Week 2 Task: SoC Fundamentals & The BabySoC

Part 1: Theory & Conceptual Understanding

1. What is a System-on-Chip (SoC)? An SoC, or System-on-Chip, is an integrated circuit that consolidates all or most components of a complete electronic system onto a single piece of silicon.

The "System" Part: Instead of having separate chips for a processor, memory, and controllers on a circuit board, an SoC integrates them all into one. The "Chip" Part: This integration leads to a single, compact chip that is more power-efficient, faster (due to shorter internal distances), and cheaper to produce in volume. Analogy: Think of a modern smartphone. Its brain isn't just a CPU; it's an SoC that includes the CPU, GPU, modem, memory, and other controllers all in one tiny package, enabling powerful, portable, and battery-efficient devices.

2. Key Components of a Typical SoC A typical SoC is a complex ecosystem of interconnected components. The main building blocks are: - CPU (Central Processing Unit): The "brain" of the system. It executes software instructions and manages the other components. - Memory: Includes both on-chip memory (like SRAM caches for fast access) and memory controllers to interface with external, larger memory (like DRAM). - Peripherals: These are the functional blocks that interact with the outside world. - UART: For serial communication (e.g., console output). - SPI/I2C: For communicating with sensors and other simple chips. - GPIO: General-purpose pins that can be programmed as input or output. - Interconnect (Bus Fabric): This is the "nervous system" or "highway" of the SoC. It is a network of wires and switches that allows all the components (CPU, memory, peripherals) to communicate with each other according to a specific protocol (e.g., Wishbone, AXI, AHB).

3. Why BabySoC? A Simplified Learning Model The "BabySoC" is a deliberately simplified model designed to teach the fundamental concepts of SoC design without the overwhelming complexity of a commercial SoC. - Focus on Fundamentals: It strips away advanced features like multi-core processors, complex cache hierarchies, and high-speed interconnects. - Core Components Intact: A typical BabySoC might contain a simple CPU core (like a RISC-V), some basic memory (ROM/RAM), and a few standard peripherals (like a UART and a Timer), all connected via a simple bus (e.g., Wishbone). - Manageable Learning Curve: By working with a BabySoC, we can understand the entire flow—from functional modelling and RTL design to verification and synthesis—in a manageable and comprehensible way. It's the "Hello, World!" of the SoC world.

4. The Role of Functional Modelling Before writing a single line of detailed hardware description language (HDL) code for the RTL stage, we create a Functional Model. What is it? A high-level, software-based simulation of the intended SoC. It models the behavior of the system, but not its precise timing or hardware structure. Purpose: - Architectural Validation: It answers the question, "Does my system design work correctly?" We can run

software programs on the model to see if the CPU, memory, and peripherals interact as expected. - Early Software Development: Software and firmware can be developed and tested on the functional model long before the actual silicon is available. This is known as "shift-left" in the development cycle. - Creating a "Golden Reference": The functional model serves as a reference for the expected output of the system. The subsequent, more detailed RTL design can be checked against this model to ensure correctness. Tools Used: In this task, we use Icarus Verilog (for simulation) and GTKWave (for viewing timing diagrams and debugging) to build and analyze our functional model of the BabySoC.