LEON2 TIMING PERFORMANCE IN AUTOMOTIVE, OFFICE AUTOMATION AND SECURITY APPLICATIONS

Heleodoro Rios, Jiaxin Guo, Bao Liu and Eugene John
Department of Electrical and Computer Engineering
The University of Texas at San Antonio
Kho147@my.utsa.edu

Abstract
This paper examines the timing performance of the LEON2 processor using MiBench benchmarks in the embedded application domains of automotive, office and security industries. By measuring the total run time of each benchmark and dividing the run time by the clock period, the total number of cycles could be determined. In order to run each benchmark and find their run time, the benchmark source codes were cross compiled using the BCC cross-compiler utility to get the desired machine code needed by the LEON2 processor SPARC architecture. The resulting binaries were directly placed into the processor data file RAM on the LEON2 microprocessor. Then QuestaSim 6.4c was initialized with a fixed clock period of 20 ns to simulate the LEON2 processor running each of these benchmarks. Since each benchmark varied in size and tested different functionalities, the results were diverse ranging from short to long simulation times. The longer being *basicmath* within the Automotive package with 227,063,029 cycles and the shortest being *stringsearch* with 555,807 cycles from the Office Automation package. This LEON2 processor simulation-based research project well compliments the undergraduate curriculum by given a deeper understanding on computer architecture and digital system simulation techniques. It further provides an excellent opportunity for the participants to improve research and development skills such as literature review, technical writing and presentation.

Introduction

Evaluation performance has turned in to a more critical parameter over time. Today technology is imaginable and is a subject only to great TV shows and books of science fiction from a couple of decades earlier. Today microprocessors have overpopulated the world we live on, have made us so reliably on their great applications and make industries today run smoothly. The market for microprocessors is demanding such as their specifications. There is a wide range of computation power and features from processor family to family, each with its own application purpose. The question to ask is how good a processor can perform doing its job? Today Benchmarks are a crucial step in the development of system-on-a-chip (SOC) design when considering a processor core speed for the chip and many other parameters such as power consumption. In this paper with look at *MiBench* which is a commercially representative embedded benchmark suite to evaluate the LEON2 processor clock cycle on the demanding automotive, office automation and security industries.

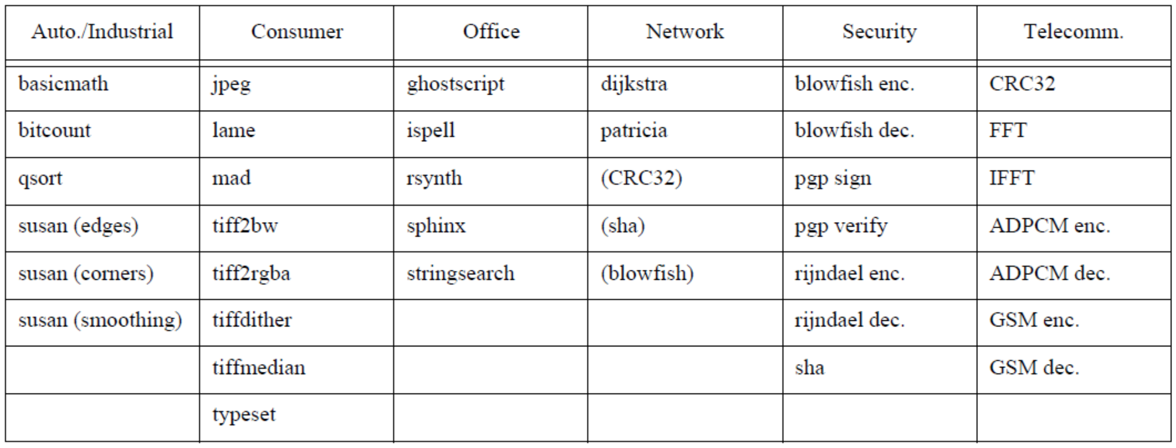
LEON2 processor

LEON2 is a highly configurable and a Very High Speed Integrated Circuit Hardware Description Language (VHDL) synthesizable SPARC V8 32-bit processor making it flexible and very suitable for embedded applications. Originally developed by the European Space Agency, now it

is an open source microprocessor owned by Gaisler Research which can be used freely and unlimited under the GNU Public License (GPL) and Lesser GNU Public License (LGPL).[1] In this study the use of the 1.0.30-xst version was used to run the MiBench benchmarks. Since the LEON2 processor design was conceived more than a decade ago, technical support has particularly decrease tremendously making it hard to debug or find appropriate tools with great community support. Making the task of cross compilation almost impossible.
For example the current version of RTEMS cross compiler is not supported with the 1.0.20-xst version of the LEON2 processor and the T-Sim debugging tool offered by Gaisler abandoned LEON2 a long time ago.

MiBench

MiBench is a free, commercially representative embedded benchmark suit composed of 35 embedded applications for benchmarking purposes. These benchmarks are divided into six suits with each suite targeting a specific area of the embedded market. [2] In Fig. 1, all six categories are listed under the application they are specific to. All the programs are available as standard C Source Code.



Auto./Industrial	Consumer	Office	Network	Security	Telecomm.
basicmath	jpeg	ghostscript	dijkstra	blowfish enc.	CRC32
bitcount	lame	ispell	patricia	blowfish dec.	FFT
qsort	mad	rsynth	(CRC32)	pgp sign	IFFT
susan (edges)	tiff2bw	sphinx	(sha)	pgp verify	ADPCM enc.
susan (corners)	tiff2rgba	stringsearch	(blowfish)	rijndael enc.	ADPCM dec.
susan (smoothing)	tiffdither			rijndael dec.	GSM enc.
	tiffmedian			sha	GSM dec.
	typeset				

Figure 1: MiBench Benchmark suit with their corresponding applications

In particular the *basicmath* and *bitcount* falls under the automotive and industrial control suite which is intended to demonstrate the use of embedded processors in embedded control systems. Particularly *basicmath*, strongly stresses the ability of a processor on basic math abilities and bit manipulations such as cubic function solving and conversion from degrees to radians calculations and *bitcount* tests the bit manipulation abilities of a processor by counting the number of bits in an array on integers. The *stringsearch* benchmark falls under the Office Automation suit that targets applications on text manipulation and office machinery like printers. Specifically this benchmark searches for given words in phrases using a comparison algorithm. The *blowfish* benchmark is under the security category that includes several common algorithms for data encryption, decryption and hashing. Particularly *blowfish* is a symmetric block cipher with a variable length key with ASCII text files as an input data set.

MiBench has many similarities to the most widely used benchmarks, in particularly it possess a similar instruction distribution compare to SPEC (Standard Performance Evaluation Corporation) [2], see Fig. 2.

Motivation

By examining the timing performance of the benchmarks on question. A comparable analysis between two slightly modify LEON2 processor could be used to verify if an increased in throughput and speed was obtained. This allows us to specifically observe how each internal component modification, such as a floating point unit, could affect the performance of the LEON2 processor.

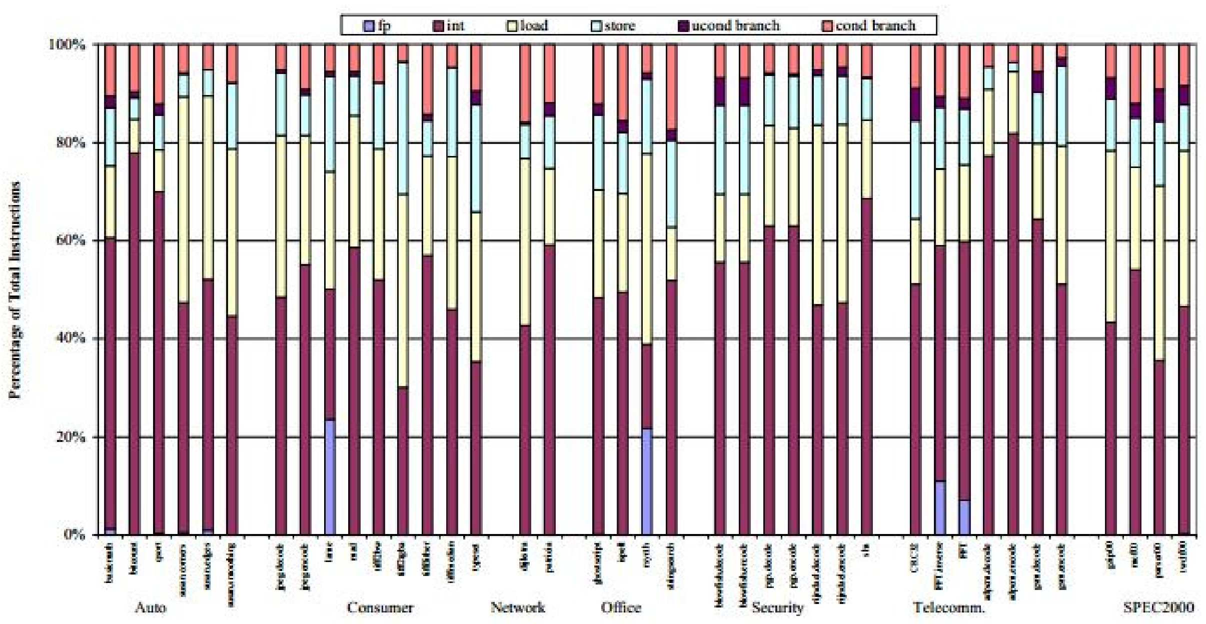


Figure 2: fp, int, load, store, uncond branch, cond branch instructions distribution of MiBench benchmark compare to SPEC 2006. Suits (left to right respectably): Auto, Consumer, Network, Office, Security, Telecomm are listed with their corresponding benchmarks in the vertical axis.

The rest of this paper is organized as fallows. Section 2 walk us to the methodology taken to run each of the MiBench benchmarks on the LEON2 processor, section 3 interprets the results obtain on the simulation process and section 4 will consist of the analysis and conclusion of the data gather.

Methodology

In order to successfully load the LEON2 processor with MiBench benchmarks, the default internals of the processor had to be modified. This was done using a Graphical User Interface (GUI) integrated with the LEON2 files to make easy the configurability of the processor. Table 1 shows the processor configuration use to run all benchmarks. After configuring the processor, the next step was to load our desired benchmarks into the memory of the processor. There were 3 ways to consider to do this. The first method was to synthesize the processor on a physical bard and load the benchmark source code onto the board using a serial connection and an S-RECord utility. This method was not consider for the lack of resources, especially of the absence of a physical board. The other two methods consisted of cross compiling the benchmarks into SPARC object code and directly place the binaries either into Read Only Memory (ROM) or Random Access Memory (RAM) of the LEON2. The later shown a more clearly advantage over the prior at run time so the RAM option was chosen. Throughout the rest of this paper the other two methods are abandoned and would not be discuss further. Although the ROM path could have being an attractive alternative if the benchmarks source code had being too big to fit into RAM. Once in RAM, QuestaSim 6.4c, a VHDL simulator, was use to simulate the processor.

Clock Frequency	50 MHz
Test-Bench Configuration	32-bit
Instruction cache	
Associativity (Sets)	1
Set Size (Kbytes/set)	8
Line Size (bytes/set)	32
Data cache	
Associativity (sets)	1
Set Size (Kbytes/set)	8
Line Size (bytes/set)	32
Multiplier Latency	5 cycles
Memory Controller	PROM

Table 1: LEON2 configuration

However, since the benchmarks had to be directly place into RAM. This limited the ability to test a more comprehensibly set of benchmarks within a suit. Hence only benchmarks that contain not input files were used in the study. Fig. 3 shows the list of non-input benchmarks with their corresponding clock cycles and simulation run time numbers.

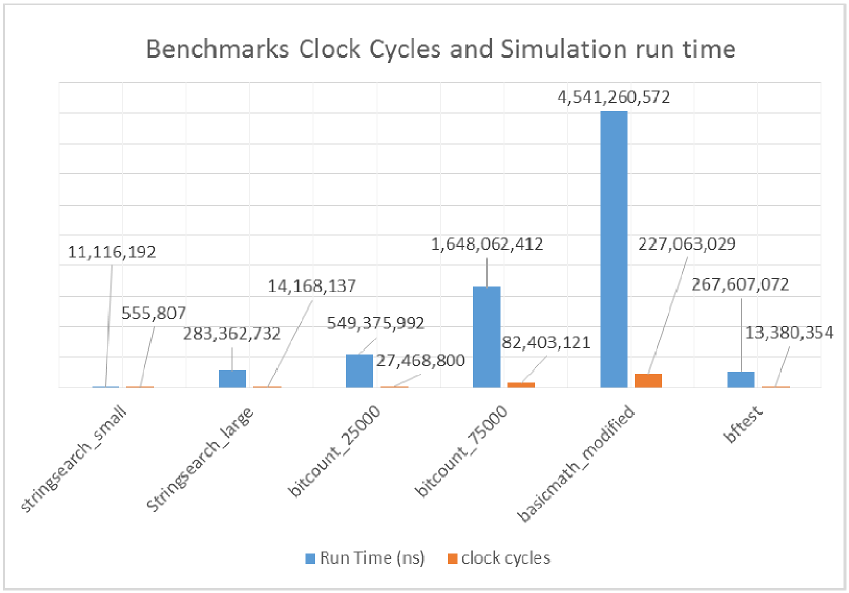


Figure 3: Non-input MiBench benchmark run time and total number of cycles. System clock period kept at 20 ns (50 MHZ)

Results and conclusion

As it could be observed on Fig 3, basic math took both the longest time for simulating and the greatest amount of clock cycles to complete. By contrast, the *stringsearch_small* benchmark took significantly less amount of processing and simulating time. Overall, all non-input benchmarks were successfully ran and useful data was gathered. This data produce in this study along with the flexibility nature to configure the LEON2 processor could be used to the advantage of the embedded system engineer in the automotive, office automation and security applications when measuring the effect and trade-off between different LEON2 configurations.

Acknowledgment

This work was supported in part by the National Science Foundation (NSF REU) under Grant No NSF CNS-1063106.

Reference

[1] LEON2 Processor User's Manual. Version 1.0.30. XST Edition. Aeroflex Gaisler

[2] MiBench: A free, commercially representative embedded benchmark suite by Matthew R. Guthaus, Jeffrey S. Ringenberg, Dan Ernst, Todd M. Austin, Trevor Mudge, Richard B. Brown, IEEE 4th Annual Workshop on Workload Characterization, Austin, TX, December 2001. < http://www.eecs.umich.edu/mibench/Publications/MiBench.pdf>