Week 2 Task: BabySoC Fundamentals & Functional Modelling

Part 2: Labs – Hands-on Functional Modelling

This section focuses on the practical application of SoC concepts by building and simulating a functional model of the BabySoC.

1. Objective & Toolchain The goal is to verify the logical behavior of the BabySoC design before proceeding to detailed RTL design and physical implementation. - Icarus Verilog (iverilog): An open-source Verilog simulation and synthesis tool. It acts as our compiler, translating the high-level Verilog code into a simulation executable. - GTKWave: An open-source waveform viewer. It is our debugging microscope, allowing us to visualize the signals, track data flow, and verify the timing behavior of our simulated design.

2. Lab Workflow Summary The process follows a standard digital design simulation flow: - Clone the Repository: Obtain the BabySoC source code. - Compile (iverilog): All Verilog modules (*.v files) are compiled into a single executable simulation file. - Simulate (Run): The executable is run, which executes the testbench, stimulates the design, and generates a Value Change Dump (.vcd) file containing the waveform data. - Analyze (GTKWave): The .vcd file is opened in GTKWave to inspect signals and verify correct operation.

Deliverables & Observations

Simulation Logs The simulation log, typically output to the console, is the first indicator of success. A clean log without errors or unexpected warnings confirms that the design compiled and simulated without fundamental issues. - What to look for: A log that ends with a $finish statement or a success message from the testbench, indicating that the simulation completed its intended sequence.

GTKWave Waveform Analysis The waveforms provide visual proof of the BabySoC's functionality. Below are key scenarios to analyze.

Observation 1: Reset Operation - What the Waveform Represents: This screenshot captures the initial state of the system when the reset signal (rst) is asserted. - Explanation: - When rst is high (1), the system is in a forced known state. - The Program Counter (pc) is reset to 0x00000000, indicating the CPU is ready to fetch the first instruction from the beginning of memory. - Critical control signals and register outputs are forced to their default (idle) values. - Once rst is de-asserted (low (0)), the system begins normal operation, and the pc starts incrementing. This validates that the design initializes correctly.

Observation 2: Clocking and Core Operation - What the Waveform Represents: This shows the steady-state operation of the BabySoC, driven by its primary clock. - Explanation: - The Clock (clk) is a regular, toggling signal. All synchronous operations happen on the clock edges (e.g., rising edge). - The Program Counter (pc) increments on

clock cycles, showing the CPU is moving through its program. - The Instruction (instr) bus shows the actual opcode being read from memory at the address specified by the pc. - This demonstrates the fundamental Fetch (get instr) -> Execute (run opcode) cycle of the CPU core.

Observation 3: Dataflow between Modules (e.g., Memory Write) - What the Waveform Represents: This screenshot captures a key moment of interaction, such as the CPU writing data to a peripheral or memory-mapped location. - Explanation: - The CPU places a specific Address (addr) on the bus, targeting a peripheral (e.g., the UART's transmit register). - It asserts a Write Enable (we) signal. - It places the Data to be written (wd) on the bus (e.g., the ASCII value for a character 'A' or 0x41). - The corresponding peripheral (like uart_tx) would then become active and process this data. This trace visually confirms the successful initiation of a write operation across the SoC's interconnect.

Conclusion By the end of Week 2, the theory and practice converge. - Theory: We understand that an SoC is an integrated system with a CPU, memory, peripherals, and an interconnect. - Practice: We have successfully used Icarus Verilog and GTKWave to build a functional model of the BabySoC. We have observed and documented its reset sequence, clock-driven operation, and internal dataflow, proving that the fundamental design is sound and ready for the next stages of the SoC design journey.