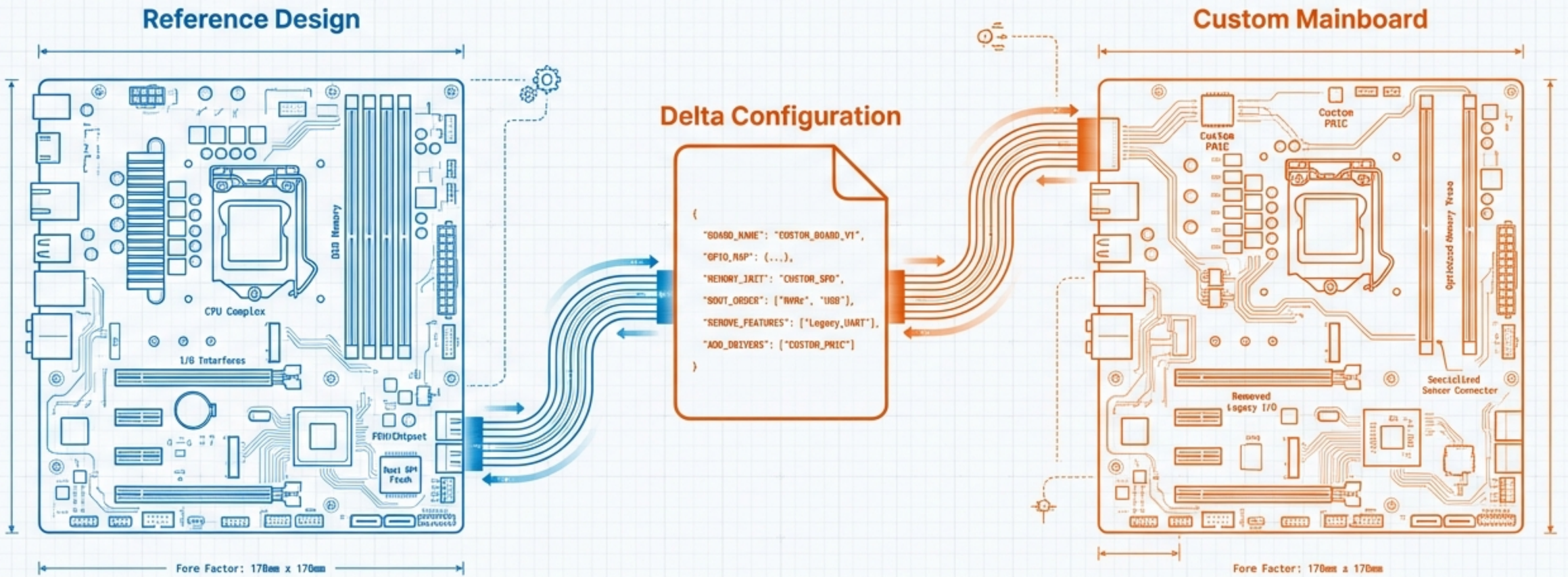
Porting Slim Bootloader to Custom Hardware

A practical guide to adapting Intel Reference Designs using the Delta configuration methodology.

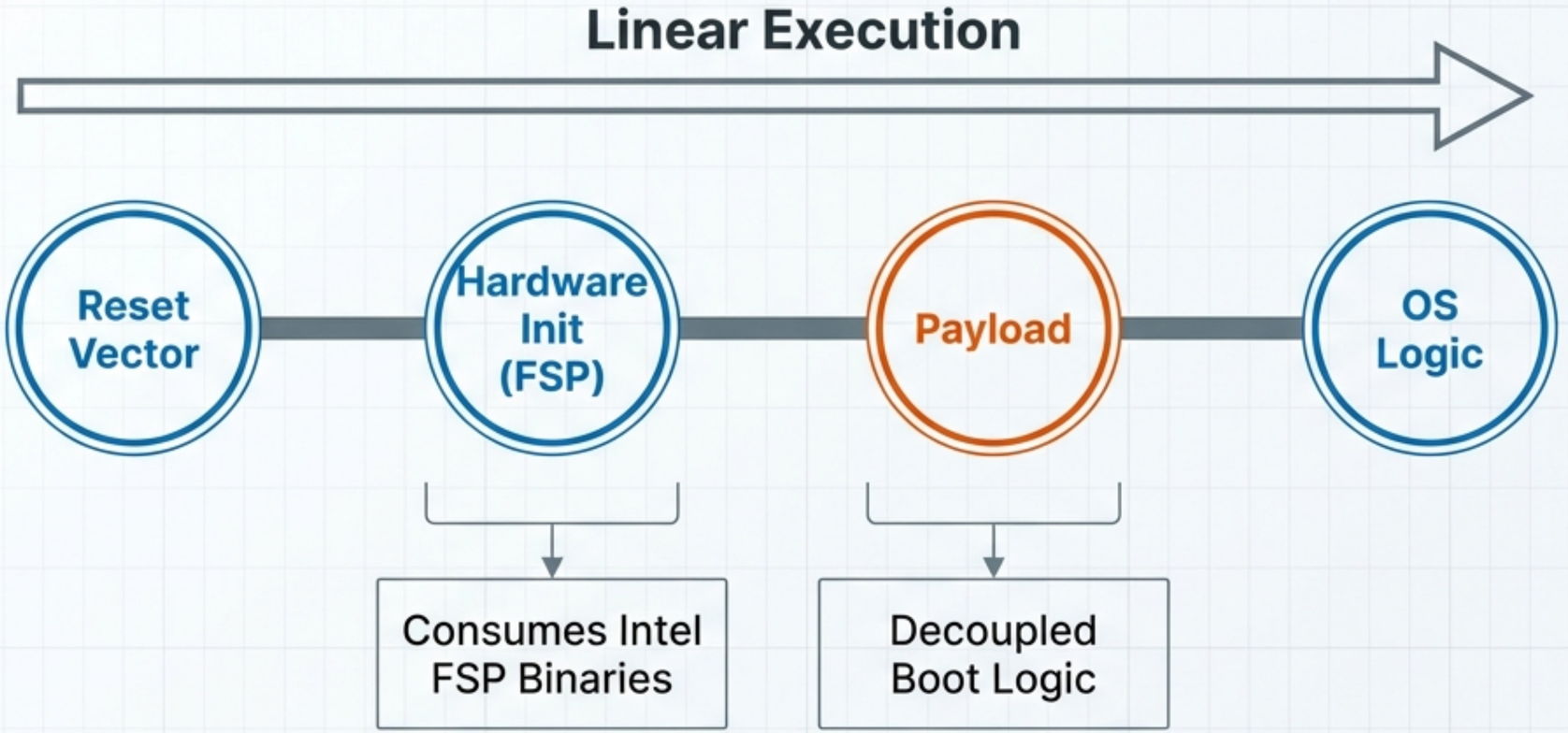


Based on the SBL Workflow | Intel IoT Group

NotebookLM

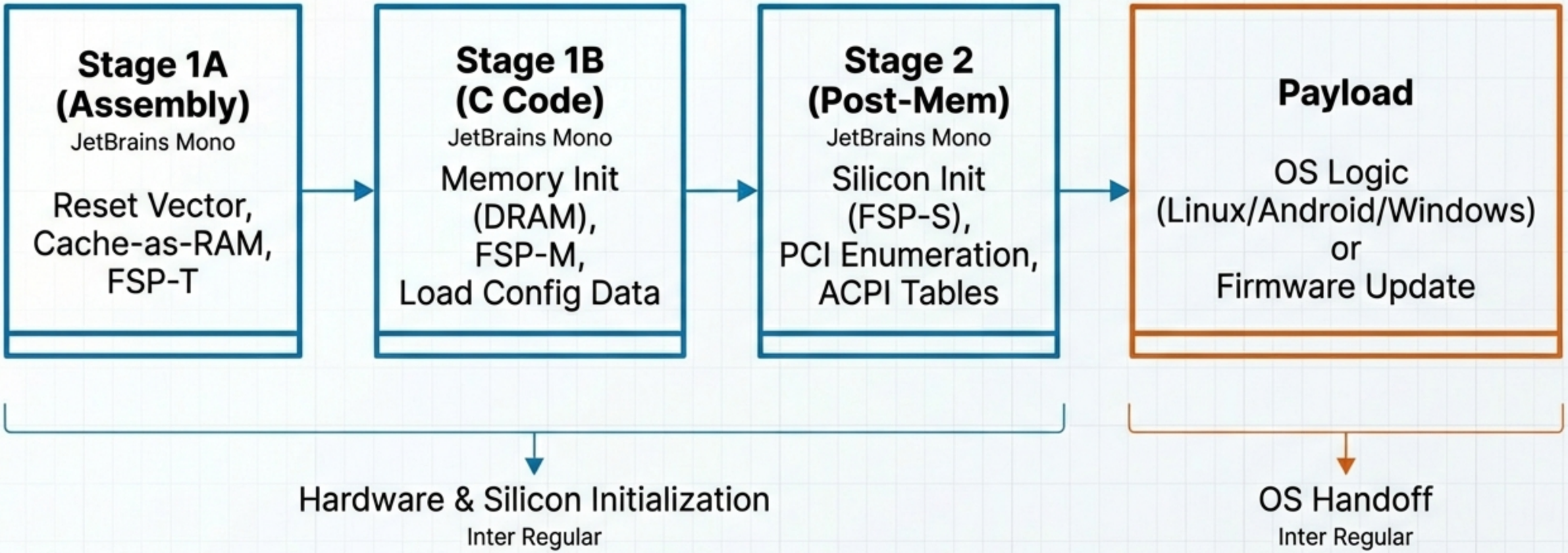
The SBL Philosophy: Linear, Predictable, Fast

Unlike legacy BIOS complexity, SBL follows a strict sequential path. It handles silicon complexity via FSP binaries, leaving a clean path for platform initialization.



NotebookLM

Architecture Breakdown: The Four Boot Stages



NotebookLM

The Developer’s Role: Silicon vs. Platform

Focus your effort on the Platform directory. Intel handles the complex silicon initialization.

Intel Provided (Silicon)

Silicon Reference Code (SRC) & FSP Binaries.

Read-Only / Fixed Base.

`Silicon/`

Developer Focus (Platform)

Mainboard Porting & Configuration.

Your Customization Area.

`Platform/`

NotebookLM

Environment Setup and Toolchain

Pre-flight Checklist

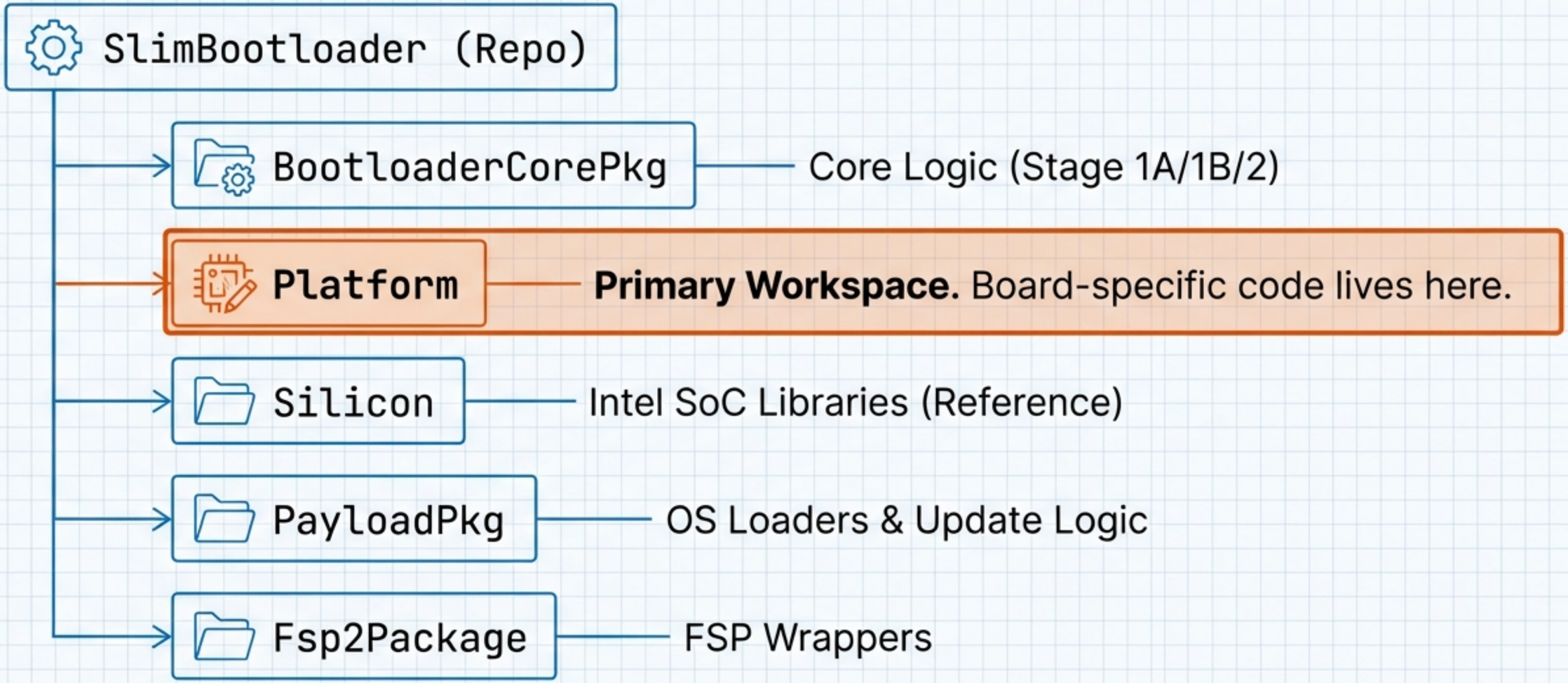
Supported Operating Systems	
☐ Windows	Visual Studio 2015+ (x86 C/C++ Tools)
☐ Linux	Ubuntu 18.04 LTS (GCC Compiler)

Required Toolchain	
☐ **Python** : JetBrains Mono **Bold**	: Critical for Build Scripts & Config Editor
☐ NASM : JetBrains Mono	: Assembly Compilation
☐ OpenSSL : JetBrains Mono	: Build-time Signing & Crypto
☐ iASL : JetBrains Mono	: ACPI Image Generation

NotebookLM

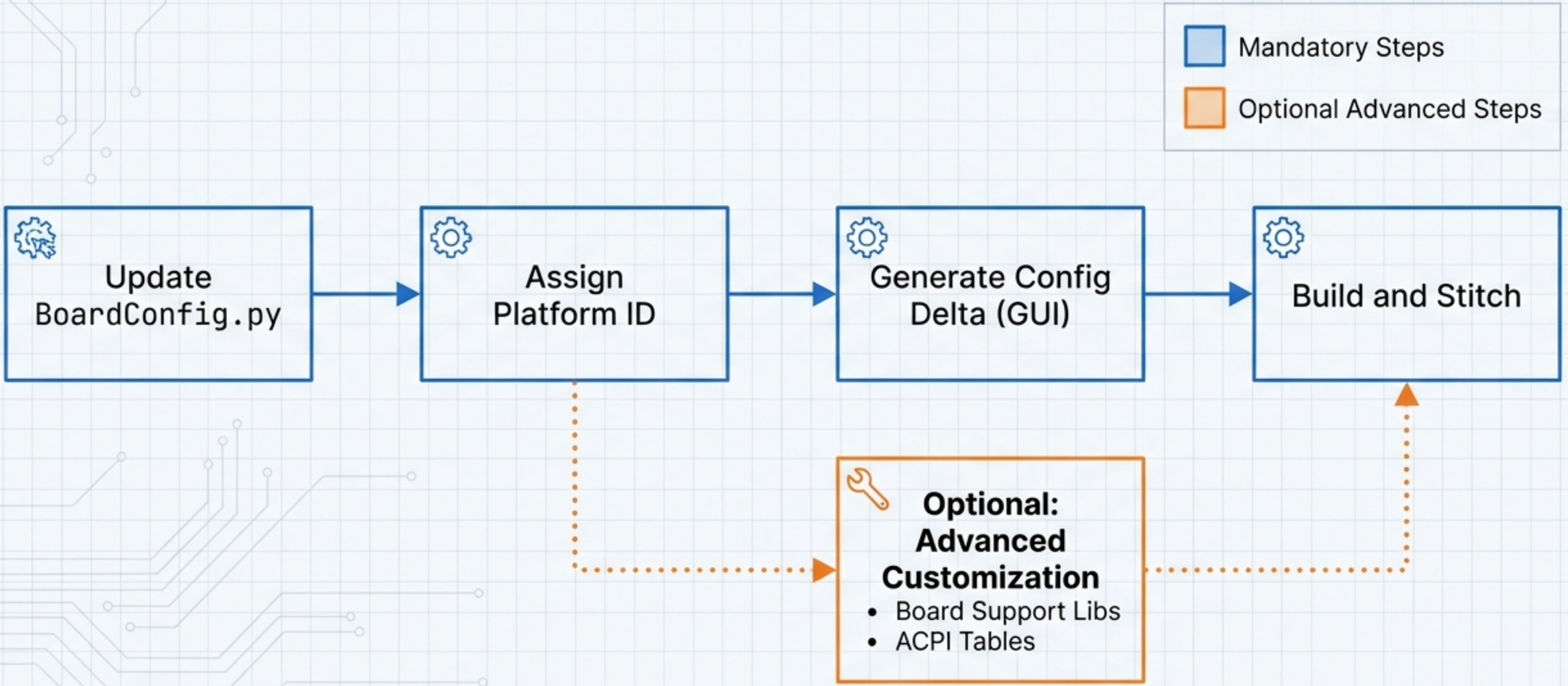
Navigating the Source Code Structure

90% of the porting work occurs within the Platform directory.



NotebookLM

The Porting Workflow: From Reference to Custom



NotebookLM

Step 1: Defining Static Settings

The BoardConfig.py file acts as the foundational definition for your platform.

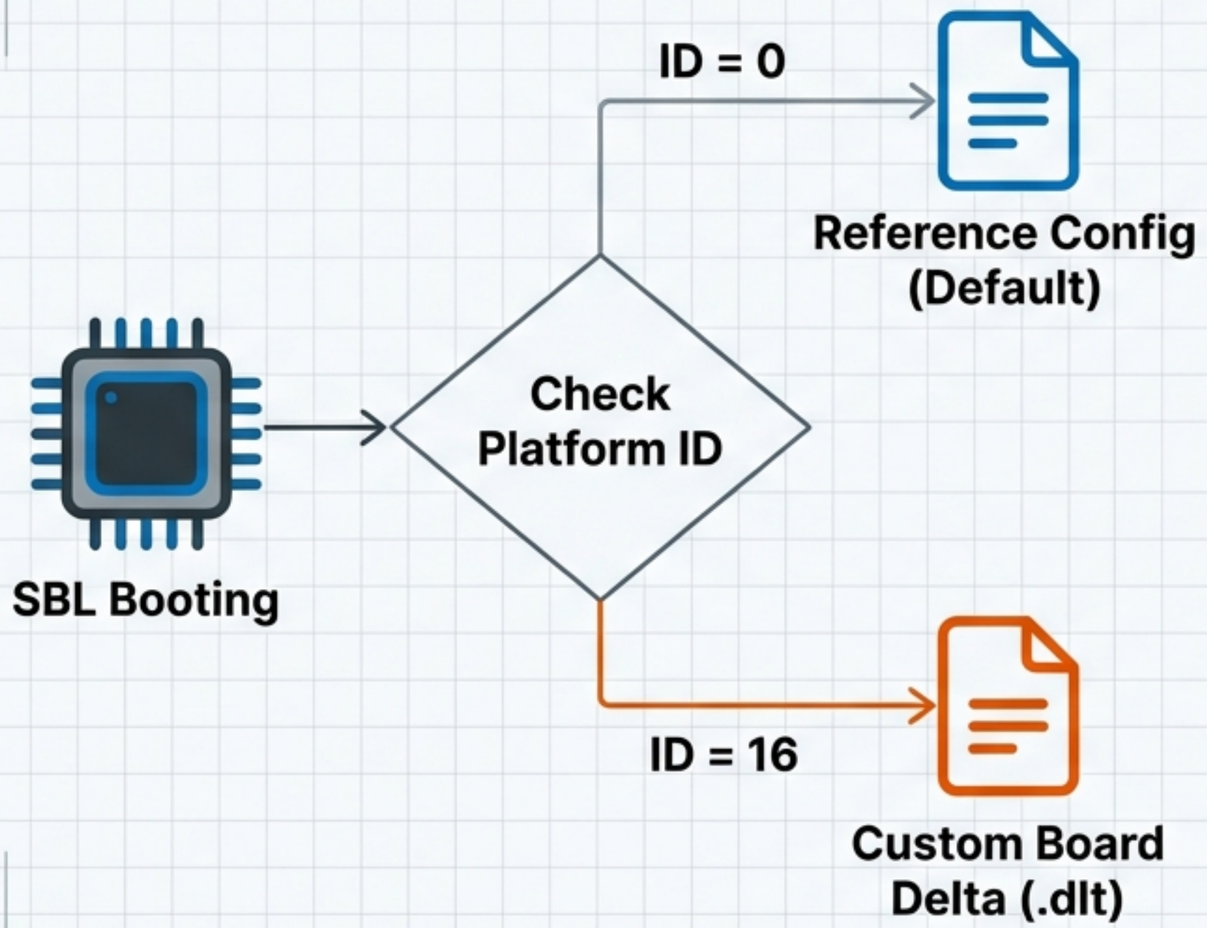
```
class BoardConfig(BaseConfig):
    def __init__(self, index):
        # Image Layout
        self.VERINFO_IMAGE_ID      = 'SB_CFL'
        self.VERINFO_PROJ_MAJOR_VER = 1

        # Silicon Features
        self.ENABLE_VTD            = 1
        self.ENABLE_MEASURED_BOOT = 0 # Disable for Dev
```

Defines Image Layout

Toggle Silicon Features

NotebookLM

Step 2: Assigning a Unique Platform ID

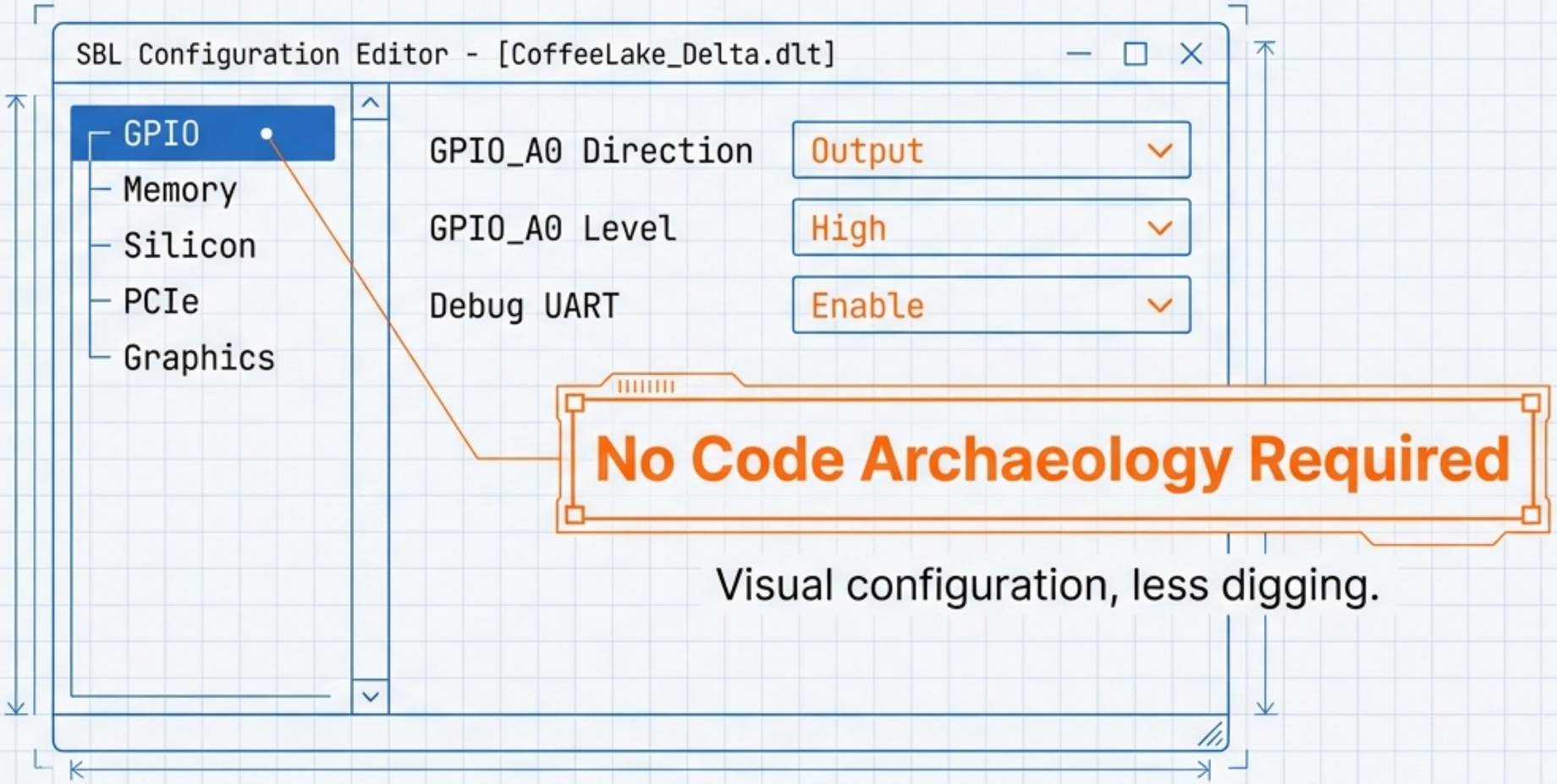


- SBL supports up to 32 distinct `Platform IDs` in a single binary.
- `ID 0` is reserved for the **Base Reference**.
- The **Platform ID** triggers the loading of the specific **Delta file** at runtime.

NotebookLM

Step 3: The Config Editor (GUI)

Replace header-file hunting with a visual interface. Save changes as a Delta (.dlt) file.

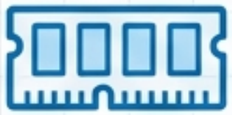


Visual configuration, less digging.

NotebookLM

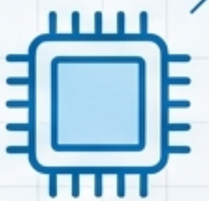
Deep Dive: What Can You Configure?

Memory



Configure 'Memory Down' designs, read SPD from BIOS, or set FSP-M parameters.

Silicon Features



Enable/Disable Embedded Controllers (EC), Legacy Serial IO, and Debug ports.

PCIe & Root Ports

Configure Root Ports (RPs) and lane bifurcations via simple dropdowns.

Graphics

Set aperture size, and toggle EDP / HDMI outputs.

NotebookLM

Solving the GPIO Complexity

The Old Way

```
GPIO_CFG = 0x84000100, 0x00000000, 0x00000001...
```

⚠ Complex Hexadecimal (4 Dwords per pad). Error-prone.

The SBL Way

Pin: GPIO_12

Direction: Output

State: High

Human-readable. Integrated with Intel EDS specs.

NotebookLM

When the GUI isn't enough...

Step 4 (Optional): Advanced Customization

Board Support Libs

- Use when hardware deviates from reference.
- **Example: Custom SuperIO UART implementation.**

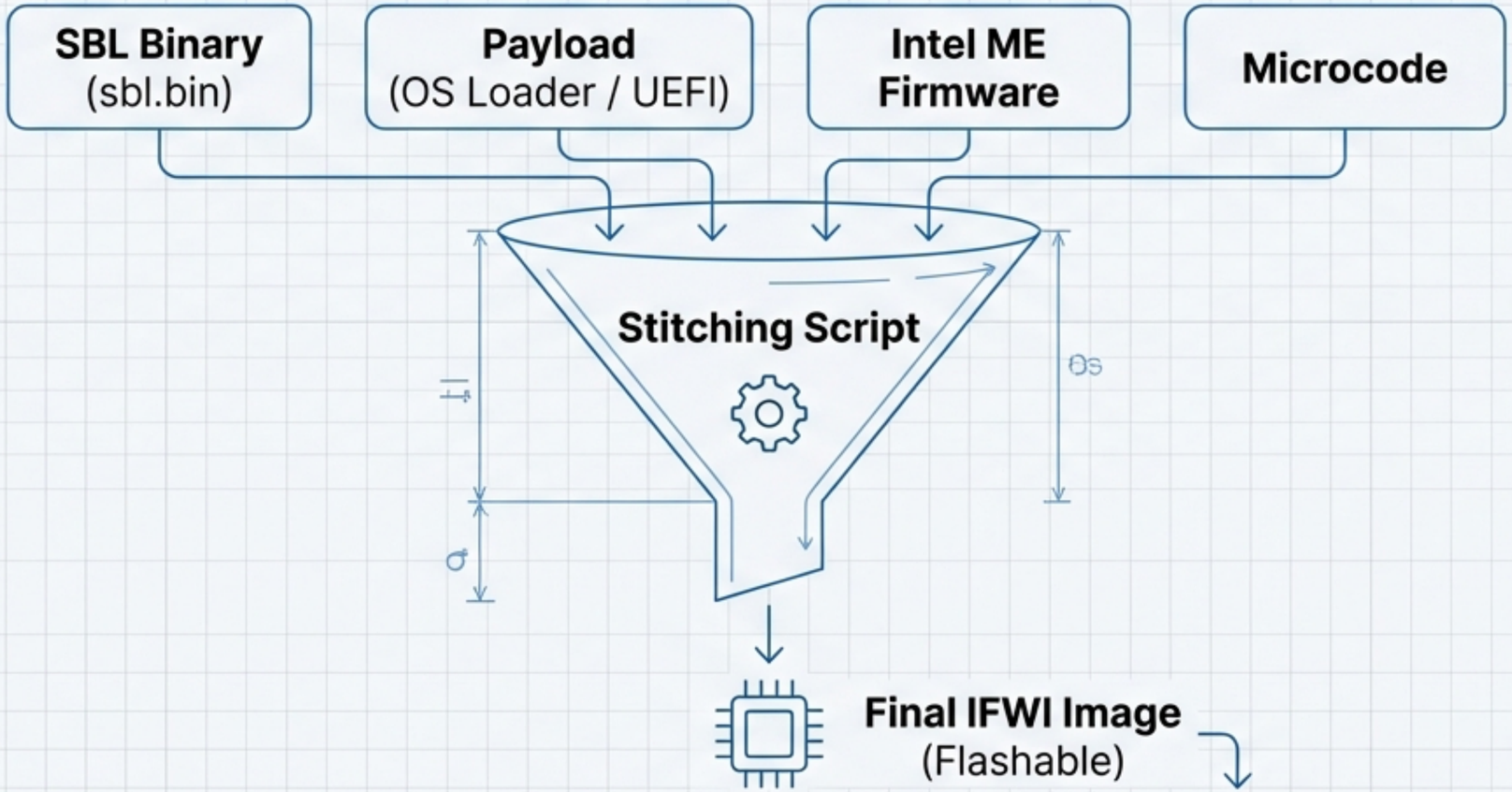
ACPI / ASL

- Modify DSDT tables.
- Add new `.asl` files for **custom hardware** features.

NotebookLM

Step 5: Payload, Build, and Stitch

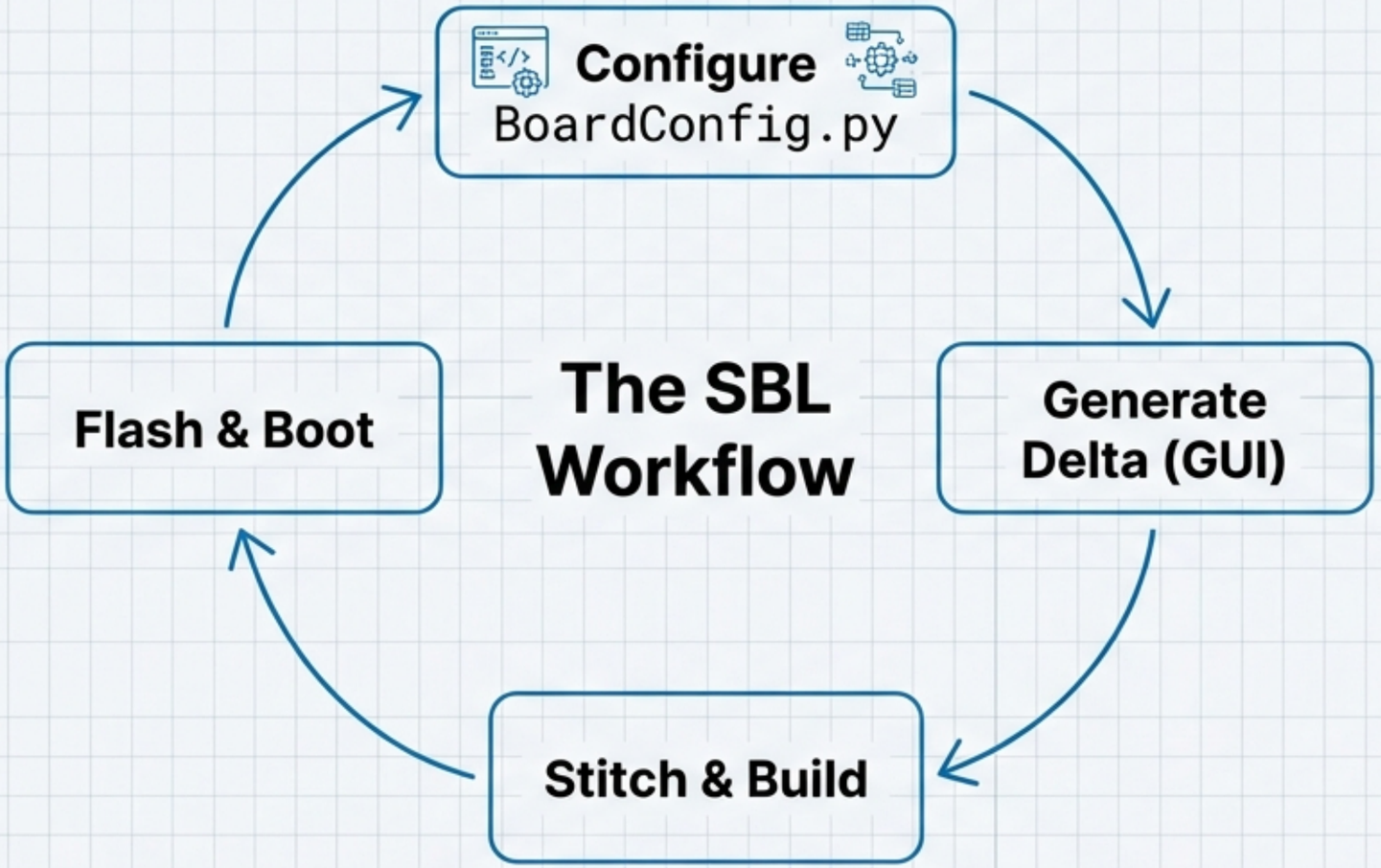
Combine your configuration with binary ingredients to create the final flashable image.




```
python BuildLoader.py build <Platform>
```

NotebookLM

Summary & Resources



Next Steps

- Clone the GitHub Repository
- Review the “Up Extreme” Demo in source
- Join the SBL Community

Documentation: slimbootloader.github.io

NotebookLM