32- Bit Pipeline RISC Processor in VHDL using Booth Algorithm

Swati Joshi[1],Sandhya Shinde[2], Amruta Nikam[3]

E & TC Dept., DYPIEMR, swatijoshi57@yahoo.com

Abstract-*When low power issues are considered into account for encoding and cyclic controlling .By the no. of stages (pipelines) in traditional RISC architecture the issues related to performance & speed are solved. As power has become an main aspect in the design of general purpose processors. Low power consumption helps to reduce the heat dissipation, increases battery life and boost the device reliability. The minimization of power dissipation is done at different levels of design process by applying various low power techniques like Booth Multiplier.*

Key words- **VHDL, Booth multiplier, ALU, ISA, CISC**

I. INTRODUCTION

The dominant architecture in the PC market is the Complex Instruction Set Computer (CISC) design, which has complex nature instruction set architecture (ISA) .Such type of complex instruction set, is designed to provide an instruction set which closely supports the operations and data structures used by Higher-Level Languages (HLLs). The decision of CISC processor designer to provide a different type of addressing modes gives to variable-length instructions. The side effect of this variable length instruction types is that the number of clocks needed to execute instructions varies widely. This leads to problems in instruction scheduling and pipelining. . For these reasons, in the early 1980s designers started looking at simple ISAs. Requirement of the simple ISAs give rise to new term Reduced Instruction Set Computer.

RISC designers areConcerned with fast processing, and so they use, pipelining. Pipeliningis a design technique where the computer's hardware gives more than one instruction at a time, and doesn't wait for one instruction to complete before starting the next.

Remember our typical CISC machine? For same
Number of stages the RISC machine executes each stage in parallel form. As soon as one stage completes, it passes on the results to the next stage and then begins working on another instruction.
Performance of pipelined system depends on the time it takes only for any one stage to be completed---not on the total time for all the stages as with non-pipelined designs.

In a typical pipelined RISC, each instruction takes 1 clock cycle for each stage, so the processor can accept 1 new instruction per clock. Pipelining cannot be improving the latency of instructions, but it does improve the overall throughput. As with Complex instruction set computers, the ideal is not always achieved. Sometimes pipelined instructions take more than one clock to complete the stage. When that happens, the processor has to stall and not accept new instructions until the slow instruction has moved on to the next stage.

Speed.Since simplified instructions are allowed for pipelined, superscalar design RISC processors often achieve 2 to 4 times the performance of Complex instruction set computer processors using comparable semiconductor technology and the same clock rates.

Simpler hardware.The instruction set of reduced instruction set computer processor is so simple, it uses up much less chip space; extra functions, such as memory management unit , arithmetic units, can also be placed on the same chip. Small chip to allow a semiconductor manufacturer to place more parts on a single silicon wafer, which can lower the per-chip cost dramatically.

Shorterdesigncycle. Since reduced instruction set computer processors are simpler than corresponding Complex instruction set computer processors, they can be designed more quickly, and take an advantage of other technological developments sooner than corresponding Complex instruction set computer designs, leading to greater leaps in performance between generations.

II.SYSTEMMODELANDIMPLEMENTEDDESIGN

Overall design is composed of pipeline stages instruction fetch unit ,instruction execution ,decode unit ,Memory ,write back stage , RT-level analysis(Resister Transfer Level) Functional unit composition, and control signal generation Pipeline Stage.

Instruction Fetch

The Instruction Fetch (IF) stage is important for obtaining the requested instruction from memory. The instruction and the program counter (which is adjunct the next instruction) are stored in the IF/ID pipeline register as temporary storage so that it may be used in the next stage at the start of the next clock cycle.

Instruction Decoder

ID stage sends control command to other units of the processor based on decode of .Instruction Instruction is sent to control the unit and decoded here. Read register fetches data from register file. Branch unit is also included in ID stage. Input of ID stage is from IF stage. Instruction decode stage decodes instruction to control signals and prepared the operand.

Execution

EXE stage executes arithmetic operation. Main component of EXE stage is ALU. Arithmetic logic unit and shift-register compose of ALU. . Function of EXE stage is to do operation of instruction, such as add and subtraction. ALU sends result to EX/MEM pipeline register before entering MEM stage.

Memory and IO

The function of MEM stage is to fetch data from memory and store data to memory. Another function is to input data to the processor and output data. If instruction is not memory instruction or IO instruction, the result is sent to WB stage. The MEM stage structure is shown in Figure below Storing data in the register is the main function after the result is calculated. Same result may be not stored in RAM defined, and same result can be written to register directly.

Write-Back

WB stage executes of writing result, store data and input data to register the file. The purpose of WB stage is to write data to destination registry. For example, ADD R1, R2, R3 instruction memories result in R1 register to make programs run
faster.

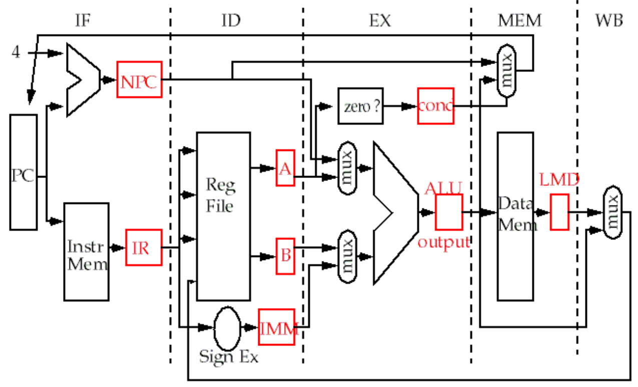

Fig.1 Block Diagram of Pipelineing stages

III REALIZATION

Booth Algorithm

In order to achieve high speed of multiplication, multiplication algorithms using parallel counters, such as the Booth algorithm have been proposed, and some multipliers based on the algorithms have been implemented for practical use. This type of multiplier controls much faster than an array multiplier for longer operands because its computation time is proportional to the length of operands. Booth multiplication is a technique that allows for smaller, faster multiplication circuits, by recording the numbers that are multiplied

It is possible to minimize the number of partial products by half, by using the technique of radix-4 Booth recoding. The basic idea is that, the shifting and adding for every column of the multiplier term and multiplying by 1 or 0, we take only every second column, and multiply by ±1, ±2, or 0, to obtain the same results. This method is the halving of the number of partial products is The advantage of this method. To Booth recode the multiplier term, we consider the bits in blocks of three, such that each block flaps the previous block by one bit. Grouping starts from the Lower side bits, and the first block only uses two bits of the multiplier.

To Booth algorithm the multiplier term, we consider the bits in blocks of three, such that each block overlaps the foregoing block by one bit. Grouping starts from the LSB, and the first block only use two bits of the multiplier. For the partial product generation, we adopt the Radix-4 Modified Booth algorithm to reduce the number of partial products for roughly one half.

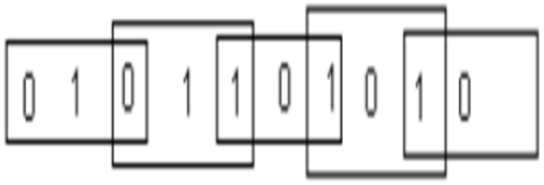

Fig. 2 Grouping of bits from the multiplier terms

.Each block is decoded to produce the correct partial product. The encoding of the multiplier Y, using the booth algorithm, originates the following five signed digits, -2, -1, 0, +1, +2. Each encoded digit in the multiplier performs a certain operation of the multiplicand, X, illustrated in the table

Block	Recoded	Operation on X
000	0	0X
001	+1	+1X
010	+1	+1X
011	+2	+2X
100	-2	-2X
101	-1	-1X
110	-1	-1X
111	0	0X

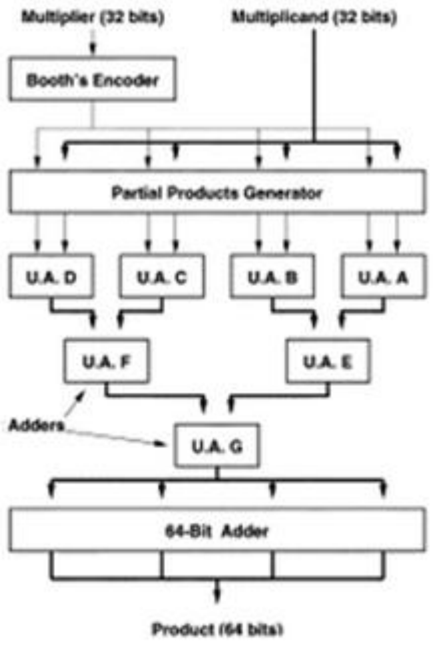

Fig. 3 Block Diagram of Booth Algorithm

IV CONCLUSION

32-bit RISC processor using VHDL this research is a trade-off between speed and area, so we use implement pipelining approach in order to obtain both of our requirements. With the Structural design approach and by using multiplexer for few control interface has minimize the area and complexity of control statements, We have used finite state machine for instruction decoding and generation of new control statement according to pipeline sequence, so synchronization problem is resolved.

REFERENCES

[1]Gaonker,Ramesh,MicroprocessorArchitecture,programmingand Application,PenramInternationalPublishing.

[2]Implementationofa32bitRISCProcessorfortheData-Intensive ArchitectureProcessing-In MemoryChipJereyDraper,JeSondeen, SumitMediratta,IhnKimUniversityofSouthernCalifornia InformationSciences Institute.

[3] VLSIProcessorArchitectureJOHN L.HENNESSY

[4] 64-

bitFloatingPointProcessingUnitforaRISCMicroprocessor
FujitsuLaboratories Ltd.

[5] RISCAND CISC Computer
ArchitectureByFarhatMasood
[6] Design and Implementation of a 64-
bitRISCProcessorusingVHDL UKsim 2009