New Pipelined Architecture for DSP

Jean Noel* and Vojin G. Oklobdzija
Department of Electrical and Computer Engineering
University of California
Davis, CA 95616

Abstract

DSP's have been applied to digital filtering, modems, transmultiplexers, speech processing, high-fidelity audio, and graphics. The new DSP is in the family of the scalar fixed point Digital Signal Processor. The type of instructions are similar to the RISC instructions. The best aspecs of the DSP will be explained ,including possible modifications in order to maximize performance of DSP. This paper compares the features found in new architectures, pointing to areas of possible improvement in new DSPs introduced by different research laboratories in the world. The architectural innovations and concepts as related to the pipeline structure and its benefits are described.

1 Improvements in the new DSP

1.1 Instructions

The new DSP has the same instructions as the RISC. Therefore it can fully support parallelism and overlapped arithmetic and memory access. The RISC architecture opens the possibilities for parallel operations including conditional execution.

1.2 32-bit fixed point

The new length with 32-bit fixed point is the right idea for a fixed point because the accuracy is always needed in calculation operations. The 32-bit data size instead of 16-bit is advantageous for not having to use double-precision when accuracy is important. This allows a lot of cycles to be saved. The DSP uses the processor power fully with the 32-bit accuracy. For example, for speech processing, the DSP calculates the result of an LPC (Linear Predictive Coding) or an auto-correlation of a speech signal without using the double-precision. This is the same case for calculing the spectrum of a speech signal. For power consumption, the double-precision is the worst case. Increasing of the number of bits (here 32 bits) is the correct solution.

1.3 Architecture

The real time implementation is very important for communication, such as the digital mobile telephone. In this case, the power consumption must be minimized. High speed systems are required for products like modems and digital equipment.

The new DSP consists of five majors blocks: execution unit, address unit, program control unit, decoder and interrupt controller. Data is loaded to the registers before it can be used by the right functional unit. In fact, the use of registers allows optimization of the pipeline. Therefore, the speed of operation is increased. The content of the memory is loaded to a register first and then passed to the arithmetic units in the following cycle.

The **Execution Unit** uses 72-bit register to increase the dynamic range with 32-bits for the base precision, 32-bits extended precision and 8-bit guard bit part. The multiplication of 32-bit x 32-bit takes only one cycle. The ALU has 72 bits. The **Address Unit** uses two address arithmetic units AUX and AUY. In the **Program Control Unit**, a loop counter, a repeat counter and repeat boundary registers are used.

The new DSP is based on a Harvard architecture with three buses: two for the data X BUS and Y BUS and one bus for the DMA. D BUS and PC BUS are used to transfer the instructions and P BUS for the access to the memory mapped periphical registers. For the external memory, the new DSP uses a Von Neuman structure.

1.4 Programming

The programming of the new DSP is in high-level language, the same as for a RISC processor using RISC type instructions. Versatility and high-speed are excellent qualities of the new DSP. The combination of a RISC processor and a DSP gives the DSP

*Ecole Supérieure d'Ingénieur en Electrotechnique et Electronique, 93162 Noisy le Grand CEDEX FRANCE

1058-6393/96 $5.00 © 1996 IEEE
Proceedings of ASILOMAR-29

great possibilities, but more research can ameliorate the structure of the new DSP.

2 New Architectures for DSP

2.1 Memory configuration

The Harvard architecture ...

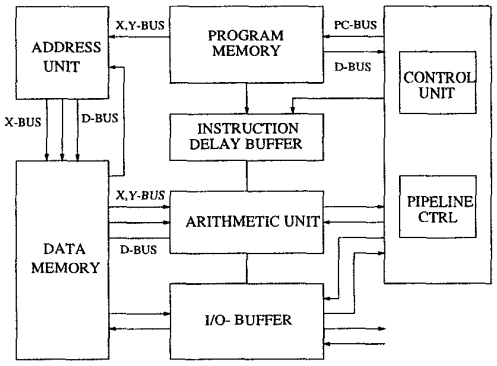

Figure 1: Harvard architecture.

2.2 Device coding

This part explains the two main codings: **time stationary** and **data stationary**. The first coding means that each line of code specifies what each piece of hardware does in one instruction cycle. The second one is used in the more advanced Bell Labs device, the WE DSP32[4], where a multiply, accumulate, and write instructions are specified in one line, even though it takes several cycles to execute. Then there are only two load data registers instead of eight as for RISC instructions for each bus. The immediate displacement is not available for data load. It is useful to increase this number to allow for data independence and to avoid pipeline hazards.

2.3 Pipeline

This part introduces different structures for the optimization of DSP pipeline. In fact, there are two major problems with pipelined architecture. First, when a branch occurs and second when an interrupt breaks the pipeline. Data dependencies and branch instructions are two major impediments to performance. Pipeline hazards decrease the calculation power of the DSP. However, several methods exist that can aleviate the branch penalty.

First, the **static prediction** method predict the branch decisions. The compiler uses different data sets and calculates the branch percentage for each branch instruction. By simulating this technique, the average branch accuracy of 83 percent can be achieved [5].

Dynamic prediction is one solution that can resolve this problem. This means that for each branch instruction in the program, the hardware will increase or decrease the value of the prediction whether the predict branch is taken or not. The new DSP uses this mechanisnm to decrease the branch penalty.

On the other hand, the **delayed branch** method has been used successfully on several machines (RISC, IBM 801 and MIPS) with only one-cycle branch delay.

One other solution, for preventing pipeline interruption, is to share it with different programs or tasks. In this case, there is no dependency between the instructions, and the branch cost is reduced. The hardware has enough cycles to fetch the instructions in the subroutine. For example, with a 5-stage pipeline, independence in instructions is the best method when the pipeline is shared between 5 tasks.

Branch bypass and multiple prefetch are used to eliminate the need to predict which path will be taken after a branch. The DSP can fetch both paths and throw away the incorrect one when the branch is resolved. The simulation shows that the performance increases proportional to $\sqrt{j}$, where j levels of unresolved branches exist. However, this technique is very hardware consuming.

3 Research

The purpose of this research is to increase the calculation power of a DSP through a new structure for the pipeline. The pipeline structure of the model architecture used in the research is a modified version of the popular load/store reduced instruction set computer, called DLX [2], utilizing a fixed point structure.

3.1 Architectural model

To describe the various models, model architecture has been chosen with the following features:

- a load/store reduced instruction set;
- an easily decoded instruction set having total of 15 instructions;
- four-stage pipeline: instruction fetch, instruction decode/register fetch, execution/memory access, and write back;
- register file with sixteen 32-bit registers;
- 16-bit address bus and 32-bit data bus;

- three non-pipelined functional units: arithmetic logic, complex multiply/divide, and memory access;

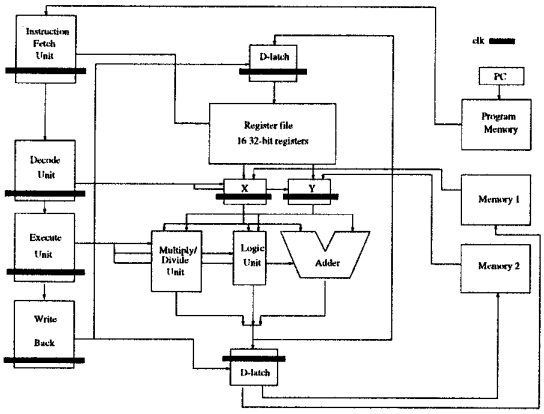

Figure 2: Block diagram of a pipelined architecture with multiple functional units.

3.2 DSP instruction set

The DSP instructions use complex representation to make calculation directly with complex data. A lot of calculations need to use complex data for example calculating the spectrum of a speech signal with FFT.

The result of the use of instructions with complex calculation mode is to optimize the number of calculations, for exemple: the calculation for a spectrum of a voice signal with the FFT. The FFT uses a large number of calculations with complex numbers, so the complex mode for the calculation permits a decrease in the numbers of time-cycle. The speed of the DSP is increased.

3.3 The Tomasulo's Algorithm

Tomasulo's algorithm [9] used by IBM in 1967 demonstrated how to resolve data dependencies (or out-of-order instruction issue) with multiple functional units. The **reservation station (RS)** could be associated to optimize the problem of functional units allocations. An extension to the Tomasulo's algorithm is the **Register Renaming Unit (RRU)** [11] [12]. This is a hardware mechanism that resolves dependencies dynamically and, at the same time, guarantees precise interrupts.

3.4 Register Renaming Unit

Most of the cases, registers can not be overwritten until all prior instructions which reference the old value of the register have accessed that value. For a fixed point format DSP it is possible to implement a new version of the algorithm invented by Tomasulo for the IBM System/360 Model 91 [11] : Register Renaming Unit (RRU).

The organization that was adopted is illustrated in Fig 3. R0 and R1 are the rename registers. They contain an opcode field, a target-register field, and two source-register fields. The map table contains the correspondence of an architectural register to a physical register.

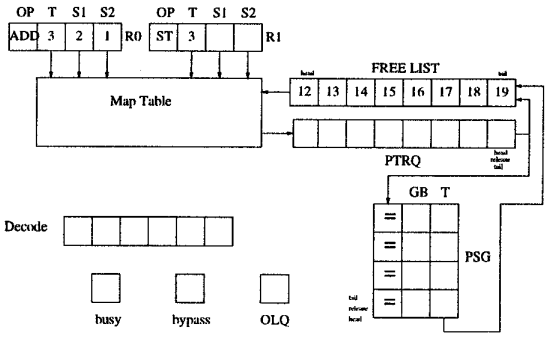

Figure 3: Register-renaming structure

The free list (Fl) contains a list of currently unassigned physical registers. In the initial state is initialized to identify and the renaming registers are placed on the free list. Since there are 20 physical registers, the FL can contain a maximum of eight entries. The Fl is maintain as a circular queue and uses a head pointer and a tail pointer.

The pending-target return queue (PTRQ) contains those physical registers which are being used by instructions in the decode phases, and will become free as soon as these instructions pass decode. It also has a maximum size of 8. Like the FL, it is maintained as a circular queue with head and tail pointers. It also has an additional pointer. The release pointer keeps register tags on the PTRQ until all prior arithmetic instructions which could have required the data in the corresponding physical register have decoded.

The BUSY and BYPASS registers contain the physical register number of the instruction currently in the first and second execution stages. If any register field of an instruction in decode compares with the BUSY register, it is prevented from decoding. If a source field compares with the BYPASS register, the data is read from the execution pipeline and not from the register file.

Original stream		Rename table	Free head	Renamed stream	PTRQ
ADD	R1 , R2 –>R3	(1,1), (2,2), (3,3)	12	R1, R2, R3	
ST	R3	(3,3)	12	R3	
LD	R3	(3,3)	12	PR12	3
MUL	R1 , R3 –>R6	(1,1), (3,12), (6,6)	13	R1, R12, R6	
SUB	R2 , R6 –>R2	(2,2), (6,6), (2,2)	13	R2, R6, R2	
LD	R3	(3,12)	13	PR13	12

Table 1: Register renaming.

The outstanding load queue (OLQ) contains the physical register number of the next load whose data will return from the cache. It stops instructions from decoding if they require data which has nit returned from the data cache.

4 Conclusion

The future DSP must calculate with a high speed while resolving the problems of dependency dynamically. In order to optimize the different buses and functional units, some research laboratories are currently attempting to design a DSP with a reconfigurable architecture. The time required to reconfigur the DSP is a problem. Those researchers have to develop a DSP with a high capacity of reconfiguration. The high performance DSP processor in the future will be able to change the hardware architecture for each specific application.

References

[1] David J. Lilja, "Reducing the Branch Penalty in Pipelined Processors," in *IEEE*, pp. 47-55, July 1988.

[2] Chia-Jiu Wang, Frank Emmett, "Implementing Precise Interruptions in Pipelined RISC Processors," in *IEEE Micro*, pp. 36-43, August 1993.

[3] Rolf Ernst, "Long Pipelines in Single-Chip Digital Signal Processors– Concepts and Case Study," in *IEEE Transactions on Circuits and Systems*, vol. 38, pp. 100-108, January 1991.

[4] Edward A. Lee, David G. Messerschmitt, "Pipeline Interleaved Programmable DSP's: Architecture," in *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 35, September 1987.

[5] Scott McFarling, John Hennessy, "Reducing the Cost of Branches," in *IEEE*, 1986.

[6] Gurindar S. Sohi, "Instruction Issue Logic for High-Performance, Interruptible, Multiple Functional Unit, Pipelined Computers," in *IEEE Transactions on Computers*, vol. 39, March 1990.

[7] G. F. Grohosky, "Machine Organization of the IBM Risc System/6000 processor," in *IBM Journal of Research and Development*, Vol.34, No1, pp.37, January 1990.

[8] J. Cocke, G. F. Grohosky, and V. G. Oklobdzija, "Instruction Control Mechanism for a Computing System with Register Renaming, MAP Table and Queues Indicating Available Registers," *US Patent No 4,992,938. Issued: February 12, 1991.*

[9] R.M. Tomasulo, "An efficient algorithm for exploiting multiple arithmetic units," *IBM J. Res. Develop.*, pp. 25-33, Jan. 1967.

[10] S. Welss and J.E.Smith, "Instruction issue logic in pipelined supercomputers," *IEEE Trans. Comput.*, vol. C-33, pp. 1013-1022, Nov. 1984.

[11] G. F. Grohosky, "Machine Organization of the IBM Risc System/6000 processor," in *IBM Journal of Research and Development*, Vol.34, No1, pp.37, January 1990.

[12] J. Cocke, G. F. Grohosky, and V. G. Oklobdzija, "Instruction Control Mechanism for a Computing System with Register Renaming, MAP Table and Queues Indicating Available Registers," *US Patent No 4,992,938. Issued: February 12, 1991.*