**Automating FPGA Benchmarking with a Bash Script**

This section explains the automated FPGA benchmarking infrastructure I developed using a Bash-based script. The documentation aims to help readers understand the tool's purpose, how it works, and its significance in streamlining benchmarking workflows.

**Overview**

Field Programmable Gate Arrays (FPGAs) are reconfigurable chips widely used in applications like telecommunications, AI, and data centers. Benchmarking these devices helps developers evaluate performance metrics such as resource utilization, power consumption, and timing efficiency.

Manually benchmarking FPGA designs can be complex, time-consuming, and prone to errors. This script simplifies the process, ensuring consistent and reproducible results.

**Key Features**

* **Support for Multiple Tools**: Automates workflows for tools like Raptor, Vivado, Radiant, and Libero.
* **Customizable Options**: Enables user-defined parameters like device selection, synthesis modes, and placement options.
* **Environment Setup**: Automatically configures paths and environmental variables for RTL files and constraints.
* **Build Organization**: Outputs benchmarking results in well-structured, timestamped directories.
* **Version Control Integration**: Allows users to push results to Git repositories for collaboration.

**How It Works**

**1. Input Configuration**

The script accepts several command-line arguments for defining parameters like:

* **Design Selection**: Specifies the VHDL/Verilog/SystemVerilog design.
* **Tool Specification**: Identifies the FPGA development tool (e.g., Vivado, Raptor).
* **Device Targeting**: Chooses the device to benchmark.
* **Optimization Options**: Includes place-and-route (PnR) directives and synthesis modes.

Example command:

./run\_benchmarking.sh -design=my\_design -tool=vivado -device=zynq7k -synmode=delay

**2. Tool-Specific Setup**

Based on the input tool, the script:

* Loads the required tool module.
* Sets environment variables for tool-specific constraints and settings.
* Configures the build environment.

**3. Automated Workflow**

The script automates:

1. **Synthesis**: Configures and compiles the design.
2. **Place-and-Route (PnR)**: Optimizes design placement on the FPGA fabric.
3. **Constraint Integration**: Applies user-defined or default constraints for accurate benchmarking.

**4. Build Logging**

Generates comprehensive logs capturing configuration settings, tool outputs, and build results for further analysis.

**5. Git Integration**

Users can:

* Push results automatically.
* Opt for confirmation before pushing.
* Skip pushing entirely if desired.

**Why It Matters**

**Time Efficiency**

Eliminates repetitive manual tasks, saving significant time during benchmarking.

**Consistency and Reproducibility**

Ensures consistent settings across runs, reducing variability and enabling meaningful comparisons.

**Scalability**

Adapts to various tools and design configurations, accommodating larger workflows as needed.

**Collaboration**

Facilitates teamwork by enabling seamless sharing of benchmarking data via version control.

**Use Cases**

* Evaluating FPGA design performance on different architectures.
* Comparing vendor tools to select the optimal solution for your project.
* Analyzing and improving FPGA resource utilization and performance metrics.

**Explore More**

For additional information and a closer look at the script and its details, visit my [GitHub repository.](https://github.com/Talha-Moghul/Benchmarking)