Implementation of Circular Queues (FIFO) in Hardware

Problem Desciption:

Circular Queue is a linear data structure in which the operations are performed based on FIFO (First In First Out) principle and the last position is connected back to the first position to make a circle.

Create a circuit that works like a circular queue. The inputs to the circuit are ‘d\_in’ , ‘write’ and ‘read’. The outputs are ‘d\_out’, ‘full’ and ‘empty’.

The signals ‘d\_in’ and ‘d\_out’ are 16 bit data input and output respectively. The 1 bit ‘write’ and ‘read’ signals are high when the user wants to write or read from the queue respectively. The 1 bit ‘full’ and ‘empty’ signals are high whenever the queue is full or empty respectively.

Literature Survey:

Working of a circular queue:

A circular queue work as follows:

* Two pointers called ‘front’ and ‘rear’ are used to keep track of the first and last elements in the queue.
* When initializing the queue, we set the value of ‘front’ and ‘rear’ to 0.
* On enqueing an element, we place the new element in the position pointed by ‘rear’ index and circularly increase the value of ‘rear’ index.
* On dequeueing an element, we return the value pointed to by ‘front’ and circularly increase the ‘front’ index.
* Before enqueing, we check if queue is already full. The queue is full when rear + 1 is equal to front.
* Before dequeuing, we check if queue is already empty. The queue is empty when front is equal to rear.

Verilog modules description:

* bin\_counter

This module takes a 1 bit input ‘in’ and gives a ‘4’ bit output ‘out’. The output represents a binary counter whoose value goes from ‘0000’ to ‘1111’. The value increases by 1 everytime the ‘in’ signal goes high. This module is used to keep track of the **front** and **rear** addresses of our circular queue. This has been implemented using 4 d-flip-flops which are connected to 4 ‘half-adders’.

* reg\_file\_slice

This module takes in 1 bit data input ‘d\_in’, two 4 bit addresses ‘wr\_addr’ and ‘rd\_addr’, two 1 bit signals ‘wr’ and ‘rd’ and gives a 1 bit data output ‘d\_out’. It consists of 16 rows of memory (d-flip-flops), 1 bit each. Data can be written and read every clock cycle using the 4 bit addresses.

* full\_empty

This module takes two 4 bit addresses ‘front’ and ‘rear’ and gives two 1 bit signals ‘full’ and empty. The signals ‘full’ and ‘empty’ go high by comparing the two addresses for the queue full and queue empty condition.

* memory

This module is similar to the implementation of the register file given in our lab assignments. However, this module contains only a single read port as we need to read a single value in a clock cycle. It consists of 16 instantiations of the ‘reg\_file\_slice’ module to store 16 bits of data.

* fifo

This is the main module that instantiates all the above modules as shown in the previous circuit diagram. It takes in the inputs as mentioned in the problem statement and gives the required outputs.