Global Journal of Computer Science and Technology: B
Cloud and Distributed
Volume 18 Issue 1 Version 1.0 Year 2018
Type: Double Blind Peer Reviewed International Research Journal
Publisher: Global Journals
Online ISSN: 0975-4172 | Print ISSN: 0975-4350 | DOI: 10.17406

An Evolution of Parallel Pipeline System with the Classification of Operation Code

By Dr. Narasimha Rao Yamarthi, Adinew Belay, Mr. Abdisa L & Mahesh N
Mizan Tepi University

Abstract- *Objectives:* Multiple pipelines are used to enhance the throughput. Multiple data are fetched at pipeline and classified into different data by selecting and decoding appropriate bits in Instruction field.

Methods/Statistical analysis: Enhancing the throughput is one of the important issue need to be considered in multitasking system. In the present system parallel pipeline is used to improve the data rate. The data rate further can be enhanced by doing classification before it feeding to pipeline. In the present system it is achieved by data classification which is done at each pipeline based upon mode selecting bits and operation codes.

Findings: The decoded data from any program is fetched and send simultaneously to arithmetic logic unit or any output device simultaneously for further processing. The data is classified and separately passed through the stages of pipeline. Based on the type of the data present in the machine code the opcode is classified and separated.

Keywords: *pipeline, parallel processing, parallel pipeline, instruction cycle, throughput, classification.*

GJCST-B Classification: *B.2.1, C.1.3*

Strictly as per the compliance and regulations of:

An Evolution of Parallel Pipeline System with the Classification of Operation Code

Dr. Narasimha Rao Yamarthi[α], Adinew Belay [σ], Mr. Abdisa L[ρ] & Mahesh N[ω]

Abstract- ***Objectives:*** Multiple pipelines are used to enhance the throughput. Multiple data are fetched at pipeline and classified into different data by selecting and decoding appropriate bits in Instruction field.

Methods/Statistical analysis: Enhancing the throughput is one of the important issue need to be considered in multitasking system. In the present system parallel pipeline is used to improve the data rate. The data rate further can be enhanced by doing classification before it feeding to pipeline. In the present system it is achieved by data classification which is done at each pipeline based upon mode selecting bits and operation codes.

Findings: The decoded data from any program is fetched and send simultaneously to arithmetic logic unit or any output device simultaneously for further processing. The data is classified and separately passed through the stages of pipeline. Based on the type of the data present in the machine code the opcode is classified and separated. The classified data is passed through the pipeline to optimize the data throughput. The design of the circuit is implemented in Proteus simulation software. It is observed that the multi-byte data or instructions are classified and feed through the pipeline. The time gap between two sequential bytes is drastically reduced and found they are almost sent side by side.

Application/Improvements: The present method can be used in digital systems and parallel processing systems where high speed data rates are required. The data rate further can be improved by increasing the number of stages in the pipeline.

Keywords: pipeline, parallel processing, parallel pipeline, instruction cycle, throughput, classification.

I. Introduction

In the recent days the parallel processing computing plays vital role in many applications like soft-computing, Cloud computing, Image processing, Industrial automation, Satellite communication, Robotics and many more. Different methodologies are present in implementing parallel processing at different instances and to meet different requirements of consumers. Parallel computing can be implemented through parallel processors where, multiple processors are used to perform multiple tasks in small time or same time. Similarly, parallel programming is used to process and execute multiple tasks to perform in the computing system. On the other hand parallel computing can also be done by integrating digital elements like pipeline in microprocessors or multiple stages of pipelines between computers. The present paper mainly concentrated on pipeline and their moderate use in the processor. In parallel computing the pipeline integration is already used for fastening of the operations in digital systems. There are many techniques effectively involved in improving pipeline performance in achieving high data speeds in digital systems[1]. In the present paper new technique is proposed along with the discussions of present flaws. In the present paper the challenges faced by parallel pipeline are reviewed and new methodology is proposed to improve the hit ratio and speed of data fetching through pipeline.

A novel Classification technique is proposed in pipeline designing in order to achieve high speed computing in various digital circuits like MAC, Arithmetic processors, Co-processor design, Peripheral Interface, measurements and etc. In most of the programs the instructions can be classified into groups according to their operations. if the instructions are classified in to categories like arithmetic, logic, load and etc., then it became very easy to the microprocessor to load the instructions into ALU and control unit parallelly. In the traditional pipeline methods the op-codes (operation codes) of the instructions are loaded into the execution unit through pipeline. Although the op-codes are fetched simultaneously through the pipeline, but they has to wait before they enter into execution unit. The present system can reduce this waiting time by selecting the data type and arithmetic type instructions simultaneously at execution unit. This can be implemented by checking the decoded bits at MSB of the operation code of the instruction.

The instruction pipeline functionality mainly depends on the MSB bits of the decoded instruction. Based on the decoded instructions the operations can be classified into groups according to the type of their operation. Before it is discussed intensely, first talk over about the instruction cycle carried through computer and its organization. After fetching the instruction from code memory, it is decoded into two essential parts as

Author α: Professor, Department of Computer Science, School of Computing & Informatics, Mizan Tepi University.
e-mail: narasimha.yamarthi@gmail.com
Author σ: HOD, Department of Computer Science, School of Computing & Informatics, Mizan Tepi University.
e-mail: adinewb2@gmail.com
Author ρ: Lecturer, Department of Computer Science, School of Computing & Informatics, Mizan Tepi University.
e-mail: abdisalechisa@gmail.com
Author ω: Assistant Professor, Department of IT, School of Computing & Informatics, Mizan Tepi University.
e-mail: mahesh781991@gmail.com

shown in figure 1. They are operation code part and the second part is the address of the operand specified in the instruction. In general, the first 12-bits (D0-D11) of the instruction binary code is address part which points to an operand and the last 4-bits (D12- D15) at MSB treated as operation code. These operands are fetched from the data segment of the memory. Based on the op-code the task will be completed on opcode. The super scalar architectures the instructions are executed simultaneously. Hence multiple instruction cycles are carried out in the control unit. The number of cycles here depends on the number of stages [2,3,4]. In this case, the instruction may contain immediate data, direct address, or indirect address. This phenomenon of transferring data from one type of source to any type of destination is called mode operation.

a) Control unit

The control unit initially fetches the address of the operand and fetches the operand from the memory and loaded into either in memory, register, or ALU. The loading of the operand in the respective destination is depends on 4-bit opcode of 16-bit instruction. Out of four bits the most significant bit (D15) represents the mode of addressing. The remaining three bits (d12-D14) are feed to control unit to produce control signals for decoding the instruction. The control signals are decoded to select specified operation such as immediate, register transfer, accumulator referenced operations and etc as shown in figure 2. The operands are fetched from the address specified in address field of the instruction and the specified operations are performed on the retrieved operands[5].

In figure 2 the *M* bit used to drive the decoder to select different addressing schemes such as direct or indirect. The operands are used to select immediate data. The opcode will drive the control unit to produce the control signals in order to activate the necessary circuit to perform the specified task. The control signals will select any circuit such as arithmetic circuits, logic circuits, memory references, or register references [5].

II. Pipeline

In traditional pipeline the address field and opcode are fetched in sequential clock cycles, and decoding in execution unit and fetching the next instruction has done simultaneously as shown in figure 3. In parallel pipeline simultaneously multiple instructions are fetched into pipeline and ecoding also done simultaneously[6,7] as shown in figure 4.The notation R_1, R_2,R_3,and R_4 are number of registers. In traditional pipeline, the registers are inter-connected with logical circuits to perform various arithmetic operations simultaneously and immediately after fetching the code from memory.

In parallel pipeline two linear pipelines are connected in parallel and synchronized with a clock pulse as shown in figure 4. The linear pipelines are active individual at alternative clock pulses. In the positive clock cycle the top one and in the negative clock pulse the bottom one will be active[8,9,10]. In this sense, the current work is motivated to enhance the performance of the parallel pipeline.

III. Proposed Method

The proposed method is motivated from the enhanced results obtained from parallel pipeline system [4,8,10]. In the proposed method multiple pipelines are used to access different types of data from memory. The pipeline is classified according to the type of data classified at organization of instruction. Hence separate pipelines are used for different types of instructions executed in the processor. The instructions are classified by combining M and opcode bit and the instructions are passed to separate pipelines.

I1: MOV R1,@#32

ADD R1

For instance in the above program the first instruction is decoded and identified as data instruction, and the second instruction is an arithmetic instruction which performs addition operation. While send the data to pipeline the device called bus controller will direct the instructions to different pipelines with respect to the type of machine code obtained from the program. The immediate code will be send to top pipeline which is meant only for data. Similarly the second instruction opcode will be sent to the next pipeline which is meant only for fetching the arithmetic instructions. The opcode part and operand part will differentiate the type of operations performed by the instructions. The devices such as decoders, bus controllers, and transceiver will reconfigure all the pipelines in the existing system. The pipelines will be reconfigured[11,12] by these devices every time the corresponding and respective data is identified. The proposed parallel pipeline system is shown in figure 5.

All the control signals in the proposed are narrated as micro operations in the following paragraph. In figure 5 the left decoder outputs are used to drive the bus controller signals. The control signals of bus controller are used to drive the decoder, clock management and transceiver. The clock management circuit is used to synchronize the stages of pipeline[13,14,15].The transceiver upon receiving micro signals from bus controller it controls the data transfer between register/memory to register/memory except memory to memory. Multiple units of register set are selected with the help of multiplexer (MUX). The control signals for MUX are provided by bus controller.

IV. Results

In the figure 6 the first wave form represent the clock signal obtained from clock management circuit. The second wave represents the output of the first pipeline and third wave represents the output of the second pipeline. The fourth waveform represents the output observed at the output of execution unit (EU).The instructions are classified with the help of decoders and bus controller. As per the control signals the classified data is served to different pipelines. For instance the load instruction opcode loaded into a pipeline, where the arithmetic operation code loaded into a different pipeline. While the second instruction processed in the second pipeline the first pipeline will be ready to accept the next micro operation. From the results it is observed the classification is effectively done by modifying the hardware architectural design.

V. Conclusion

The classification of instructions based on their opcode is done in the present work. The results are obtained from Proteus virtual design tool. Although the number of hardware components in the proposed system increased, when compared with the previous circuit the current system effectively involved in fetching the opcode as per the input opcode. Almost in the same clock pulse two data are entered into the parallel pipeline with small delay difference.

References Références Referencias

1. EBY G. Friedman, JH Mulligan. Pipeliing of high performance synchronous digital systems. INT J Electronics, 1991, 70(5), pp. 917-935.
2. Thomas gay C. Timing constraints for wave pipelined systems. IEEE transactions on Computer aided design of integrated circuits. 1994 August, 13 (8), pp. 987-1004.
3. Yin Yong-Sheng, DU Gao-Ming, SONG Yu-Kun. Study on the Multi-pipeline Reconfigurable Computing System. International Conference on Computer Science and Software Engineering. IEEE Computer Society, 2008. pp 122-125.
4. Narasimha Rao Y and Samuel Varaprasada Raju G. Enhancing Data Fetching Rates with Parallel Pipeline. International Journal of Computer Applications, 2014, 85(6), pp. 31-34.
5. David A. Patterson and John LHennessy. Computer Organization and Design. 3rd edition. Morgan Kaufmann Publishers, USA, by Elsevier, 2005.
6. Narasimha Rao Y, dr. Samuel Vara Prasada Raju G, & Penmetsa V Krishna Raja. ASIC Design, Implementation and Exploration on High Speed. Vol 14 (6), 2014.
7. Kumar NS, Rama KotiReddy DV, Pramod Kumar B, Chaitanya Prabhu. A. Two Way Clock Scheme in Pipeline to Minimize the Clock Skew. GJCST. 2010 sep, 10(9), pp 61-63.
8. Mehdi hatamian, Glenn cash L. A 70-MHz 8-bit X 8-bit Parallel Pipelined Multiplier in 2.5-pm CMOS.IEEEJournal of solid-state circuits. 1986 Aug, Vol.21, No.4, pp 505-513.
9. Eby G. Friedman. Clock Distribution Networks in Synchronous Digital Integrated Circuits. Invited paper, Proceedings of the IEEE, 2001 May, 89(5), pp 665.
10. Suresh K N, ramakoti reddy DV. Optimizing Technique to Improve the Speed of Data Throughput through Pipeline. INDIA 2016, Jan 2016, by Springer AISC series, Vol 435. Pp 237-246.
11. K. Compton, S. Hauck. Reconfigurable Computing: A Survey of Systems and Software. ACM Computing Surveys. February, 2002,vol. 34.pp. 171-210
12. Suresh Kumar, Reddy DVRK. Measurement system for wide range frequency at nonlinear devices. FCICS, Published by Taylor and Francis group. 2015. Pp 143-146.
13. Hwang, K. Adrnnced Computer Architecture with Pllrallel Programming, McGraw-Hill, New York, 1993.
14. Kozyrakis, c., and D. Patterson. Scalable vector processors for embedded systems. IEEE Micro 23:6(November- December), 2003, pp 36--45.
15. Kumar N S, Reddy DVRK. A New Method to Enhance Performance of Digital Frequency Measurement and Minimize the Clock Skew, IEEE Sensor J. 2011, 11(10), pp 2421-2425.

OP-Code		Addreses of the operand	
D15	D12	D11	D 0

Figure 1: Organization of Instruction

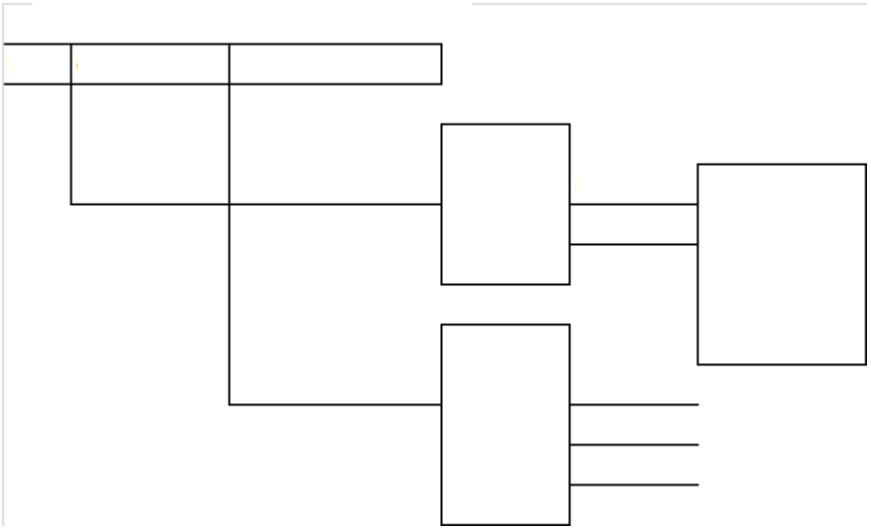

Figure 2: Determining of Addressing for operands

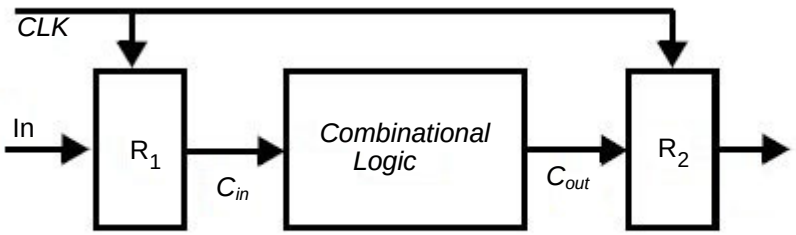

Figure 3: Traditional pipeline

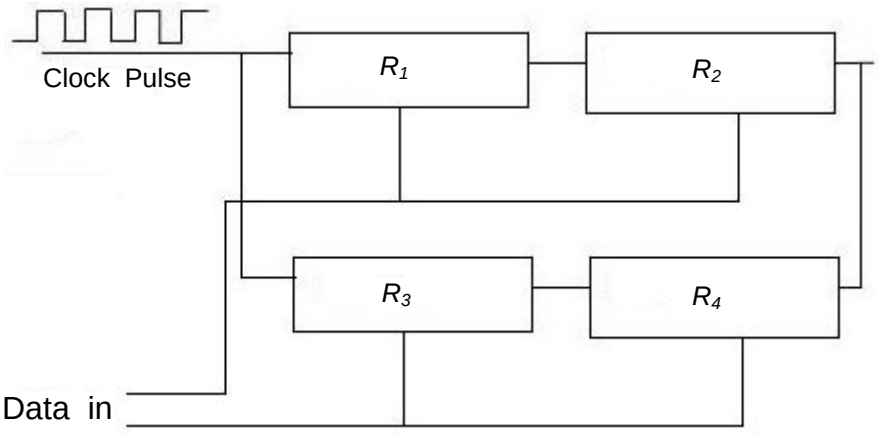

Figure 4: Parallel pipeline

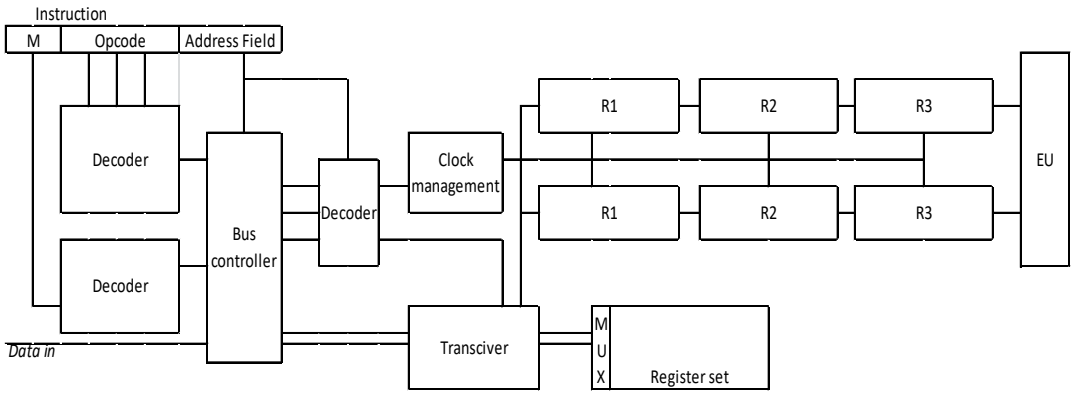

Figure 5: Proposed Parallel pipeline fetch system

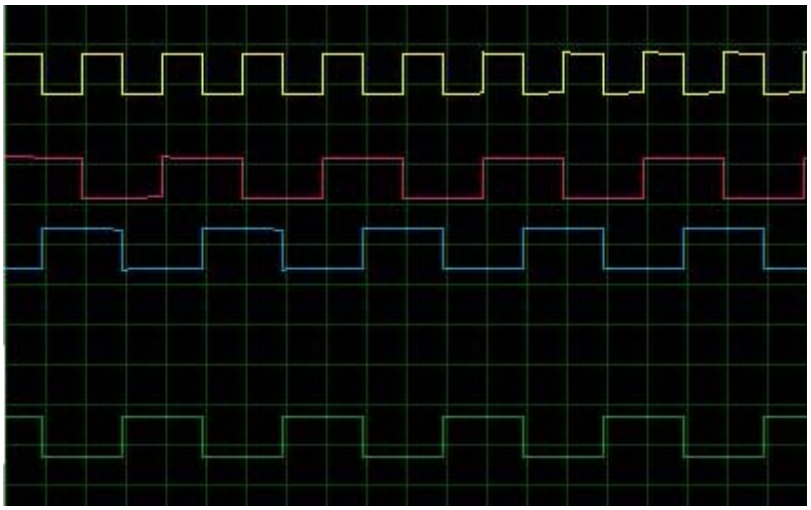

Figure 6: The binary code obtained at the output of the pipeline