A Review on Instruction Set Architecture and Clock gating signals for Low Power Risc Processor

Asha Rani M[1], Parvathi S. J[2]
[1, 2] Department of Computer Science and Engineering
[1, 2] GSSSIETW, Mysuru

Abstract- *The battery life of portable electronics today makes power and energy consumption vital factors. This paper presents a Description of instruction set architecture design to reduce power consumption and clock gating signals to reduce the power utilization. The methods described involve increasing code density by using reduced instruction word lengths. An idea for further increasing code density is also presented.*

I. INTRODUCTION

Designs today increasingly move toward targeting portable and battery powered systems rather than traditional desktop systems, increasing the importance of low power consideration during the design process. For the increasing portable electronics demand today, low power design techniques not only increase the system's battery life, but can also decrease system cost by allowing for smaller, light weight packaging, smaller, low cost batteries, and a reduction in managing heat dissipation [2]. In order to lower power consumption, existing low power design techniques sacrifice performance, showing the general tradeoff between performance and power consumption. Low power design may sacrifice too much performance, making it important to consider the designs effect on performance as well as its effect on power consumption [1].

The design of most instruction set architectures today only consider performance, without considering power dissipation, resulting in designs that are generally power hungry. An instruction set architecture (ISA) designed for a portable system should consider lowering power consumption during the design process [5]. The specification of the ISA has a large impact on both hardware and software (compiler) designs, both of which have a large effect on power consumption. When designing an ISA, we define three main characteristics of the ISA: the register organization, the memory organization, and the instruction set [6].

Register organization defines the number of registers and their sizes [6]. The size of the register file in hardware design, and the number of memory accesses in software and compiler design are both determined by an ISA's register organization. A small to moderate size register file consumes less power per register access, but more memory accesses are required, increasing the power consumed from overall memory accesses. The number of memory accesses is reduced by using a larger register file (register windows), reducing the power consumed by overall memory accesses, but the power required per register access is significantly increased [1].

Memory organization defines the address space and addressablity of the ISA. The address space is the number of memory locations addressable, which plays a large role in determining register and bus width. The addressablity is the number of bits stored at each address [6]. For a larger addressablity, more power is consumed per memory access, but more data is transferred, reducing the power consumed by overall memory accesses. A memory organization with a smaller addressablity requires more memory accesses, but less power is consumed per memory access.

The instruction set specifies the list of opcodes, the addressing modes, and the instruction formats for the ISA. The list of opcodes define the instructions supported by the ISA, and the addressing modes define how operand values are obtained. Instruction formats specify the binary format for encoding instructions, given the opcodes and addressing modes. The instruction format specification specifies the instruction width, which is traditionally a fixed width for RISC designs [7]. The instruction width has a large effect on the resulting design's code density, which is explained in more detail in section 2.

This paper explores how reducing instruction word width and code compression techniques can lower power consumption. Section 2 describes code density and it's effect power consumption. Section 3 describes the idea of reducing code density by reducing instruction widths. Section 4 proposes an idea for further code compression. Section 5 discusses future work and concludes the paper.

II. CODE DENSITY

Code density is a measurement of the number of bytes of code required to implement an application. A higher

code density means that less memory is required to store program code, and less code density means that more memory is required to store program code [5]. The goal with code density is to improve the density of fixed length 32-bit RISC architectures and avoid the complexity of CISC architectures [7].

A complex ISA design (CISC design) generally has more complex instructions, resulting in a higher code density than a simpler ISA design (RISC design) [1]. In CISC design, the higher code density reduces the power consumed by instruction fetches, but increases the power consumed by required control logic and instruction decoding. The RISC approach reduces power consumed by its simpler instruction decoding and control logic, but results in lower code density. The power consumed by instruction fetches is generally more with the RISC approach, because more instructions are fetched for a given application [1] [2] [3].

III. INCREASING CODE DENSITY

One approach to save power in 32-bit RISC architectures involves increasing code density. The following two methods to increase code density involve reducing instruction word length.

3.1 Instruction Width Reduction

Instruction word length effects the power consumption per instruction fetch. For a larger instruction width, more power is consumed fetching that instruction. Less power is consumed during an instruction fetch of smaller width instructions [1] [2].

Reducing instruction width is one approach to increasing code density and decreasing the power consumption. Most general purpose processors today utilize a 32-bit architecture in order to meet memory address space requirements [1]. One approach to improving power for these 32-bit architectures is to reduce instruction widths to a fixed length of 16-bits [4] [5]. This approach is used by Hitachi's SuperH and Motorola's M·CORE architectures [7]. The size of memory read during a 16-bit instruction fetch is half of the memory read for a 32-bit instruction, which generally can reduce the energy consumed by that memory read by up to 50% [1].

The M·CORE architecture is an example of reducing the instruction width of a RISC design to save power. The M·CORE is a 32-bit Load/Store RISC architecture that attempts to achieve high code density by utilizing a 16-bit instruction set [2] [4] [5].

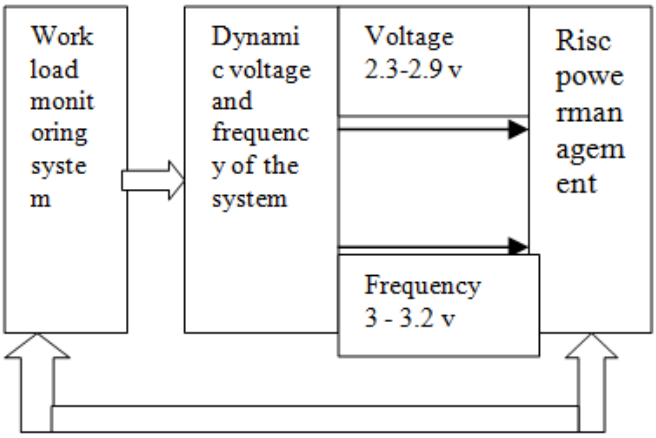
The M·CORE reduces instruction memory traffic by 40% of 32-bit instruction designs, reducing power consumed by instruction fetches [2]. Benchmark results show this design's resulting code density is generally higher than the code density of many CISC designs [4].

IV. RISC CLOCK GATING ARCHITECTURE

The power efficiency is the important constraint in designing portable computers, because lower power results in lower operating costs, lower fan noise, and lower cooling requirements. Therefore, designers of modern portable systems focus on increased system performance while reducing operating power consumption. Increasing the operating frequency, using more powerful, higher density chips achieves increased system performance, but increasing the performance level increases power consumption. Power consumption can be controlled during system operation depending upon the processing workload. This approach is called dynamic voltage frequency scaling [4,13]. According to the CPU workload, there are synchronous between the variation of the operating frequency and supply voltage [14]. For the cases that the workload is less than the minimum supply voltage requirements to drive the CPU, the enable signal of the clock gating technique will activate, during the activation of the enable signal the CPU consumes zero power. Hence the minimum energy reduction will obtain The clock gating technique identifies low processing requirement periods and reduces operating voltage with clock frequency (voltage-frequency scaling), resulting in reduced average operating power consumption. According to the CPU workloads, f and V can be reduced to its minimum levels or zero levels based on the software control. shows the block diagram of the CPU dynamic energy reduction using clock gating technique.

Block Diagram of Dynamic Power Reduction using Clock Gating technique. The workload monitor, predict the RISC activities based on a software implementation, while the voltage frequency scaling unit implemented in hardware, with the clock gating design of figure 3 inside or before the RISC. The overall block diagram controls the RISC energy reduction by - Controlling the supply voltage V. - Controlling the operating Clock frequency f. - Turning the RISC power off during idle periods. The voltage-scaling unit accepts a command from the workload monitor, and generates the required supply voltage and clock frequency, or turns the RISC power off .

4.1 Clock Gating Technique for Low Power RISC .



processor we analyze the RISC model first. Any IP core (except combinational circuit) can be modeled as an Finite State Machine (FSM) which includes several states: Idle, Ready, Run and so on, as shown in the box of 5. Each circle is a state and each arrow shows a transition from a state to another. The state and the transition will be mapped to the sequential circuit and the combinational circuit respectively by synthesis. When an IP core finishes the work, it enters the idle state (IS) and stay there until it accepts another request from the system bus. We call each of those states except IS working state (WS). Hence, all states in an IP core are classified to two classes: IS and WS. commonly used instructions that can be used in an accumulation manner.

Operands would be limited to register values or very small immediate values. This scheme hopes to improve code density overall for heavy, non-conditional, iterative computational applications.

V. CONCLUSION: FUTURE WORK

Future research for this idea takes three parts. The first thing to research is if a typical application can be broken down into enough sequential accumulator-based simple functions to improve code density. The code density achieved would have to be more than that of using strictly the uncompressed set of instructions. The second thing to look into would be the effect on performance. The power savings would have to be achieved without an unreasonable decrease in performance. The third thing to research is to define which functions of the uncompressed set would be encoded into the compressed 8-bit set. Instruction encoding for both the compressed 8-bit set and the uncompressed set of 16 and 32 bit instructions need to be defined before researching both the effect on performance and the code density achievable.

The main goal of further research is to determine whether or not this idea would be a realistic energy efficient solution for designing a low power instruction set architecture. Clock gating design can be used for low power applications to enhance the battery life of the devices. Power can be reduced further by applying the clock gating technique at a higher level of granularity.

REFERENCES

[1] T.D. Burd and R.A. Brodersen, Energy Efficient Microprocessor Design, Boston: Kluwer Academic Publishers, 2002.

[2] Bill Moyer, Low-Power Design for Embedded Processors, Proceedings of the IEEE, vol. 89, no. 11, 2001

[3] Flavius Gruian, Microprocessors: Low Power and Low Energy Solutions, Paper for the Advanced Issues in Computer Architectures course, 1999.

[4] Jeff Scott, Lea Hwang Lee, John Arends, Bill Moyer, Designing the Low-Power M·CORE Architecture, Proc. IEEE Power Driven Microarchitecture Workshop, pp 145-150, June 1998

[5] http://www.cs.utexas.edu/users/skeckler/cs395t_eaa/papers/mcore_lowpower.pdf

[6] S. J. Patel, W-M. W. Hwu, and Y. N. Patt, Instruction Set Architectures, In General, 2002

[7] Schlett, Manfred, “Trends in Embedded Microprocessor Design”, Computer, August 1998, pp 44-49.

[8] Phelan, Richard, Improving ARM Code Density and Performance, Thumb-2 Core Technology Whitepaper, June 2003

[9] http://www.embedded.com/showArticle.jhtml?articleID=1770128

[10] Deterministic Clock Gating Microprocessor Power Reduction, Hai Li, Swarup Bhunia, Yiran Chen, T. N. Vijaykumar, and Kaushik Roy 1285 EE Building, ECE Department, Purdue University <hl, bhunias, yc, vijay, kaushik>@ecn.purdue.edu

[11] A Clock Gating Circuit for Globally Asynchronous Locally Synchronous Systems, Jonas Carlsson, Kent Palmkvist, and Lars Wanhammar Department of

Electrical Engineering, Linkopings universitet SE-58 1
83 Linkoping Sweden