NCTU-EE IC LAB - Fall 2019

Lab04 Exercise

Design: Denavit-Hartenberg (D-H) Convention

Data Preparation

1. Extract test data from TA's directory:

   % tar xvf ~iclabta01/Lab04.tar

2. The extracted LAB directory contains:

   a. **00_TESTBED**

   b. **01_RTL**

   c. **02_SYN**

   d. **03_GATE**

Design Description

A commonly used convention for selecting frames of reference in robotics applications is the *Denavit-Hartenberg (D-H) convention* which was introduced by Jacques Denavit and Richard S. Hartenberg. The robot limb with coordinate frames conforming to D-H convention is shown in Figure 1.

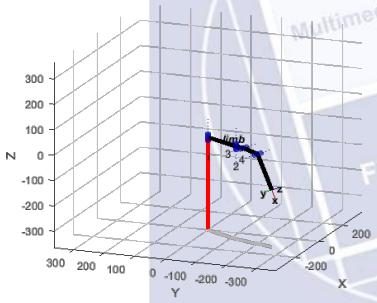


Figure 1. The robot limb with coordinate frames conforming to D-H convention.

Figure 2. Gimbal Stabilizer [1]

In this lab, you are required to apply the D-H convention to calculate the positions of endpoints of the 4-axis gimbal stabilizer in various specifications. An example of specifications is shown in Figure 2.

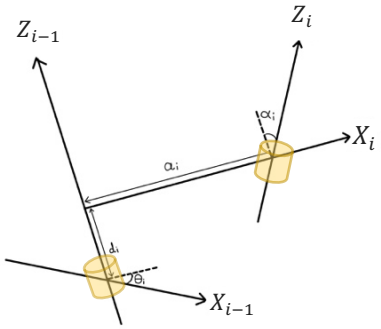


Figure 3. The four parameters of D-H convention

The following four transformation parameters are known as D-H parameters (as shown in Figure 3):

- d_i : offset along Z_{i-1} to the common normal
- θ_i : joint angle (i.e., angle of the i^{th} motor in our case)
- a_i : link length (length of the common normal). Assuming a revolute joint, this is the radius about Z_{i-1}
- α_i : link twist (angle about common normal), from Z_{i-1} to Z_i

The D-H parameters for each joint of the gimbal stabilizer are given in Table 1:

Table 1. D-H parameter for joint *i* of the gimbal stabilizer

i	θ_i	d_i	a_i	α_i
1	θ_1	d_1	a_1	α_1
2	θ_2	d_2	a_2	α_2
3	θ_3	d_3	a_3	α_3

The D-H transformation matrix for adjacent coordinate frames, *i* and *i*-1, is given as

$$^{i-1}T_i = Rot_{Z_{i-1}}(\theta_i)Trans_{Z_{i-1}}(d_i)Trans_{X_i}(a_i)Rot_{X_i}(\alpha_i)$$

$$= \begin{bmatrix} cos(\pi\theta_i) & -sin(\pi\theta_i) & 0 & 0 \\ sin(\pi\theta_i) & cos(\pi\theta_i) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & a_i \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & cos(\pi\alpha_i) & -sin(\pi\alpha_i) & 0 \\ 0 & sin(\pi\alpha_i) & cos(\pi\alpha_i) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} \cos(\pi\theta_i) & -sin(\pi\theta_i)cos(\pi\alpha_i) & sin(\pi\theta_i)sin(\pi\alpha_i) & a_i cos(\pi\theta_i) \\ sin(\pi\theta_i) & cos(\pi\theta_i)cos(\pi\alpha_i) & -cos(\pi\theta_i)sin(\pi\alpha_i) & a_i sin(\pi\theta_i) \\ 0 & sin(\pi\alpha_i) & cos(\pi\alpha_i) & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Therefore, the transformation matrix from the reference frame {0} to the endpoint frame {4} is calculated as $^0T_4 = {}^3T_4\,{}^2T_3\,{}^1T_2\,{}^0T_1$. The position $(x_{out}, y_{out}, z_{out})$ in the reference frame can be obtained by

$$\begin{bmatrix} x_{out} \\ y_{out} \\ z_{out} \\ 1 \end{bmatrix} = {}^3T_4\,{}^2T_3\,{}^1T_2\,{}^0T_1 \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} = {}^3T_4\,{}^2T_3\,{}^1T_2 \begin{bmatrix} a_1 cos(\pi\theta_1) \\ a_1 sin(\pi\theta_1) \\ d_1 \\ 1 \end{bmatrix}$$

At first, we will send the specifications for 4 joints of the gimbal stabilizer ($\alpha_i, a_i,$ and d_i, $i = 1\sim4$) in 4 cycles. α_1 is useless in this lab, so it will always be sent as unknown. Then, we will send joint angles (i.e., angle of four motors) of *N* time instants ($\theta_1[t_j], \theta_2[t_j], \theta_3[t_j], and\ \theta_4[t_j]$, $t_j = t_0\sim t_{N-1}$) for *N* cycles, which is ranged $1 \le N \le 100$. You can start to apply D-H convention of time instant t_0 right after receiving the joint angles of the time instant t_0, and then apply same operations for input sequences of time instant $t_1, t_2, \dots, t_{N-1}$. It is recommended to use the pipeline architecture. Once your results are ready, you need to output the results of time instants $t_0\sim t_{N-1}$ sequentially for *N* cycles.

In this exercise, you need to perform the fixed-point operation in your design, that to define how many bits for integer part and fractional part of numbers. The fractional bits of input and output signals are set, but you can determine how many fractional bits should be remained by yourself in the intermediate calculation. For example, 1.625, which is 01.101 in binary 2's complement

representation, you can choose to drop bit to 2nd fractional bit 01.10 = 1.5, or round to 2nd fractional bit 01.11 = 1.75, or keep 3 fractional bits as original to have 1.625. The more bits you preserve the higher accuracy you can get, but also cost more hardware. It is recommended to use more fractional bits at the beginning, if the accuracy is overdesign, then try to reduce fractional bits afterwards.

You must use Sine and Cosine IP (DW_sincos) from Designware in this lab, and determine the parameters of DW_sincos by yourself.

The quantization of the fractional bits and the size of designware are determined by yourself. The only requirement is that error of each pattern should < 0.05. The error is defined as follows:

$$Error = \frac{\sqrt{\frac{\sum_{j=0}^{N-1}\left(\left\|OUT\_X_{golden}[t_j] - OUT\_X_{your}[t_j]\right\|^2 + \left\|OUT\_Y_{golden}[t_j] - OUT\_Y_{your}[t_j]\right\|^2 + \left\|OUT\_Z_{golden}[t_j] - OUT_{Z_{your}}[t_j]\right\|^2\right)}{N}}}{\sqrt{{a_1}^2 + {d_1}^2} + \sqrt{{a_2}^2 + {d_2}^2} + \sqrt{{a_3}^2 + {d_3}^2} + \sqrt{{a_4}^2 + {d_4}^2}}$$

where $OUT\_X_{golden}[t_j]$, $OUT\_Y_{golden}[t_j]$, and $OUT\_Z_{golden}[t_j]$ is the golden values of $x_{out}, y_{out},$ and z_{out} of time instant t_j, and $OUT\_X_{your}[t_j]$, $OUT\_Y_{your}[t_j]$, and $OUT\_Z_{your}[t_j]$ is your output results $x_{out}, y_{out},$ and z_{out} of time instant t_j. In the pattern, error calculation uses floating-point ("*real*" type) rather than fixed-point, and so does the golden value.

The golden values are generated by the sin function and cos function in *MATLAB*.

sin - Sine of argument in radians
This MATLAB function returns the sine of the elements of X.
Documentation > MATLAB > Mathematics > Elementary Math > Trigonometry

cos - Cosine of argument in radians
This MATLAB function returns the cosine for each element of X.
Documentation > MATLAB > Mathematics > Elementary Math > Trigonometry

You can generate the golden pattern through other software. The accuracy should be similar.

Inputs and Outputs

- The following are the definitions of input signals

Input Signals	Bit Width	Definition
clk	1	Clock.
rst_n	1	Asynchronous active-low reset.
IN_VALID_1	1	High when ALPHA_I, A_I and D_I is valid.
IN_VALID_2	1	High when THETA_JOINT_1, THETA_JOINT_2 and THETA_JOINT_3 is valid.
ALPHA_I	6	α_i: link twist
A_I	3	a_i: link length
D_I	3	d_i: offset along previous z-axis to the common normal
THETA_JOINT_1	6	$\theta_1[t_j]$: joint angle of the first joint (angle of the first motor of the gimbal stabilizer) of time instant t_j
THETA_JOINT_2	6	$\theta_2[t_j]$: joint angle of the second joint (angle of the second motor of the gimbal stabilizer) of time instant t_j
THETA_JOINT_3	6	$\theta_3[t_j]$: joint angle of the third joint (angle of the third motor of the gimbal stabilizer) of time instant t_j
THETA_JOINT_4	6	$\theta_4[t_j]$: joint angle of the fourth joint (angle of the fourth motor of the gimbal stabilizer) of time instant t_j

- The following are the definitions of output signals

Output Signals	Bit Width	Definition
OUT_VALID	1	High when OUT_X, OUT_Y and OUT_Z is valid.
OUT_X	9	x_{out}: x-component of the position in the reference frame
OUT_Y	9	y_{out}: y-component of the position in the reference frame
OUT_Z	9	z_{out}: z-component of the position in the reference frame

1. The input signal **IN_VALID_1** is delivered for **4 cycles** continuously. The input signal **IN_VALID_2** is delivered for ***N* cycles**. Note that *N* is a random integer in the interval [1, 100].
2. The input signal **ALPHA_I** is delivered for **4 cycles**. The order is $\alpha_1, \alpha_2, \alpha_3, \alpha_4$, but α_1 will always delivered as unknown because it is useless in this lab. When **IN_VALID_1** is low, input is tied to unknown state.
3. The input signal **A_I** is delivered for **4 cycles**. The order is a_1, a_2, a_3, a_4. When **IN_VALID_1** is low, input is tied to unknown state.
4. The input signal **D_I** is delivered for **4 cycles**. The order is d_1, d_2, d_3, d_4. When **IN_VALID_1** is low, input is tied to unknown state.
5. The input signal **THETA_JOINT_1** is delivered for ***N* cycles**. The order is $\theta_1[t_0], \theta_1[t_1], \ldots, \theta_1[t_{N-1}]$. When **IN_VALID_2** is low, input is tied to unknown state.
6. The input signal **THETA_JOINT_2** is delivered for ***N* cycles**. The order is $\theta_2[t_0], \theta_2[t_1], \ldots, \theta_2[t_{N-1}]$. When **IN_VALID_2** is low, input is tied to unknown state.
7. The input signal **THETA_JOINT_3** is delivered for ***N* cycles**. The order is $\theta_3[t_0], \theta_3[t_1], \ldots, \theta_3[t_{N-1}]$. When **IN_VALID_2** is low, input is tied to unknown state.
8. The input signal **THETA_JOINT_4** is delivered for ***N* cycles**. The order is $\theta_4[t_0], \theta_4[t_1], \ldots, \theta_4[t_{N-1}]$. When **IN_VALID_2** is low, input is tied to unknown state.
9. **A_I** and **D_I** are 3-bit *unsigned integer numbers* in the interval [1, 7]
10. **ALPHA_I, THETA_JOINT_1, THETA_JOINT_2, THETA_JOINT_3** and **THETA_JOINT_4** are 6-bit *2's complement signed fixed-point numbers*, which are of the form S.XXXXX, where S is the sign bit. They are in the interval [-0.5, 0.5]:

	Binary (Decimal)
Range	**0.10000** (0.5) **0.01111** (0.46875) ⋮ **0.00000** (0) **1.11111** (-0.03125) ⋮ **1.10001** (-0.46875) **1.10000** (-0.5)

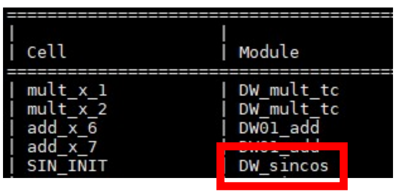
11. All input signals will be changed at clock **negative** edge.
12. You should output three sequence of 9-bit *2's complement signed fixed-point numbers* **OUT_X**,

OUT_Y and **OUT_Z**, which are of the form SXXXXX.XXX, where S is the sign bit.

13. The output signal **OUT_X** must be delivered for ***N*** **cycles** continuously. The order is $x_{out}[t_0], x_{out}[t_1], \dots, x_{out}[t_{N-1}]$.
14. The output signal **OUT_Y** must be delivered for ***N*** **cycles** continuously. The order is $y_{out}[t_0], y_{out}[t_1], \dots, y_{out}[t_{N-1}]$.
15. The output signal **OUT_Z** must be delivered for ***N*** **cycles** continuously. The order is $z_{out}[t_0], z_{out}[t_1], \dots, z_{out}[t_{N-1}]$.
16. All outputs must be synchronized at clock **positive** edge.
17. The **IN_VALID_2** will come in right after **IN_VALID_1** pulled down.
18. **OUT_VALID** can be raised after **IN_VALID_2** is high. Overlap of **OUT_VALID** and **IN_VALID_2** is allowed.
19. The next input pattern will come in 1~5 cycles after **OUT_VALID** falls.
20. The test pattern will check whether your output sequence is correct or not at clock negative edge.

Specifications

1. Top module name: DH (design file name: DH.v)
2. It is asynchronous reset and active-low architecture. If you use synchronous reset (considering reset after clock starting) in your design, you may fail to reset signals.
3. The reset signal (rst_n) would be given only once at the beginning of simulation. All output signals should be 0 after the reset signal is asserted.
4. All output signals should be 0 when your **OUT_VALID** is pulled down.
5. The latency is limited in 200 cycles. The latency is the clock cycles between the rising edge of **IN_VALID_1** and the falling edge of **OUT_VALID**.
6. The error of each pattern should < 0.05. If the error exceeds this value, you will fail this lab.
7. You can adjust your clock period by yourself, but the maxzimum period is **8 ns**. The precision of clock period is 0.1, for example, 4.5 is allowed, 4.55 is not allowed.
8. The in/output delay are set to **0.5*(clock period)**, and the output loading is set to **0.05.**
9. The synthesis result of data type **cannot** include any **Latch**.
10. The gate level simulation cannot include any timing violations without the *notimingcheck* command.
11. After synthesis, you can check DH.area, DH.timing and DH.resource. The area report is valid when the slack in the end of timing report should be **MET**.
12. In this lab, you must use DW_sincos IP from Designware. We will check it in DH.resource as shown in Figure 4.



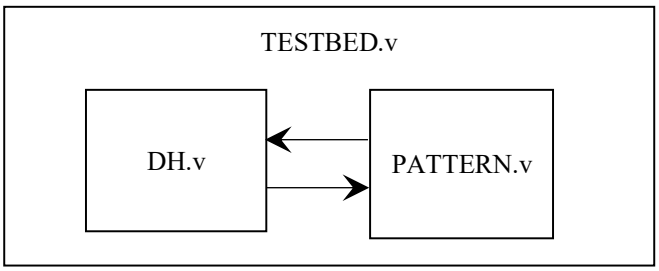
Cell	Module
mult_x_1	DW_mult_tc
mult_x_2	DW_mult_tc
add_x_6	DW01_add
add_x_7	DW01_add
SIN_INIT	DW_sincos

Figure 4. The information of sine and cosine IP in DH.resource file

Grading Policy

1. Function Validity: 75%
2. Performance: 25 %
   - Area * Simulation time: 25%

Block diagram



Note

1. **Please upload the following files on e3 platform before 12:00 on Oct. 14:**
   - **DH_iclab??.v** and **clock_cycle_iclab??.txt** (ex. DH_iclab99.v & 4.5_iclab99.txt)
   - If the uploaded file violating the naming rule, you will get 5 deduct points on this Lab.
2. **Template folders and reference commands:**

   01_RTL/ (RTL simulation) **./01_run**

   02_SYN/ (Synthesis) **./01_run_dc**

   (Check if there is any **Latch** and **Error** in your design in **syn.log)**

   (Check the design's timing in /Report/DH.timing to see if the slack is **MET**)

   03_GATE / (Gate-level simulation) **./01_run**

You should run the clean command (./09_clean_up) first before you rerun each simulation (./01_run).

Sample Waveform

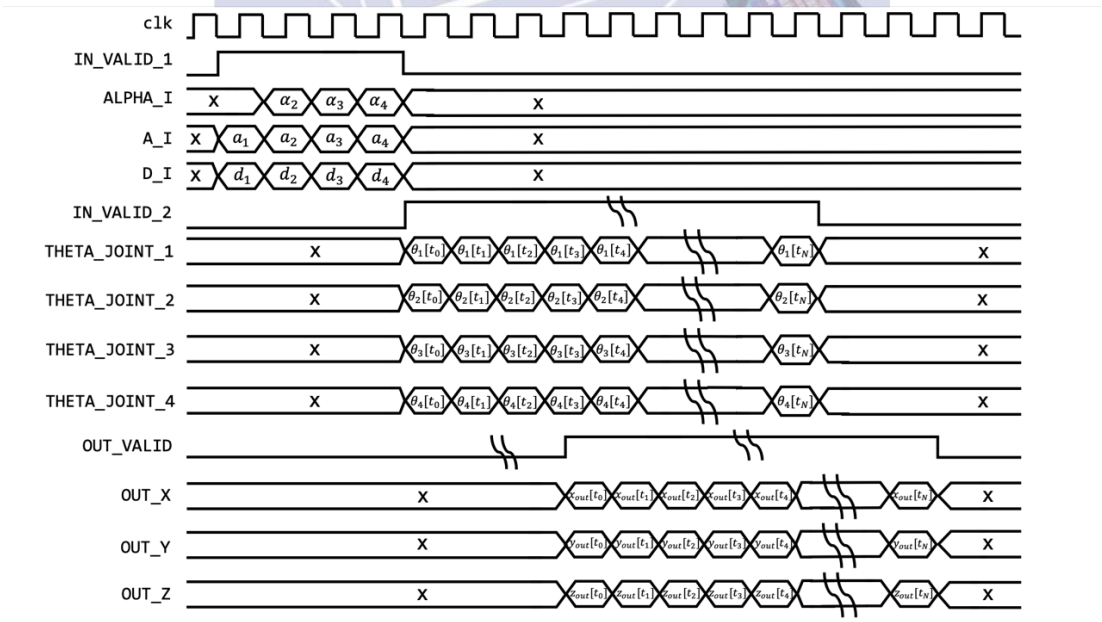


Figure 5. Sample waveform

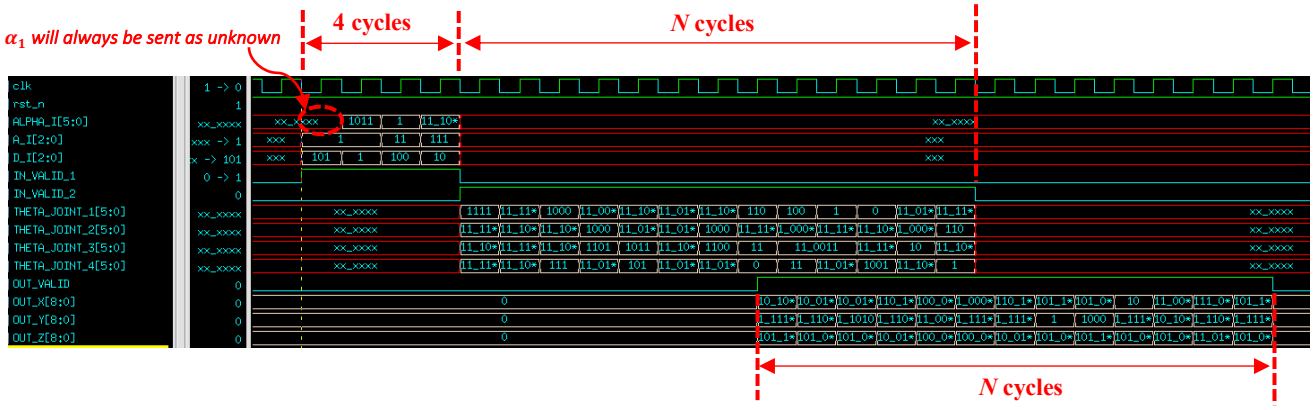


Figure 6. Sample waveform (nWave)

Reference

[1] http://www.timing.com.tw/productPage.asp?id=325