FPGA Experiment 4 for SoC Students

Hao Yang

September 24, 2017

# Cortex M0+ Integration and Simulation

In this experiment, you will integrate a simple SoC system including a Cortex M0+ processor, a DW\_AHB bus,a UART(EX2) peripheral, a ROM and several RAM devices. The key point of this experiment is designing and constructing your SoC system, then making sure your processor can access each valid address range successfully.

**Recommended finish time: 3 weeks.**

## Tasks

* Getting familiar with the Cortex M0+ processor, its memory model and instruction-executing procedure, especially the *Cortex™-M0+ Integration and Implementation Manual* in the *./M0+/AT590-BU-60000-r0p1-00rel0/doc* directory.
* Fully understand the AMBA 3 AHB-Lite Protocol. For reference, please refer to the *AMBA® 3 AHB-Lite Protocol Specification* in the *./material* directory.
* Use the cm0p integration kit as your template, modify the rtl files according to your needs.
* Use the CM0PINTEGRATION level abstraction in your design, and configure all the parameters correctly.
* You need to design the clk generator and reset controller by yourself. Do not use the cm0p\_ik\_clk\_gen.v and cm0p\_ik\_rst\_ctl.v provided by the IK. Generate synchronous clocks by frequency division, and generate reset signals based on “asynchronous reset synchronously release”. When synthesizing, your clk generator and reset controller should be designed in gate level, that is using the standard cells provided by specific technic. Consider this problem in the rtl design process. For standard cell primitives, please refer to the */home/techLib/UMC55/library\_for\_terran/G-9LT-LOGIC\_MIXED\_MODE55N-SP\_LOW\_K\_UM055LSCSPMVBPL-LIBRARY/UM055LSCSPMVBPL\_B01\_TAPEOUTKIT/Verilog* directory on the #204 server.
* After integration, you need to write a test program to verify the functionality of your processor. You can write your own test program by IAR or Keil based on the IAR project in the fpgaEx4\material\M0test directory. Make sure your program is successfully compiled and the bin file can be generated which is going to be loaded into your ROM in system simulation. Your test should test the AHB read and write process to each valid address range at least once.
* For system compilation and simulation you can refer to the scripts provided in the *./material/scripts* directory. But you need to make appropriate modification.

## Notice

* Organize your project directory structure. For example:

./m0\_integration/

----rtl/

--------cmop\_mcu/

--------dw\_ahb/

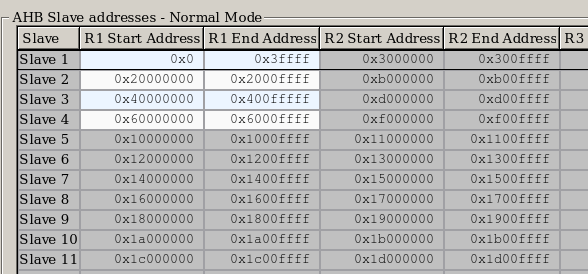
----scripts/

----sim/

----tb/

* Give meaningful names to your signals, like “sys\_haddr\_cortexm0plus”, “sys\_hrdata\_rom”, “ahbout\_hwdata”, etc.
* Had better backup your project files occasionally.

## Memory Map



Please modify AHB slave addresses by coreConsultant in order to suit your system and get memory map.

## Basic Simulation Procedure

VCS compile and generate simv file.

Run the simv file with your program bin file and generate fsdb file.

View the waveform with Veridi with the fsdb file in last step.

## Reference

**“asynchronous reset synchronously release” example**

module reset\_gen

#(

parameter RST\_OUT\_VLD\_HIGH = 1'b0

)

(

input clk\_i,

input areset\_i,

output reg reset\_o

);

reg [3:0] vector;

always @ ( posedge clk\_i or posedge areset\_i )

begin

if( areset\_i )

begin

vector <= { 4{RST\_OUT\_VLD\_HIGH} };

end

else

begin

vector <= { vector[2:0], ~RST\_OUT\_VLD\_HIGH };

end

end

assign reset\_o = vector[3];

endmodule