

Digital Signal Processor

Field of the Invention

This invention relates to a Digital Signal Processor (DSP), and
5 more particularly, to the architecture of a DSP.

Background of the Invention

In general, DSPs are designed to process large amounts of data
as quickly as possible and their architecture is optimised for this
purpose. In most DSP applications the processing carries out
10 arithmetical operations to multiply data samples by predetermined
coefficients and to accumulate the sum of the products according to
the expression:-

$$Y = a_0.X_0 + a_1.X_1 + a_2.X_2 + a_3.X_3 + \dots + a_n.X_n$$

where a is the predetermined coefficient, and

15 X is the data sample.

In order to carry out this processing efficiently, the
architecture of the known DSPs utilises a Random Access Memory
(RAM) to store the predetermined coefficients and the data samples
and a multiplier to multiply the appropriate coefficient and data
20 sample. The product of the multiplier is then passes to an adder
where it is added to the previous sum and the new sum is stored in
an accumulator. In some cases values need to have their bits shifted
so as to maintain their accuracy and, in order to do this, a shifter is
provided in parallel with the multiplier and the value to be shifted
25 and the value stored in the accumulator need to be stored in the
RAM to allow the value to be shifted to pass through the shifter
without being added to the previous sum in the adder. All the
values being written to the RAM from the accumulator or being read
from the RAM to the shifter or multiplier pass via a data bus.

30 Although this architecture can efficiently process according to
the multiply-and-accumulate expression given above, there are
some applications, such as correcting picture geometry on a monitor,
that require digital signal processing according to the polynomial
expression:

35
$$Y = a_0 + a_1.X + a_2.X^2 + a_3.X^3 + \dots + a_n.X^n$$

Using the conventional DSP architecture to calculate such a
polynomial expression is not very efficient since it requires that

2-2000138

temporary data must be written to and read from the RAM many times in order to calculate one product and therefore uses many extra clock cycles.

Brief Summary of the Invention

5 The present invention therefore seeks to provide a DSP which overcomes, or at least reduces the above-mentioned problems of the prior art.

Accordingly, in a first aspect, the invention provides a Digital Signal Processor (DSP) comprising:

10 a Random Access Memory (RAM) for storing values including predetermined coefficients and received data;

 a first latch coupled to the RAM for temporarily storing a value from the RAM;

 a multiplier having a first input coupled to the first latch and a
15 second input, the multiplier being arranged for multiplying a value from the first input and a multiplicand from the second input to produce a product at an output;

 a second latch having an input coupled to the output of the multiplier and an output selectively coupled to the second input of
20 the multiplier, the second latch being arranged for temporarily storing the product ;

 an adder having a first input coupled to the output of the second latch, a second input and an output, the adder being arranged for adding values at the first and second inputs to produce
25 a sum at the output thereof; and

 an accumulator having an input coupled to the output of the adder and an output selectively coupled to the second input of the adder, the accumulator being arranged for storing the sum from the adder.

30 In a preferred embodiment, the DSP further comprises a shifter coupled between the output of the multiplier and the input of the second latch for selectively shifting significant digits of the product before it is temporarily stored in the second latch.

 Preferably, the DSP can further comprise a first multiplexer
35 having a first input coupled to the output of the second latch and an output coupled to the second input of the multiplier for selectively coupling the output of the second latch and the second input of the

multiplier. Conveniently, the first multiplexer further comprises a second input coupled to the output of the accumulator for selectively coupling the output of the accumulator and the second input of the multiplier. The first multiplexer preferably further comprises a
5 third input coupled for receiving a value of 1 for selective use by the multiplier as the multiplicand.

In one embodiment, the DSP further comprises a second multiplexer having a first input coupled to the output of the accumulator and an output coupled to the second input of the adder
10 for selectively coupling the output of the accumulator and the second input of the adder. Preferably, the second multiplexer further comprises a second input coupled for receiving a value of 0 for selective use by the adder.

According to a second aspect, the invention provides a method
15 of determining the sum of a polynomial expression comprising a plurality of terms, each term comprising a coefficient multiplied by a data value raised to a power value, the method comprising the steps of:

storing the coefficients for each term, the power values for
20 each term and the data value in a memory;

determining a term product for each term of the plurality of terms by:

latching the data value in a first latch;

multiplying the data value from the first latch by a value
25 of one in a multiplier and storing the product in a second latch;

multiplying the data value from the first latch by the product from the second latch in the multiplier and storing the product in the second latch;

repeating the previous multiplying step a number of
30 times equivalent to the power value for the term less one to produce a data value product latched in the second latch;

latching the coefficient for the term in the first latch; and

multiplying the coefficient from the first latch by the data value product from the second latch in the multiplier to
35 produce the term product latched in the second latch;

adding the term product for each term from the second latch to a previous accumulator value from an accumulator in an adder to produce a new accumulator value; and

repeating the previous adding step a number of times
5 equivalent to the number of terms in the polynomial expression whereby the final accumulator value is the sum of all of the term products forming the polynomial expression.

Brief Description of the Drawings

One embodiment of the invention will now be more fully
10 described, by way of example, with reference to the drawings, of which:

FIG. 1 shows a schematic diagram of a conventional DSP architecture; and

FIG. 2 shows a schematic diagram of a DSP architecture
15 according to one embodiment of the present invention.

Detailed Description of the Drawings

Thus, as mentioned above, in monitor geometry correction applications, the calculation model for a correction waveform can normally be approximated by a polynomial expression as follows:

20
$$Y = a_0 + a_1.X + a_2.X^2 + a_3.X^3 + \dots + a_n.X^n$$

where a_i is the predetermined coefficient which is normally stored in the RAM, X is the data sample which is received at the inputs, and Y is the output used to control the geometry correction circuit.

FIG. 1 shows a block diagram of a conventional DSP
25 architecture, in which a data bus 40 is used for data communication between a RAM 32, a shifter 35, a first latch 33, a multiplier 34, an accumulator 38 and an Input/Output (I/O) buffer circuit 39. During calculation of a term, such as $a_i.X^i$, temporary data needs to be moved back and forth to/from the RAM 32.

30 To calculate such a term, the coefficient a_i is first communicated from the RAM 32, via the data bus 40, to the first latch 33, where it is temporarily stored. On the next clock cycle, this coefficient is taken at one input of the multiplier 34 and the data X is taken at the second input of the multiplier and the two are
35 multiplied together and passed from the output of the multiplier to a second latch 36, where the product is stored. This product then passes through a multiplexer 41 and an adder 37 to the accumulator

38, from where, on successive clock cycles, it can be written into the RAM 32 and then read from the RAM 32 into the first latch 33, so that it can be multiplied in the multiplier 34 by the data X. It will be appreciated that these steps will need to be repeated $i-1$ times in order to allow for the term $a_i.X^i$ to be calculated. As a result, it takes a large number of clock cycles to finish calculation of a polynomial expression and is not very efficient. The adder 37, as will be appreciated, is used for adding two such terms together, and the shifter 35 is used, when required via the multiplexer 41, to shift the significant bits of the product from the second latch 36.

For example, the calculation sequence for the expression

$$Y = a_1.X + a_3.X^3$$

is listed as follows, showing that a total of 17 clock cycles are required:

1. RAM(a_1) -> 1st Latch
2. 1st Latch(a_1) * RAM(X) -> 2nd Latch
3. 2nd Latch($a_1.X$) -> Accumulator
4. Accumulator($a_1.X$) -> RAM
5. RAM(a_3) -> 1st Latch
6. 1st Latch(a_3) * RAM(X) -> 2nd Latch
7. 2nd Latch($a_3.X$) -> Accumulator
8. Accumulator($a_3.X$) -> RAM
9. RAM($a_3.X$) -> 1st Latch
10. 1st Latch($a_3.X$) * RAM(X) -> 2nd Latch
11. 2nd Latch($a_3.X^2$) -> Accumulator
12. Accumulator($a_3.X^2$) -> RAM
13. RAM($a_3.X^2$) -> 1st Latch
14. 1st Latch($a_3.X^2$) * RAM(X) -> 2nd Latch
15. 2nd Latch($a_3.X^3$) + RAM($a_1.X$) -> Accumulator
16. Accumulator($a_1.X + a_3.X^3$) -> RAM
17. RAM($a_1.X + a_3.X^3$) -> I/O

Thus, as can be seen, using the conventional DSP architecture to calculate such a polynomial expression is not very effective since much temporary data must be moved back and forth to/from the RAM and many extra clock cycles are needed.

Turning now to FIG. 2, there is shown a block diagram of a DSP architecture in accordance with an embodiment of the present

invention. This DSP architecture is designed to perform the polynomial calculation for monitor geometry correction applications more effectively.

5 In this embodiment, a data sample X , which is received at an input terminal 11 is firstly stored in a RAM 12. The predetermined coefficients a_i are also stored in the RAM 12. To calculate a term, such as $a_i.X^i$, the predetermined coefficient a_i is first loaded into a first latch 13, whose output is coupled to a first input of a multiplier 14. A second input of the multiplier 14 is coupled to an output of a
10 first multiplexer 20. The first multiplexer 20 has three inputs and selects the value at one of the three inputs as a multiplicand to the second input of the multiplier 14. In the first instance, it selects the first input 22, which is a constant value of "1". Since the multiplicand at the second input of multiplier 14 is 1, the multiplier
15 14 simply passes its first input (a_i) to its output which is coupled to the input of a shifter 15. The purpose of the shifter 15 is for data scaling in order to minimize truncation error. An output of the shifter 15 is coupled to an input of a second latch 16. Thus, if no scaling is necessary, the coefficient a_i is loaded into the second latch
20 16 at the next clock cycle and the data sample X is then read from the RAM 12 and loaded into the first latch 13.

The first multiplexer 20 now selects its third input to provide the multiplicand to the multiplier 14. The third input of the first multiplexer 20 is coupled to the output of the second latch 16, which
25 holds the coefficient a_i at this time. Thus, the data sample X is now passed to the first input of the multiplier 14 and the coefficient a_i is passed to the second input of the multiplier 14, where they are multiplied together. The product of the multiplication, the value $a_i.X$ is provided at the output of the multiplier 14 and is scaled by the
30 shifter 15, if necessary, before it is loaded into the second latch 16. Without changing the contents of the first latch 13, which currently holds the data sample X , the first multiplexer 20 again selects its third input, which is the output of the second latch 16, so that the inputs to the multiplier will be the data sample X and the output
35 from the second latch 16, which is the value $a_i.X$. This process is repeated $i-1$ times, until the value $a_i.X^i$ is present in the second latch 16.

At the end of the calculation of a term $a_i.X^i$, the value of the term $a_i.X^i$, is either loaded into an accumulator 18 for the first time or added to the value already in the accumulator 18 thereafter. The input of the accumulator 18 is coupled to an output of an adder 17, which has a first input coupled to the output of the second latch 16 and a second input coupled to the output of a second multiplexer 21. To load the value of a term into the accumulator 18, the second multiplexer 21 selects its first input 23, which is a constant value of "0", as the addend to the adder 17. Since the addend is a value of "0", the adder 17 simply passes the value of the term $a_i.X^i$ from its first input to its output which is coupled to the input of the accumulator 18 into which it is loaded at the next clock cycle.

To add the value of a term to the accumulator 18, the second multiplexer 21 selects its second input which is coupled to the output of the accumulator 18 to provide the value laded in the accumulator 18 as the addend to the second input of adder 17. In this case, the sum of the contents of the second latch 16 ($a_i.X^i$) and the contents in the accumulator 18 (the sum of previously calculated terms) is generated by the adder 17, and then loaded into the accumulator 18 at the following clock cycle. Repeating the same process for each term, the final result of the polynomial expression can be obtained in accumulator 18. The output of the accumulator 18 is coupled via an output buffer circuit 19 to an output terminal 24.

The output of the accumulator 18 is also preferably coupled back to the second input of first multiplexer 20, where it can be passed to allow for calculations which utilise the result Y of the polynomial expression.

In order to illustrate the increased efficiency of this architecture, the calculation sequence for the earlier example of

$$Y = a_1.X + a_3.X^3$$

is listed as follows, showing that a total of only 11 clock cycles are required:

1. RAM(a_1) -> 1st Latch
2. 1st Latch(a_1) * Constant 1 -> 2nd Latch
3. RAM(X) -> 1st Latch
4. 1st Latch(X) * 2nd Latch(a_1) -> 2nd Latch

5. 2nd Latch($a_1.X$) + Constant 0 $\rightarrow$ Accumulator ;
RAM(a_3) $\rightarrow$ 1st Latch
6. 1st Latch(a_3) * Constant 1 $\rightarrow$ 2nd Latch
7. RAM(X) $\rightarrow$ 1st Latch
8. 1st Latch(X) * 2nd Latch(a_3) $\rightarrow$ 2nd Latch
9. 1st Latch(X) * 2nd Latch($a_3.X$) $\rightarrow$ 2nd Latch
10. 1st Latch(X) * 2nd Latch($a_3.X^2$) $\rightarrow$ 2nd Latch
11. 2nd Latch($a_3.X^3$) + Accumulator($a_1.X$) $\rightarrow$ Accumulator ;
Accumulator($a_1.X + a_3.X^3$) $\rightarrow$ Output

10 It will thus be seen that the architecture of this embodiment eliminates the need for temporary data to be moved back and forth to/from the RAM and hence simplifies the data bus structure and requires fewer clock cycles to perform the same calculation of a polynomial expression than does the conventional DSP architecture.

15 It will be appreciated that although only one particular embodiment of the invention has been described in detail, various modifications and improvements can be made by a person skilled in the art without departing from the scope of the present invention.



Claims

1. A Digital Signal Processor (DSP) comprising:
 - a Random Access Memory (RAM) for storing values including
5 predetermined coefficients and received data;
 - a first latch coupled to the RAM for temporarily storing a value from the RAM;
 - a multiplier having a first input coupled to the first latch and a second input, the multiplier being arranged for multiplying a value
10 from the first input and a multiplicand from the second input to produce a product at an output;
 - a second latch having an input coupled to the output of the multiplier and an output selectively coupled to the second input of the multiplier, the second latch being arranged for temporarily
15 storing the product ;
 - an adder having a first input coupled to the output of the second latch, a second input and an output, the adder being arranged for adding values at the first and second inputs to produce a sum at the output thereof; and
 - 20 an accumulator having an input coupled to the output of the adder and an output selectively coupled to the second input of the adder, the accumulator being arranged for storing the sum from the adder.
- 25 2. A DSP according to claim 1, further comprising a shifter coupled between the output of the multiplier and the input of the second latch for selectively shifting significant digits of the product before it is temporarily stored in the second latch.
- 30 3. A DSP according to either claim 1 or claim 2, further comprising a first multiplexer having a first input coupled to the output of the second latch and an output coupled to the second input of the multiplier for selectively coupling the output of the second latch and the second input of the multiplier.
- 35 4. A DSP according to claim 3, wherein said first multiplexer further comprises a second input coupled to the output of the



accumulator for selectively coupling the output of the accumulator and the second input of the multiplier.

5. A DSP according to either claim 3 or claim 4, wherein said first
5 multiplexer further comprises a third input coupled for receiving a value of 1 for selective use by the multiplier as the multiplicand.

6. A DSP according to any preceding claim, further comprising a
10 second multiplexer having a first input coupled to the output of the accumulator and an output coupled to the second input of the adder for selectively coupling the output of the accumulator and the second input of the adder.

7. A DSP according to claim 6, wherein said second multiplexer
15 further comprises a second input coupled for receiving a value of 0 for selective use by the adder.

8. A method of determining the sum of a polynomial expression comprising a plurality of terms, each term comprising a coefficient
20 multiplied by a data value raised to a power value, the method comprising the steps of:

storing the coefficients for each term, the power values for each term and the data value in a memory;

25 determining a term product for each term of the plurality of terms by:

latching the data value in a first latch;

multiplying the data value from the first latch by a value of one in a multiplier and storing the product in a second latch;

30 multiplying the data value from the first latch by the product from the second latch in the multiplier and storing the product in the second latch;

repeating the previous multiplying step a number of times equivalent to the power value for the term less one to produce a data value product latched in the second latch;

35 latching the coefficient for the term in the first latch; and



multiplying the coefficient from the first latch by the data value product from the second latch in the multiplier to produce the term product latched in the second latch;

5 adding the term product for each term from the second latch to a previous accumulator value from an accumulator in an adder to produce a new accumulator value; and

repeating the previous adding step a number of times equivalent to the number of terms in the polynomial expression whereby the final accumulator value is the sum of all of the term
10 products forming the polynomial expression.

Abstract

Digital Signal Processor

5 A Digital Signal Processor (DSP) includes a Random Access
Memory (RAM) (12) for storing predetermined coefficients and data
samples and a first latch (13) coupled to the RAM (12) for
temporarily storing either a coefficient or a data sample from the
RAM. A multiplier (14) is arranged for multiplying a value from a
10 first input, which is coupled to the first latch (13), and a
multiplicand from a second input to produce a product. A second
latch (16) is coupled to temporarily store the product from the
multiplier (14). An output of the second latch (16) is coupled back
to the second input of the multiplier (14), via a multiplexer (20), to
15 enable the multiplier (14) to multiply the product by the value in
the first latch without requiring further reading of the RAM, thus
enabling efficient calculation of a term including X^i . The output of
the second latch (16) is also coupled to an adder (17) and an
accumulator (19).

20

FIG. 2

1/2

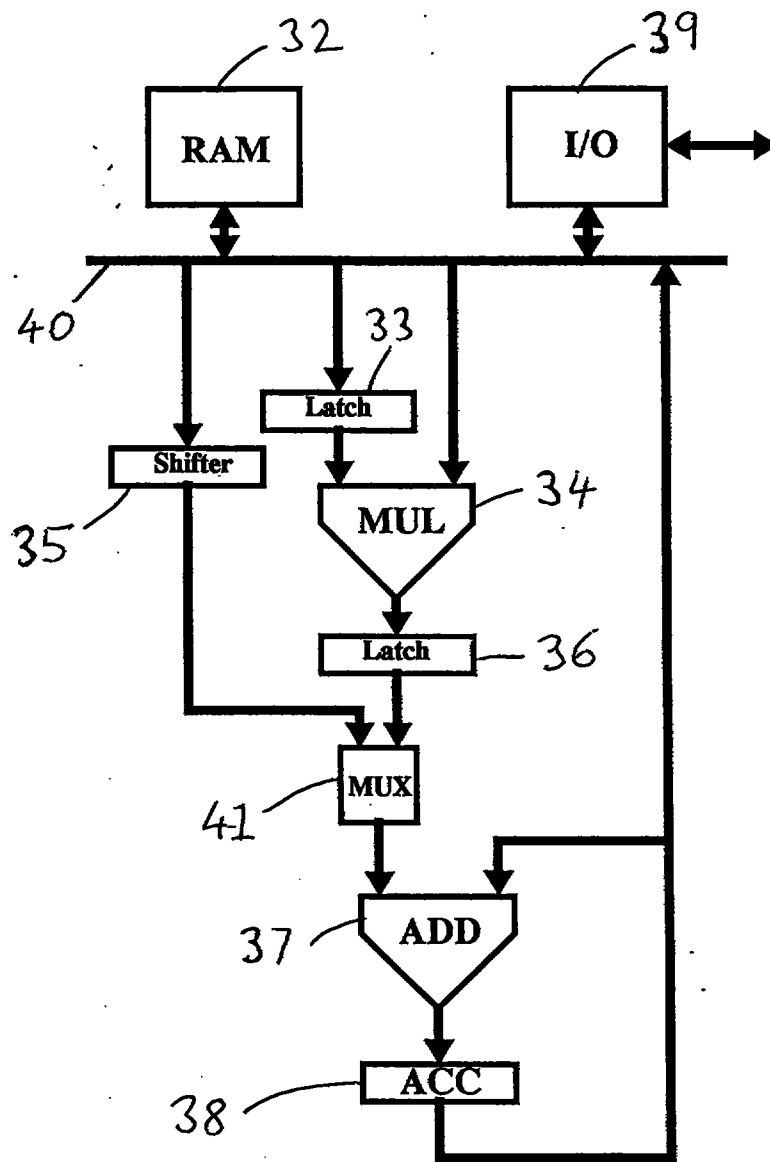


FIG. 1
PRIOR ART

2/2

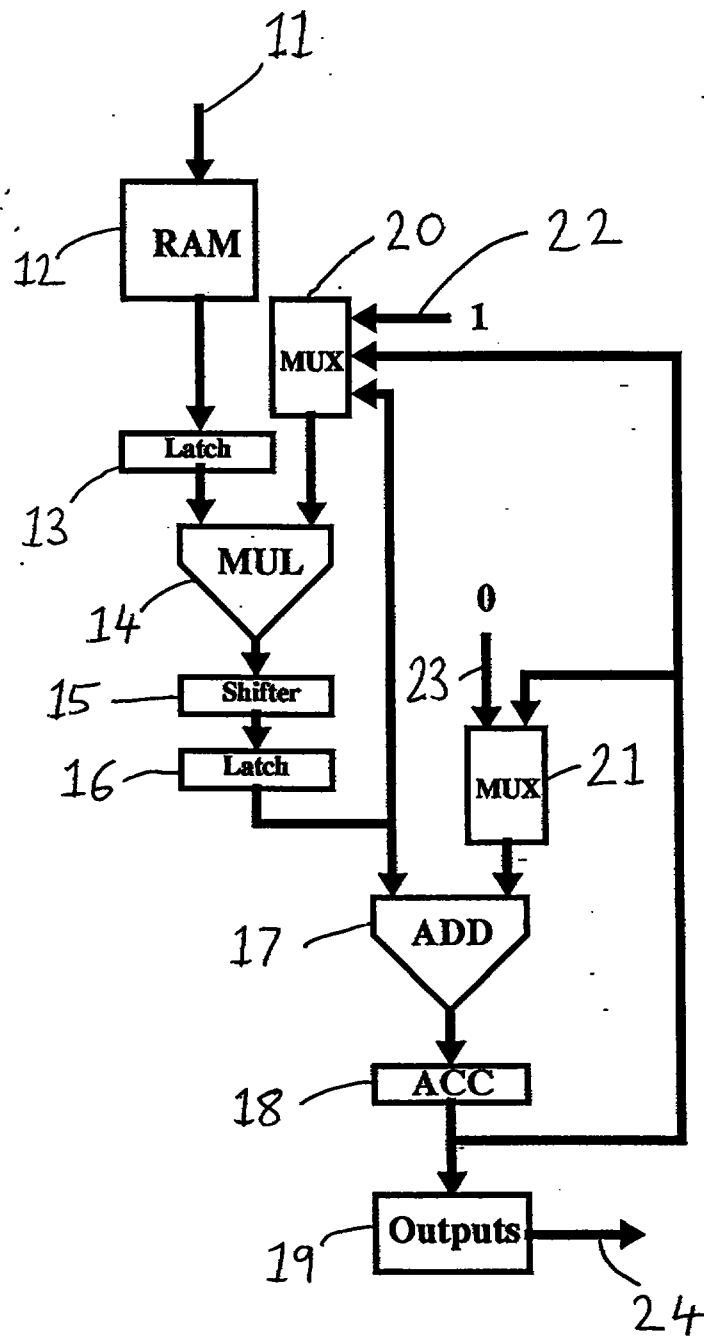


FIG. 2

00000000