SECOND ORDER RECURSIVE DIGITAL FILTER DESIGN WITH THE TRW MULTIPLIER–ACCUMULATORS

R.J. Karwoski
TRW LSI PRODUCTS
2525 E. El Segundo, Calif. 90245

SECOND ORDER RECURSIVE DIGITAL FILTER DESIGN WITH TRW MULTIPLIER ACCUMULATORS

1. INTRODUCTION

Just as the second order analog filter has played an important role in the synthesis of analog filters, the digital counterpart has gained equal acceptance as a basic building block for complex digital filter development. Analog Butterworth and elliptical filters are synthesized by cascading any number of specified second order sections. Similarly, digital Butterworth and elliptical filters may be realized using the same technique. In addition, a particular implementation of finite impulse response (FIR) filters uses a bank of second order digital filters. It is the object of this paper to develop some of the fundamental concepts of second order recursive digital filters and to describe some very efficient hardware implementations using the TDC1010J multiplier accumulator.

2. MATHEMATICAL PRELIMINARIES*

The general second order discrete filter has the transfer function in equation (1)

$$\frac{Y(Z)}{X(Z)} = C_1 \frac{Z^2 - 2r_o \cos\theta_o Z + r_o^2}{Z^2 - 2r_p \cos\theta_p Z + r_p^2} = H(Z) \qquad (1)$$

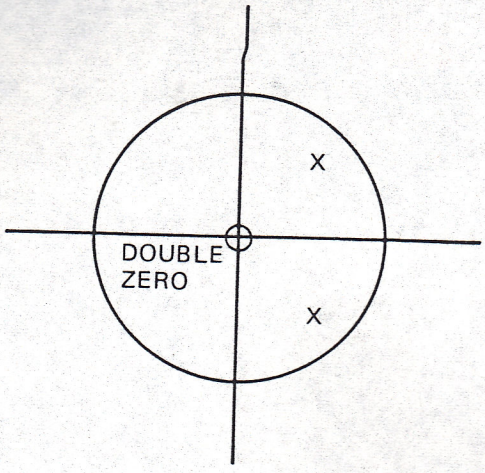
Equivalently equation (2) gives H(Z) in terms of its poles and zeros in the Z plane; equations (1) and (2) are equal.

$$H(Z) = C_1 \frac{(Z - r_o e^{-j\theta_o})(Z - r_o e^{+j\theta_o})}{(Z - r_p e^{-j\theta_p})(Z - r_p e^{+j\theta_p})} \quad C_1 = \text{constant} \qquad (2)$$

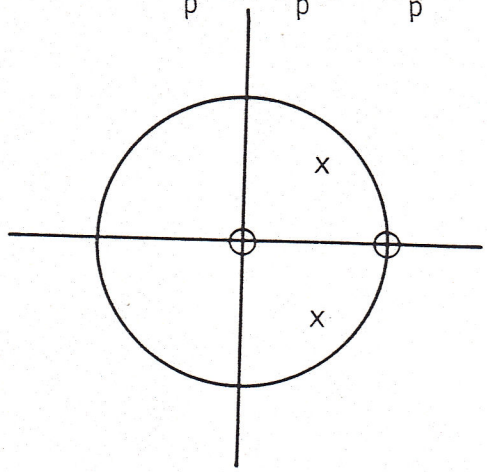
---

*The reader should consult TRW application note TDSP101 on the "Z transform, its derivation and application to digital filters" for a more complete coverage of the mathematics and signal configurations covered in this section.

a) $$H(Z) = \frac{C_1 Z^2}{Z^2 - 2r_p \cos\theta_p Z + r_p^2}$$ *Lowpass*



b) $$H(Z) = C_1 \frac{Z(Z-1)}{Z^2 - 2r_p \cos\theta_p Z + r_p^2}$$ Bandpass



c) $$H(Z) = C_1 \frac{(Z-1)^2}{Z^2 - 2r_p \cos\theta_p Z + r_p^2}$$ High Pass

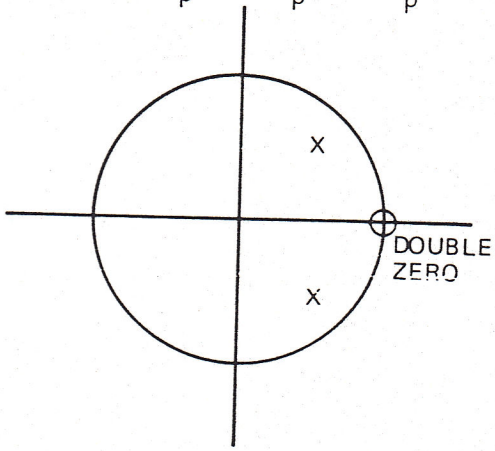


Figure 2. Corresponding Pole Zero Plots

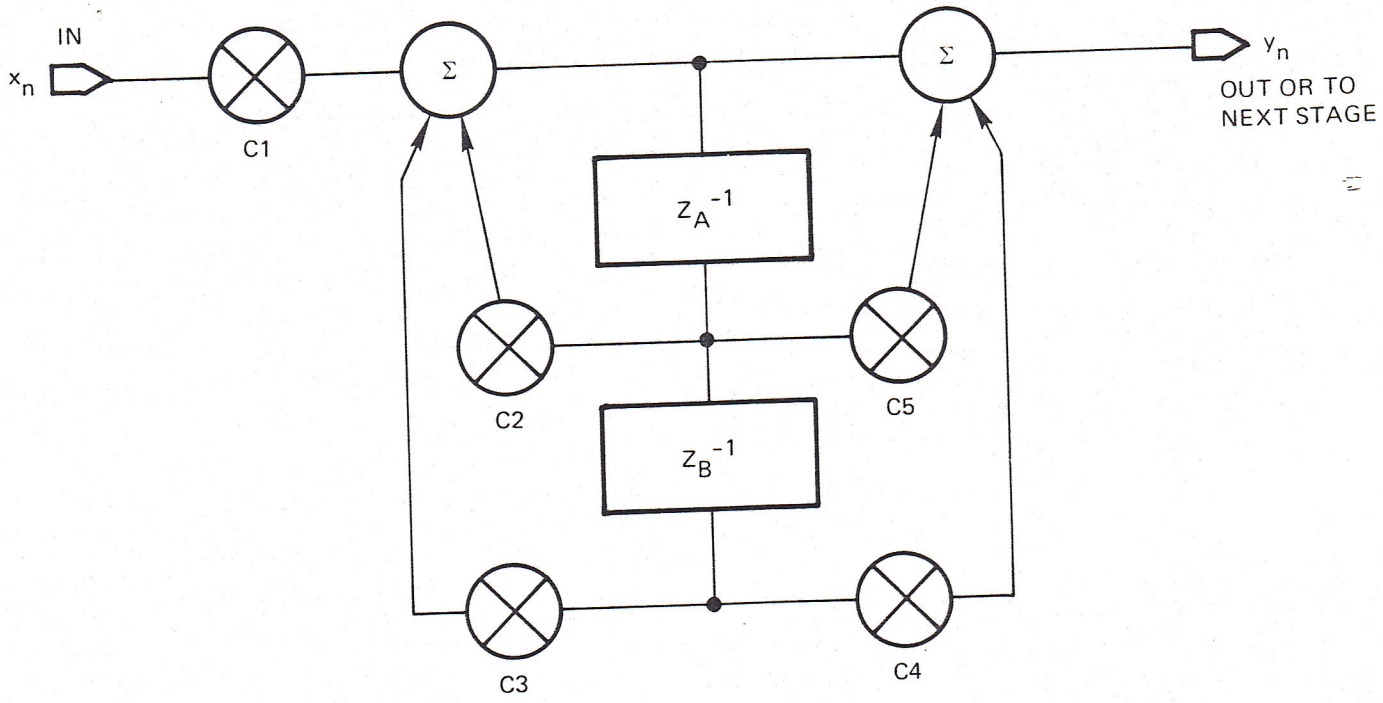


Figure 4. Canonical Form

Figure 4 is a more hardware-oriented version of Figure 3. C_1 through C_5 represent multiplications by coefficients as listed (refer to Figure 1 for coefficient effects on pole and zero locations):

C_1 = input scaling attenuator

$C_2 = +2r_p \cos \theta_p$

$C_3 = -r_p^2$

$C_4 = +r_o^2$

$C_5 = -2r_o \cos \theta_o$

x_n = digital input signal

It may reside in a register of its own, from a previous filter operation, an A/D converter or a memory location. Z_A^{-1} and Z_B^{-1} represent delays of one clock time each. The actual implementation of the delay is usually done with fast random access memory. Schemes for addressing and controlling data in and out of memory are discussed in detail in TRW application note TDSP102. y_n is the filter output, the result of the computations.

- Memory load (MEMLD): Clock enable for loading data into memory.

- Memory address (MEMA).

3) TDC1010J Multiplier Accumulator (MAC)

LSI device which performs all arithmetic and some temporary storage for all digital filter operations.

TDC1010J Controls

- Multiplier accumulator load (MACLD): Clock enable for loading data into X, Y, and accumulator registers.

- Multiplier accumulate (MACACC): Active logic level directing an accumulate operation.

- Multiplier output enable (MACOE): Active logic level allowing the multiplier product onto the main data bus.

4) Multiplier Input Multiplexer (MUX): Data selection device for receiving data from the main bus or from the scratch pad for pipeline operations.

MUX Controls

- Scratch pad enable (SCE): Active logic level enabling data into the multiplier from the scratch memory.

5) Input Source (X): A data register or buffer memory which supplies sequential or batched data to the digital processor.

Source Controls

- Source output enable (XOE): Active logic level enabling X data onto the main data bus.

6) Output Storage Register (OR): A data register whose purpose is to collect filtered data samples and make them available to the outside world. This register may be a large memory.

Output Register Controls

- Output load (ORLD): Clock enable for loading the register.

3.1 SECOND ORDER CANONICAL FILTER COMPUTATION SEQUENCE FOR A REALTIME SOURCE X

With the ALU appropriately defined, the next order of business is to step through a filter sequence, note the data paths, and define a preliminary set of instructions. First, some clarifications are in order. Regarding the ALU control signals, in particular those for the TDC1010J, it is initially assumed that all output enables (i.e., MACOE, MEMOE, etc.) are logic 1 for an enabling condition. This simplifies definition and avoids confusion during the conception stages. In actuality, all elements will have particular requirements suited to the logic (TDC1010J for instance has four output enable inputs and three clock inputs); and in light of the fact that tristate is used for bus driving, all output enables will ultimately require logic 0 conditions. No attempt is made at this point to discuss data loading particulars or valid data conditions on the bus, since there is no need to do so. Discussion on timing and other related details will be deferred until later.

Let data samples, x_n, be available from X every T sec (the sampling interval). Then all computations must be completed before the acceptance of a new x_n. Referring to Figure 6, the result of input summing point additions will be placed in memory at address A_i. Contents of A_{i-1} and A_{i-2} represent identical operations having occurred one and two sample periods before. The recursion diagram of Figure 6 says that data written into memory at sample time T_i (i.e., e_n), consists of past data extracted from memory plus present input data. The difference equation for the summing point defines this process exactly

$$e_n = C_o x_n + [2r_p \cos \theta_p] e_{n-1} - [r_p^2] e_{n-2}$$

Thus, the memory is being used as a tapped digital delay line or shift register. The Z^{-1} (delay) terms are created through memory storage. Actual addressing of memory for continuous operation is discussed thoroughly in TRW application note TDSP102 on address control (Section 4, "Using Random Access Memory as a Tapped Delay Line"). For now let us assume that whenever we wish to read from or write into memory, the proper address is immediately available at the memory address port.

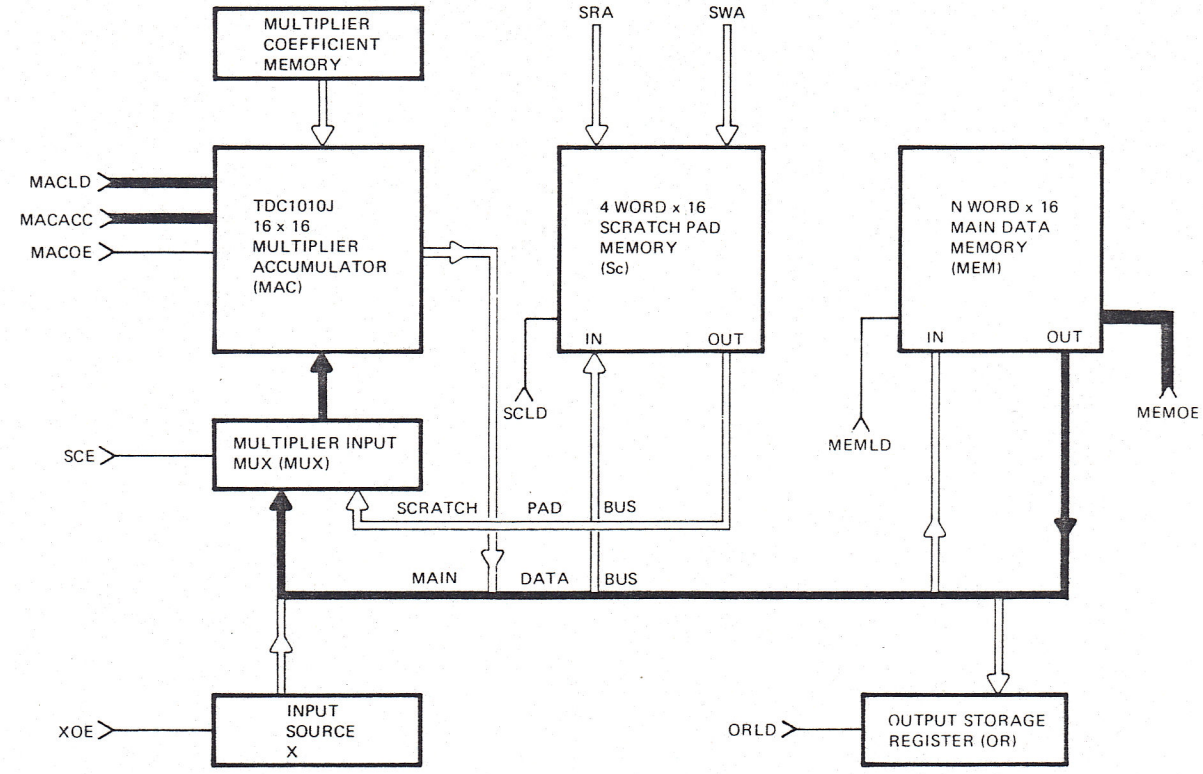
The sequence begins with data sample x_n residing in X. The steps listed in Table 1 define the computation sequence with the appropriate data paths indicated in Figure 7. Figure 5 is helpful in relating signal flow to each instruction.

Table 1. Computational Sequence

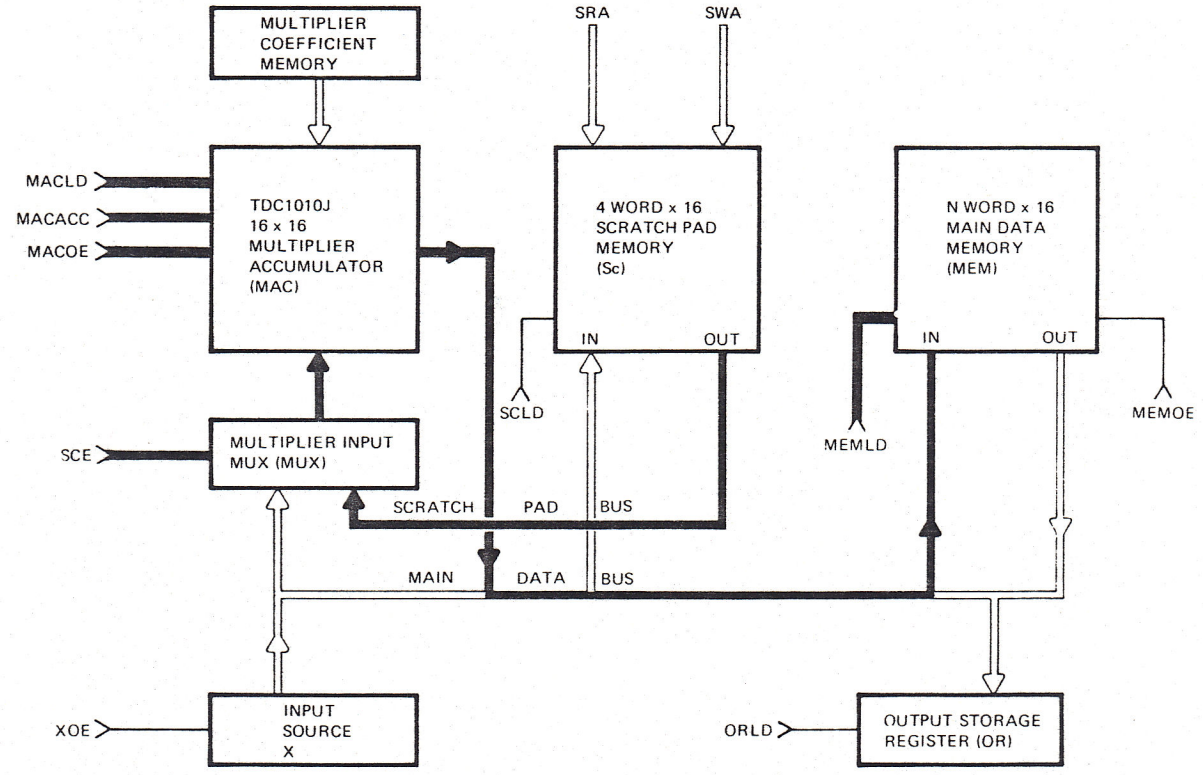
Instruction Description	Instruction	Data Notation
1) Multiply e_{n-1} by C_2, store in 1010 accumulator, store e_{n-1} in Sc address 0	MemMult/STO0	C•MEM→MAC Mem→Sc
2) Multiply x_n by C_1, add result to value residing in 1010 accumulator, place this sum in 1010 accumulator	XMAC	C·X+MAC→MAC
3) Multiply e_{n-1} by C_3, add result to value residing in 1010 accumulator, place this sum in 1010 accumulator	MemMAC	C·Mem+MAC→MAC
4) Load MAC output into MEM, multiply data in Sc address zero by C_5, store product in 1010 accumulator	MWM/ScOMult	MAC→MEM C·Sc→MAC
5) Multiply e_{n-1} by C_4, add result to value in 1010 accumulator, place this sum into accumulator	MemMAC	C·Mem+MAC→MAC
6) Multiply C_n to C_6, add result to value residing in 1010 accumulator, place this sum in 1010 accumulator	MemMAC	C·Mem+MAC→MAC

3.2 COMPUTER TIMING

To complete the exercise on the second order filter implementation some information on timing is included. A basic discussion of the computer controller at this time may serve as a good introduction.



Micro Instruction 3, 5, 6: MEMMAC



Micro Instruction 4: MWM/SCOMULT

Figure 7. Arithmetic Unit Showing Data Paths for Individual Micro Instructions (Cont)

The program addresses are selected in μPP by the sequencer. In the case of a fixed filter sequence, the sequencer need only be a counter which addresses the μPP. However, some flexibility is required in the sequencer if random selection of μPP instructions is desired, or if particular filter subroutines are to be executed and repeated. In general, the sequencer output resides at the same μPP address for a complete clock interval (machine cycle) before incrementing or jumping to another address. Therefore, the μPP reports the state of the machine during that cycle. Some ALU elements may require that certain signals be present for only a certain portion of the cycle (e.g., load pulses, some output enables, etc.). Additional combination logic may be necessary to provide the proper timing intervals for these signals, thereby using the μPP code signals as enables where appropriate.

A good illustration is the diagram in Figure 10 for a typical TDC1010J multiplier loading sequence. Machine cycle T_i contains an instruction requiring a multiplication (or multiply-accumulate). The 1010 contains input registers for X and Y operands. Since it is usually not necessary to waste a complete cycle to load X and Y (typical multiply accumulate time for the 1010 is under 150 nsec), the clock interval may be subdivided such that loading, multiplying, and accumulating are carried out in various portions of the same instruction cycle. The total microinstruction is complete by the end of the fourth fast clock pulse of T_i. Addition of some combinational logic required to accommodate some of the particulars of the ALU elements, results in a savings of microprom and overall machine complexity.

3.3 TIMING FOR THE SECOND ORDER CANONICAL FILTER

The timing diagram in Figure 11 describes each control signal to the ALU for the filter described in Table 1. The sequence starts with a word (e_{n-1}) being read from memory and transferred via the main data bus into the TDC1010J multiplier, and simultaneously into scratch memory address zero. For this instruction, MEMOE is activated (logic 1), the 1010 input loading sequence occurs (MACACC is low indicating no accumulation) and ScLD is activated. At the end of this machine cycle $C_2 \cdot e_{n-1}$ resides in

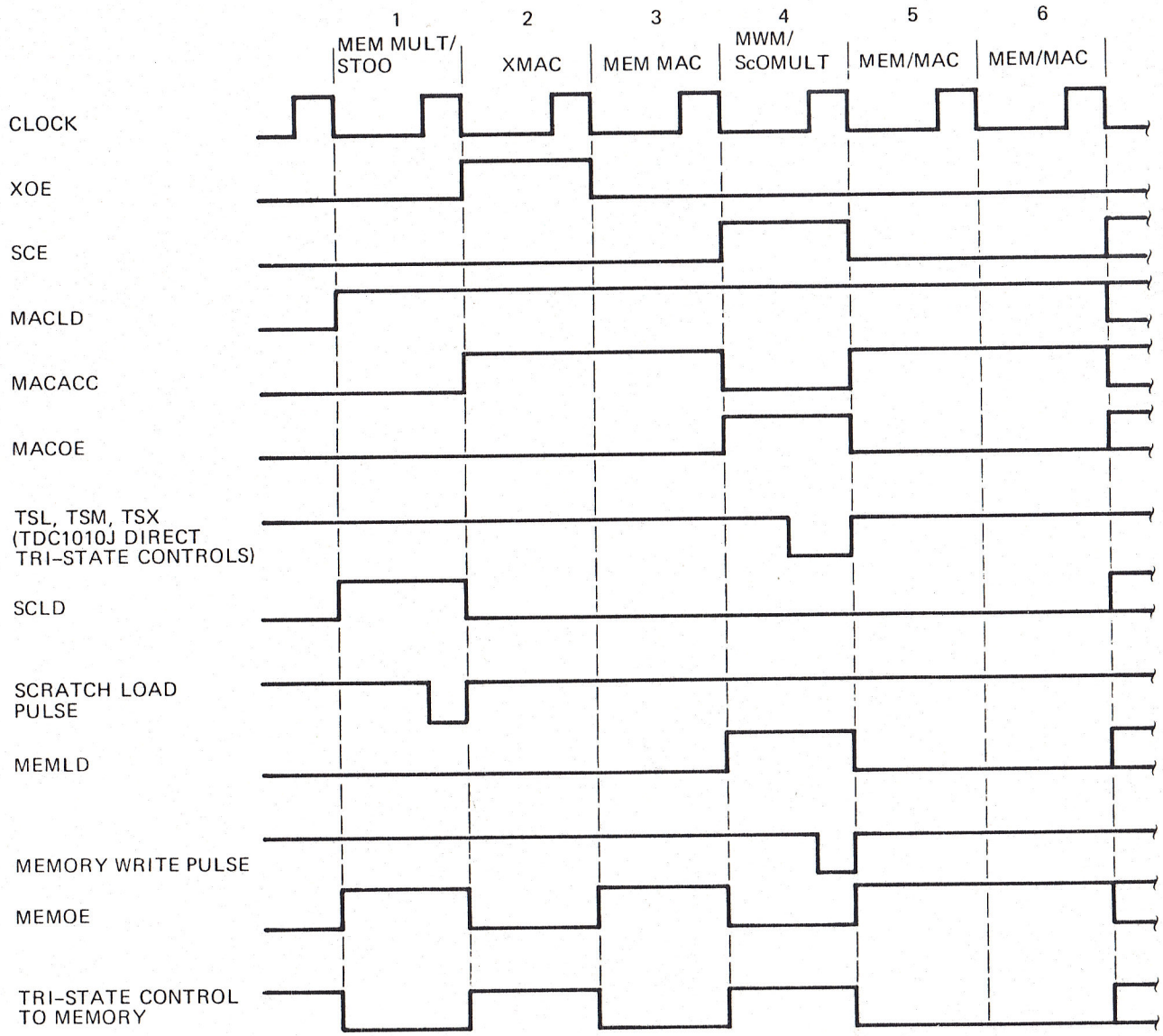


Figure 11. Complete Timing Diagram for Second Order Canonical Sequence in Table 1

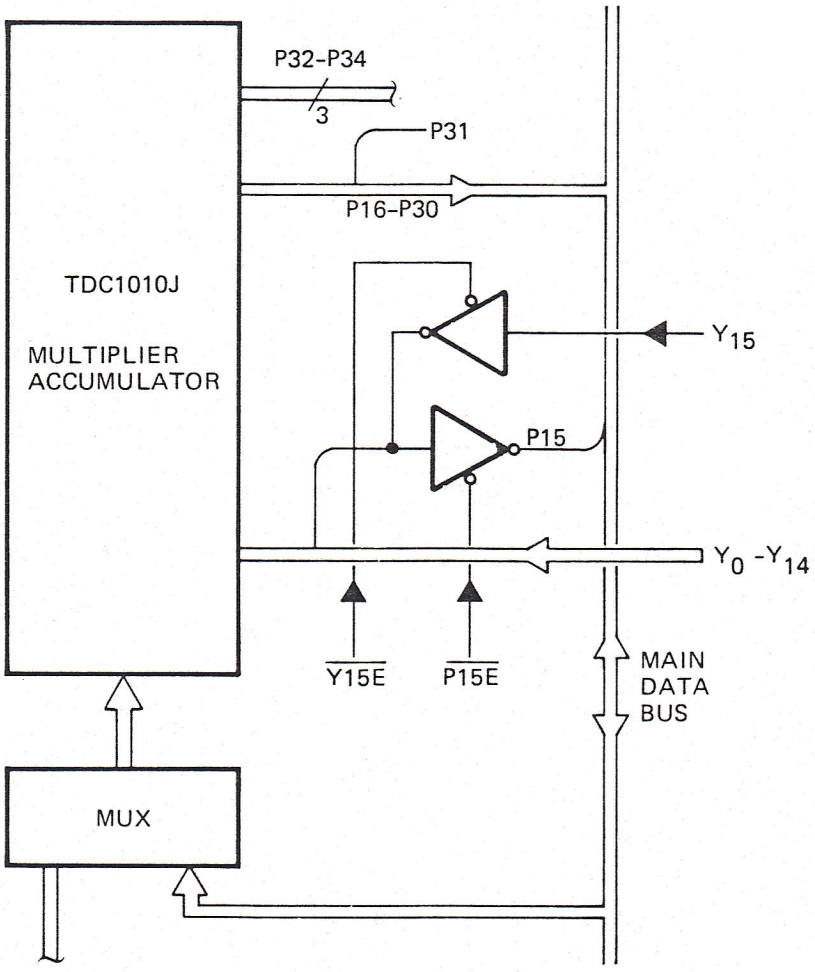


Figure 12. Circuit Configuration for Multiplying by 2.

Figure 1 locates the pole and zero pairs for arbitrary r_o, θ_o, r_p and θ_p.

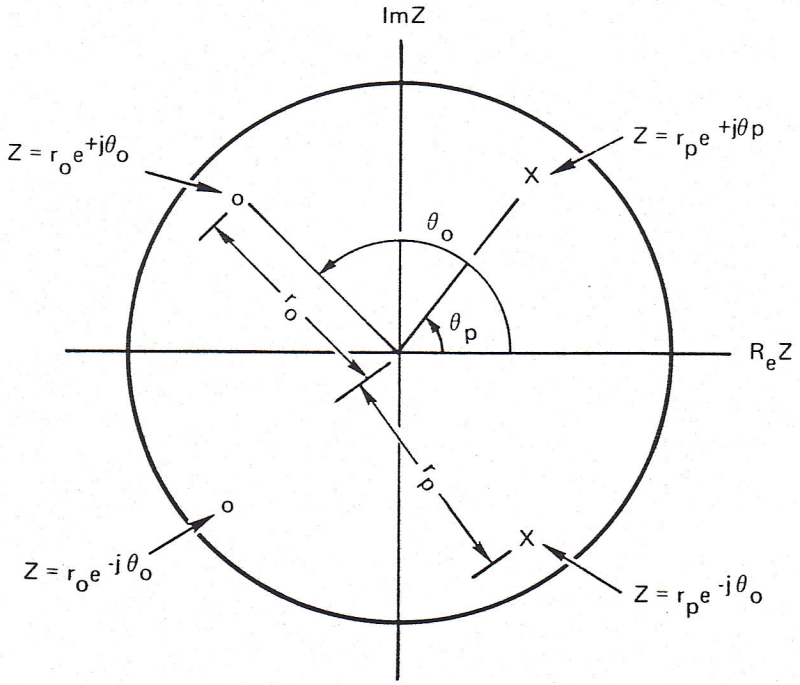


Figure 1. Poles and zeros in Z plane for second order discrete filter (poles: $Z = r_p e^{\pm j\theta p}$, zeros: $Z = r_o e^{\pm j\theta o}$)

Variations of equation (1) are shown with their corresponding pole zero plots in the Z plane (Figure 2a, b, c,).

Once the designer has arrived at a suitable second order transfer function which he wishes to implement, the configuration is arrived at by inspection. Figure 3 is the signal flow graph for the canonical form of this filter.

Variations of equation (1) are shown with their corresponding pole zero plots in the Z plane (Figure 2a, b, c).

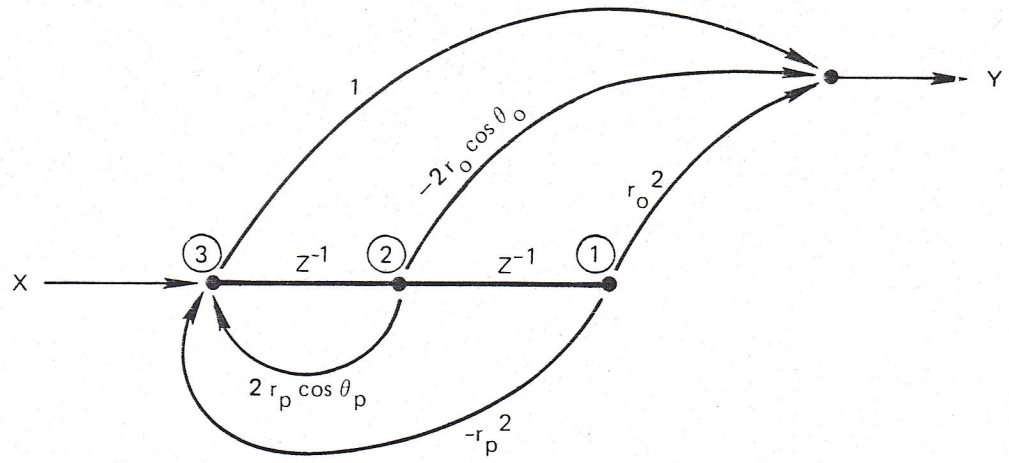


Figure 3. Signal Flow Graph for the Canonical Second Order Filter

The Z^{-1} terms denote delays of one sample time. The appropriate multiplying coefficients are associated with the proper signal path. For example, the output of node 1 (which may correspond to a digital word in a register or particular location in memory) is multiplied by r_o^2 and summed with similar results from nodes 2 and 3 to form output y_n. Node 1 must also be multiplied by $-r_p^2$ to form one of the two feedback branches.

In contrast to analog filters which have similar configurations*, the coefficient multiplications are carried out sequentially in the digital filter, whereas in the analog filter, signals flow simultaneously through resistors and capacitors selected according to the necessary filter parameters. The reader may have deduced at this point that timing is crucial in the digital machine. That topic will be discussed in Section 3 for the second order recursive filter.

---

* One might observe the visual similarities between Figure 2 and the canonical form for an analog filter, when the Z^{-1} elememts are replaced by analog integrator elements. However, the similarities end here, since any linear operator is usable as the element, and the resulting transforms have unique forms and are particularly suited to specific applications. This topic falls under the theory of state variables and state space.

It may finally reside in the TDC1010J accumulator as will be shown. Although Figure 3 gives the feeling that signals are simultaneously routed to their proper locations, each multiplication, add, and memory access is usually done separately in time as previously mentioned. When the filter is implemented with only one hardware multiplier and one adder whose functions are timeshared around the filter branches, a two-pole, two-zero filter can be executed in about 800 nsec. A 10-section elliptical filter can be implemented in under 10 μsec. This means that for processing realtime contiguous data, a 100 kHz sampling rate could be realized.

3. IMPLEMENTING THE DIGITAL FILTER

Various architectures are suitable for the implementation of digital filter hardware. The one illustrated in Figure 5 fits most design requirements, including those for the second order filters discussed in the previous sections. It is a two data bus system (plus coefficient and control address buses), allowing good speed through various pipelining modes. The following is a list of control signals for the ALU.

1) Scratch Pad Memory(SC): Four words of memory used for temporary storage of data. The filter microinstructions require that this memory need only receive data from the TDC1010J and drive data via the scratch bus to the multiplier.

   Scratch Pad Controls

   - Scratch Pad Load (SCLD): Clock enable used to load the memory.
   - Scratch read address, scratch write address (SRA, SWA): Two bit words directing data in and out of memory. Addresses may be tied together for most applications (SCA).

2) Main Data Memory (MEM): Temporary storage for the data to be processed. Memory size is dependent on the accumulated order of the digital filter in a realtime system. An eighth order filter requires eight words of memory.

   Memory Controls

   - Memory output enable (MEMOE): Logic level for enabling memory data onto the main bus.

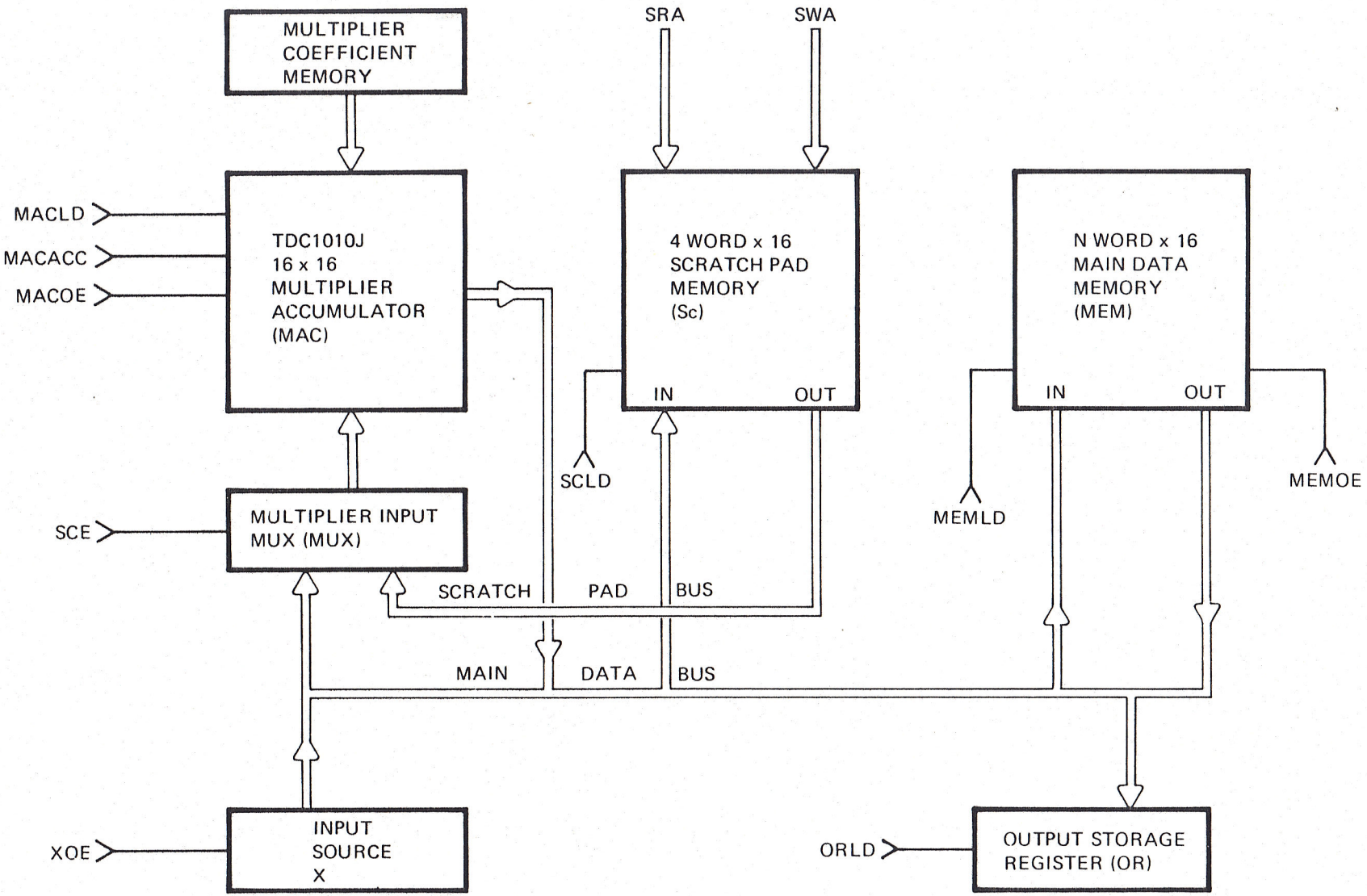


Figure 5. Two Data Bus Arithmetic System

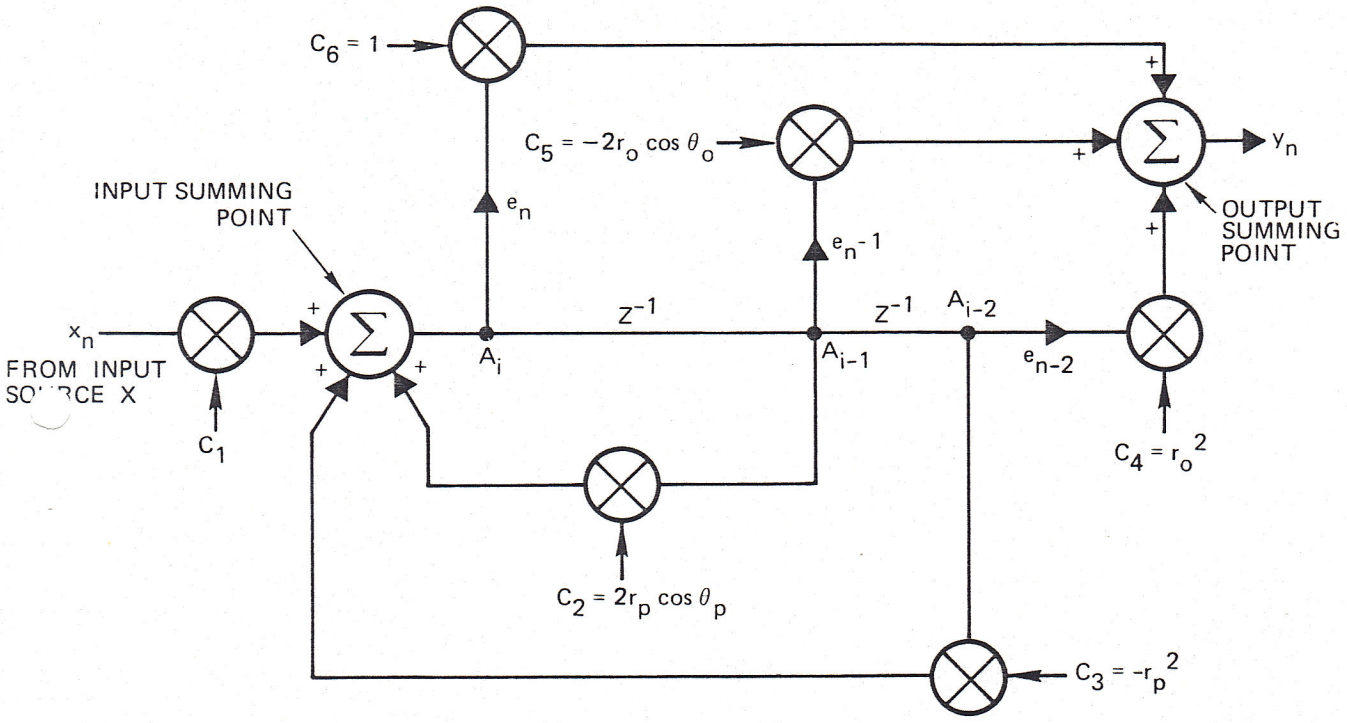
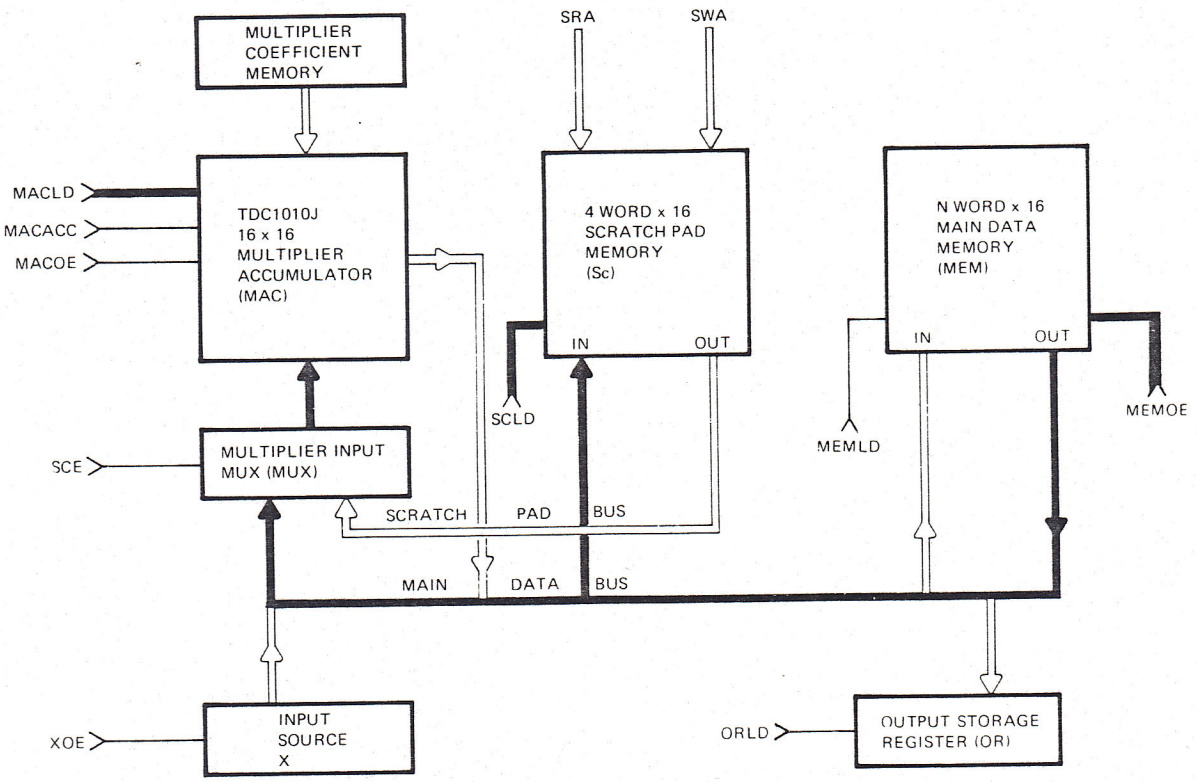


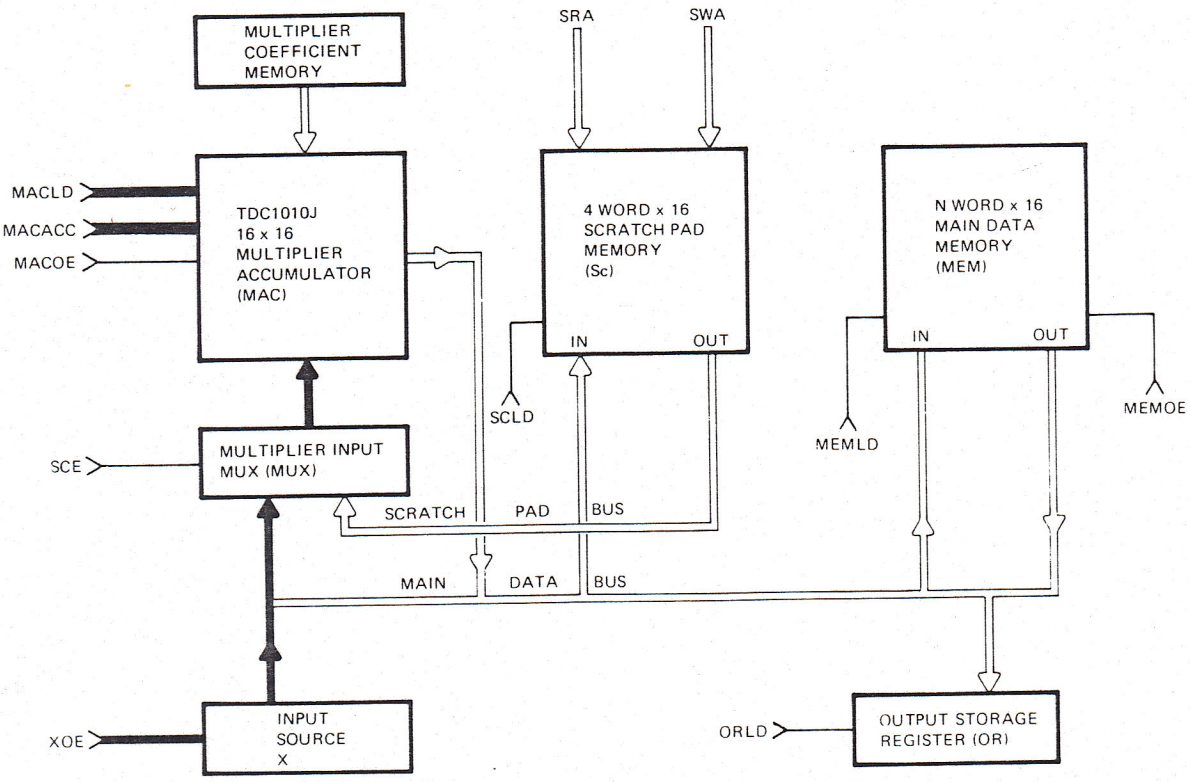
Figure 6. Computation Diagram for Second Order Filter

At the completion of this sequence, y_n resides in the 1010 accumulator register where it may be used as the input of another filter section. The user may find that additional variations on these instructions are necessary to suit his needs, depending on the data source or the type of filters to be implemented. This usually amounts to nothing more than a minor change in or addition to microcode.

Variations in architecture may effect filter computation speed for some implementations. If speed is not a problem, the scratch pad memory may be used only in extraneous situations where temporary storage is needed between filter operations. In this case the scratch pad bus could be eliminated, but the pipeline capability that it provides would be sacrificed. Eliminating this memory altogether is not advisable, since main memory address controllers usually have a definite sequential nature, and do not lend themselves readily to the storage of randomly sourced data.



Micro Instruction 1: MEMMULT/ST00



Micro Instruction 2: XMAC

Figure 7. Arithmetic Unit Showing Data Paths for Individual Micro Instructions

Figure 8 is a simplified diagram of the total processor. The controller consists of microprogram prom (μPP) and a sequencer. The μPP is usually a very wide memory (typically 20 to 60 bits). Each address contains the output enables, load enables, multiplexer codes, etc., for a corresponding instruction. Additional control bits may also be included as next address information which is fed back to the sequencer. As an example, the ALU portion of the μPP may be broken down as shown in Figure 9. The microcode corresponds to instruction MWM/ScMult (see Table 1, instruction 4).

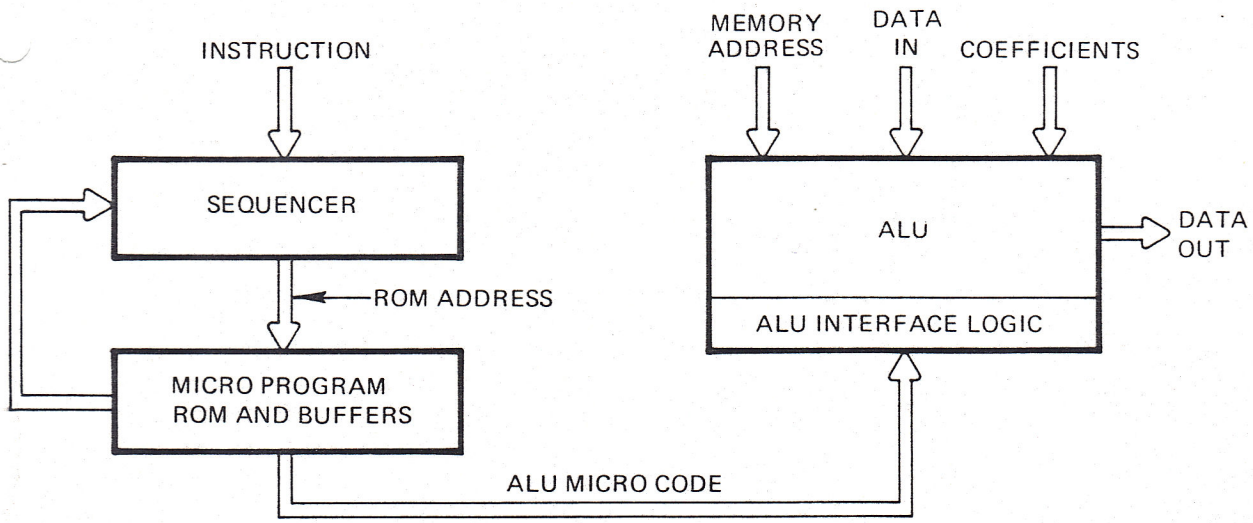


Figure 8. Simplified Diagram of the Complete Digital Filter Processor

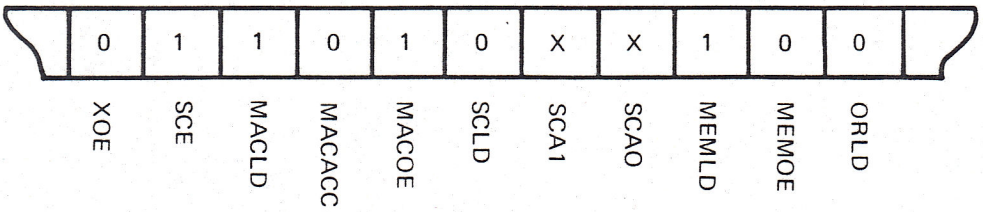


Figure 9. Typical ALU Portion of Microprogram Prom (MWM/ScMult microcode resides in memory at this hypothetical address)

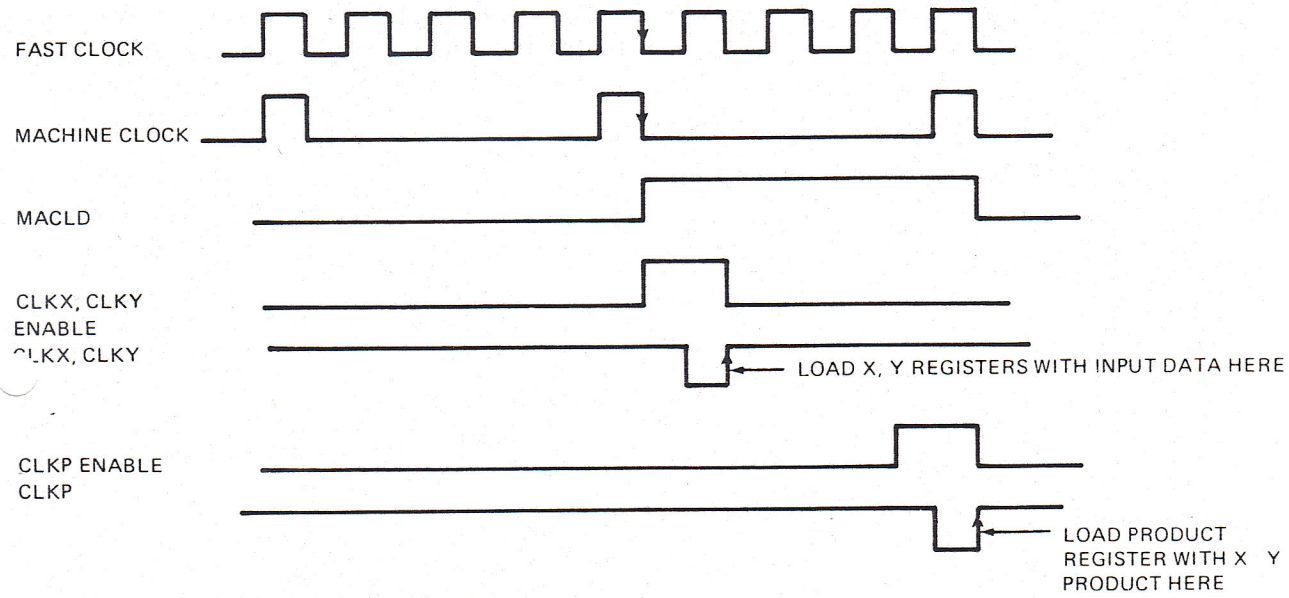


Figure 10. Timing Diagram for a Typical Multiplier Load Sequence

the multiplier output register and e_{n-1} resides in Sc address zero. The simultaneous loading of Sc0 is not absolutely necessary since e_{n-1} can be taken again from memory at a later time (instruction 4) and multiplied by C_5 to form part of the output sum (a simple MemMult operation would suffice.)

However, since the main memory bus is tied up during the writing operation in instruction 4, we take advantage of the direct bus from scratch pad to multiplier and perform a pipeline data transfer, thereby saving a machine cycle. The reader begins to realize that the combination of bus structure and instruction set can affect program execution time significantly. Had the main memory input and output ports been brought out separately, instruction 1 could have also been simplified to a single transfer instruction (MemMult) with the original speed realized. For systems with large memory requirements this may not be possible, in which case the services of a scratch pad memory would be required.

The reader may note that all output enables exist for the duration of the machine cycle with the exception of the TDC1010J output enables (TSL, TSX, TSM). These are active (logic low) during the latter half of a cycle where MACOE is present from μPP. This requirement is due to the timeshared nature of the TDC1010J inputs with the 16 least significant product outputs. Since the multiplier must be capable of providing coefficients greater than one for second order filters, a data shift must be effected in the multiplier output/data bus connection. Assuming that the data bus is a 16 bit bus (D_0-D_{15}), a coefficient of 2 can be realized through the multiplier if its P30 output is connected to D_{15}. This means that P_{15} is connected to D_0. Now P_{15} is timeshared with Y_{15}. Hence, if Y and X are loaded in the first half machine cycle, P_{15} through P_{30} may be enabled during the last half without causing logic ambiguities on the bus. Figure 12 gives a scheme for multiplication by 2. The tristate buffer from the coefficient bus facing Y_{15}-P_{15} is enabled during X, Y loading. If the coefficients come from a memory with tristate outputs, the memory may be enabled at this time. P_{15} is enabled onto D_0 in the latter half of the cycle.