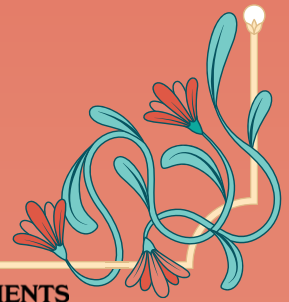
THE LINUX FOUNDATION
OPEN SOURCE SUMMIT
EUROPE

Embedded Linux Conference

Stop Rewiring: Scale Embedded Hardware Testing

Presented By: Franklin Cooper



THE LINUX FOUNDATION

Texas Instruments

About us: TI Processors and Open-source







Decades of contribution and collaboration

Ingrained culture to give back to the community





Upstream FIRST!





Focus on long term, sustainable and quality products

Upstream and Open-source ecosystem in device architecture





















Upstream FIRST mentality!

About Your Speaker

Franklin Cooper, *Senior Software Engineer (MGTS), Dallas, Texas (USA).*

15+ years in embedded software
Linux • Zephyr • Yocto • U-Boot • CI/CD & test automation



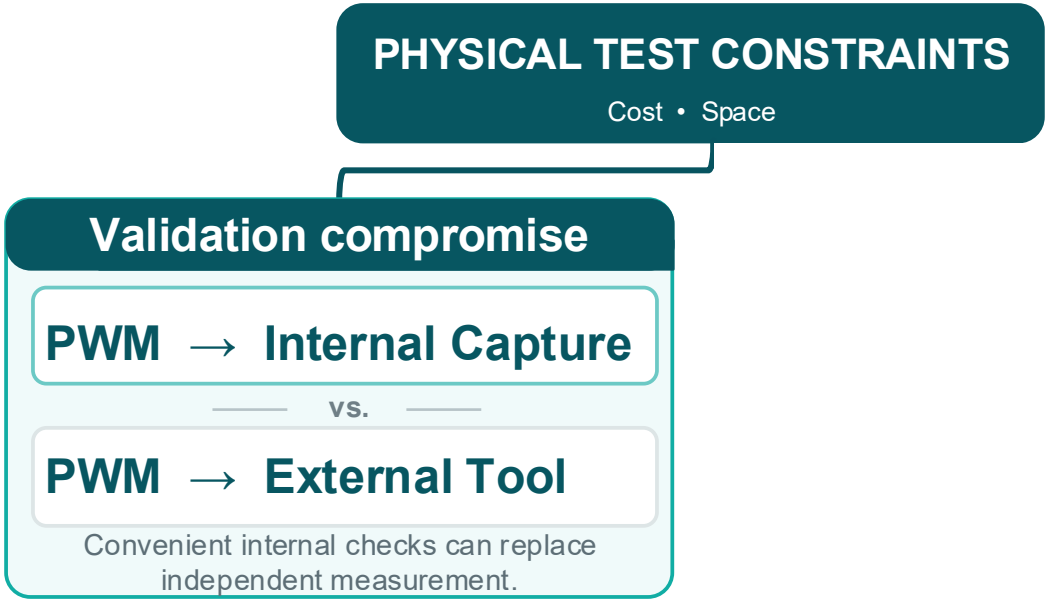
Disclaimer

- This is a technology presentation, not product-readiness or roadmap commitment
- Opinions presented here are that of the speakers and may not reflect that of Texas Instruments Inc.

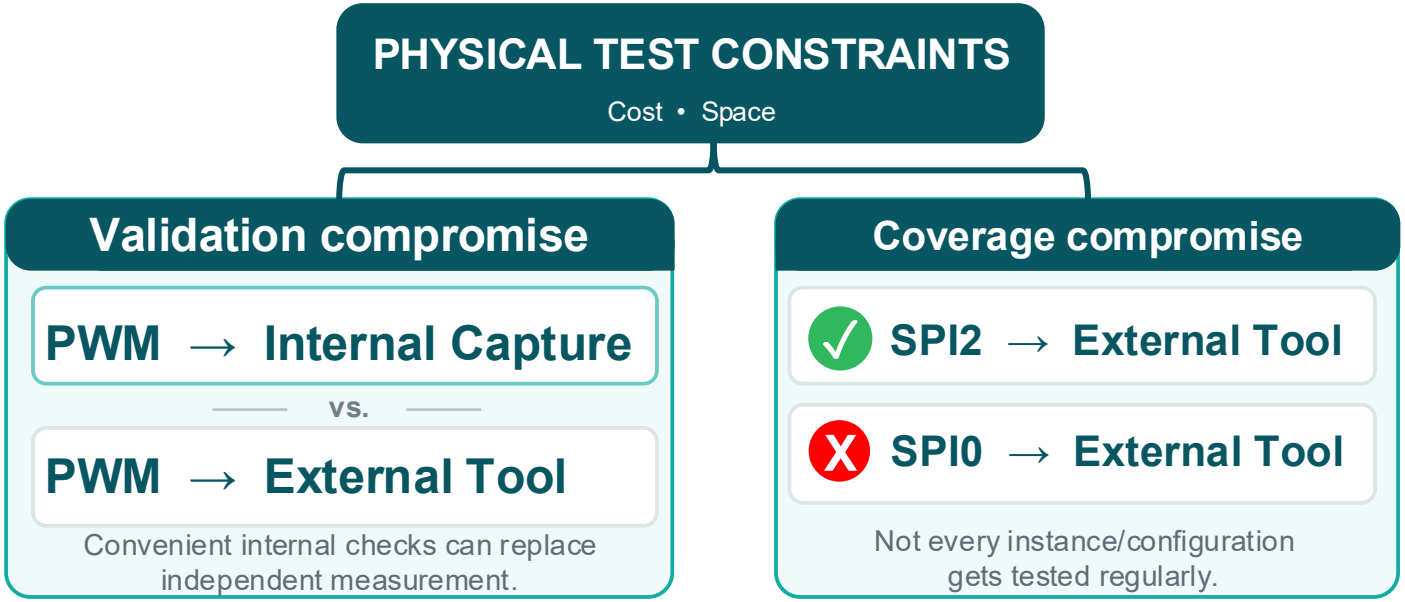
Overview

- Why fixed wiring limits automated testing
- Making the wiring programmable
- WireWeaver: architecture and workflow
- Where it works — and where it doesn’t
- Getting started and next steps

We test less—and sometimes test less effectively

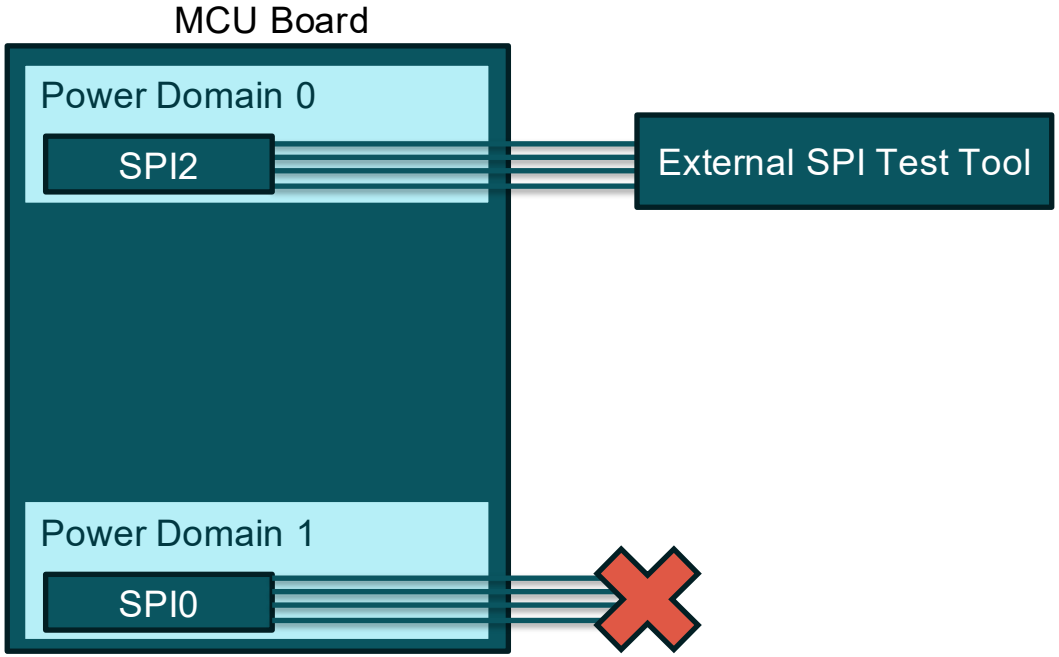


We test less—and sometimes test less effectively



Result: defects can escape automated test

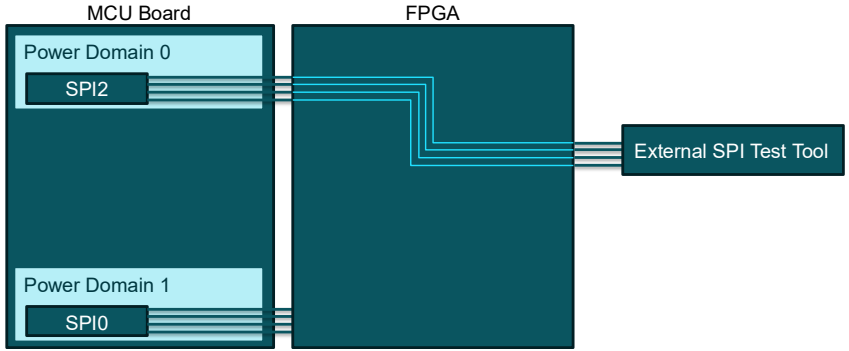
Fixed Wiring Determines What We Can Test



SPI0 requires rewiring to receive equivalent automated external validation.

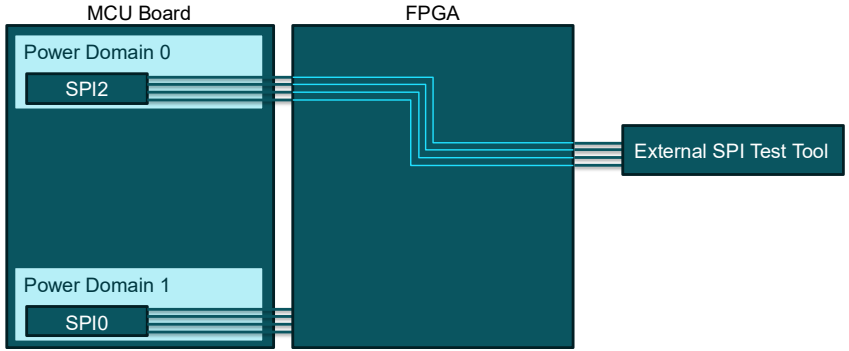
Make the Wiring Programmable

Test 1: SPI2 → External Tool

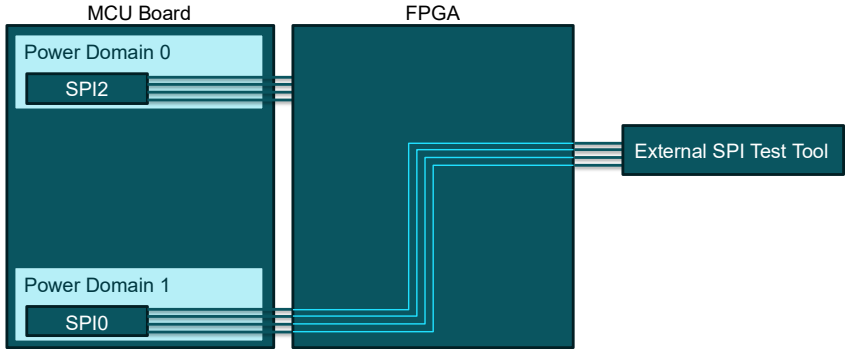


Make the Wiring Programmable

Test 1: SPI2 → External Tool



Test 2: SPI0 → External Tool



Same hardware setup. Different routing per test.

Why Validation-Oriented Approaches Didn't Scale for Software Teams

Validation-oriented approaches I encountered

- Built for **silicon / platform validation**
- **Custom hardware** + complex FPGA designs
- **$2,500+ per setup**

Programmable FPGA routing wasn't new. Making it practical at software-team scale was the goal.

Why Validation-Oriented Approaches Didn't Scale for Software Teams

Validation-oriented approaches I encountered

- Built for **silicon / platform validation**
- **Custom hardware** + complex FPGA designs
- **$2,500+ per setup**

Software-team requirements

- Affordable for **individual developer**
- Replicable across **developer desks and CI**
- No FPGA development expertise required
- Open, cross-platform tooling

Programmable FPGA routing wasn't new. Making it practical at software-team scale was the goal.

WireWeaver - Architecture

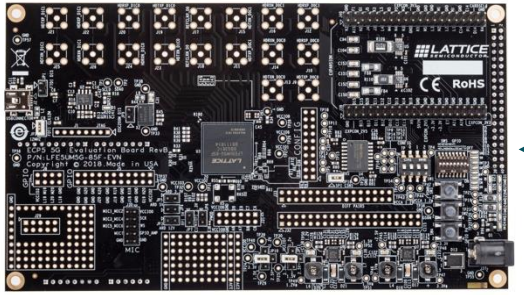
Host PC

Python Routing Tool

Open-source FPGA Toolchain

Generate / Program







DUT

MCU Board

FPGA Board

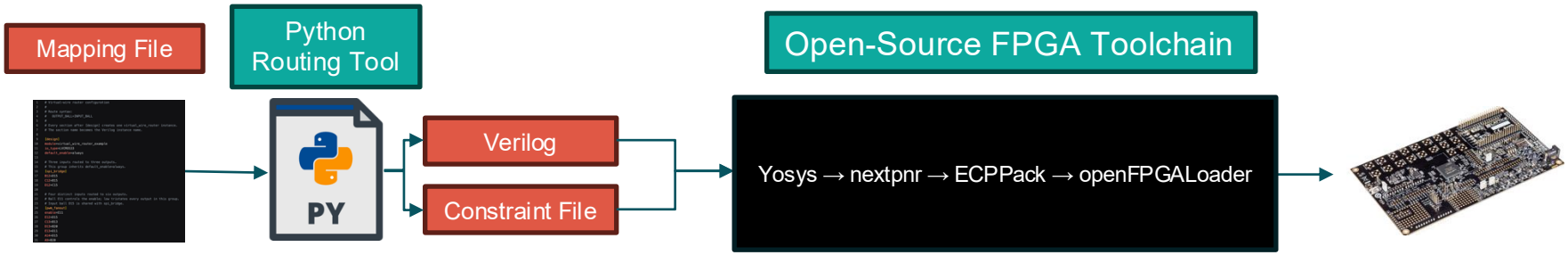
ECP5-5G

100+ routable I/O

External Tool / Other Board

Host software selects the routing. The FPGA connects the physical interfaces

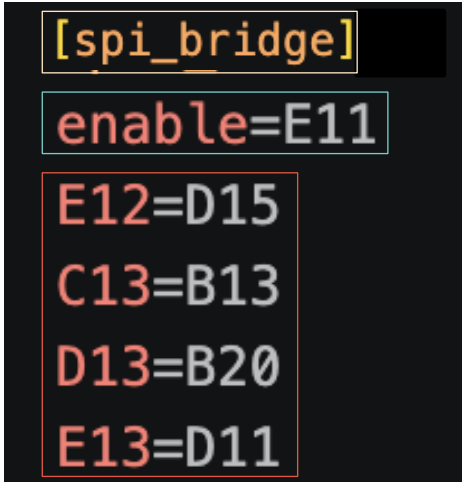
WireWeaver - Python Routing Tool



Software engineers describe the routing — the tool handles the FPGA implementation.

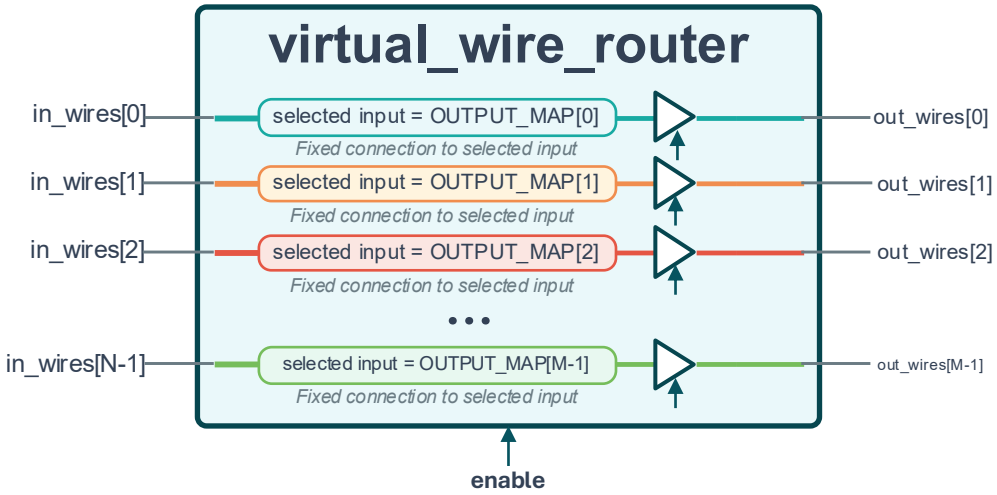
Tests Describe the Connections They Need

- **Groups** - keep related signals together
- **Routes –** define pin-to-pin connections
- **Enable** - optionally connect/disconnect a group



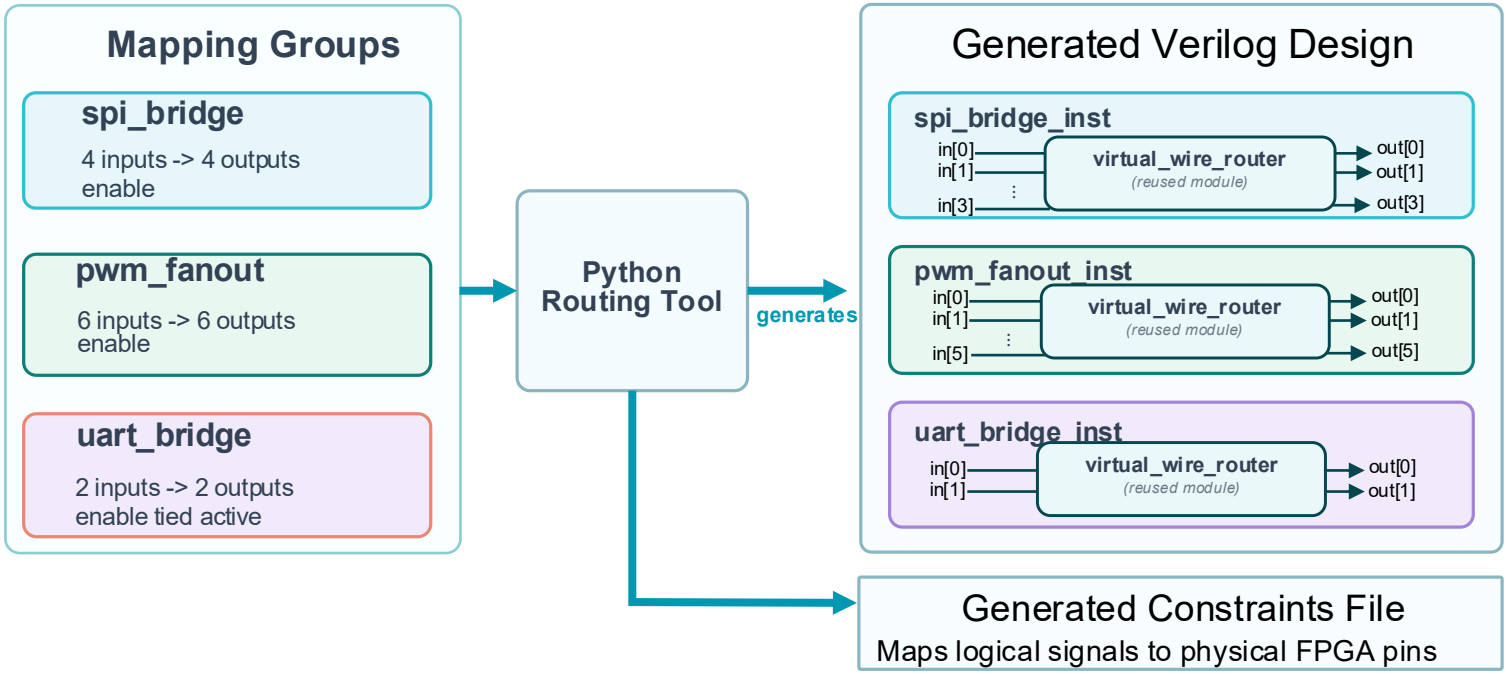
User Generated Mapping File

Reusable Routing Module



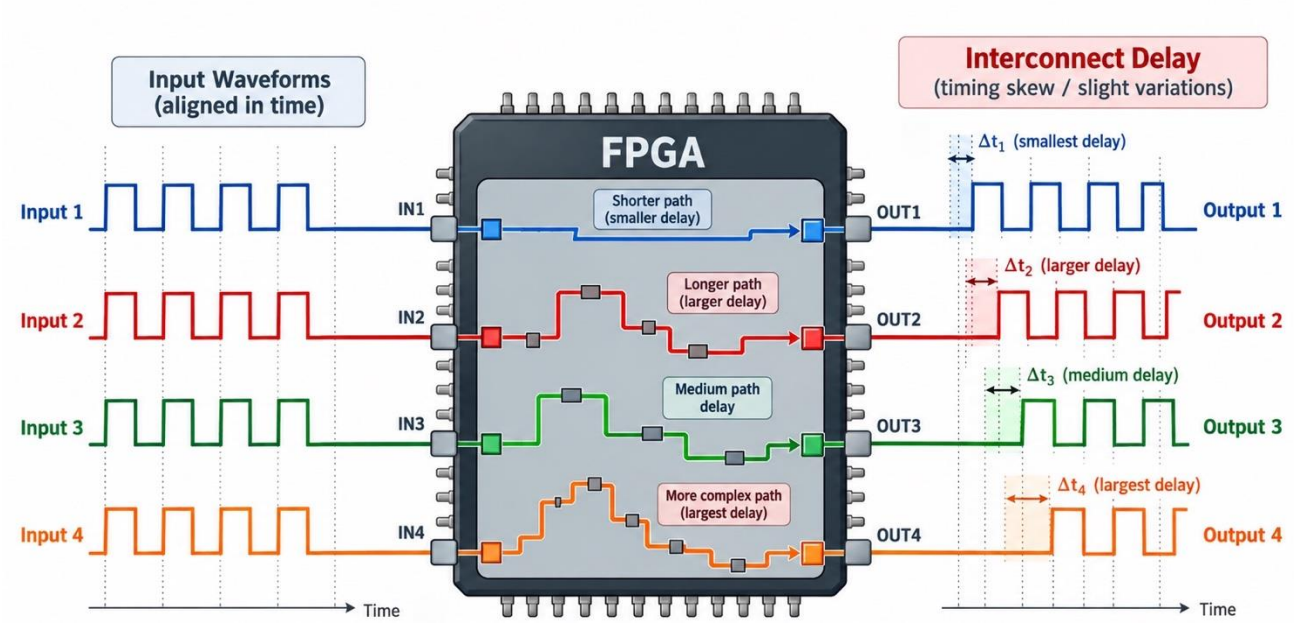
- **Parameterized input and output counts**
- **OUTPUT_MAP defines a fixed input for each output**
- **Optional enable tri-states all outputs**

Generated Design = Multiple Routing Module Instances



One reusable module definition. One generated instance per mapping group. Together, they form the complete routing design.

FPGA Routing Adds Delay and Skew



Different FPGA routes introduce different propagation delays, creating skew between related signals.

Where This Approach Fits

Good Fit

- GPIO
- UART
- PWM
- **SPI (low/moderate speed)**
- **CAN_TX / CAN_RX**
- RTC_OUT
- and more

Where This Approach Fits

Good Fit

- GPIO
- UART
- PWM
- **SPI (low/moderate speed)**
- **CAN_TX / CAN_RX**
- RTC_OUT
- and more

Not supported by current design

- **Bidirectional protocols**
- **Open drain based peripherals**
- **Analog peripherals**
- **High-speed peripherals**

Best suited to low- and moderate-speed, single-ended digital signals.

Performance Speaks For Itself

Reconfiguration time

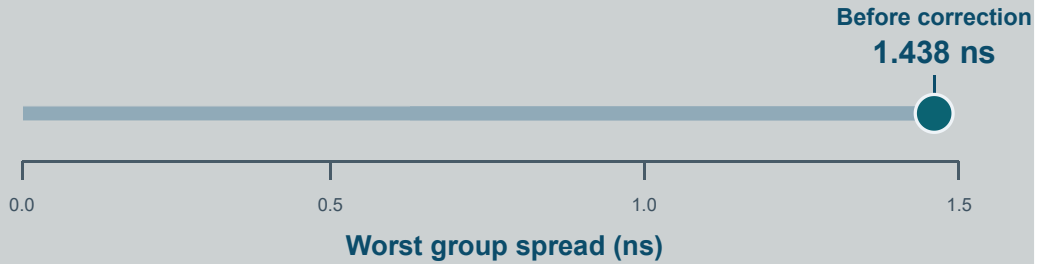
Performance Speaks For Itself

Reconfiguration time	
First configuration	11 s
Subsequent configuration	4 s
Bitstream generation	8 s → 1 s
FPGA programming	3 s

Performance Speaks For Itself

Reconfiguration time		Delay Correction Reduces Routing Skew
First configuration	11 s	
Subsequent configuration	4 s	
Bitstream generation	8 s → 1 s	
FPGA programming	3 s	

Performance Speaks For Itself

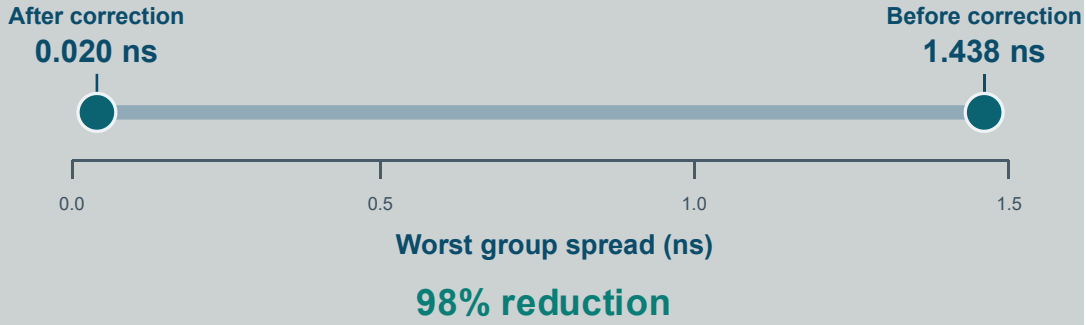
Reconfiguration time		Delay Correction Reduces Routing Skew
First configuration	11 s	
Subsequent configuration	4 s	
Bitstream generation	8 s → 1 s	
FPGA programming	3 s	

Based on map file with 6 groups using 42 total pins

Performance Speaks For Itself

Reconfiguration time	
First configuration	11 s
Subsequent configuration	4 s
Bitstream generation	8 s → 1 s
FPGA programming	3 s

Delay Correction Reduces Routing Skew



Based on map file with 6 groups using 42 total pins

Getting Started in Less than 5 Minutes

- 1. Install OSS CAD Suite
  - Download and extract
- 2. Clone the repository
- 3. Create a mapping file
- 4. Run one command

```
python3 build_virtual_wires.py <mapping.ini>
```

Generates Verilog + constraints → bitstream → programs FPGA

default.ini

```
1
2
3
4   [build]
5   output_dir=generated
6   oss_cad_suite=oss-cad-suite
7   runs=1
8   jobs=1
9   board=ecp5_evn
```

Optional configuration

default.ini sets reusable build defaults

Next Steps: Integrate and Package

Ensure True Flexibility

- **Fix Windows specific issues**
- **Remove hardcoded reference to Lattice FPGA.**

Adapter PCB

- Build the physical adapter between the DUT and FPGA
- Adding analog switches

Zephyr Twister Integration

- Tests declare the required hardware connections
- Twister automatically configures the FPGA before the test runs

Simplify Map File Generator

- Generator based on device tree
- Generic GUI

Credits and Acknowledgement

- Texas Instruments Inc.
- The Linux Foundation.
- OSS CAD Suite
  - Yosys
  - Nextpnr
  - Ecppack
  - OpenFpgaLoader
- AI coding agent

Q&A

Contact Information:
Franklin Cooper fcooper@ti.com

Project Link:
https://github.com/fcooper/WireWeaver

Also on IRC @ libera.chat #linux-ti

Learn more about TI products

- https://www.ti.com/opensource
- https://www.ti.com/zephyr
- https://www.ti.com/processors
- https://www.ti.com/edgeai



Why choose TI MCUs and processors?

Scalability
Our products offer scalable performance that can adapt and grow as the needs of your customers evolve.

Efficiency
We design products that extend battery life, maximize performance for every watt expended, and unlock the highest levels of system efficiency.

Affordability
We strive to make innovation accessible to all by creating cost-effective products that feature state-of-the-art technology and package designs.

Availability
Our investment in internal manufacturing capacity provides greater assurance of supply, supporting your growth for decades to come.

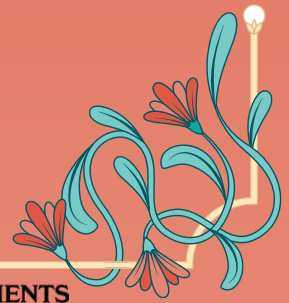
THE LINUX FOUNDATION

OPEN SOURCE SUMMIT



EMBEDDED LINUX CONFERENCE

EUROPE



Texas Instruments