Arm Sophia Internships 2023

1. Duration: 6 months
2. For students in their final year at university or engineering school, in electronics or computer science
3. Internship location is Sophia-Antipolis, France
4. Apply on careers.arm.com/ (keyword "Internship" and location "France")

As we are shifting in a new era of AI, integrating our engineering teams in Sophia Antipolis will offer you the excitements of shaping the Future Wave of Computing that will be fundamental in making the world we live in, more efficient and more sustainable.

In the next pages, the internship subjects for this season are listed per team:

- CPU Microarchitecture & Design
- CPU Physical Implementation
- CPU Verification
- CPU Microarchitecture Modelling & Performance Analysis
- Physical Design Group

CPU Microarchitecture & Design

To apply for this offer:

- Head to https://careers.arm.com/.
- Search for offers with keyword “Internship” and location “France”
- Click on “2023 Engineering Internship – CPU Microarchitecture and Design”

[2023-D1] Efficient support of complex access patterns of Scalable Matrix Extension

Data prefetching is key to harness high-throughput execution engine. Some patterns found in SME, are found to be challenging. The goal of this internship is to investigate and implement strategy(ies) to improve how those complex patterns are handled.

In this internship, you will focus on either RTL or model. For the former, you should expect:

1. Prepare the test set by selecting relevant workloads
2. Devise modifications to the current RTL expected to improve performance on the test set
3. Implement those on actual RTL keeping in mind Performance, Power and Area (PPA)

For the latter, you should expect:

1. Prepare the test set by selecting relevant workloads
2. Devise various modifications to the model expected to improve performance on the test set
3. Test those in the model keeping in mind RTL feasibility

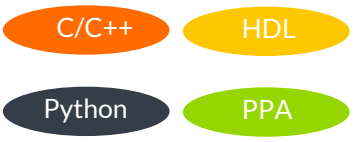
[2023-D2] Efficiency Verification and Improvements of Performance Driven Power Management features at Core Block Level

Performance and Power Management features are important to control CPU PPA aggressiveness.

The goal of this internship is to improve development methods by verifying efficiencies of such techniques and gain confidence to better choose their evolution.

In this internship you will:

1. Gain knowledge of such features
2. Investigate benchmark results and identify more interesting ones
3. Change existing environment to exercise these features and propose design evolution in the end



[2023-D3] Hardware Stack protection mechanism

Return oriented programming (ROP) is a common method used by attackers to execute malicious code and bypass security defences. The goal of this internship is to implement a hardware mechanisms to record and check the return addresses in a superscalar processor.

During this internship, you will:

1. Study the existing behaviour of different processors
2. Model a solution proposed by architects to automatically protect this Call Stack
3. design and test a hardware implementation of this solution

[2023-D4] Implement Register Bank prefetching techniques

High performance processors often support register caching to reduce power consumption of the Register File's read ports and hide read latency. In case of misprediction these caches are sometimes full of useless data. The goal of this internship is to study and implement techniques to have those caches always filled with useful and updated data.

In this internship you will:

1. Study the issue queue module and register caching mechanisms in a state-of-art microarchitecture
2. Implement in RTL different register caching techniques
3. Analyse performance and PPA

[2023-D5] L2 Translation Lookaside Buffer Clustering improvements

Translation lookaside buffers (TLB) are a key component of modern CPU designs. With the steadily increasing sizes of workloads' memory footprint, the processor's TLB must be able to keep up to avoid being the performance bottleneck. The goal of this internship is therefore to study optimization techniques allowing to increase the reach of processor's TLB in different system configurations.

During the internship, you will:

1. Study state of the art of the TLB, discover advanced techniques that allow increasing its reach, such as TLB clustering
2. Propose new solutions to address these limitations
3. Implement and verify the impact of new solution in a CPU model
4. Implement the best solutions in RTL and analyse performance and PPA

CPU Physical Implementation

To apply for this offer:

- Head to https://careers.arm.com/.
- Search for offers with keyword “Internship” and location “France”
- Click on “2023 Engineering Internship – CPU Physical Implementation”

[2023-I1] CPU Memory BIST Analysis Investigation

Testability is key to get working silicon, especially on memory. Optimal memory testability is usually obtained thanks to Memory Built In Self Test (MBIST).

This internship aims at evaluating impact of MBIST insertion on a real CPU

- You will first discover a standard CPU implementation flow from synthesis to signoff
- then you will ramp up on MBIST concepts and understand our MBIST insertion flow
- Once MBIST inserted, you will have to extract timing, area, and power both in functional mode and in MBIST mode, including IRdrop analysis
- a comparative report with and without MBIST will have to be produced
- Depending on progress though the internship, a comparison with a different MBIST implementation will be considered.

Curiosity, Proactivity, Organisation, Basic understanding of HDL will be needed for this internship

[2023-I2] Develop a Hierarchical Cloning Implementation Flow

Time To Market is a key aspect of CPU architecture implementation. Hierarchical implementation is becoming a mandatory methodology to achieve acceptable TTM. Besides, al lot of sub designs are instantiated multiple times in almost all CPU architectures.

The Goal of this internship is to specify and develop a cloning capability in our hierarchical top-bottom implementation flow to reduce implementation run time even more on design architecture where the number of multiple instantiated modules is significative.

After becoming familiar with the hierarchical implementation flow and cloning capability, you will have to enable cloning features provided by EDA providers and insure that PPA targets are met and comparable to a flat implementation methodology. Finally, you will demonstrate the flexibility and the portability of the cloning mechanism from one design to another.

Earlier exposure to implementation tools, flow aspects and PPA concerns would be a plus.

[2023-I3] Investigate ML for CPU Power and Performance Improvement

Every new generation of CPU is reaching new performance heights thanks to micro-architecture but is also closely linked to EDA tool capability.

The aim of this internship is to employs artificial intelligence technology EDA tool capability to identify key parameters and provide High Performance and Power efficient CPU implementation models for advanced technology node.

You will learn about the existing physical implementation flow, develop new features based on artificial intelligence EDA capability, generate High Performance and Power efficient CPUs models, test your solution on current advanced technology node and evaluated results.

Knowledge of CPUs micro-architecture and TCL languages would be a plus.

[2023-I4] Flexible Emulator Unit Power Environment Development

Power estimation is a key component of the development process for efficient CPU design but is usually slow or non-representation of real-life stress scenarios.

The goal of this internship is to bring-up a unit-level power-analysis platform allowing fast-turnaround on industry standard benchmarks that will be used as feedback in early design investigations and exploration.

You will have to get familiar with the existing power-analysis infrastructure and improve it to enable sub-unit level simulation. To improve resource usage, you will have to make the most out of the reports generated and identify redundant simulations. Optionally, to spread the use of the platform, you may investigate CI integration to make deployment faster and more reliable.

You will mainly develop in Python and TCL. A high-level understanding of CPU micro-architecture is required. Basic knowledge of (System)Verilog, SQL and CI tools is a plus.

CPU Verification

To apply for this offer:

- Head to https://careers.arm.com/.
- Search for offers with keyword “Internship” and location “France”
- Click on “2023 Engineering Internship – CPU Verification”

[2023-V1] Improve ARM capacity to detect and debug CPU defects on FPGA / Silicon

RIS tools, which are verification software generating embedded smart stressful stressing self-checking tests, are an essential step of a CPU verification process. They are usually adopted before the CPU is physically implemented, but they can still find hardware bugs on manufactured silicon.

GenASM-MP is a Top-Level RIS tool capable of generating and running real assembly code on numerous testing platforms including simulation, FPGA and even silicon.

FPGA and Silicon allow to run more cycles in a shorter time span. But it is difficult to debug fails on these platforms because easily exploitable debug tools are not available or can be extremely slow.

The goal of this internship is to increase the added value of FPGA and Silicon as test platforms by developing an innovative embedded debug-ability solution. While also optimizing stress performance of the CPU verification tool.

In this internship you will:

1. Study the Arm RIS tool and its usage for FPGA/Silicon verification
2. Discuss with the internship tutor about means to improve capacity to detect bugs and improve debug-ability
3. Implement the selected solution
4. Test them for CPU verification

[2023-V2] Optimization of data structure consumed by a Machine Learning algorithm

Verification is an important part of CPU design, and a Machine Learning flow has been implemented to increase stimulus quality of tests launched.

The goal of the internship is to reshape the data structures used to train the model of this algorithm. This will allow for a quicker and more efficient model creation.

In this internship you will:

1. Analyse the existing data structure
2. Interact with the Machine Learning team
3. Suggest solutions to improve the data structure
4. Implement modifications chosen with the internship tutor

[2023-V3] Learning process to select relevant subset of parameters

In theory, the verification process should generate all scenarios to totally verify the CPU behaviour.

In reality, physical limitations do not allow to do that. The verification team tries to cleverly select configurations to generate representative subsets of scenario.

The goal of this internship is to develop a process to extract a subset of relevant parameters which impact given metrics.

In this internship you will:

1. Study how the parameters given to a test impacts the verification
2. Search in IEEE different implementations of machine learning process (decision tree, weight …)
3. Implement a first version of the solution demonstrating its potential
4. Evaluate the script performance (time, memory, reliability) for scaling
5. End goal: generate lists of configurations that finds bugs consistently

Python HDL

[2023-V4] FPGA implementation of a CPU

The CPU Sophia FPGA team is developing innovative and efficient Tools and Methods to debug CPU on FPGA along with the prototyping of next-gen CPU FPGA image.

The goal of this internship to synthesize a CPU reference design onto an FPGA. Then use this image to demonstrate performance improvements and usability.

In this internship you will:

1. Familiarize yourself with CPU verification on FPGA
2. Study a CPU RTL code to be synthesized on FPGA
3. Work on the synthesis of the RTL and generate re-usable images
4. Analyse runtime metrics to validate functionality

FPGA **Scripting**

[2023-V5] Core testbench boot sequence optimization

CPU verification simulation time is expensive and needs to be used efficiently. Instead of executing the same boot sequence of instructions on every test we could use other ways to configure the CPU core.

The goal of this internship is to find and implement new configuration methods of the CPU registers.

In this internship you will:

1. Study CPU core verification environment
2. Understand how to read/write system registers in simulation
3. Find solutions to initialize the core registers
4. Test your solution and validate the performance improvements

Python C/C++

[2023-V6] ISA-Formal verification applied to ARMv9-A SME

ISA-Formal verification is a technique using formal verification and Architecture specification to ensure the design is working as it should.

The goal of this internship is to write a pipeline follower in Verilog and use ISA-Formal methodologies to verify instructions. The verification method will be applied on ARMv9-A Scalable Matrix Extension (SME) feature.

In this internship you will:

1. Study formal verification methodologies
2. Familiarize with ISA-Formal verification
3. Develop a pipeline follower in Verilog
4. Use it to verify SME instructions

[2023-V7] Formal verification of PMU events

Performance monitors (PMU) gather various statistics on the operation of the core during runtime.

The goal of this internship is to apply formal verification techniques to PMU events.

In this internship you will:

1. Study performance monitors usage
2. Familiarize yourself with formal verification techniques
3. Apply formal verification to specific PMU events
4. Prove that the formal verification of the PMUs has improved the code quality.

CPU Microarchitecture Modelling & Performance Analysis

To apply for this offer:

- Head to https://careers.arm.com/.
- Search for offers with keyword “Internship” and location “France”
- Click on “2023 Internship - CPU Performance Analysis and Modelling”

[2023-P1] CPU Microarchitecture modelling and exploration – Temporal ICache Prefetching

Prefetching techniques are key techniques to improve the hit rates in CPU caches, leading to significant performance improvements. The goal of this internship is to study a specific type of prefetching, Temporal Correlation, and apply it to the Instruction side.

In this internship you will:

1. Start with a bibliography study on state-of-the-art Temporal Prefetching and Instruction Cache prefetching
2. Become familiar with an Arm internal performance / micro-architectural model (written in C) and the overall simulation environment
3. Identify and implement the most promising solution in the model
4. Run performance studies and incrementally optimise the solution.

C/C++ **CPU uarch**

[2022-P2] CPU Microarchitecture – Dead Blocks predictors

Cache memories are key elements in modern CPU. Maximizing their usage and efficiency is critical to achieve the best possible performance.
The goal of this internship is to study the unnecessary blocks (“dead blocks”) stored in the caches and find/implement some mechanisms that can improve the cache efficiency by getting rid of such blocks.

In this internship you will:

1. Start with a bibliography study on state-of-the-art Dead Block prediction
2. Become familiar with an Arm internal performance / micro-architectural model (written in C) and the overall simulation environment
3. Identify and implement the most promising solution in the model
4. Run performance studies and incrementally optimise the solution.

C/C++ **CPU uarch**

[2022-P3] CPU design exploration assisted by genetic algorithms

The number of interdependent behaviours in a modern CPU design is exploding, especially in the key domain of data prefetchers.

The goal of this internship is to create a tool that explores the design space by varying different parameters available in an in-house CPU micro-architectural model, and analysing the results, improving the efficiency of performance exploration.

In this internship you will:

1. Create a tool to exploit the high configurability of an existing CPU microarchitectural model
2. Automate the runs, gather and synthesize the results into Performance-Power-Area data points, and register them in a database
3. Use Genetic Algorithms or Machine learning to analyze results and optimize CPU designs

Python **Data Science**

Physical Design Group

To apply for this offer:

- Head to https://careers.arm.com/.
- Search for offers with keyword “Internship” and location “France”
- Click on “2023 Engineering Internship – Physical Design Group” or “2023 Engineering Internship – APD Physical Design”

[2023-PDG1] Metal Insulator Metal compiler

With the increasing current budget required by multi core processors, traditional strategies are not sufficient any longer to tackle the IR drop problem. Metal Insulator Metal capacitances, or MIMCAP, are a valid solution for this issue, acting as a decoupling capacitance at no area loss. The goal of this internship is to lay the groundworks to realize a MIMCAP compiler, a software tool capable of automatically generating MIMCAP tiles for any given technology node and target floorplan.

In this internship you will:

1. Analyse how the physical layout of a MIMCAP tile affects its Effective Series Capacitance (or ESC), its Effective Series Resistance (or ESR) and its frequency response.
2. Understand how to optimize a MIMCAP tile to improve its capacitance and frequency response.
3. Design layout generation algorithms to automatise the layout optimization of MIMCAP tiles and generate all the required model views for selected tiles.
4. Run a prototypal automatic generation of MIMCAP tiles for a test Arm core and analyse their effectiveness in dampening the IR drop.

[2023-PDG2] High performance Physical Implementation

Meeting Area, Frequency, and Power requirements with a reasonable runtime is one of most meaningful activities we encounter when implementing physical IPs, CPU and GPU cores. As part of the Advanced Physical Design implementation team, you will be given ownership of an investigation topic directly related to physical implementation performance push. You will gain the opportunity to learn how implement high performance processors.

In this internship you will:

1. Analyse timing correlation between the main physical implementation steps (from standard cells placement to detail routing)
2. Correlate critical timing paths between static timing analysis and spice simulation
3. Improve physical implementation setup and recipes to improve timing convergence
4. Quantify the impact of improved timing convergence on max achievable frequency