What is a System on Chip

A System on Chip is a single integrated circuit that brings together a full computing system on one piece of silicon. Typical contents include one or more processor cores, a memory hierarchy, I/O interfaces, interconnect, clocking, and often mixed signal blocks such as ADCs and DACs.

Key components of an SoC

- CPU and accelerators: general purpose cores, DSPs, or application specific engines that execute software and control the system.
- Memory: on chip SRAM for caches and scratchpads plus external DRAM and non volatile memory such as flash or ROM.
- Peripherals and I/O: GPIO, timers, UART, SPI, I2C, USB, PCIe, and display or camera interfaces for connection to the outside world.
- Interconnect fabric: shared buses or networks on chip that move data among CPUs, memories, and peripherals while managing coherency, bandwidth, and latency.
- Clocking and mixed signal: PLLs for clean clocks, power management blocks, ADCs, and DACs that let the digital system sense and drive the analog world.

Why BabySoC is a good learning platform

VSDBabySoC keeps the architecture small and clear while still demonstrating real SoC concerns. It integrates:

- RVMYTH, a simple RISC V CPU used for instruction execution and register level control.
- An 8x PLL that multiplies and stabilizes a reference clock for synchronous operation.
- A 10 bit DAC that converts digital codes into an analog output signal.

This scope allows quick simulation and easy reasoning while covering digital control, clock generation, and conversion to analog.

Role of functional modelling

Functional modelling checks intended behavior before committing to detailed RTL or physical design. It focuses on algorithms, protocols, and value ranges, which makes it ideal for catching

specification gaps, wrong assumptions, or corner cases early. Later stages such as RTL signoff and physical implementation concentrate on synthesizability, timing closure, power, and layout signoff. Discovering logic intent issues during those stages can be expensive, so early functional modelling saves time and effort.

BabySoC dataflow in brief

After reset, the PLL locks and provides a stable clock to the whole system. The RVMYTH core updates a working register that holds the next digital value for the DAC. The 10 bit DAC converts these codes into a continuous analog voltage that appears on the OUT signal, which demonstrates a complete path from software visible state to an analog waveform.

Simulation workflow for this task

Use Icarus Verilog to run behavioral testbenches for BabySoC and generate VCD or FST traces. Inspect signals in GTKWave to confirm reset timing, PLL activity, register update cadence, and DAC input or output behavior. Change data formats to hex, decimal, or analog step when needed to make waveforms readable, and save the view so that formatting persists across sessions.

What this exercise teaches

- How SoC building blocks fit together and why integration on one die improves power, performance, and form factor.
- How clean clocking from a PLL enables reliable synchronous operation.
- How a digital to analog boundary works through a DAC and how numeric scaling affects analog results.
- How to plan and read simulations that validate system intent well before synthesis.

Functional vs RTL vs Physical comparison

Stage	Goal	Model or artifacts	Primary checks	Typical tools
Functional modelling	Validate behavioral intent early before committing to RTL and implementation	Behavioral or transactional models and simple testbenches with VCD or FST for inspection	Algorithm correctness, interface protocol behavior, value ranges, and control sequencing under scenarios	Fast simulators and waveform viewers that generate or consume VCD or FST traces
RTL signoff	Ensure RTL is structurally sound and ready for synthesis and integration	Synthesizable Verilog or VHDL with lint, CDC, and low power intent artifacts	Lint, CDC, assertion coverage, functional coverage, and readiness for implementation	RTL simulators, static analyzers, and coverage tools
Physical design	Implement the netlist and close timing, area, and power for tapeout readiness	Gate level netlists, constraints, libraries, parasitics, and layout data	Static timing, IR or EM, DRC, and LVS signoff checks	Logic synthesis, place and route, STA, and signoff verification tools

Conclusion

BabySoC offers a compact yet complete slice of real SoC work. By simulating the CPU to DAC path under a PLL driven clock, this task builds practical skills in reading waveforms, confirming behavior, and documenting findings, which sets up later weeks for RTL and physical design progress.