Integrated Device Technology, Inc.

IMPLEMENTATION OF DIGITAL FILTERS USING IDT7320, IDT7210, IDT7216, AND IDT7383

APPLICATION NOTE AN–32

By Tao Lin and Dahn Le Ngoc

INTRODUCTION

Traditionally, signal processing tasks were performed with specialized analog processors. However, it is well known that digital techniques have some inherent advantages such as flexibility, accuracy, reliability over analog techniques. Moreover, because of the rapid progress in digital computer and VLSI technology, both digital processing units and storage devices are becoming less expensive year by year. Therefore, the digital approach is usually preferred over modern signal processing.

Digital filtering is one of the most important digital signal processing techniques. This technique has found many applications in a variety of areas. Perhaps the most widely known applications of digital filtering have been in the area of speech processing and communication. In many situations, speech signals are degraded in ways that limit their effectiveness for communication. In such cases digital filtering techniques are applied to improve speech quality (to remove noise or echoes from speech, etc.). Lowpass and bandpass digital filters have also been utilized in speech analysis and synthesis, speech coding, and data compression. Digital filtering techniques have also been widely used in the area of image processing: enhancement of the image to make it more acceptable to the human eye; removal of the effects of some degradation mechanism; separation of features for easier identification or measurement by human or machine. For example, we can use two-dimensional digital filters to reduce spatial low-frequency components in an X-ray image, and this process will make features with large high-frequency components such as fracture lines easier to identify.

BASIC THEORY OF DIGITAL FILTERS

General Form

A digital filter is a system or device which transforms an input sequence {μ(k)} into an output sequence {y(k)}. As shown in Figure 1, a digital filter is characterized by its impulse response {h(k)} or by its transfer function H(z).

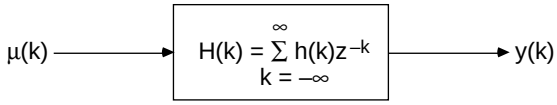


Figure 1. Block Diagram of Digital Filter

The output sequence can be calculated from the input sequence as follows:

$$y(n) = \sum_{k=-\infty}^{\infty} h(k)\,\mu(n-k)$$

A digital filter is said to be causal or realizable if the output at $n = n_0$ is dependent only on values of the input for $n \le n_0$. This implies that the impulse response h(n) is zero for n < 0. The most important subset of the class of causal digital filters is that where the transfer function H(z) can be described by an Nth-order rational function

$$H(z) = \sum_{k=0}^{\infty} h(k)z^{-k} = \frac{a_0 + a_1 z^{-1} + \ldots + a_N z^{-N}}{1 - b_1 z^{-1} - \ldots - b_N z^{-N}} \qquad (2\text{–}1)$$

FIR FILTERS

A digital filter is said to be a finite impulse response (FIR) filter if the number of nonzero h(k) is finite. Otherwise, it is said to be an infinite impulse response (IIR) filter. It can be readily seen that for FIR filters, the denominator of H(z) is 1, i.e. the transfer function becomes

$$\begin{aligned} H(z) &= h(0) + h(1)z^{-1} + h(2)z^{-2} + ... + h(N)z^{-N} \\ &= a_0 + a_1 z^{-1} + a_2 z^{-2} + ... + a_N z^{-N} \end{aligned} \qquad (2\text{–}2)$$

The output of an Nth-order FIR filter can be calculated from N+1 input data as follows:

$$y(k) = a_0\mu(k) + a_1\mu(k-1) + a_2\mu(k-2) + ... + a_N\mu(k-N), \quad \text{for } k = 0, 1, 2, ... \qquad (2\text{–}3)$$

with initial conditions: $\mu(-1) = \mu(-2) = ... = \mu(-N) = 0$. Therefore, FIR filters are nonrecursive and can be implemented by using adders, multipliers and delay elements without a feedback path. The canonical form of an FIR filter is illustrated in Figure 2.

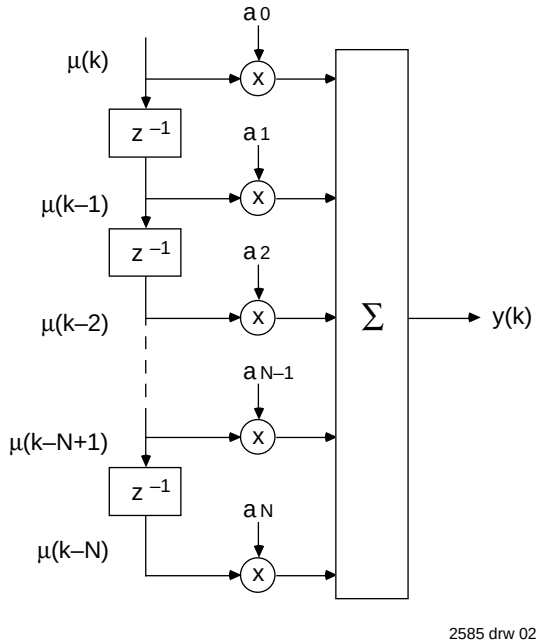


Figure 2. Block Diagram of Canonical FIR Filter

IIR Filters

On the other hand, IIR filters are recursive, i.e., the output of an IIR filters is calculated from both input data and previous output data as follows:

$$y(k) = b_1y(k–1) + b_2y(k–2) + ... + b_Ny(k–N) + a_0\mu(k) + a_1\mu(k–1) + a_2\mu(k–2) + ... + a_N\mu(k–N), \text{ for } k = 0, 1, 2, ... \quad (2–4)$$

with initial conditions: $\mu(–1) = \mu(–2) = ... = \mu(–N) = 0$. The canonical form of IIR filters is shown in Figure 3.

While FIR filters have the advantages of being unconditionally stable, less sensitive to quantization error and linear phase, IIR filters have lower order than FIR filters with equivalent performance. Therefore, IIR filters require less memory and fewer arithmetic operations than FIR filters.

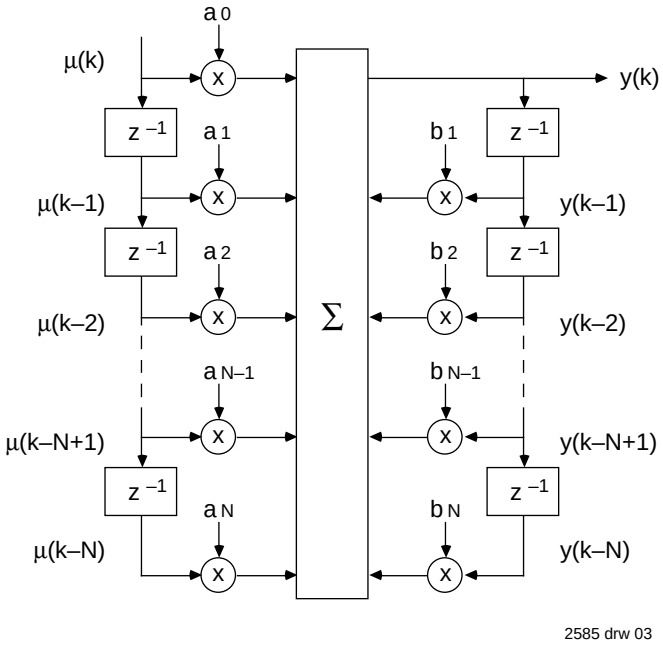


Figure 3. Block Diagram of Canonical IIR Filter

In this application note, we will discuss various implementations of both FIR and IIR filters using the IDT 16–bit DSP building block family: IDT7320, 16–bit 8–level pipeline register; IDT7210, 16x16–bit multiplier–accumulator; IDT7216 16x16–bit multiplier and IDT7383, 16–bit ALU.

IMPLEMENTATIONS OF FIR FILTERS

There are many FIR filter structures. Two particular structures are universally utilized: the transversal structure and the lattice structure.

Transversal Structure

The transversal structure is a rather direct realization of the equation (2–3) in terms of delays, multiplications, and additions. As shown in Figure 4 for a 7th–order (8-tap) filter, the output y(k) of the transversal structure is simply the weighted sum of the current input μ(k) and the delayed inputs μ(k–1), μ(k–2). The coefficient a_0, a_1, ... determine the frequency response of a particular filter such as lowpass, bandpass or highpass.

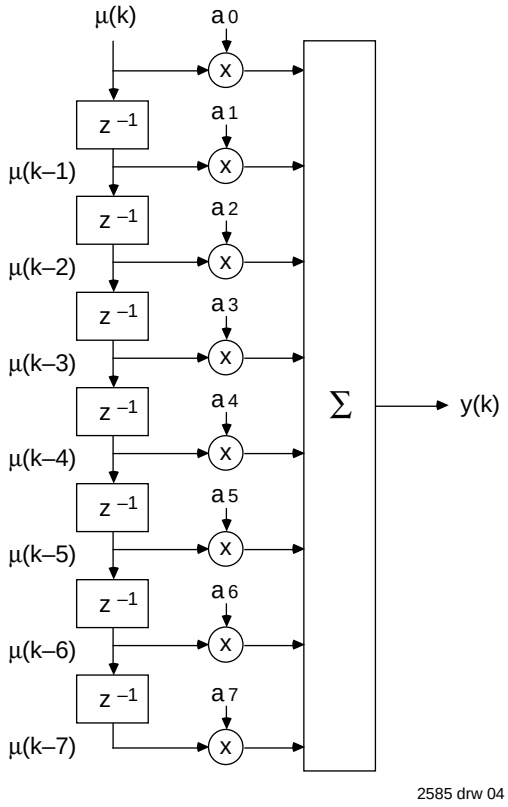


Figure 4. 8–Tap Transversal Structure

Figure 5 illustrates the implementation of the 8-tap transversal structure using two IDT7320s and an IDT7210. One IDT7320 is for the storage of input data, and the other for the storage of filter coefficients. The IDT7210 is used to perform multiplication and accumulation. Registers REG G_1 and REG H_2 of the IDT7320s are connected to X and Y input registers of the IDT7210, respectively. Both IDT7320s are shifted every clock cycle. Thus, the input data and the coefficients are loaded into the input registers of the IDT7210 in the sequence shown in Figure 6. In the first clock cycle, a new input word is loaded into REG A_1 and the pipeline registers A_1–G_1 shift down. In the next seven clock cycles, the output of REG G_1 is connected to the input of REG A_1, so that the pipeline registers A_1–G_1 shift as a ring every clock cycle. Similarly, the output of REG H_2 is connected to the input of REG A_2. A new output is unloaded every eight clock cycles. A sequence controller generates the clock and the control signals SEL and ACC. The filter coefficients are preloaded into the pipeline registers A_2–H_2. Generally, for an N-tap transversal structure, a filter cycle has N clock cycles. Therefore, a filter cycle time is NSt_{MA}, where t_{MA} = 25ns is the multiply–accumulate time of IDT7210.

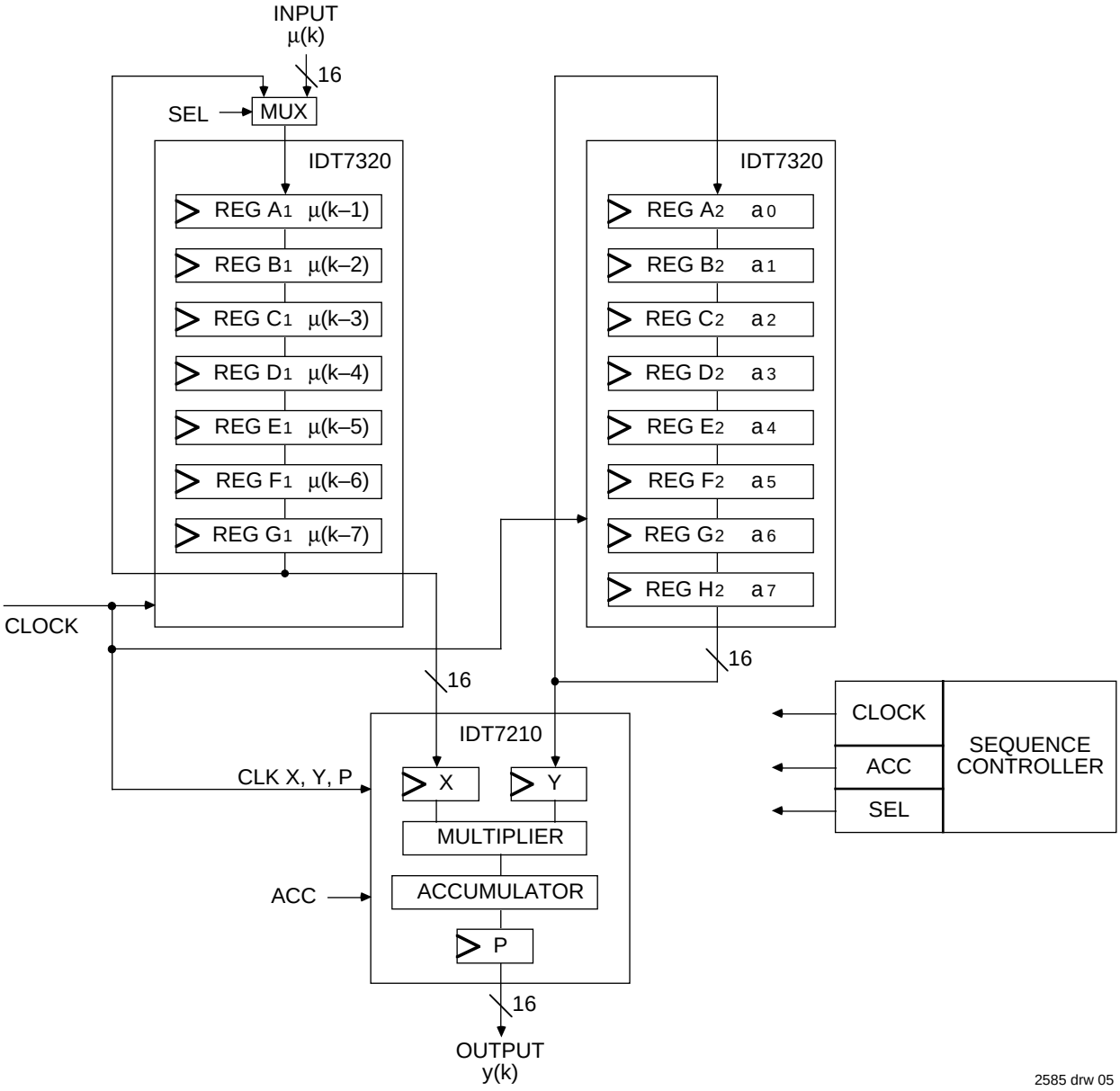
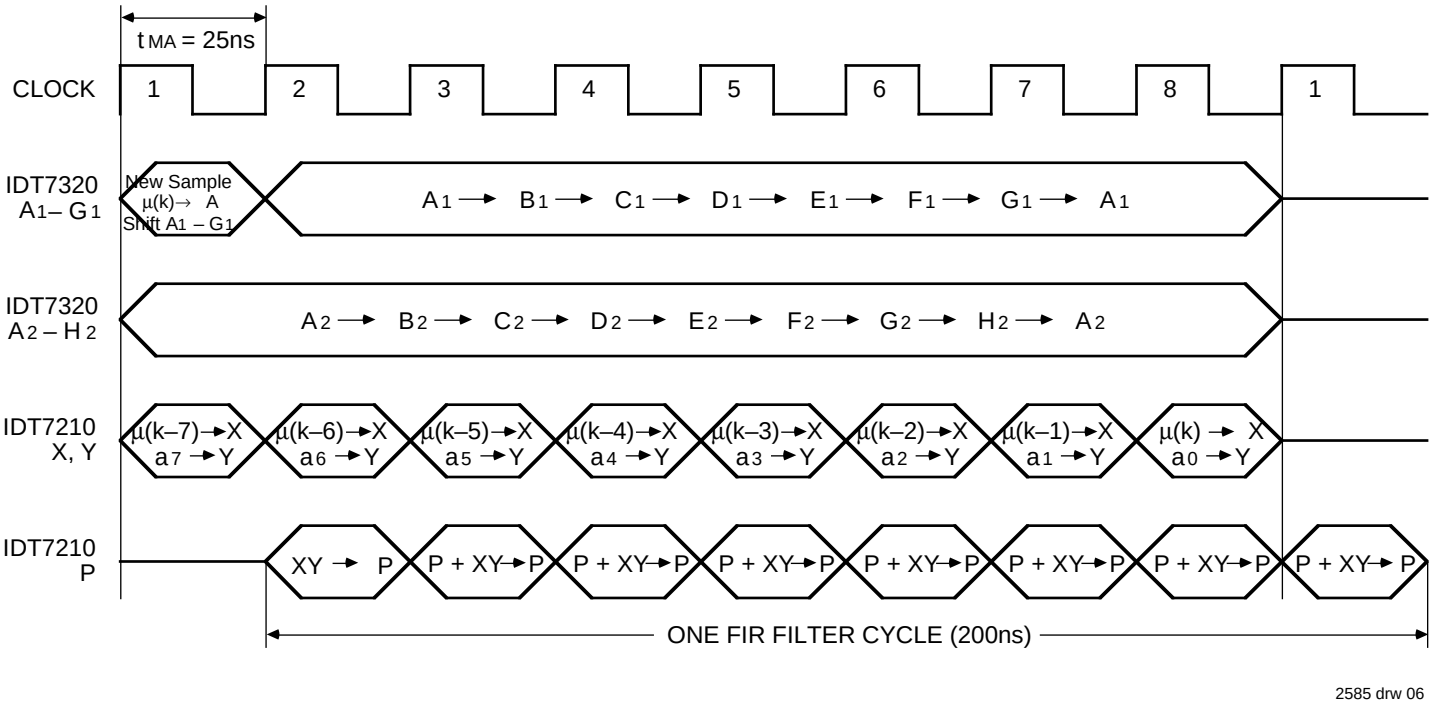


Figure 5. Implementation of the Transversal Structure Using IDT7320s and IDT7210



2585 drw 06

Figure 6. Operation Sequence of IDT7320s and IDT7210 for the Transversal Structure

Lattice Structure

The lattice structure of an Nth-order FIR filter shown in Figure 7 for n = 8. The lattice structure is equivalent to the transversal structure, in the sense that any transfer function which can be represented by the transversal structure can also be represented within a multiplicative constant by the lattice structure. The origin and utility of the structure is that it has several advantages over the transversal structure in the field of adaptive filtering.

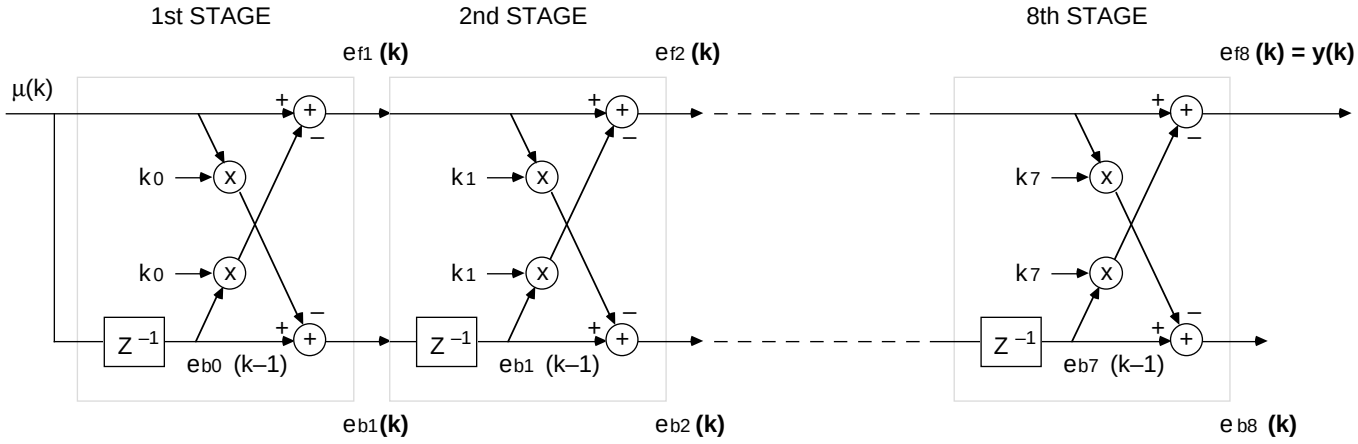


Figure 7. Block Diagram of the Lattice Structure

From Figure 7, we can see that an Nth-order lattice structure consists of N stages, each having two inputs and two outputs. The outputs $\{e_{fm}(k), k = 0, 1, 2, ...\}$ and $\{e_{bm}(k), k = 0, 1, 2, ...\}$ of the mth (1mN) stage are called mth-order forward and backward prediction errors, respectively, which are related to the inputs of the stage (the outputs of the previous stage) as follows:

$$e_{fm}(k) = e_{f(m-1)}(k) - k_{m-1}\, e_{b(m-1)}(k-1) \quad (3\text{–}1a)$$
$$e_{bm}(k) = e_{b(m-1)}(k-1) - k_{m-1}\, e_{f(m-1)}(k) \quad (3\text{–}1b)$$

where the input of the first stage is

$$e_{f0}(k) = e_{b0}(k) = \mu(k) \quad (3\text{–}1c)$$

Equation (3–1) shows that we need to store $\{e_{b0}(k-1), e_{b1}(k-1), ..., e_{b(N-1)}(k-1)\}$, the backward prediction errors at time k–1, for calculating the outputs of all stages at time k: $\{e_{b1}(k), e_{f1}(k), e_{b2}(k), e_{f2}(k), ..., e_{bN}(k), e_{fN}(k)\}$. An implementation of the 8-stage lattice structure is given in Figure 8 using IDT7320s, 7216s, and 7383s. Two IDT7320s store the previous outputs $e_{b0}(k-1), e_{b1}(k-1), ..., e_{b7}(k-1)$ and the coefficients $k_0, k_1, ..., k_7$. The multiplications of

$$k_{m-1}\, e_{b(m-1)}(k-1) \text{ and } k_{m-1}\, e_{f(m-1)}(k)$$

are performed by two IDT7216s. Two IDT7383s execute the subtractions

$$e_{f(m-1)}(k) - k_{m-1}\, e_{b(m-1)}(k-1) \text{ and } e_{b(m-1)}(k-1) - k_{m-1} e_{f(m-1)}(k).$$

The sequence of the operations is shown in Figure 9. A filter cycle consists of 8 clock cycles. In the first clock cycle, $e_{b0}(k-1) = \mu(k-1)$ stored in REG H_1 is loaded into registers X_1 and R_1; k_0 stored in REG H_2 is loaded into registers Y_1 and Y_2; and the new input $\mu(k) = e_{f0}(k)$ is loaded into registers X_2 and R_2. The new input $\mu(k) = e_{b0}(k)$ is also loaded into REG H_1. After a time delay of t_{MUC} = 30ns, the results of multiplication appear at the output pins of the IDT7216s which are directly connected to the input of the ALU of the IDT7383s. Then, after another time delay of t_{ALU} = 25ns, we obtain the outputs of the first stage, $e_{b1}(k)$ and $e_{f1}(k)$, at the output pins of the IDT7383s. In the second clock cycle, $e_{b1}(k-1)$ stored in REG G_1, k_1 stored in REG G_2, and $e_{f1}(k)$ appeared at the output port of the IDT7383 are loaded into corresponding input registers of the IDT7216s and 7383s. At the same time, $e_{b1}(k)$ is loaded into REG G_1. After a time delay of $t_{MUC} + t_{ALU}$ = 55ns, we obtain $e_{b2}(k)$ and $e_{f2}(k)$, at the output pins of the IDT7383s, and so on. Finally, in the eighth cycle, we obtain $e_{b8}(k)$ and $y(k) = e_{f8}(k)$. It should be noted that in each clock cycle, the IDT7216s first perform the multiplication, then the IDT7383s complete the subtraction. Therefore, the time of a clock cycle is $t_C = t_{MUC} + t_{ALU}$ = 55ns. For an Nth-order lattice FIR filter, the filter cycle time is 55 x N nanoseconds (440ns for N = 8).

The signals $I_{0–3}$ control to which register of the IDT7320 a new backward prediction error will be written. The signals $SEL_{0–2}$ select one of the eight registers of the IDT7320s to be read from the output port. A sequence controller is needed to generate the clock and the control signals $I_{0–3}$ and $SEL_{0–3}$. The filter coefficients k_0–k_7 are preloaded into the registers A_2–H_2.

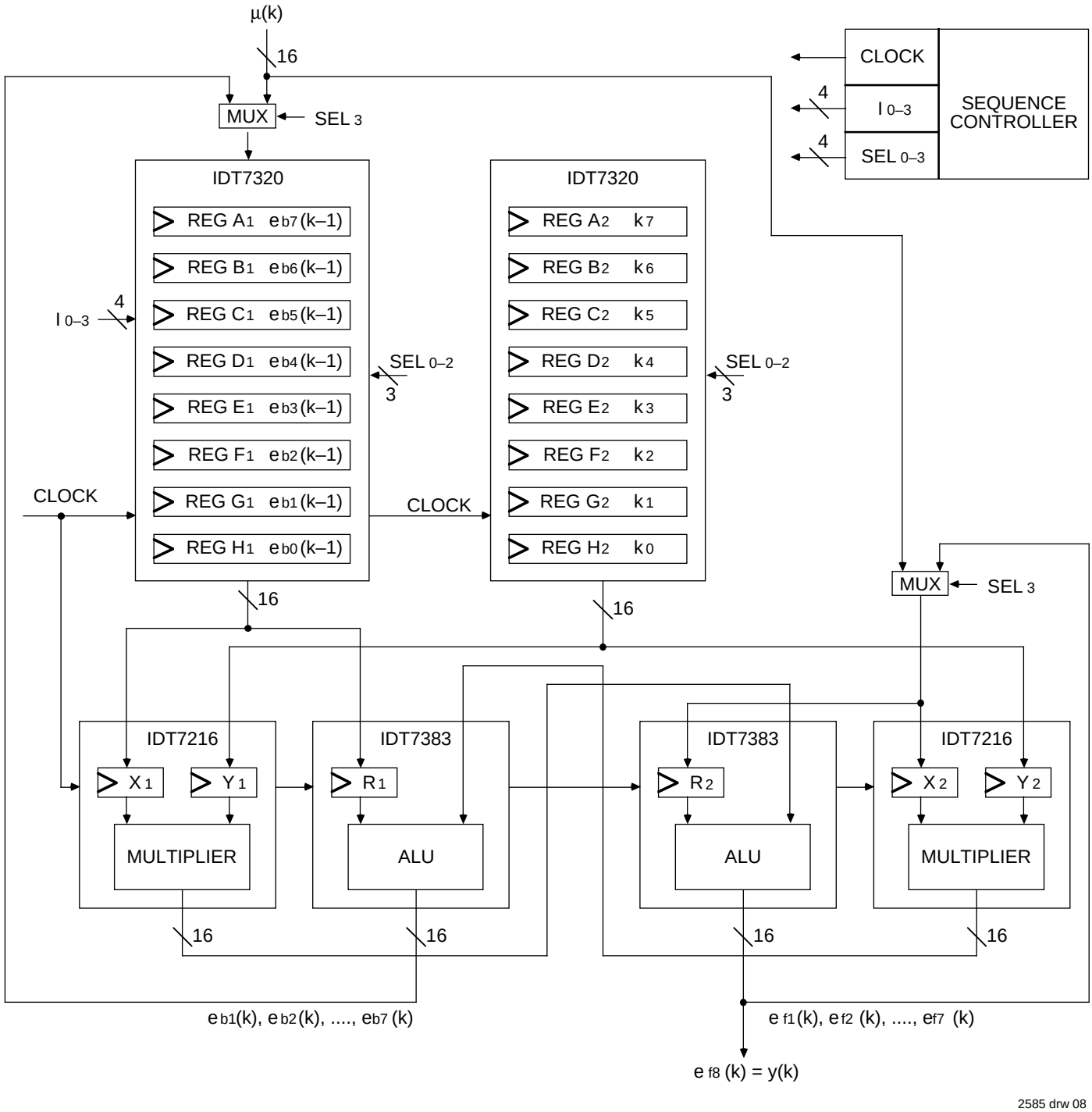


Figure 8. Implementation of the Lattice Structure Using IDT7320s, IDT7216s and IDT7383s

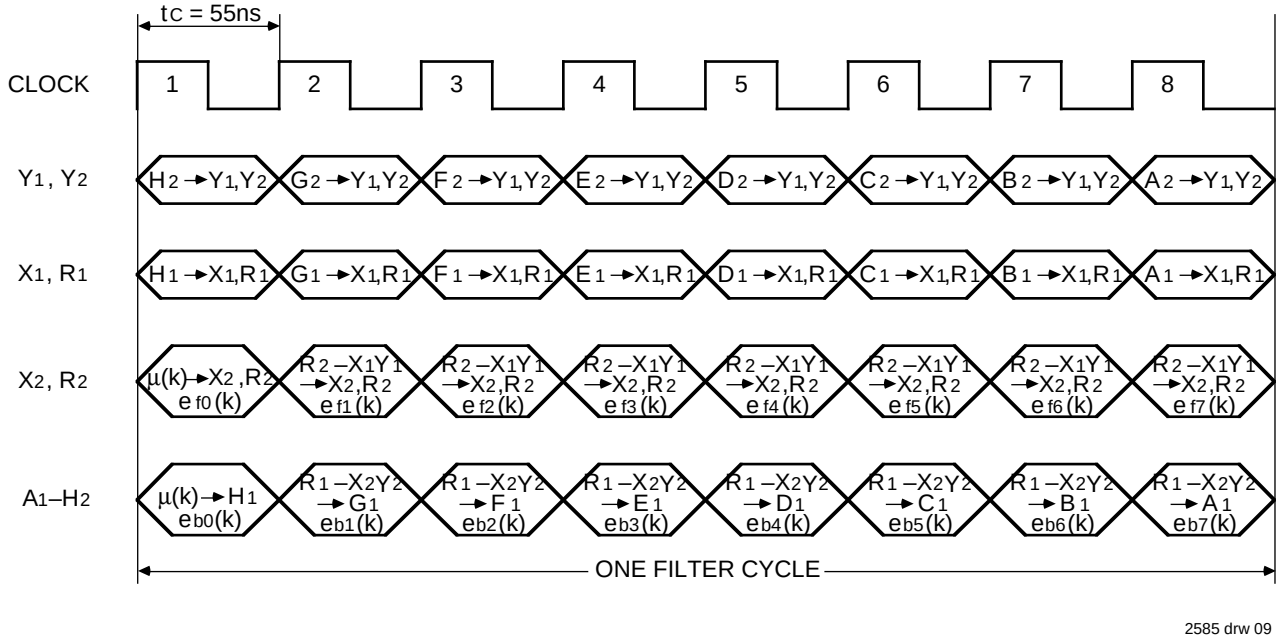


Figure 9. Operation Sequence of IDT7320s, IDT7216s and IDT7383s for the Lattice Structure

IMPLEMENTATIONS OF IIR FILTERS

Since IIR filters have feedback elements, architecture for implementing IIR filters are more complex than those for filters. Moreover, roundoff errors of multiplication may accumulate and be amplified through the feedback loop so that the roundoff noise at the filter output becomes a serious problem. However, IDT's flexible and high-precision DSP product lines provide unique solution for implementing IIR filters.

There are a variety of structures to implement IIR filters, such as direct form structure, cascade structure, parallel structure, lattice structure, ladder structure, state-space structure. Among these, direct form, parallel and cascade structures are popular in many applications. In the following, we will consider how to implement these filter structures using the IDT7320, 7210, and 7383.

Direct Form Structure

The direct form structure is the simplest implementation of IIR filters and requires the fewest multiplication, addition and delay elements. This means that it can achieve higher speed and needs less hardware than other structures. The disadvantage of the direct form structure is that it may have multiplication roundoff noise. This can be overcome by using the IDT high-precision 16-bit multiplier–accumulator (MAC), where the whole 32-bit product is preserved and used in the accumulator.

Let U(z) and Y(z) be the z-transforms of the input {μ(k)} and

$$Y(z) = H(z)\,U(z) = \frac{a_0 + a_1 z^{-1} + \ldots + a_N z^{-N}}{1 - b_1 z^{-1} - \ldots - b_N z^{-N}} U(z) = \frac{A(z)}{B(z)}\,U(z). \quad (4\text{–}1)$$

Define $W(z) = \frac{1}{B(z)}\,U(z)$, we obtain

the output {y(k)}, respectively, then an IIR filter is described by

$$W(z)B(z) = U(z) \text{ and } Y(z) = A(z)W(z). \quad (4\text{–}2)$$

Equation (4–2) can be written in the time domain as

$$w(k) = \mu(k) + b_1 w(k-1) + b_2 w(k-2) + \ldots + b_N w(k-N) \quad (4\text{–}3a)$$

and

$$y(k) = a_0 w(k) + a_1 w(k-1) + a_2 w(k-2) + \ldots + a_N w(k-N) \quad (4\text{–}3b)$$

From (4–3), we get the direct form structure of the IIR filter as shown in Figure 10. The direct form structure can be

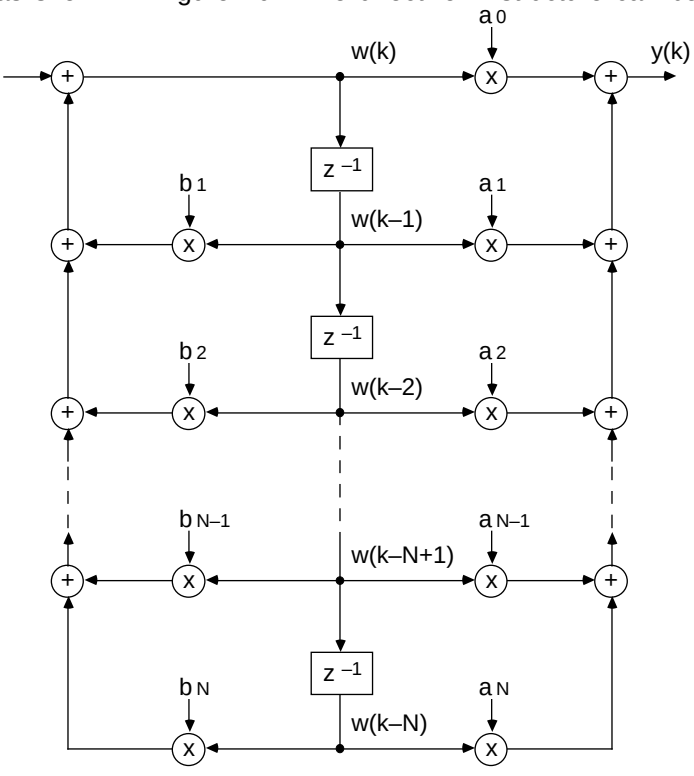


Figure 10. Block Diagram of the Direct Form Structure

implemented using a single MAC or two MACs.

Implementation Using A Single MAC

Using a single MAC, for each new input μ(k), we first calculate w(k) given by (4–3a), and then calculate y(k) by (4–3b). The implementation of a 7th-order filter is shown in Figure 11. Three IDT7320s are used to store {w(k)}, the coefficients {b_1, b_2, ..., b_7, 1} and the coefficients {a_0, a_1, ..., a_6, a_7}. The new input μ(k) and the data {w(k)} stored in the IDT7320 are sent to X input port of the IDT7210 through a multiplexer, while

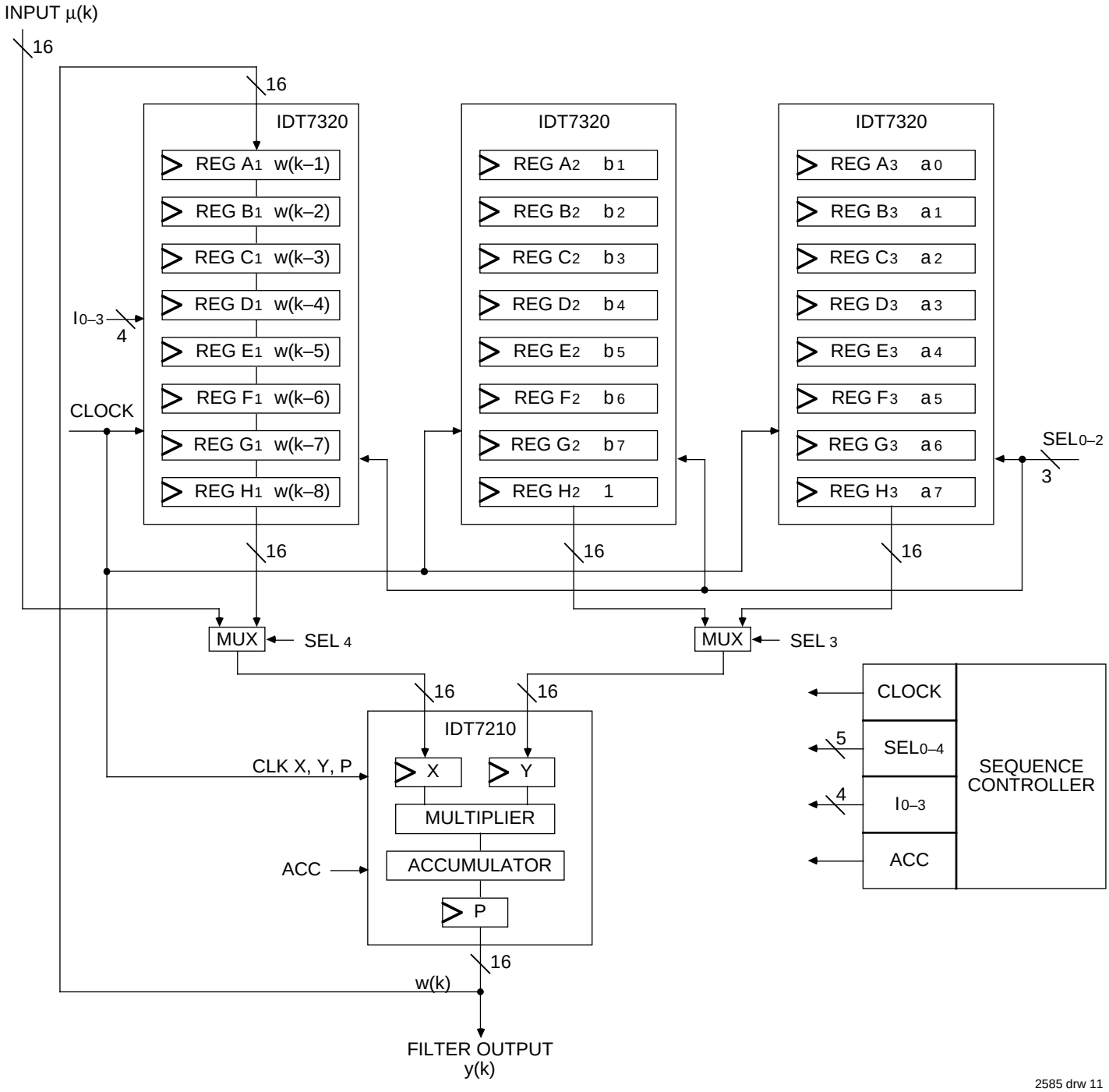
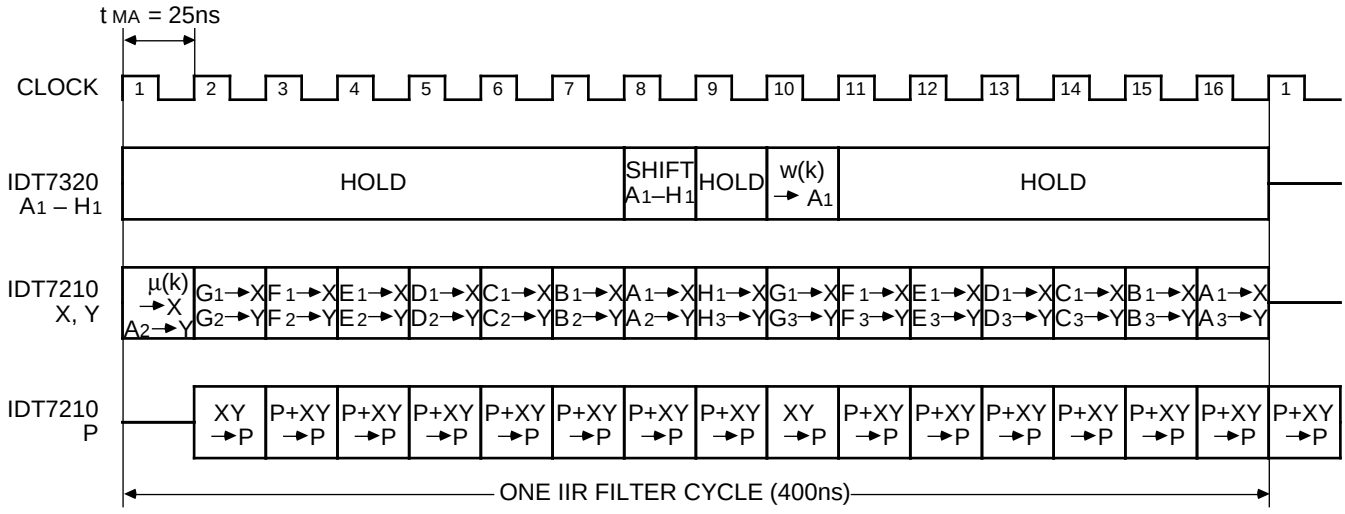


Figure 11. Direct Implementation of IIR Filter Using a Single MAC

the coefficients {a_0, a_1, ..., a_6, a_7} are sent to Y input port of the IDT7210 through another multiplexer.

As shown in Figure 12, each filter cycle consists of 16 clock cycles. The first eight clock cycles calculate w(k) while the last eight clock cycles calculate y(k). In the first clock cycle, the new input μ(k) and the content of REG H_2 are loaded into the input registers of the IDT7210. Since 1 is stored in REG H_2, the result obtained in the output register of the IDT7210 is μ(k). In the second clock cycle, the contents of REG G_1 and REG G_2 are sent to the input registers of the IDT7210, and so on. Then, in the ninth clock cycle, w(k) is obtained in the output register of the IDT7210. In the tenth cycle, we load w(k) into REG A_1 which will be used in the sixteenth cycle. Before w(k) is loaded, in the eighth cycle, we shift down the pipeline registers A_1–H_1, so that in the ninth cycle, the data stored in H_1 is not w(k–8) but w(k–7) which is multiplied by a_7 stored in REG H_3. A new output y(k) is obtained in the first clock cycle of the next filter cycle. It should be noted that in this implementation, we obtain different data at the output ports of the IDT7320s by using the output selection signal $SEL_{0–2}$, not



2585 drw 12

Figure 12. Sequence of Operations of IDT7320s and IDT7210 for the Direct Form Implementation Using a Single MAC

by shifting the pipeline registers every clock cycle.

Implementation Using Two MACs

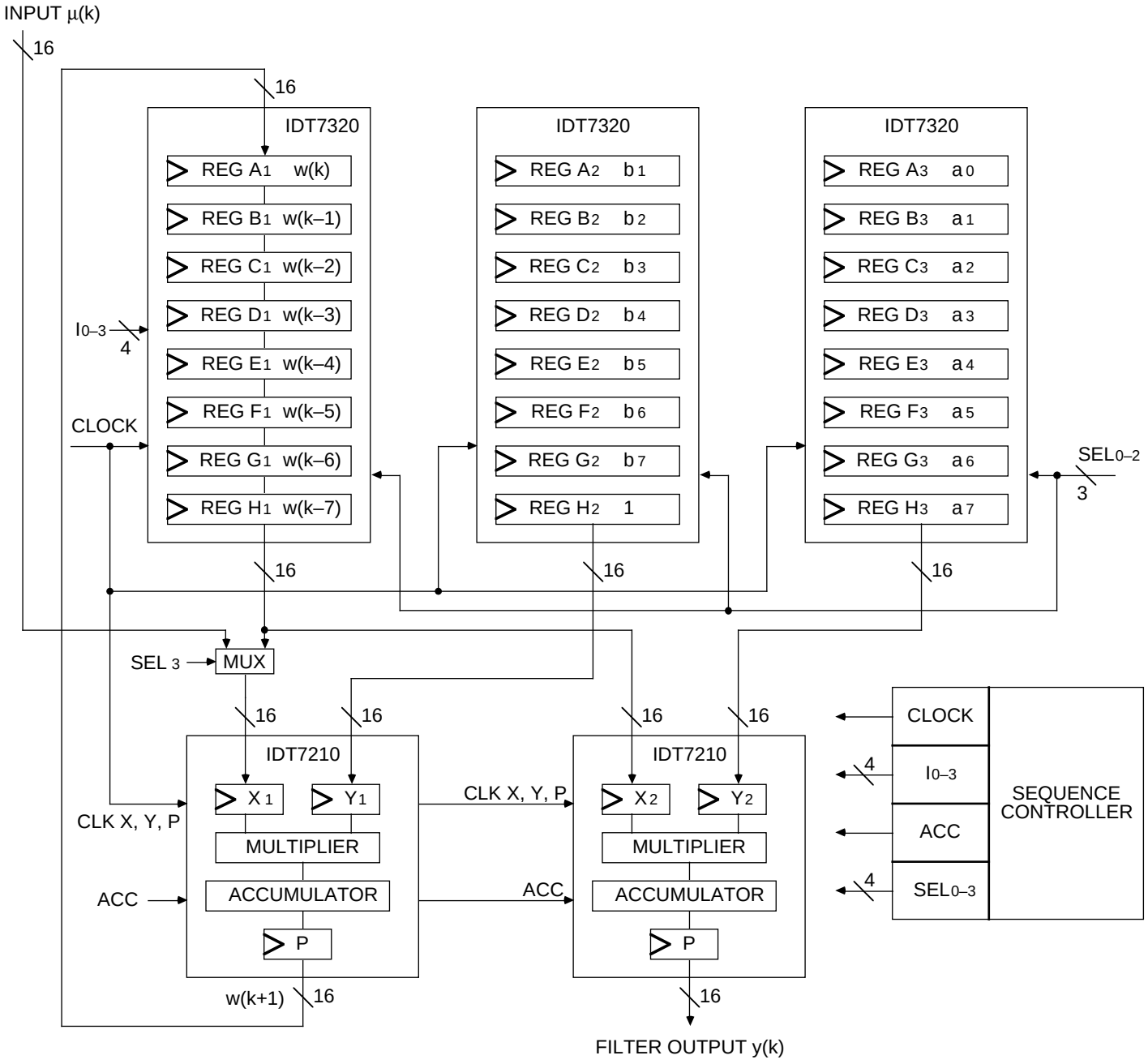
In the implementation mentioned above, we use a single MAC to calculate w(k) and y(k) alternately. If we use two MACs, one MAC for calculating y(k) given by

$$y(k) = a_0w(k)+a_1w(k-1)+a_2w(k-2)+...+a_Nw(k-N) \quad (4-4a)$$

and another MAC for simultaneously calculating w(k+1) given by

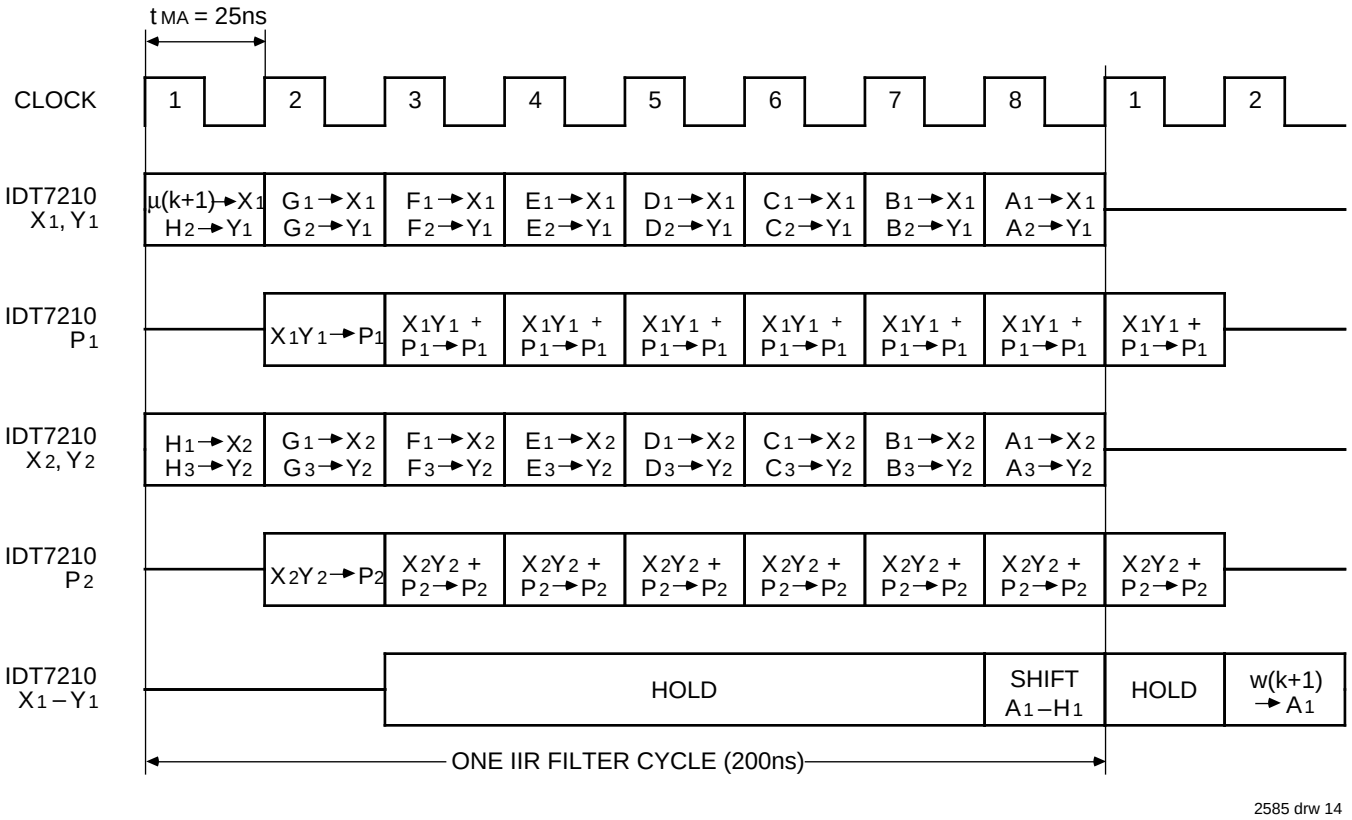
$$w(k+1) = \mu(k+1)+b_1w(k)+b_2w(k-1)+...+b_Nw(k-N+1) \quad (4-4b)$$

then the processing speed can be doubled. The implementation of a 7th-order filter using two IDT7210s is shown in Figure 13. A filter cycle has 8 clock cycles as shown in Figure 14. In the first cycle, μ(k+1) is loaded into register X, and multiplied by the content of REG H_2 which is one, so that the result in the output register P_1 is still μ(k+1). In the next seven cycles, w(k–6), w(k–5), ..., w(k) are loaded into register X_1 through the multiplexer and multiplied by b_7, b_6, ..., b_1, respectively. In the eighth cycle, we shift down the pipeline registers A_1–H_1 to prepare for the next filter cycle. Then, in the first clock cycle of the next filter cycle, we obtain w(k+1) in the output register P_1 which is loaded into REG A_1 in the second clock cycle. When one IDT7210 calculates w(k+1), another IDT7210



2585 drw 13

Figure 13. Direct Form Implementation of IIR Filter Using Two MACs



2585 drw 14

Figure 14. Sequence of Operations of IDT7320s and IDT7210 for the Direct Form Implementation Using Two MACs

calculates y(k) and an output is unloaded from register P_2 every 8 clock cycles.

Whether using a single MAC or two MACs, the signals $I_{0–3}$ control whether or not the pipeline registers shift or hold and to which register of the IDT7320 a new result w(k) will be written. On the other hand, the signals $SEL_{0–2}$ select one of the eight registers of the IDT7320s to be read from the output port. A sequence controller generates the clock and the control signals. The filter coefficients are preloaded into IDT7320s as in the FIR filter case.

Parallel Structure

The parallel structure has the advantage of less multiplication roundoff noise and coefficient quantization sensitivity than the direct form structure. However, the parallel structure uses more hardware. The basic principle of the parallel structure is that an Nth-order rational transfer function

$$H(z) = \frac{a_0 + a_1 z^{-1} + ... + a_N z^{-N}}{1 + b_1 z^{-1} + ... + b_N z^{-N}} \quad (4–5)$$

can be expanded to partial fraction as follow

$$H(z) = H_1(z) + ... + H_M(z) \quad (4–6)$$

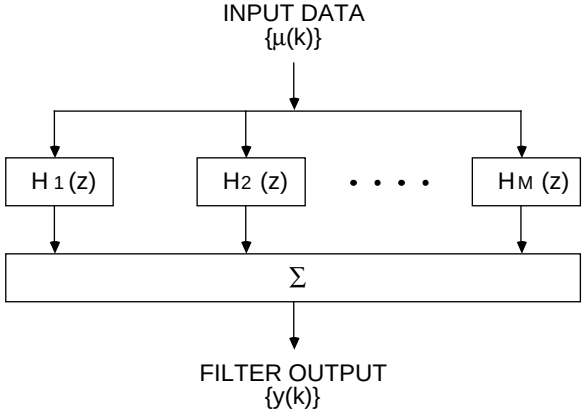
The expansion suggests that the filter could be implemented

$$H_i(z) = \frac{a_0 + a_1 z^{-1}}{1 + b_1 z^{-1}} \quad (4–7)$$

in a parallel structure shown in Figure 15. To minimize the roundoff noise and coefficient sensitivity, $H_i(z)$ is usually a first-order filter or a second-order filter

$$H_i(z) = \frac{a_0 + a_1 z^{-1} + a_2 z^{-2}}{1 + b_1 z^{-1} + b_2 z^{-2}} \quad (4–8)$$

which can be implemented by the direct form structure mentioned before. For example, a fourth-order filter can be implemented with two parallel sections, each being a second-order filter, as shown in Figure 16. In this particular implementation, each section uses two MACs. The outputs of two sections are added by the IDT7383 to obtain the filter



2585 drw 15

Figure 15. Parallel Structure of IIR Filters

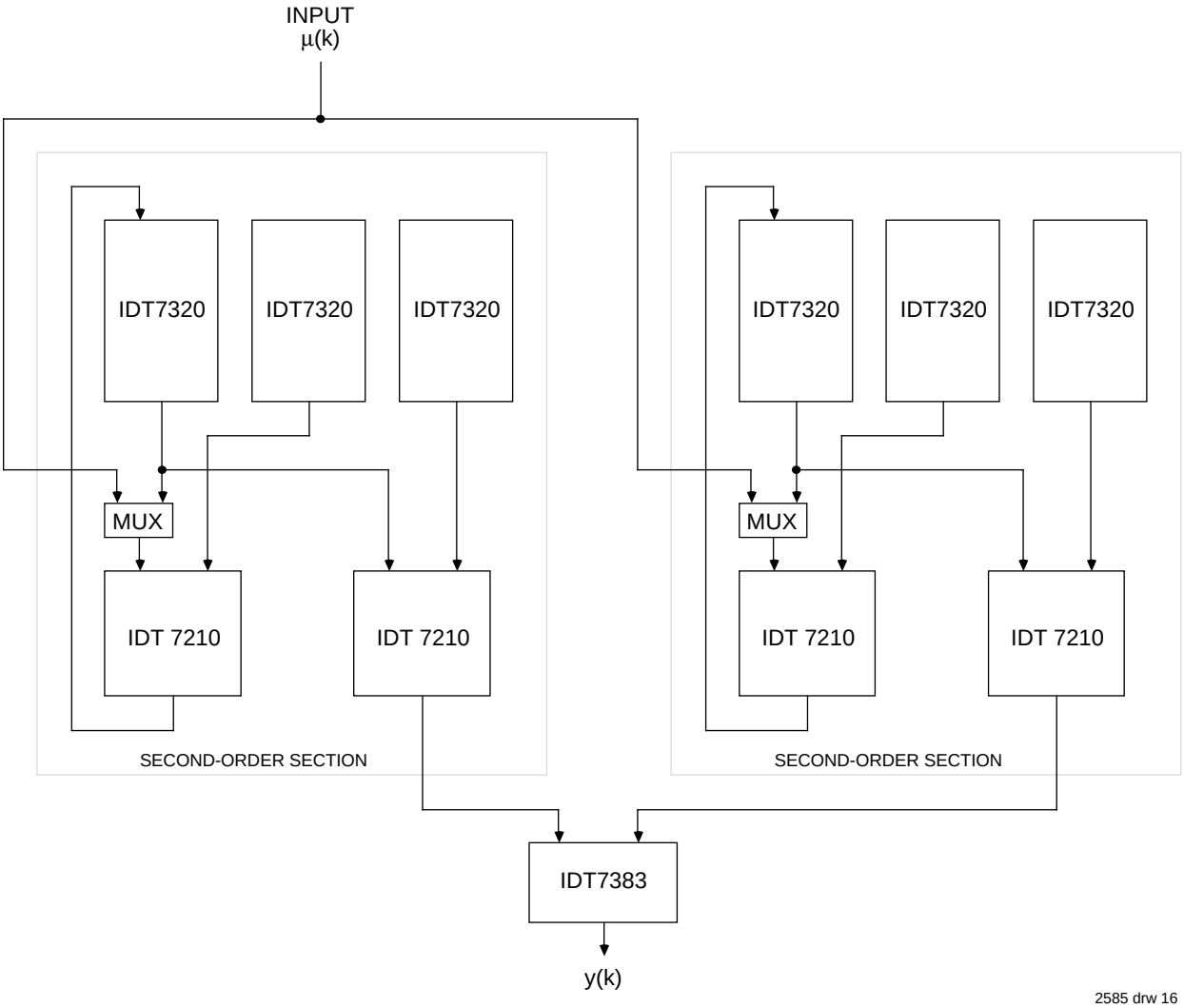


Figure 16. Parallel Implementation of a Fourth–Order Filter Using Two Second–Order Sections

output {y(k)}.

Cascade Structure

Like the parallel structure, the cascade structure has the advantage of less multiplication roundoff noise and coefficient quantization sensitivity and the disadvantage of more hardware than the direct form structure. The basic principle of the cascade structure is to decompose an Nth-order rational transfer function given by (4–5) into first-order or second-order sections as follows:

$$H(z) = H_1(z)S \ldots . SH_M(z) \qquad (4–9)$$

where $H_i(z)$ is given by (4–7) or (4–8). From the decomposition, the filter of (4–5) can be implemented by the direct form structure. For example, a fourth-order filter can be implemented with two cascade sections, each being a second-order filter, as shown in Figure 18. The output of the first section is the input of the second section.

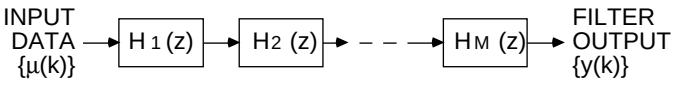


Figure 17. Cascade Structure of IIR Filters

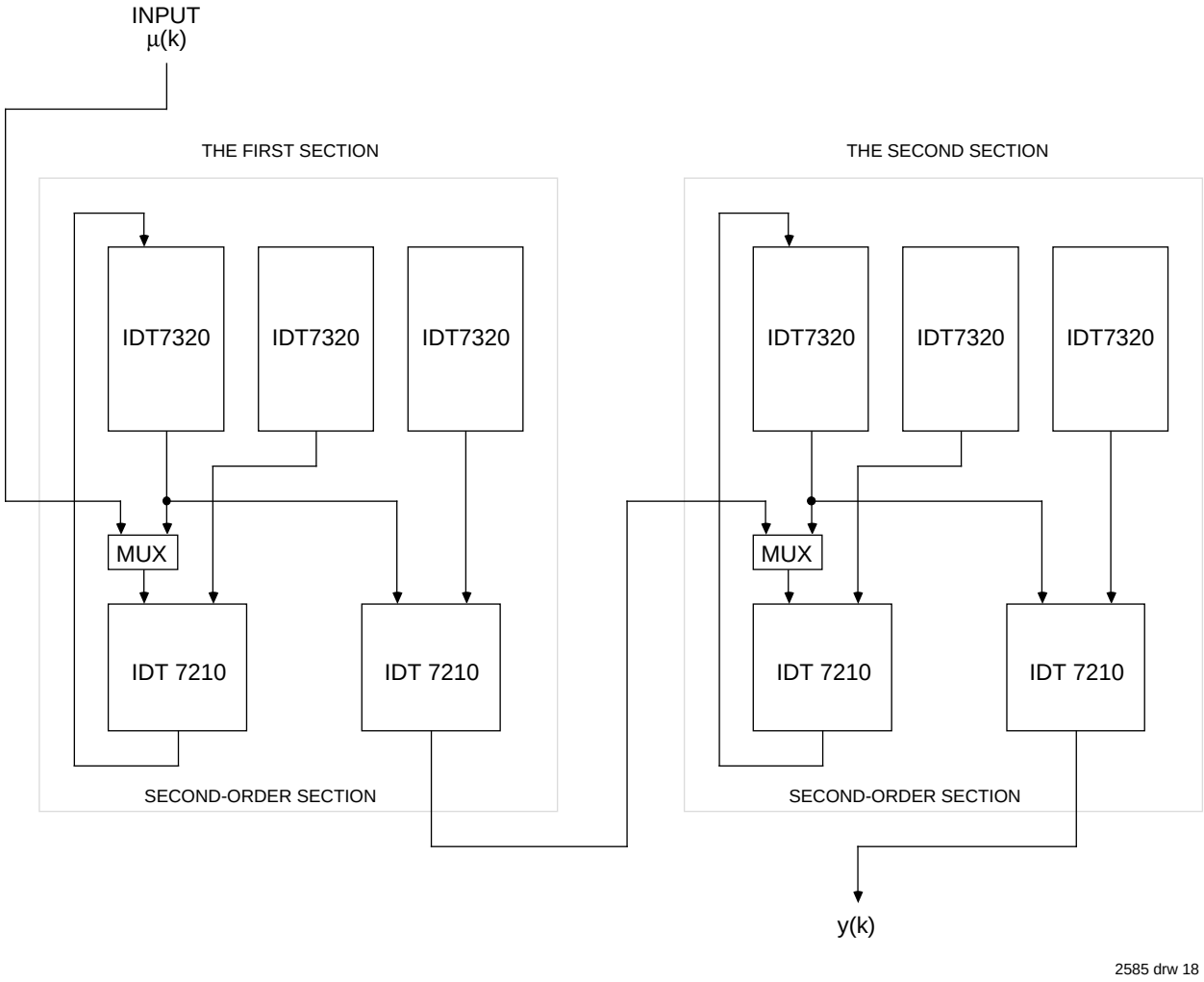


Figure 18. Cascade Implementation of a Fourth–Order Filter Using Two Second–Order Sections

CONCLUSIONS

In this application note, we have discussed the basic methods to implement a variety of structures of both FIR and IIR filters using IDT DSP building block family: pipeline registers, MACs, multipliers, and ALUs. Which structure and implementation should be selected in a particular application is decided by many factors such as the available filter design tools, cost, speed, etc.. In many applications, the FIR transversal structures is used because of the simplicity of filter design and implementation. The FIR lattice structure is employed in the application where the filter coefficients have to be adaptively changed and fast convergency of the coefficients is required. In applications requiring high speed and compact hardware, IIR filters are usually preferred. Different IIR filter structures may have completely different finite wordlength effects (roundoff error, coefficient error and limit cycles). The direct form structure is the simplest one and uses the least hardware. However, if the filter order is large and the bandwidth of the filter is very narrow, then the direct form structure may have severe roundoff noise and limit-cycles so that the actual input-output characteristic of the filter dramatically deviates from the ideal one. In this situation, parallel structure or cascade structure should be utilized.

The building block approach discussed in this application note can achieve 10 times the performance of some simple FIR and IIR filter structures. Table 1 gives a comparison between the building block approach using IDT's DSP building blocks and the single chip DSP approach using the latest Texas Instruments TMS320C25–50, a single chip digital signal processor with an instruction cycle time of 80ns.

IDT's extensive and flexible product lines provide various possibilities to implement different filter structures. In the FIR filter implementations and IIR filter direct form implementations shown in this application note, we have using at most two MACs, multipliers and ALUs. The number of the clock cycles in a filter cycle is proportional to the filter order. If more MACs, multipliers and ALUs are used, a filter cycle can contain only a single clock cycle to achieve the highest speed.