Week 2 – Introduction to BabySoC Concepts and Functional Modelling

Learning Objective

This task focuses on strengthening your foundational understanding of System-on-Chip (SoC) architecture and practicing functional modeling using simulation tools such as Icarus Verilog and GTKWave. By the end of this exercise, you should be able to describe the key elements of an SoC and understand how BabySoC simplifies complex SoC design principles for educational purposes.

1. Understanding System-on-Chip (SoC)

A System-on-Chip is an integrated circuit that consolidates multiple components of a computer system into a single silicon chip. It typically includes a central processing unit (CPU), memory modules, communication interfaces, and peripheral units. This integration allows for compact designs, reduced power consumption, and improved performance compared to traditional multi-chip systems.

In modern electronics, SoCs power everything from smartphones to embedded controllers. Their architecture is designed to balance processing efficiency, cost, and scalability. Learning the structure of an SoC forms the foundation for advanced hardware and digital system design.

2. The BabySoC Model

BabySoC serves as a simplified representation of a real SoC, designed specifically for learning and experimentation. It models the essential features of a complete system—CPU, memory, and peripherals—but in a minimal, easy-to-understand format. This approach allows learners to grasp fundamental design concepts without the complexity of industrial-scale architectures.

Through BabySoC, students gain hands-on experience with design hierarchy, data flow, and module interconnection. This prepares them to transition smoothly from conceptual understanding to more advanced stages like Register-Transfer Level (RTL) design and physical implementation.

3. Purpose of Functional Modelling

Functional modelling provides a high-level simulation of how an SoC operates before proceeding to detailed hardware design. It enables designers to verify logic correctness, test interactions between modules, and evaluate system performance in a virtual environment. Tools like Icarus Verilog and GTKWave help visualize and analyze simulation results, making them essential for early design validation.

Task Deliverable

Prepare a brief 1–2 page summary on your GitHub repository explaining your understanding of SoC fundamentals and how BabySoC serves as a learning model for

system integration and functional simulation. Your summary should reflect your personal insights from exploring SoC concepts and using simulation tools during this exercise.

— End of Document —

Week 2 – Practical Exercise: BabySoC Functional Simulation

Objective

The goal of this exercise is to apply your understanding of System-on-Chip (SoC) concepts by performing hands-on functional modelling using the BabySoC framework. You will utilize Icarus Verilog and GTKWave to simulate, observe, and analyze system behavior, gaining experience with real-world digital design workflows.

1. Overview of the Practical Task

This session emphasizes the practical application of SoC design theory through the BabySoC project. By simulating Verilog-based modules, learners can visualize data transfer, clocking operations, and reset mechanisms across the SoC components. The workflow mirrors professional digital design practices, offering insight into the functional verification stage of system development.

2. Tools and Setup

Before starting, ensure that the following software tools are installed and functional:

• Icarus Verilog – a compiler used to build and simulate Verilog source files.
• GTKWave – a waveform viewer for analyzing and visualizing simulation output files (.vcd).

3. Experimental Procedure

Follow these steps to complete the BabySoC functional modelling task:

1. Clone the official BabySoC project repository.
2. Use Icarus Verilog to compile the BabySoC module files and generate the simulation output.
3. Run the simulation to produce Value Change Dump (.vcd) waveform files.
4. Open the .vcd files in GTKWave to visualize and analyze SoC behavior.
5. Focus on the following aspects during analysis:
• System reset behavior
• Clock signal transitions
• Data transfer between SoC components
6. Capture relevant screenshots and record observations for each waveform segment.

4. Required Submissions

At the end of this exercise, compile your work and upload it to your GitHub repository. Include the following items:

• Simulation output logs
• Screenshots of GTKWave displaying expected BabySoC behavior
• Brief written notes explaining the functionality shown in each captured waveform

5. Learning Outcome

By completing this activity, you will gain the ability to correlate theoretical SoC knowledge with its simulated behavior. You should be able to demonstrate an understanding of how BabySoC modules interact, interpret timing waveforms, and validate correct system operation through functional simulation.

— End of Document —