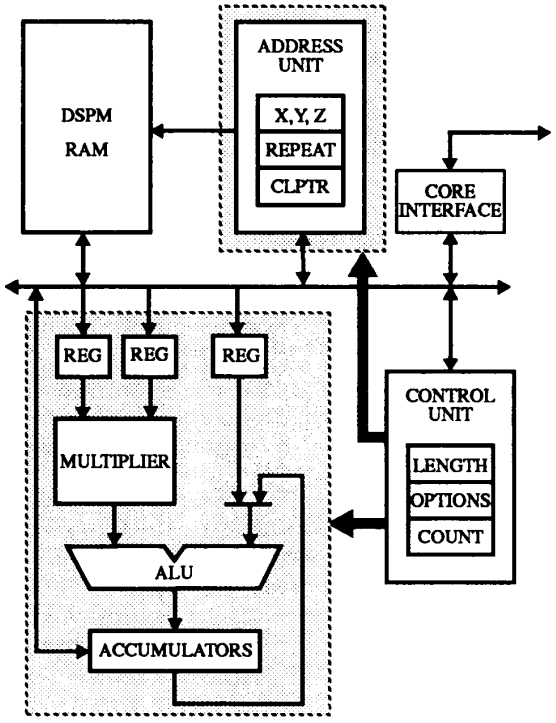

Figure 6: DSPM Block Diagram

provides a flexible bus to the user's system, which enables connection of off-chip RAM, ROM and other peripheral devices to the NS32AM160.

Development mode is used during the software development phase, when off-chip RAM is loaded with the application code. In this mode, the NS32AM160 provides hooks for connecting peripherals that are required in the development system, or for using the NS32AM160 in applications other than DAMs.

5.0 The DSP Module

The DSP Module (DSPM) is a complete processing unit, capable of autonomous operation, parallel to the CPU core operation. The DSPM executes programs stored in internal on-chip RAM, and manipulates data stored either in internal RAM or in external off-chip memory. To maximize utilization of hardware resources, the DSPM contains a pipelined DSP-oriented datapath with a hardware multiplier, and a control logic that implements a set of DSP vector commands. The DSPM block diagram is illustrated in Figure 6.

Computations are performed by vector commands. These commands employ the DSP-oriented datapath in a pipe-

lined manner, thereby maximizing the utilization of on-chip hardware resources. A set of dedicated registers is used to specify operands and options for subsequent vector commands. These dedicated registers can be loaded and stored by appropriate commands between initiations of vector commands. Additional commands are available for controlling the flow of execution of command lists (DSPM programs), for example, for programming loops and branches.

Internal RAM is used by the DSPM for fetching commands to be executed, and for reading or writing data that is required in the course of program execution. DSPM programs are encoded as command lists, and are interpreted by the command list execution unit.

The CPU core interface maps the DSPM internal RAM as a contiguous block within the CPU core's address space. This makes it possible for normal CPU instructions to access and manipulate data and commands in the DSPM internal RAM. In addition, the CPU core interface contains control and status registers that are needed to synchronize the execution of CPU core instructions concurrently with execution of DSPM command lists.

Execution of a command list begins when the CPU core writes a value into the CLPTR control register. This causes the DSPM command list execution unit to begin executing commands, starting at the address written to the CLPTR register.

Once started, execution of the command list continues until a HALT command is executed. The CPU core can either poll the DSPM status register to detect the end of execution, or wait until the DSPM issues an interrupt upon termination of command list execution.

An example of one of the DSPM vector instructions is the Vector Complex Multiply and ACcumulate (VCMAC) instruction. This vector operation performs an FIR filter on complex data. The VCMAC instruction performs a convolution sum of the X and Y complex vectors, and stores the result in Z[0].

```
{
  complex  X,Y,Z ;
  complex_acc  A ;
  for (n=0; n < LENGTH; n++ )
  {
    A.re =A.re +(X[n].re*Y[n].re - X[n].im*Y[n].im) ;
    A.im =A.im +(X[n].re*Y[n].im + X[n].im*Y[n].re) ;
  }
  Z[0]=(complex)A;
}
```

The execution unit of the DSPM consists of four independent units that operate in a pipelined manner. The units can execute any of the following in parallel: one load or store operation, an address pointer increment operation, a multiply operation and an ALU operation. Table 1 shows the DSPM pipeline during the execution of a VARiable Multiply and Accumulate (VARMAC) instruction. This instruction performs repeated iterations of the following operation:

A = A + X[n].low * Y[n].low + X[n].high * Y[n].high

Cyc	Load/Store	Inc.	MUL	ALU
1	Load X[0]			
2	Load Y[0]	X		
3	Load X[1]	Y	temp1 = X[0].low *Y[0].low	
4	Load Y[1]	X	temp2 = X[0].high *Y[0].high	A += temp1
5	Load X[2]	Y	temp1 = X[1].low *Y[1].low	A += temp2
6	Load Y[2]	X	temp2 = X[1].high *Y[1].high	A += temp1

TABLE 1. VARMAC pipeline execution

6.0 Summary

Digital Answering Machines (DAMs) provide more features and are more reliable than traditional answering machines. National Semiconductor's solution for the DAM application, the NSC DAM three-chip cluster based on the NS32AM160, offers a highly integrated cost-effective solution to the DAM application.

To define the DAM cluster, we analyzed the requirements and structure of the application. We identified tasks in the application, distributed them among the elements of the system, and integrated those elements into the cluster components. This methodology used by National Semiconductor offers a constructive approach to embedded processor definition, and may be used in other applications as well.

References

[1] T. Weaver. A digital-storage answering machine using a national 32000-family microprocessor. In 1991 Microprocessor Forum, San Francisco, CA, November 1991.

[2] G. Interater and D. Biran. Application specific micro processors. In 1990 IEEE International Conference on Computer Design: VLSI in Computers and Processors, Cambridge, MA September 1990.

[3] Embedded System Processors Databook. National Semiconductors, 1989.

[4] NS32CG160 Data Sheet. National Semiconductor, 1991.

[5] TP5512 Data Sheet. National Semiconductor, 1991.

[6] NS32AM160 Data Sheet. National Semiconductor, 1991.