Week 7

Codefest: Rethinking, reorienting, scaling back?

ECE 410/510
Spring 2025

Challenge #23

Overview and context:
Many of you started off with a project that was perhaps a little too ambitious. You may now realize that things won't exactly lead to success unless you change your approach. That's a great insight to have and part of science, engineering, and life, where things rarely are a straight line.

Learning goals:

- Remind yourself of the project requirements and how success is defined.
- Learn how to rethink, reorient, and scale back your project so you can declare success at the end.
- Apply co-design principles in doing so.
- Rethink and refine your LLM prompt engineering

Project requirements [REMINDER]:
See also https://docs.google.com/document/d/1_HSDXhJEF1F2Qu76zJOlygvPm_F9NqX5VZ0f8CGXI34

Goals:

1. Design, test, and benchmark a co-processor chiplet that accelerates parts of some AI/ML code/algorithm of your choice. Start with a blank slate for your design.
2. [ALTERNATIVE] Design a stand-alone chiplet. Your chiplet could, for example, include an ARM core.
3. You will be deciding what design constraints to impose (e.g., power budget) and what metrics to use (e.g., throughput).
4. Apply HW/SW co-design principles to decide what part of your algorithm is executed in HW.
5. The chiplet should be described in a HW-description language (e.g., Verilog). The design should be synthesizable in order to obtain your relevant metrics.
6. The deeper you can go into the HW design, the better. The ultimate goal is to go all the way down to an ASIC description. E.g., by using OpenLane 2's workflow to convert HDL into GDS. See codefest assignments for additional info, tools, hints, etc.

How is project success defined?

- You successfully benchmarked your SW algorithm and identified the bottleneck(s).
- You designed, built, and tested (in software, e.g., Verilog) a custom HW accelerator chiplet to remove the bottlenecks.
- You synthesized your HW design down to the ASIC/transistor level and obtained relevant performance metrics, e.g., max. frequency, number of transistors, etc.
- You applied co-design and an iterative approach to improve your design throughout the term.
- You evaluated and benchmarked your entire system, i.e., SW + HW + communication in-between and provided the evidence that your HW accelerator indeed accelerates the initial SW version.

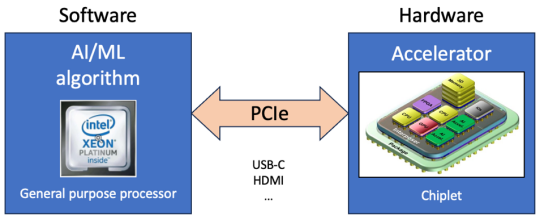

Figure 1. Applying a system-level co-design approach where the SW algorithm informs the HW.

Rethink your prompt engineering:

Core Principles

- **Be specific about the design stage** - Specify whether you need help with architecture exploration, RTL design, verification, physical design, or post-silicon validation
- **Provide technical context** - Include details about technology node, design constraints, and performance targets
- **Use domain-specific terminology** - Utilize standard chip design terminology to ensure clarity (HDL, DRC, timing constraints, etc.)

Effective Strategies

1. **Break down complex tasks** - Instead of asking for an entire chip design, focus on specific modules or functional blocks
2. **Specify design constraints clearly** - Include power, performance, area (PPA) requirements and other key parameters
3. **Request step-by-step approaches** - For complex design problems, ask for methodical solutions
4. **Leverage visual aids** - Request diagrams or suggest formats for outputs (block diagrams, timing diagrams, etc.)
5. **Create test cases and test benches** - For any design solution, systematically create test cases and test benches.

Example Prompts

- **Poor prompt:** "Design a chip for me."
- **Better prompt:** "Help me create an architectural block diagram for a low-power IoT SoC in 28nm technology with the following specifications: CPU core (ARM Cortex-M4), 256KB SRAM, Bluetooth LE radio, and temperature/humidity sensors. Target power consumption should be under 10mW in active mode."
- **Poor prompt:** "Check if my design works."
- **Better prompt:** "Review my SystemVerilog testbench approach for verifying a RISC-V processor cache controller. I need to validate coherence protocols across multiple cache levels. What assertions should I include to verify proper handling of modified cache lines during eviction? Please suggest specific SVA properties I could implement."
- **Poor prompt:** "How do I make my chip use less power?"
- **Better prompt:** "I'm designing a 7nm mobile SoC with dynamic voltage and frequency scaling. My current power analysis shows excessive leakage in the memory subsystem (approximately 25mW at idle). Suggest three specific techniques to reduce leakage power without compromising access time. Include both circuit-level and architectural approaches."
- **Poor prompt:** "Help with my chip layout."
- **Better prompt:** "I'm working on floorplanning a 5nm FinFET design with 12 processing cores and an on-chip network. My current approach places memory blocks at the periphery, but I'm seeing timing violations on critical paths. Considering IR drop and thermal constraints, what alternative floorplan organization would you recommend? Please explain the tradeoffs with respect to timing closure and signal integrity."
- **Poor prompt:** "Write Verilog code for a processor."
- **Better prompt:** "Help me implement a parameterized FIFO buffer in SystemVerilog with configurable depth and width. I need support for almost-full and almost-empty flags, with threshold values set at compile time. The design will be synthesized for an FPGA target (Xilinx Ultrascale+), so please optimize for resource utilization rather than timing. Include comments explaining synchronization mechanisms."
- **Poor prompt:** "Design an amplifier."
- **Better prompt:** "I need to design a low-noise amplifier for a 5G RF front-end operating at 28GHz in a 22nm FD-SOI process. Target noise figure is <2dB with IIP3 >10dBm while consuming less than 15mW. Suggest a circuit topology that meets these requirements, explaining key design considerations for biasing and matching networks. Include typical component values."