

Design and Implementation of a Duplex AMBA-TMS Transducer

Hansu Cho, Samar Abdi and Daniel Gajski
 Center for Embedded Computer Systems
 University of California, Irvine, CA - 92697
 {hscho,abdi,gajski}@cecs.uci.edu

October 10, 2005

Abstract

Communication between components, with different interface protocols, requires an extra component that must translate one protocol to another. This component is referred to as a transducer. In this paper we describe the design and implementation of a transducer between AMBA bus and TMS DSP bus. The transducer allows system designers to send data from AMBA compliant components to TMS compliant ones, and vice versa. The transducer was modeled in Verilog and implemented on Xilinx VirtexII FPGA board.

I. INTRODUCTION

With rising design complexity, designers are forced to implement more functionality in a shorter period of time by employing IP reuse. However, different IPs, typically, have different interface protocols and bus widths. Therefore, communication from one IP to another might not be possible using a common bus. This communication must be routed through an additional component known as a transducer.



Fig. 1. A transducer for incompatible PE1 and PE2

The transducer is a component with two interfaces as shown in Figure 1. We have two different Processing Elements (PEs), namely PE1 and PE2, using different protocols on Bus1 and Bus2, respectively. We also have a transducer component with two interfaces, one connected to Bus1 and the other to Bus2. Assume that PE1 wants to send data to PE2. Since PE2 does not support the protocol of Bus1, the communication must go through the transducer component. The data is first sent from PE1 to transducer using Bus1. The transducer locally buffers the data and then sends it to PE2 using Bus2. Thus the transaction is completed.

In this paper, we present the design and implementation of a transducer for a system consisting of an ARM1020T core [1] and a TMS320C50 DSP [2] core. The two processors have different bus interface protocols and

are running at different clock speeds. The rest of the paper is organized as follows. In Section II, we give the specification and design of the transducer. In Section III, we discuss the hardware implementation. Experimental setup and results are presented in Section IV.

II. SPECIFICATION AND DESIGN

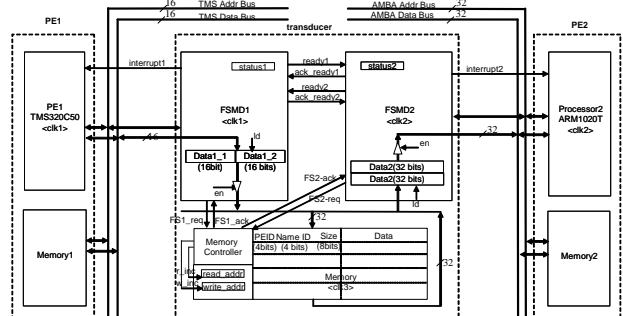


Fig. 2. Block diagram of the AMBA-TMS transducer

Figure 2 shows the architecture of the AMBA-TMS transducer. It is composed of three different subcomponents, namely FSMD1, FSMD2, and FIFO. Each subcomponent is modeled as a finite state machine with datapath (FSMD). FSMD1 implements the protocol of Bus1 (TMS) and enables PE1 (TMS) to read (write) to (from) the FIFO. FSMD1 runs at the same clock speed as PE1. FSMD2 implements the protocol of Bus2 (AMBA) and enables PE2 read (write) from (to) the FIFO. As in the previous case, the clock speed for FSMD2 is the same as that of PE2. The FIFO consists of a local memory and a controller that communicates with FSMD1 and FSMD2 using double handshake protocol. Mutually exclusive access to the FIFO is guaranteed by the handshake signals between FSMD1 and FSMD2. The asynchronous handshake between the FIFO and the two FSMDs allows different clock speeds for FSMD1, FSMD2 and the FIFO. As a result, the transducer is capable of interfacing PEs running at different clock speeds.

In order to send a message (size limited by FIFO capacity) from PE1 to PE2, FSMD1 and FSMD2 require the size and the source/destination ID, so that the transducer can identify the number of bus cycles needed for the

transaction and the right PE to notify about the transactions. Thus the message must be packetized as follows:

$|PE1id(1byte)|PE2id(1byte)|msgid(2bytes)|size(2bytes)|rawdata...|$

It is assumed that a new message is not initiated unless all the data from the previous message has been sent.

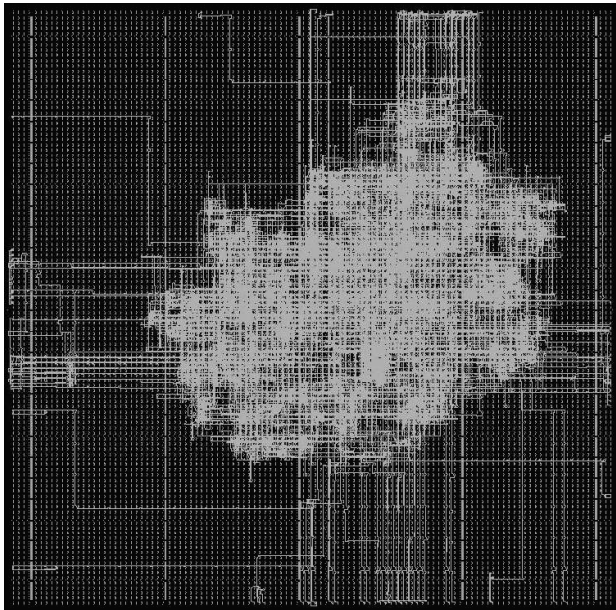


Fig. 3. FPGA implementation of the transducer

III. IMPLEMENTATION

The HDL implementation consisted of 3 top level modules, namely FSMD1, FSMD2 and the FIFO. At the RT level, FSMD1 required 10 states and FSMD2 require 15 states. The FIFO consisted of a controller with 10 states and a 1 kB RAM. The design was synthesized on the Xilinx VirtexII FPGA board. A built-in RAM on the FPGA was used to implement the memory in the FIFO. The FSMD2 block implements the interface to the AMBA bus. It can be seen that a register file (RF) is used to hold the data in transit to(from) the FIFO. This is because AMBA allows burst transfers of various sizes. The HSIZE/HTRANS signals of AMBA are used to select the addresses and number of registers in the RF that are used to hold data for a burst transfer. While sending data to ARM processor, this RF is always filled from the FIFO before initiating a transfer. Figure 3 shows the floorplan of the transducer on a Xilinx VirtexII DS031 [3] board.

IV. EXPERIMENTAL RESULTS

Table I shows the synthesis results for the transducer. It can be seen that the ARM processor is allowed to run at a maximum frequency of 183.5 MHz (1/FSMD1 clk), while the TMS processor is allowed to run at a maximum of 182.9 MHz (1/FSMD2 clk). The waveform showing the progress of a single word going through the transducer is

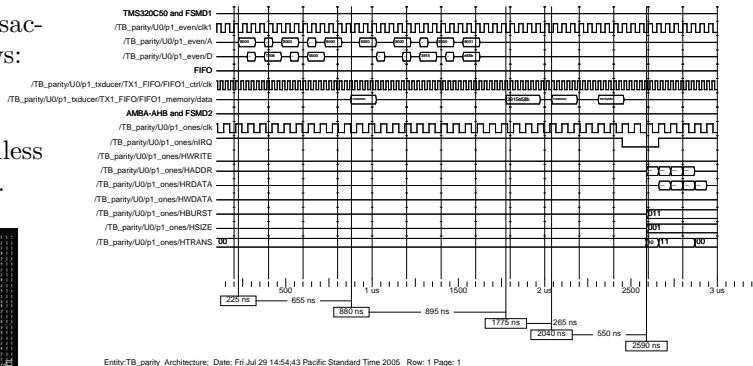


Fig. 4. Simulated waveform showing transaction progress

TABLE I
KEY IMPLEMENTATION METRICS

Feature	Value
V_{dd} core/IO	1.5 V / 3.3 V
No. of F/F (Area)	1561 slices
No. of 4-inp LUTs(Area)	856 slices
Min. clk period (FSMD1)	5.450 ns
Min. clk period (FSMD2)	5.468 ns
Min. clk period (FIFO)	7.322 ns

shown in Figure 4. Table II shows the extra number of cycles used to write data through the transducer, as compared to direct write to memory. The delays are shown in sender's cycles for three sets of clock speeds.

TABLE II
TRANSDUCER DELAY

Clk1	Clk2	Clk3	ARM→TMS	TMS→ARM
50ns	70ns	30ns	44	23
70ns	50ns	30ns	35	29
50ns	50ns	50ns	48	35

V. CONCLUSIONS

In this paper we presented the design and implementation of a useful communication element called the transducer that allows communication between AMBA compatible and TMS compatible cores. We observed that such “glue logic” can be automatically generated, given parameters such as message formatting, bus characteristics and protocol specifications. A transducer allows designers to put together different IPs with incompatible interfaces in the same system.

REFERENCES

- [1] ARM Inc. ARM1020T Manual. Available http://www.arm.com/pdfs/DDI0135A_1020T

[2] Texas Instruments Inc. TMS32C5x User's Guide.
Available <http://focus.ti.com/>.

[3] Xilinx Inc. DS031 - Virtex-II Platform
FPGAs: Complete Data Sheet. Available
<http://direct.xilinx.com/bvdocs/publications/ds031.pdf>.