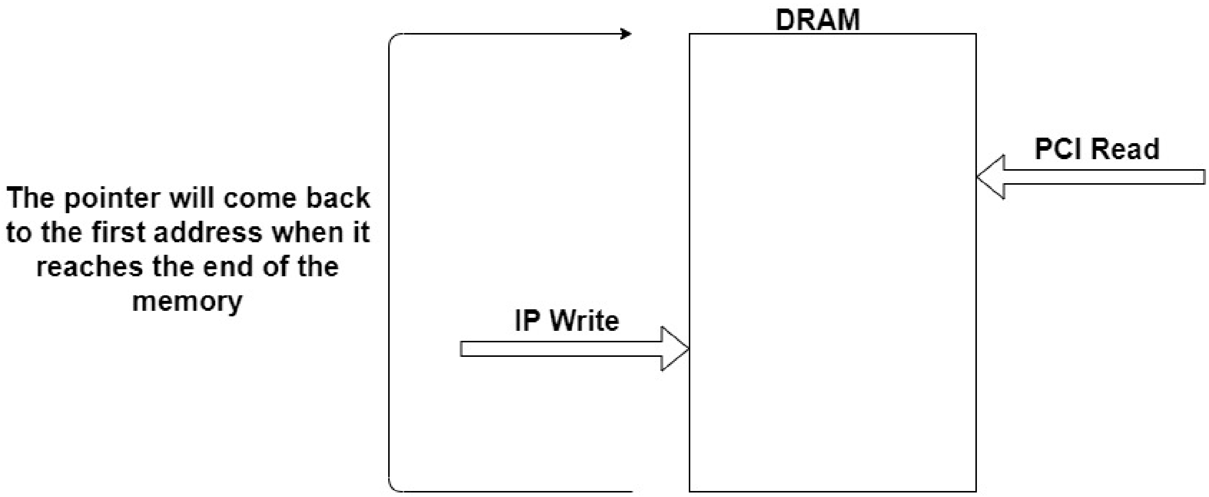
Hi Oliver,
In this short document, I represent the general idea of what I'm currently doing and there are some questions I want to ask you.

I. General Idea:

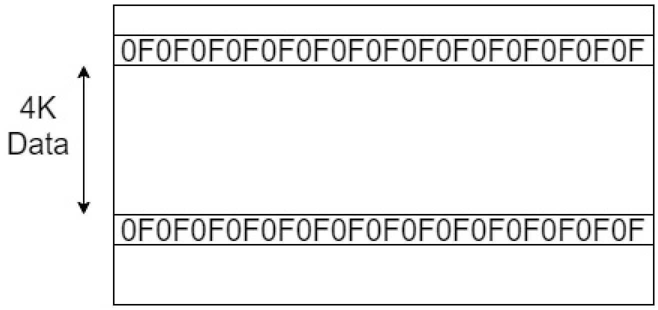

DRAM
PCI Read
The pointer will come back to the first address when it reaches the end of the memory
IP Write

The idea is that the IP writes data consecutively and the host will follow the IP to read data back over PCI interface. Because the speed of the IP and PCI are different, we need a way to synchronize the IP and PCI to make sure:

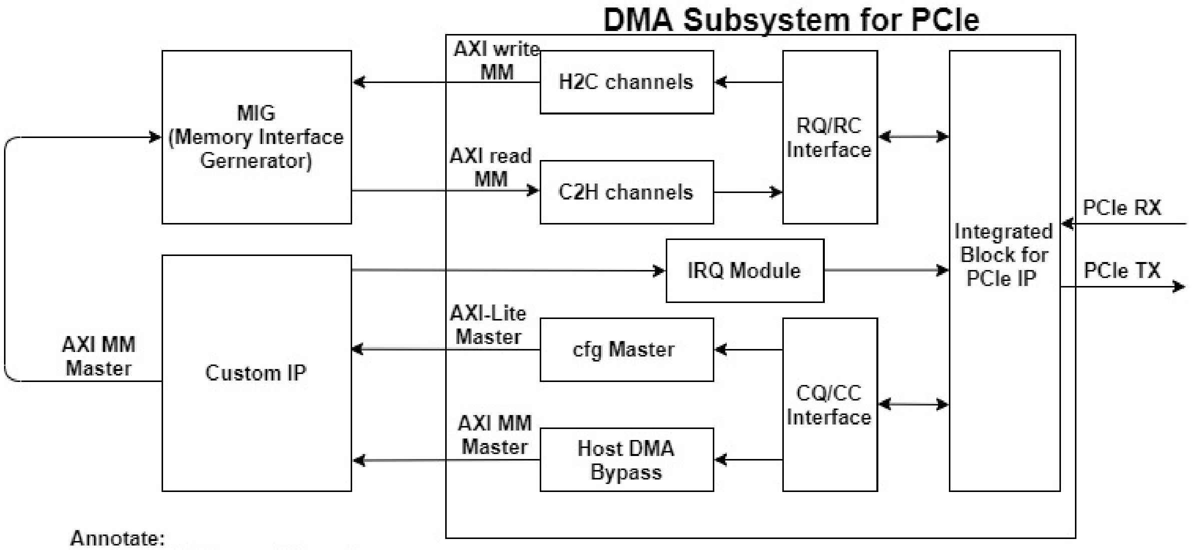
- The PCI's read speed must be faster than the IP's write speed so we don't mistakenly read the garbage values.
- The IP must not overwrite the old values that PCI hasn't read, because the write pointer jumps back to the start address and overwrites the old value when it reaches the end of the DRAM.

To achieve that I have a few ideas as follows: (All methods will update the current PCI read address in the IP core. The IP bases on this information to make sure not overwrite the data hasn't read)

- **Polling address:** the host program will continuously poll the current write address of the IP and wait until the data is accumulated enough in the memory. As soon as the data is sufficient, the PCI starts reading to catch up with the IP.
- **Polling start signal:** almost the same as the first approach, but the IP manages the data accumulation itself. After writing a predefined amount of data, it turns on a signal. The host programm polls this signal to know when it should read data.
- **Interrupt read trigger:** the IP generates an interrupt signal to the host program after writing a predefined amount of data. This one is more complex and requires more time.
- **Predefined pattern:** the IP writes some specific patterns at the start and the end of a predefined amount of data . The host programm bases on these patterns to detect the valid values. By this way, the host can read data continuously and doesn't need to synchronise with the IP. The host can extract the valid data based on the pattern. The idea is illustrated in following figure:


4K
Data
0F0F0F0F0F0F0F0F0F0F0F0F0F0F0F0F
0F0F0F0F0F0F0F0F0F0F0F0F0F0F0F0F

II. Block design Idea:



Annotate:
AXI MM: AXI Memory Mapped
cfg Master: Configure Master

The above picture is the general block design idea I'm currently doing. Basically, we have 4 channels which share the same AXI-Full master to access the memory and our IP also has a AXI-Master interface to write data to the memory. Besides that, the host program can get access to our IP (to poll address or start signal) by the AXI-Lite master of the configured Master block or the host DMA bypass. There could also be an interrupt signal to synchronise between our IP core and the host program.

III. Questions:

1. I have represented you 4 approaches that I think could be implemented. I don't know exactly what bandwidth we have now and I'm not sure which one suits best for our needs. Could you please give me some advice? Which one do you think is the good direction to go?
2. Do you have any other idea how to synchronise the PCI and our IP core?
3. Could you give me some feedback about the block design? Should I modify anything?
4. Now I'm using AXI MM master to write data to memory. I don't know how you will transfer data to our IP core. Do you use AXI-Stream or AXI Memory Mapped? Is the interface I'm using okay or should I change it to AXI-Stream and then use AXI-DMA?