CPU-FPGA communication protocol

1. Introduction

The following is a description of the proposed communication protocol between the CPU and the FPGA. The main design goals were the simplicity of the hardware (in terms of area) and the performance. For achieving these two objectives, we divided part of the effort between the hardware implementation and the software tools.

With the present document, we will mainly focus on the hardware side, while briefly detailing what software functionalities are in in turn needed.

The document is organized as follow: section 2 illustrates the interface of the internal buffer and describes the overall functionalities. With section 3, we provide an overall description of the IP Manger interfaces. Sections 4, 5 describe read/write transaction and interrupt requests respectively.

2. Buffer Interface

In order to simplify the design we decided to implement (by means of a synthesizable VHDL design) the buffer as a full custom dual-port register file. Figure 1 shows the interface. Port_0 is dedicated for the communication with CPU, whereas Port_1 is controlled by the IP manager. It is worth noting that there is an additional output port, called `ROW_0`. The purpose of this port is to enable an easy access to the row 0 by the IP manager, which (as discussed in details in section 4) uses this row to perform the correct route for reading/writing the selected IP.

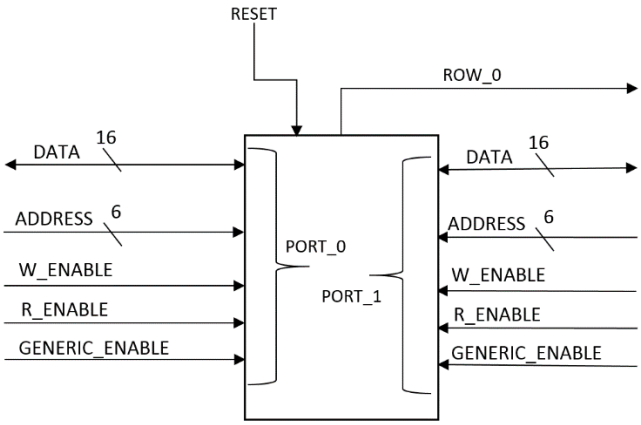

Figure 1 - Buffer interface

Another important aspect is that, as it happens with real register files, the module is completely asynchronous. Indeed, the presence of a clock would slow down the performance of the IP Manager. In order to read/write on one of the two ports, the generic enable signal must be asserted as well as the required control signal (`W_ENABLE/R_ENABLE`). Then the correct values must be placed on data and address lines.

3. IP Manager interfaces

Main tasks of the IP Manager are the correct selection of the target IP and the interrupt handling. Figure 2 shows the overall architecture of the whole system.

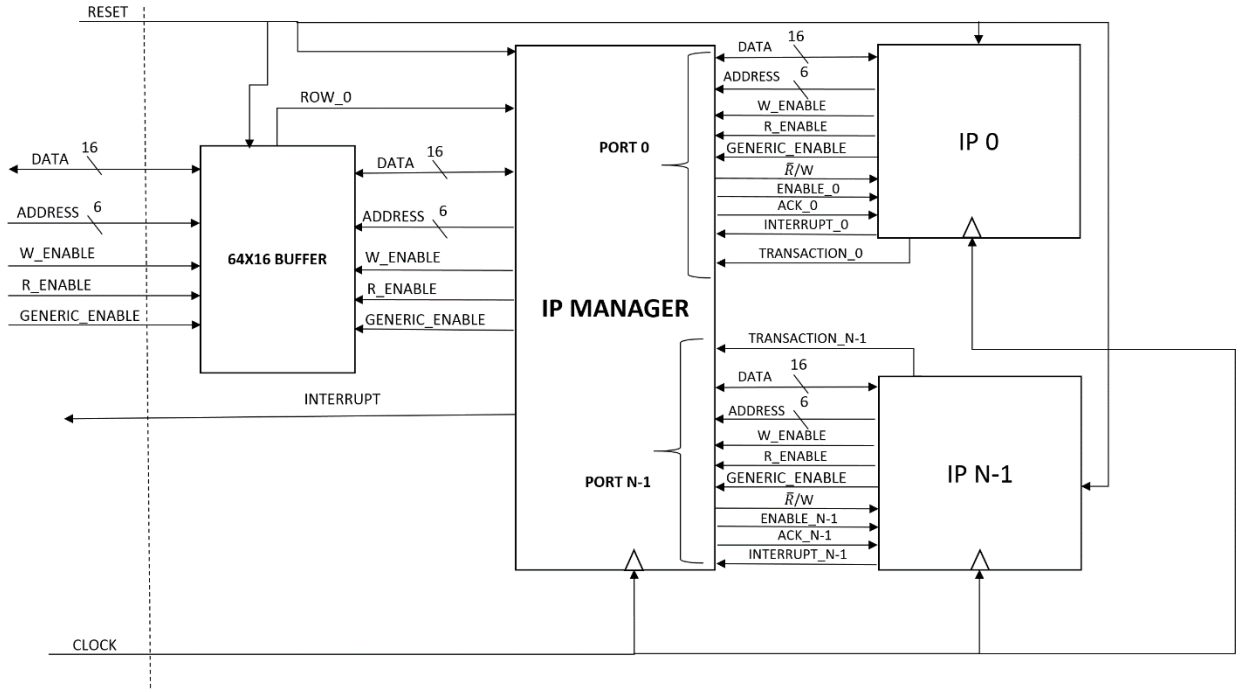

Figure 2 - System architecture

As shown in the figure, the IP manger has a standard interface with the buffer. This interface is used to redirect data to/from the target IP and to access the buffer itself (e.g. to write the row 0). However, it is provided with a direct access (read-only) to the internal element to speed-up the routing process whenever a new transaction begins. It is important to underline that for each instantiated IP there are a set of standard signals and some protocol-specific signals. The former are the very same signals used for accessing the buffer, whilst the latter are used for specifying which operations the target IP should start and handling interrupt requests.

4. Read/Write Transaction

Whenever the CPU wants to begin a transaction with one of the IPs present on the FPGA, it writes at address 0 of the buffer a data packet compliant with the following specifications:

Bit(s)	Purpose	Value(s)
Bit 15	It specifies whether the transaction is a read or write transaction.	Read = 0, Write = 1
Bit 14	Specifies whether the current transaction is due to an interrupt request	Normal = 0, Interrupt = 1

Bit 13	Signals the begin/end of a transaction	Begin = 1, End = 0
Bit 12	Unused	Unused
Bit 11-0	The physical address of the target IP (in general is different form the IP number)	From 0 up to N-1

In details, this is what happens during a read or write transaction:

- READ transaction: the CPU writes at address 0 the control word for the IP manger. In particular, the IP manager will always assume that the IP address that is written from bit 11 up to bit 0 is already the physical address of the IP (i.e. one of the output port) and it will not perform any further translation. Hence, it is up to the software environment to properly connect the right IP to the right port and to configure correctly the device drivers. Figure 3 shows the association that the software tool is supposed to do:

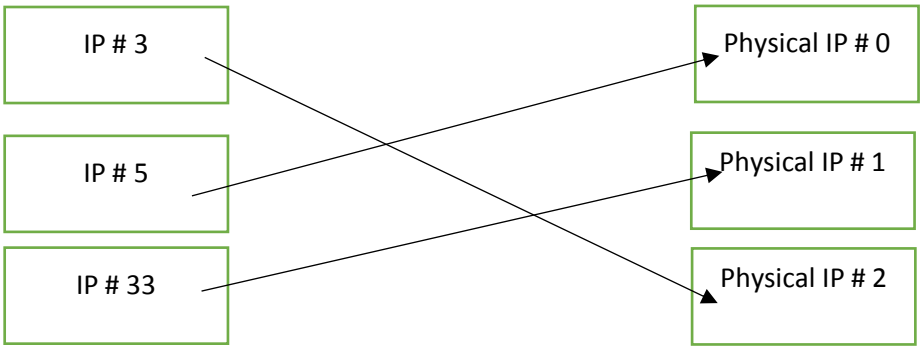

Figure 3 - Association IP number with physical IP

Assuming that the read transaction begins due to a normal transaction (i.e. no interrupt request), bit 15 will be set at 0, bit 14 at 0 and bit 13 at 1. Then when it finishes the transaction, it performs again a write at address 0 of the buffer with the very same pattern as before, but bit 13 is kept at 0. It is important to signal the begin and the end of a transaction because this information will be exploited by the IP manager to understand whether it can signal to the CPU that an interrupt arose (more details in section 5.

- WRITE transaction: the same as before, except the bit 15, which is set to 1.

Of course, both CPU and IP have to drive the buffer control signals according to the specifications of section 2.

Form the viewpoint of the IP manger, the behaviour is also the same whether it is a read or a write transaction. Depending on the IP address, it will enable one of the IPs (asserting the `ENABLE_x` signal of Figure 2). The IP manager does not care about how many packets have to be written or read, since it is IP-dependent. It must only keep the port associated to the target IP open as long as the value written in the IP address field do not change. Of course, since the only means of communication between the IPs and the CPU is the buffer, the IP manger is in charge for specifying whether the target IP has to start a read or write transaction with the internal buffer. For this purpose, each IP has an `R/W` signal driven by the IP manger itself. In particular, if the CPU starts a write transaction it means that it will write data to the buffer. From the IP point of view, it means that it has to read the data from the buffer. Thus, the IP Manager drives the `R/W` signal to READ. The same reasoning applies also for a read transaction from the CPU. Note that, since this time the IP has to write to the buffer, it is up to the software layer (i.e. device driver written by the IP vendor) to wait the right amount of time before start reading the buffer.

5. Interrupt Handling

The association depicted in Figure 3 will be driven by priority of the IP (specified by the user). Highest priority will be always associated to IP manager port 0 (properly connected to the right IP), and the lowest to port N-1. The IP manager will always assume that IP connected to port 0 is the one with the higher priority.

Since the architecture is a master (CPU) slave (FPGA) architecture, the IP manager must guarantee that an interrupt request from one or more of IPs do not interrupt a transaction started by the master. For doing so, the manager leverages bit 13. Only when it is set to zero it signals to the CPU, through the `INTERRUPT` signal, that an interrupt service routine has to start. At the very same time, it writes to the address 0 of the buffer the IP address of the IP requesting the interrupt.

Before starting any transaction, the CPU reads the content of address 0 so that it knows which is the requesting IP. At this point, the CPU starts a typical read/write transaction, with the exception that bit 14 now is set. Doing so the manager knows that the interrupt has been received, and it can clear `INTERRUPT` signal.

It is worth noting that, if an interrupt request arrives while there is still an ongoing transaction, the manager must maintain the interrupt request alive. Indeed, the requesting IP will not clear the `INT_x` signal until it receives the `ACK_x` signal. The manager will raise this signal if and only if there are not any further transaction.

As a final remark, the manager must also avoid to serve interrupts while an IP is still transferring data to/from the buffer. For this reason, there is a further signal called `TRANSACTION_x` (Figure 2) for each IP. This signal is set whenever the IP starts accessing the buffer, and as soon as it is set, the manger will not serve any interrupt request. Hence, in order to serve an interrupt request two conditions must hold: bit 13 of address 0 must be set to zero (end of any transaction by the master) and all the `TRANSACTION_x` signals must also be at zero.