Problem Statement

This project delves into designing a compact, open-source **System on Chip (SoC)** based on **RVMYTH**, a RISC-V-based processor core. The SoC integrates a **Phase-Locked Loop (PLL)** for precise clock generation and control, alongside a **10-bit Digital-to-Analog Converter (DAC)** for interfacing with external analog systems. By converting digital signals into analog, this DAC allows BabySoC to communicate with devices that accept analog inputs, such as televisions and mobile phones, enabling output in the form of audio or video. Ultimately, this Sky130-technology-based SoC aims to provide a highly documented, educational platform for learning and experimentation in digital-analog interfacing.

1. Understanding System on a Chip (SoC)

A **System on a Chip (SoC)** is like a mini-computer built on a single chip. Instead of needing separate parts for each function, an SoC combines everything into one small package. This makes it especially useful for devices where space, power, and efficiency are important, like smartphones, smartwatches, and tablets. Let's break down what an SoC includes and why it's essential:

Key Parts of an SoC

1. **CPU (Central Processing Unit)**:
    - The brain of the SoC, handling all main instructions and decisions.
    - Manages tasks like calculations, data processing, and running applications.
2. **Memory**:
    - **RAM** (Random Access Memory) for temporarily storing data as you use the device.
    - **ROM** or **Flash Storage** for keeping information saved even when the device is off.
3. **I/O Ports (Input/Output)**:
    - Connects the SoC to other parts or devices, like a camera, USB, or even your headphones.
    - These ports let the SoC send and receive data externally.
4. **Graphics Processing Unit (GPU)**:
    - Responsible for creating visuals on your screen.

    - Used for gaming, watching videos, or any activity involving images or animations.

5. **Digital Signal Processor (DSP)**:
    - Specialized in processing audio and video signals.
    - Helps with tasks like noise reduction in phone calls or enhancing video quality.
6. **Power Management**:
    - Regulates power usage within the SoC, making sure the chip operates efficiently.
    - This is crucial for extending battery life in portable devices.
7. **Special Features**:
    - Additional features may include Wi-Fi, Bluetooth, and even security modules for safe data handling.
    - These features vary depending on the specific purpose of the SoC.

Why SoCs Are Awesome

- **Space Saving**: By combining everything into one chip, SoCs help make devices smaller and more portable.
- **Energy Efficient**: Because all the parts are so close together, they use less power, which is especially important for battery-operated devices.
- **High Performance**: Since data doesn't have to travel far, SoCs can process information faster.
- **Cost Effective**: Building a single chip is often cheaper than using multiple parts, reducing the cost for manufacturers and, ultimately, for consumers.
- **Reliable**: Fewer parts mean fewer points of failure, making devices with SoCs generally more dependable.

Where You'll Find SoCs

- **Smartphones & Tablets**: Almost all modern mobile devices use SoCs because of their compact size and efficiency.
- **Wearables**: Devices like smartwatches rely on SoCs for their small size and low power use.
- **IoT Gadgets**: Internet of Things devices, like smart home sensors, often use SoCs to handle tasks like monitoring and connecting to Wi-Fi.

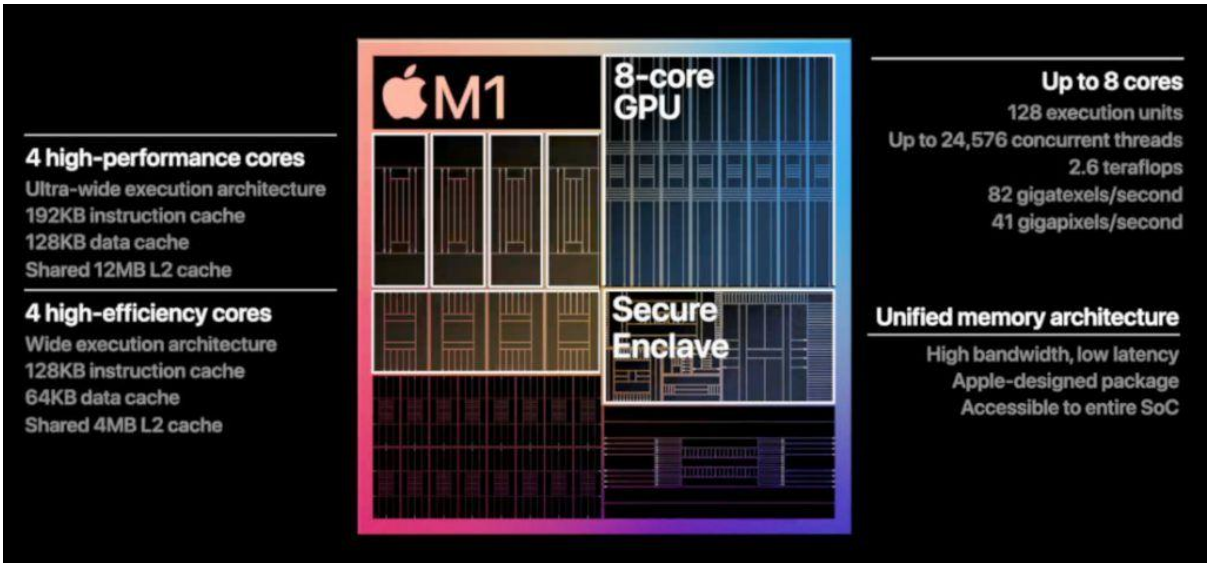
- **Cars, TVs, and More**: Embedded systems in cars, TVs, and appliances may also use SoCs to manage their internal functions.

Some Popular SoCs You Might Know

- **Apple A-Series**: Powers iPhones and iPads.
- **Qualcomm Snapdragon**: Found in many Android phones.
- **Samsung Exynos**: Built for Samsung devices.
- **NVIDIA Tegra**: Powers devices like the Nintendo Switch.

Challenges with SoCs

- **Complex Design**: Creating an SoC is complicated. Combining multiple functions in one small space requires advanced design skills.
- **Heat Issues**: Packing many components together can lead to overheating. SoCs need cooling solutions to work well over time.
- **Less Flexibility**: Once an SoC is designed, it's hard to change. This is because each SoC is built for specific tasks or devices.

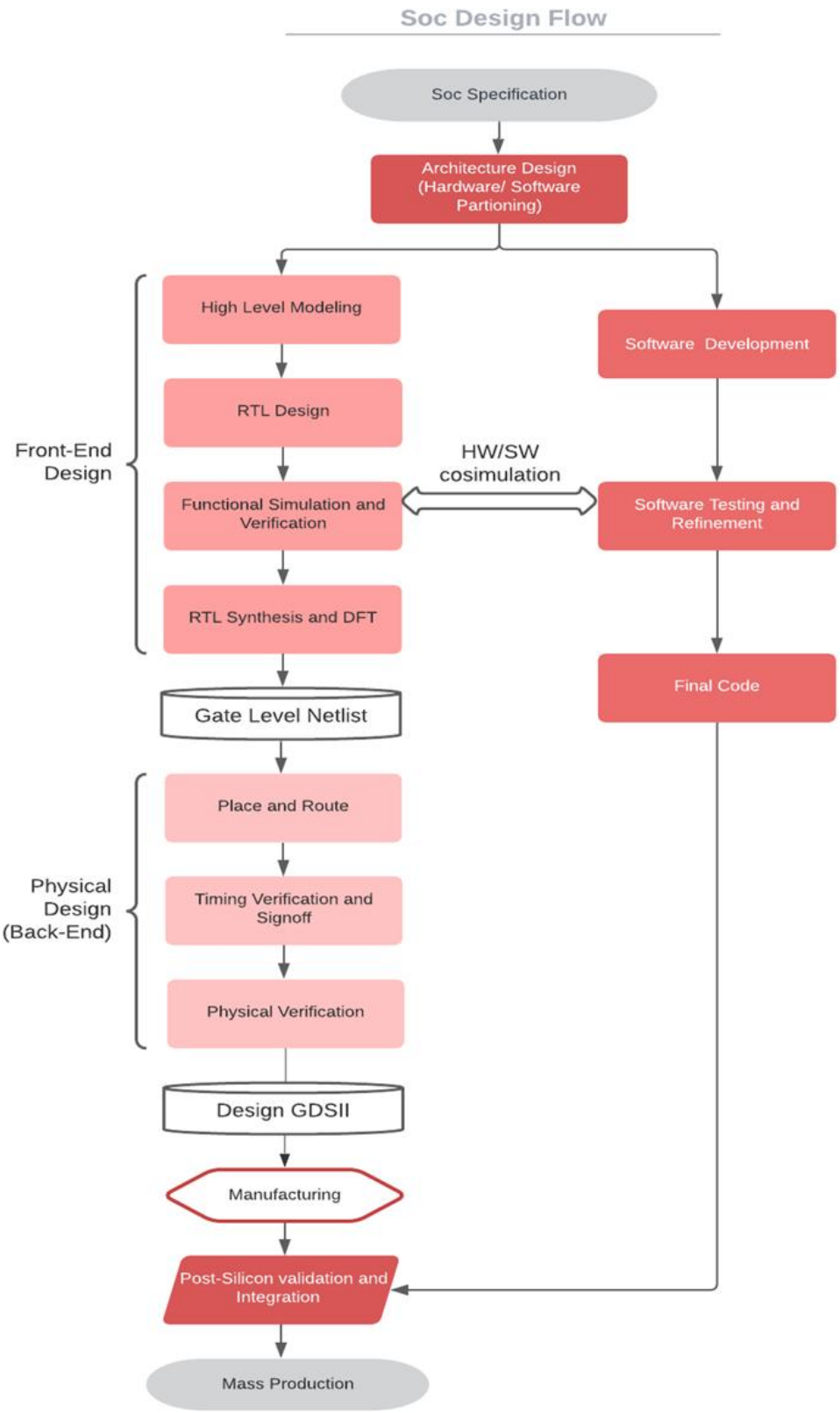


In summary, **System on a Chip (SoC)** technology allows us to create powerful, efficient, and compact devices by combining multiple components into one chip. This is why our phone, smartwatch, and even some household appliances can do so much in such a small package.

2. Types of SoCs

- **Microcontroller-based SoC**: These SoCs are built around a microcontroller, designed for simple control tasks in everyday devices. Known for their low power usage and efficiency, they're perfect for applications like home appliances, car systems, and IoT devices, where processing needs are minimal, and power savings are essential.
- **Microprocessor-based SoC**: This type features a microprocessor, which can handle more demanding tasks and run operating systems. Commonly used in smartphones and tablets, microprocessor-based SoCs manage multiple tasks and support complex applications, providing the higher processing power necessary for interactive and data-intensive applications.
- **Application-Specific SoC**: Custom-designed for specific, high-performance tasks, these SoCs excel in areas like graphics processing, network management, and multimedia applications. Optimized for speed and efficiency in their designated roles, they're often used in graphics cards, AI hardware, and specialized industrial or financial systems that require precise, fast processing.

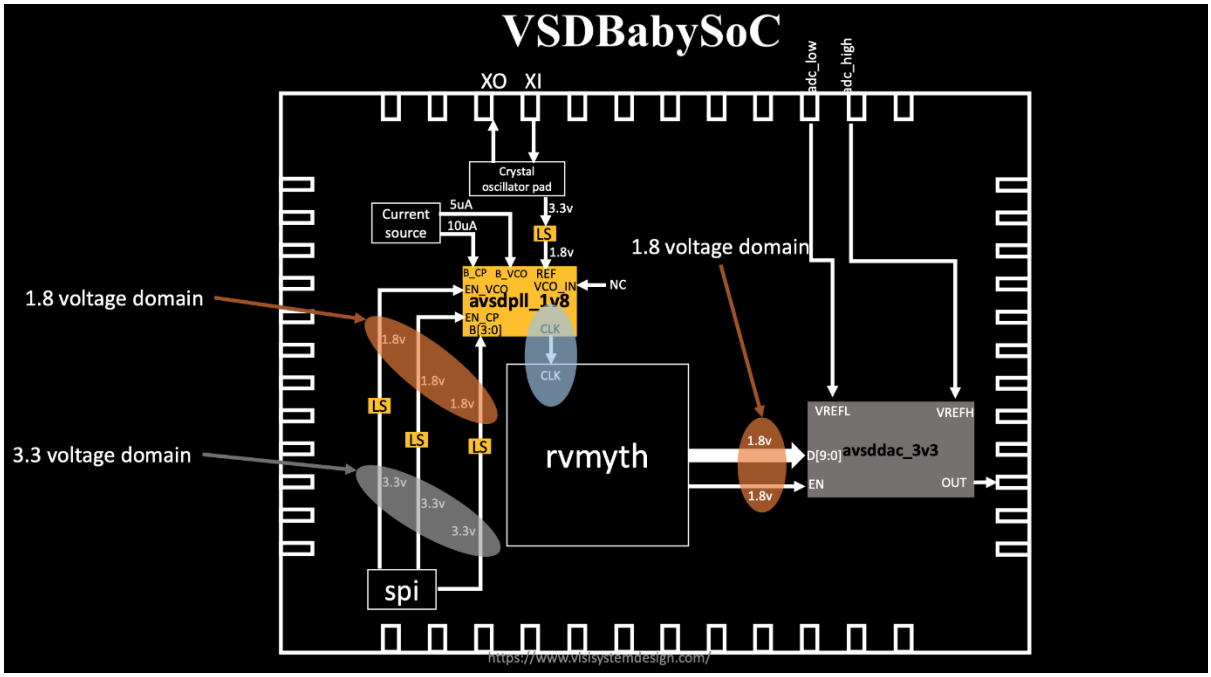
SoC Design Flow


Soc Design Flow
Soc Specification
Architecture Design (Hardware/ Software Partioning)
High Level Modeling
RTL Design
Functional Simulation and Verification
RTL Synthesis and DFT
Front-End Design
Gate Level Netlist
Place and Route
Timing Verification and Signoff
Physical Verification
Physical Design (Back-End)
Design GDSII
Manufacturing
Post-Silicon validation and Integration
Mass Production
HW/SW cosimulation
Software Development
Software Testing and Refinement
Final Code

3. Introduction to VSDBabySoC

VSDBabySoC is a compact yet highly capable System on Chip (SoC) based on the RISC-V architecture. The primary objective of designing this small-scale SoC is to facilitate the simultaneous testing of three open-source intellectual property (IP) cores for the first time while also calibrating its analog components. The VSDBabySoC incorporates an RVMYTH microprocessor, an 8x phase-locked loop (PLL) for generating a stable clock signal, and a 10-bit digital-to-analog converter (DAC) that enables communication with various analog devices.

1. **Initialization and Clock Generation**: Upon receiving an initial input signal, BabySoC activates the PLL. The PLL generates a stable and synchronized clock signal, which is essential for coordinating the activities of the RVMYTH processor and DAC. By synchronizing the system, the PLL ensures that all components operate in harmony, avoiding timing mismatches and ensuring data integrity.
2. **Data Processing in RVMYTH**: Within BabySoC, RVMYTH plays a central role in processing data. Specifically, it utilizes its r17 register to hold and cycle through values that are used by the DAC. As RVMYTH executes instructions, it sequentially updates r17 with new data, preparing it for analog conversion. This cyclical processing allows BabySoC to generate continuous data streams that the DAC can output.
3. **Analog Signal Generation via DAC**: The DAC receives the processed digital values from RVMYTH and converts them into an analog signal. This output, saved in a file named OUT, can be fed to external devices like TVs and mobile phones, which interpret the analog signals to produce sound or video. This functionality enables BabySoC to interface with consumer electronics, showcasing how digital data can drive multimedia outputs in real-world applications.



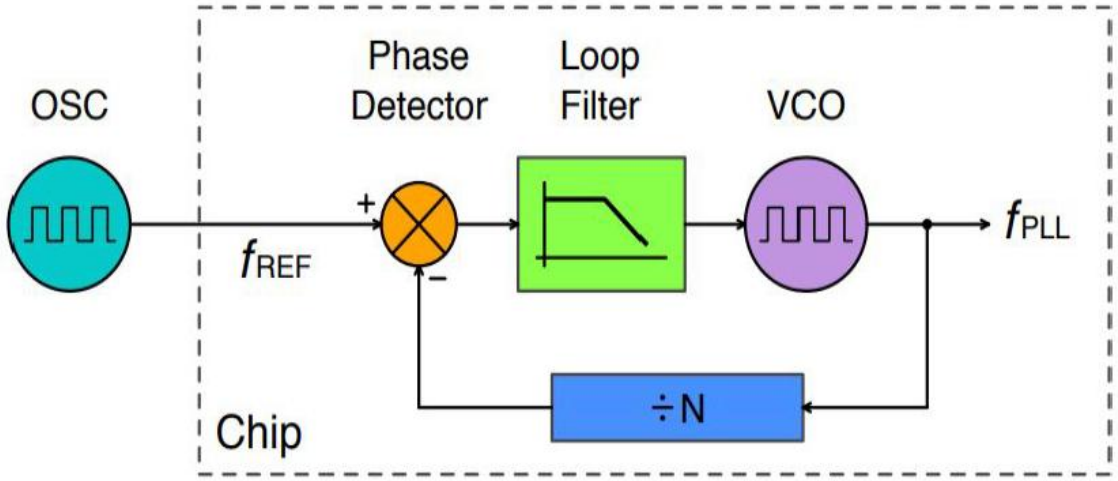
BabySoC components

- **RVMYTH (RISC-V CPU)**: RVMYTH is the brain of BabySoC, based on the open-source RISC-V design. It's a simple, customizable CPU that handles processing tasks and communicates with other parts of the SoC. This flexibility makes RVMYTH ideal for learning and experimenting with CPU architecture.
- **Phase-Locked Loop (PLL)**: The PLL generates a stable clock signal to keep everything in BabySoC running in sync. It matches the SoC's clock with a reference frequency, ensuring reliable timing for RVMYTH and DAC. PLLs are widely used to keep signals aligned in communication and timing circuits.
- **Digital-to-Analog Converter (DAC)**: The DAC turns digital signals from RVMYTH into analog output, like sound or video. This allows BabySoC to connect with external devices that use analog signals, such as speakers or displays.

PLL

- A Phase-Locked Loop (PLL) is a control system that generates an output signal whose phase is synchronized with an input signal. Both signals will have the same frequency and can either have no phase difference or a constant phase difference.

Block Diagram



Control loop locks **phase** of f_{PLL} to f_{REF}

A PLL typically consists of three main components:

- **Phase Detector:** Compares the input signal (reference) with the output signal from the oscillator and generates an error signal based on the phase difference.
- **Loop Filter:** Usually a low-pass filter that processes the error signal to produce a control voltage.
- **Voltage-Controlled Oscillator (VCO):** Adjusts its frequency based on the control voltage to match the input frequency.

Functionality:

- The PLL aims to lock the output frequency to the input frequency, maintaining a constant phase relationship between the two signals.
- In some cases, a frequency divider may be used in the feedback loop to produce an output that is a multiple of the reference frequency.

Why Can't Off-Chip Clocks Always Be Used?

1. Clock Distribution Delays:
    - Using a single clock source for an entire chip can lead to delays due to long wiring distances, which can affect timing.
2. Clock Jitter:
    - Off-chip clocks may experience variations in signal timing, known as jitter, which can disrupt synchronization.

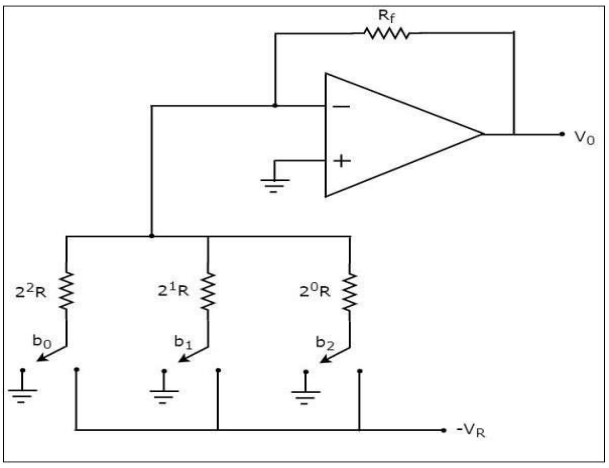
3. Different Frequency Requirements:
    - Various blocks within the same chip may require different clock frequencies. For example, one block might need 200 MHz while another needs 100 MHz.
4. Crystal Frequency Deviations:
    - When quartz crystals are used as clock sources, they come with a frequency error measured in parts per million (ppm).
    - A higher ppm error means that the frequency can deviate more from the desired value, affecting timing precision.
5. Frequency Stability:
    - The stability of a crystal's frequency can vary with temperature. Crystals with higher ppm errors are more likely to exhibit larger frequency variations when temperature changes.
6. Total Frequency Error:
    - The overall frequency error of a crystal includes contributions from:
        - Frequency Tolerance: The initial error in frequency.
        - Frequency Stability: Variation over temperature.
        - Aging: Changes in frequency over time.
    - Higher ppm errors in any of these factors can lead to larger total frequency errors, impacting the accuracy of timing references in electronic systems.

DAC

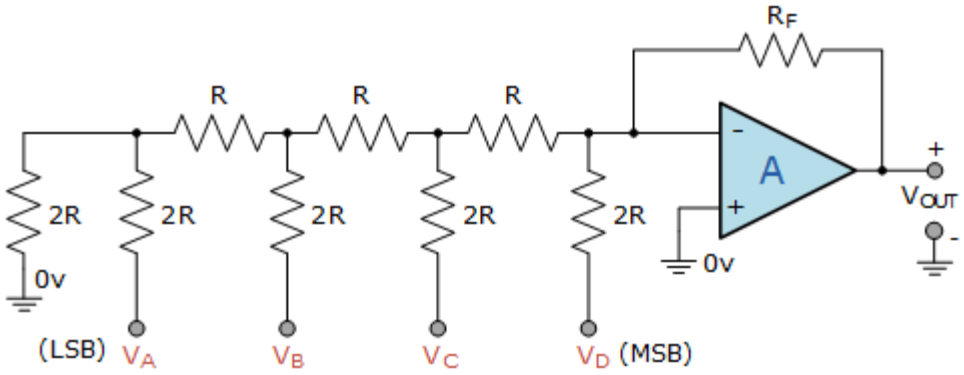
- A Digital to Analog Converter (DAC) is an electronic device that converts a digital input signal (represented in binary code) into an analog output signal.

1. Digital Signal Representation:
    - The digital input is composed of bits, specifically 0s and 1s, which represent the digital information.
2. Structure:
    - A DAC typically has multiple binary inputs and a single analog output.
    - The number of binary inputs is usually a power of two (e.g., 2, 4, 8, 16).
3. Types of DACs:
    - There are primarily two common types of DACs:

- Weighted Resistor DAC: Uses resistors with different weights to convert the digital signal into an analog voltage.



- R-2R Ladder DAC: Uses a repeating network of resistors to achieve the same effect, allowing for simpler design and easier scaling.



4. In VSDBabySoC:
    - In the VSDBabySoC design, we are utilizing a 10-bit DAC, which means it can take a digital input represented by 10 bits and convert it into an analog output.

This document outlines the structure and components of BabySoC, along with a basic understanding of SoCs and their types. By mastering these concepts and understanding how BabySoC operates, one gains a solid foundation in modern embedded systems design and digital-to-analog interfacing.