UTTUNGA-02

User Guide

V 1.4

1. Revision history

Version	Date	Description	Author
1.0	08th Jul, 2024	Initial version	Madhan Muruganantham
1.1	24th Mar, 2025	Changed document title to “Uttunga-02 Software User Guide”. Added section for update of SD card image	Madhan Muruganantham
1.2	02nd Apr, 2025	Updated for the latest release board Added compiler guide	Madhan Muruganantham
1.3	17th Apr, 2025	Updated CPML static and shared build flags Added detailed steps in preparing SD card	Madhan Muruganantham
1.4	05th May, 2025	Added hardware and start up details	Madhan Muruganantham Ravi Raju

2. Table of contents

3. Introduction

Calligo technologies TUNGA (Technology for Unum-based Next Generation Arithmetic) is the Revolutionary World's First Posit-enabled RISC-V CPU for General Purpose Computing. Posit is a system, when used in silicon, claims to deliver better performance and accuracy for high-performance applications – specifically those that involve significant mathematical computations.

With a distinctive market differentiator, powered by TUNGA, Calligo's accelerator Card UTTUNGA, is poised to deliver superior computational performance and accuracy by integrating advanced hardware combined with robust software acceleration.

UTTUNGA-02 is next generation single board based accelerator card with much simplified setup and higher performance.

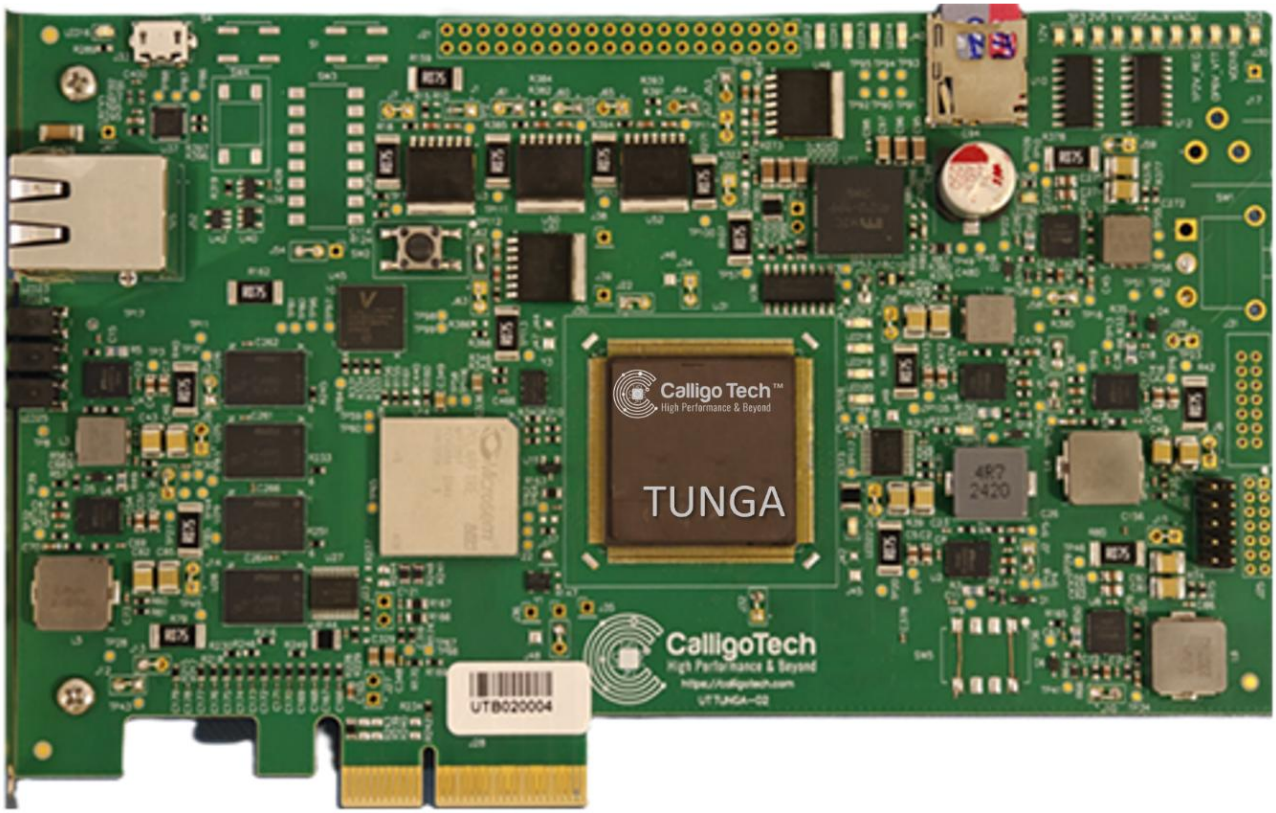
Hardware section of this document explains usage details and the steps to be followed for setting up the hardware, powering up the board and booting Linux OS.

Software section of this document explains the steps to be followed for access of the UTTUNGA card along with software stack that is shared with early adopters of the technology and partners working closely with Calligo in the journey towards next generation arithmetic.

4. Package Contents

The package contains:

- UTTUNGA-02 board
- SD Card
- USB Cable



5. Hardware Assembly

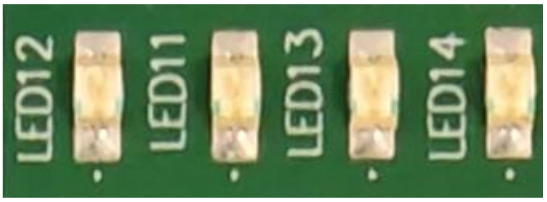
UTTUNGA accelerator card can be used on any system supporting PCIe interface. The card can be plugged into any slot that supports PCIe 3.0 Gen2 (5.0 Gbps).

6. Power Up and Booting

Once the accelerator card is inserted in the system and the system is powered up, the card powers up.

Power on reset kicks in at the time of card powering up. Stage wise, the card powers up. At first FPGA is programmed. Successful programming is indicated by the following LED pattern:

LED 12	LED 11	LED 13	LED 14
ON	OFF	OFF	OFF



Once FPGA is programmed, DDR and PCIe are initialized. After this, TUNGA CPU start the Linux booting process.

Linux booting can be observed by connecting the USB cable to the board Micro B type connector (J33 – see picture below) and other end connected to the machine where an application to emulate a terminal (like Tera Term or picocom) can be run. Follow the settings below. Typically cable with one end USB-A and other end Micro B type is used for the purpose, with USB-A side connected to the machine:

COM Port: Always Connect to the Enhanced Port (Port number varies by system/server).

Baud Rate: 115200

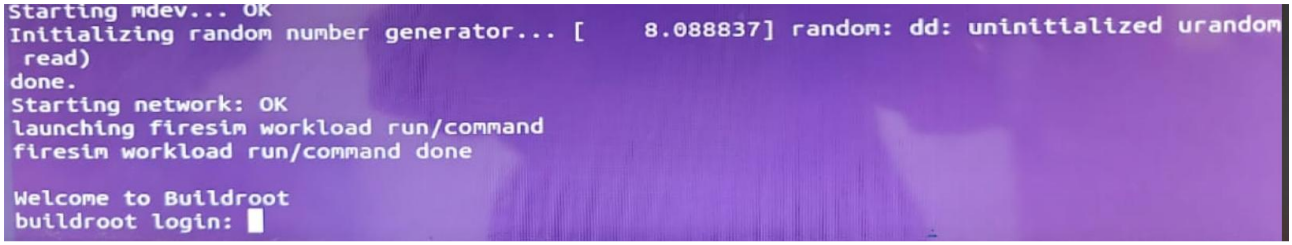
Data Bits: 8

Parity: None

Stop Bits: 1

Flow Control: None

After successful booting you should see the login prompt.



Login with the credentials provided to work from device side.

The PCIe will be initialised and can be accessed by host side.

Please refer to the Software Section (Creating and updating SD card image) for details on using the card as POSIT accelerator.

7. UTTUNGA Hardware Design

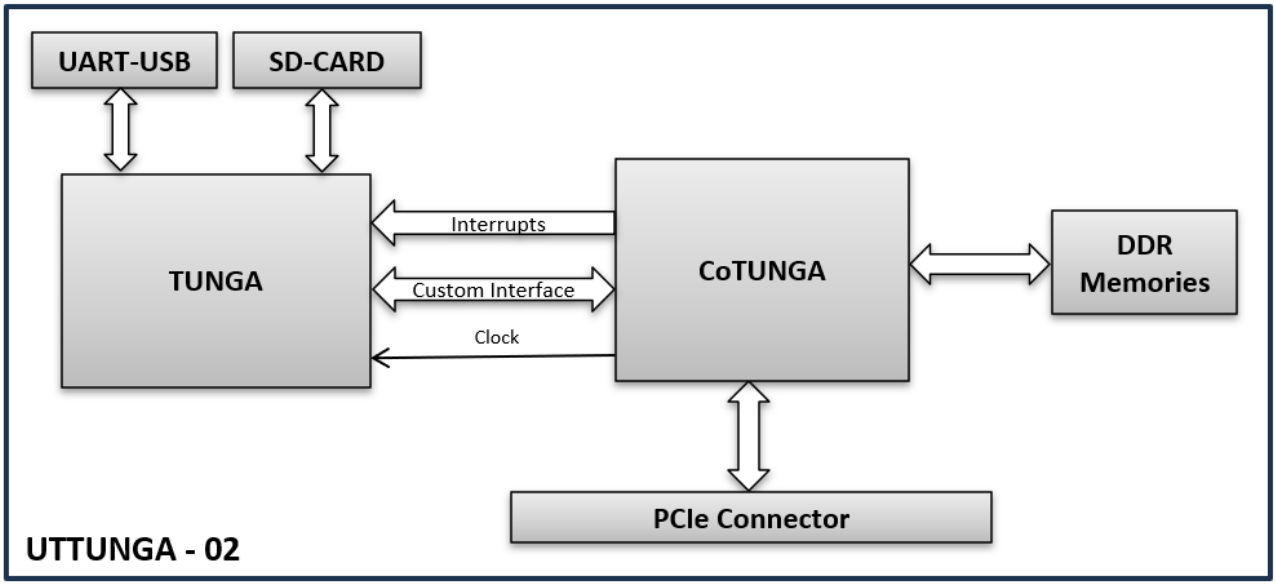
TUNGA 1.0 – The RISC V based Octa core CPU from Calligo Tech is designed to boot Linux OS and provide POSIT based accelerator. The hardware solution is implemented on the PCIe card. While TUNGA implements the CPU core, the chip is designed to work with an add-on chip named CoTUNGA. CoTUNGA implement controllers that extend TUNGA functionality. CoTUNGA is implemented on MicroChip PolarFire FPGA. The hardware is implemented on a PCIe Card UTTUNGA. This section provided the design details of UTTUNGA.

TUNGA and CoTUNGA are connected through a proprietary bus through which TUNGA accesses the memory space. The memory space is decoded in CoTUNGA to map to DDR4 memory. CoTUNGA implements interface block that connects proprietary bus to DDR 4 controller. DDR 4 controller connects to the DDR4 memory.

CoTUNGA also implements PCIe that enables the Host processor to access the DDR4 memory. Host and TUNGA uses the DDR4 memory for exchange of data to implement POSIT acceleration.

TUNGA, CoTUNGA, DDR 4 memories along with the associated circuitry and PCIe connector is implemented in the UTTUNGA board which essentially the POSIT accelerator.

Below figure provides the overview of UTTUNGA 02 accelerator card.



7.1. Address Map

TUNGA is the local master that can access 4GB of memory space. Refer the following table for address space:

Master	Slave	Address
Device (TUNGA 1.0)	Peripherals internal to TUNGA	0000 0000 to 7FFF FFFF
Device (TUNGA 1.0)	DDR 4	8000 0000 to FFFF FFFF

Host can access the UTTUNGA resources through the PCIe interface. Refer the following table for address space:

Master	Slave	Address
Host	PCIE Registers	0000 0000 to 0FFF FFFF
Host	DMA	1000 0000 to 1FFF FFFF
Host	DDR 4	F000 0000 to FFFF FFFF

7.2. TUNGA CPU

TUNGA is an octa core RISC V based CPU designed to boot Linux OS with the following external interfaces:

- Proprietary interface to FPGA
    - Access to 2BG DDR memory is provided through this interface
- SDCard interface
    - Linux Image is loaded from the SDCard through this interface

- UART Interface
    - Interface used for Linux boot messages and login

8. CoTUNGA

CoTUNGA is implemented in Microchip PolarFire FPGA (MPF100T-1FCVG484E). Following are the interfaces on CoTUNGA:

- Proprietary interface to TUNGA
    - Access to 2BG DDR memory is provided through this interface
- DDR interface
    - 4GB DDR4 SDRAM MT40A1G8SA-062ER (2GB Mapped to TUNGA CPU)
- PCIe Interface
    - PCIe 3.0 Gen2 (5.0 Gbps)

9. Creating and updating SD card image

9.1. Preparing a new SD card

- All the steps mentioned in this section are necessary for a new SD card. For update of the image in an existing card follow steps in the next section.
- Copy Script file from Calligo's FTP server.
    - scp -r username@ip_address:/home/ftpuser/Calligo_SDCard_Setup/
- Change directory path to Calligo_SDCard_Setup/linux_images.
    - cd Calligo_SDCard_Setup/linux_images
    - Connect SD card to the HOST machine and pass /dev / partition to the below command. In my case it is "/dev/sdc"
    - ./SD_Card_Setup.sh -d /dev/sdc
    - The above command creates two partitions on the SD Card.
- Remove and re-connect SD card to the Host machine.
- Format the second partition using the following command.
    - sudo mkfs.ext4 /dev/sdc2 (sdc2 is where the SD card device partition 2 is mounted)
- Remove and reconnect the SD card again. Now users can make use of the second partition to copy binaries or other dependency files.

9.2. Copying content on SD card

- Run the following command to flash the OS image on partition 1.
    - cd Calligo_SDCard_Setup/linux_images
    - sudo dd if=Calligo_linux_boot_image_v1 of=/dev/sdc1 (sdc1 is where the SD card device partition 1 is mounted)
- Copy runtime libraries to the second partition using the following commands.
    - sudo cp -r Calligo_SDCard_Setup/runtime_libraries/calligo_pnu_runtime_libraries/ /media/<username>/sdcard_unique_hashid/

  - sync
  - Eject the SD card.
- Attach the SD card to the Device and power it on.

10. Calligo Posit Compiler

Calligo RISC-V Posit compiler should be used to compile the code to generate a binary that can execute on the UTTUNGA card. Note that this is a cross compiler that creates Posit binaries only. This does not work for normal IEEE floating point or double data types. All float declarations are converted to Posit <32,2> and all double declarations are converted to Posit <64,3> by this compiler.

The Posit Compiler is built in a Docker for easy installation and use.

Contact Calligo Technologies for a node locked license and compiler download credentials.

Details of getting the compiler up and running is explained in the Readme,txt provided with the compiler.

Installation scripts will give details of path settings that need to be done on the host before trying to access the compiler inside the docker. Make a note of this and save it for future use.

10.1. Compiling the code

Once the docker is installed and path settings (export command) are done successfully now all it takes to create a Posit binary of the existing C-program is to recompile as below.

- To check the compiler status
  - ***cppc-linux-gcc --version***
- Sample code
  - Clone the following repository for sample codes
  - git clone https://github.com/eswarnitk/PNU_Examples.git
  - cd PNU_Examples
- Compilation command
  - ***cppc-linux-gcc -march=rv64imafd -mabi=lp64d --static -DUSE_CPML Example1.c -o Example /usr/riscv_gnu_pnu_12_2_compiler_binaries/sysroot/usr/lib/libcpml.a***
  - The git repo above has a compile.sh file that contains this long command. This can be edited to make things easier.
- Compilation parameters
  - ***-DUSE_CPML***: To link default math calls to calligo's posit math library(**mandatory**) and for using conversion modules. CPML is the Calligo Posit Math Library that handles all math functions like sqrt, sin, etc. in Posit format.
  - ***/usr/riscv_gnu_pnu_12_2_compiler_binaries/sysroot/usr/lib/libcpml.a***: To build with CPML library. The library, compiler and other tools are packaged inside a Docker environment with access restrictions. These are not available for direct use from any user login.
  - ***–static***: To build static binaries, it's easy to move and run on the device without dependencies issue.
  - For running with shared CPML libraries build with **-lcpml.** The library file is already available on the SD card.

11. Executing on the device

- The UTTUNGA device has a removable SD card that is used to copy executables from a development server (host) to the card (device)
- Run the executable on the device from serial (UART) interface