AXI4-Stream Upsizing/Downsizing Data Width Converters for Hardware-In-the-Loop Simulations

Luis Vega, Philipp Schläfer, Christian de Schryver.
Microelectronic Systems Design Research Group
University of Kaiserslautern, Germany
e-mail: vegaluisjose@gmail.com, {schlaefer, schryver}@eit.uni-kl.de

***Abstract*—Hardware prototyping is an essential part in the hardware design flow. Furthermore, hardware prototyping usually relies on system-level design and hardware-in-the-loop simulations in order to develop, test and evaluate intellectual property cores. One common task in this process consist on interfacing cores with different port specifications. Data width conversion is used to overcome this issue. This work presents two open source hardware cores compliant with AXI4-Stream bus protocol, where each core performs upsizing/downsizing data width conversion.**

***Keywords*-Hardware-in-the-loop, downsizing, upsizing, data width converter, AXI4-Stream.**

I. INTRODUCTION

Nowadays, IP-core development involves regularly Hardware-In-the-Loop (HIL) simulations. Furthermore, reconfigurable embedded processors and bus architectures are required in order to validate IP-core functionality [1]. Therefore, HIL relies on system-level design. One common task in system-level design is assembling components. This assembly or interconnect is done either between intellectual property (IP) cores or, an IP-core and a bus system.

Although interconnecting components may seem a trivial task, there are situations where it can not be done directly. The reasons are two fold: one, at block-level, is related to different number of ports. One IP-core may have more ports than other. Secondly, at port-level, because of word sizes. For example, component ports may have different word sizes such as 8, 32 or 64 bits. Generally speaking, a hardware developer requires assembling components under limited input and output ports. A natural solution to this issue is upsizing and/or downsizing data width through converters. However, it turns out that these data width converters are built to solve particular interconnect issues avoiding flexibility and reusability.

Our contribution is to provide an open source hardware cores (converters)[1] that perform upsizing and downsizing data width and compliant with AXI4-Stream bus protocol [2]. Furthermore, the interfaces were developed in VHDL, providing simple, compact and flexible features that can be easily integrated and reused in hardware-in-the-loop simulations.

[1]Source code is freely available on http://ems.eit.uni-kl.de/msdlib

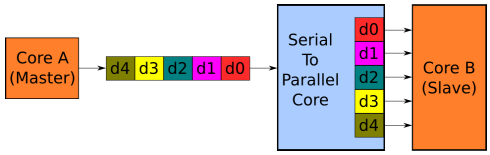

Figure 1. Upsizing data width scenario

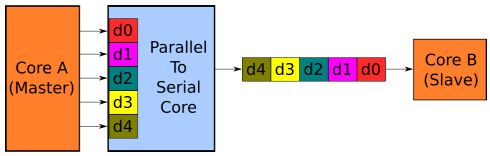

Figure 2. Downsizing data width scenario

The rest of the paper is organized as follows: In Section II, a brief background is given. Then, interfaces are described in Section III. Next, a typical application, hardware-in-the-loop, is shown in Section IV. Finally, results and a short conclusion are given in Sections V and VI respectively.

II. THEORETICAL BACKGROUND

A short review about technical terms cover by this report is given in the following subsections.

A. Upsizing and downsizing data width

Upsizing data width, also known as Serial-To-Parallel data conversion, is required when two components, master and slave, need to be interconnected and the master component has fewer ports than the slave component. For example, a typical situation is given in Figure 1. In this case, the *Core A* (Master) needs to be plugged to *Core B* (Slave) but the numbers of ports among them does not match. Furthermore, *Core A* has fewer ports than *Core B*. Therefore, the upsizing interface is needed here in order to interconnect the cores.

Similarly, downsizing data width, also known as Parallel-To-Serial data conversion, is useful when it comes to component interconnect. However, the situation here is that master

Table I
AXI4-STREAM MASTER/SLAVE SIGNAL LIST

Signal	Description
tdata	Data payload
tlast	Boundary of a packet
tvalid	A transfer takes place when both tvalid and tready are asserted
tready	A transfer takes place when both tvalid and tready are asserted

Table II
UPSIZER/DOWNSIZER COMPONENT GENERIC PARAMETERS

Name	Type	Description
G_RESET_ACTIVE	std_logic	Set reset active low or high
WIDTH	integer	Data word size
NUM_REG	integer	Number of ports

component has more ports than the slave component. For example, a typical scenario is given in Figure 2.

B. AXI4-Stream bus protocol

The AXI4-Stream bus protocol is a subset of AMBA AXI4 protocol. Furthermore, AXI4-Stream is a interconnect standard optimized for FPGAs [3]. It provides a streaming interface for point-to-point communication between components.

AXI4-Stream consists of master and slave ports, which are used for write and read respectively. Both ports share the same meaning for control and data signals but have different port direction. For example, master-(tdata, tlast, and tvalid) ports are outputs, while slave-(tdata, tlast, and tvalid) are inputs. Similarly, the master-tready port is an input, while slave-tready port is an output. The signal description list for AXI4-Stream is shown in Table I.

III. CORE INTERFACE DESCRIPTION

In this section, the upsizer and downsizer interface specifications are provided. The specification involves the following aspects of hardware implementation: component generic parameters and port interface description. It is important to mention that component generic parameters are the same for both cores. Thus, there are not distinction between them in the following subsection.

A. Component generic parameters

The cores can be adjusted to a specific configuration through component generic parameters. The parameters are related to reset configuration, data word size, and number of registers. This last parameter, number of registers, can be seen also as the number of ports for the interface. A more detailed information about these parameters is shown in Table II.

The main parameters are WIDTH and NUM_REG, which define the data signal size as is shown in Figure 3. Besides the data signal size, it is important to denote how data

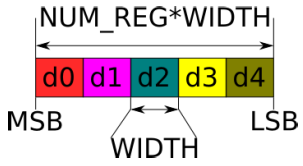

Figure 3. Data arrangement

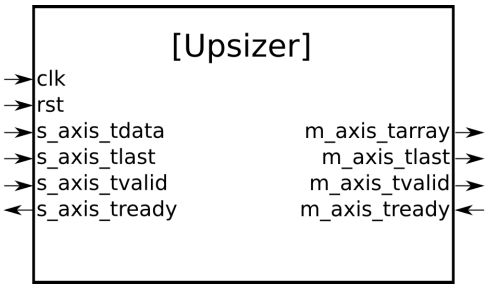

Figure 4. AXI4-Stream upsizer schematic view

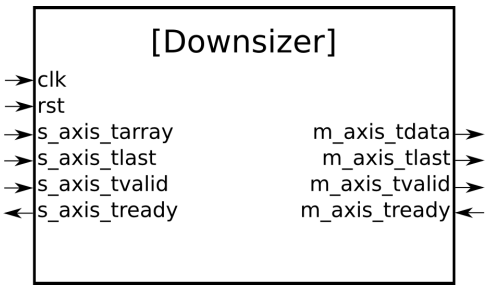

Figure 5. AXI4-Stream downsizer schematic view

is arranged. The convention here is that the first received value is placed in the Most Significant Bits (MSB) of the signal. Then, the second received value is placed in the following WIDTH-bits. This is performed till the signal is completely filled, reaching the Least Significant Bits (LSB). For example, in the Figure 3, d_0 represents the first received value and d_4 represent the last received value.

B. Port interface description

The schematic view for the upsizer and downsizer core are given in Figure 4 and 5 respectively. The following notation are used for the interfaces. AXI4-Stream Slave ports are denoted with "s_" at the beginning of the port name. Meanwhile, AXI4-Stream Master ports are denoted with "m_". The port names ending with "tarray" and "tdata" correspond to the ports where the payload conversion occurs. For example, the upsizer core reads data serially from "s_axis_tdata" port and parallelizes it to the "m_axis_tarray" port. Similarly, the downsizer reads data in parallel from "s_axis_tarray" port and serializes it to the "m_axis_tdata". Data arrangement in "tarray" ports is done as stated before in *component generic parameters* subsection.

The main AXI4-Stream control ports are "tvalid" and

Table III
UPSIZER - PORT INTERFACE INFORMATION

Signal	Direction	Width(bits)	Type
s_axis_tdata	input	[WIDTH-1:0]	data
s_axis_tlast	input	1	control
s_axis_tvalid	input	1	control
s_axis_tready	output	1	control
m_axis_tarray	output	[NUM_REG*WIDTH-1:0]	data
m_axis_tlast	output	1	control
m_axis_tvalid	output	1	control
m_axis_tready	input	1	control

Table IV
DOWNSIZER - PORT INTERFACE INFORMATION

Signal	Direction	Width(bits)	Type
s_axis_tarray	input	[NUM_REG*WIDTH-1:0]	data
s_axis_tlast	input	1	control
s_axis_tvalid	input	1	control
s_axis_tready	output	1	control
m_axis_tdata	output	[WIDTH-1:0]	data
m_axis_tlast	output	1	control
m_axis_tvalid	output	1	control
m_axis_tready	input	1	control

”tready”. The ”tvalid” port shows whether there are valid data or not. In the meantime, the ”tready” port shows whether the slave or master port is ready for reading or writing data respectively. As golden rule, data transfers will take place only when both tvalid and tready signal are asserted.

For simplicity, ”tlast” functionality is not implemented on these cores. Therefore, ”tlast” can be connected to a dummy signal or assigned to zero depending on the component instantiation. Finally, Tables III and IV summarize the port interface description.

IV. TYPICAL APPLICATION

A typical application, where upsizing/downsizing data width is needed, is in hardware-in-the-loop simulation. The reason for HIL simulation is testing IP-cores under development or legacy IP-cores in a heterogeneous environment, which consists of reconfigurable processors, specialized IPs, and bus systems.

Overall, this application consists on interfacing IP-cores to a particular bus protocol, which in this case is the AXI4-Stream bus. As stated before, AXI4-Stream is suitable for point-to-point communication between streaming IP-cores. Therefore, the IP-core interfacing, for these applications, can be divide in two phases: one related to data width conversion, another to control signal matching.

The data width conversion phase has been covered already throughout this report. On the other hand, the control signal matching is related to interconnecting control signals of the IP-core to the bus. The trivial case here is when both interfaces are AXI4-Stream compliant. However, there are many cases when the IP-core under test is a legacy component

Table V
CONTROL SIGNAL MATCHING

Legacy IP (FIFO)	AXI4-Stream IPs
write_enable	s_axis_tvalid
not(full)	s_axis_tready
read_enable	m_axis_tready
not(empty)	m_axis_tvalid

Table VI
DEVICE UTILIZATION FOR 32-TO-128/128-TO-32 CONVERSION

	Upsizer	Downsizer
Slice Registers	163	166
Slice LUTs	203	206
max. clock frequency	390 MHz	370 MHz

that might have push/pull, stall/valid, or write/read_enable signals. At first glance, it might seem that control signal matching is not possible, but in most cases is feasible by mapping correctly. In other words, it is a naming convention problem. For example, a AXI4-Stream ”tready” signal can be connected to an inverted ”stall” signal. Similarly, a ”tvalid” signal can be connected to a ”write_enable” signal. An example of control signal matching for legacy IP-cores is given in the Table V. The example shows how control signals of a legacy FIFO IP-core can be mapped to AX4-Stream control signals.

V. RESULTS

In order to get a rough idea of area utilization, post place & route results are given in Table VI. The aforementioned cores were synthesized using Xilinx ISE 14.4 and the Xilinx ML605 board with a Virtex-6 device as target.

VI. CONCLUSION

This work presents open source AXI4-Stream upsizer and downsizer cores. Furthermore, the main contribution of this work is to provide flexible and reusable interfaces that are commonly used in many applications, specially hardware-in-the-loop simulations, and improve productivity by stopping ”reinventing the wheel” every time data width conversion is needed.

REFERENCES

[1] E. Jones and J. Sprinkle. autoVHDL: a domain-specific modeling language for the auto-generation of VHDL core wrappers. *Proceedings of the compilation of the co-located workshops on DSM'11, TMC'11, AGERE!'11, AOOPES'11, NEAT'11, VMIL'11*.

[2] AMBA AXI4 Interface Protocol. [Online]. Available: http://www.xilinx.com/ipcenter/axi4.htm.

[3] E. Stavinov. *100 Power Tips for FPGA Designers*. CreateSpace, 2011.