Development and Performance Evaluation of Hard Real Time Application on RISC V Processor Series: CASE study on Spacecraft On Board Software VYOMA

Kiran Desai[#], Dr. Vishwanath Y[*]
kiranastro@gmail.com[1] , vishwanath.y@reva.edu.in[2]
School of Computer Science and Engineering
REVA University
Bengaluru 560 064, India

Abstract- Space Industries, dealing with Mission or Safety critical On Board Softwares traditionally use Baremetal Cyclic executives, as they are traditionally proven and are more predictable in terms of behaviour. But as the computational loads, time complexities and parallelism is increasing manifold, its imperative to move towards RTOS based Application development on Single/Multi core processors. With increase in computational loads and parallelism in activities, Avionics industries started moving into multi core processors like Power PC, LEON4 based on ARINC 653 specifications which are proprietary. The operating systems used are VxWorks, RTEMS etc. With advent of Open RISC V architecture, it brings with it a flair of advantages like: openness, modularity, extensibility and stability. Many RISC V designs come with single/multi core architectures with open source RTOS support. In order to establish this tool chain into Space/Avionics industry, we need to develop a prototype, wherein a hard real time satellite application software is executed on RTOS Stack on RISC V series. Another target is to develop a library to allow portable application development on any flavor of RISC V.

In this paper we develop a sample hard real time application and evaluate its performance on an RTOS running on single and multicore RISC V processor. The responses of an RTOS (FreeRTOS) to a multithreaded application, show that Mission Critical Softwares can be coded on a Shakti-C Class Processors. Of course F class will be (extension of C class) will be the best, which will be the future course of work. This paper focuses on evaluating the feasibility of executing Spacecraft On Board Critical Softwares on RISC V Architectures. Current paper focuses on Single core processors.

Keywords: RTOS, Microprocessor, Hard Real Time Systems, RISC V, FPGA, Compiler, Control System, Closed Loop systems

I. INTRODUCTION

Hard Real Time systems work on the assumptions that whether a system succeeds in meeting the deadlines or not. It's a Boolean outcome unlike Soft Real time systems. Generally, HRT systems are Critical systems, failure to which may result in social/financial loss. Spacecraft On Board Softwares are such Mission/Safety critical Software wherein deadlines are of essence. In such systems, selection of Microprocessors and corresponding execution environment plays an important role. Various flavours of Microprocessor cores such as Power PC, ARM LEON3 (SparcV8) etc are extensively used, which are proprietary. Most of the applications in such hard real time systems will be developed in Bare-metal real time cyclic executive software to achieve the required HRT functionalities.

With the advent of Open RISCV architecture [4], it comes along with various advantages such as Layered and Extensible ISA; Common set of shared tools and Development resources; Flexibility to customize processor; Accelerate Time to market etc. With all these advantages, manufacturers and FPGA developers all over the world are developing various flavours of RISC-V Processors with varied applications. In this regard, we have decided to explore RISC V processors and use it for a CASE study wrt a Spacecraft On Board Software application [5] to prove that, RISC V implementation can be used for Space Applications. Study in [5] shows that due to modular and open nature of RISC V, it provides range of alternatives to proprietary solutions in the frame of new architectures for on-

978-1-5090-1489-7/16/$31.00 ©2016 IEEE

board embedded systems. RISC V will help in removing monopolistic nature of embedded systems and thereby encouraging academics too to enter into this field.

Hence, current work proposes usage of Shakti C Class processors and its compatible FPGA, Compiler tool chain. In succession to [5], we have put efforts in building a real time application for Spacecraft On board software. With the help of above mentioned tool chain we have developed a typical Attitude and Orbit Control System (AOCS) Application.

II.SHAKTI C CLASS PROCESSOR AND TOOL CHAIN:

The Shakti processor program was initiated by IIT-M RISE group in 2014. Its major aim is to bridge industry and academia by providing innovative and customized solutions without royalties. The Shakti processors designed by IIT(M) come in various classes based on the application at hand to be solved.

For any Development ecosystem to be built for Spacecraft On Board Software development, the following elements are necessary and were selected as shown in Fig.1; [1]

A. *Processor development board:*
ARTY A7 100T FPGA Board was selected.

B. Processor selection:
Shakti C Class Processor was selected. As its new upgraded version (upcoming) from RISE will be Fault tolerant class (F Class)

C. RTOS Selection:
Among existing RTOS flavours like RT Linux, FreeRTOS and Zephyr RTOS: FreeRTOS was selected for development

D. RISC V GNU GCC Compiler

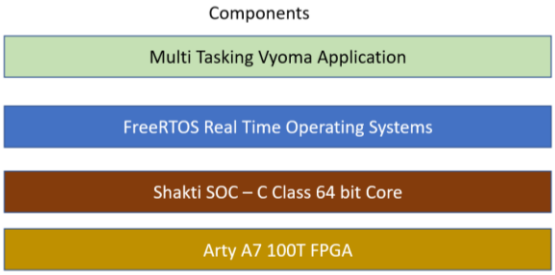


Fig. 1: Shakti C Class Stack

Next Section talks about the architecture design of the prototype to be developed and tested on RISC V Shakti C Class Processor. We have named it Vyoma Application.

III. SYSTEM DESIGN OF VYOMA APPLICATION

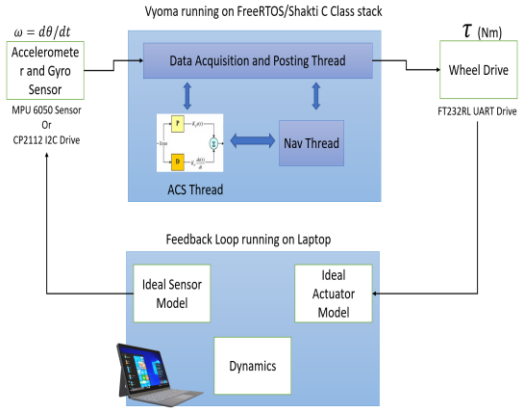


Fig. 2: Vyoma System Context Diagram

Vyoma System consists of the following architectural components:

A. Shakti C Class IP Core:

Shakti C Class IP is compiled using BlueSpec Compiler and Vivado Tool Suite. Generated bit/mcs file is loaded onto ArtyA7 100T FPGA. This FPGA is now configured as Shakti C Class Processor (64 bit single core processor).

B. Vyoma Spacecraft On Board Application:

This application is developed with a conceptual understanding of a multithreaded application running on a FreeRTOS OS stack. The Vyoma application is a multithreaded application consisting of three main threads:

1) ACS (Attitude Control System) Thread

   ACS Thread is responsible for Attitude Control of the Spacecraft, by;

   - Reading the Angular rate Sensor data acquired through Data Acquisition thread ω (radians/s)
   - Convert the rates into Quaternions using small angle approximation theorem

     $\Delta q = (\frac{1}{2})\dot{\omega}$
     $Q = Qbn \times \Delta q$
     $Q = correct(Q)\ (Sign\ correctness)$

   - Compare with commanded attitude of Spacecraft and generate error:

     $$\tau = -K_{rate} \times (\omega_{err}) - K_{prop} \times (2.0 \times Qerr)$$

   - Generate Torque signal
   - This thread runs at 16 ms frequency with 4 ms interval

2) Data Acq Thread: This thread is responsible to
   - Read Sensor (angular rate) from MPU6050 if real sensor is put in loop or CP2112 I2C driver is connected.
   - Also, Its responsible for posting Wheel torque to Wheel drive connected currently to UART FT232RL Drive.
   - This thread runs at 4 ms frequency with 2 ms interval

3) Nav Thread: This thread is kept for future Kalman Filter Design, NI etc which will help in providing Precise Navigation information to Vyoma
4) Software Timer Thread: This is a timer, which is initiated and is responsible for triggering all other threads in a timely manner. Timer runs at 1 ms frequency.

Some of important salient features of the application:

1) Based on Nyquist Sampling, Sampling is 2 times faster than the PD Controller computational rate.
2) As explained in the Fig3, after a brief initialization of all state variables and devices, a timer interrupt is initiated, which at regular intervals, based on the Thread frequency specified, triggers execution.

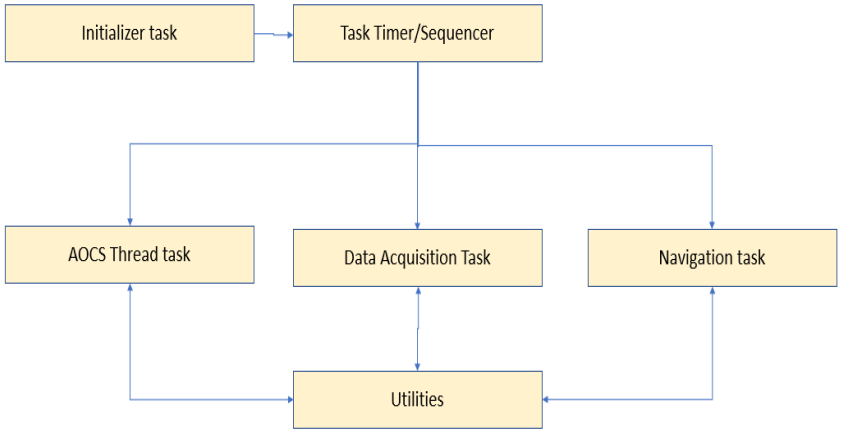


Fig. 3: Multi threaded architectural view

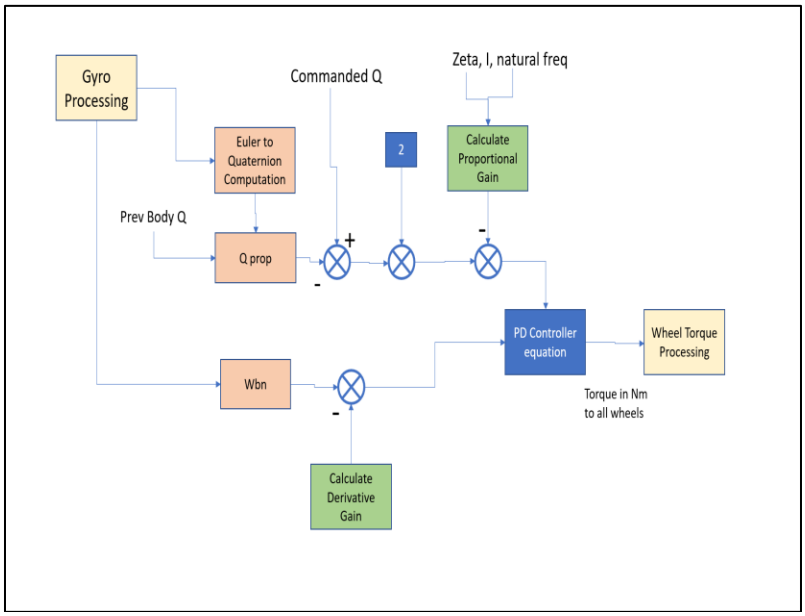


Fig. 4: Controller details of Vyoma Application

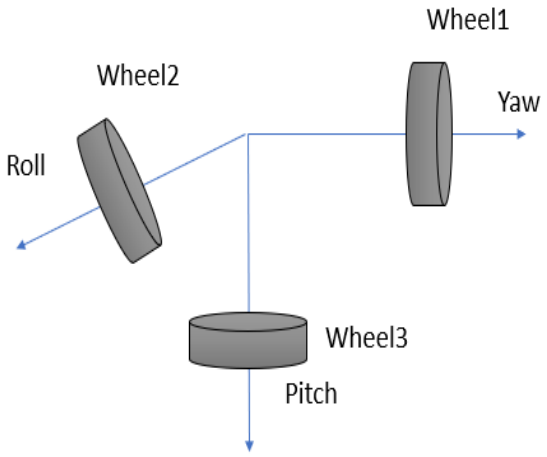
3) The Architecture developed is portable enough to be run on an POSIX based RT-Linux, of course its not yet experimented.

C. Spacecraft Dynamics:

Following are the simulations done at the Laptop, to simulate Spacecraft environment;

1) Actuator Model: Wheel Model

An Ideal Wheel is modelled with Torque distributed to all the three wheels (each mounted in each principal axis of Body)



UART driver provides wheel torque to Actuator Model, and based on Spacecraft Inertia (which is considered to be only along Principal axis Ixx,Iyy,Izz) is Integrated to body momentum. This momentum is integrated to form the angular rate and Wheel momentum. This Body rate so computed is fed to the sensor model

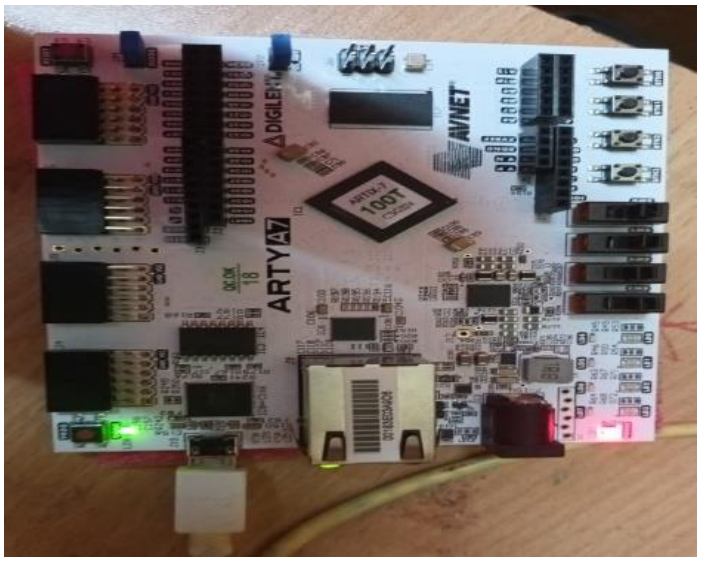
2) Gyro Sensor Model: This sensor model takes rates as input as passes through the sensor model, which takes care of various sensor characterstics such as;
   - Sensor Bias
   - Sensor Gaussian Random noise
   - Misalignment factors
   - Drift rate of Gyro

   Based on the model constructed from above parameters, we compute the Gyro rate and is fed through CP2112 I2C drive, thus closing the loop
   The Gyro model, provides rate along all the three axis in sensor frame. For ideal conditions, we have considered sensor frame and body frame are aligned with each other.

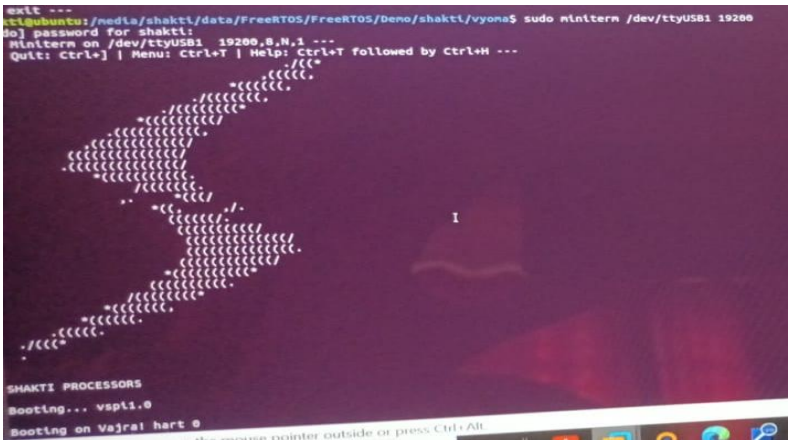
IV. SETUP AND RESULTS OF EXECUTION

The Entire hardware setup in its current state looks like below images;

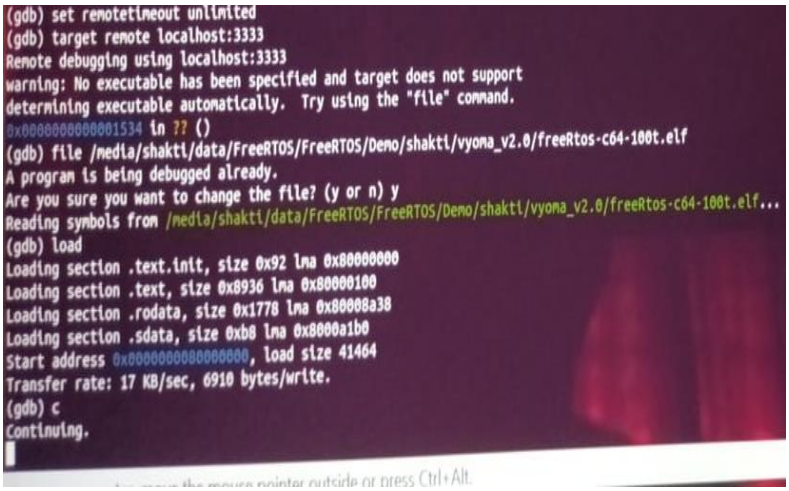
1) Development board powered on:



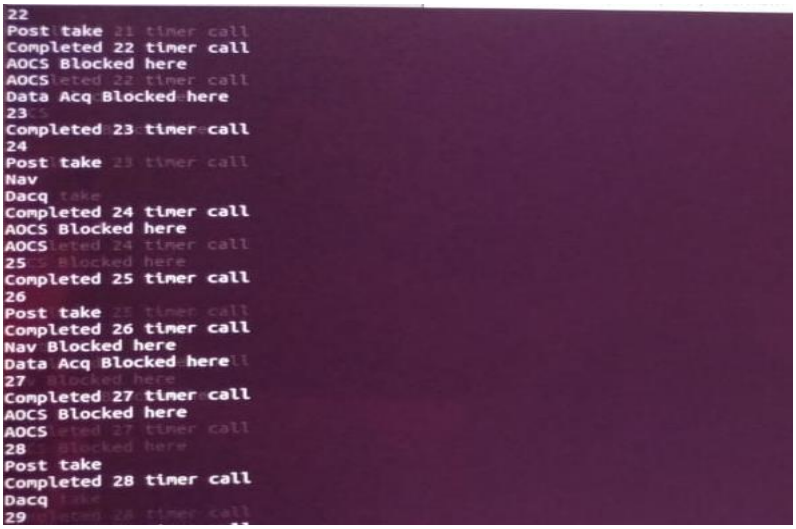
2) Shakti Powered ON:



3) Connecting through OpenOCD and GDB:

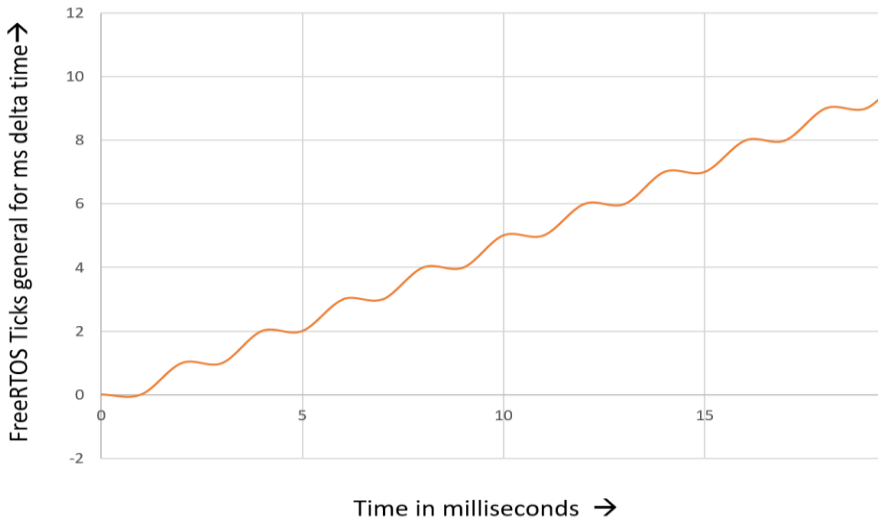


4) Execution of Threads:

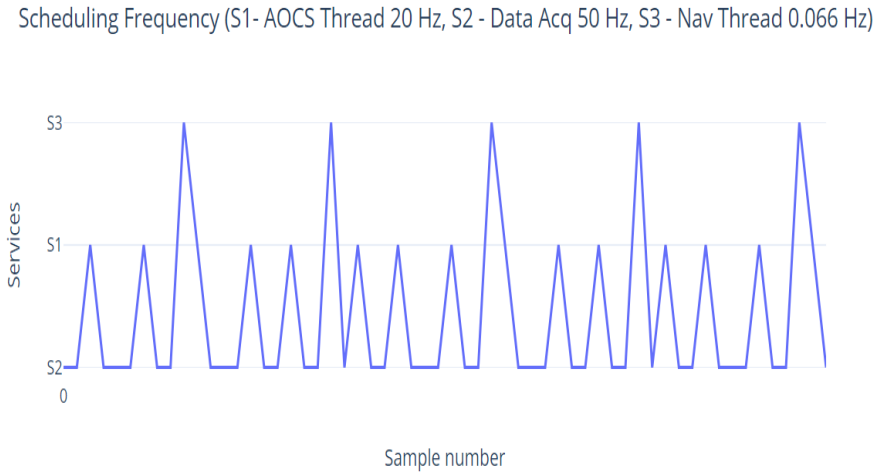


5) Time tick response of FreeRTOS:

As the threads are of minimum 4 ms revisit time, we have 2 ticks available from FreeRTOS which provides necessary trigger for threads to start. FreeRTOS [6] in current implementation provides, 1 tick/2ms. Efforts have been ON to make it much finer of the order of 0.5 ms, taking care future requirements.



6) Thread Response with Frequency 50 Hz, 20 Hz, 0.066Hz. Scheduler used in FreeRTOSConfig file for the same is Fixed Priority Preemptive Scheduling [6]. As can be seen, since Service S1 is of higher priority, pre-emption of S3 takes place as soon as S1 frequency has come into effect. With the above results, its imperative that RISC V and FreeRTOS combination is providing suitable environment for Engineers to use the tool chain. Further closed loop studies are under way to prove it effectiveness furthermore.
7)



V. FUTURE WORK

Current work focussed on setting up architectural and detailed design of Space craft On board Software on Free RTOS RISC V stack on a single core on an experimental basis. Future work encompasses:

1) Portability prototype development between RT-Linux and FreeRTOS
2) Performance enhancement and evaluation on Multi core Shakti Processors [2]
3) Performance evaluation on Fault Tolerant F-class processors
4) Development of comprehensive Guidance and Navigation Control Software for Satellites wrt Academic purpose

VI. CONCLUSIONS

With the above results, we can tell that the Vyoma architecture defined over RISC V - FreeRTOS stack for a single core is able to meet hard deadlines as specified by the application. Hence, We can consider RISC V C class processor and FreeRTOS stack suitable for Hard Real Time Applications.

ACKNOWLEDGMENT

Authors deeply acknowledge the support by Director, Reva School of Computer Science and Engineering and the entire academic team. We are also extremely thankful to Vice Chancellor in going ahead with this initiative.

REFERENCES

[1] Neel Gala, G. S. Madhusudan, Paul George, Anmol Sahoo, Arjun Menon, V. Kamakoti (2018, Sept). SHAKTI: An Open-Source Processor Ecosystem. In Advanced Computing and Communications

[2]Farhang Nemati, Johan Kraft and Thomas Nolte, Mälardalen Real-Time Research Centre, Mälardalen University, Box 883, 72123, Sweden, "Towards Migrating Legacy Real-Time Systems to Multi-Core Platforms", IEEE 2008

[3] Hannu Leppinen, Aalto University, Espoo, Finland, "Current Use of Linux in Spacecraft Flight Software", IEEE 2017

[4] Periasamy, K., Periasamy, S., Velayutham, S., Zhang, Z., Ahmed, S. T., & Jayapalan, A. (2022). A proactive model to predict osteoporosis: An artificial immune system approach. *Expert Systems*, *39*(4), e12708.

[5] Jimmy Le Rhun, Vicente Nicolau, Antonio Garcia-Vilanova, Jan Andersson, and Sergi Alcaidee, "DE-RISC: LAUNCHING RISC-V INTO SPACE", EUROPEAN WORKSHOP ON ON-BOARD DATA PROCESSING (OBDP2021), 14-17 JUNE 2021

[6] FreeRTOS Reference Manual, Verision 10.0.0, Issue 1