User Logic, Custom IP and Test Bench: Accumulator with Block RAM

By Prawat Nagvajara

Synopsis
A note on a design of user logic as a custom intellectual property (IP) core – peripheral to FPGA embedded processor and its verification using C application.

Design Flow
The flow consists of design of user logic, custom IP and test bench where the IP is a peripheral of the processing system. The IP verification is done by a C test application running on the processing the system.

User Logic
Begin with an HDL design of the user logic – accumulator with block RAM (Fig. 1). The logic comprises a core generated by a block RAM generator and a bridge multiplexing the inputs (address, data and controls) from either the bus or the accumulator. The data out of the block RAM (mem) goes to the accumulator which is internal to the logic and, to the bus which is the logic primary output. The bus has the priority in accessing the block RAM via a control signal (bus2mem_en) at which the bridge busy output flag indicates the bus-to-memory access available to the accumulator.

Figure 2 and 3 summarize the parameters used for the block RAM. The read latency is 1 cycle when not using (checking) output registers.

Custom IP
The steps in Vivado IDE: Manage IP, new IP location, create new AXI peripheral, tools → create and pack ip → create new AXI peripheral with 5 registers and edit IP

Add Files in the order of the hierarchy of the user logic, top down
1. Add Source user_logic.vhd then
2. Add Source for the bridge component
3. Add Source for the block RAM, the .xci component
4. In IP Files Groups: Merge files, check Is Include, Right click on VHDL Synthesis
   Add sub-core reference.

Figure 4 and 5 show the design hierarchy and the Ip File Grops,

![Fig. 4 Design Sources](image)

![Fig. 5 IP File Groups](image)

**Test bench**

Create a project with processing system and the IP as its peripheral. The steps in Vivado IDE:

1. Block diagram: zynq processing system UART_1, clocks for Programmable Logic, reset, AXI master
2. Project Setting IP repository rep_ip
3. Place the IP and connection
4. Create wrapper and generate bit stream
5. Export Hardware and launch SDK
6. In SDK create a C application, connect the board, program FPGA and run application

**Results**

The application printed the input data written to address 0, ..., address 9 and the total sum at address 10 (Fig. 6 shows 2 runs).


**Conclusions**

This note covers an effective design flow of user logic. The flow consists of HDL design with IP cores and the hardware verification. The verification test bench is based on processor and the user logic as its peripheral. Test application (C code running on the processor) applies test vectors and reads the results. The design method offers a low-cost development platform for high complexity hardware.