AN APPROACH FOR AUTOMATIC DESIGN OF APPLICATION SPECIFIC INSTRUCTION SET PROCESSORS (ASIP)

C.M.M. Mansoor
Computer Unit
Faculty of Arts and Couture,
South Eastern University of Sri Lanka.
cmm.mansoorr@gmail.com,

Abstract:

Today's hectic world is primarily dominated by electronics. The use of large and hard electronic devices is rapidly being replaced with simple, light and easy-to-carry ones. The past was primarily filled with the notion of performing varied tasks under one roof and the gadgets were built to serve that purpose through the introduction of General Purpose Processors (GPPs) into them. On the contrary, the present tendency is towards devices that are able to perform specific tasks. In the line of this modern concept, the world of embedded system is chiefly dominated by Application Specific Instruction set Processors (ASIPs) because they are geared to perform specific tasks without placing heavy burdens in many respects on the part of the users. Application Specific Instruction set Processors (ASIP), also known as customized processorsare processors designed for particular applications or, for a set of applications.They can be optimized for speed, chip area, and power consumption taking advantage of the flexibility of a synthesized semi-custom implementation. The development of application-specific instruction- set processors is currently the exclusive domain of the semiconductor houses and core vendors. This is due to the fact that building such an architecture is a difficult task that requires expertise in different domains. The main aim of this paper is to propose an approach that will automatically design an ASIP based on the application requirement, which is given as the input to the system. We propose to achieve the same by analyzing different types of RISCMIPS assembly instructions to map the corresponding target VHDL processor to customize maximize the memory access and components of the central processing unit and count the occurrences for the 32-bit RISC CPU based on MIPS. In this work, we also analyze MIPS instruction format, instruction data path, RISC CPU instruction set.

Keywords: *Application Specific Instruction set Processor (ASIP), Custom Processor, Embedded System, MIPS.*

Introduction

Today's hectic world is primarily dominated by electronics. The use of large and hard electronic devices is rapidly being replaced with simple, light and easy-to-carry ones. An Application Specific Instruction set Processor (ASIP) is a processor designed for a particular application or for a set of applications for a specific domain. One main objective of ASIP is its short turnaround time, which is determined by the ability to translate a given processor specification into hardware. An ASIP exploits special characteristics of application(s) to meet the desired performance, cost and power requirements. ASIPs [1] are a balance between two extremes: ASICs (Application Specific Integrated Circuit) and GPP (General

Programmable Processors). Since an ASIC is specially designed for one behavior, it is difficult to make any changes at a later stage. In such a situation, the ASIPs offer the required flexibility at lower cost than GPP. ASIP can be easily used in many embedded systems such as automotive control, household appliances, cellular phones, avionics etc. GPP are designed for general use. Many times it happens that specific applications need a certain mix which does not match the GPP resource mix. [2]

RISC (Reduced Instruction Set Computer) and CISC (Complex Instruction Set Computer) architectures have evolved in different applications. Most modern CISC processors are used in servers and desktop computers, while modern RISC processors dominate the embedded applications. Because of this historical difference, RISC processors have been optimized for low power systems and CISC processors for performance. The difference between RISC and CISC can lays on many levels, lots of plausible arguments are put forward by both sides, such as code density, transistor counts, memory compiler and decode complexity etc. RISC and CISC are two important philosophies in designing the computer architecture these two terms are compared with each other in various terms that are cycles, program, design, complexity, memory unit and pipelining etc. [3] [4]

In this study we have developed the algorithm to analyze the different types of MIPS32 RISC assembly instruction to map the corresponding target VHDL processor to customize and maximize the memory utilization.

Materials and methods

Research and development in the area of ASIPs has been flourishing for a couple of years. Numerous tool suites have been developed [5]. Apart from specific instruction set generation, customization of processor architectural features such as register and functional units, has been studied [6] for performance enhancement

The past few decades have seen a dramatic development in the area of ASIPs since numerous researches are being done in this regard. This has resulted in the development of tool kits. As the first step towards the generation of ASIPs is to create an instruction set that is appropriate to the given application. Numerous alternative designs can be created in accordance with the given instructions [7]. The research on the use of the automated tools in the generation of the instruction sets and design space exploration is actively on.

MIPS (Million instructions per second) is a RISC microprocessor architecture. The MIPS Architecture defines thirty-two, 32-bit general purpose registers (GPRs). Instruction Set Architecture (ISA) of processor is composed of instruction set and corresponding registers. Program based on same ISA can run on the same instruction set. MIPS instruction has been developed from 32-bit MIPSI to 64-bit MIPSIII and MIPSIV since it was created. To assure downward compatibility, every generation production of MIPS instruction directly extends new instruction

based on old instruction but not abnegates any old instruction, so MIPS processor of 64-bit instruction set can execute 32-bit instruction
MIPS instruction are divided into three types: R, I and J. Every instruction starts with a 6-bit opcode. In addition to the opcode, R-type instructions specify three registers, a shift amount field, and a function field; I-type instructions specify two registers and a 16-bit immediate value; J-type instructions follow the opcode with a 26-bit jump target.

The following are the three formats used for the core instruction set:

Type	Formats(bits)					
R	opcode(6)	rs(5)	rt(5)	rd(5)	shamt(5)	Funct(6)
I	opcode(6)	rs(5)	rt(5)	Immediate(16)		
J	opcode(6)	address(26)				

The MIPS single-cycle processor performs the tasks of instruction fetch, instruction decode, execution, memory access and write-back all in one clock cycle. First the PC value is used as an address to index the instruction memory which supplies a 32-bit value of the next instruction to be executed

In this section, we first give an overview of our design methodology and then present the algorithms used in the design. We developed the algorithm which is used to analysis the different type of mips instruction set and filter the types of instructions and count the number of occurrences .This algorithm eliminates the redundant operation opcode in application and matching the corresponding faction in the target VHDL processor. When another input application is fed the same process takes place. This means that the tool becomes automated and keeps producing customized processors.

Results

With above methodology we input mips32 assembly programming to analyze the occurrences of type of instruction. It produce the following result depends on the input MIPS32 instruction set. All types of opcode are analyzed to target the desired VHDL processor.

```
1  # A[i] = A[i/2] + 1;
2      lw      $t0, 0($gp)       # fetch i
3      srl     $t0, $t0, 1       # i/2
4      addi    $t1, $gp, 28      # &A[0]
5      sll     $t0, $t0, 2       # turn i/2 into a byte offset (*4)
6      add     $t1, $t1, $t0     # &A[i/2]
7      lw      $t1, 0($t1)       # fetch A[i/2]
8      addi    $t1, $t1, 1       # A[i/2] + 1
9      lw      $t0, 0($gp)       # fetch i
10     sll     $t0, $t0, 2       # turn i into a byte offset
11     addi    $t2, $gp, 28      # &A[0]
12     add     $t2, $t2, $t0     # &A[i]
13     sw      $t1, 0($t2)       # A[i] = ...
14 # A[i+1] = -1;
15     lw      $t0, 0($gp)       # fetch i
16     addi    $t0, $t0, 1       # i+1
17     sll     $t0, $t0, 2       # turn i+1 into a byte offset
18     addi    $t1, $gp, 28      # &A[0]
19     add     $t1, $t1, $t0     # &A[i+1]
20     addi    $t2, $zero, -1    # -1
21     sw      $t2, 0($t1)       # A[i+1] = -1
```

Figure 01

```
1      lw      $t0, 0($gp)
2      srl     $t0, $t0, 1
3      addi    $t1, $gp, 28
4      sll     $t0, $t0, 2
5      add     $t1, $t1, $t0
6      lw      $t1, 0($t1)
7      addi    $t1, $t1, 1
8      lw      $t0, 0($gp)
9      sll     $t0, $t0, 2
10     addi    $t2, $gp, 28
11     add     $t2, $t2, $t0
12     sw      $t1, 0($t2)
13
14     lw      $t0, 0($gp)
15     addi    $t0, $t0, 1
16     sll     $t0, $t0, 2
17     addi    $t1, $gp, 28
18     add     $t1, $t1, $t0
19     addi    $t2, $zero, -1
20     sw      $t2, 0($t1)
```

Figure 0

Discussion

According to the result we obtain above, it can be seen that the all types of mips32 instruction are filtered without the any other register information and noted that all of them in one of three format and occurrences. Similarity values have not changed significantly with the dataset and therefore it is clear that the results do not depend on the datasets chosen.

Conclusion

Embedded processors are the key for rapidly growing application fields ranging from automotive to personal mobile communication, computation, entertainment, etc. The work presented in this paper to make target application in efficient way. This work was centered to a single cycle processor. In future, the direction of the work will be towards a multi-cycle processor and a pipeline- enabling processor.

References

[1] Liem, C.; May, T.; Paulin, P., "Instruction-set matching and selection for DSP and ASIP code generation.", In Proc. EURODAC-94, 28 Feb.-3 March 1994, pp. 31-37.
[2] Arthur Abnous and Jan Rabaey, "Ultra-Low-Power Domain-Specific Multimedia Processors. Proceedings of the VLSI Signal Processing Workshop." in *IEEE Spectrum*, Oct. 1996, p. 461470.
[3] Masood, Farhat; "RISC and CISC". http://arxiv.org/ftp/arxiv/papers/1101/1101.5364.pdf. Januray, 2011.
[4] Lorenzoni, R. K.; "Análise de Desempenho e Consumo Energético entre Processadores ARM e x86". December, 2011.
[5] N. Binh, M. Imai, and Y. Takeuchi. A performance maximization algorithm to design asips under the constraint of chip area including ram and rom size. In *ASPDAC*, 1998.
[6] J.-H. Yang, B.-W. Kim, et al. Metacore: an application speci_c dsp development system. In *DAC*, 1998.
[7] Atasu K., Pozzi L., and Ienne P. 2003. Automatic application- specific instruction-set extensions under microarchitectural constraints. In *Proc.40th DAC*. pp 256-261.