Customized System Design Example
Implementation of an Oversampling Digital Filter for ADC

The oversampling of ADC's have become very popular due to noise reduction. The objective in this design is of implementing a 4 times oversampling Finite Impulse Response (FIR) digital filter for 8 bit ADC. The FIR filter is normally characterized by a finite difference relation

$$Y_k = C_O X_k + C_1 X_{k-1} + \ldots\ldots + C_m X_{k-m}$$

where X_k is the present input and X_{k-m} is the m^{th} previous input. The order of this filter is (m + 1).

For the filter to be designed, following are the specifications:

Order = 13

Passband ripple = 0.2 dB

Stopband attenuation = 0.2 = -13.98 dB

Processing Speed = 176.4 kHz

i.e. total processing time < 5.67 μs

For the 13th order filter, the output Y expression is

$$Y = h_0 X_n + h_1 X_{n-1} + h_2 X_{n-2} + \ldots\ldots + h_{12} X_{n-12}$$

where h_0 …… h_{12} coefficients are selected to meet the stop band attenuation and pass band ripple.

A simple direct implementation of FIR filter is shown in the figure. The delay elements z^{-1} are master-slave latches holding the filter states. ∇ are the multipliers which multiply latch outputs with the respective coefficients h's which are stored in ROM. The products are then added by 13 input adder and stored in the output latch. This is, however, a very general implementation and a suitable implementation is to be found for a specific problem.

2.2.4 IMPLEMENTATION OF FIR FILTER

A direct implementation of FIR filter is shown in figure 6.

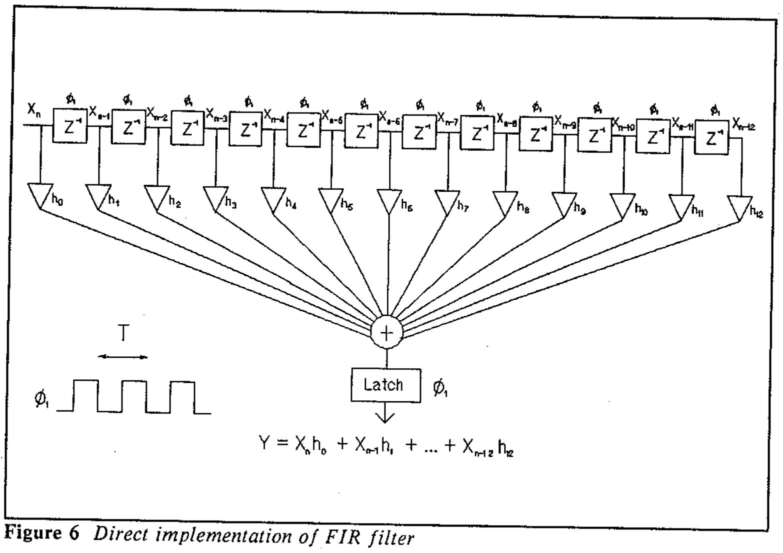

Figure 6 *Direct implementation of FIR filter*

The structure consists of only latches, multipliers and an adder. Latches are arranged in master-slave configuration to hold the filter states. The multiplier coefficients can be stored in roms. Due to the master - slave configuration of the shift registers, data will be latched at the falling edge of the clock ϕ_1. During the cycle of ϕ_1, the output from the shift registers will be multiplied simultaneously with their respective multiplication coefficients. The results are summed in an n-input adder for an n^{th} order filter.

For this design, a digital filter designing software package was used to calculate the coefficients h_0, h_1, …… h_{12} whose values are as shown.

$$H(0) = h_0 = \frac{7}{512} = 2^{-2}\left(2^{-4} - 2^{-7}\right) = h_{12} = H(12)$$

$$H(1) = h_1 = -\frac{48}{512} = 2^{-2}\left(-2^{-2} - 2^{-3}\right) = h_{11} = H(11)$$

$$H(2) = h_2 = -\frac{10}{512} = 2^{-2}\left(-2^{-4} - 2^{-6}\right) = h_{10} = H(10)$$

$$H(3) = h_3 = \frac{32}{512} = 2^{-2}\left(2^{-3} - 2^{-3}\right) = h_9 = H(9)$$

$$H(4) = h_4 = \frac{80}{512} = 2^{-2}\left(2^{-1} - 2^{-3}\right) = h_8 = H(8)$$

$$H(5) = h_5 = \frac{126}{512} = 2^{-2}\left(2^{0} - 2^{-6}\right) = h_7 = H(7)$$

$$H(6) = h_6 = \frac{132}{512} = 2^{-2}\left(2^{0} + 2^{-5}\right)$$

After these coefficients are known, it is easy to simplify the implementation of the figure on the last page. Since two coefficients are always equal, X_{n-k} can be added X_{n-12+k} before the multiplication reducing the number of multiplication by a factor of 2. Still, the previous implementation needs 7 multipliers which need a large area. Since all the coefficients can be expressed as a sum/difference of powers of 2, they can be implemented as barrel shifters. Hence, we will need 7 barrel shifters and 13 input adder.

Since we have been given a very liberal timing target of 5.67 μs, one can only use one shifter and accumulate the sum. This is because a typical operation takes time around 10ns-30ns. If we can achieve the last latch transfer in 20 – 30ns, there will be a lot of time available for shift operation. Since we have plenty of time, the 7 shifting operations can even take 100ns to 200ns each without any problem. This is quite feasible and leads to the implementation shown on the next page. The clocking scheme is shown on the following page.

3.4 Final Implementation

The final implementation of the digital filter is as shown in figure 8.

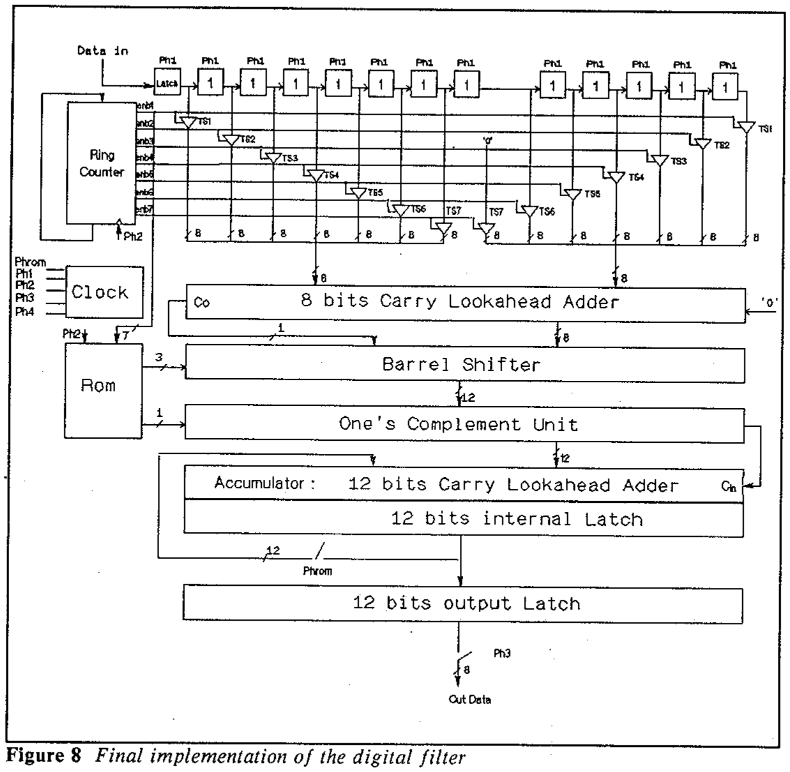

Figure 8 *Final implementation of the digital filter*

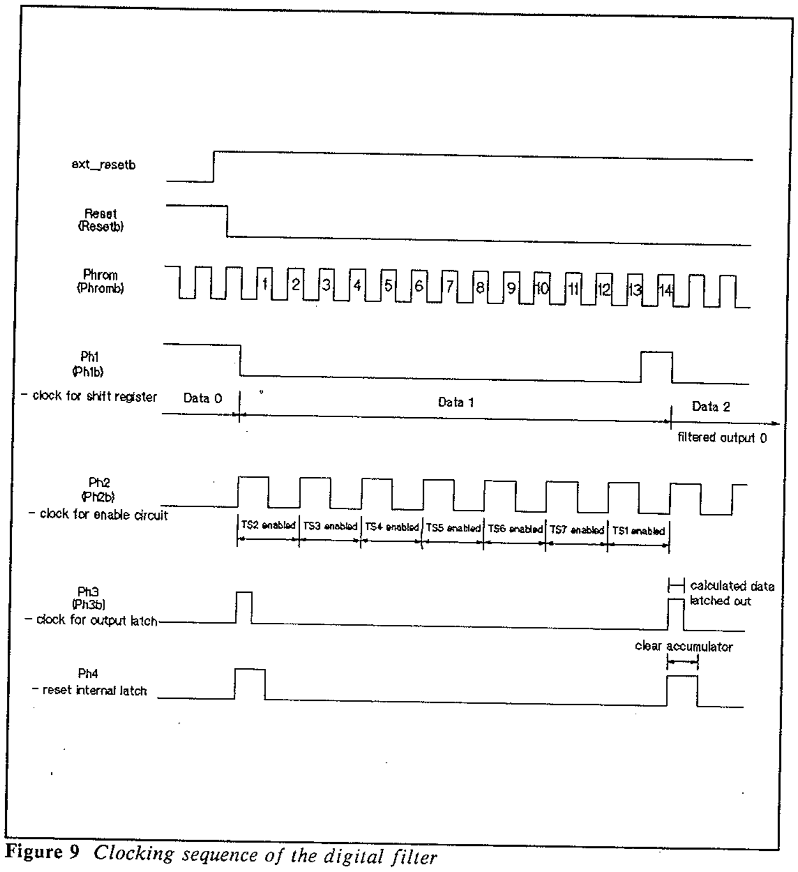

Figure 9 *Clocking sequence of the digital filter*

The inputs to the filter are:

(i) Ext_resetb to reset and synchronize with the ADC operation.

(ii) 8 bit inputs as the data.

(iii) Ext_clk to clock the entire circuit.

Frequency of this clock is 14 times the data rate i.e. 14×176.4 kHz. This generates all the internal clock signals e.g. Phrom, Phromb, ph1, ph1b, etc.

The output of the filter is 12 bit data in 2's complement form. This is needed as some coefficients are negative. The circuit operation can be explained as follows:

1. Immediately after reset, data is latched into the shift register which implements the delay element labeled as $\boxed{1}$.

   The data will be stored for a period of ph1 after which new data is entered shifting the other data.

2. Ring-counter enables the tri-stated gates in pairs which moves inputs to be added to the 8 bit carry look ahead adder. Since data has just entered TS1, it might not be stable yet. For this reason, TS2 ……. TS7 are

enabled first and then TS1. Hence the two 8 bit numbers are added and moved to the input of the barrel shifter. The bit shift information is now obtained from the ROM which is signaled appropriately by enb signal to release appropriate shift code to the barrel shifter.

3. The outputs of the TS's are added by the 8 bit CLA adder. The result is held for 2 periods of Phrom so that the barrel shifter data is stable.
4. During the 2 clock periods of Phrom, 2 shift operations take place each taking one period of Phrom. One of the shift operation is the first term and second the second term in the coefficients in the array H. The 2 bit shift is hard wired and the remaining shift operations are decided by (encoded by) CTRL1, CTRL2, CTRL3 binary numbers from the ROM. 3 bits are enough as the largest multipliers is 2^{-7}.
5. Since some of the coefficients are negative, 2's complement number system is used. After each shift operation, the result is complemented if the multiplier coefficient is negative. 2's complement representation

is 1’s complement +1 which is easily implemented by a 1’s complement unit going to an adder whose $C_{in} = 1$ i.e. LSB is 1 which tells the unit if the multiplier coefficient is negative.

6. The result is then accumulated in 12 bit adder to preserve accuracy. The adder has as built-in 12 bit latch.
7. The shift operation and the addition are carried out twice for a pair of gates as there are two multipliers coefficients.
8. Now in step 2 a new pair of tri-state gates is enabled and whole process 3 to 7 is repeated for all the 6 pairs. During the last barrel shift operation, the output of the accumulator is directly transferred to the output 12 bit latch by clock Ph3. The internal latch of the accumulator is now reset by Ph4 and one is ready for new set of data.

We have already learnt how to implement many of the modules. Once these blocks and timing requirements are know, the full system can be easily synthesized with the help of library of cells. If the system is more complicated, automated SYNPSYS based synthesis can be done as you have learnt with the help of HDLs.

This brings us to the conclusion of this course.