NCTU-EE IC LAB - Spring 2022

Online Test

Design: CORDIC

Data Preparation

1. Extract test data from TA's directory:

   % tar -xvf ~iclabta01/OT_2022_spring.tar

CORDIC

CORDIC, coordinate rotation digital computer, is a hardware efficient iterative method that uses several rotations to calculate special functions with a predetermined rotation angle.

Suppose we would like to rotate a vector by an arbitrary angle θ. Choose the origin as the center of rotation; we will get to the point (x_1, y_1) by rotating the point (x_0, y_0) by θ.

$$x_1 = x_0\cos\theta - y_0\sin\theta$$
$$y_1 = y_0\cos\theta + x_0\sin\theta$$

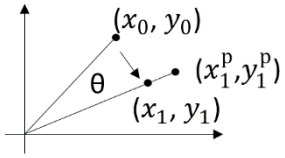
The above shows that we will need to perform four multiplications for one rotation.

The CORDIC algorithm resorts to two fundamental ideas to achieve rotation without multiplication. First is to dismantle the angle θ into several smaller angles θ_i, i=0, 1, ……, n-1, and try to meet

$$\theta = \sum_{i=0}^{n-1} d_i * \theta_i, \text{ while } d_i \in \{-1,1\}.$$

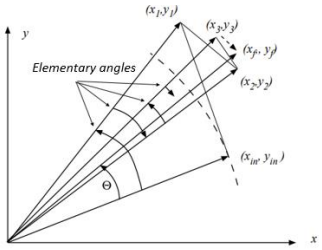
Second is that we choose the elementary angle in the way that $\tan\theta = 2^{-i}$, so the elementary angles θ are obtained by $\tan^{-1}(2^{-i})$ for i = 0, 1, …., n-1. We could obtain the desired accuracy by increasing the number of rotations.

Here we define pseudo rotation $\begin{cases} x_1^p = \frac{x_1}{\cos\theta} = x_0 - y_0\tan\theta \\ y_1^p = \frac{y_1}{\cos\theta} = y_0 + x_0\tan\theta \end{cases}$



$$\sqrt{(x_1^p)^2 + (y_1^p)^2} = \frac{1}{\cos\theta}\sqrt{{x_1}^2 + {y_1}^2} = \frac{1}{\cos\theta}\sqrt{{x_0}^2 + {y_0}^2}$$

By the pseudo rotation that we defined above, we could find that the length will increase by $\frac{1}{\cos\theta}$ in every rotation. We could derive pseudo rotation at step i :



$$x_{i+1}^p = \frac{x_i}{\cos\theta} = x_i - y_i\tan\theta_i \qquad y_{i+1}^p = \frac{y_i}{\cos\theta} = y_i + x_i\tan\theta_i$$

$$\Rightarrow \quad x_{i+1}^p = \frac{x_i}{\cos\theta} = x_i - y_i d_i 2^{-i} \qquad y_{i+1}^p = \frac{y_i}{\cos\theta} = y_i + x_i d_i 2^{-i}$$

As θ_i becomes smaller when i is big, $\cos\theta_i$ is approached to 1. As a result, we could set a constant

$$\mathbf{K} = \frac{1}{\cos 45^\circ} \times \frac{1}{\cos 26.565^\circ} \times \ldots \times \frac{1}{\cos 0.223811^\circ} \times \ldots \times \frac{1}{\cos 0.000437^\circ} \approx \mathbf{1.64679.}$$

To sum up, the CORDIC algorithm is

$$x_{i+1}^{p} = x_i - y_i d_i 2^{-i}$$
$$y_{i+1}^{p} = y_i + x_i d_i 2^{-i}$$
$$z_{i+1} = z_i - d_i \tan^{-1}(2^{-i})$$

z_i represents phase

$d_i \in \{-1,1\}$ determines the rotation direction. If $y_i < 0$ then d_i = 1, else d_i = -1.

After several times of rotation, for example n, you can find that y_n becomes very close to 0. In this time, x_n will be the magnitude multiple constant K and z_n will be the phase.

$$x_n = K\sqrt{{x_0}^2 + {y_0}^2}$$
$$y_n \to 0$$
$$z_n = \tan^{-1}\frac{y_0}{x_0}$$

※ The elementary angle $\theta_i = \tan^{-1}(2^{-i})$ table：

i	θ_i(deg)	θ_i (pi)	θ_i(pi,(unsigned,1int+20fraction)
0	45.00000	0.25	21'h04_0000
1	26.565051	0.147583617	21'h02_5c81
2	14.036243	0.077979128	21'h01_3f67
3	7.125016	0.03958342	21'h00_a222
4	3.576334	0.01986852	21'h00_5162
5	1.789911	0.00994395	21'h00_28bb
6	0.895174	0.00497319	21'h00_145f
7	0.447614	0.00248674	21'h00_0a30
8	0.223811	0.001243394	21'h00_0518
9	0.111906	0.0006217	21'h00_028b
10	0.055953	0.00031085	21'h00_0146
11	0.027976	0.00015542	21'h00_00a3
12	0.013988	0.0000771	21'h00_0051
13	0.006994	0.00003886	21'h00_0029
14	0.003497	0.000019428	21'h00_0014
15	0.001749	0.000009717	21'h00_000a
16	0.000874	0.00000486	21'h00_0005
17	0.000437	0.000002428	21'h00_0003

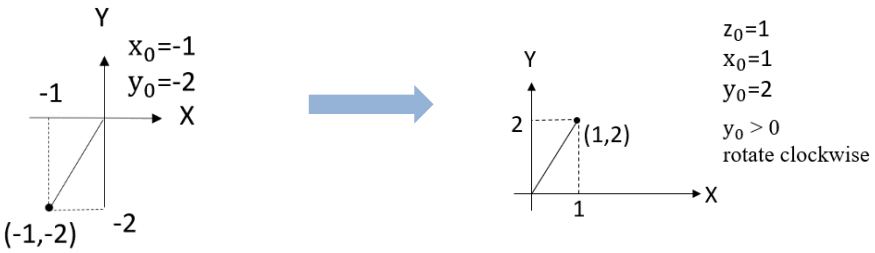
Design Description

In the online test, you are asked to calculate the polar representation of the input x, and y coordinates using the CORDIC Algorithm. **The times of rotation are fixed at 18**, you should try your best to meet the accuracy requirement. **Input signals will be randomly given for n cycles continuously from TA's pattern, where n is ranging from 100 to 1000**. After in_valid signal is down, you could output the polar representation for n cycles continuously. You **must utilize SRAM provide by TA** instead of too many registers or you may not meet the area limitation criteria. Furthermore, **you should meet the total average error ratio criteria 0.1%**. Notice that the input x and y may be negative, it is recommended to rotate coordinate to the first quadrant.

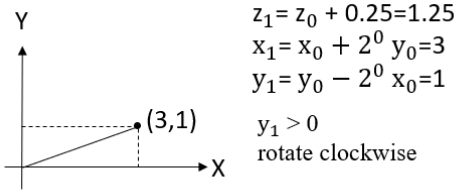
Example

in_x = -1 , in_y = -2

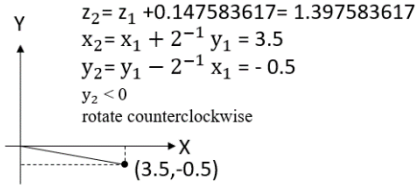
The input point (-1, -2) is in the third quadrant, you should rotate it to the first quadrant for the convenience of the following CORDIC rotation steps.



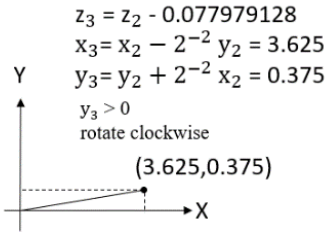
STEP1



STEP2

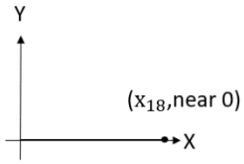


STEP3



Keep rotation until step 18.

STEP18



out_phase = z_{18}

out_mag = $x_{18}/1.64679 = x_{18} * 0.6072387695$

Signals

Input Signal	Bit Width	Definition
clk	1	Clock.
rst_n	1	Asynchronous active-low reset.
in_valid	1	High when input signals are valid.
in_x	12	The input x coordinate. {1'b sign, 3'b int, 8'b fraction}
in_y	12	The input y coordinate. {1'b sign, 3'b int, 8'b fraction}

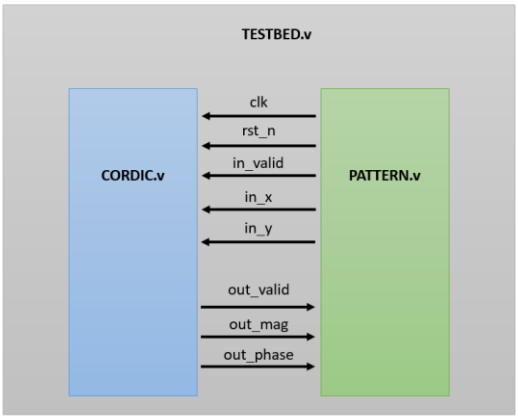
Output Signal	Bit Width	Definition
out_valid	1	High when output is valid.
out_mag	12	The output magnitude. {4'b int, 8'b fraction}
out_phase	21	The output phase(pi). {1'b int, 20'b fraction} $0 \leq$ out_phase $< 2\pi$

Inputs

1. The **in_x[11:0]** and **in_y[11:0]** will be valid for n cycles (100≤ n ≤1000) when **in_valid is high.**
2. All input signals will be synchronized at the **negative edge** of the clock.
3. The next input pattern will be triggered in 1 ~ 5 cycles after **out_valid** falls.

Outputs

1. All outputs should be tied low after **rst_n** assert.
2. **out_valid** should not be raised when **in_valid** is high.
3. Output signals should be synchronized at clock positive edge.
4. **out_valid** is set to high when the output value is valid and it will be high for n cycles ($100 \leq n \leq 1000$) continuously when triggered.
5. TA's pattern will capture your output for checking at **negative** clock edge.



Specifications

1. Top module name：CORDIC (Filename：CORDIC.v)
2. It is asynchronous reset and active-low architecture. If you use synchronous reset in your design, you may fail to reset signals.
3. The clock period of the design is **10ns.**
4. The **rst_n** would be triggered only once. All output signals should be reset to low after the reset signal is asserted.
5. The out_valid and output data must be high successively for n cycles. n is the same as in_valid cycles.

6. The next group of inputs will come in 1~5 cycles after your out_valid pull-down.
7. The synthesis result of data type cannot include any LATCH.
8. The slack in the timing report should be non-negative and the result should be MET.
9. The gate-level simulation cannot include any timing violation.
10. The latency of your design in each pattern should not be larger than 100000 cycles.
11. The area of your design should not be larger than 1000000.
12. You can only use two SRAMs provided by TA in your design.
13. The average error of phase and magnitude should be **less than 0.1%**.

$$\textbf{Phase error} = \frac{\textbf{abs}(phase_{golden} - phase_{yours})}{phase_{golden}}$$

$$\textbf{Magnitude error} = \frac{\textbf{abs}(\textbf{magnitude}_{golden} - \textbf{magnitude}_{yours})}{\textbf{magnitude}_{golden}}$$

```
--------------------------------------------------------------------------------------
                                Notice!
        Average error ratio (%) Magnitude: 116.345480, Phase: 0.000912  at  0 pattern
--------------------------------------------------------------------------------------
```

→ means that the average error ratio of output so far doesn't meet the accuracy requirement.

But, whether you pass or not is decided by the average error ratio of all output.

- FAIL if the average error ratio of all output exceeds 0.1%.

```
--------------------------------------------------------------------------------------
                                FAIL!
        Average error ratio (%) Magnitude: 116.343836, Phase: 0.000911
--------------------------------------------------------------------------------------
```

- PASS if the average error ratio of all output is under 0.1%.

```
--------------------------------------------------------------------------------
                            Congratulations!
                    You have passed all patterns!
        Average error ratio (%) Magnitude: 0.040707, Phase: 0.001570
--------------------------------------------------------------------------------
```

14. The look-up table method is forbidden. **(TA will check your design)**
15. Don't use any wire/reg/submodule/parameter name called *error*, *congratulation*, *latch* or *fail* otherwise you will fail the lab. Note: * means any char in front of or behind the word. e.g: error_note is forbidden.
16. **Any Designware IP is forbidden in online test**.
17. Any form of display or printing information in Verilog design is forbidden. You may use this methodology during debugging, but the file you turn in should not contain any coding that is not synthesizable.

Grading Policy

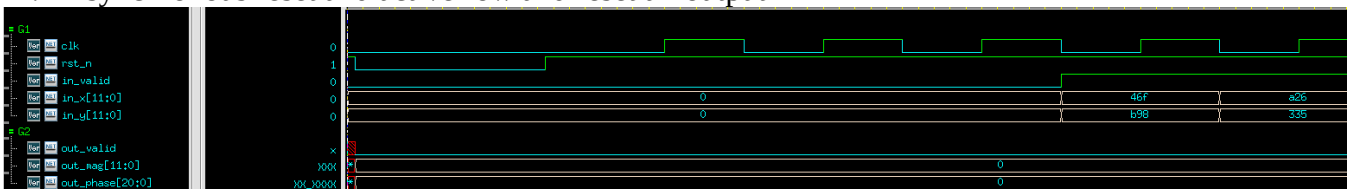
1. You only have **one chance** to demo. If you fail, you may have to take it home.
   Functionality：1de: 100%, Take home: 50% **(Take home deadline：04/12 (TUE) 23:59)**
2. Submit your design through OT_2022_spring/09_SUBMIT/01_submit
3. Please submit your design before
   Group A：19:00, Group B：15:45
4. For who using your personal laptop, please submit your screen record to below link before **04/09** 20:00 or your OT will be regarded as FAIL.
   https://drive.google.com/drive/u/2/folders/1VMYos3DocYARJaieFdalgUyxVgIxbKLY

File name：iclabxxx.mp4

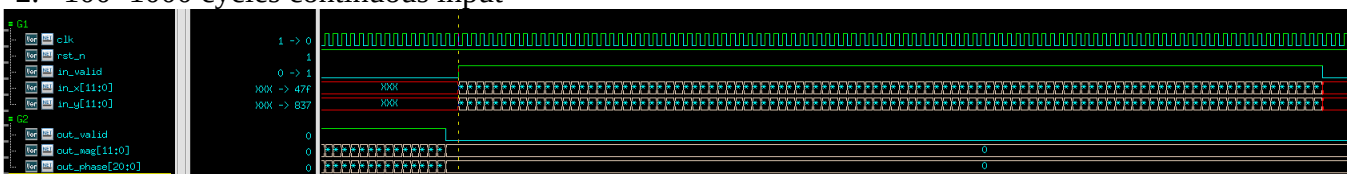
5. If the uploaded file, including the video, violates the naming rule, you will get 5 deduct points.
6. 09_SUBMIT/01_submit means submit your design and doesn't mean demo, you can submit your design before deadline several times but we will only demo for the last summit.
7. **Don't submit your design after deadline, if the timestamp of the submitted file is later than the deadline, we will regard it as FAIL.**
8. **Don't use linux01!!**

Sample Waveform

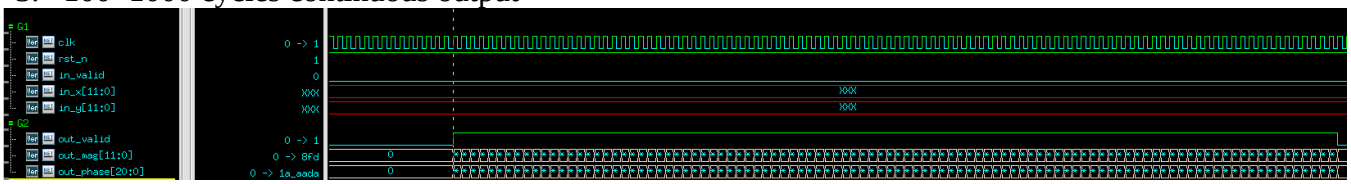
1. Asynchronous reset and active-low and reset all output



2. 100~1000 cycles continuous input



3. 100~1000 cycles continuous output



4. out_meg, out_phase, out_valid should be 0 after continuous output.