FPGA Lab Project

Sachinkumar Dubey - EE20MTECH11009

Download all codes from

```
https://github.com/sachinomdubey/FPGA_Lab/
    project_sachin/
```

1 Step for Flashing ESP32

- Download the ESP32 webserver code from below link: https://github.com/sachinomdubey/FPGA_Lab/tree/main/project_sachin/webserverESP32
- ESP32 should connect to your android hotspot to host the webserver, for this make changes in below lines as per your hotspot credentials:

```
const char* ssid = "SOD";
const char* password = "sachin009";
```

- We need to know the IP address of ESP32 to access the http web server. To know the IP address, also uncomment the below line in ESP32webserver code before flashing.

```
// Serial.println(WiFi.localIP());
```

- Select the following settings in Arduino IDE before flashing:
  - Board : ESP32 Dev Module
  - Upload speed : 115200
  - CPU frequency : 240 MHz
  - Flash frequency : 40 MHz
  - Flash size : 4 MB
- Use TTL to USB connector to flash the code to ESP32. While flashing, connect IO0 pin of onboard ESP32 of Vaman to GND. Now press Upload on arduino IDE. While it says "connecting", connect the EN pin of ESP32 to GND for 1-2 Sec and subsequently disconnect the EN pin. The flashing process should start and complete.
- After you reset, open the serial terminal where you can see the IP address. Note this down.
- Repeat the flashing process by again commenting the below line. This is to make sure, it does not interfere when esp32 sends data to ARM UART.

```
// Serial.println(WiFi.localIP());
```

2 Step for Flashing Vaman ARM

- Download the Vaman code from below link: https://github.com/sachinomdubey/FPGA_Lab/tree/main/project_sachin/Vaman_Signal_Processing
- On terminal, Run "make" command in GCC_project folder to generate .bin file.
- Flash the .bin file along with top.bin to vaman using following command. NOTE: change directory paths and port as per your system.

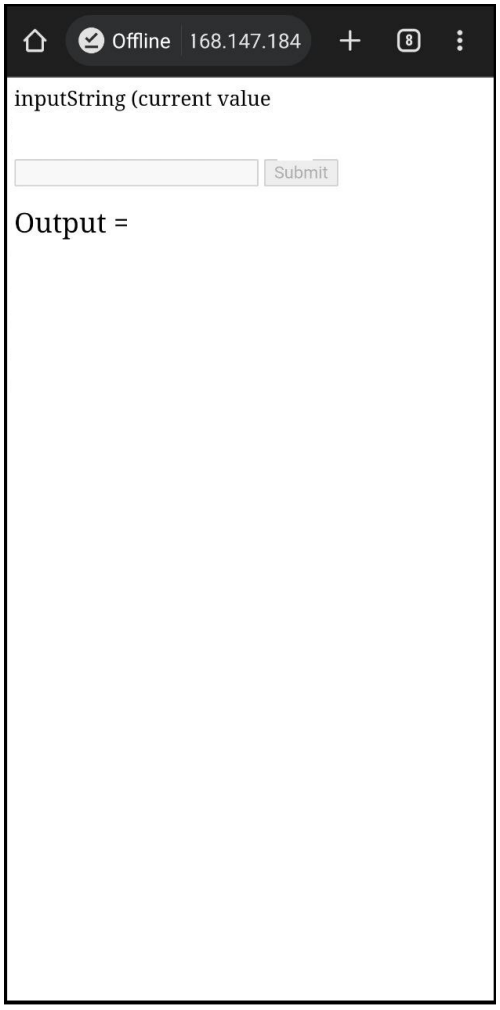
```
python3 /home/sachin/Documents/Vaman/
    qorc-sdk/TinyFPGA-Programmer-
    Application/tinyfpga-programmer-
    gui.py --port /dev/ttyACM0 --
    m4app /home/sachin/Documents/
    Vaman/project_sachin/GCC_Project/
    output/project_sachin.bin --appfpga
     /home/sachin/Documents/Vaman/top.
    bin --mode fpga-m4 --reset
```

3 Wiring Diagram

https://github.com/sachinomdubey/FPGA_Lab/blob/main/project_sachin/Figures/FPGA_lab_project_wiring.pdf

4 Checking the result

- Make connections as per the wiring diagram.
- Reset the controller.
- ESP32 gets connected to Android hotspot automatically.
- Go the IP address which was acquired during flashing project in chrome browser. you can observe the following interface.



- Enter the signal samples x[n] you want to convolve or correlate and press submit. (Note: we are sending the x[n] and h[n] is defined inside the ARM code itself). A sample x[n] and the expected output y[n] is given below.

x[n] = 0.5377, 1.8339, −2.2588, 0.8622, 0.3188, −1.3077, −0.4336, 0.3426, 3.5784, 2.7694, −1.3499, 3.0349, 0.7254, −0.0631, 0.7147, −0.2050, −0.1241, 1.4897, 1.4090, 1.4172

convolution y[n] = 0.3611 0.5822 −3.3456 5.4983 0.8055 −2.8740 4.1062 −0.4103 −1.4459 −1.8042 −4.3505 13.2034 −2.7157 4.8211 10.2696 −4.4270 9.7560 2.9032 −0.5928 4.6351 −7.3842 0.3368 −9.3994 −8.5450 3.6132 −9.3192 −3.9125 0.5994 −3.3785 −1.2455 −3.9723 −7.0416 −5.2957 −4.1727

correlation y[n] = −1.5832 −5.8348 4.5913 −3.2871 −0.8481 6.4676 −2.2871 3.2946 −10.0747 −9.5041 −1.4903 −14.6805 0.2470 −5.4371 −12.3220 7.0057 −4.8595 0.2204 0.6648 0.1413 5.3020 −4.1038 2.3260 8.0337 −5.2602 3.1251 2.8166 1.9617 5.2222 4.0704 1.4387 0.3154 −0.7651 0.9516

- The entered samples goes through ESP32 UART to ARM UART. ARM then performs the required operation (convolution or correlation) depending on the status of GPIO4 pin. (GND - Convolution, 3.3V - Correlation)
- ARM sends the output back through ARM UART to ESP32 UART. ESP32 then display the output on webserver as shown in below figure: