Journal of Electronics and Communication Systems
Volume 1 Issue 1

Design and Implementation of Low Power Pipelined 32-bit High Performance RISC Core

Shashidhar R, Mahalingaswamy A M, Santhosh Kumar R, Roopa M
Department of ECE, Dayananda Sagar College of Engineering, Bengaluru, India
E-mail: Shashidhar.rin@gmail.com

Abstract

In this paper, we are proposing low power design technique in front end process. Harvard architecture is used which has distinct program memory space and data memory space. Low power consumption helps to reduce the heat dissipation, lengthen battery life and increase device reliability. To minimize the power of RISC Core, clock gating technique is used in the architectural level which is an efficient low power technique. 7-SEG LEDs are connected to the RISC IO interface for testing purpose, Verilog code is simulated using Modelsim and then implementation is done using FPGA board.

Keywords*: RISC processor, architectural power reduction, clock gating*

INTRODUCTION

Over many years, RISC instruction sets have tended to grow in size. Thus, some have started using the term "load-store" to describe RISC processors, since this is the key element of all such designs [1]. Instead of the CPU itself handling many addressing modes, load-store architecture uses a separate unit dedicated to handling very simple forms of load and store operations. Today, RISC CPUs (and microcontrollers) represent the vast majority of all CPUs in use. The RISC design technique offers power in even small sizes and thus has come to completely dominate the market for low-power "embedded" CPUs. Embedded CPUs are by far the largest market for processors. RISC had also completely taken over the market for larger workstations for much of the 90s. After the release of the Sun SPARC station the other vendors rushed to compete with RISC based solutions of their own. Even the mainframe world is now completely RISC based. Low-power embedded processors are used in a wide variety of applications including cars, phones, digital cameras,

printers and other such devices. There were lots of techniques like Clock Gating, Supply Voltage Reduction, Multi Vdd, Dynamic Voltage Frequency Scaling etc. to reduce the power. In the design considered as low power RISC, all the arithmetic, branch and logical operation are performed and the resultant value is stored in the memory/registers and retrieved back from memory when required [2]. In the design, power reduction is done in front end process so that low power RISC processor is designed without any complexity. In this work, we have 4-stage pipelining and low power techniques in front-end design process and it can further be planned to add more pipeline stages so as to increase the performance. Back-end low power techniques can also be applied to decrease the power [3].

The classic RISC pipeline consists of:

- Instruction Fetch.
- Instruction Decode and Register Fetch.
- Execute.
- Memory access.
- Register Write Back.

METHODOLOGY

The proposed processor is 32 bit low power RISC processor with pipelining architecture which gets instructions on a regular basis using dedicated buses to its memory, executes all its native instructions in stages with pipelining. It can communicate with external devices with its dedicated parallel IO interface. Clock Gating low power technique is used to reduce the dynamic power dissipation.

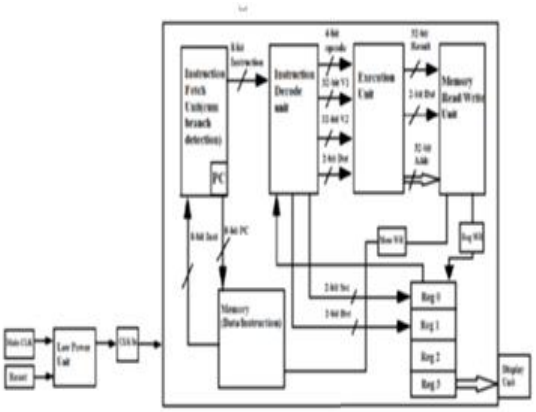

Fig. 1: *Architecture of Pipelined RISC Processor.*

The architecture consists of four stage pipelining: Instruction fetch, Instruction decode, Execute, Memory Read/Write Back. The architecture comprises of Modified Harvard Architecture. The architecture comprises of low power unit. The architecture of the pipelined RISC processor is shown in the above Figure 1. Arithmetic operations either take two registers as operands or take one register and a sign extended immediate value as an operand. Some arithmetic instructions are ADD, SUB and MUL. The result is stored in a third register. Logical operations such as AND OR, XOR do not usually

differentiate between 32-bit and 64-bit. Other logical instructions are NAND, NOR, NOT etc. [4].

Load/Store Instructions usually take a register (base register) as an operand and a 16-bit immediate value. The sum of the two will create the effective address. A second register acts as a source in the case of a load operation. In the case of a store operation the second register contains the data to be stored. Examples are: LW, SW etc. Branches and Jumps instruction: Conditional branches are transfer of control. A branch causes an immediate value to be added to the current program counter. Some common branch instructions are BZ (Branch Zero), BRZ (Branch Register Zero), JMP (Jump Instruction), JMPZ (Jump when zero) etc.

FEATURES OF PROPOSED ARCHITECTURE

- It is 4-stage low power pipelining processor.
- It provides hazard detection unit to determine when stall must be added.
- It is capable of providing high performance.
- Pipelining would not flush when branch instruction occurs as it is implemented using dynamic branch prediction.

Take cares of order-of-execution. Clock gating is a well-established power-saving technique that has been used for years. Synthesis tools such as Power Compiler can detect low-throughput data paths where clock gating can be used with the greatest benefit and can automatically insert clock-gating cells in the clock paths at the appropriate locations [5]. Clock gating is relatively simple to implement because it only requires a change in the net list. No additional power supplies or power infrastructure changes are required. The registers are implemented by Design Compiler by use of feedback loops. However, these registers maintain the same logic value through multiple cycles and unnecessarily use power. Clock gating saves power by eliminating the unnecessary activity associated with reloading register banks.

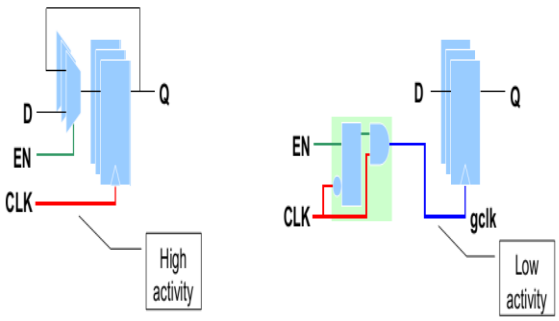

Fig. 2: *Clock Gating.*

With clock gating, portions of a design that are inactive have their clocks disabled on-the-fly to reduce dynamic power. An

example of this technique in use is the camera on a cell phone. When the camera is not in use, its logic may not be clocked. Bear in mind that although this technique will definitely reduce dynamic power, it does not reduce leakage power.

Benefits of Clock Gating:

Dynamic Power Savings

a. With low toggle rate on clock pin, internal power of registers is reduced.

b. Gated by the enable signal, the clock network has less switching activity and consumes less switching power.

Area Savings

a. Eliminating multiplexers saves area.

b. No RTL code change is required.

c. Clock gating is automatically inserted by the tool.

d. Technology independent.

Clock gating reduces the clock network power dissipation, relaxes the data path timing and reduces routing congestion by eliminating feedback multiplexer loops. For designs that have large multi-bit registers, clock gating can save power and reduce the number of gates in the design. However, for smaller register banks, the overhead of adding logic to the clock tree might not compare favorably to the power saved by eliminating a few feedback nets and multiplexers. Figure 3 shows three different clock-gating styles: general, multi-stage and hierarchical.

In multi-stage clock gating, further, gating is done for the clock-gating cells which have common enable signals. As a result, more dynamic power can be saved [6]. In the hierarchical clock-gating style, clock gating is done at the logic hierarchy level by gating hierarchical logic modules that share a common enable and the same clock group. This reduces the number of clock-gating cells in the design [7, 8].

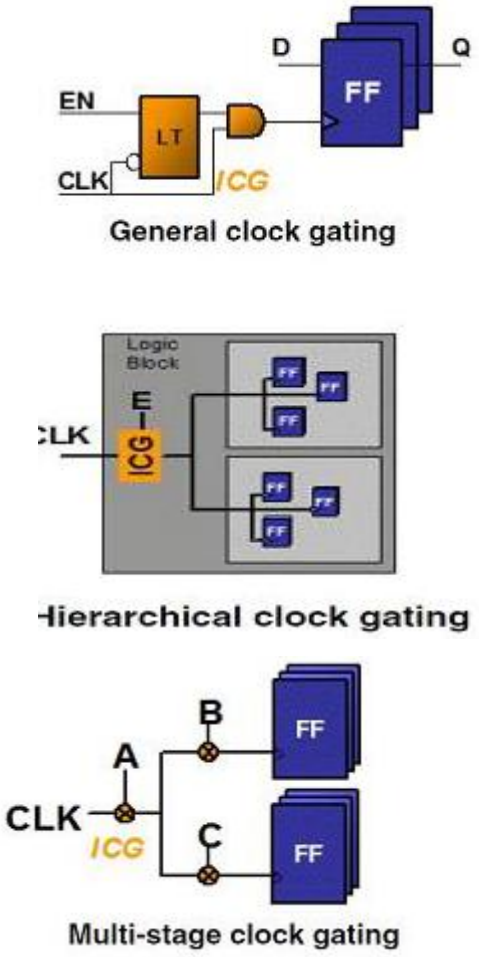

Fig. 3: *Clock-Gating Styles.*

SIMULATION RESULTS

Modelsim is used for simulation and results have been verified. Once the functional simulation is done, it is implemented using QUARTUS-II and Altera FPGA board. Testing and debugging of the work was done on FPGA board. The complete project has been developed on FPGA Development board platform, which has different testing and debugging scopes such as on-board push-button switches, toggle switches and LEDs. So, all the testing has been carried out on the board. 7-SEG LEDs is connected to the RISC IO interface for testing purpose. Different patterns have been generated with RISC instructions and are sent to 7-SEG LEDs through parallel IO interface of RISC.

Here, in this ALTERA family, many different devices were available in the tool. In order to implement this design the device named as "QUARTUS-II" has been chosen and the package as "EPF20F484C7" is used on ALTERA DE1 board platform. The design of 32 bit-RISC is simulated and its results were analyzed as follows:

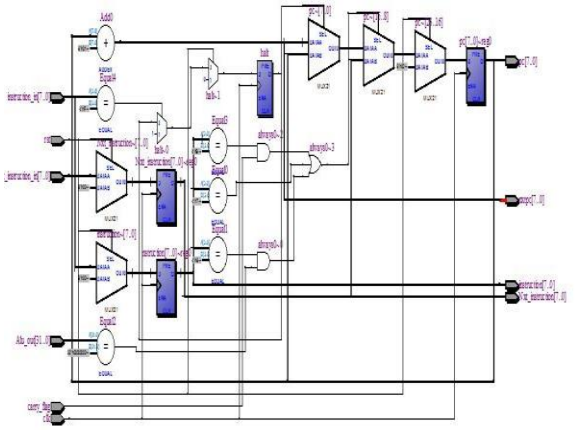

Fig. 4: *RTL Schematic of Fetch Unit.*

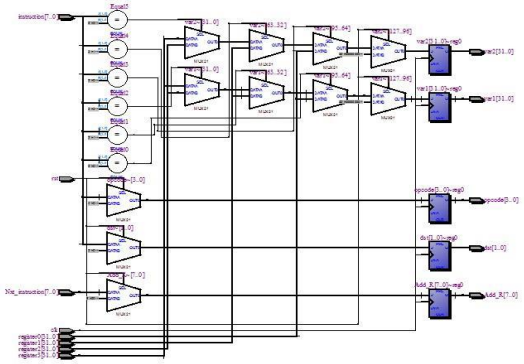

Fig. 5: *RTL Schematic of Decode Unit.*

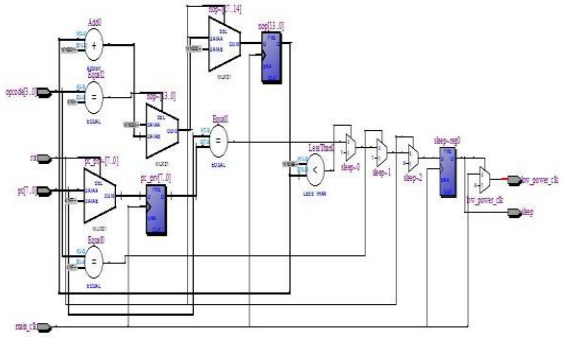

Fig. 6: *RTL Schematic of Low Power Unit.*

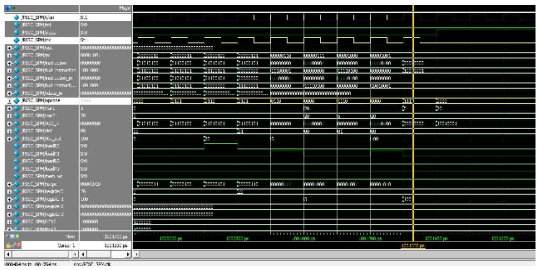

Fig. 7: *Simulation Result of the Low Power Unit when Instruction is Halt.*

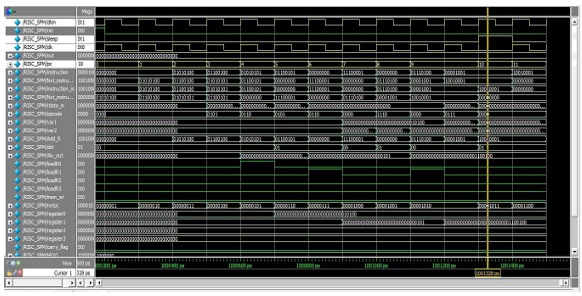

Fig. 8: *Simulation Result of the Low Power Unit when Continuous NOP Operations.*

TIMING SUMMARY

Minimum Period: 8.986 ns (Maximum Frequency: 111.286 MHz)

Minimum Input Arrival Time before Clock: 6.135 ns

Maximum Output Required Time after Clock: 7.184 ns

Maximum Combinational Path Delay: 8.410 ns

Total REAL Time to Xst Completion: 99.00 secs

Total CPU Time to Xst Completion: 99.46 secs

Analysis & Synthesis Resource Usage Summary

	Resource	Usage
1	Estimated Total logic elements	16,161
2		
3	Total combinational functions	7873
4	◢ Logic element usage by number of LUT inputs	
5	-- 4 input functions	7476
6	-- 3 input functions	230
7	-- <=2 input functions	167
8		
9	◢ Logic elements by mode	
10	-- normal mode	7727
11	-- arithmetic mode	146
12		
13	◢ Total registers	8493
14	-- Dedicated logic registers	8493
15	-- I/O registers	0
16		
17	I/O pins	31
18	Embedded Multiplier 9-bit elements	2
19	Maximum fan-out node	Low_Power_Unit:U_lpu|low_power_clk
20	Maximum fan-out	8470
21	Total fan-out	56498
22	Average fan-out	3.45

Fig. 9: *Resource Usage Summary.*

CONCLUSION

The design of 4-stage 32-bit low power RISC processor performing arithmetic, logical, memory and branch instructions is presented in this paper. Architecture is devised in order to felicitate the writing of codes in Verilog. The Verilog coding synthesis issues play a vital role in the speed-area optimality because RTL schematic depends heavily on how we have coded in Verilog. The design is simulated on Modelsim SE 6.4e tool and then synthesized and verified on Altera FPGA board. The proposed architecture is able to prevent pipelining from flushing when branch instruction occurs and able to provide halt support. It can be inferred from the synthesis report that proposed

architecture offers speed which is approximately 111.286 MHz‟s. In the future, one can extend this project to 64-bit RISC processor with high performance and lower power consumption using clock gating techniques or other lower-power techniques which can be implemented in both front end and back end with 45nmtechnology. By increasing the number of instructions and make a pipelined design with less clock cycles per instruction and more improvement can be added in the future work.

REFERENCES

1. M.Kishore Kumar. FPGA based implementation of 32 Bit risc processor; 2012.
2. Preetam Bhosle, Hari Krishna Moorthy. FPGA implementation of low power pipelined 32-Bit risc processor; 2012.
3. Neenu Joseph, Sabarinath S. FPGA based implementation of high performance architectural level low power 32-Bit risc core; 2009.
4. Indu M, Arun Kumar M. Design of low power pipelined risc processor; 2013.
5. Kui YI, Yue-Hua Ding. 32-Bit risc cpu based on mips instruction fetch module design; 2009.
6. Rupali S. Balpande, Rashmi S. Keote. Design of FPGA based instruction fetches & decode module of 32-Bit risc (mips) processor; 2011.
7. Mahendra Pratap Dev, Deepak Baghel, Bishwajeet Pandey et al. Clock gated low power sequential circuit design; 2013.
8. Sivarama P Dandmundi. Guide to RISC processor for programmers and engineer.