

## **DSP EXECUTION UNIT FOR EFFICIENT ALTERNATE MODES OF OPERATION**

### **TECHNICAL FIELD**

This invention relates to digital signal processors,  
5 and more particularly to digital signal processors for  
processing reduced data sizes.

### **BACKGROUND**

Digital signal processing is concerned with the  
representation of signals in digital form and the  
10 transformation or processing of such signal representation  
using numerical computation. Digital signal processing is  
a core technology for many of today's high technology  
products in fields such as wireless communications,  
networking, and multimedia. One reason for the prevalence  
15 of digital signal processing technology has been the  
development of low cost, powerful digital signal processors  
(DSPs) that provide engineers the reliable computing  
capability to implement these products cheaply and  
efficiently. Since the development of the first DSPs in the  
20 early 1980's, DSP architecture and design have evolved to  
the point where even sophisticated real-time processing of  
video-rate sequences can be performed.

Typically, DSPs are constructed of a fixed size. The size of a DSP is selected based on the maximum size of the data to be processed. For example, a DSP that will be used to process 16 bit data needs multipliers and  
5 accumulators of a specific size to ensure that the data is processed correctly. While these DSPs can process data having less than 16 bits, doing so causes a portion of the DSP hardware to remain unused. This decreases the efficiency of the DSP.

10

#### DESCRIPTION OF DRAWINGS

These and other features and advantages of the invention will become more apparent upon reading the following detailed description and upon reference to the accompanying drawings.

15

Figure 1 is a schematic of a digital signal processor used for general n-bit modes of operation in accordance with an embodiment of the present invention.

20

Figure 2 a schematic of a digital signal processor for either n-bit or (n/2)-bit modes of operation according to an embodiment of the present invention.

Figure 3 is a schematic of a digital signal processor for either n-bit or (n/2)-bit modes of operation

providing interpolation functions according to an embodiment of the present invention.

Figure 4 a schematic of a digital signal processor including a split multiplier for  $(n/2)$ -bit operation  
5 according to an embodiment of the present invention.

Figure 5 a schematic of a digital signal processor for either  $n$ -bit or  $(n/2)$ -bit modes of operation according to an alternate embodiment of the present invention.

#### DETAILED DESCRIPTION

10 A general purpose digital signal processor (DSP) 100 is illustrated schematically in Figure 1. The DSP 100 includes multiplexers 115, 120, 150, 155, 160, a multiplier 135, flops 140, 145, 170, an arithmetic logic unit (ALU) 165, and an accumulator 175. The DSP 100 is designed for  
15 processing  $n$ -bit data, with  $n$  equal to 16 for the exemplary DSP 100 illustrated in Figure 1. It can be appreciated that the size of the data for which the DSP 100 operates is a matter of design choice. In addition, it should be understood that the scope of the present invention is not  
20 limited to DSPs with these elements.

The DSP 100 receives data in the form of 16-bit data 105, 110 from data buses. Each data bus provides a

plurality of 16-bit data sets 105, 110 to the DSP 100. The plurality of 16-bit data sets 105 is input into the multiplexer 115 and the plurality of 16-bit data sets is input into the multiplexer 120. The multiplexer 115  
5 selects a single 16-bit data set 125 from the plurality of 16-bit data sets 105. The multiplexer 120 selects a single 16-bit data set 130 from the plurality of 16-bit data sets 110. The selected 16-bit data sets 125, 130 are processed by the DSP 100.

10           The multiplier 135 receives the selected 16-bit data sets 125, 130 from the multiplexers 115, 120. The multiplier 135 may be configured to multiply two n-bit data sets. To ensure proper operation, the multiplier 135 may be at least  $2n$  bits in size. In the present exemplary  
15 embodiment, the multiplier 135 may be at least 32 bits in size to allow multiplication of two 16-bit numbers. Of course, the multiplier 135 may multiply two numbers of any size up to and including 16-bit numbers. However, if the multiplier 135 multiplies numbers smaller than 16 bits, a  
20 portion of the multiplier 135 may remain unused. Because of the unused portion of the multiplier 135, the DSP 100 may not be operating at desired efficiency.

In this embodiment, the result of the multiplication operation is transferred to the flops 140, 145. The flops 140, 145 are memory elements that utilize electronic latching circuits. The flops 140, 145 contain partial products from the multiplication process. The combination of the partial products in the flops 140, 145 equals the result of the multipliers 135. The flops 140, 145 pass the partial products to the multiplexers 150, 155. The multiplexers 150, 155, and 160 select the appropriate data to pass to the ALU 165. The multiplexer 160 receives as an input the value of the accumulator 175. If the result of the multiplier 135 alone is desired, the multiplexer 160 would not pass the data to the ALU 165. The ALU 165 performs basic arithmetic and logical operations. In one embodiment, the ALU 165 is constructed of full adders. Full adders add three bits at a time, and produce results in the form of a sum and a carry. The ALU 165 takes the result from the multiplier 135 and adds that result to the previous value of the accumulator 175 stored in the multiplexer 160.

The output of the ALU 165 may be provided to a flop 170 and to the accumulator 175. The flop 170 has the result of the last value of the ALU 165. The accumulator

175 value represents a total of all of the previous results from the ALU 165. The most recent output of the ALU 165 is added to the accumulator 175, and the new accumulator value is then provided in a feedback loop to the multiplexer 160  
5 for possible inclusion in the next ALU 165 operation. The new accumulator value is also provided as an input to the multiplexer 180. The value of the flop 170 may also provided as an input to the multiplexer 180. The multiplexer 180 allows the DSP 100 to choose whether to  
10 output the value of the accumulator 175 or the most recent result from the ALU 165. Once selected, this data is sent to an output 185.

Figure 2 shows a schematic of a DSP 200 for either n-bit or  $(n/2)$ -bit modes of operation according to the  
15 present invention. The DSP 200 includes multiplexers 115, 120, 205, 210, 235, 240, 245, 250, 285, a multiplier 135, flops 140, 145, 265, 270, arithmetic logic units (ALUs) 225, 230, 255, 260, and accumulators 275, 280. The DSP 200 may be used for processing either n-bit data or  $(n/2)$ -bit  
20 data, with n equal to 16 for the exemplary DSP 200 illustrated in Figure 2. Thus, the DSP 200 may operate efficiently in either an 8-bit mode or a 16-bit mode. It can be appreciated that the size of the data for which the

DSP 200 operates is a matter of design choice and does not affect the scope of the present invention.

The DSP 200 receives data in the form of 16-bit data 105, 110 from data buses. Each data bus provides a plurality of 16-bit data sets 105, 110 to the DSP 200. The  
5 plurality of 16-bit data sets 105 is input into the multiplexer 115 and the plurality of 16-bit data sets is input into the multiplexer 120. The multiplexer 115 selects a single 16-bit data set 125 from the plurality of  
10 16-bit data sets 105. The multiplexer 120 selects a single 16-bit data set 130 from the plurality of 16-bit data sets 110. The selected 16-bit data sets 125, 130 are processed by the DSP 200.

The DSP 200 may process the selected 16-bit data sets 125, 130 in either 8-bit mode or 16-bit mode. The DSP  
15 200 includes multiplexers 205, 210, 235, 240 and ALUs 225, 230 which may operate in parallel with the multiplexer 135 and support the  $(n/2)$ -bit, or 8-bit, operation of the DSP 200. The multiplexers 205, 210 receive the selected 16-bit  
20 data sets 125, 130 from the multiplexers 115, 120. The multiplexer 205 selects the appropriate 16-bit data from the input data 125, 130 and outputs 8-bit data 215. The multiplexer 210 also selects the appropriate 16-bit data

sets from the input data 125, 130 and outputs 8-bit data 220. Of course, the original data from the data bus may have been 8-bit, in which case the 8-bit data is passed through the multiplexers 115, 120, 205, 210 to the ALUs  
5 225, 230.

The ALUs 225, 230 receive the 8-bit data 215, 220 and perform basic arithmetic and logical operations as directed by the DSP 200. The results of these operations are then output to the multiplexers 235, 240. The  
10 multiplexers 235, 240 may then provide the data to either the multiplexers 245, 250 or directly to the multiplexer 285. The multiplexers 235, 240 also allow for the selection of both the sum and difference functions, thereby allowing for processing of an absolute value function.

15 The multiplier 135 may be used if n-bit processing is desired. As stated above, the multiplier 135 is designed to multiply two n-bit numbers and is therefore at least  $2n$  bits in size. If the multiplier 135 multiplies numbers smaller than n bits, a portion of the multiplier  
20 135 may remain unused and the DSP may not operate at desired efficiency. Therefore, if  $(n/2)$ -bit numbers or smaller need to be processed, the ALUs 225, 230 may be used. This increases the efficiency of the DSP 200.

The result of the multiplication operation is transferred to the flops 140, 145. The flops 140, 145 contain partial products from the multiplication process. The combination of the partial products in the flops 140, 145 equals the result of the multipliers 135. The flops 140, 145 pass the partial products to the multiplexers 245, 250. The multiplexers 235, 240 also pass the result of the mathematical operations from the ALUs 225, 230 to the multiplexers 245, 250. Depending on the operation mode of the DSP 200, the multiplexers 245, 250 select the appropriate data to pass to the ALUs 255, 260. Two  $(n/2)$ -bit ALUs 255, 260 are used in place of the one  $n$ -bit ALU in the 16-bit DSP 100 of Figure 1. The ALU 255 takes the data from the multiplexer 245 and adds that result to the previous value of the accumulator 275. The ALU 260 takes the data from the multiplexer 250 and adds that result to the previous value of the accumulator 280.

The output of the ALU 255 may be provided to the flop 265 and to the accumulator 275. The flop 265 simply contains the result of the last value of the ALU 255. The accumulator 275 value represents a total of all of the previous results from the ALU 255. The most recent output of the ALU 255 may then be added to the accumulator 275.

The new accumulator value is then provided in a feedback loop back to the ALU 255. The new accumulator value may also be provided as an input to the multiplexer 285. The value of the flop 265 may also be provided as an input to  
5 the multiplexer 285.

In a similar manner, the output of the ALU 260 may be provided to the flop 265 and to the accumulator 280. The flop 270 always contains the most recent value of the ALU 260. The accumulator 280 value represents a total of  
10 all of the previous results from the ALU 260. The most recent output of the ALU 260 may then be added to the accumulator 280. The new accumulator value may then be provided in a feedback loop back to the ALU 260. The new accumulator value may also be provided as an input to the  
15 multiplexer 285. The value of the flop 270 may also be provided as an input to the multiplexer 285.

The multiplexer 285 allows the DSP 200 to choose which value should be output by the DSP 200. Once selected, this data is sent to an output 290.

20 Figure 3 is a schematic of a DSP 300 for either n-bit or  $(n/2)$ -bit modes of operation providing interpolation functions. The DSP 300 combines the n-bit or

( $n/2$ )-bit multiplication stage of the DSP 200 of Figure 2 with the single ALU 165 of the DSP 100 of Figure 1.

Figure 4 shows a schematic of a DSP 400 including a split multiplier for ( $n/2$ )-bit operation according to an embodiment of the present invention. The DSP 400 receives data in the form of 16-bit data sets 105, 110 from data buses (not shown). Each data bus provides a plurality of 16-bit data sets 105, 110 to the DSP 400. The plurality of 16-bit data sets 105 are input into the multiplexer 115 and the plurality of 16-bit data sets 110 are input into the multiplexer 120. The multiplexer 115 selects a single 16-bit data set 125 from the plurality of 16-bit data set 105. The multiplexer 120 selects a single 16-bit data set 130 from the plurality of 16-bit data set 110.

The multiplexer 405 receives the data 125 and outputs multiple 8-bit data strings 415, 420. The 8-bit data strings 415, 420 are received as inputs by the multiplier 435. The multiplexer 410 receives the data 130 and outputs multiple 8-bit data strings 425, 430. The 8-bit data strings 425, 430 are received as inputs by the multiplier 440. Of course, the original data from the data bus may have been 8-bit, in which case the 8-bit data is

simply passed through the multiplexers 115, 120, 405, 410 to the multipliers 435, 440.

The multiplier 435 receives the selected 8-bit data 415, 420 from the multiplexer 405. The multiplier 440  
5 receives the selected 8-bit data 425, 430 from the multiplexer 410. The multipliers 435, 440 are each designed to multiply two  $(n/2)$ -bit data strings. To ensure proper operation, the multipliers 435, 440 are at least  $n$  bits in size. The multipliers 435, 440 may be embodied  
10 using portions of the larger multiplier 135. While the larger multiplier 135 may be used to multiply numbers smaller than 16 bits, a portion of the multiplier 135 may remain unused and not operate at desired efficiency. By dividing the multiplier 135 into smaller components, or  
15 simply using two smaller multipliers 435, 440,  $(n/2)$ -bit data strings may be processed in a more efficient manner.

The results of the multiplication operations are transferred to the flops 140, 145. The flops 140, 145 pass the results to the multiplexers 245, 250. The remainder of  
20 the DSP 400 may operate as described above with reference to Figure 2. The output 445 of the DSP 400 may be the result of the flop 265, the accumulator 275, the accumulator 280, or the flop 270.

Figure 5 a schematic of a digital signal processor for either n-bit or (n/2)-bit modes of operation according to an alternate embodiment of the present invention. The DSP 500 in Figure 5 is a modified version of the DSP 200 of Figure 2. In Figure 5, a multiplexer 505 is placed between the multiplexer 235 and the flop 140. A second multiplexer 510 is placed between the multiplexer 240 and the flop 145. The multiplexers 505 and 510 allow selection of the input to provide to the flops 140, 145. The DSP 300 of Figure 3 can be similarly modified.

The DSP according to an embodiment of the present invention may be used in place of an ASIC in devices requiring digital processing. Some examples include digital video cameras, computers, cellular telephones, and personal digital assistants. For example, the DSP of according to one embodiment of the invention may be used in a mobile video communicator with Internet access. The DSP may perform the calculations necessary to process the video data.

Numerous variations and modifications of the invention will become readily apparent to those skilled in the art. Accordingly, the invention may be embodied in other specific forms without departing from its spirit or essential

characteristics. The detailed embodiment is to be considered in all respects only as illustrative and not restrictive and the scope of the invention is, therefore, indicated by the appended claims rather than by the foregoing description.

- 5 All changes which come within the meaning and range of equivalency of the claims are to be embraced within their scope. For example, while the disclosure describes division based on  $(n/2)$  data, the same techniques could be applied for any separation, e.g.  $(n/3)$  ,  $(n/4)$ , etc. Generally, the
- 10 division is into  $(n/m)$  parts.

**WHAT IS CLAIMED IS:**

1. An integrated circuit comprising:

a first processing path which processes data up to  $n$ -bits; and

5 a second processing path which processes data up to  $(n/2)$ -bits.

2. The digital signal processor of Claim 1, wherein the first processing path and the second processing path operate in parallel.

10 3. The digital signal processor of Claim 1, wherein the first processing path includes a multiplier.

4. The digital signal processor of Claim 1, wherein the second processing path includes a plurality of arithmetic logic units.

15 5. The digital signal processor of Claim 1, wherein the first processing path processes data up to 16 bits.

6. The digital signal processor of Claim 5, wherein the second processing path processes data up to 8 bits.

7. The digital signal processor of Claim 1, wherein the first processing path comprises an n-bit multiplier and the second processing path comprises a plurality of  $(n/2)$ -bit arithmetic logic units.

5 8. The digital signal processor of Claim 1, wherein data of  $(n/2)$ -bits or less is processed in the second processing path.

9. A method of processing data comprising:

determining a size of the data to be processed; and

10 selecting a first processing path if the data size is greater than a first size; and

selecting a second processing path if the data size is less than a first size.

10. The method of Claim 9, further comprising storing  
15 a result from the first processing path in a flop.

11. The method of Claim 9, further comprising storing a result from the second processing path in a flop.

12. The method of Claim 10, further comprising adding the result from the first processing path to a running total in at least one arithmetic logic unit.

13. The method of Claim 11, further comprising adding  
5 the result from the second processing path to a running total in at least one arithmetic logic unit.

14. The method of Claim 9, wherein the first processing path is used for approximately  $n$ -bit data and the second processing path is used for approximately  $(n/2)$ -  
10 bit data.

15. The method of Claim 14, further comprising operating the first processing path and the second processing path in parallel.

16. The method of Claim 14, further comprising  
15 reconfiguring the first processing path to create the second processing path.

17. The method of Claim 16, further comprising:  
  
collecting the processed data from the first processing path and the second processing path at an  
20 output; and

selecting the appropriate data to output.

18. A system comprising:

a bus; and

a digital signal processor comprising:

5           a multiplier having a first structure and a  
second structure, the first structure processing data  
up to  $n$ -bits and the second structure processing data  
up to  $(n/2)$ -bits; and

          a data size selector which configures the  
10       multiplier into the first structure when the data is  
greater than  $(n/2)$ -bits and configures the multiplier  
into the second structure when the data is  $(n/2)$ -bits  
or less.

19. The system of Claim 18, wherein the first  
15       structure is a single  $n$ -bit multiplier.

20. The system of Claim 18, wherein the second  
structure includes two  $(n/2)$ -bit multipliers.

21. The system of Claim 18, further comprising a plurality of arithmetic logic units to collect the processed data.

22. The system of Claim 18, further comprising a flop  
5 which stores the result of the multiplier.

23. The system of Claim 22, further comprising at least one arithmetic logic unit which adds the result from the multiplier to a running total.

24. A method comprising:  
10 determining a size of data to be processed;  
configuring a first processing path for data of  $n$ -bits if the data size is greater than  $(n/m)$ -bits; and  
dividing the first processing path into multiple processing paths if the data size is  $(n/m)$ -bits or less.

15 25. The method of Claim 24, further comprising configuring each of the multiple processing paths for data sizes smaller than the first processing path.

26. The method of Claim 24, further comprising dividing the first processing path into  $m$  processing paths.

27. The method of Claim 24, further comprising including an n-bit multiplier in the first processing path.

28. The method of Claim 24, further comprising defining  $m=2$ .