ISSN XXXX XXXX © 2017 IJESC

Research Article Volume 7 Issue No.5

Design of Low Power Embedded Processor using Clock Gating (Power Reduction) Technique

Miloni Ganatra[1], Hansa Shingrakhia[2], Zalak Patel[3], Shikha Singh[4]
Assistant Professor[1, 2, 3, 4]
Department of Electrical & Electronics Engineering [1, 3, 4]
Department of Electronics & Communication Engineering [2]
Indus University, Rancharda,Via Thaltej, Ahmedabad, Gujarat, India

Abstract:
Recently the market of embedded processors is growing rapidly according as there increase different demands for their applications to mobile computing/communication, consumer electronics, etc. Although 4, 8, and 16-bit microprocessors have dominated conventionally the embedded processor market, these advanced applications require not only higher performance but also more extensive versatility. In addition, with the recent advance of the fabrication technology, any of ARM and SPARC processors can be soon replaced by another one through a renewal of fabrication process, which has space enough to admit a variety of additional functions. The market of the embedded processors grows remarkably, since the quantity of products with the use of them is getting larger than that of general purpose processors.

Keywords: ARM, SPARC, Power consumption, Switching frequency, RISC, clock gating, low power consumption

I. INTRODUCTION TO LOW POWER EMBEDDED PROCESSOR

An embedded processor is a processor that has been “embedded” into a device. It can be programmed to interact with different pieces of hardware. Performance wise, an embedded processor can outperform a microcontroller, but does not have as much performance as a general-purpose microprocessor. Low-power embedded processors are used in a wide variety of applications including cars, phones, digital cameras, printers, and other such devices. The reason for their wide use is that they are small; therefore, they do not take up much die area and are cost effective to fabricate. Also, embedded processors are verified, eliminating the need to spend additional engineering processor that will support a pre-defined instruction set. This processor will follow man-hours tracking down hardware flaws. Another great advantage in using embedded processors is that they run software, which enables one to deal with changing specifications as various system requirements change. Low power processors are the key to the realization of portable electronic devices, in which power consumption is an important factor. Low-power consumption helps to reduce heat dissipation, lengthen battery life, and increase device reliability. In this paper we will discuss a 16-bit RISC type embedded RISC architecture. There are several power saving techniques that can be used in design; however, the main focus of our processor’s low power architecture will be clock gating.

II. REASONS FOR LOW POWER

There are several reasons for emphasizing low power dissipation in modern processor designs. Some of these reasons are device performance related issues, while others may be manufacturing issues. In accordance with what was stated in the introduction, with low power processors being very important in today’s mobile devices, one of the main advantages of having a low power design is that the battery life for these devices can be prolonged. Another performance related issue is that, in many cases, reducing the switching current in the processor increases the reliability of the device. Manufacturing issues such as packaging costs can come into play when considering advantages for a low power design. Heat dissipation is one of the major factors considered when chip packaging takes place. The low power design of a processor greatly reduces heat dissipation, which will in turn reduce the packaging costs also.

III. WAYS OF REDUCING POWER CONSUMPTION

There are many ways to reduce power consumption of a processor. Some of the methods that we will employ in our design are listed below.

- **Reduced supply voltage:** Power consumption (P) of a CMOS based processor is related to the supply voltage (V), switching frequency (f), and CMOS gate capacitance (C). The relationship can be described as:

$$P \mu C \cdot f \cdot V^2 \qquad [1]$$

The above relationship shows that power consumption can be reduced by reducing the supply voltage. But we also need to be aware of the fact that switching frequency is directly proportional to the supply voltage as well; that is:

$$V \mu f \qquad [2]$$

Therefore, while lowering supply voltage will reduce power consumption, it will also result in lower switching frequency, and thus slower processor speed. A trade off must be made in the form of speed in order to reduce power usage.

- **Full custom design:** Full custom design will help to reduce the number of logic gates in the implementation of the necessary functionality of the processor. Since each logic gate requires power to operate, lower gate count means fewer switching and thus power need.

- **Clock gating:** Clock gating is a method where certain parts of the processor are prevented from receiving the clock signal. If a part of the processor is not needed for a given operation, then the clock signal to that part can be stopped. Since switching requires power and in the absence of the clock signal no switching will take place, gating the clock will lower power need. However, the main focus of our processor's low power architecture will be **clock gating**.

IV. ARCHITECTURE OF 16 BIT RISC TYPE EMBEDDED PROCESSOR

T

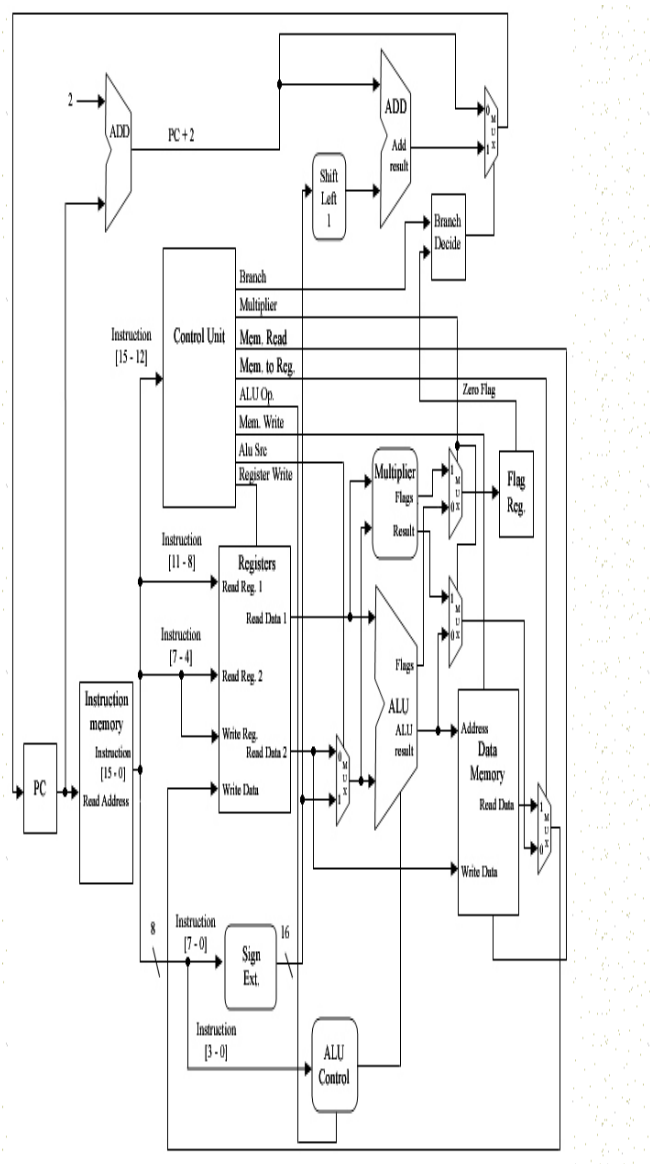

Figure.1. Block Diagram of 16 Bit Risc Type Embedded Processor

The overall diagram of the processor architecture is shown in figure. As seen from the diagram, the architecture consists of a five stage pipeline. The stages are Instruction Fetch, Instruction Decode, Execute, Memory, and Write Back. In an effort to reduce power consumption there is clock gating and signal gating throughout the design wherever applicable.

(a) Instruction Fetch:

This stage consists of the Program Counter, Instruction Memory, and the Branch Decide Unit.

- **Program Counter:**

The Program Counter (PC) contains the address of the instruction that will be fetched from the Instruction Memory during the next clock cycle. Normally the PC is incremented by one during each clock cycle unless a branch instruction is executed. When a branch instruction is encountered, the PC is incremented/decremented by the amount indicated by the branch offset. The PC Write input of the PC serves as an enable signal. When PC Write signal is high, the contents of the PC are incremented during the next clock cycle, and when it is low, the contents of the PC remain unchanged.

- **Instruction Memory:**

The Instruction Memory contains the instructions that are executed by the processor. The input to this unit is a 16-bit address from the Program Counter and the output is a 16-bit instruction word. Since the address bus is 16 bits wide, the size of the instruction memory can be at most 2^{16} or 64 kilo bytes.

- **Branch Decide Unit:**

The Branch Decide Unit is responsible for determining whether a branch is to take place or not based on the 2-bit Branch signal from the Control Unit and the Zero flag from the Arithmetic Logic Unit (ALU). The output of this unit is a 1-bit value which is high when a branch is to take place, and otherwise it is low. This output controls a multiplexer which in turn controls whether the PC gets incremented by one or by the amount indicated by the branch offset.

(b) Instruction Decode:

This stage consists of the Control Unit, Register File and the Sign Extend Unit .

- **Control Unit:**

The control unit generates all the control signals needed to control the Coordination among all the component of the processor. The input to this unit is the 4-bit opcode field of the instruction word. This unit generates signals that control all the read and write operations of the Register File and the Data Memory. It is also responsible for generating signals that decide when to use the multiplier and when to use the ALU, and it also generates appropriate branch flags that are used by the Branch Decide unit. In addition, this unit provides clock gating signals for the ALU Control and the Branch Adder module.

- **Register File: `**

This is a two port register file which can perform two simultaneous read and one write operation. It contains sixteen 16-bit general purpose registers. The registers are named R0 through R15. R0 is a special register which always contains the value zero and any write request to this register is always ignored. When the Reg Write signal is high, a write operation is performed to the register indicated by the write address, otherwise the value contained in the registers indicated by the read addresses are outputted.

- **Sign Extend Unit:**

The input to this unit is an 8-bit immediate value provided by all the immediate type instructions. This unit sign extends the 8-bit value to a 16-bit value signed value.

(c)Execute:

This stage consists of the Branch Adder, Multiplier, Arithmetic Logic Unit (ALU), and the ALU Control Unit.

- **Branch Adder:**

The branch adder adds the 12-bit signed branch offset with the current value of the PC to calculate the branch target. The 12-bit offset is provided by the branch instruction. The output of this unit goes to the PC control multiplexer which updates the PC with this value only when a branch is to be taken.

- **Multiplier:**

It consists of four distinct components. They are the Booth Encoder, Partial Product Generator, Carry Save Adder, and the Carry Look ahead Adder. Our multiplier architecture employs two main techniques to increase the speed of the multiplication process. First technique is to reduce the number of partial products and the second is to increase the speed at which the partial products are added.

- **Flag Register:**

This register holds all the flag bits. The bits are the Z (Zero), V (Overflow), C (Carry-out), and N (Negative).

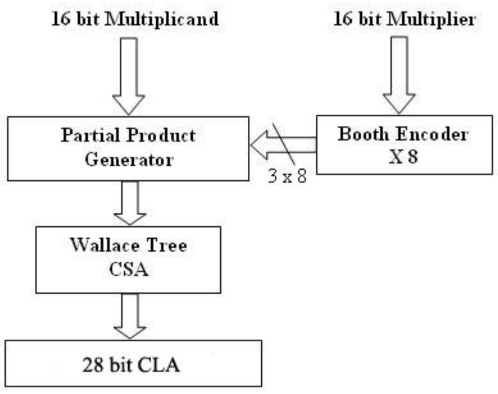

Figure. 2. 16 Bit Multiplier

- **Arithmetic Logic Unit (ALU)**:

The ALU is responsible for all arithmetic and logic operations that take place within the processor. These operations can have one operand or two, with these values coming from either the register file or from the immediate value from the instruction directly. The low power design of the ALU involves the gating the input signals to each of the separate components of the ALU. These inputs are gated using transmission gates. When a particular component of the ALU is not being used, the input to that component will be in a High Z state due to the output of the transmission gate. The operations supported by the ALU include add, subtract, compare, and, or, not, xor, logical shift, and arithmetic shift. The output of the ALU goes either to the data memory (in the case where the output is an address) or through a multiplexer back to the register file. The add, subtract, and compare operations are performed by the adder component. This adder component is essentially a 16-bit carry look-ahead adder, which performs the specified operation based on the input signals it receives from the ALU Control Unit. The shift operations are performed by the shift component. The shifter is capable of performing arithmetic shift left or right, as well as a logical shift left or right based on the inputs it receives. The difference between the logical shift and the arithmetic shift is that in the logical shift operation zeros are pushed into the vacated bit positions, whereas in the arithmetic shift operation, the vacated bit positions are replaced with the bit values that have been pushed out of the operand in a wraparound fashion. The XOR, OR and AND operations take two 16-bit values and performs respective bitwise operation on those two operands. The Not is an unary operation that takes only one 16-bit value and inverts all the bits.

- **ALU Control Unit:**

This unit is responsible for providing signals to the ALU that indicates the operation that the ALU will perform. The input to this unit is the 4-bit opcode and the 4-bit function field of the instruction word. It uses these bits to decide the correct ALU operation for the current instruction cycle. This unit also provides another set of output that is used to gate the signals to the parts of the ALU that it will not be using for the current operation.

(d) Memory:

This stage consists of the Data Memory module.

- **Data Memory:**

This module supports up to 64k words of 16-bit data words. The Load and Store instructions are used to access this module. When new data is to be written to the memory, the Mem_Write signal is asserted. When the Mem_Write signal is low, a read operation is performed for the given memory location.

(e)Write Back:

This stage consists of some control circuitry that forwards the appropriate data, generated by the ALU/MAC or read from the Data Memory, to the register files to be written into the designated register.

(f) Data Forward Unit:

This unit is responsible for maintaining proper data flow to the ALU and the Multiplier. The primary function of this unit is to compare the destination register address of the data waiting in the Memory and Write Back pipeline registers to be written back to the register file with the current data needed by the ALU or the Multiplier and forward the most up-to-date data to these units. It also performs the same operation with the Y-register data as well. By forwarding the data at the appropriate time, this unit makes sure that the pipeline works smoothly and does not stall as a result of data dependencies

V. POWER REDUCTION METHOD

The main power reducing method that has been explored in this architecture is clock gating. Clock gating is a method where the clock signal is prevented from reaching the various modules of the processor. The absence of the clock signal prevents any register and/or flip-flop from changing their value. As a result of this, the input to any combinational logic circuit remains unchanged, and thus no switching activity takes place in those circuits. Since, in CMOS circuits, most of the power dissipation results from switching activity, clock gating greatly reduces the overall power consumption. Clock power reduction is important in synchronous systems, since as was noted earlier, it can contribute to a large portion of the overall power budget.

Minimization of clock power falls in to several categories including clock distribution optimizations, clock gating, and low-swing clocking techniques. Gated clocking is a commonly applied technique used to reduce power by gating off of clock signals to registers, latches, and clock regenerators. Gating may be done when there is no required activity to be performed by logic whose inputs are driven from a set of storage elements. Since new output values from the logic will be ignored, the storage elements feeding the logic can be blocked from updating to prevent irrelevant switching activity in the logic. Fig shows an example of clock gating.

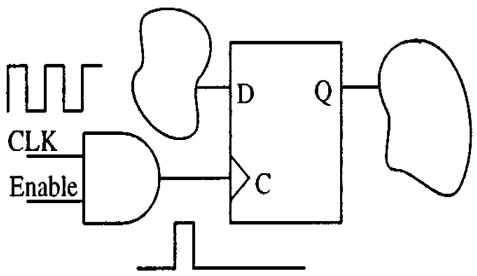

Figure.3.clock Gating

Clock gating may be applied at the function unit level for controlling switching activity by inhibiting input updates to function units such as adders, multipliers, and shifters whose outputs are not required for a given operation. Entire subsystems may be gated off by applying clock gating in the distribution network. This provides further savings in addition to logic switching activity reduction since the clock signal loading within the subsystem does not toggle. Overhead associated with generation of the enable signal must be considered to ensure that power saving actually occurs, and this generally limits the granularity at which clock gating is applied. It may not be feasible to apply clock gating to single storage elements due to the overhead in generating the enable signal, although self-gating storage elements have been proposed that compare current and next state values to enable local clocking . If the switching rate of input values is low relative to the clock, a net power saving may be obtained. In this design, mostly concentrated on the Multiplier, ALU, ALU Control Unit, and the Branch Adder for clock gating. The reason for this is that these are the biggest modules in our architecture in terms of number of logic gates,. Therefore, a significant improvement in power use can be achieved by gating these components. The Control Unit is responsible for generating the clock gating signal based on the current instruction. It then forwards these signals to the appropriate pipeline registers to make sure that the write operation is disabled to these registers for the duration of the execution cycle of the instruction. Also, in an effort to be more power efficient, the ALU has been built in a special modular fashion. Each of the operations supported by the ALU is performed by a different sub-module inside the ALU. Since almost 70% of the instructions in the instruction set use the ALU, it is beneficial to be able to operate only the part of the ALU needed by the current instruction and turn off the rest. Each of the modules of the ALU is preceded by a set of transmission gates that controls all the inputs to that module. When a module is needed, the transmission gates allow the data to pass through; otherwise they simply put that portion of the ALU in an electrically disconnected state. The ALU Control unit is responsible for generating signals that controls the blocks of transmission gates.

VI. FPGA DESIGN FLOW:

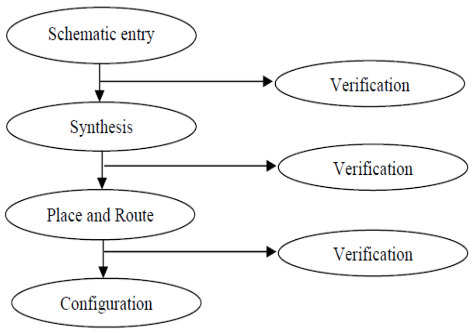

Figure .4. FPGA Design Flow

- **Schematic Entry:**

The design is entered into a synthesis design system using a hardware description language.

- **Synthesis:**

A netlist is generated using the VHDL code and the Xilinx synthesis tool.

- **Place and Route:**

The place process decides the best location of the cells and the best routing strategy for the given design and desired performance. The route process makes the connections between the cells and the blocks. This process was also completed with Xilinx ISE.

- **Verification:**

At each step of the design process, we verified our architecture using software simulation. We used ModelSim XE II software package for simulating our VHDL code.

VII. SIMULATION RESULTS .

Figure 5 below shows the result of gating the input signals of the ALU and the multiplier. At any given clock cycle, either the Multiplier, or only one module of the ALU receives the input signals. The rest of the modules are disconnected from the circuit by the use of transmission gates. As a result, we see that most of the modules are in HIGH Z state (represent by the blue lines) except for when they are needed to execute the current instruction. For example, since there are only two multiplication operations in the given sequence of instructions, the input to the MAC is only defined for only two clock cycles. Similar observations can be made.

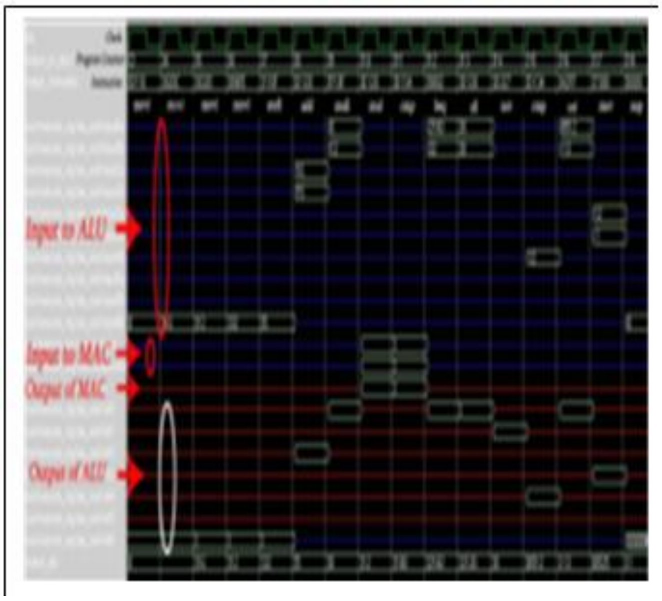

Figure. 5. Input Signal Gating for ALU and MAC

Figure 6 below shows the simulation result of clock gating for the ALU Control Unit and the Branch Adder. The inputs to the Branch Adder change only twice since it encounters only two branch instructions. When the instruction is something other than branch, the inputs remain unchanged as a result of clock gating. In a similar fashion, the inputs to the ALU Control Unit remain unchanged when the current instruction does not require the operations provided by the ALU (such as, multiplication and branch).

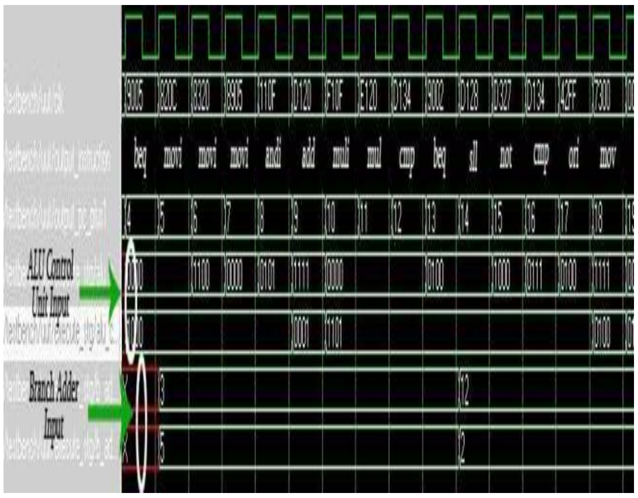

Figure.6. Clock gating for ALU Control Unit Branch Adder

VIII. CONCLUSION

Low-power design requires attacking the power dissipation problem at all levels of the design hierarchy. No single target will be sufficient to extract the efficiency required for future handheld products. Clock power optimizations will remain a challenge as higher frequencies and increased pipelining are applied to extract increased performance. Parallelism must be efficiently extracted without sacrificing the goal of low power. Software generation strategies that are based on power cost functions will be increasingly common in future systems. Even though a broad range of power reducing techniques have been proposed, the challenge still remains to integrate them into a design flow in which power plays as large a role as performance. Power consumption rate will continue to improve in embedded processors as technology will unravel new and more efficient ways to decrease power consumption.

IX. REFERENCES

[1]. Brown, Richard, “A Microprocessor Design Project in an Introductory VLSI Course”, IEEE Transactions on Education

[2].http://www.microsoft.com/windows/embedded/docs/Power_Management.doc

[3].IEEE Paper-Low-Power Consumption Architecture for Embedded Processor Yukihiro YOSHIDAt, Bao-Yu SONGH, Hiroyuki OKUHATAN, Taka0 ONOYEH, and Isao SHIRAKAWAfl

[4]. Hamblen, James and Furman, Michael, Rapid Prototyping of Digital Systems, 2nd edition, Boston: Kluwer Academic Publishers.

[5]. “Introduction to Embedded Processors”,http://www .cs .ucsd.edu/classes/sp02/cse291_E/slides/armlect.pdf

[6]. “A Microelectronics Primer”, http://www. cmc.ca/ about/corporation/plan/Module5/appendix5a.html

[7]. “ECE 252 / CSE 252 Digital Systems Design Lecture 2”, http://www.engr.uconn.edu/~chandy/ece252/252ln02 .pdf

[8] .Hamacher, Vranesic, and Zaky. Computer Organization, 5th edition, NewYork: McGraw- Hill Companies, 2002.

[9]. Liao, and Roberts, “A High-Performance and Low-Power 32-bit Multiply-Accumulate Unit with Single-Instruction-Multiple-Data (SIMD) Feature”, IEEE Journal of Solid- State Circuits, Vol. 37, No. 7

[10]. Design of a Low–Power Embedded Processor Architecture Using Asynchronous Function Units by Yong Li, Zhiying Wang, Xuemi Zhao, Jian Ruan,Kui Dai Springer link