Digital Design Verification

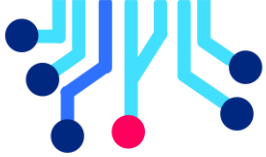
Weekly Task

RISC-V – UART Integration and Digital Clock Testing

Release: 1.0

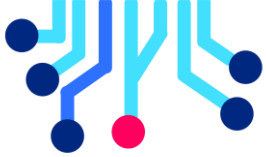
Date: 14-June-2024

NUST Chip Design Centre (NCDC), Islamabad, Pakistan



Revision History

Revision Number	Revision Date	Revision By	Nature of Revision	Approved By
1.0	14/06/2024	Ali Aqdas	Complete manual	-



Contents

Contents 2
Objective 3
Instructions 3
Introduction to System on Chip Design 4
Task 1: Basic Implementation 4
Task 2: Digital Clock in Simulation Terminal 6

Objective

The objective of this lab is to:

- Understand System-on-Chip Integration
- Integrate UART with your Microprocessor

Tools

- Cadence Xcelium

Instructions

The students are required to design and submit the updated files for evaluation of this lab. Additional files for top-level of the SoC and UART transmission and receiver are to be added to the submission.

./<student-name>/	./rtl/	program_counter.sv inst_mem.sv data_mem.sv reg_file.sv imm_gen.sv alu_logic.sv branch_comp.sv control_unit.sv top.sv uart_tx.sv uart_rx.sv soc_top.sv
	./tb/	program_counter_tb.sv inst_mem_tb.sv data_mem_tb.sv reg_file_tb.sv imm_gen_tb.sv alu_logic_tb.sv branch_comp_tb.sv control_unit_tb.sv top_tb.sv soc_top_tb.sv

Introduction to System on Chip Design

Historically, computing systems comprised individual integrated circuits. With shrinking technology nodes, the idea of integrating all components needed for functionality of a computing system on a single silicon die became a reality. Today, system on chip designs may include, multiple processors, specialized accelerators, memory, external interfaces etc.

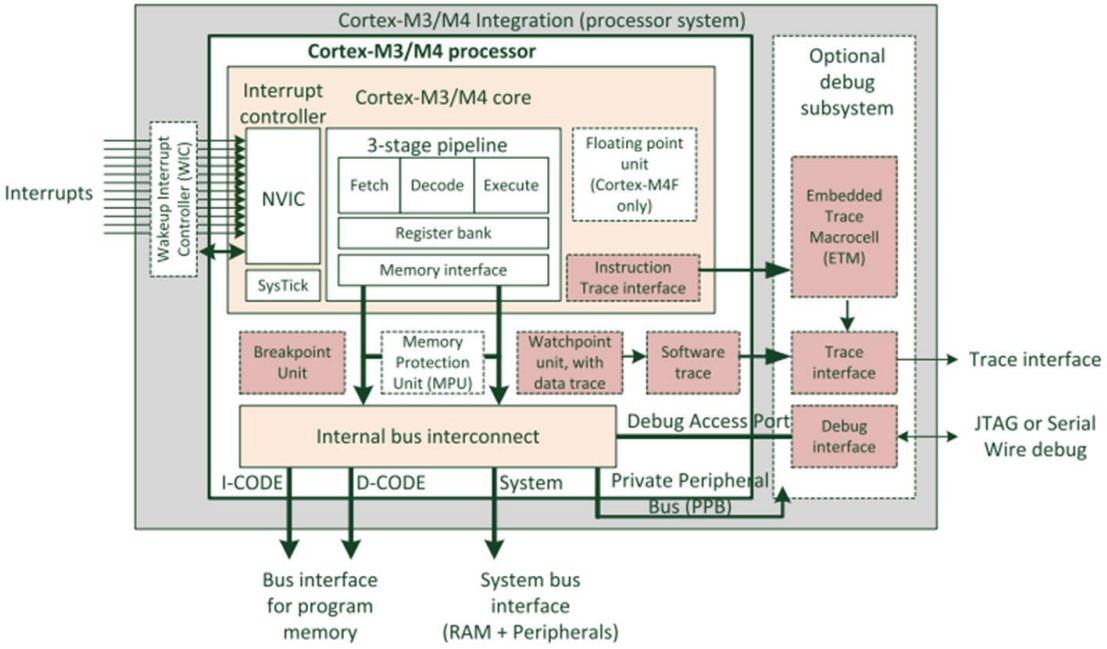


Figure 1: (The Definitive Guide to Cortex M3 and M4 Processor, Yui. J, Figure 3.3)

While this is a complex design, and you may not understand many components of the block diagram, it is a reference to industry standard designs.

Task 1: Basic Implementation

Now, how does that fit in our circumstances is a burning question. For our implementation, a memory mapping unit, which is simplified version of interconnect, would partition the address space (0x00000000-0xFFFFFFFF) into two parts and route the associated signals to the desired paths.

A block diagram of different components is presented in Figure 2.

1. **Core:** The main core includes the data and control path, excluding the memories.
2. **Instruction/Data Memory:** As described in the lab manuals, they are used for instructions and data.
3. **Memory Mapping Unit:** Partitions the address space and routes the data to desired paths
4. **UART-TX:** Transmitter Module for UART (includes memory mapped configuration and data registers.

It includes two buses, an instruction bus and a data bus, each comprising of multiple signals. **In individual cases, some of the signals may be different from the ones mentioned in the table and may include other signals as well.**

Buses	Signals	Signal Widths	Signal Direction*

Instruction Bus	Instruction Address Instruction Data	32-Bit 32-Bit	Output Input
Data Bus	Data Address Write Enable Byte Enable	32-Bit 32-Bit Variable Variable	Bidirectional Output Output Output

Table 1: Instruction Bus Description

**Signal direction is from the perspective of the core*

At any instant, the memory mapping unit may access either the UART module or Data Memory. You may partition the address space as per your needs. In software, the address space division is reflected in accessing the data from different components.

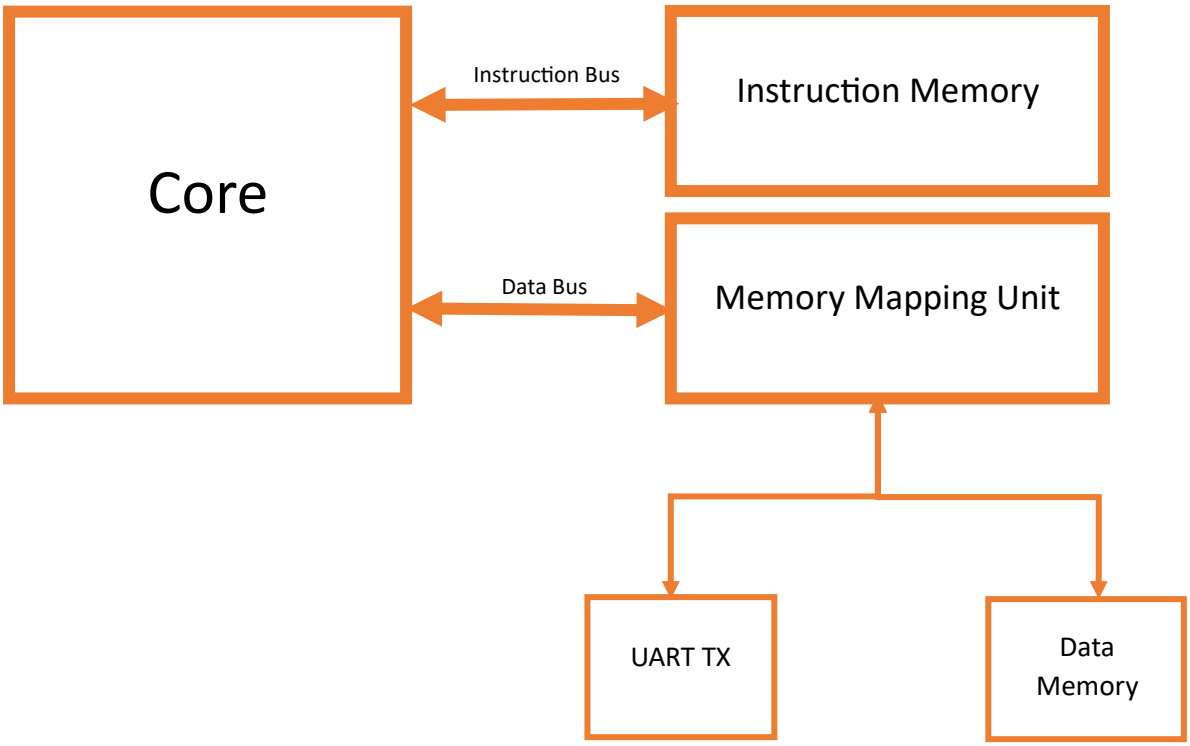


Figure 2: System on Chip Integration of Core

Now that integration has been completed, you are to design a test bench, which would include the system on chip top module and a UART receiver module. In the receiver module, you are required to display the received output in the simulation terminal.

A rough estimate of expected design is shown in Figure 3.

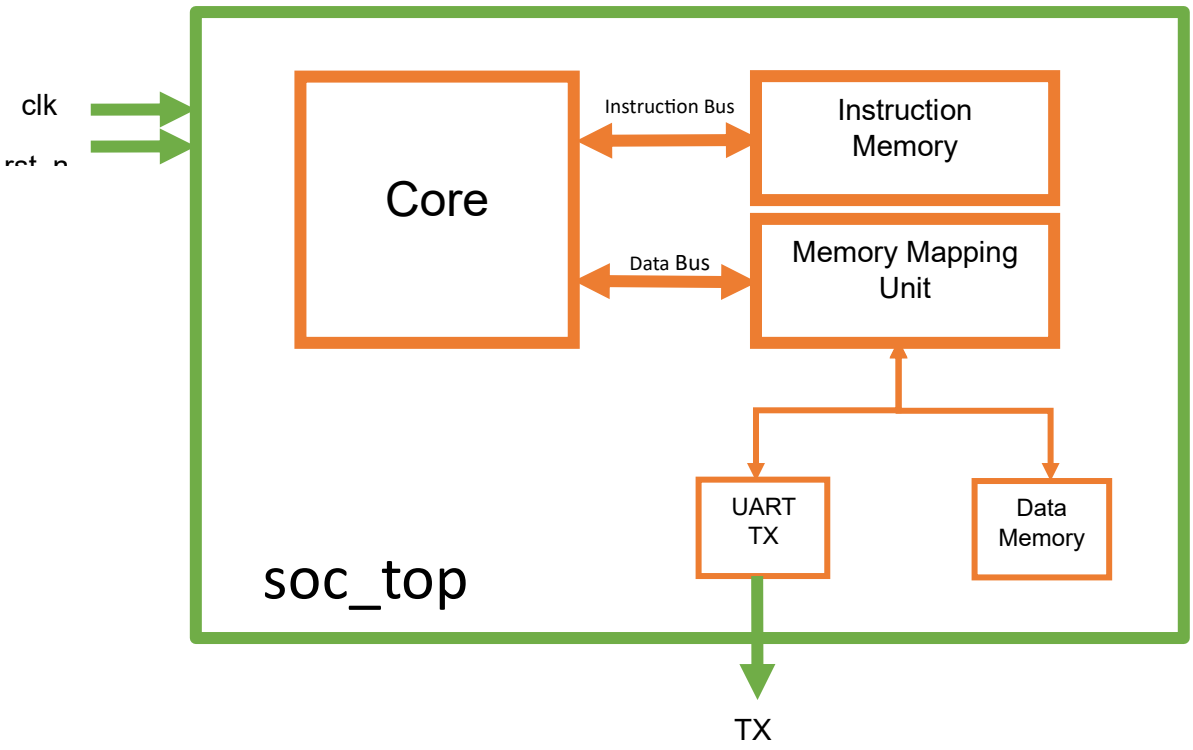


Figure 3: Top Level Wrapper for System-on-Chip

Task 2: Digital Clock in Simulation Terminal

Now that your processor can display data in the simulation terminal, modify the digital clock from previous weekly task to display on simulation terminal instead of seven-segment display.

Note that we do not have a timer module, hence we may use an empty loop to approximate the delays we are using from timer. You may use venus to dump hexadecimal data for instructions and data (if any) and load the respective in your instruction and data memory at the beginning of simulation in an **initial block.**

"If you've made it this far, Congratulations! You have seamlessly bridged the gap between hardware and software."