Assignment 1

COMPSYS 701 – Advanced Digital Design

Department of Electrical and Computer Engineering
University of Auckland

Developing a simple Application Specific Processor (ASP) and its Network Interface (ANI) in SystemC

Due: Thursday 30 March 9:00 am

The purpose of this assignment is to develop a SystemC model for an Application Specific Processor (ASP) and its Network Interface (ANI) as explained later. The model will be used to verify functionality of the ASP/ANI before it is used as the starting point for synthesisable design on RTL level. The ASP is used as a hardware accelerator in the heterogeneous multi-processor system on chip (MPSoC) to accelerate some digital signal processing operations, where it is accessed by general purpose cores as the shared resource. The network interface (ANI) is an adaptor between the ASP and the Network on Chip (NoC) used as the system interconnection network.

The ASP performs some operations on elements of two vectors (one dimensional arrays) A and B. The number of elements for vectors A and B used in these operations is a parameter but it can be maximum 512 (indexed from 0 to 511) and each element is 16 bits . These vectors should be initialized properly before the start of the specific operation. The ASP has a 32-bit data input and a 32-bit data output, used to communicate with ANI, as well as some control signals that synchronise operations of writing and reading from the ASP. Data transferred to and from ASP use specific format (or packet format) to enable interpretation of different bits in ASP, or prepare data in ASP before transmitting to ANI, respectively. On the system level, general purpose processors such as Java processor (JOP) use the ASP as an accelerator by issuing commands to the ASP and receiving results from the ASP. Figure 1 shows examples of packet formats (for communication between a Java processor and ASP).

After **RESET** the ASP will automatically initialise all elements of both vectors A and B to zero.

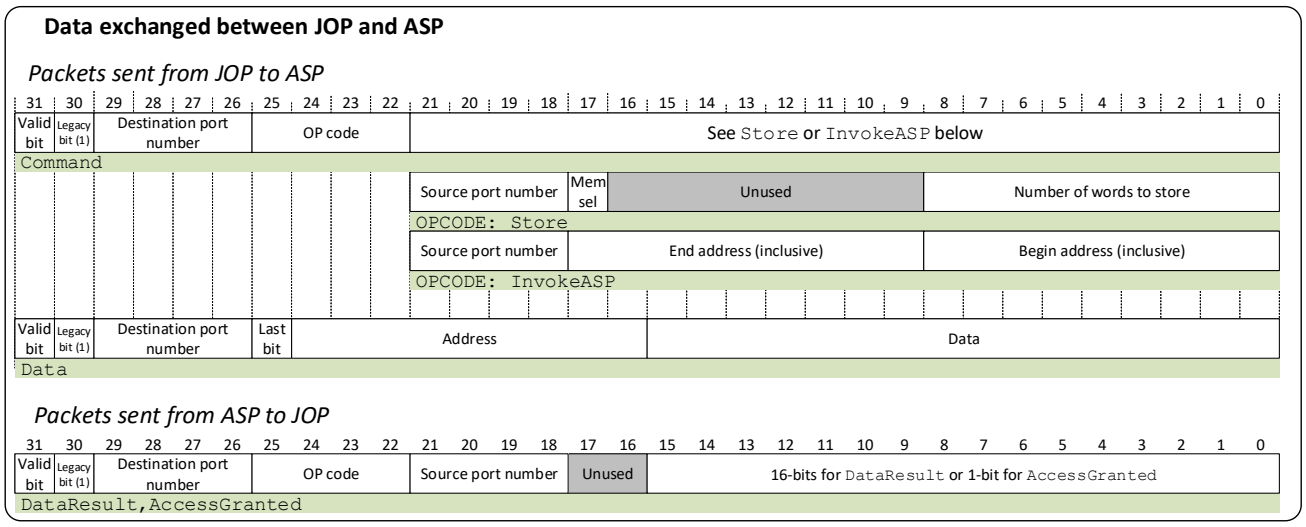

Figure 1: Examples of 32-bit Packet formats

The ASP receives on its input 32-bit packets, stores them into an input data register and interprets them according to the contents of individual fields. This interpretation may depend on a previous received packet (for example, in the case of STORE operation as explained below), which can be interpreted as either command or data. There is an output 32-bit data register which stores the packet to be sent by the ASP, which is the result or part of the results of ASP operation and possibly some additional status bits. Additional output signals such as ***busy*** may be included to indicate if the ASP is LOCKED to finish the current computations. The OPCODE field of the packet specifies which operation is to be performed by the ASP. Table 1 indicates the different types of OPCODES.

Table 1: ASP commands description (using the received packet format)

OPCODE	TYPE	Operation
0000	**STORE**	Initialize all elements of vector A (Mem sel bit 0) or vector B (Mem sel bit 1) to 0.
0001	**STORE**	**STORE:** Initialize vector A (Mem sel bit 0) or B (Mem sel bit 1) from index m to n (m is indicated through bits 0 to 8 and n is indicated through bits 9 to 17 in the ***InvokeASP*** packet which will be immediately after that). After this packet, data packets (using ***Data*** packet format) will be sent. Bits 0 to 15 indicate 16-bit data, bits 16 to 24 indicate the specific vector element to be initialized.
0010	**InvokeASP**	**XOR** bit-wise operation on elements of vector A from index m to n (m is indicated through bits 0 to 8 and n is indicated through bits 9 to 17 in the ***InvokeASP*** packet)
0011	**InvokeASP**	**XOR** bit-wise operation on elements of vector B from index m to n (m is indicated through bits 0 to 8 and n is indicated through bits 9 to 17 in the ***InvokeASP*** packet)
0100	**InvokeASP**	**MAC** (Multiply and accumulate) – for elements of vectors A and B from index m to n (m is indicated through bits 0 to 8 and n is indicated through bits 9 to 17 in the ***InvokeASP*** packet) $S(m,n) = \sum A(i)B(i), i = m, m+1, \dots, n$
0101	**InvokeASP**	**AVE** – (moving average filter on vector A and store the result back to the original location). Moving window size is programmable and can be L= 4 or L=8. For the boundaries of vector a solution has to be defined. A(i)=1/L(∑A(k)) , k=i,i+1,...,i+L-1
0110	**InvokeASP**	**AVE** – (moving average filter on vector B and store the result back to the original location). Moving window size is programmable and can be L= 4 or L=8. For the boundaries of vector a solution has to be defined. B(i)=1/L(∑B(k)) , k=i,i+1,...,i+L-1

For example, in order to initialize 8 elements of vector B in the ASP, the input packet (using *Command STORE* format) to be received should be as: **11** XXXX **0001** XXXX **1 0000 0000 000001000** followed by Data packets. (Note that bit 30 or legacy bit should always be '1').

The result of all operations should include all significant bits. The assumption is that all arithmetic is on unsigned (non-negative) numbers. The result is sent to the output 32-bit ***Data_out*** pins using the packet format (ASP to JOP) as shown in Figure 1. ***Res_ready*** indicates that a valid result is available (it may remain active longer if more than one packet needed for the result). ANI delivers a 32-bit input packet on ***Data_in*** pins of ASP. If ASP is already busy for a previous operation (i.e LOCKED) ***busy*** output pin is high, so ANI should wait until it is low to deliver the packet and sets ***valid*** pin high to indicate a valid packet is ready. ***D_to_NoC*** and ***D_from_NoC*** are 32-bit input and output pins respectively for ANI and are used based on the NoC packet format.

Description of ANI:

ANI is used to transfer the 32-bit output packet from ASP to TDMA-MIN network on chip. (Details about this NoC will be given separately). Figure 2 indicates an example of connecting ASP to NoC using the network interface (ANI).

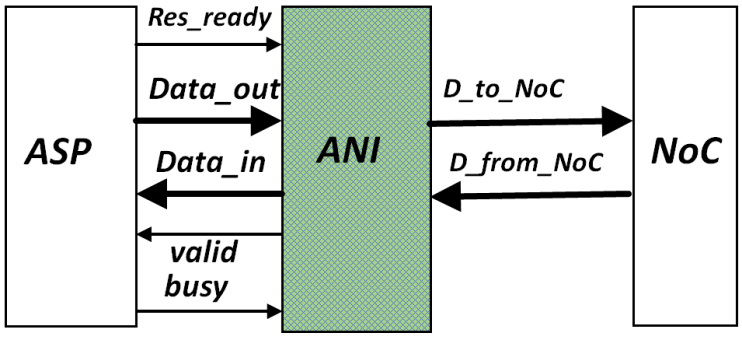

Figure 2: Input and output connections (ASP – ANI – NoC)

(***Note that additional signals may be added if necessary***)

Tasks:

1. Describe the ASP in SystemC to perform the specified operations.
2. Describe the network interface (ANI) in SystemC
3. Develop a proper testbench in SystemC to test ASP, ANI and their interaction using the specified packet formats.

A zip file, which includes all the design files and a short report should be submitted before the demo session in the lab.