**Course/Topic:** DSP / DSP processors
**Course Outcome:**
**Activity Chosen:** Think pair share
**Faculty:** N.Mary Leena
Think pair share**:**

Think-pair-share (TPS) is a collaborative learning strategy where students work together to solve a problem or answer a question about an assigned reading. This strategy requires students to (1) think individually about a topic or answer to a question; and (2) share ideas with classmates. Discussing with a partner maximizes participation, focuses attention and engages students in comprehending the reading material. Hence this topic was chosen.

**Details of the Implementation:**

The activity was planned for duration of 40 minutes. First the students were asked to think about the topic given. Then they were paired among their nearby students. They should then share the ideas that they have recollected and list it down in a paper. Later some among the students (pair) were asked to come front and share their ideas that they have written with the entire class.

A digital signal processor (DSP) is a specialized microprocessor (or a SIP block), with its architecture optimized for the operational needs of digital signal processing. The goal of DSP is usually to measure, filter or compress continuous real-world analog signals. Most general-purpose microprocessors can also execute digital signal processing algorithms successfully, but may not be able to keep up with such processing continuously in real-time. Also, dedicated DSPs usually have better power efficiency, thus they are more suitable in portable devices such as mobile phones because of power consumption constraints. DSPs often use special memory architectures that are able to fetch multiple data or instructions at the same time.

Circular Buffering Digital Signal Processors are designed to quickly carry out FIR filters and similar techniques. To understand the hardware, we must first understand the algorithms. In this section we will make a detailed list of the steps needed to implement an FIR filter. In the next section we will see how DSPs are designed to perform these steps as efficiently as possible. To start, we need to distinguish between off-line processing and real-time processing. In offline processing, the entire input signal resides in the computer at the same time. For example, a geophysicist might use a seismometer to record the ground movement during an earthquake. After the shaking is over, the information may be read into a computer and analyzed in some way. Another example of off-line processing is medical imaging, such as computed tomography and MRI. The data set is acquired while the patient is inside the machine, but the image reconstruction may be delayed until a later time. The key point is that all of the information is simultaneously available to the processing program. This is common in scientific research and engineering, but not in consumer products. Off-line processing is the realm of personal computers and mainframes. In real-time processing, the output signal is produced at the same time that the input signal is being acquired. For example, this is needed in telephone communication, hearing aids, and radar. These applications must have the information

immediately available, although it can be delayed by a short amount. For instance, a 10 millisecond delay in a telephone call cannot be detected by the speaker or listener. Likewise, it makes no difference if a radar signal is delayed by a few seconds before being displayed to the operator. Real-time applications input a sample, perform the algorithm, and output a sample, over-and-over. Alternatively, they may input a group of samples, perform the algorithm, and output a group of samples. This is the world of Digital Signal Processors.

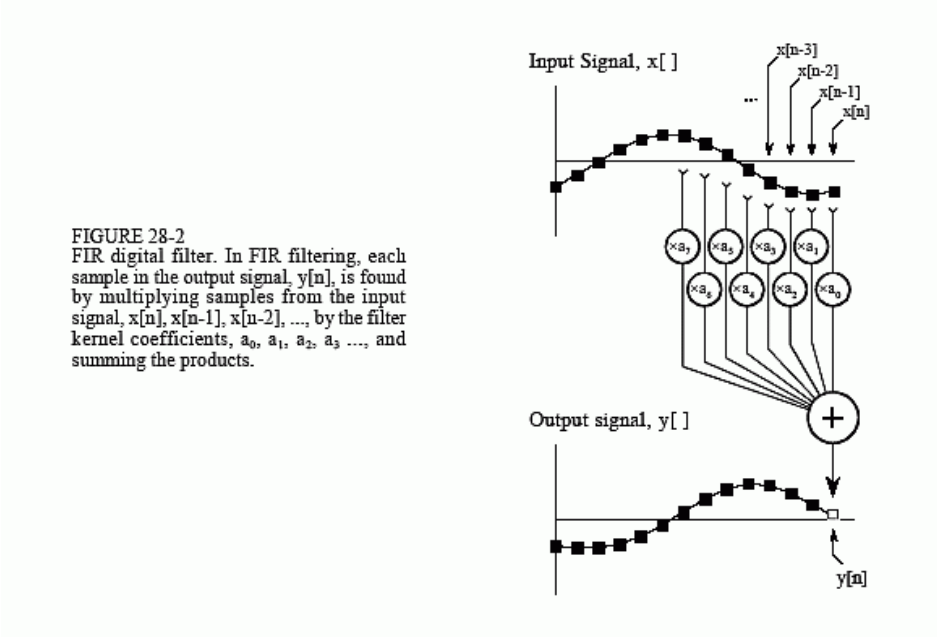


FIGURE 28-2
FIR digital filter. In FIR filtering, each sample in the output signal, y[n], is found by multiplying samples from the input signal, x[n], x[n-1], x[n-2], ..., by the filter kernel coefficients, $a_0$, $a_1$, $a_2$, $a_3$ ..., and summing the products.

Now look at Fig. 28-2 and imagine that this is an FIR filter being implemented in real-time. To calculate the output sample, we must have access to a certain number of the most recent samples from the input. For example, suppose we use eight coefficients in this filter, a0, a1, … a7. This means we must know the value of the eight most recent samples from the input signal, x[n], x[n1], … x[n-7]. These eight samples must be stored in memory and continually updated as new samples are acquired. What is the best way to manage these stored samples? The answer is circular buffering.