SOFTWARE FOR REGISTERS STRUCTURE HDL-DESCRIPTION GENERATION WITH AMBA PROTOCOLS SUPPORT

Burko L., Kaiky M., Tushinskaya E.
Department of Informatics, Belarusian State University of Informatics and Radioelectronics
Minsk, Republic of Belarus
E-mail: {burkoliana, kaikymykhailo}@gmail.com, tushkan@fksis.by

This work discusses a software tool for automatically building and checking registry structures with programmatic access using AMBA protocols. A specialized tool has been developed for constructing an HDL description of register structures and a verification environment with the generation of test sequences for testing them.

Introduction

In the modern system-on-chip (SoC) design stack, software-accessible registers for accessing peripheral devices and hardware accelerators play an important role. Modern SoCs have dozens of hardware accelerators in their composition, which are accessed using various interfaces. An example of such interfaces is the AMBA (Advanced Microcontroller Bus Architecture) family of protocols proposed by ARM and including the following interfaces: AXI3, AXI4, AHB, APB, ASB. Despite this wide variety of interfaces, the model of register memory that is accessed is unchanged and is a set of numbered cells. Due to the necessity of designing various register models for hardware accelerators, an approach to automatically generate such structures was proposed. In addition to generating register structures, the issue of verifying the resulting HDL descriptions using the SystemVerilog is also considered.

I. Structure of Software-accessible Registers Hardware Implementation

In the general case, hardware implementations of software-accessible registers consist two fundamental components: an interface controller (for example, AXI4-Stream) and the memory itself. A memory block can be implemented in various ways, for example based on registers or SRAM cells. The implementation of a memory block depends on the requirements specified by the developers, such as read/write latency, the need to reset the value of cells after read/write operations, etc. In this paper, the authors propose a universal structure for describing an interface controller and a memory block for implementing a software-accessible unit. The proposed HDL-description consists of a reconfigurable interface controller, which allows, at the design stage, to select an implementable interface for communication with the host controller. The memory block is connected to the interface controller using a native interface (always remains unchanged), with the following signals: en, we, addr, wdata, rdata. Using this design approach, we can replace only the interface part without making significant adjustments to the memory structure.

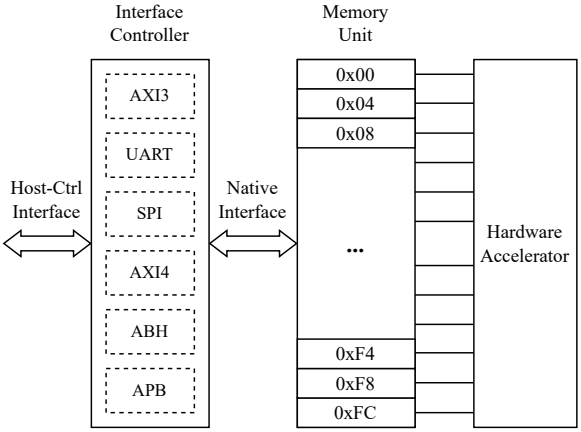

Figure 1 – Software-accessible Registers Structure

II. Generating Memory Structures

It was proposed to obtain an HDL description of memory structures using the developed generator. This generator operates based on a set of constraints for memory requirements, and the generator output is an HDL description in SystemVerilog for the memory unit. Constraints and requirements for a memory unit are specified in the following format:

```
‘define RST_TYPE sync
‘define RST_LEV 1
‘define MEM_TYPE DFF
‘define DFF_SYNC rising_edge
‘define MEM_HOST_WLATENCY 2
‘define MEM_HOST_RLATENCY 1
‘define MEM_HW_WLATENCY 1
‘define MEM_HW_RLATENCY 0
‘define status // 32 // W/R // R // 32’h12
```

When such a description is transmitted to the generator, a memory will be generated based on D-flip-flops featuring synchronous high-level reset, write/read latency for the host controller - 2/1 system clock cycles, for a hardware accelerator - 1/0, and will contain one A 32-bit register named status. The access mode for the host controller is read and write, the access mode for the hardware is read only, the value at initialization/reset is 12 in hexadecimal.

III. Verification of Software-accessible Registers Structure

To verify the generated structure, together with the interface controller, a series of test benches are recommended, simulating register sccess through a specified protocol (for example, AXI4-Lite or UART). For this purpose, “automatic tasks" were developed in the SystemVerilog language. Generation of addresses and data during read/write operations is carried out using root cases and the random function. Test sets are supplied to the block of software-accessible registers in such a way as to cover all possible modes of access to registers (read/write from the host controller and hardware accelerator) and cover all switching of the address decoder in the memory unit. After passing the verification stage, only when coverage of switching in the decoder and operating modes is achieved can a conclusion be made about the functional correctness of the generated structure.

An example of such a design for a write operation using the AXI4-Lite protocol (fig. 2):

```
task automatic axi_write;
  input [S_AXI_ADDR_WIDTH - 1 : 0] addr;
  input [S_AXI_DATA_WIDTH - 1 : 0] data;
  begin
    s_axi_wdata = data;
    s_axi_awaddr = addr;
    s_axi_awvalid = 1;
    s_axi_wvalid = 1;
    s_axi_wstrb = 4'b1111;
    wait(s_axi_awready && s_axi_wready);

    @(posedge s_axi_aclk) #1;
    s_axi_wstrb = 4'b0000;
    s_axi_awvalid = 0;
    s_axi_wvalid = 0;
  end
endtask
```

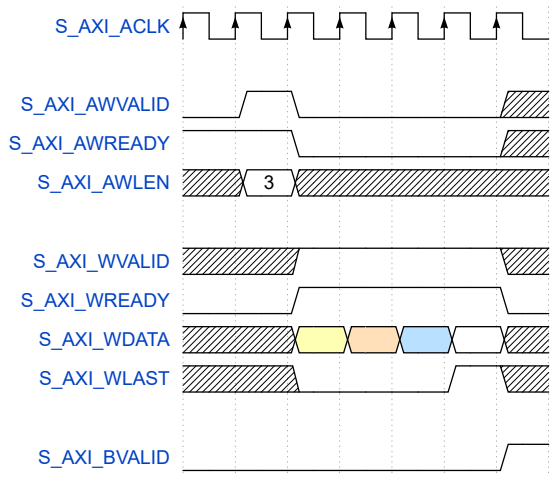

Figure 2 – AXI4-Lite Write Sequence

Read operation using the AXI4-Lite protocol (fig. 3):

```
task automatic enforce_axi_read;
  input [S_AXI_ADDR_WIDTH - 1 : 0] addr;
  input [S_AXI_DATA_WIDTH - 1 : 0] expected_data;
  begin
    s_axi_araddr = addr;
    s_axi_arvalid = 1;
    s_axi_rready = 1;
    wait(s_axi_arready);
    wait(s_axi_rvalid);
    if (s_axi_rdata != expected_data) begin
      $display("Error: Mismatch in AXI4
      read at %x:", addr,"expected %x,
      received %x",expected_data, s_axi_rdata);
    end
    @(posedge s_axi_aclk) #1;
    s_axi_arvalid = 0;
    s_axi_rready = 0;
  end
endtask
```

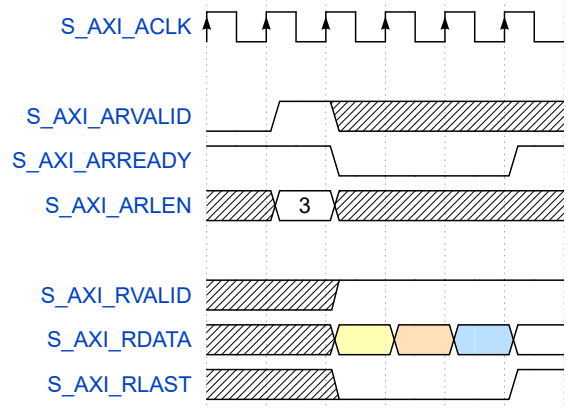

Figure 3 – AXI4-Lite Read Sequence

IV. Conclusion

A generator of HDL descriptions was developed for the structures of software-accessible registers with support for an arbitrary memory unit structure and various interface controllers (UART, SPI, AXI4-Lite). The generator receives as input a file with restrictions and parameters of the register memory model, and generates an HDL description of the implementation of a memory unit and a set of test sequences in the SystemVerilog.

1. Advanced Microcontroller Bus Architecture (AMBA), ARM Computing [Electronic resource] – Mode of access: `https://developer.arm.com/Architectures/AMBA`. – Date of access: 11.10.2023.
2. Design and Performance Analysis of 32 × 32 Memory Array SRAM for Low-Power Applications, Xue, Xingsi and Sai Kumar, Department of Electrical Engineering, Linköping University, 2022 [Electronic resource] – Mode of access: `https://www.researchgate.net/publication/368381578_Design_and_Performance_Analysis_of_32_32_Memory_Array_SRAM_for_Low-Power_Applications`. – Date of access: 15.10.2023.