use of 32-bit arithmetic is particularly crucial for IIR filtering. We examine this case in Section 6.5.

6.3 OVERVIEW OF REAL-TIME PROCESSING

This section discusses important topics on real-time signal processing and defines the terms used in measuring real-time performance. We introduce concepts of real-time and non-real-time processing and explain the sample-by-sample and block processing modes. To benchmark real-time performance in embedded signal processing, we measure the cycle counts of running different tasks on the Blackfin processor. A brief introduction of the power measurement and its relation to the processing speed is also given, but a detailed discussion on power-saving features of the Blackfin processor is postponed to Chapter 8.

6.3.1 Real-Time Versus Offline Processing

A real-time system processes data at a regular and timely rate. As shown in Figure 6.25, the inputs of a real-time system are often associated with signal capturing devices like microphones, cameras, thermometers, etc. The inputs can also come from digital media streaming devices like audio and video players. The outputs can be devices like loudspeakers, video display, etc. that play back the processed signals.

To be more explicit, a real-time system must satisfy certain response time constraints. The response time is defined as the time between the arrival of input data sample(s) and the output of processed data sample(s). For example, the response time constraint for a typical speech processing system is to digitize the analog speech, process the digital speech, and output the processed signal within a given sampling period. If the response time exceeds the sampling period, the new speech sample cannot be retrieved on time and thus violates the real-time constraint. A more detailed explanation is provided in Sections 6.3.2 and 6.3.3.

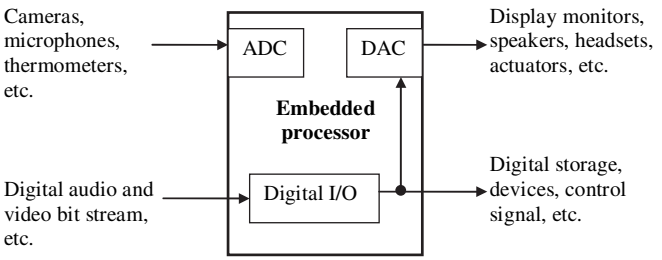

Figure 6.25 A typical real-time system that receives, processes, and transmits data

In contrast, an offline processing system is not required to complete the task within allocated time. For example, we can sample a noisy speech signal, save it in a data file, and run a program on a computer to read the speech samples from the file and perform noise reduction. After processing, we can store the clean signal to a data file and play it back with a loudspeaker. In this way, there is no timing constraint to complete the overall receive-process-transmit chain. Offline processing always involves extensive memory storage, and it is often used in film and music postproduction. The Blackfin simulator can be used to simulate DSP algorithms in an offline manner because the test data are stored in a data file and no time constraint is imposed on the processing. However, when the same algorithm is implemented on the Blackfin processor, real-time processing must be carried out on the incoming signal.

6.3.2 Sample-by-Sample Processing Mode and Its Real-Time Constraints

The sample-by-sample processing mode requires that all operations must be completed within the given sampling period. As shown in Figure 6.26, an audio signal can be sampled at a sampling period of every T_s seconds. Latency (or response time) of processing can be defined as the total time from the instant the data sample is read in to the time the digital output is written to the memory. This latency contains the three subintervals listed below:

1. T_{in} is the time needed for the processor to copy the current sample from the ADC into the processor memory. It also includes the program access time.
2. T_{sp} is the time needed for processing the current data samples. This duration depends on the complexity of the algorithm and the efficiency of the program.
3. T_{out} is the time needed to output the processed data to the DAC.

The overall overhead time for sample-by-sample processing is denoted as T_{os}, which includes both T_{in} and T_{out} and the response time to interrupt. At the beginning of every sampling interval, the ADC samples new data, and the DAC sends out the processed data.

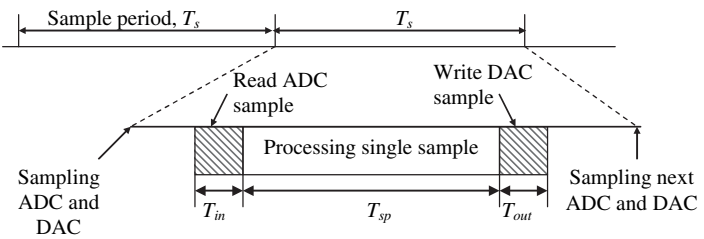

Figure 6.26 Timing details for sample-by-sample processing mode

The real-time constraint of the sample-by-sample processing is that the processing time T_{sp} must satisfy

$$T_{sp} \leq T_s - T_{os}. \tag{6.3.1}$$

The advantages of using sample-by-sample processing are:

1. Delay between the input and the output is always kept within one sampling interval.
2. Single-sample storage for input and output samples. In some applications, multiple channels are acquired, processed, and output within the sampling interval. The memory requirement is increased to store the multichannel data samples.
3. Results are kept current within the sampling period.

The disadvantage of sample-by-sample processing is the overhead of program setup, program access, and latency in reading a new data sample and writing the processed sample in every sampling interval. The processor must be fast enough to complete all processing tasks before the arrival of the next input sample to avoid distortion caused by missing data. A possible method to reduce the processing speed requirement is to group a block of data samples and perform processing on this block of data as a batch. The block processing method is introduced in Section 6.3.3.

In addition, some DSP algorithms, such as the fast Fourier transform described in Chapter 3, require a block of data samples for processing. In this type of block processing algorithm, sample-by-sample processing cannot be implemented and the block processing mode must be applied.

6.3.3 Block Processing Mode and Its Real-Time Constraints

In the block mode, data samples are gathered into an input buffer of N samples and a whole block of samples are processed after the buffer is full. At the same time, a new block of N samples are acquired. Figure 6.27 shows block processing for a block of five samples. The block processing system starts by sampling the first five input samples from the ADC to form block i. A more detailed description of how data can be acquired into the processor is presented in Chapter 7. The system continues to sample another five data samples to form block $i + 1$. At the same time, the processor operates on data samples in block i and sends the five previously processed samples to the DAC. During the next block period, $i + 2$, another five newer samples are acquired. The processor operates on the data samples in block $i + 1$ and outputs the processed data samples in block i. Therefore, the data samples are output to the DAC after a delay of $2NT_s$ seconds.

To perform the block processing shown in Figure 6.27, we need to continuously save data samples in different memory buffers. Therefore, the memory requirement is increased. A memory buffering approach known as double (or ping pong) buffer-

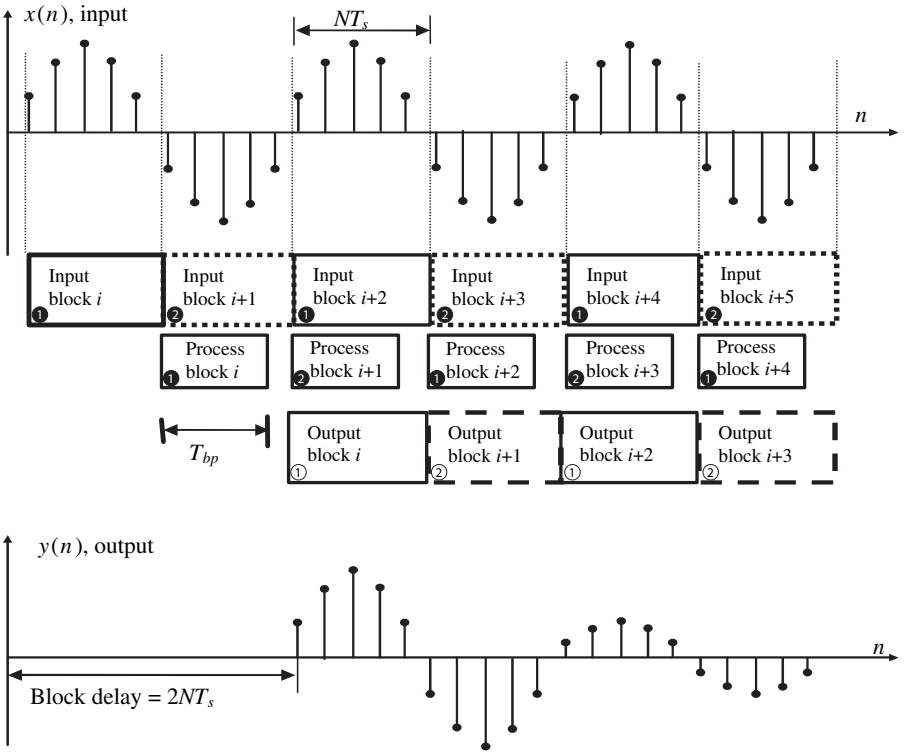

Figure 6.27 Block processing mode ($N = 5$)

ing is recommended. As shown in Figure 6.28, double buffering uses two memory buffers of length N for the input of data and another two buffers of the same length for the output. When the processor is operating on data in buffer(in) ❶, new input samples $x(n)$ are saved in buffer(in) ❷. The function of these two buffers is alternated every NT_s seconds. This "ping-pong" switching mechanism between data acquisition and processing is shown in Figure 6.28, with the labels identifying the buffer used in every block. In the same fashion, the output of data to the buffer and the sending of data out to the DAC are also alternated between two output buffers, buffer(out) ① and buffer(out) ②.

To meet real-time constraints for block processing shown in Figure 6.27, the computational time for block processing T_{bp} must satisfy

$$T_{bp} \leq NT_s - T_{ob}, \tag{6.3.2}$$

where NT_s is the block acquisition time in seconds and T_{ob} is the overhead for block processing, which is mainly caused by program setup and the response time to get data in and out of the processor. In general, the overhead for block processing is lower than for sample-by-sample processing because the program access and setup time for block data transfer is lower than for single-sample transfer. That is,

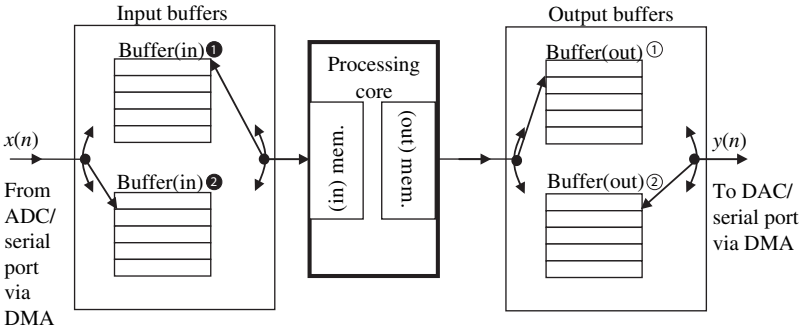

Figure 6.28 Implementation of double buffering

$$T_{ob} < NT_{os}. \tag{6.3.3}$$

Therefore, more time is available for the processor to process signal. However, the disadvantages of using block processing are:

1. Four memory buffers of length N are required for holding input and output data samples with the double-buffering method. In addition, another two memory buffers (in and out) are needed for internal processing by the processor. A detailed explanation of the data acquisition program and how to reduce the memory buffer is given in Chapter 7.
2. A delay of $2NT_s$ is incurred in block processing.
3. More complicated programming is needed to manage the switching between buffers.

A detailed introduction on setting up ADC and DAC to transfer data samples into the internal memory of processor using the serial ports and the DMA is provide in Chapter 7.

QUIZ 6.9

An analog signal is sampling at 48 kHz. Frequency analysis using FFT is applied to a buffer of 5 ms.

1. How many data samples in the 5-ms buffer?
2. What is the order of the FFT that can be used if the FFT is based on the radix-2 algorithm?
3. What is the frequency resolution?
4. What is the memory requirement if double buffering is used?
5. What is the maximum time available for the processor to compute FFT?

The core clock frequency F_{core} used by the processor and the sampling frequency f_s of the real-time system determine the total cycle counts possible within