Does it make sense to define I/O subsystems that use a combination of memory mapping and command driven, communication

**Yes**, it does make sense in defining I/O subsystems that use a combination of memory mapping and command driven communication since when one may need in finding the path in which the I/O subsystem uses, one can get to know and also if there are fault in the I/O subsystems one may need to check the path they use.

Registers connected with each port are one way of connecting with devices.

Memory-mapped I/O is another method for connecting with devices.

In this approach, the device is assigned a portion of the processor's address space, and communications are carried out by reading and writing directly to and from those memory locations.  
Memory-mapped I/O is ideal for devices like graphics cards that need to move huge amounts of data quickly. Memory-mapped I/O can be used instead of traditional registers or in conjunction with them more frequently. Graphics cards, for example, still employ registers for control information like selecting the video mode. When a process is allowed to write directly to the address space used by a memory-mapped I/O device, a possible problem arises.  
Note that memory-mapped I/O differs from direct memory access (DMA).

The main benefit of a memory-mapped I/O subsystem is that the CPU may transport data between the CPU and a memory-mapped I/O device using any instruction that accesses memory. Although the MOV instruction is the most common method of sending and receiving data from a memory-mapped I/O device, any instruction that reads or writes data in memory is allowed. If you have a read/write I/O port, for example, you can read the port, add data to the value read, and then send data back to the port using the ADD instruction.