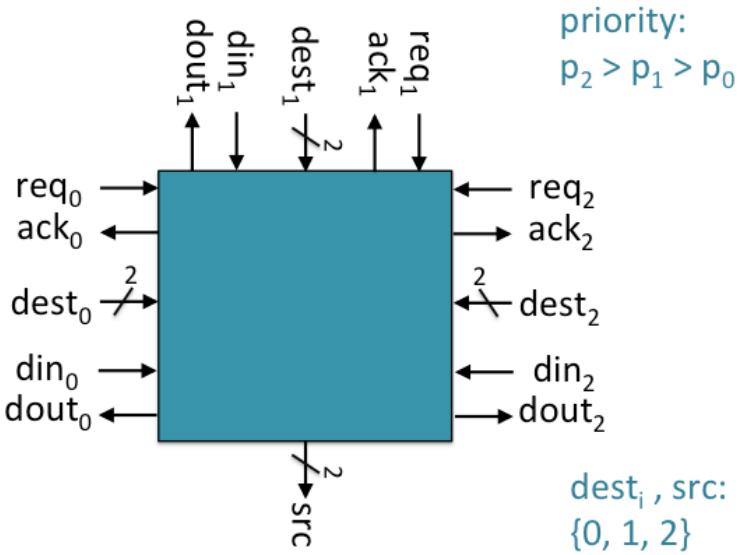
Lab exercise 2 : Three-Port Switch

Design and implement a 3-Port Switch with the functional description given below.

The switch has 3 ports, numbered 0, 1 and 2. Each port has the following inputs and outputs.

din_i : data input
$dout_i$: data output
$dest_i$: destination port number (input)
req_i : request (input)
ack_i : acknowledgement (output)



The function of the switch is to connect din_i of i^{th} port to $dout_j$ of j^{th} port when there is a request at i^{th} port to do so. The request for connection is indicated by req_i and the destination j is specified by $dest_i$.

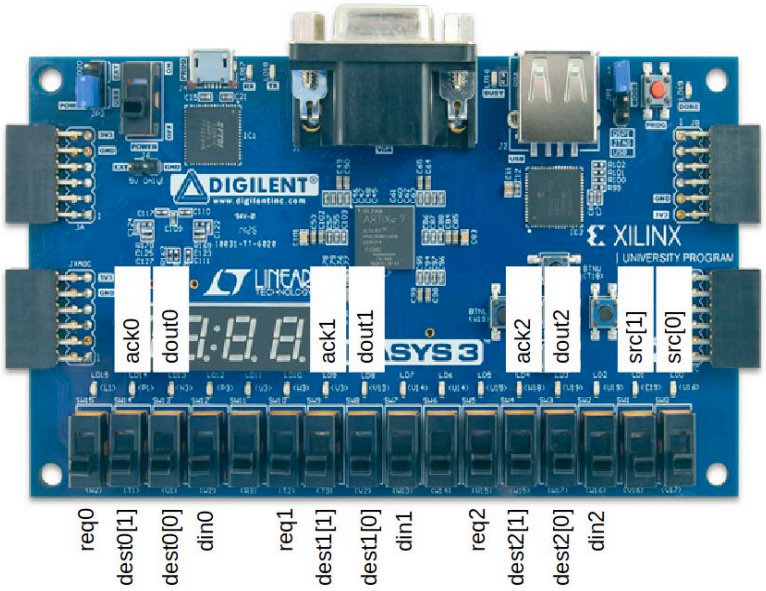
The switch is capable of connecting only one source to its destination at a time. It arbitrates among concurrent requests as per the priorities assigned to the ports. The priorities are as follows, p_i indicating the priority of i^{th} port.

$p_2 > p_1 > p_0$

Switch conveys the result of arbitration through ack_0, ack_1 and ack_2. Also, the port number of the port that wins arbitration is given by src output. For example, if port 1 and port 2 simultaneously request then, port 2 will win the arbitration and its port number will show up on src.

Carry out the design in two phases. First design the building blocks such as priority encoder, mux, de-mux etc. Then use these building blocks to design the whole switch. It will be explained to you in the lab how a design can be encapsulated as a new component.

Use slide switches for inputs (req, dest, din). Display dout and src outputs on LEDs. Figure below shows the pin mapping:



Port addressing is given below-

Port Number	Port Address
0	00
1	01
2	10

Assume the port switch is always driven with valid inputs (example: port address “11” is not requested). Further, none of the ports requesting for data transfer is a valid condition.

Please follow the naming as indicated below:

Entity Name	lab2_port_switch		
Pin Name	**Direction**		**Purpose**
req0	Input	port0	request
dest0[1:0]	Input		destination port number
din0	Input		data input
ack0	Output		acknowledgement
dout0	Output		data output
req1	Input	port1	request
dest1[1:0]	Input		destination port number
din1	Input		data input
ack1	Output		acknowledgement
dout1	Output		data output
req2	Input	port2	request
dest2[1:0]	Input		destination port number
din2	Input		data input
ack2	Output		acknowledgement
dout2	Output		data output
src[1:0]	Output		source port

Note that this design suffers from the following limitations. We may consider overcoming these limitations in a later exercise.

- It allows only one connection at a time.
- A request has to persist till it gets an acknowledgement. There is no queuing of requests.
- A request on a higher priority port can abort an active connection made from a lower priority port.
- Requests at lower priority port(s) can be denied acknowledgement indefinitely by extensive usage of the switch by higher priority request(s), leading to starvation.