Simulation of Vehicular Application using Leon Environment

Abhijit S. Titarmare
Research scholar: Department of Electronics & Telecommunication Engineering.
G.H.Raisoni College of Engineering
Nagpur, India
abhijit.titarmare@yahoo.com

Dinesh V. Padole
Assistant professor: Department of Electronics Engineering.
G.H.Raisoni College of Engineering
Nagpur, India
dv_padole@rediffmail.com

***Abstract:* LEON3 is a 32-bit processor core conforming to the IEEE-1754 (SPARC V8) architecture. It is designed for embedded applications, combining high performance with low complexity and low power consumption. TSIM2 is a simulator that can emulate LEON3 based computer systems. It can also emulate user specific system-on-a-chip solutions by means of loadable modules. TSIM2 allows emulation of the LEON3 processor core with a completely user defined memory and I/O architecture. The emulated processor core communicates with the AHB module using an interface similar to the AHB master interface in the real LEON3 VHDL model. The AHB module can then emulate the complete AHB bus and all attached units.**

This paper presents simulation of vehicular application using LEON3 environment (TSIM2). The application includes wiper control and temperature control of AC in vehicle.

Keywords: LEON3 Processor, TSIM2, BCC, SPARC V8 Processor, RT Linux, elf –Snapgear.

I. INTRODUCTION

Vehicular electronics systems have changed radically over the last two-three decades. Early vehicular electronics systems were made up of switches, wires, relays and controlled motors or lamps. Modern vehicles however incorporate a wide range of sensors and actuators coupled to advanced electronics control system. The main motivation behind this is because of fast response, lower cost, reduced weight, new and innovative functionalities, most user-friendly and faster design cycles. The work presented here is the designing of vehicular control system, designed by LEON processor. The system is functionally tested using simulator of LEON processor.

A. Vehicular System:

In automotive electronics, electronic control unit (ECU) is a generic term for any embedded system that controls one or more of the electrical systems or subsystems in a vehicle. Modern vehicles have multiple ECUs figureure 1 show various controls in vehicle. Embedded software in ECUs continues to increase in line count, complexity, and sophistication. Managing the increasing complexity and number of ECUs in a vehicle has become a key challenge.

Types of electronic control units:

- Airbag Control Unit
- Door Control Unit
- Engine Control Unit
- Electric Power Steering Control Unit
- Temperature control
- Wiper Control
- Speed Control Unit

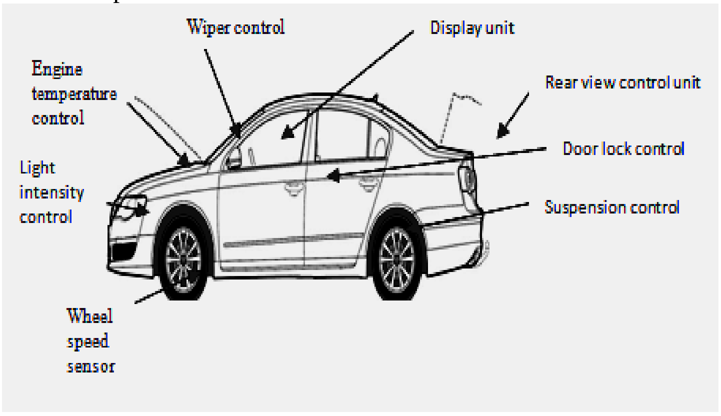


Figure 1: Vehicular control System

B. LEON3:

The LEON3 core has the following main features:

- Implements 32-bit Scalable Processor Architecture(SPARC-V8)
- 7-stage pipeline with Harvard architecture i.e. Separate Instruction and Data caches
- AMBA-2.0 AHB bus interface
- New modules can easily be added using the on-chip AMBA AHB/APB buses.
- Hardware multiplier and divider
- Synthesizable VHDL model
- Highly configureurable
- Suitable for system-on-a-chip (SOC) designs
- Availability of Full Source Code
- on-chip debug support and multiprocessor

LEON3 is a 32-bit processor core conforming to the IEEE-1754 (SPARC V8) architecture. It is designed for embedded applications, combining high performance with low complexity and low power consumption. The LEON3 integer unit implements the full SPARC V8 standard, including hardware multiply and divides instructions [1].

On-chip Debug support

The LEON3 pipeline includes functionality to allow non-intrusive debugging on target hardware. To aid software debugging, up to four watch-point registers can be enabled. Each register can cause a breakpoint trap on an arbitrary instruction or data address range. When the (optional) debug support unit is attached, the watch-points can be used to enter debug mode [1].

C. LEON3 Simulator:

TSIM is a generic SPARC architecture simulator capable of emulating ERC32- and LEON-based computer systems. TSIM provides several unique features:

- Accurate and cycle-true emulation of ERC32 and LEON2/3/4 processors
- Superior performance: +30 MIPS on high-end PC (AMD64@2.4 GHz)
- Accelerated simulation during processor standby mode
- Standalone operation or remote connection to GNU debugger (gdb)
- 64-bit time for unlimited simulation periods
- Local scratch-pad RAM (LEON3/4)
- Loadable modules to include user-defined I/O devices
- Non-intrusive execution time profiling
- Also provided as library to be included in larger simulator frameworks [4]

TSIM supports solaris-2.8, Linux, linux-x64, Windows 2K/XP, and Windows 2K/XP with Cygwin UNIX emulation.

LINUX support for LEON2 and LEON3 is provided through a special version of the Snap Gear Embedded Linux distribution. Snap Gear Linux is a full source package, containing kernel, libraries and application code for rapid development of embedded Linux systems. The LEON port of Snap Gear supports MMU and non-MMU LEON configureurations, as well as the optional V8 mul/div instructions and floating-point unit (FPU). The ports include symmetric multi-processing (SMP) support for LEON3 systems with multiple processors. A single cross-compilation tool-chain is provided which is capable of compiling the kernel and applications for any configureuration. BCC is a cross-compiler for LEON2 and LEON3 processors. It is based on the GNU compiler tools and the Newlib standalone C-library. The cross-compiler system allows compilation of sequential (non-tasking) C and C++ applications. It supports both hard and soft floating-point operations, as well as both V7 and V8 multiply and divide instructions [4].

II. VEHICULAR CONTROL SYSTEM DESIGN

In this paper, Auther has designed vehicular control system for applications Air Conditioner control, Wiper Control & Airbag control

Figureure 2 shows the block diagram of vehicular application system. The various sensors used here are temperature sensor, moisture sensor, speed sensor & pressure sensor. Temperature sensor will be placed inside the car to monitor the temperature in car. Moisture sensor is use for monitoring water droplets on the windshield of car. Speed sensor used to measure the speed. Pressure sensors used to measure the intensity of dash caused by accident.

1. Air Conditioner control:

With technology in the driver's seat today, new age innovations are making life easier for us. Let's take for example: cars. Over the past few years, one of the most common features in all automobiles has been air conditioning. In the past, air conditioning was a premium feature available only in premium cars or fully-loaded variants of cars. But increasingly off late, air conditioning has become standard equipment for almost all medium and large sized cars, not to forget that it is also very popular on commercial vehicles. As the technological advancements strive to provide better and better facilities to the car users, we find that car manufacturers are also working hard. So, now you have cars, which boast of automatic temperature control (ATC) systems that work to not just regulate cooling, but also regulate heating for complete passenger comfort throughout the year.

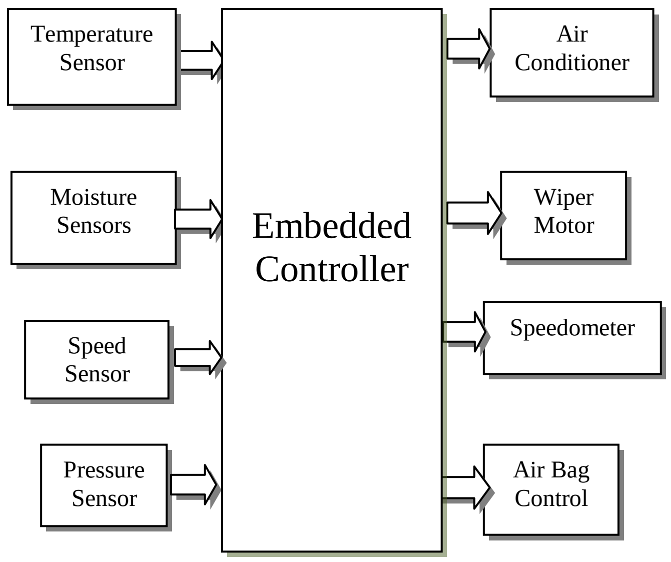
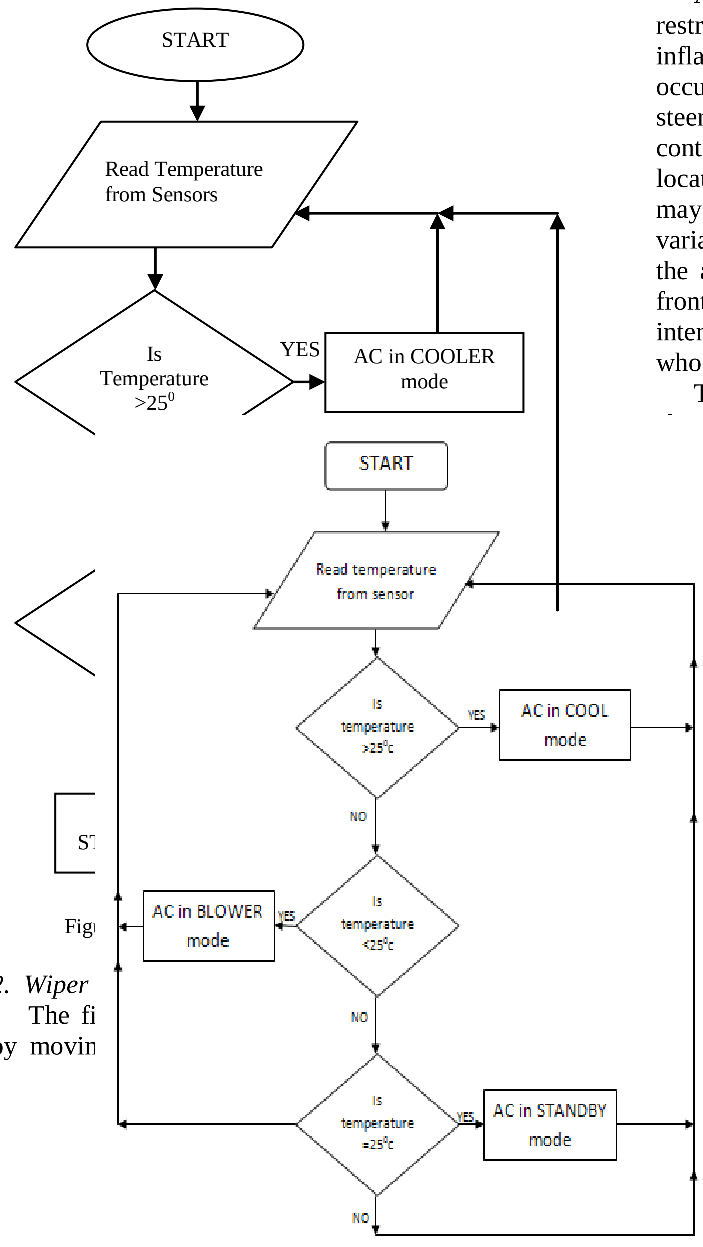


Figure 2: Block diagram of Vehicular control System

Air conditioner control system contains what we’ll refer to as two major component groups. These are: The Controller – The “brains” of the system; The Sensors – Supply the controller with input data; The Temperature controller is the “brains” of the system. It processes the data it receives from its sensors and issues output commands to the various devices it controls. Really, it has only one job: to do whatever it takes to keep the interior of the vehicle at a stable set temperature as selected by the vehicle occupants. Temperature controllers are essentially microprocessor devices, mini-computers. They gather input data from sensor; make decisions based on the sensor data, then issue appropriate commands to controlled devices. The temperature controller usually communicates with other electronic devices or modules. Very often, the various electronic modules in a vehicle will be connected together by a communication line, often referred to as the "data bus". They can share data back and forth many times per second.

Air conditioner control allows you to control the ambient temperature in your vehicle based on the driver's and/or passengers requirements. So how does Air conditioner control do this? Well the Temperature controller is programmed to contact your air conditioning and heating systems directly. Besides this, some Temperature controller come with a memory, so you can store different temperature settings based on your own personal needs. Climate control is a viable option to replace manual air conditioning and heating control system.

Figure 3 shows the flow chart of temperature control application. Initially we set the reference temperature to required value. When it found reference temperature to be less than the current value of temperature then air conditioner will automatically turn on in cooling mode, and if reference temperature is greater than the current temperature the heater mode is switched on and the temperature inside car is maintained to reference value, at this value air conditioner is in standby mode.



S

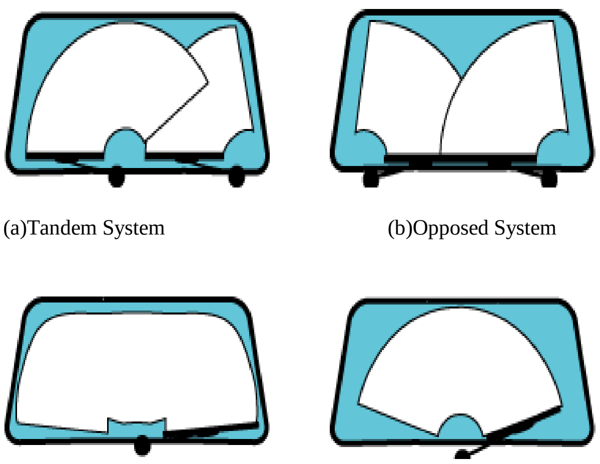
Fig

2. *Wiper*

The fi

by movin

application is designed to control loop automatically by sensing moisture in outside air.


(a)Tandem System (b)Opposed System

(c) Single Arm-Controlled (d) Single Arm

Figure 4: Different wiper blade schemes used by various cars

3. *Airbag control:*

An airbag is a vehicle safety device. It is an occupant restraint consisting of a flexible envelope designed to inflate rapidly during an automobile collision, to prevent occupants from striking interior objects such as the steering wheel or a window. Modern vehicles may contain multiple airbags in various sides and frontal locations of the passenger seating positions, and sensors may deploy one or more airbags in an impact zone at variable rates based on the type and severity of impact; the airbag is designed to only inflate in mild to severe frontal crashes. Airbags are normally designed with the intention of supplementing the protection of an occupant who is correctly restrained with a seatbelt.

The use of an airbag can protect your head, neck and areas. Airbags are fixed in by the vehicle facturers for the safety of the driver and passengers ling in the vehicle. Normally, they emerge out of eering wheel or from the dashboard, within a few econds of the collision. When your head hits the ;, the airbag starts deflating slowly, allowing you to it of the car.

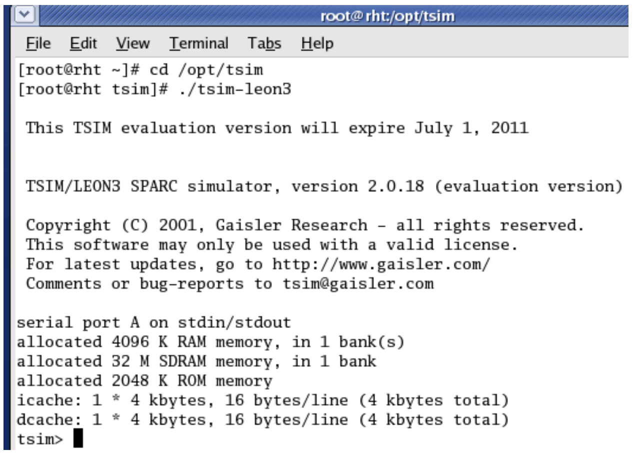
'hen the crash sensor in the car detects a collision, it a signal to the control module which deploys the ;. There are various types of crash sensors, like the ones which were placed in the front of the car (in rash zone area), and the latest micro machined ɜrometers that are installed inside the control le or the airbag brain. The micro machined ɜrometers actually measure the speed and severity of ɔllision. There are also sensors placed in the doors, eploying the side airbags. The front and the side rs only work with the front and the side airbags, ctively. An airbag installed in the dashboard or in ːeering wheel will only be deployed, if there is a end collision, such as in the case of a head-on ion or within 30 degrees from any side from the core of the car. The same rule applies to airbags installed at the sides of the car. The airbag is deployed when the car is hit at a certain angle. The ones on the left won't deploy, if the collision is on the right side and vice versa.

III. RESULTS

Author has developed programming for above applications & tested on the LEON processor simulator TSIM.

A. Loading of TSIM software:

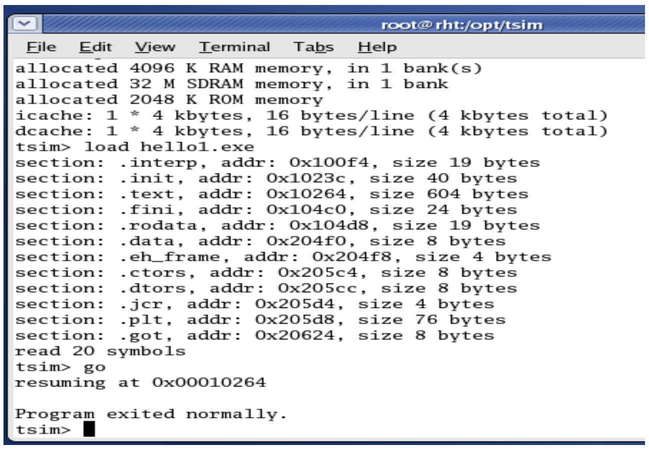
TSIM can operate in two modes: standalone and attached to gdb. In standalone mode, ERC32 or LEON applications can be loaded and simulated using a command line interface. A number of commands are available to examine data, insert breakpoints and advance simulation.



```
[root@rht ~]# cd /opt/tsim
[root@rht tsim]# ./tsim-leon3

 This TSIM evaluation version will expire July 1, 2011


 TSIM/LEON3 SPARC simulator, version 2.0.18 (evaluation version)

 Copyright (C) 2001, Gaisler Research - all rights reserved.
 This software may only be used with a valid license.
 For latest updates, go to http://www.gaisler.com/
 Comments or bug-reports to tsim@gaisler.com

serial port A on stdin/stdout
allocated 4096 K RAM memory, in 1 bank(s)
allocated 32 M SDRAM memory, in 1 bank
allocated 2048 K ROM memory
icache: 1 * 4 kbytes, 16 bytes/line (4 kbytes total)
dcache: 1 * 4 kbytes, 16 bytes/line (4 kbytes total)
tsim> █
```

Figure 5: TSIM Installation Window

TSIM dynamically loads libreadline.so if available on the host system; this will provide command history and completion with the tab-key. Iflibreadline.so is not found a simpler commandline will be used with no history and poor editing capabilities. If the file .tsimrc exists in the home directory, it will be used as a batch file and the commands in it will be executed at startup. TSIM is highly optimised, and capable of simulating ERC32 systems faster than real time. On high-end Athlon processors, TSIM achieves more than 1 MIPS / 100 MHz (CPU frequency of host). Enabling various debugging features such as watch points, profiling and code coverage can however reduce the simulation performance with up to 40%.



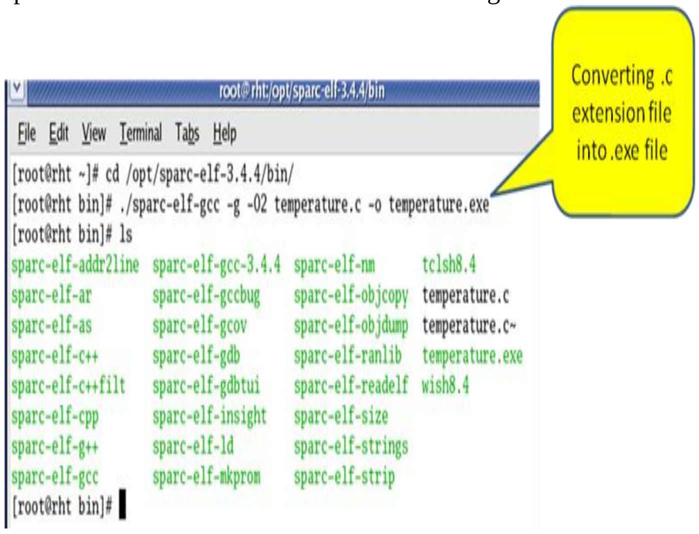
```
allocated 4096 K RAM memory, in 1 bank(s)
allocated 32 M SDRAM memory, in 1 bank
allocated 2048 K ROM memory
icache: 1 * 4 kbytes, 16 bytes/line (4 kbytes total)
dcache: 1 * 4 kbytes, 16 bytes/line (4 kbytes total)
tsim> load hello1.exe
section: .interp, addr: 0x100f4, size 19 bytes
section: .init, addr: 0x1023c, size 40 bytes
section: .text, addr: 0x10264, size 604 bytes
section: .fini, addr: 0x104c0, size 24 bytes
section: .rodata, addr: 0x104d8, size 19 bytes
section: .data, addr: 0x204f0, size 8 bytes
section: .eh_frame, addr: 0x204f8, size 4 bytes
section: .ctors, addr: 0x205c4, size 8 bytes
section: .dtors, addr: 0x205cc, size 8 bytes
section: .jcr, addr: 0x205d4, size 4 bytes
section: .plt, addr: 0x205d8, size 76 bytes
section: .got, addr: 0x20624, size 8 bytes
read 20 symbols
tsim> go
resuming at 0x00010264

Program exited normally.
tsim> █
```

Figure 6: after executing “HELLO WORLD” example in TSIM simulator

B. Temperature Control Application:

This paper contains temperature control strategies and methodologies using LEON3 processor simulator. In the temperature control application, one reference temperature is set and with this reference temperature the temperature inside car is checked. If it found greater or less then depending upon that temperature the mode of Air Conditioner will be selected either it is in cool mode or in heater mode. The performance of the system is evaluated through TSIM2 simulator for LEON3 processor. Figure below shows simulation results of temperature control application using TSIM2 simulator for LEON3 processor.

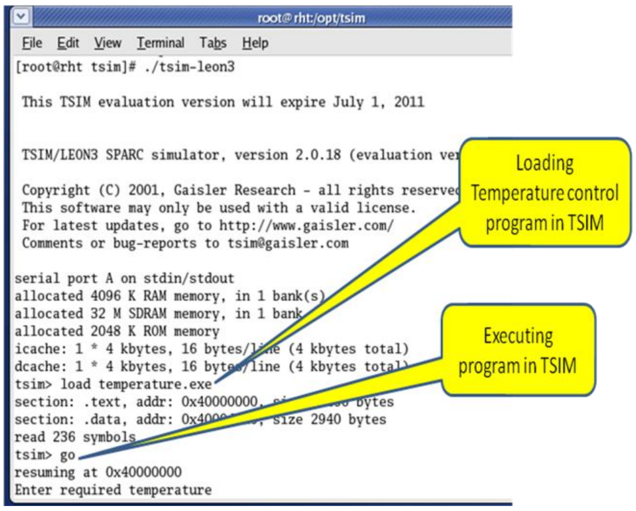
TSIM2 environment of LEON processor executes .exe files generated in sparc-elf software. Sparc is v8 instruction set which supports the TSIM simulator. The sparc-elf converter window is shown in the figure 7.


root@rht:/opt/sparc-elf-3.4.4/bin
File Edit View Terminal Tabs Help
Converting .c extension file into .exe file

```
[root@rht ~]# cd /opt/sparc-elf-3.4.4/bin/
[root@rht bin]# ./sparc-elf-gcc -g -O2 temperature.c -o temperature.exe
[root@rht bin]# ls
sparc-elf-addr2line  sparc-elf-gcc-3.4.4  sparc-elf-nm       tclsh8.4
sparc-elf-ar         sparc-elf-gccbug     sparc-elf-objcopy  temperature.c
sparc-elf-as         sparc-elf-gcov       sparc-elf-objdump  temperature.c~
sparc-elf-c++        sparc-elf-gdb        sparc-elf-ranlib   temperature.exe
sparc-elf-c++filt    sparc-elf-gdbtui     sparc-elf-readelf  wish8.4
sparc-elf-cpp        sparc-elf-insight    sparc-elf-size
sparc-elf-g++        sparc-elf-ld         sparc-elf-strings
sparc-elf-gcc        sparc-elf-mkprom     sparc-elf-strip
[root@rht bin]# █
```

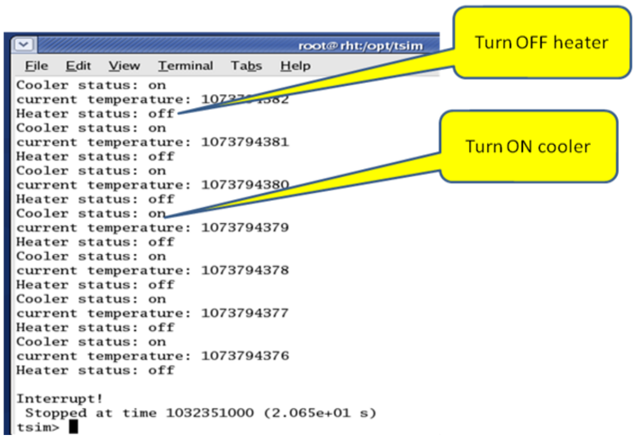
Figure 7: Sparc-elf window

Figure 8 shows TSIM simulator execution window, where temperature control application program file is loaded and executed using execution commands.


root@rht:/opt/tsim
File Edit View Terminal Tabs Help
Loading Temperature control program in TSIM
Executing program in TSIM

```
[root@rht tsim]# ./tsim-leon3

 This TSIM evaluation version will expire July 1, 2011


 TSIM/LEON3 SPARC simulator, version 2.0.18 (evaluation ver

 Copyright (C) 2001, Gaisler Research - all rights reserve
 This software may only be used with a valid license.
 For latest updates, go to http://www.gaisler.com/
 Comments or bug-reports to tsim@gaisler.com

serial port A on stdin/stdout
allocated 4096 K RAM memory, in 1 bank(s)
allocated 32 M SDRAM memory, in 1 bank
allocated 2048 K ROM memory
icache: 1 * 4 kbytes, 16 bytes/line (4 kbytes total)
dcache: 1 * 4 kbytes, 16 bytes/line (4 kbytes total)
tsim> load temperature.exe
section: .text, addr: 0x40000000, [illegible] bytes
section: .data, addr: 0x400[illegible], size 2940 bytes
read 236 symbols
tsim> go
resuming at 0x40000000
Enter required temperature
```

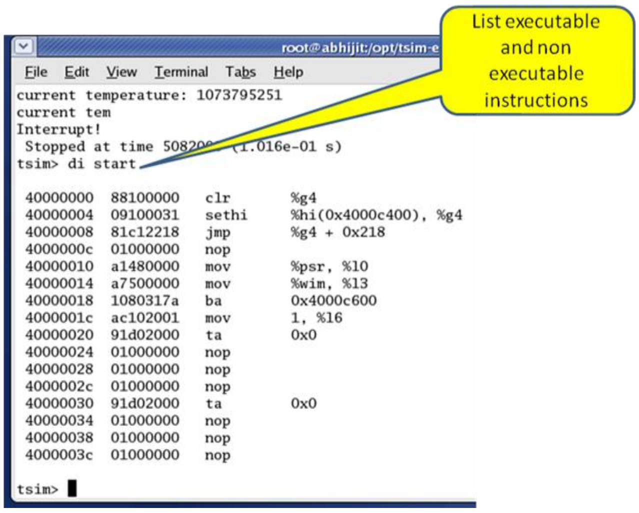
Figure 8: Loading of temperature control program on TSIM2 window



```
Cooler status: on
current temperature: 1073[illegible]382
Heater status: off
Cooler status: on
current temperature: 1073794381
Heater status: off
Cooler status: on
current temperature: 1073794380
Heater status: off
Cooler status: on
current temperature: 1073794379
Heater status: off
Cooler status: on
current temperature: 1073794378
Heater status: off
Cooler status: on
current temperature: 1073794377
Heater status: off
Cooler status: on
current temperature: 1073794376
Heater status: off

Interrupt!
 Stopped at time 1032351000 (2.065e+01 s)
tsim> █
```

Figure 9: Simulation result.

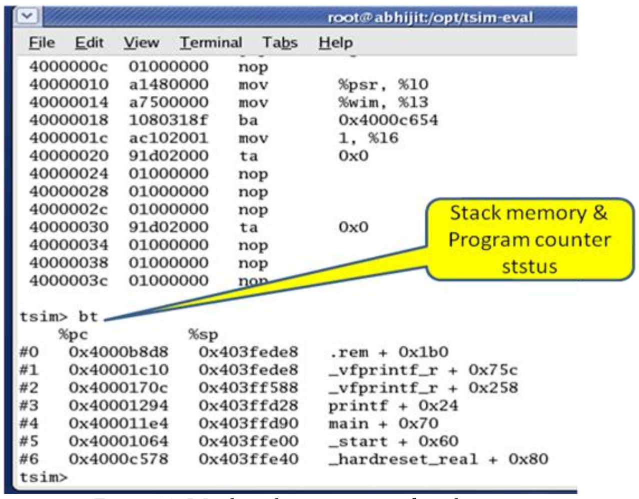
After executing the temperature control application, depending upon the status of current temperature simulator shows different mode of operation of air conditioner is shown in figure. 9.


root@abhijit:/opt/tsim-e
File Edit View Terminal Tabs Help
List executable and non executable instructions

```
current temperature: 1073795251
current tem
Interrupt!
 Stopped at time 50820[illegible] (1.016e-01 s)
tsim> di start

 40000000  88100000  clr    %g4
 40000004  09100031  sethi  %hi(0x4000c400), %g4
 40000008  81c12218  jmp    %g4 + 0x218
 4000000c  01000000  nop
 40000010  a1480000  mov    %psr, %l0
 40000014  a7500000  mov    %wim, %l3
 40000018  1080317a  ba     0x4000c600
 4000001c  ac102001  mov    1, %l6
 40000020  91d02000  ta     0x0
 40000024  01000000  nop
 40000028  01000000  nop
 4000002c  01000000  nop
 40000030  91d02000  ta     0x0
 40000034  01000000  nop
 40000038  01000000  nop
 4000003c  01000000  nop

tsim> █
```

Figure 10: Window showing list of executable instruction

Figure 10 & 11 shows executable and non executable instructions & status of program counter and stack pointer of LEON3 processor respectively.


root@abhijit:/opt/tsim-eval
File Edit View Terminal Tabs Help
Stack memory & Program counter ststus

```
 4000000c  01000000  nop
 40000010  a1480000  mov    %psr, %l0
 40000014  a7500000  mov    %wim, %l3
 40000018  1080318f  ba     0x4000c654
 4000001c  ac102001  mov    1, %l6
 40000020  91d02000  ta     0x0
 40000024  01000000  nop
 40000028  01000000  nop
 4000002c  01000000  nop
 40000030  91d02000  ta     0x0
 40000034  01000000  nop
 40000038  01000000  nop
 4000003c  01000000  nop

tsim> bt
     %pc          %sp
#0   0x4000b8d8   0x403fede8   .rem + 0x1b0
#1   0x40001c10   0x403fede8   _vfprintf_r + 0x75c
#2   0x4000170c   0x403ff588   _vfprintf_r + 0x258
#3   0x40001294   0x403ffd28   printf + 0x24
#4   0x400011e4   0x403ffd90   main + 0x70
#5   0x40001064   0x403ffe00   _start + 0x60
#6   0x4000c578   0x403ffe40   _hardreset_real + 0x80
tsim>
```

Figure 11: Window showing status of stack pointer

IV. CONCLUSION

This paper present LEON processor based vehicular control system. Auther has discussed methodology & programming for some vehicular application. The simulation result has presented.

REFERENCE

[1] Jiri Gaisler. *The LEON-3 Processor User’s Manual*, Version 1.0.20, February 2009 http://www.gaisler.com,

[2] *Snehal Dongare*, *Dinesh Padole*, *Dr. Preeti Bajaj*, “Design of Shared Resource Based Multicore Embedded Controller Using LEON Processor”, ICETET-10, Goa, India.

[3] *Dinesh Padole, Dr. Preeti Bajaj, “*Fuzzy Arbiter Based Multi Core System-On-Chip Integrated Controller For Automotive Systems: A Design Approach”, IEEE CCECE08, Canada

[4] TSIM product sheet

[5] BCC user manual Version 1.0.24, May. 2006

[6] BCC user manual Version 1.0.34, June. 2010

[7] P.M. Heysters, G.K. Rauwerda and L.T. Smit, “A flexible, low power, high performance DSP IP Core for Programmable Systems-on-chip.” in Proceedings of IP/SOC 2005”, Dec 2005