EFFICIENT EMULATION DISPATCH BASED ON INSTRUCTION WIDTH

TECHNICAL FIELD

This invention relates to digital signal processors, and more particularly to controlling multiple instructions received from the emulation instruction register.

BACKGROUND

Digital signal processing is concerned with the representation of signals in digital form and the 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 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, DSP architecture and design have evolved to the point where

even sophisticated real-time processing of video-rate sequences can be performed.

DSPs are often used for a variety of multimedia applications such as digital video, imaging, and audio. DSPs can manipulate the digital signals to create and open such multimedia files.

MPEG-1 (Motion Picture Expert Group), MPEG-2, MPEG-4 and H.263 are digital video compression standards and file formats. These standards achieve a high compression rate of the digital video signals by storing mostly changes from one video frame to another, instead of storing each entire frame. The video information may then be further compressed using a number of different techniques.

The DSP may be used to perform various operations on the video information during compression. These operations may include motion search and spatial interpolation algorithms. The primary intention is to measure distortion between blocks within adjacent frames. These operations are computationally intensive and may require high data throughput.

The MPEG family of standards is evolving to keep pace with the increasing bandwidth requirements of

multimedia applications and files. Each new version of the standard presents more sophisticated algorithms that place even greater processing requirements on the DSPs used in MPEG compliant video processing equipment.

Video processing equipment manufacturers often rely on application-specific integrated circuits (ASICs) customized for video encoding under the MPEG and H.263 standards. However, ASICs are complex to design, costly to produce and less flexible in their application than general-purpose DSPs.

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.

Figure 1 is a block diagram of a mobile video device utilizing a processor according to one embodiment of the present invention.

Figure 2 is a block diagram of a signal processing system according to an embodiment of the present invention.

Figure 3 is a block diagram of an alternative signal processing system according to an embodiment of the present invention.

Figure 4 illustrates exemplary pipeline stages of the processor in Figure 1 according to an embodiment of the present invention.

Figure 5 is a block diagram of a emulation system according to one embodiment of the present invention.

Figure 6 illustrates the process of receiving and executing multiple instructions from the emulation instruction register according to one embodiment of the present invention.

DETAILED DESCRIPTION

Figure 1 illustrates a mobile video device 100 including a processor according to an embodiment of the invention. The mobile video device 100 may be a hand-held device which displays video images produced from an encoded video signal received from an antenna 105 or a digital video storage medium 120, e.g., a digital video disc (DVD) or a memory card. A processor 110 may communicate with a cache memory 115 which may store instructions and data for the processor operations. The processor 110 may be a

microprocessor, a digital signal processor (DSP), a microprocessor controlling a slave DSP, or a processor with an hybrid microprocessor/DSP architecture. For the purposes of this application, the processor 110 will be referred to hereinafter as a DSP 110.

The DSP 110 may perform various operations on the encoded video signal, including, for example, analog-to-digital conversion, demodulation, filtering, data recovery, and decoding. The DSP 110 may decode the compressed digital video signal according to one of various digital video compression standards such as the MPEG-family of standards and the H.263 standard. The decoded video signal may then be input to a display driver 130 to produce the video image on a display 125.

Hand-held devices generally have limited power supplies. Also, video decoding operations are computationally intensive. Accordingly, a processor for use in such a device is advantageously a relatively high speed, low power device.

The DSP 110 may have a deeply pipelined, load/store architecture. By employing pipelining, the performance of the DSP may be enhanced relative to a non-pipelined DSP. Instead of fetching a first instruction,

executing the first instruction, and then fetching a second instruction, a pipelined DSP 110 fetches the second instruction concurrently with execution of the first instruction, thereby improving instruction throughput. Further, the clock cycle of a pipelined DSP may be shorter than that of a non-pipelined DSP, in which the instruction are fetched and executed in the same clock cycle.

Such a DSP 110 may be used in video camcorders, teleconferencing, PC video cards, and High-Definition Television (HDTV). In addition, the DSP 110 may also be used in connection with other technologies utilizing digital signal processing such as voice processing used in mobile telephony, speech recognition, and other applications.

Turning now to Figure 2, a block diagram of a signal processing system 200 including DSP 110 according to an embodiment is shown. One or more analog signals may be provided by an external source, e.g., antenna 105, to a signal conditioner 202. Signal conditioner 202 is configured to perform certain preprocessing functions upon the analog signals. Exemplary preprocessing functions may include mixing several of the analog signals together, filtering, amplifying, etc. An analog-to-digital converter

(ADC) 204 is coupled to receive the preprocessed analog signals from signal conditioner 202 and to convert the preprocessed analog signals to digital signals consisting of samples, as described above. The samples are taken according to a sampling rate determined by the nature of the analog signals received by signal conditioner 202. The DSP 110 is coupled to receive digital signals at the output of the ADC 204. The DSP 110 performs the desired signal transformation upon the received digital signals, producing one or more output digital signals. A digital-to-analog converter (DAC) 206 is coupled to receive the output digital signals from the DSP 110. The DAC 206 converts the output digital signals into output analog signals. The output analog signals are then conveyed to another signal conditioner 208. The signal conditioner 208 performs post-processing functions upon the output analog signals. Exemplary post-processing functions are similar to the preprocessing functions listed above. It is noted that various embodiments of the signal conditioners 202 and 208, the ADC 204, and the DAC 206 are well known. Any suitable embodiment of these devices may be coupled into a signal processing system 200 with the DSP 110.

Turning next to Figure 3, a signal processing system 300 according to another embodiment is shown. In this embodiment, a digital receiver 302 may receive one or more digital signals and to convey the received digital signals to the DSP 110. As with the embodiment shown in Figure 2, DSP 110 performs the desired signal transformation upon the received digital signals to produce one or more output digital signals. Coupled to receive the output digital signals is a digital signal transmitter 304. In one exemplary application, the signal processing system 300 is a digital audio device in which the digital receiver 302 conveys to the DSP 110 digital signals indicative of data stored on the digital storage device 120. The DSP 110 then processes the digital signals and conveys the resulting output digital signals to the digital transmitter 304. The digital transmitter 304 then causes values of the output digital signals to be transmitted to the display driver 130 to produce a video image on the display 125.

The pipeline illustrated in Figure 4 includes eight stages, which may include instruction fetch 402-403, decode 404, address calculation 405, execution 406-408, and write-back 409 stages. An instruction i may be fetched in one clock cycle and then operated on and executed in the

pipeline in subsequent clock cycles concurrently with the fetching of new instructions, e.g., i+1 and i+2.

Pipelining may introduce additional coordination problems and hazards to processor performance. Jumps in the program flow may create empty slots, or "bubbles," in the pipeline. Situations which cause a conditional branch to be taken or an exception or interrupt to be generated may alter the sequential flow of instructions. After such an occurrence, an new instruction may be fetched outside of the sequential program flow, making the remaining instructions in the pipeline irrelevant. Methods such as data forwarding, branch prediction, and associating valid bits with instruction addresses in the pipeline may be employed to deal with these complexities.

Figure 5 is a block diagram illustrating an emulation system 500 according to one embodiment of the present invention. The emulation system 500 includes the connection of an in-circuit-emulator (ICE) 502 to the DSP 110 through a JTAG (Joint Test Action Group) system 504. In-circuit-emulation is a system which includes a peripheral device referred to as an in-circuit-emulator (ICE) 502 that is embedded within a target processor system which is implicitly aware of the target processor's operations and

can generate real-time trace information for reconstructing processor execution in an external host emulator. The ICE 502 may control the processor and monitor and modify the state of the registers within the processor. An ICE 502 may include its own ICE bus, separate from normal data, address or control busses found on the processor integrated circuit, so as not to interfere with processor behavior while the ICE 502 generates trace information.

Emulation may be performed during procedures such as debugging, hardware development, or software development using a JTAG system 504 as defined by the standard specified by IEEE 1149.1. Instructions that are to be executed during emulation may be scanned in from the JTAG system 504 to the emulation instruction register (EMUIR) 505. The instructions may be scanned serially from the ICE 500 to the JTAG system 504. After the JTAG system 504 is loaded from the ICE 500, the JTAG system 504 loads either of the instruction registers 515, 520 in the EMIUR 505 in parallel. For example, a first 64-bit instruction may be loaded from the ICE 500 to the first instruction register 515 and a second 64-bit instruction may be loaded from the ICE 500 to the second instruction register 520. Of course, each of the 64-bit instructions may include a single

instruction, or a plurality of instructions. For example, the 64-bit instructions may include a 32-bit instruction and 2 parallel 16-bit instructions.

The first 64-bit instruction may be loaded serially into the JTAG system 504 in 64 clock cycles and the second 64-bit instruction may be loaded serially into the JTAG system 504 in an additional 64 clock cycles. However, after the JTAG system 504 is loaded with the first and second instruction, the JTAG system 504 may load in parallel the first 64-bit instruction into the first instruction register 515 in a single clock cycle. The JTAG system 504 may also load in parallel the second 64-bit instruction into the second instruction register 520 in a single clock cycle. The first instruction and the second instruction also remain in the JTAG system 504 so they may be re-loaded into the EMUIR 505 if necessary.

After the instructions are loaded into the instruction registers 515, 520, the JTAG system may enter a run-test idle (RTI) state to tell the instructions to go down the pipeline. After entering the RTI state, the first instruction goes down the pipeline to write-back, and then the second instruction goes down the pipeline to write-back. After the second instruction reaches write-back, the

JTAG system 504 waits for the next instruction. If the ICE 502 is to repeat the first instruction or the second instruction, the instructions do not need to be reloaded into the JTAG system 504. Thus, the JTAG system may load the instructions in parallel into the instruction registers 515, 520 in a single clock cycle for each instruction. When the first or second instructions are repeated, the clock cycles necessary to load the instructions into the JTAG system 504 are saved.

The RTI state allows certain operations to occur depending on the current instruction. Entering the RTI state consumes a clock cycle, and thus, slows down the emulation of the DSP 110. By allowing the emulation instruction register 505 to provide multiple instructions in a single cycle, the DSP 110 may not need an RTI after every instruction is executed, thus saving time.

The emulation system 500 according to one embodiment of the present invention also includes emulation control logic 522, a state machine 523, a multiplexer 525, a register 527, and a decoder 530. The emulation control logic 522 includes the state machine 523 and provides control signals to the instruction registers 515, 520, the multiplexers 525, and the register 527. The control

signals from the emulation control logic controls the flow of the instructions in the pipeline as illustrated in Figure 4. In one embodiment, the emulation instruction register is a 128-bit instruction register 510, which includes a plurality of smaller instruction registers such as the 64-bit first and second instruction registers 515, 520. Typically, the instruction registers 515, 520 may supply one instruction at a time, with the instruction being up to 64-bits in length. However, according to one embodiment of the present invention, multiple instructions may be supplied simultaneously from the 64-bit instruction registers 515, 520. As shown in Figure 5, the first instruction 515 and the second instruction 520 may be loaded in the 64-bit instruction register. Of course, the size of the first instruction 515 and the second instruction 520 must not exceed 64-bits. Thus, the first instruction 515 may be a 32-bit instruction and the second instruction 520 may be a 32-bit instruction. The first and second instructions 515, 520 may also be 16-bit or other size, provided the combined size of the instructions fit into the 64-bit instruction register 510.

The emulation instruction register 505 provides the contents of the instruction registers 515, 520 to the

multiplexer 525. Because the instruction registers 515, 520 may contain a plurality of instructions, the emulation control logic 522 may control the flow of the instructions received from the emulation instruction register 505. The emulation control logic 522 includes logic described below to supply the instructions to the decoder 530. The state machine 523 may determine if multiple instructions are present, and whether the instructions are valid. The state machine 523 may then provides these instructions to the decoder 530 via the register. This may provide the instructions to the decoder 530 while reducing the disruption to the decoder 530.

The present invention is described using a 64-bit instruction register providing multiple instructions of 32-bits or smaller. Of course, the invention may be accomplished on any size instruction register (N-bit) providing multiple instructions of (N/2)-bits or smaller.

The process 600 for processing instructions by the emulation control logic 522 is shown in Figure 6. The process 600 begins at a start block 605. Proceeding to block 610, the process 600 waits for an RTI to begin the flow of instructions. The RTI may come from the JTAG system 504. Proceeding to block 615, the process

determines whether an RTI is detected. If no RTI is detected, the process 600 proceeds along the NO branch back to block 610 to wait for the RTI. The process 600 remains in this loop until an RTI is detected.

Returning to block 615, once an RTI is detected, the process proceeds along the YES branch to block 620. In block 620, the validity of the first instruction is determined. An instruction may include a corresponding set of width bits defining the validity and size of the instruction. In one embodiment of the invention, the width bits are a 2-bit signal. With a 2-bit signal, there are 4 possible values for the 2-bit width signal. For example, width bits of 00 indicates the instruction is invalid, width bits of 01 indicates a 16-bit instruction, width bits of 10 indicates a 32-bit instruction, and width bits of 11 indicates a 64-bit instruction. By reading the width bits, the DSP 110 may determine both the validity and size of the instruction.

If the instruction is valid, the process 600 proceeds along the YES branch to block 625. In block 625, the first instruction flows down the pipeline until write-back for execution by the decoder 530. Following execution of the first instruction, the process 600 proceeds to block

630. Returning to block 620, if the instruction is invalid, the process 600 proceeds along the NO branch to block 630. A no-operation (NOP) instruction may also cause the instruction to be declared invalid. If a NOP instruction is detected, the process 600 may proceed without the NOP being processed in the pipeline. Therefore, the second instruction may be processed immediately.

In block 630, the second instruction is received by the DSP 110. Because the first and second instructions are stored in the emulation instruction register at the same time, the second instruction may be retrieved without having to enter another RTI state. The DSP 110 may have previously determined the size of the first instruction using the width bits. Based upon the size of the first instruction, the DSP 110 may determine the location in the emulation instruction register of the beginning of the second instruction. For example, if the first instruction was a 32-bit instruction, the DSP 110 may find the first bit of the second instruction at the 33rd bit of the emulation instruction register. The width bits of the second instruction may also provide information on how many bits to retrieve from the emulation instruction register.

Thus, the DSP 110 may be able to retrieve the second instruction from the emulation instruction register containing a plurality of instructions.

Proceeding to block 630, the validity of the second instruction is determined. The validity of the second instruction may also be determined by examination of the width bits as described above. If the instruction is valid, the process 600 proceeds along the YES branch to block 635. In block 635, the second instruction flows down the pipeline until write-back for execution by the decoder 530. Following execution of the second instruction, the process 600 proceeds to block 640. Returning to block 630, if the instruction is invalid, the process 600 proceeds along the NO branch to block 640.

In block 640, the process 600 determines whether the DSP 110 should exit the emulation mode. The determination to exit the emulation mode may be provided by the emulation control logic 522. If further emulation is indicated, the process proceeds along the NO branch back to block 610 to wait for the next RTI. Returning to block 640, if the emulation control logic 522 provides instructions to exit the emulation mode, the process 600 proceeds along the YES branch to an end block 645.

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.

WHAT IS CLAIMED IS:

1. A method of providing instructions to a processor from an emulation instruction register comprising:

receiving a plurality of instructions simultaneously from the emulation instruction register;

determining the validity of a first instruction of the plurality of instructions;

providing the first instruction of the plurality of instructions to a decoder if the first instruction is valid;

determining the validity of a second instruction of the plurality of instructions; and

providing the second instruction of the plurality of instructions to the decoder if the second instruction is valid.

2. The method of Claim 1, further comprising determining the size of the plurality of instructions.

3. The method of Claim 1, further comprising storing the plurality of instructions in a single instruction register.

4. The method of Claim 1, further comprising loading the second instruction of the plurality of instructions after determining the first instruction is invalid.

5. The method of Claim 1, further comprising providing the plurality of instructions to the decoder during one RTI.

6. The method of Claim 1, further comprising loading the plurality of instructions in parallel into the emulation instruction register.

7. The method of Claim 1, further comprising providing instructions to a digital signal processor.

8. A method of processing instructions within a processor comprising:

loading a plurality of instructions into a single instruction register;

receiving an RTI;

simultaneously providing the plurality of instructions to the processor; and

processing the plurality of instructions.

9. The method of Claim 8, further comprising loading the plurality of instruction into an N-bit emulation instruction register.

10. The method of Claim 8, further comprising determining the validity of each of the plurality of instructions before processing.

11. The method of Claim 10, further comprising aborting the processing of any invalid instructions and loading a next instruction of the plurality of instructions.

12. The method of Claim 8, further comprising loading a next instruction of the plurality of instructions if a no-operation instruction is loaded.

13. The method of Claim 8, further comprising providing the plurality of instructions to a digital signal processor.

14. A processor comprising:

an instruction register adapted to store a plurality of instructions;

emulation control logic adapted to control the flow of the plurality of instructions to a processor pipeline following detection of a single RTI; and

a decoder which may receive the plurality of instructions for processing.

15. The processor of Claim 14, wherein the instruction register is an emulation instruction register.

16. The processor of Claim 14, wherein the control logic determines the validity of the plurality of instructions and discards any invalid instructions.

17. The processor of Claim 14, wherein the control logic loads a next instruction immediately after detecting a no-operation instruction.

18. The processor of Claim 14, wherein the processor is a digital signal processor.

19. An apparatus, including instructions residing on a machine-readable storage medium, for use in a machine system to handle a plurality of instructions, the instructions causing the machine to:

load the plurality of instructions into a single instruction register;

receive and RTI;

provide the plurality of instructions to the processor; and

process the plurality of instructions.

20. The apparatus of Claim 19, wherein the instruction register is an emulation instruction register.

21. The apparatus of Claim 19, wherein the validity of each of the plurality of instructions is determined before processing.