

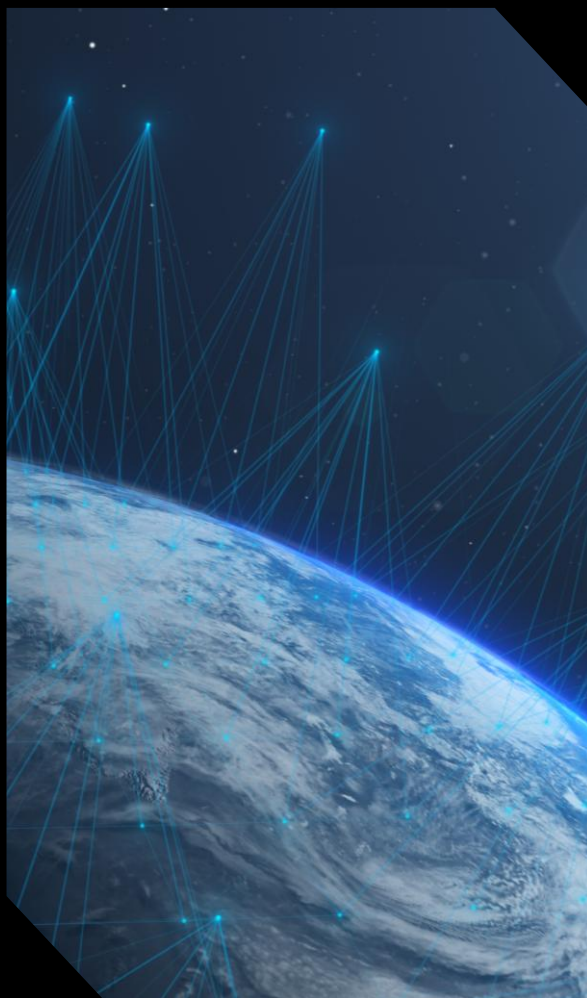


AMD Ross™

**Agentic AI Assistant for
Embedded Development**



Agenda



Introducing AMD Ross™ Agentic AI Assistant

01



AMD Ross: Key Components For Your Design Team

02



Examples of How AMD Ross Helps with Designs

03



Getting Started & Additional Resources

04



Introducing AMD Ross™ Agentic AI Assistant

Too Much Time on Tool Mechanics, Less Time on Innovation



Tool Complexity

Juggling tools and scripts on every project



Knowledge Silos

Expertise lives in individual engineers' heads, not systems



Slow Debug Cycles

Trial & error and manually parsing log files



Repetitive Tasks

Manual setup and configuration repeated across every design

INTRODUCING...

AMD Ross

**YOUR AGENTIC AI ASSISTANT
FOR EMBEDDED DEVELOPMENT**

Across Adaptive and x86 Embedded Compute

System
Architecture

Schematic
& PCB

FPGA
Design

Algorithm
Design

Embedded
Software

AI Inference

Debug

Deployment

AMD
Power Design
Manager

AMD
Vivado

AMD
Vitis

AMD
Embedded
Solutions

AMD
Vitis
AI

AMD
RYZEN AI
Software

AMD
ROCm.AI



From Individual Productivity to Connected Intelligence



Accelerate Complex Design Workflow

Automate repetitive work to focus on differentiation



Scale AMD Expertise Across the Team

Bring proven AMD expertise into every developer's workflow



Context-Aware Intelligence Across the Flow

Carry context across tools from design to debug





AMD Ross™: Key Components For Your Design Team

Key Components of Ross

Natural-Language Interaction Grounded in AMD Expertise and Connected Tools

ASK



INTERPRET



CONNECT



INFORM OR ACT

Client - Agnostic Interface

VS Code

Claude Code

Cursor

Devin

OpenCode CLI

Any MCP
Compatible Client

AMD Ross™ Agentic Workflow

Connection

Model Context Protocol (MCP)

AMD Vivado™ Design Suite | Vitis™ Tools | PDM | ChipScope
ROCm™ Software | Ryzen™ AI Software | Docs

Action

Agent Skills

Autonomous Flows | Domain Expertise | Execution | Reports

Context

Vectorized Knowledge Base

Flow

Design Examples

Key Components of Ross: MCP & Agent Skills

Based on the Open Standards

MCP gives the **information**



Model Context Protocol

Open standard by Anthropic; adopted across the industry

What it provides:



Tool Discovery:
LLM discovers
available AMD
tools at runtime



**Knowledge
Access:** RAG-
powered search
across AMD docs



Session Control:
Start, stop, monitor
AMD tool sessions



Execution Bridge:
Run any command
via standardized
API

Why it matters:

- **IDE-Agnostic:** VS Code, Cursor, Claude Code, Devin, any MCP client
- **LLM-Agnostic:** Claude, GPT, Gemini, Qwen, Llama — any model

Agent Skills give the **power to act**



Anthropic Open Standard

Markdown files encoding domain expertise as executable workflows

What it provides:



**Autonomous
Workflows:** Multi-
step flows run end-
to-end



Domain Expertise:
Hardware debug,
timing closure,
software
optimization



**Structured
Execution:** Real
commands,
decision trees, no
hallucination



**Actionable
Reports:**
REPORT.md with
real names & fix
commands

Why it matters:

- **Repeatable:** Same skill, same results — eliminates tribal knowledge
- **Shareable:** Version-controlled, portable across teams & orgs

Works with :

VS Code

Cursor

Claude Code

Devin

OpenCode CLI

Any MCP
Compatible Client

Key Components of Ross: Why Skills Help

Skills provide LLMs with subject matter expertise and AMD specific guidelines, and use specific context they can load on demand

Use Cases



Domain Expertise

- Encodes AMD tools methodology & best practices
- Years of expert knowledge in executable form



Repeatable Workflows

- Consistent, predictable results
- Proven step-by-step methodology
- Eliminates trial-and-error guessing



Efficient Token Usage

- Create custom skills for your design patterns
- Share skills across teams and organizations
- Extend agent without code changes



Without Skills, LLMs...

- ↘ Need more prompting
- ↘ Require attached reference files
- ↘ Produce more inconsistent results
- ↘ Consume more tokens
- ↘ Access out of date documentation



Skills Bridge the Methodology Gap

Key Components of Ross: How Skills Work...



File System

Skills stored as
markdown files



SKILL with progressive disclosure of
instructions & resources, e.g., scripts

Metadata: always loaded

```
---
name: csim
description: Run C Simulation to verify kernel correctness and collect loop trip count profiles for synthesis.
allowed-tools: Bash C Simulation , Read
---
```

Instructions: loaded when triggered

```
# Skill: csim

Run C Simulation (csim) to verify the kernel and collect loop trip count profiles.

## When to use

Run csim:
- After modifying `src/*.cpp` or `src/*.h`, to confirm the kernel still produces correct results.
- Before csynth, to generate loop trip count data that improves HLS latency/throughput estimates.
- When debugging: a csim failure (non-zero exit or "FAIL" output) means the kernel logic is broken.

## How to run

```bash
vitis-run --mode hls --config $CONFIG --work_dir $TOP_FUNCTION --csim
```

This runs `vitis-run --mode hls --config $CONFIG --work_dir $TOP_FUNCTION --csim`, which:
```

Resources: loaded as needed

```
## Scripts

Three helper scripts live in `.claude/skills/synth/scripts/`. Run them from the repo root.

### `report_summary.py` - compact synthesis report

```bash
python3 .claude/skills/synth/scripts/report_summary.py [report.rpt]
```



## LLM

Executes expert  
workflows using skill  
instructions

- Generating code
- Running tools
- Validating results

# Key Components of Ross: Vectorized Knowledge Base (RAG)

Real Questions, Expert Answers



AMD public documentation like user guides, product guides, white papers, answer records, etc., that are configured in a Retrieval Augmented Generation (RAG) based vector database, available via cloud or stored on a local server for air-gapped environments



## System Architect

How to configure NoC for optimal bandwidth? 

→ [UG1388](#), [UG1504](#), [PG313](#)

Use traffic classes (Best Effort, Isochronous, Low Latency) with bandwidth allocation per route...



## Timing Closure Specialist

How to fix setup violations on high-fanout nets? 

→ [UG949](#), [UG1387](#), [AR#53430](#)

Use `report_high_fanout_nets` after synthesis. Apply `MAX_FANOUT` or manual replication...



## RTL Developer

RTL coding best practices for synthesis? 

→ [UG901](#), [UG1387](#), [UG949](#)

Follow UG901 guidelines. Use `RAM_STYLE`, `SHREG_EXTRACT`, `MAX_FANOUT` attributes. Avoid latches...



## Hardware Debug Specialist

PDI boot failure troubleshooting on AMD Versal™ devices? 

→ [UG1304](#), [UG908](#), [AR#76655](#)

Check boot mode pins (SW1). Verify PDI in JTAG/flash/SD. Boot via HW Manager. Check PMC error codes...



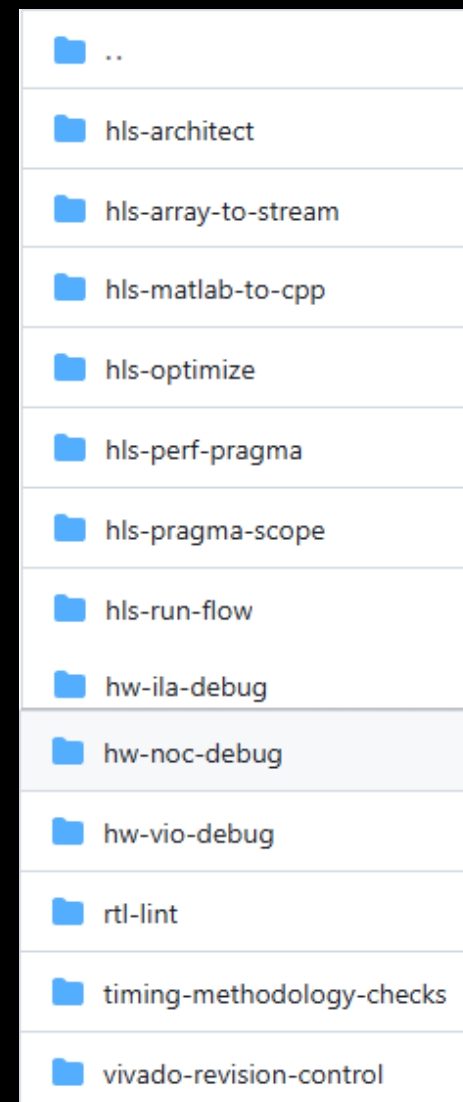
**Powered by RAG** — 500+ User Guides, Application Notes & Knowledge Base Articles

[UG903](#) · [UG904](#) · [UG908](#) · [UG949](#) · [UG1387](#) · [UG1388](#) · [UG1504](#) · [PG313](#) · [DS957](#) · [AM011](#) · [UG901](#) · [AR Articles](#) · and more...

# Key Components of Ross: Design Examples

Ready-to-run designs demonstrating how to use skills in the real-world applications

Accessible through the public AMD GitHub repository, these design examples are self-contained and are arranged by the skill



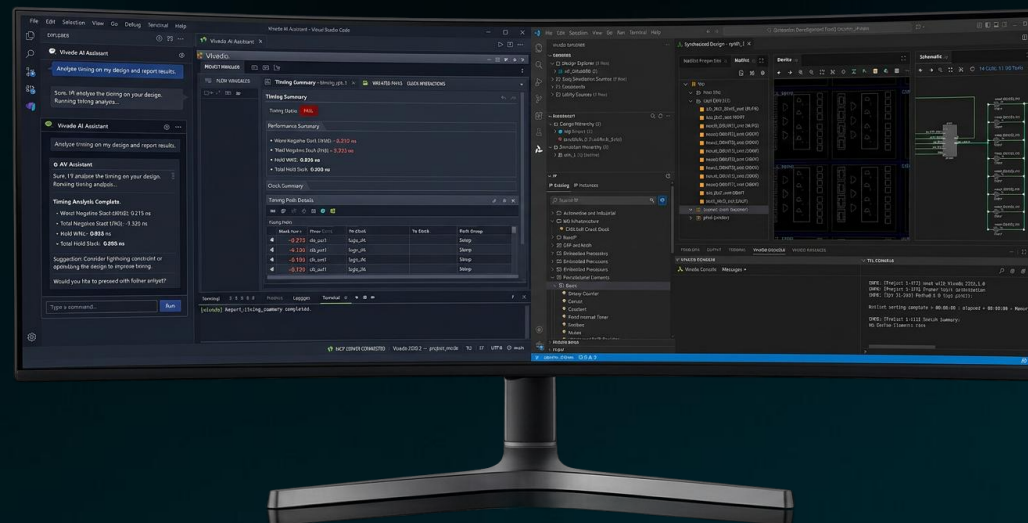
# Working Alongside Developers

Connecting to live projects via CLI of choice

VS Code, Claude Code, Cursor, Devin etc.

## IDE or CLI of Choice

Full project access—no scripting or TCL expertise required



**AMD**  
Ross

## Design Projects

GUI or batch mode execution

**MCP Servers**



Works alongside the live project



User-controlled permissions



Real-time interaction



Parallel agent sessions



# Examples of How AMD Ross™ Helps You with Your Design

---

# How AMD Ross™ Turns Design Intent to Optimized Outcomes

Ross provides one AI-guided entry point across the AMD software portfolio

01. EXPRESS INTENT

02. ROSS ORCHESTRATES

03. AMD TOOLS EXECUTE

04. DESIGN GOALS MET



## INTENT

what you want

Describe the design in natural language, HDL, C++, Python™, or an existing netlist.

**AMD**  
Ross

## AGENTIC AI ASSISTANT

Ross understands the intent, orchestrates the right AMD tools, and applies design expertise.

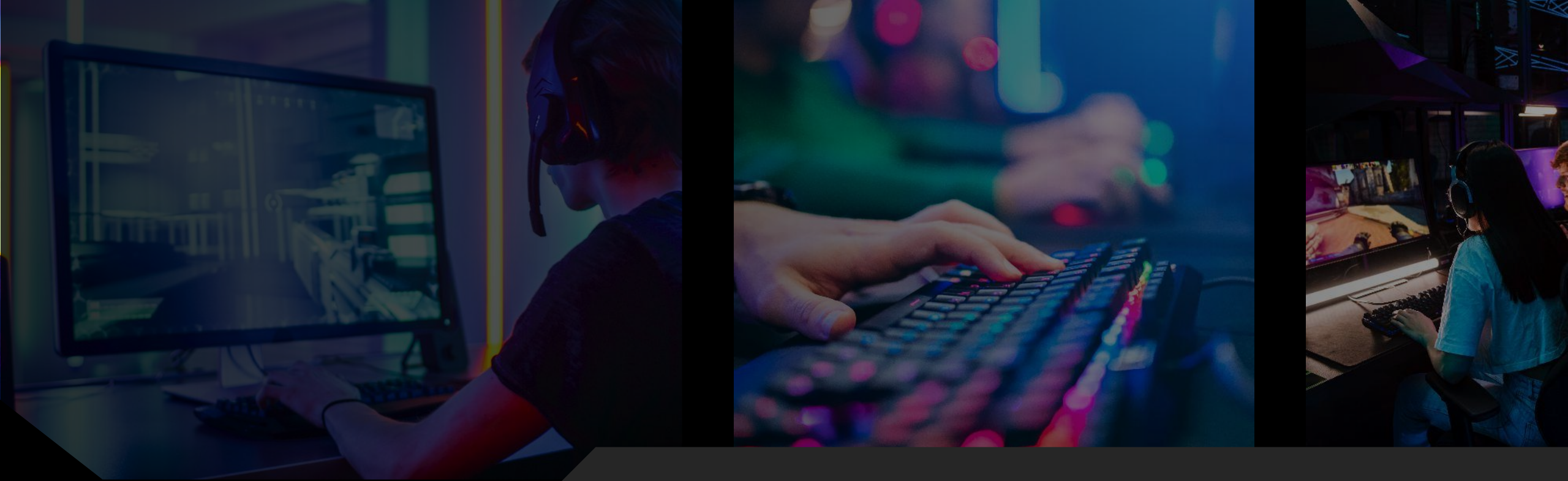
## OUTCOME

ready to implement

Timing-closed RTL, optimized C++, compiled ML models, and debug-ready netlists.



Ross scales AMD design expertise through a unified, AI-guided experience



## THREE EXAMPLES SHOWING HOW ROSS HELPS TEAMS TO DESIGN, IMPLEMENT & OPTIMIZE

---

# AMD Ross™ for Design

# “Create a microcontroller ...”

## Interprets Design Intent

## 01. “Create an AMD MicroBlaze™ V microcontroller for Spartan™ UltraScale+™ Eval Kit”



## Plans and Implements

02. Confirms the board, creates 11-step plan



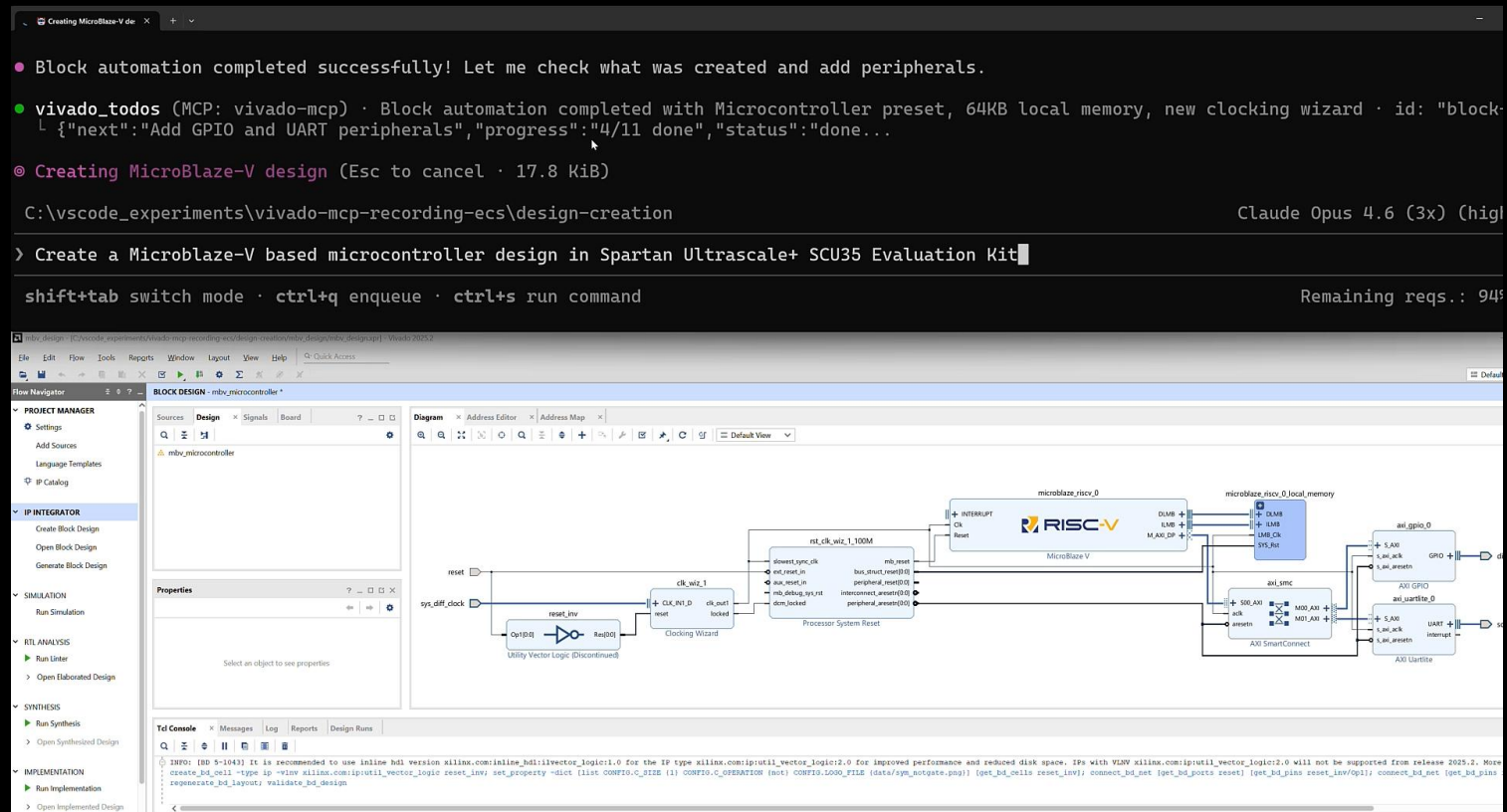
## Adapts

### 03. Resolves block config mismatch & retries



## Configures

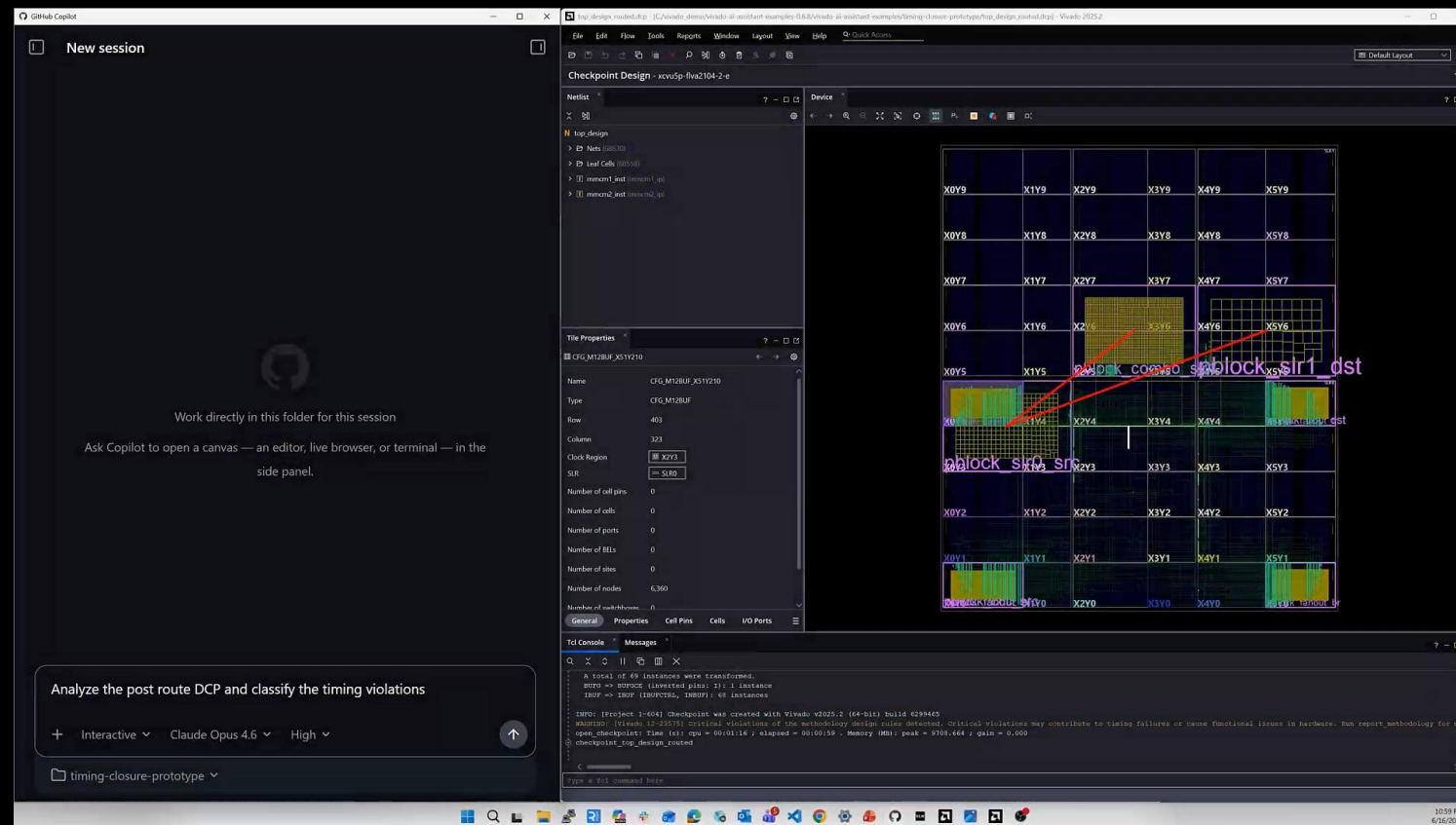
04. Applies microcontroller preset and configures local memory and clocking



# AMD Ross™ for Implementation

“Analyze my timing violations...”

01. **Invokes Timing Skill**  
Starts repeatable post-route flow
02. **Analyzes Failing Paths**
03. **Classifies 735 failures**  
Groups every path by category
04. **Fixes and Verifies**  
735 → 0 failing paths



# AMD Ross™ for Algorithm Optimization

## From 2 Days to 2 Hours at 35% Fewer Logic Resources

### > Application

- Image-Sensor Demosaic Pipeline
- MATLAB Implementation
- Performance Target: 300 Mpixels/s

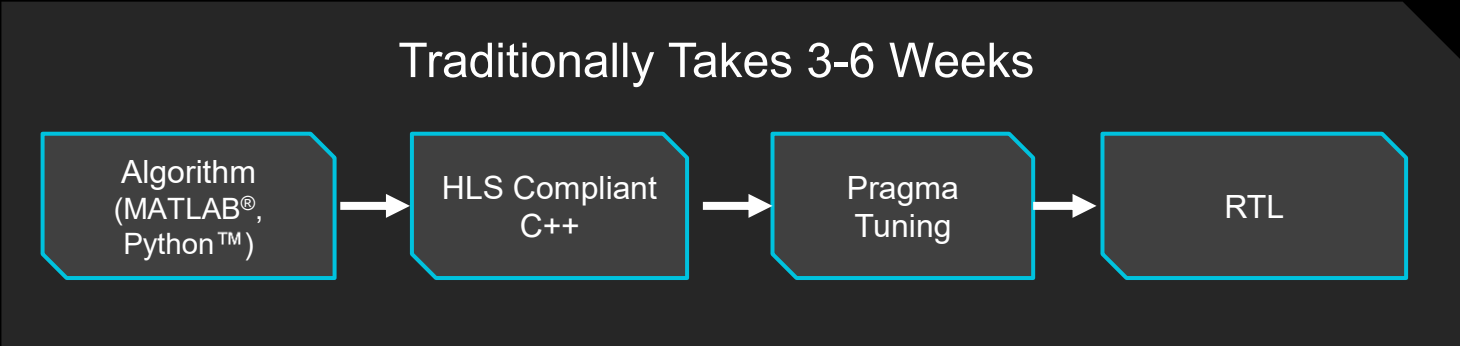
### > Comparison

- Hand-written vs. AMD Ross™ Skill
- AMD Vitis™ HLS agentic workflow

### > Result

- ✓ Performance target met
- ✓ In 2 hours vs. 2 days
- ✓ 35% Fewer Logic Fabric Resources

Metric	Expert-Written C++ for Vitis HLS	AMD™ Ross	Delta
Throughput	300 Mpixels/s	300 Mpixels/s	[equivalent throughput]
Resources	1,801 LUTs	1,177 LUTs	<b>35%</b> Fewer LUTs



Based on AMD internal testing and analysis as of September 2026. Comparison reflects one AMD Ross™ Vitis™ HLS workflow implementation and one expert-written Vitis HLS implementation using the same performance target. Resource utilization and development times are specific to the tested workflow and configuration. Results may vary



# Getting Started & Resources

---

# Let's Get Started . . . Steps



## Prerequisites

- Download, install and license the required AMD tools
- Install an MCP compatible IDE/CLI client
- Ensure access to your preferred LLM

0.



## Download the MCP Server

- Sign in with your AMD credentials
- Download the MCP server of your choice built for your OS
- Configure your IDE/CLI client to connect to MCP

1.



## Get Skills and Examples

- Clone agent skills and example designs from GitHub repository
- Install the skills where your setup expects them and verify with your client IDE/CLI

2.



## Start Your AI Journey

- Follow the design examples end to end
- Iterate with AMD embedded tools in your daily loop

3.



**End state:** a working MCP setup with AMD skills installed and your first agentic design running

# Let's Get Started . . . for VS Code Users Only



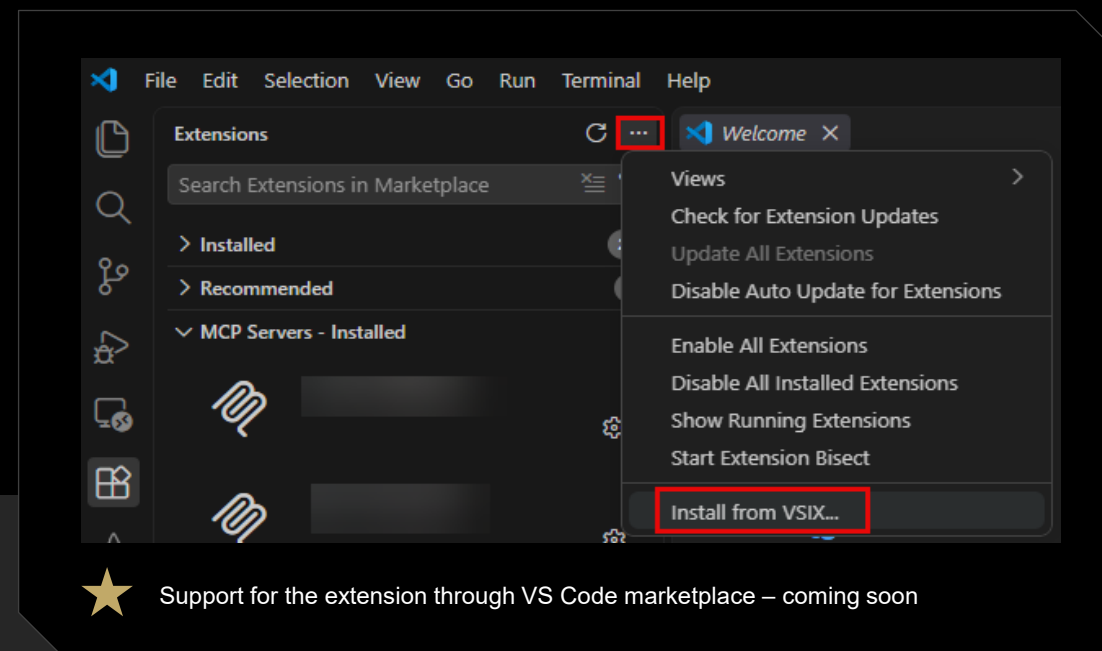
AMD provides the **VS Code extension** that comes with access to the AMD Vivado™ MCP server and RAG-enabled, multi-GB AMD knowledge base



This extension can be downloaded and configured using your VS code setup



Simple, easy to use, single step process



## Download the VS Code Extension

- Sign-in with your AMD credentials
- Download the Vivado AI VS Code extension & configure
- Download and install the agent skills and example designs from the GitHub repository

# Air-Gapped, Local Deployment Options

## For Security-Sensitive or Cost-Conscious Team Environments



Run AMD Ross™ knowledge base and MCPs locally without relying on the cloud



Keep prompts, project context, and design data within controlled infrastructure



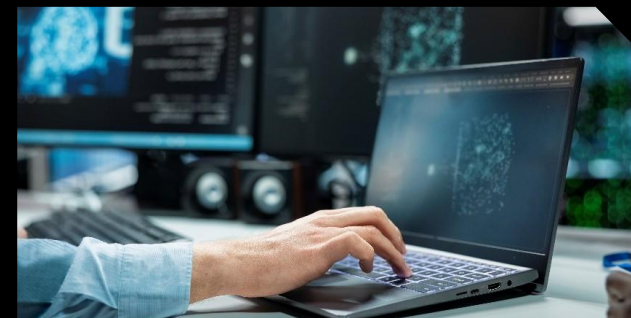
### IT Security Policies

Keep sensitive data within the enterprise



### Restricted Environments

Work where cloud access is limited



### Control Inference Costs

Avoid recurring cloud API and token fees

# Let's Get Started . . . for Users That Need a 'Air-Gapped' Solution



For users with no internet connection (air-gapped machine) who want to access the AMD knowledge base



Deploy the database once with the embedding model, vector search, and MCP server, then use it with any answer-generation model that fits your hardware and network policy



Modular, replaceable, portable solution

## STEPS INVOLVED ARE:

### Setup the foundation:

Install and run the docker image to setup the RAG-enabled knowledge base



### Use the frontier model:

Only the LLM call leaves the machine, while documentation retrieval stays local



### Use the local model:

Both LLM call and documentation retrieval are local



Note: Users need to get their own preferred LLM.

# Availability and Phased Rollout

Knowledge-Base, MCP Integration, Skills, and GitHub Repository

## Available Today

AMD Knowledge Base → RAG-Powered Access

FPGA Design → AMD Vivado™ Design Suite

Algorithm Implementation → AMD Vitis™ HLS

## Available in Q4

Power Estimation → Power Design Manager

FPGA Debug → ChipScope

AI Inference → AMD Vitis™ AI

Additional Tools & Flow Capabilities Rolling Out on a Monthly Cadence

# Early Customers Are Seeing Results

## 800+ Customers Participating in Early Access



*“The AMD Ross™ Agentic AI Assistant has been a valuable addition to our development workflow. It helps us identify and troubleshoot issues more efficiently, reducing the time and effort required during the debugging and bring-up stages.”*

### Top-Ranked AMD Ross™ Capabilities By Users

Analyze  
synthesis,  
implementation  
& timing results

Search official  
AMD  
documentation

Automate  
repetitive  
Vivado actions

Set up projects  
or configure IP

Find next steps  
after a failed  
run

Source: AMD Ross Early Access post-usage survey; Number of users who ranked capability “Extremely Useful” or “Very Useful”; 34 completed responses.

# General Disclaimer and Attribution Statement 2026

The information contained herein is for informational purposes only and is subject to change without notice. While every precaution has been taken in the preparation of this document, it may contain technical inaccuracies, omissions and typographical errors, and AMD is under no obligation to update or otherwise correct this information. Advanced Micro Devices, Inc. makes no representations or warranties with respect to the accuracy or completeness of the contents of this document, and assumes no liability of any kind, including the implied warranties of noninfringement, merchantability or fitness for particular purposes, with respect to the operation or use of AMD hardware, software or other products described herein. No license, including implied or arising by estoppel, to any intellectual property rights is granted by this document. Terms and limitations applicable to the purchase or use of AMD products are as set forth in a signed agreement between the parties or in AMD's Standard Terms and Conditions of Sale. GD-18u.

©2026 Advanced Micro Devices, Inc. All rights reserved. AMD, the AMD Arrow logo, ROCm, Ross, Ryzen, Vitis, Vivado, and combinations thereof are trademarks of Advanced Micro Devices, Inc. Other product names used in this publication are for identification purposes only and may be trademarks of their respective owners. Certain AMD technologies may require third-party enablement or activation. Supported features may vary by operating system. Please confirm with the system manufacturer for specific features. No technology or product can be completely secure.

