

NCTU-CS Digital System Lab.

LAB 05 – Stack & Queue

Design:

Data Preparation

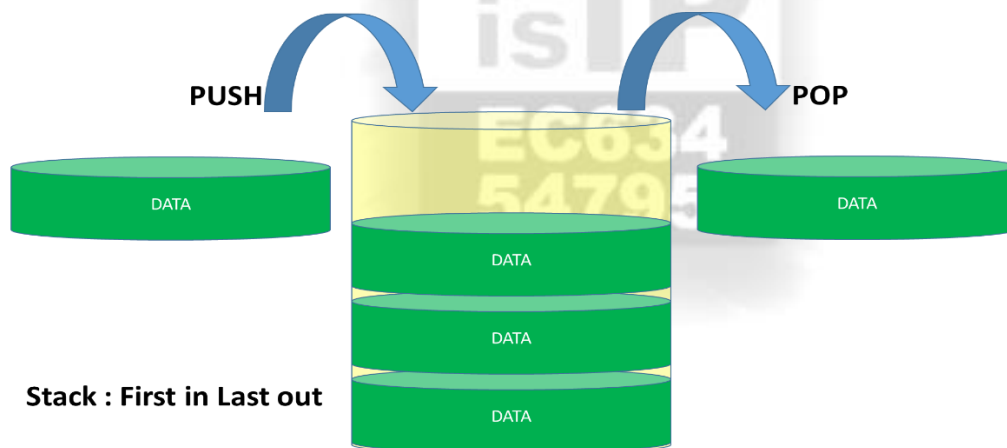
Extract LAB data for TA's directory.

% `tar xvf ~2016dlabta02/Lab05.tar`

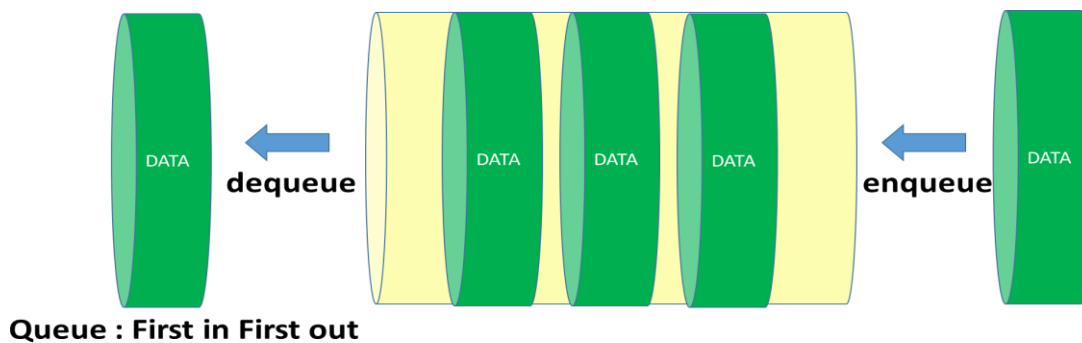
Design Description

You have learned Stack & Queue in Data Structure course. Now we are going to implement them in this Lab.

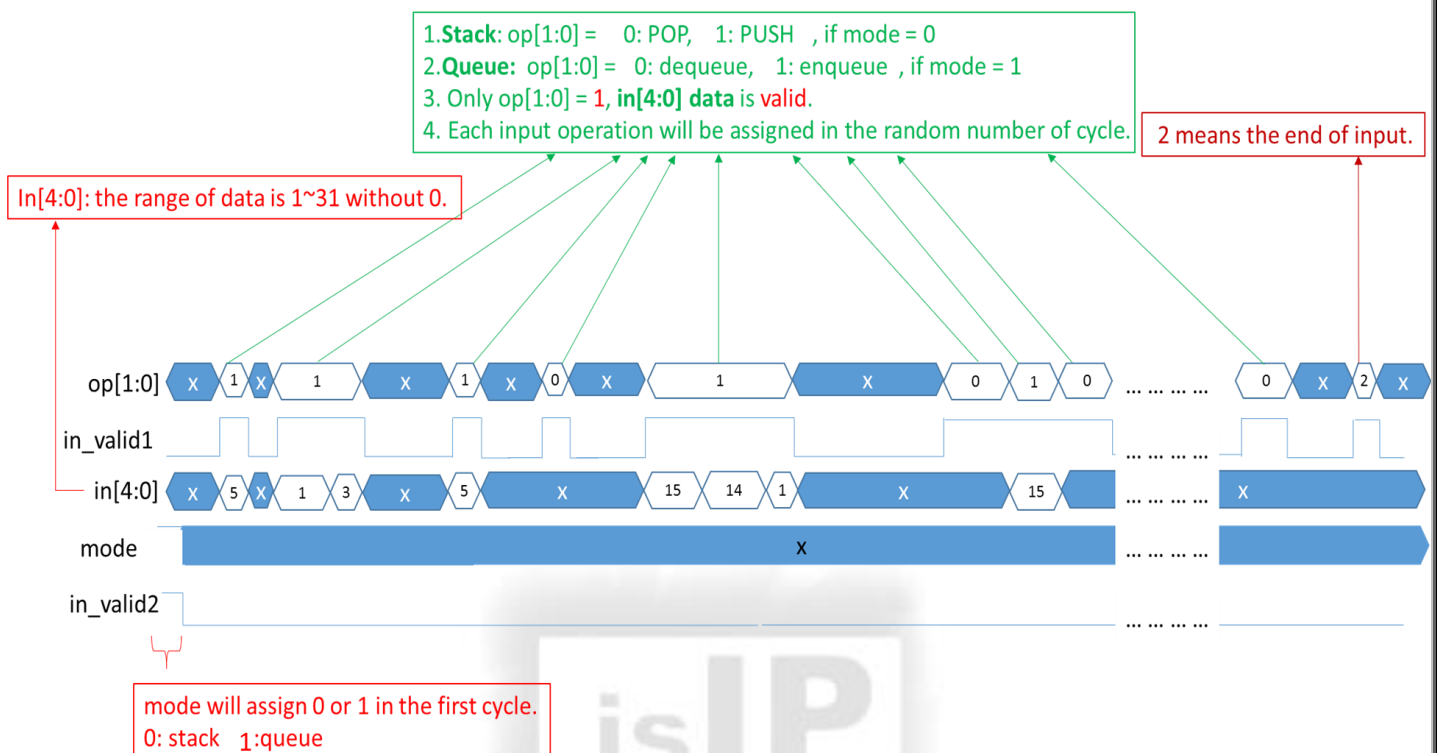
Stack:



Queue:



Input:



There are two mode, 0: stack and 1: queue, you get in the first input cycle from **mode** channel with in_valid2 high. You will receive a large number of operations from **op[1:0]** in **random** cycles between each input. The total number of operations are unknown but you receive **2** in the end of input. When **op[1:0]** is **1**, **in[4:0]** is **valid** for both of mode.

You need to implement two data structures of which size are **10** and execute all the operation you get from Pattern.

Finally, sort the remaining data from data structure in **descending order** and output them in 10 continuous cycles sequentially. If the number of the remaining data are **less** than 10(ex: pop more than push), output **0** to make up 10 cycles.

Ex:

After you execute all operations the remaining data in stack: 5, 2, 30, 23, 7, 6

Sort in descending order: 30, 23, 7, 6, 5, 2

Output:

If the number of data is less than 10, output **0** to make up 10 cycles:

30, 23, 7, 6, 5, 2, 0, 0, 0, 0

You should output **30, 23, 7, 6, 5, 2, 0, 0, 0, 0** in 10 continuous cycle.

Your goal is to accomplish this design by above rules and output the correct answer.

Inputs

1. input data for **in[4:0]** will be input in random cycles with **in_valid1** high.
2. **mode** is valid only at the first input cycle with **in_valid2** high. 0 is stack, and 1 is queue.
3. All signals are unsigned integer.
4. **Stack:** if mode = 0, **op [1:0]**: 0: POP, 1: PUSH, 2: end of input.
5. **Queue:** if mode = 1 **op [1:0]**: 0: dequeue, 1: enqueue, 2: end of input.
6. Input information :

Input port	Bit Width	Description
in	5 bits	only at op[1:0] =1 range of data: 1~31without 0
op	2 bits	If mode = 0 , 0 : POP, 1: PUSH, 2: end of input If mode = 1 , 0 : dequeue, 1: enqueue, 2: end of input
mode	1 bits	Assign in the first cycle. 0 : stack, 1 : queue
in_valid1	1 bit	High when op[1:0] is valid
in_valid2	1 bit	High when mode is valid
clk	1 bit	Clock
rst_n	1 bit	asynchronous active-low reset

Outputs

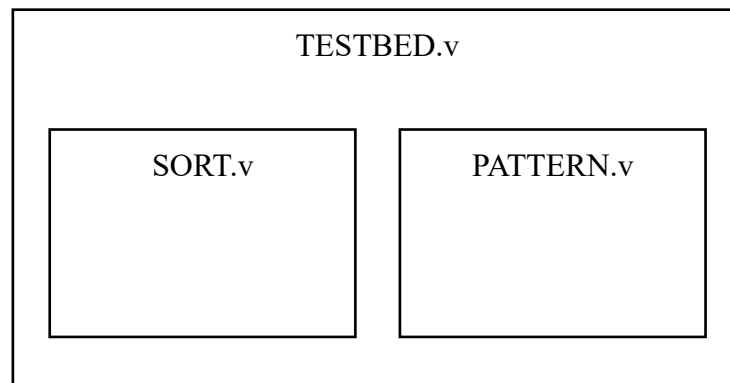
1. Your answer should be output at out in **10 continuous cycles**
2. Output information :

Output port	Bit Width	Description
out	5 bit	output after sorting
out_valid	1 bit	High when out is valid

Specifications

1. Top module name : SORT (File name : SORT.v)
2. The clock period is 4 ns.
3. It is a **asynchronous reset** and **active-low** architecture. Signal out and out_valid should be zero after reset.
4. Input signal: **in[4:0]**, **op[1:0]**, **mode**, **in_valid1**, **in_valid2**, **clk**, **rst_n**
5. Output signal: **out[4:0]**, **out_valid**
6. All inputs will be given at negative edge of clk, and all output is synchronized at positive edge of clk.
7. The latency should not be **more than 100**.
8. Grading policy:
 - Design: 70%
 - Time: 15%
 - Area: 5%
 - Question: 10%

Block Diagram



Note

1. Simulation step:
 - Put your Pattern and Testbench in 00_TESTBED
 - Simulation to check design : ./01_run
 - Show wave to debug: nWave &
 - Go to folder 02_SYN/ and check synthesis : ./01_run_dc
 - Go to folder 03_GATE/ and check s : ./01_run.f
 - Clear up : ./09_clean_up
2. Please add your student ID and name to the file name of .v file before upload file on e3 platform:
SORT_0556123_陳小明.v
3. Sample waveform:

