INTERNATIONAL JOURNAL OF ADVANCE RESEARCH, IDEAS AND INNOVATIONS IN TECHNOLOGY

ISSN: 2454-132X
Impact factor: 4.295
(Volume 5, Issue 1)
Available online at: www.ijariit.com

A survey on control implementation scheme

Boddu Praveen
boddupraveen99@gmail.com
Saveetha School of Engineering, Kuthambakkam, Tamil Nadu

P. Vamsi Krishna
prudviyadav1@gmail.com
Saveetha School of Engineering, Kuthambakkam, Tamil Nadu

ABSTRACT

In today's computer age, the processor plays a major role. The processor is the heart of the computer. These are the days where Nano components play a major role. Hence there is a need for a processor's size to be reduced and an effective control implementation scheme is necessary. Hence, this paper made a survey on how to implement an effective control scheme. It helps the computer architects in choosing the better control implementation scheme while designing a processor for equipment like laptops, mobile phones, Personal computers etc.

Keywords— *Instruction cache, Data cache, Data transfer circuits, Dynamic redundancy, Adjacent error/correction system, Instruction latency, Zero-overhead loop controller, Dynamic voltage frequency scaling Technology*

1. INTRODUCTION

Many industrial applications cannot be implemented using a single processor. Hence, the use of a multiprocessor configuration computer has a number of potential advantages over the use of a single processor design. For multi-processor configuration, more efficient control is needed. Control of a system with a multiprocessor is much more complicated. Control refers to the components of the processor that commands the Datapath, memory, I/O devices according to the instructions of the memory. Control unit is responsible for issuing control signals to all the components in the computer. The implementation of this issue of control signals is called a control implementation scheme. An effective control implementation scheme can increase the speed of instruction issue, instruction throughput and reduces the execution time. Hence the better performance of the computer is achieved.

Processors are compared with the heart and brain of human beings. The flow of instructions and data in the processor are similar to the nervous system of human beings. Processors are often called microcomputers because of the variety of functions they perform.

The processor is mainly designed to implement the memory. A processor requires less time than memories for accessing. For example, a processor is clocked at 1GHz have to access the memory in 20 picoseconds during the period of its 1GHz clock. But, the memory interfaced to it requires 10nanoseconds for access. So, the processor ends up waiting during each memory access, washing the execution cycles. In order to reduce the number of access to main memory, computer architects designed and added instruction cache and data cache to the processors. The instruction cache acts as a buffer between the external memory and the processor thus reducing the number of accesses to main memory. Cache is approximately 10 times faster than main memory. So, we can access the data in 20 picoseconds instead of 10 nanoseconds (Example). The instruction cache is used to store frequently used instructions and data cache is used to store frequently used data. Actualizing fewer guidelines and tending to modes on silicon decreases the complexity of the instruction decoder and the execution unit. This enables the machine to be timed at a quicker speed since less work should be done each clock period.

Generally, RISC architecture has an expansive arrangement of registers. The number of registers accessible in a processor can influence execution in a similar way memory access does. If the data values all reside in memory throughout the calculations, several memory accesses should be used to utilize them. If the data values are stored within the internal registers of the processor instead, their access throughout calculations is going to be quicker. It is smart then to possess a lot of internal registers.

2. LITERATURE SURVEY

Wing N. Toy et.al proposed a unified parity check scheme for control and data transfer circuits. Data word and the decoding operations are combined into a single function performed by translator circuitry to ensure no error. Expanded Parity can be used to detect a larger class of failure conditions. Same logic gates are used for performing the decoding and gating functions which minimizes the number of required gates. The author proposed to replace the complex circuitry in the conventional Check scheme

circuit with regular memory structure. More than one data bus was used to enhance the data transfer speed. But in the proposed method, only one microinstruction can be carried out at a time which reduces the processor speed. [1]

David K.' Rubin et.al proposed the self-testing and repairing computer having dynamic redundancy capability to identify fault by itself using the control implementation scheme. With this control, the implementation scheme processor can identify the fault by itself if any error occurred. Hence, it reduces the stalling and bubbles in pipelines. [2]

Richard R. Ramseyer proposed an easy and low-cost architecture for arithmetic and logical operations. The proposed system was included with multi-processing and distributed processing with better interrupt service mechanism. It has the ability to speed up the computations by avoiding redundant loads and stores. [3]

Kenji Toda et.al proposed a system with parallel processing in order to deal with a large amount of information in a short time. The proposed system has increased the number of MPU's to identify the weak points of the program and checks the correctness of the data access on the shared memory. Synchronization of signals was implemented among the different processors to enable the switching of the processors. The proposed control scheme mainly concentrates on the bit manipulation and control of I/O devices. [4]

J. Arlat et.al proposed the use of Gallium Arsenide to increase the speed of the computer. The proposed system was able to perform billions of calculations per second and also the proposed implementation scheme was able to Detect and correct the error. This was called Adjacent Error correction/detection system. [5]

A system for efficient implementation of piecewise algorithms and efficient generation of control signals was proposed by the authors Lothar Thiele et.al in the year 1991. The Proposed system was very simple and flexible and it is also independent of the size of the problem. The Datapaths can be selected dynamically but the demerit of this system was that it can be used only when the conversion is carried out only in a specific direction. [6]

Fredrik Dahlgren et.al in 1993 designed a system for efficient control implementation scheme using simple prefetching scheme and a second buffer to keep track of outstanding requests like reading, write, pre-fetch etc. to reduce the instruction latency. The proposed system has a simple control block but it has more traffic consumption and cache pollution are the major drawbacks of this system. [7]

The author Meng-Chou Chang et.al, in 2010 presented a system to handle the data hazards effectively in asynchronous pipelined processors using control implementation scheme. With this system hardware area is reduced by 13% and 22.1% performance gain was achieved when compared to the convention hazard handling schemes. The presented system operates at high speeds with low power consumption and low electromagnetic emissions. 29.2% improvement in terms of instruction decoding time was achieved. To reduce the hardware complexity the author used two special control signals sys_reset1and sys_reset2. To reduce control hazards, the author used two 1-bit registers, the instruction color register located in instruction fetch (IF) stage and the system color register located in the EXE stage to implement the coloring scheme were used. The drawback in this system was, the problem occurs when the processor is reset and the IF stage has to maintain two special registers, the 2-bit instruction index register and the 1-bit FBTI (First Branch Target Instruction) increases hardware area. [8]

Nikolaos Kavvadias et.al proposed a controller to enhance the performance of embedded RISC processors which is called as Zero-Overhead Loop Controller (ZOLC). ZOLC unit is incorporated into RISC processors to provide effective means for low-overhead looping without negative impact to the processor cycle time and it was possible by an efficient control implementation scheme. By using this controller unit 10% performance speedup and a 40% boost in cycle performance was achieved. The presented method can be applied to structures with dynamic loops that have bound values only resolved at runtime. The drawback of this method is lack of hardware-floating point support. [9]

Michel Dubois et.al proposed a MIPS rate stabilization technique to achieve stable execution time and throughput of the processor by controlling its MIPS and using Dynamic Voltage Frequency Scaling Technology (DVFS). The proposed system consumes less power per instruction. Power dissipation and energy efficiency were improved. An average of three microseconds of resynchronization stall per control window made the results more accurate. The major drawback in the proposed method is the controller is tuned only for a step input. [10]

The author Y. Sujatha et.al proposed the implementation of a 32-bit pipelined RISC processor without interlocking stage. With the proposed system, more executions can be done with a single instruction. The proposed system includes Branch predictor. Hence, the total energy consumption for the execution of the program is reduced. To implement an efficient control implementation scheme, collapsible latch controllers are used. Presence of control hazards is the drawback of this system. [11][15]

Kirat Pal Singh et.al proposed a low power MIPS processor using Clock gating technique to reduce the power consumption. The author proposed the use of advanced encryption and Decryption schemes. 5 pipeline stages were used to process several hazards. A novel scheduling scheme which can use more efficient buffer structures was devised. With this novel scheme, memory usage was reduced significantly. [12,13]

The author Aleti Shankar et.al designed and implemented a 32-bit RISC MIPS processor to implement the better control implementation scheme using the MIPS which has only three instruction types and Embedded system array block(EAB). The

designed system was added with Instruction and data cache and was provided with many internal registers. But the memory accesses were limited to load and store instructions and also only single Datapath was available. [14]

The author Shaik Afroz et.al proposed a RISC based architecture for low power applications. The proposed system has minimized memory access time and decoding time. In the proposed system the control unit can efficiently issue the control signals and it is possible to execute a number of instructions in a single instruction cycle. Since memory access is limited only to load accumulator and store accumulator, memory access time was improved dramatically. An efficient error-correcting scheme designed by the author Shaik Afroz et.al. [15]

3. BASIC MIPS IMPLEMENTATION IN PROCESSOR

MIPS refers to Million Instructions per second. It is a measure of the performance of a processor. Basic MIPS has three instruction formats (R, I and J)

R-Format

6 Bytes	5	5	5	5	6
Opcode	rs	rt	rd	shamt	Func

I- Format

31 6	5	5	16 0
Opcode	rs	rt	n/offset

J-Format

6	26
Opcode	raddr

Op: Basic operation of the instruction, traditionally called the opcode.
rs: The first register source operand
rt: The second register source operand
shamt: Shift amount (used to shift the instructions)
funct: function code. This field is used to select a specific variant of the operation in the op field.

There are three styles in MIPS implementation.

- Single Cycle: Single cycle scheme performs each instruction in 1 clock cycle. The clock cycle must be long enough for the slowest instruction.
- Multi-Cycle: Multi-cycle scheme breaks the fetch/execute scheme into multiple steps. One step is performed in each clock cycle.
- Pipelined: Pipelined scheme executes each instruction in multiple steps. One step of various instructions is performed in each clock cycle.

4. CONTROL IMPLEMENTATION SCHEME

The Control unit in the computer is responsible for issuing control signals to all of the components.

There are seven control signals generated by the Control unit.

RegDst
RegWriten
ALUSrc
PCSrc
MemRead
MemWrite
MemtoReg

Signal name	Effect when deasserted	Effect when asserted
RegDst	The register destination number for the Write register comes from the rt field (bits 20:16).	The register destination number for the Write register comes from the rd field (bits 15:11).
RegWrite	None.	The register on the Write register input is written with the value on the Write data input.
ALUSrc	The second ALU operand comes from the second register file output (Read data 2).	The second ALU operand is the sign-extended, lower 16 bits of the instruction.
PCSrc	The PC is replaced by the output of the adder that computes the value of PC + 4.	The PC is replaced by the output of the adder that computes the branch target.
MemRead	None.	Data memory contents designated by the address input are put on the Read data output.
MemWrite	None.	Data memory contents designated by the address input are replaced by the value on the Write data input.
MemtoReg	The value fed to the register Write data input comes from the ALU.	The value fed to the register Write data input comes from the data memory.

Fig. 1: Effect of the seven control signals

Figure 1 depicts the effect of all the control signals on the components of the processor. When the 1-bit control to a two-way multiplier is asserted, the multiplexer selects the input corresponding to 1. Otherwise, if the control is deasserted, the multiplexer selects the 0 input.

4.1 The ALU control

The MIPS ALU in the processor defines the 6 following combinations of four control inputs:

Table 1: ALU control fields and Function

ALU Control Lines	Function
0000	AND
0001	OR
0010	Add
0110	Subtract
0111	Set on less than
1100	NOR

Depending on the instruction class, the ALU will need to perform one of these first five functions. For Load word and store word instructions, ALU is used to compute the memory address by addition. For the R-Type instructions, the ALU needs to perform one of the five actions (AND, OR Subtract, add, or set on less than), depending on the value of the 6-bit funct (or function) field.

Instruction opcode	AluOp	Instruction operation	Funct Field	Desired ALU action	ALU control input
LW	00	load word	xxxxxx	add	010
SW	00	store word	xxxxxx	add	010
Branch eq	01	branch eq	xxxxxx	subtract	110
R-type	10	add	100000	add	010
R-type	10	subtract	100010	subtract	110
R-type	10	AND	100100	and	000
R-type	10	OR	100101	or	001
R-type	10	set on less	101010	set on less	111

Fig. 2: Setting ALU Control Bits

Figure.4 depicts how the ALU control bits are set depending on the ALUOp control bits and the different function codes for the R-Type instruction. The opcode listed in the first column determines the setting of the ALUOp bits. All the encodings are shown in binary. Notice that when the ALUOp code is 00 or 01, the desired ALU action does not depend on the function code field: in this case, it is said to be "don't care" about the values of the function code and the function filed is shown as xxxx. When the ALUOp value is 10, then the function code is used to set the ALU control input.

4.2 Designing main control unit

Table 2: Truth table for the 4 ALU control bits

ALUOp		FUNCT FIELD						
ALUOp1	**ALUOp0**	**F5**	**F4**	**F3**	**F2**	**F1**	**F0**	**FUNCTION**
0	0	X	X	X	X	X	X	0010
X	1	X	X	0	X	X	X	0110
1	X	X	X	0	0	0	0	0010
1	X	X	X	0	0	1	0	0110
1	X	X	X	0	1	0	0	0000
1	X	X	X	0	1	0	1	0001
1	X	X	X	1	0	1	0	0111

ALUOp and the function code field are the inputs. Only the entries for which the ALU control is asserted are shown. Some don't care entries have been added. For example, the ALUOp does not use the encoding 11, so the truth table can contain entries IX and XI, rather than 10 and 01. Note that when the function field is used, the first 2 bits (F5 and F4) of these instructions are always 10, so they are don't care terms and are replaced with XX in the truth table.

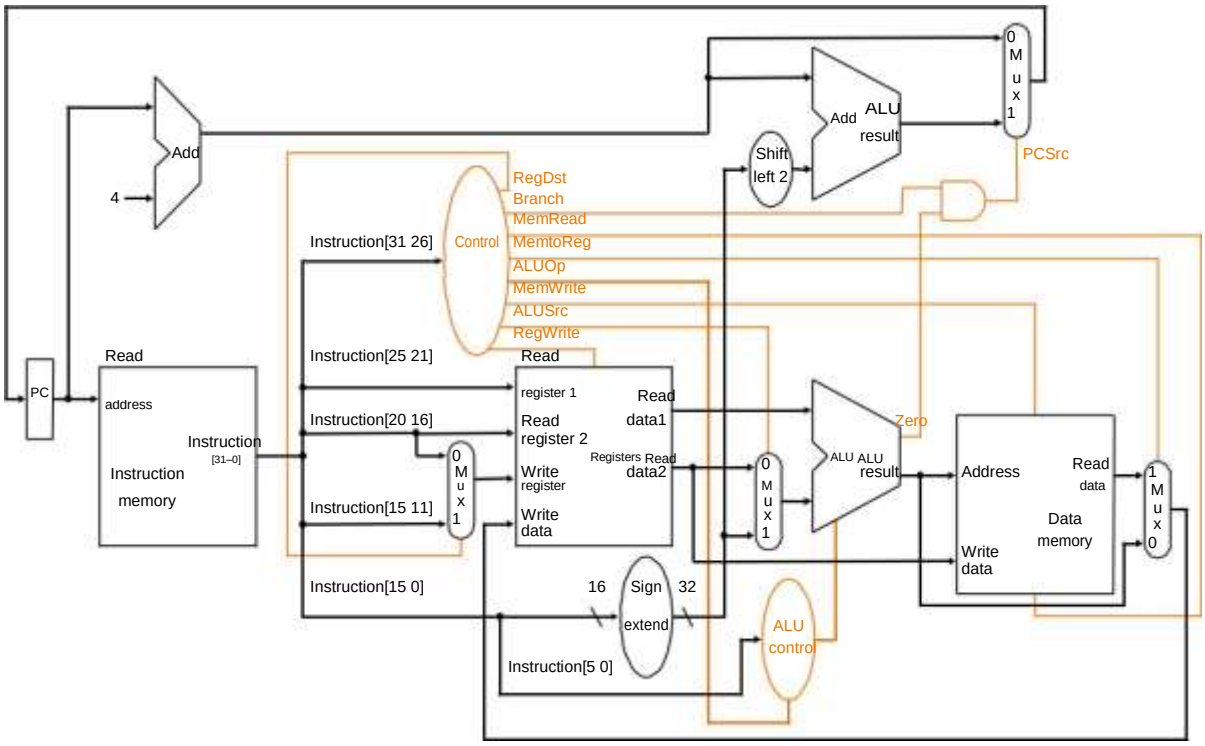

Fig. 3: Simple Datapath with the control unit

The input to the control unit is the 6-bit opcode filed from the instruction. The output of the control unit consists of three 1-bit signals that are used to control multiplexers (RegDst, ALUSrc and MemtoReg), three signals for controlling reads and writes in to register file and data memory (RegWrite, MemRead, and MemWrite), a 1-bit signal used in determining whether to possibly branch (Branch), and a 2-bit control signal for ALU (ALUOp). An AND gate is used to combine the branch control signal and the Zero output from the ALU. The AND gate output controls the selection of the next Program counter.

4.3 Finalizing control

It is shown how the instructions operate in steps. The outputs are the control lines and the input is 6-bit opcode field, Op [5:0]. Thus, a truth table can be created for each of the outputs based on the binary encoding of the opcodes

Table 3: Truth table representation for single-cycle control implementation

Input/Output	Signal Name	R-Format	Iw	sw	beq
Inputs	Op5	0	1	1	0
	Op4	0	0	0	0
	Op3	0	0	1	0
	Op2	0	0	0	1
	Op1	0	1	1	0
	Op0	0	1	1	0
Outputs	RegDst	1	0	X	X
	ALUScr	0	1	1	0
	Memtoreg	0	1	X	X
	RegWrite	1	1	0	0
	MemRead	0	1	0	0
	MemWrite	0	0	1	0
	Branch	0	0	0	1
	ALUOp1	1	0	0	0
	ALUOp0	0	0	0	1

The top half of the table gives the combinations of input signals that correspond to the four opcodes. The Op [5:0] corresponds to bits 31 to 26 of the instruction. The bottom portion of the table gives the outputs for each of the four opcodes. Thus, the output RegWrite is asserted for two different combinations of the inputs. MIPS opcodes are used in a full control implementation

5. CONCLUSION

A survey has been made on the control implementation scheme. This paper describes the basic MIPS and its formats. This paper also surveyed on various control implementation schemes proposed earlier by various authors. The basic Control implementation was described.

6. REFERENCES

[1] WING N. TOY et.al, "Check Schemes for Integrated Micro-programmed Control and Data Transfer Circuitry", IEEE Transactions on Computers, 1970, Volume: C- 19, Issue: 12, Pages: 1153-1159

[2] D. K. Rubin et.al, "The STAR (Self-Testing And Repairing) Computer: An Investigation of the Theory and Practice of Fault-Tolerant Computer Design", 1971, Volume: C-20, Issue:11, Pages: 1312-1321

[3] Ramseyer et.al, "A multi-microprocessor implementation of a general purpose pipelined CPU", 1977, a 4th annual symposium on computer architecture

[4] Kenji Toda et.al, "The control of processors and its implementing environment on the simulator for parallel processing", Springer Digital Library, IPSJ Transactions on Computer Vision and Applications

[5] J. Arlat et.al, "Implementation and Evaluation of a (b,k)-Adjacent Error-Correcting/Detecting Scheme for Supercomputer Systems", IBM Journal of research and development

[6] Lothar Thelle et.al, "Control generation in the design of processor arrays", Springer journal of VLSI signal processing

[7] Fredrik Dahlgren et.al, "Fixed and Adaptive Sequential Prefetching in Shared Memory Multiprocessors", 1993 International Conference on Parallel Processing, Volume:1, Pages: 56-63

[8] Meng-Chou Chang et.al, "Comparison of two data hazard handling schemes for asynchronous pipelined processors", IEEE 3rd International Conference on Computer Science and Information Technology, 2010, Volume:4, Pages: 36-40

[9] Nikolas Kavvadias et.al," Elimination of Overhead Operations in Complex Loop Structures for Embedded Microprocessors", IEEE Transactions on the computer, 2008, Volume:57, Issue:2, Pages:200-214

[10] Michel Dublois et.al, "Dynamic MIPS rate stabilization in out-of-order processors", ACM SIGARHCH Computer Architecture News, 37(3)

[11] Y. Sujatha et.al, "Design and Implementation of Synchronous and advanced 32-Bit VLIW-MISC processor", International Journal of Applied Research & Studies

[12] Aleti Shankar et.al, "Design & Implementation Of 32-Bit RISC (MIPS) Processor", International Journal of Engineering Trends and Technology", Volume 4 Issue 10, Oct 2013

[13] Shaik Afroz et.al, "Implementation of RISC-Based Architecture for Low power applications", IOSR Journal of Electrical and Electronics Engineering", Volume 8, Issue 4 (Nov. - Dec. 2013)

[14] Kirat Pal Singh et.al," LOW POWER ENCRYPTED MIPS PROCESSOR BASED ON AES ALGORITHM", Journal of Global Research in Computer Science, 2012, Volume 3, Issue No: 4

[15] Vuohui Wang et.al," Parallel inter-leaver architecture with new scheduling scheme for high throughput configurable turbo decoder", IEEE International Symposium on Circuits and Systems, 2013, Pages: 1340-1343

