

# SINE WAVE GENERATOR

Aamir Sohail (EE16BTECH11021), Piyush Udhan (EE16BTECH11028)

EE5811: FPGA LAB  
&  
EE3025: Independent Project

Department of Electrical Engineering,  
IIT HYDERABAD

April 25, 2019

# Project Outline

The aim of our project is to make a sine wave generator using Icoboard FPGA. We have used cordic algorithm to do so along with arduino Mega for post processing. To realise the sine wave, we have used an R2R DAC circuit to convert digital signal to analog and a digital oscilloscope (Analog Discovery) to see the waveform on the screen.

- All the verilog and arduino codes, the report and presentation are available on the github link which is a free open-source. [► Github Link](#)

# Software setup

- ➊ Open the command terminal execute the following command [▶ Link](#)  
for installing: [1]
  - wiringPi
  - IcoProg
  - IcoStorm tools
  - Arachne-pnr
  - Yosys
- ➋ Creating **Makefile** [1]

## MOTIVATION

If a unit vector with co-ordinates  $(x_1, y_1) = (1, 0)$  is rotated by an angle  $\theta$ , its new co-ordinate will be  $(x_2, y_2) = (\cos \theta, \sin \theta)$ . Thus, by finding the  $(x_2, y_2)$ ,  $\cos \theta, \sin \theta$  can easily be computed.

# Circular CORDIC Algorithm

CORDIC iteration: [2]

$$x(i+1) = x(i) - d_i y(i) 2^{-i} \quad (1)$$

$$y(i+1) = y(i) + d_i x(i) 2^{-i} \quad (2)$$

$$\alpha(i+1) = \alpha(i) - d_i \tan^{-1} 2^{-i} \quad (3)$$

Choose  $d_i \in \{+1, -1\}$  such that  $\alpha(n) \rightarrow 0$

After n pseudorotation steps, when  $\alpha(n)$  is well enough close to zero, we will get  $\sum \theta(i) = \alpha$ . Finally the CORDIC iterations in ROTATION MODE become:

$$x(n) = \mathbf{K} (x \cos \alpha - y \sin \alpha) \quad (4)$$

$$y(n) = \mathbf{K} (y \cos \alpha + x \sin \alpha) \quad (5)$$

$$\alpha(n) = 0 \quad (6)$$

where  $\mathbf{K} = \prod (1 + \tan^2 \theta(i))^{1/2}$ ,  $\theta(i) = \tan^{-1}(d_i 2^{-i})$

# Domain of Convergence

- In rotation mode, convergence of  $\alpha(n)$  to 0 is possible because each CORDIC angle is more than half the previous angle.

$$\tan^{-1}(2^{-(i+1)}) \geq 0.5 * \tan^{-1}(2^{-i})$$

- Thus, domain of convergence is  $-99.7^\circ \leq \alpha \leq 99.7^\circ$  where  $\pm 99.7^\circ$  is the sum of the CORDIC angle.
- Beyond the above range, we can use Trigonometric identities to convert the problem within the domain of convergence.

# Implementation of Circular CORDIC in RTL

- The digital design should be able to compute Sine and Cosine of the input angles  $\in [0, 2\pi]$  including the floating point angles like  $89.45^\circ$ ,  $29.7^\circ$  etc.
- A 16-bit binary scaling system has been used to represent angles. The resolution of the design is  $360^\circ / 2^{16} = 5.49316406 \times 10^{-3}$ . Angle $^\circ$  to 16-bit binary value conversion:

$$(16bit)\text{angle} = \frac{2^{16} * \text{angle}^\circ}{360^\circ}$$

- The upper two bits represent the quadrant [3].
  - 2b'00 -> represents I quadrant i.e  $(0 - \pi/2)$  range
  - 2b'01 -> represents II quadrant i.e  $(\pi/2 - \pi)$  range
  - 2b'10 -> represents III quadrant i.e  $(\pi - 3\pi/2)$  range
  - 2b'11 -> represents IV quadrant i.e  $(3\pi/2 - 2\pi)$  range

# Coding and Hardware Environment

- CORDIC Algorithm: Verilog
  - ① Xilinx ISE-Design Suite 14.7<sup>1</sup>
  - ② Simulation: Xilinx-ISim software
- Postprocessing:
  - ① Software: Arduino [Download Link](#)
  - ② Board: Arduino MEGA 2560
- Oscilloscope:
  - ① DAC: 'R2R Ladder' [Link](#)
  - ② Analog Discovery (Digital Oscilloscope)<sup>2</sup>
    - Software: Waveform [Download Link](#)

---

<sup>1</sup>L.Gregg - How to Download & Install Xilinx ISE 14.7 Windows 10 [Youtube Link](#)

<sup>2</sup>Getting Started with Analog Discovery [Youtube Link](#)

# Circuit and System Setup

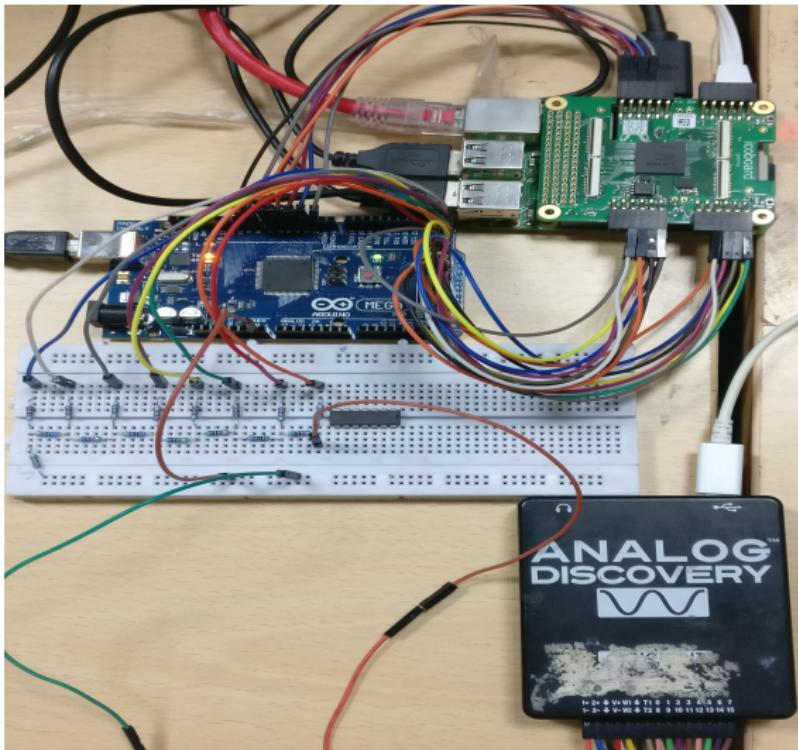


Figure: Circuit Arrangement

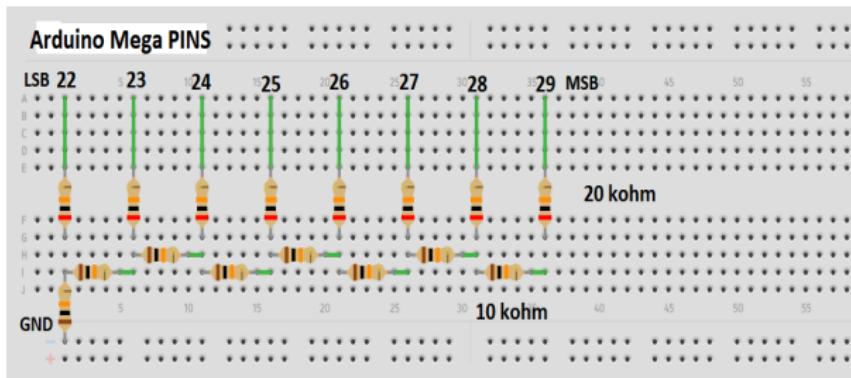


Figure: 8-bit 'R2R' DAC Circuit

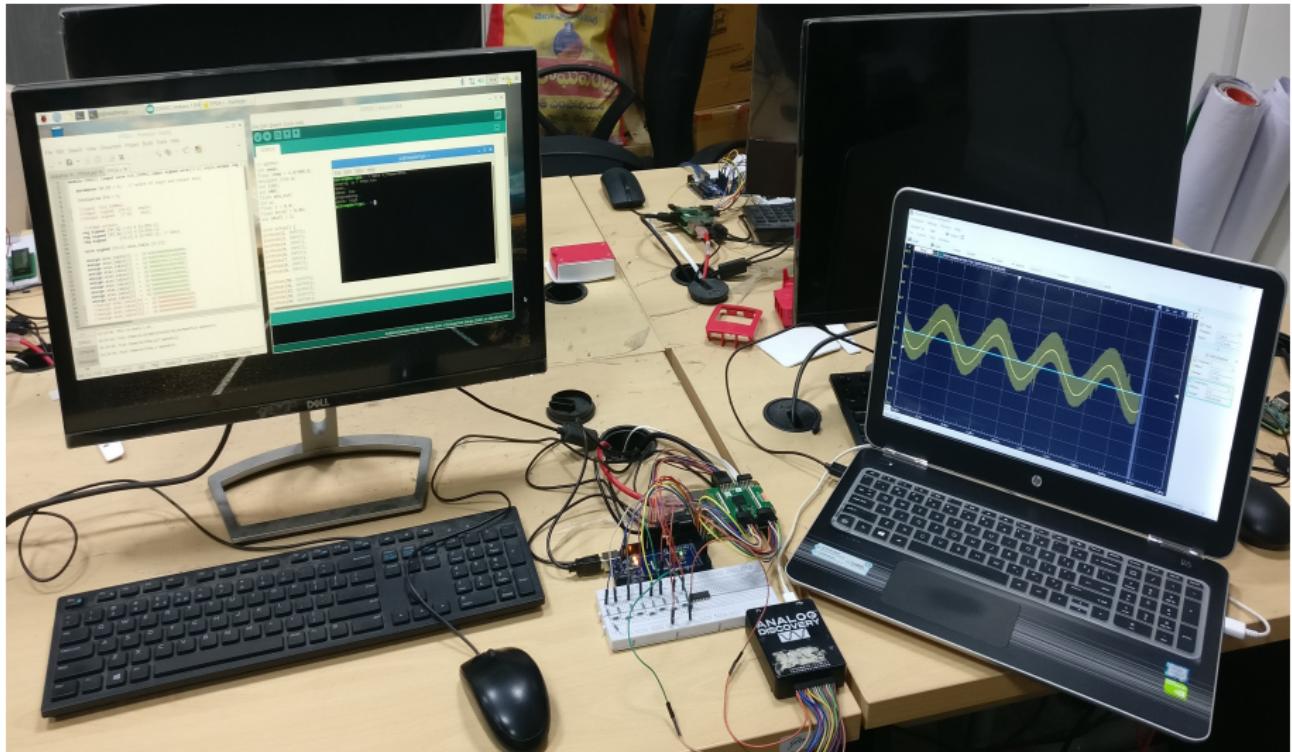


Figure: System Setup

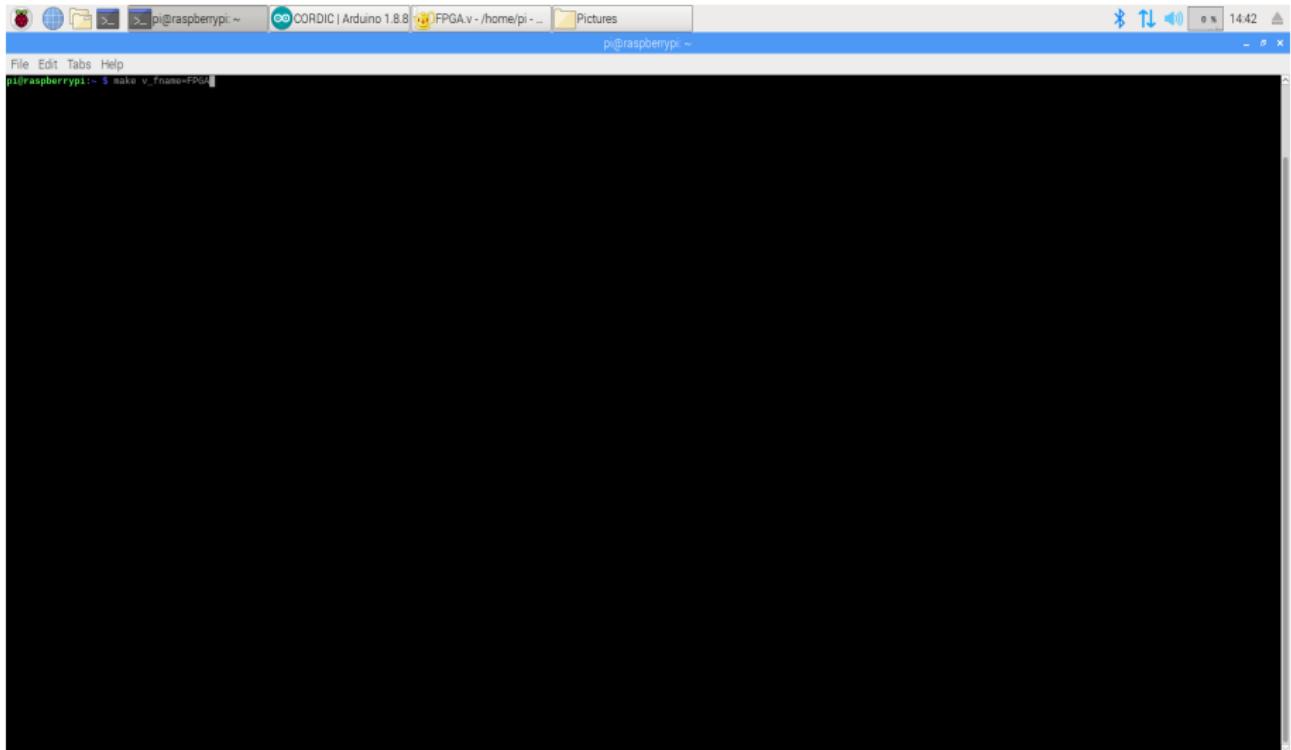


Figure: Uploading FPGA.v to Ico-Board FPGA

```
pi@raspberrypi: ~
```

```
File Edit Tabs Help
```

```
Number of public wires: 3
Number of public wire bits: 25
Number of memories: 0
Number of memory bits: 0
Number of processes: 0
Number of cells: 869
  SB_CARRY 313
  SB_DFF 8
  SB_LUT4 539

2.28. Executing CHECK pass (checking for obvious problems).
Checking module CORDIC..
found and reported 0 problems.

2.29. Executing BLIF backend.

Warnings: 3 unique messages, 3 total
End of script. Logfile hash: fed7794a91
CPU: user 21.40s system 0.41s, MEM: 21.98 MB total, 16.12 MB resident
Vcrys 0.8+19 (git sha1 ffc6e2bf, clang 3.8.1+24+rp1 -FPIC -Os)
Time spent: 20s 30x opt_clean (4 sec), 16s 30x opt_expr (3 sec), ...
arachne-pnr -d 8k -p FPGApcf -o FPGA.asc FPGA.blif
seed: 1
device: 8k
read_chipdb ./share/arachne-pnr/chipdb-8k.blif...
  supported packages: bg121, bg121:4k, cb132, cb132:4k, cm121, cm121:4k, cm225, cm225:4k, cm@1, cm@1:4k, ct256, tq144:4k
  read_blif FPGA.blif...
  promote...
  read_pcf FPGApcf...
  instantiate_io...
  pack...

After packing:
IOs      25 / 206
Gbs      0 / 0
_0B_I0s  0 / 0
Lcs      677 / 7680
DFF     325
CARRY, DFF 0
DFF PASS 0
CARRY PASS 65
BRAMs   0 / 32
NHRMBOOTs 0 / 1
PLLs    0 / 2

place_constraints...
promote_globals...
  promoted CLK_100MHZS2, 0 / 0
  promoted 1 nets
    1 clk
  1 globals
    1 clk
realize_constants...
  realized 1
place...
  initial wire length = 22631
  at iteration #50: temp = 28.0339, wire length = 12001
```

Figure: Uploading FPGA.v to Ico-Board FPGA

The screenshot shows a Raspberry Pi desktop environment with several open windows:

- Terminal Window:** Shows the command prompt `pi@raspberrypi:~$`.
- Arduino IDE Window:** Titled "CORDIC | Arduino 1.8.8". It displays a sketch for a CORDIC algorithm. The code includes declarations for parameters, local variables, and a main loop. It uses floating-point arithmetic and bit manipulation to calculate an angle.
- Verilog Editor Window:** Titled "FPGA.v - /home/pi - Geany". It contains the Verilog source code for the CORDIC module. The code defines a module named "CORDIC" with input and output ports. It includes a parameter for XY\_SIZE, local parameters for STG, and a wire for atan\_table. The module body contains assignments for atan\_table[0] through atan\_table[13] and a always @() block for the atan\_table[14] assignment. The always block includes a begin-end block with a case statement for angle[15:14]. The case block handles two-bit angles (2'b00, 2'b11) and one-bit angles (2'b0). The 2'b0 case includes a note about no pre-rotation needed for these quadrants. The 2'b1 case includes a note about X[n] being one bit larger than Xin, Vin, but Verilog handling it. The always block also includes a \$display statement.

At the bottom of the Verilog editor window, the status bar shows:

```

line 5 / 161 col 23 sel 0 INS TAB mode LF encoding: UTF-8 filetype: Verilog sc

```

Figure: Uploading FPGA.v to Ico-Board FPGA

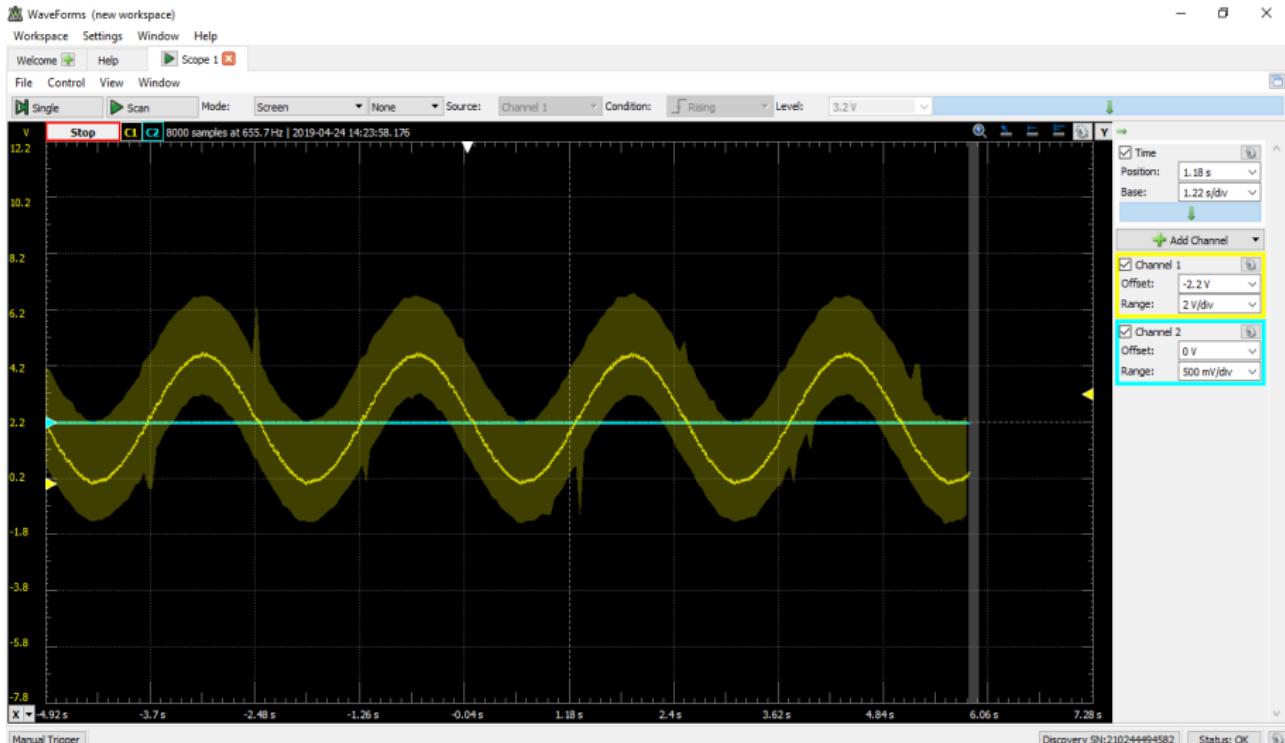


Figure: Output Waveform

# References



GVV Sharma, Hemanth Kumar Desinnedi,  
"Introduction to verilog programming on IcoBoard FPGA", [▶ Link](#)



Behrooz.Parhami., COMPUTER ARITHMETIC-Algorithms and Hardware Designs  
2nd edition., OXFORD UNIVERSITY PRESS, 2010, ch.22.



Kirk Weedman, CORDIC Design and Simulation using Verilog, [▶ Link](#)