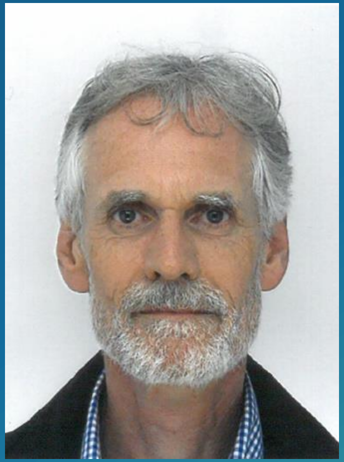
# System on Chip design for AI/ML ASICs

soclabs.org

John Darlington

Daniel Newbrook

David Flynn



## about this session

- **about:** SoC Labs, … now and going forward
- **flows:** building an ASIC
- **example flows:** idea -> fpga -> custom ASIC
- **technology:** design references
- **worked examples**
- **projects:** from initial open call design contest
- **get involved**

## Caveats/Perspective

- Most of the information in this talk is for someone who:
  - Wants to build a hardware accelerator
  - Wants to know more about how this fits within a system
- Coming from someone who is a System on Chip designer working with the above people
- This is mostly an overview but please see me or contact us if you want more information or to get involved

## about this session

- **about:** SoC Labs, … now and going forward
- **flows:** building an ASIC
- **example flows:** idea -> fpga -> custom ASIC
- **technology:** design references
- **worked examples**
- **projects:** from initial open call design contest
- **get involved**

# about SoC Labs

global academic community for System On Chip using arm architecture

- innovative ways to share, experience, knowledge and design re-use
- raise skill levels and electronic design practice within academia
- utilise both open and licensed IP to maximise research impact
- expand number of academics/institutions that produce SoCs
- improve academic output, more academic SoC designs in tier 1 publications

supported by arm, EDA vendors and Semiconductor Education Alliance

# arm ACADEMIC ACCESS

- Improve academic **design talent** by working on real world System on Chip solutions and challenges
- Accelerate **time to results** and providing the opportunity to build research around real-world commercial IP
- Make the **path to impact** less challenging by using Arm IP, the world's largest ecosystem

AAA offers our widest range of IP and tools:

- A membership model with access to an IP package on an ongoing basis via a standard membership agreement
- Free to join and has no licensing or royalty fees
- for research, education and training purposes
- Visit arm.com/academicaccess

**30+ years**
Experience in the industry

**1000+**
**Technology partners**
Industry leaders and high-growth start-ups; chip companies and OEMs

**280+bn**
Arm-based chips shipped to-date

**120+**
**Institutions and growing**
Worldwide

**11472+**
Arm IP delivered to academic institutions up to Jan 2024

arm

## about SoC Labs

community centric hardware design

- greater innovation/impact/scale than working in isolation
- less effort on repeating basics, more on unique research IP
- together we solve problems and learn faster
- create ‘centers of gravity’ around reusable designs and assets, eg. NanoSoC
- benefit from shared resources especially **verification** efforts
- community projects motivate seasoned academics and new students

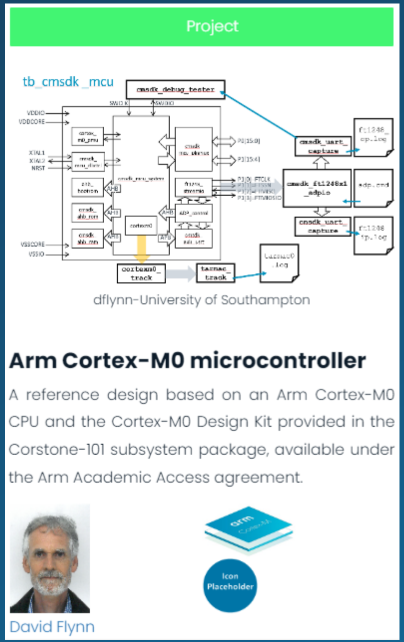
# about - soclabs.org

**global academic community**

**mutual support + collaboration**

UNIVERSITY OF PATRAS
ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΑΤΡΩΝ
UNIVERSITY OF BATH
부산대학교
PUSAN NATIONAL UNIVERSITY

**projects help ease SoC design, bridge knowledge gaps**


Project

tb_cmsdk_mcu

dflynn-University of Southampton

**Arm Cortex-M0 microcontroller**

A reference design based on an Arm Cortex-M0 CPU and the Cortex-M0 Design Kit provided in the Corstone-101 subsystem package, available under the Arm Academic Access agreement.

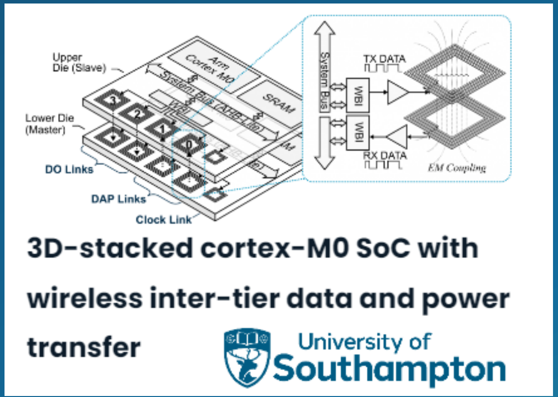
David Flynn

arm

Arm Academic Access

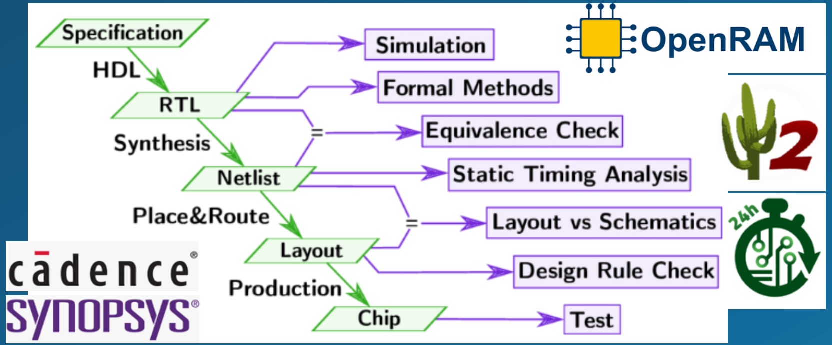
Increase research impact with free academic access to the world's leading SoC design portfolio

**arm IP + academic IP**



**3D-stacked cortex-M0 SoC with wireless inter-tier data and power transfer**

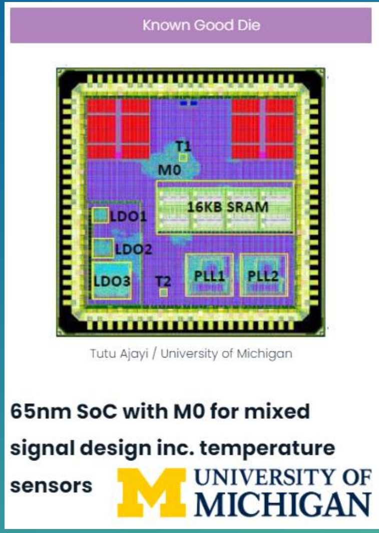
University of Southampton



**use industrial and open source EDA tools**

**more academic die in tier 1 publications**

**more researchers and students with proven skills**


Known Good Die

Tutu Ajayi / University of Michigan

65nm SoC with M0 for mixed signal design inc. temperature sensors

UNIVERSITY OF MICHIGAN

# about this session

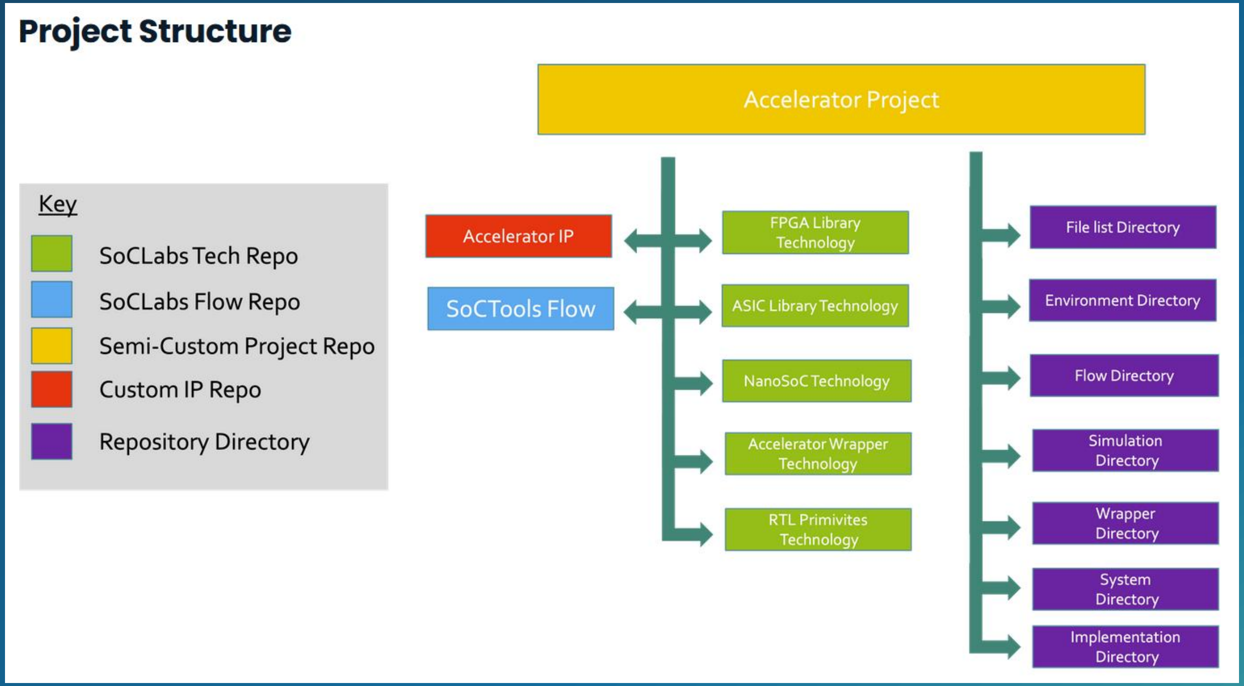
- **about:** SoC Labs, … now and going forward
- **flows:** building an ASIC
- **example flows:** idea -> fpga -> custom ASIC
- **technology:** design references
- **worked examples**
- **projects:** from initial open call design contest
- **get involved**

# design flows

- SoC labs site contains information on different stages of design flows including some example flows
- use of generic, high-level flow steps to get a sense for how to achieve each task in the SoC design life cycle
- as well as some tool specific flows
- currently based around digital SoC design
- encouraging community to add additional knowledge

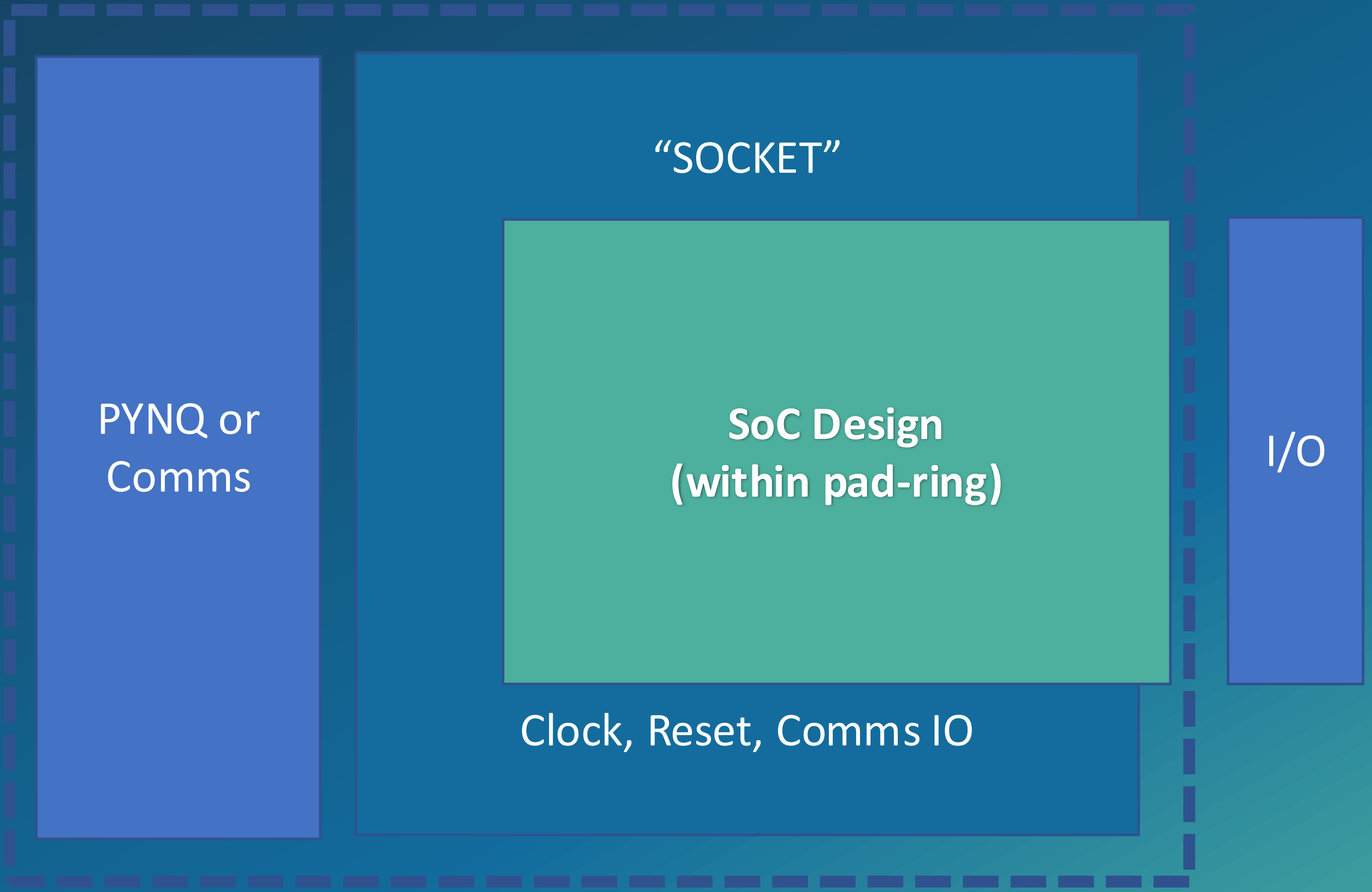
# project structure/flow

- maintaining organised **project** is key to:
  - a successful SoC scale project
  - enables efficient reuse of **technology** IP, scripts/environment setup, etc.
  - Supports collaborative working
  - mimics industry best practise
- project management can include milestones that correspond to **design flow** steps



# FPGA prototyping flows

- similar structure for both Pynq environment and bare-metal fpga (like ARM MPS3)
- use either
  - Xilinx PS/Pynq environment
  - Or direct comms over UART
- design instantiated in pad-ring-level "socket"
- IO ports mapped to board

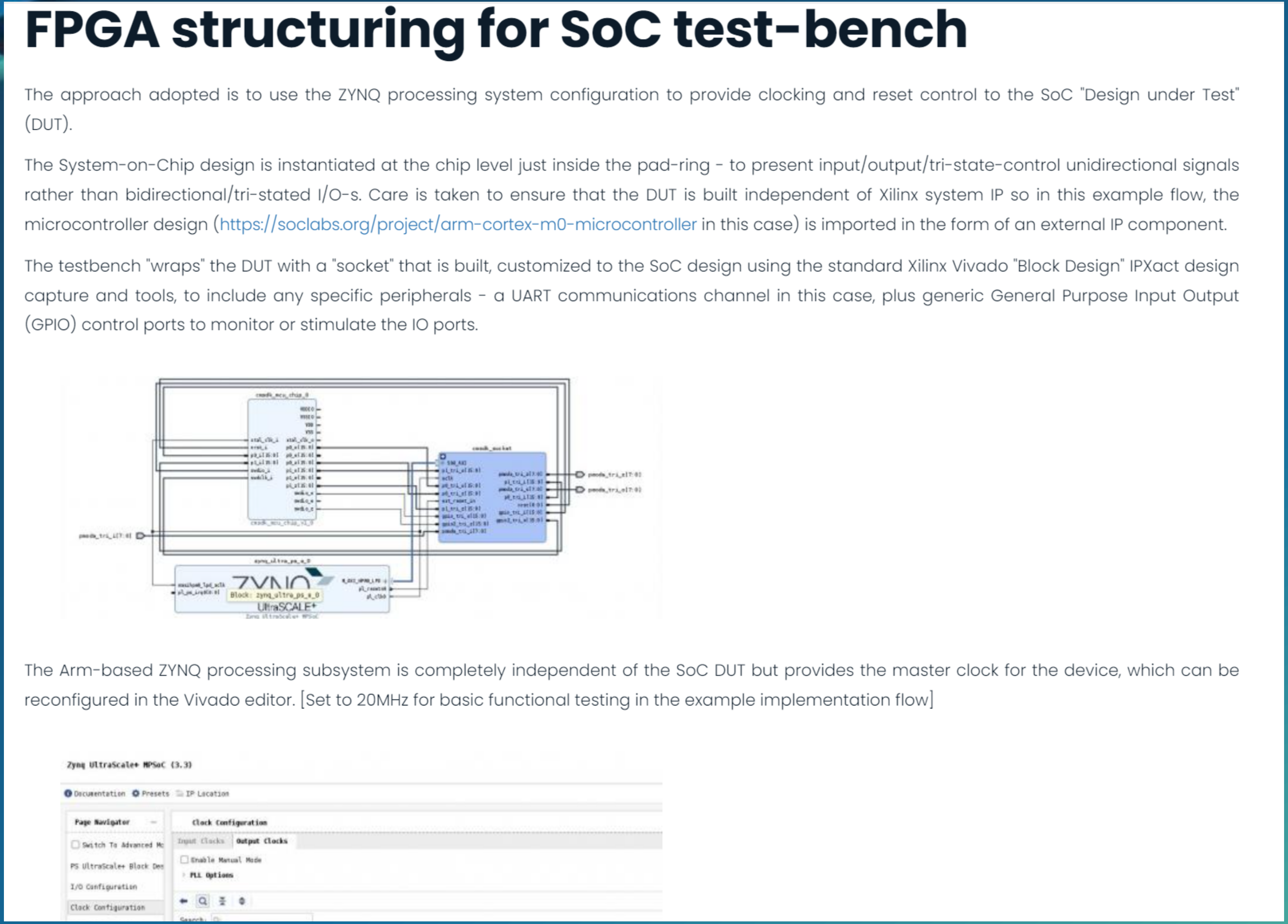


soc labs

the global community for arm-based projects

# FPGA prototyping flows

- design-flow material actively in development
  - Xilinx ZCU104 PYNQ
  - Arm MPS3 systems
  - Xilinx PYNQ Z2
  - *NEW* Kria K26 targets
    - ($250-350 Xilinx systems)
- server-based resource example for shared board targets

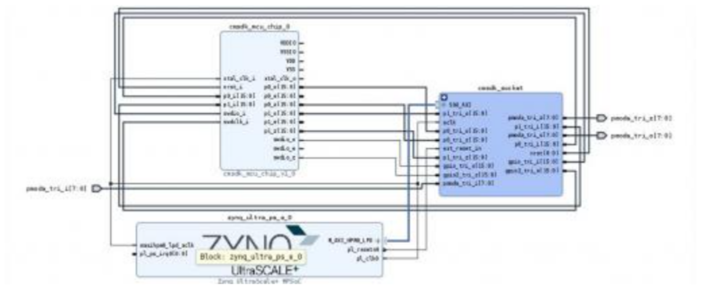


## FPGA structuring for SoC test-bench

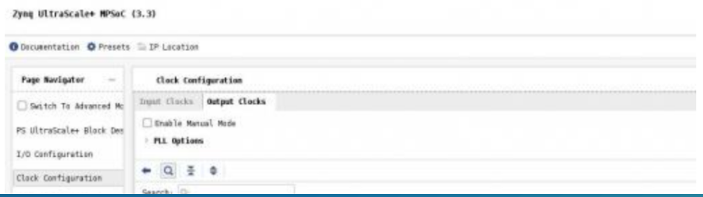
The approach adopted is to use the ZYNQ processing system configuration to provide clocking and reset control to the SoC "Design under Test" (DUT).

The System-on-Chip design is instantiated at the chip level just inside the pad-ring - to present input/output/tri-state-control unidirectional signals rather than bidirectional/tri-stated I/O-s. Care is taken to ensure that the DUT is built independent of Xilinx system IP so in this example flow, the microcontroller design (https://soclabs.org/project/arm-cortex-m0-microcontroller in this case) is imported in the form of an external IP component.

The testbench "wraps" the DUT with a "socket" that is built, customized to the SoC design using the standard Xilinx Vivado "Block Design" IPXact design capture and tools, to include any specific peripherals - a UART communications channel in this case, plus generic General Purpose Input Output (GPIO) control ports to monitor or stimulate the IO ports.



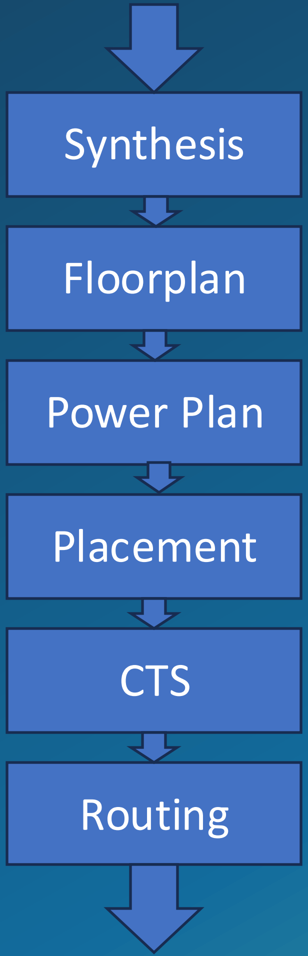
The Arm-based ZYNQ processing subsystem is completely independent of the SoC DUT but provides the master clock for the device, which can be reconfigured in the Vivado editor. [Set to 20MHz for basic functional testing in the example implementation flow]

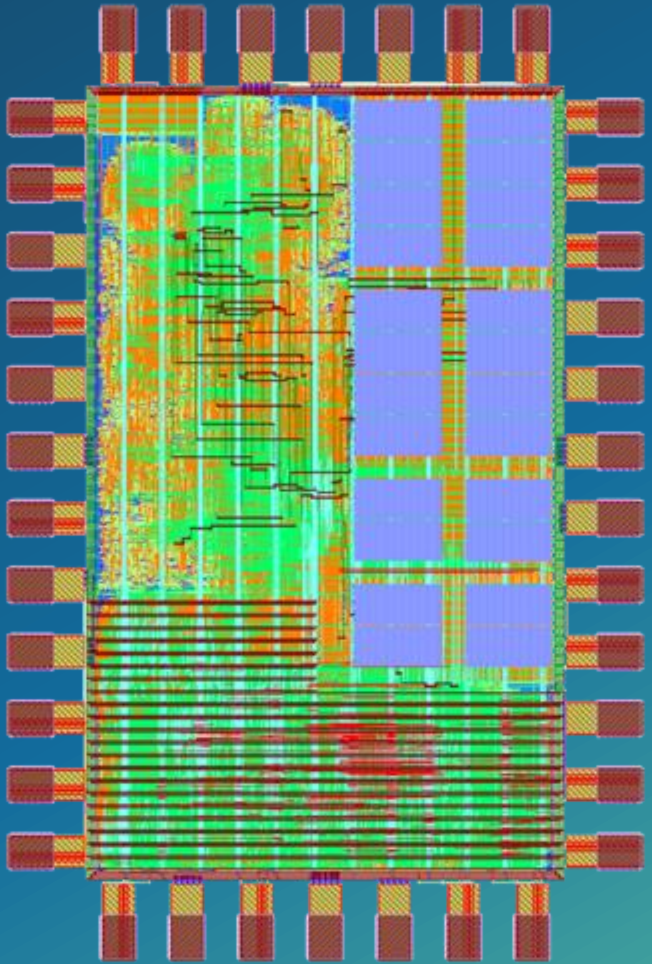


Zynq UltraScale+ MPSoC (3.3)

# ASIC flows

- reference scripts available for backend flow of nanoSoC using
    - Cadence Genus + innovus
    - Synopsys DC + ICC2
    - Synopsys Fusion compiler (under development)
- backend simulation environment the same as behavioral/frontend
- test/development board mirrors testbench environment





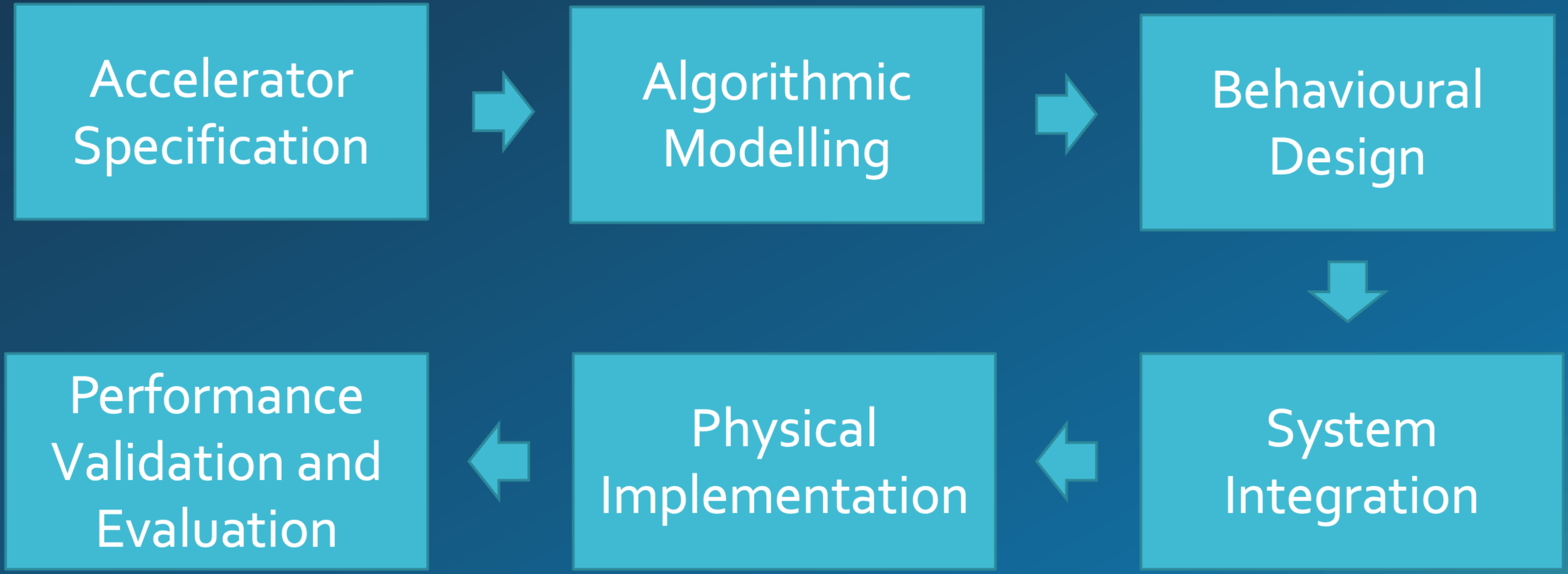
# about this session

- **about:** SoC Labs, … now and going forward
- **flows:** building an ASIC
- **example flows:** idea -> fpga -> custom ASIC
- **technology:** design references
- **worked examples**
- **projects:** from initial open call design contest
- **get involved**

soc labs

the global community for arm-based projects

# Accelerator Design Flow



soc labs

the global community for arm-based projects

# IP Specification

- High level description
  - Brief description of what your IP does
- Architectural decomposition
  - Can you break you IP down into sub-functions
  - Are any of these sub-function already available (e.g. FIFOs)
  - Describe the function of each of these sub-blocks
- Interfaces
  - E.g. main data interface : AXI, configuration interface: APB, clock(s), reset, interrupts etc.
  - Include data widths, and if memory mapped accelerator the address range needed
- Data throughput + buffering
  - Do you need buffers, what ports are these on, how deep are they
  - What data rate do you need for your IP
- Data diagram
  - How does data flow through your accelerator
  - Parallel/single stream
- Flow chart/Pseudo-code
- Feature test scheme
  - How are you going to verify your sub-blocks and IP

## Algorithmic Modelling

- Model your accelerator in a time-independent algorithmic model
- Allows flexibility and experimentation
- Gain familiarity with your IP
- Generate verification resources for you IP
- Not bound to hardware description languages
  - Typically use things like MATLAB or Python – but use whatever you’re familiar with
- This can be a complete system view, or be broken down as per your IP specification

soc labs

the global community for arm-based projects

## Behavioral design

- Convert your algorithmic model to a hardware description language
- Design sub-block at a time and verify
- Consider carefully the interface between sub-blocks
  - How does your IP handle backpressure
  - Valid-ready handshake?
- Once sub-blocks are verified, connect and re-verify

# System Integration

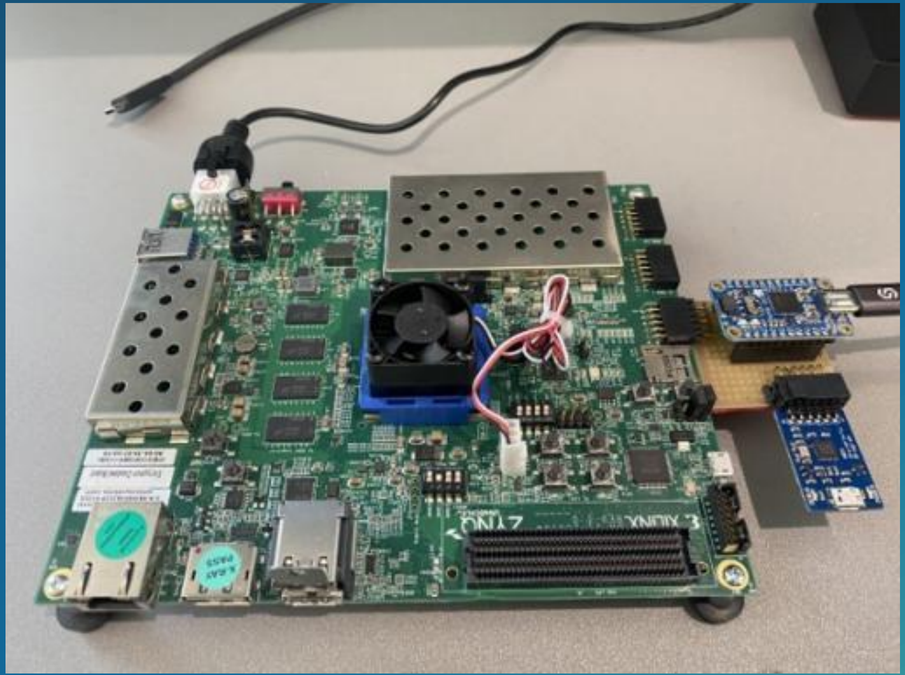
- Prior to this point, your accelerator may use only a basic handshake data interface
- You will need a top-level bus interface
  - AXI – High bandwidth
  - AHB – Moderate bandwidth
  - APB – Low bandwidth (also much simpler)
- This would also be where you add other components necessary for the system
  - Interrupts, reset, pins etc.

soc labs

the global community for arm-based projects

# Physical Implementation - FPGA

- Why?
  - Simulators do not always pick up on un-synthesizable constructs
  - Test your design in real-world
  - Testing/verification can be quicker at real world speeds (versus simulation speed)
  - Similarly software development can be easier this way



# Physical Implementation - FPGA

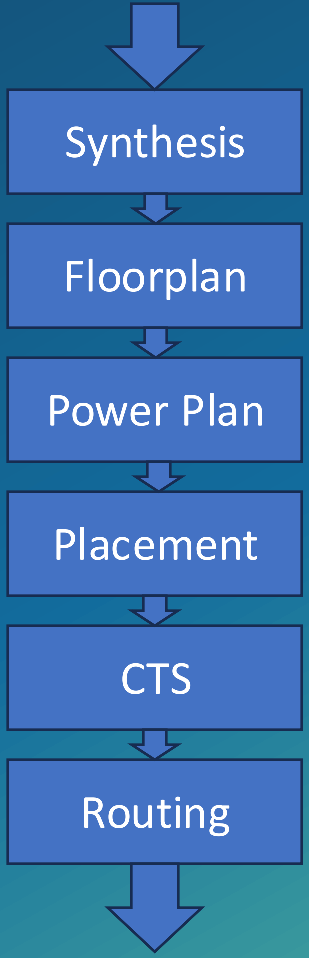
- Zynq is a popular platform
  - Processor system – Linux capable system, usually loaded with Pynq environment (a python environment for Zynq FPGAs)
  - Programmable logic – like traditional/bare-metal FPGA fabric to instantiate your design
  - AXI high bandwidth connections between PS and PL
  - Arm provide IP for AXI to AHB and AHB to APB conversion
- Bare-metal FPGA
  - Harder to evaluate individual IP's but very good for system evaluation
  - How to communicate between your IP and the outside world
    - Uart -> AXI debug bridge https://github.com/ultraembedded/core_dbg_bridge
    - JTAG -> AXI

soc labs

the global community for arm-based projects

# Physical Implementation - ASIC

- Why would you do this for a single IP?
  - If you are integrating into a system using a hierarchical approach (i.e. your block will be instantiated as a macro)
  - If you really care about your layout – the PPA of your IP can be determined by how your IP is physically laid out
  - If you want PPA for you single IP block – sometimes this can be difficult to get from a full system implementation, particularly in flat designs

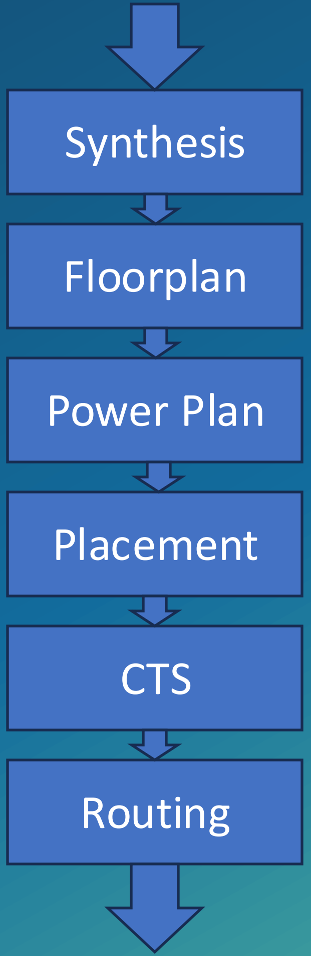


soc labs

the global community for arm-based projects

# Physical Implementation - ASIC

- Basic flow
  - Synthesis – turn your hardware description language to standard cells
  - Floorplan – decide where things are placed in your design
  - Power Plan – layout your power rails
  - Placement – Place the standard cells (some timing based and/or congestion-based optimization done here too)
  - Clock tree synthesis – Makes a tree of all the clock connections in your design and how. Optimisation of placement can be done here too
  - Routing – Final routing of all of your signals
  - Signoff – DRC checks, PPA checks, LVS, ERC



## about this session

- **about:** SoC Labs, … now and going forward
- **flows:** building an ASIC
- **example flows:** idea -> fpga -> custom ASIC
- **technology:** design references
- **worked examples**
- **projects:** from initial open call design contest
- **get involved**

# collaboration on research evaluation demonstrators

- initial focus on microcontroller infrastructure to support generic vehicles for research demonstrators:

  Software management of
  - Configuration
  - Parameter trimming and tuning
  - Mode control
  - Stimulus and response scenarios
  - Measurement and triggers for (external analysis)
- contribute to support quality publication
  - Measured (versus predicted) power/energy, performance (operations/MHz) …

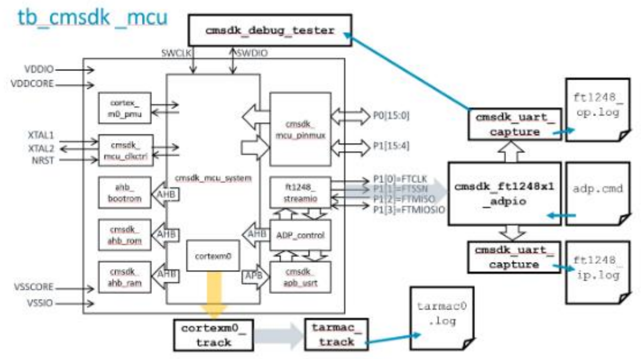
soc labs

the global community for arm-based projects

## entry to research: simple design, low cost fabrication

- AAA provides a wealth of commercially robust IP
  - And some subsystems
- enhance ‘Reference designs’
  - into reference system-on-chip realisations
- Cortex®-M0 System Design Kit (SDK) enhancement
  - Git resources to augment Arm’s simulation environment
  - Support implementation and validation
- okay for adding simple memory-mapped research experiments and components

Active Project



dflynn-University of Southampton

**Arm Cortex-M0 microcontroller**

A reference design based on an Arm Cortex-M0 CPU and the Cortex-M0 Design Kit provided in the Corstone-101 subsystem package, available under the Arm Academic Access agreement.

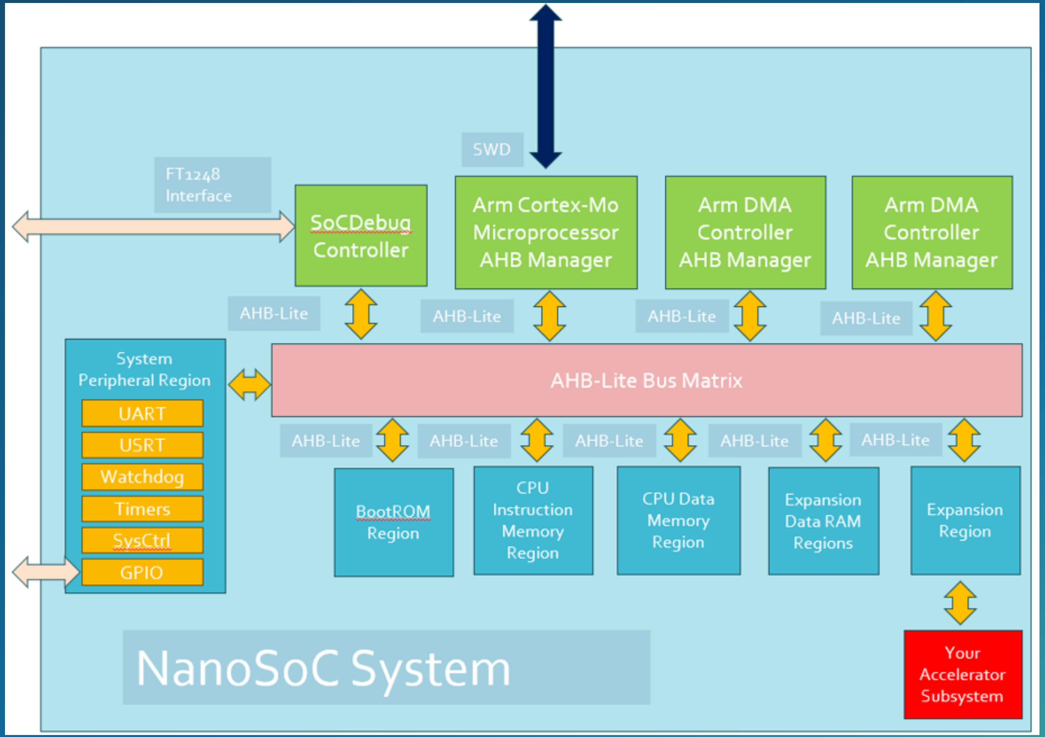
David Flynn

soc labs

the global community for arm-based projects

## entry for custom compute: ‘nanosoc’ reference design

- *single bus -> multi-master CMSDK;* efficient DMA for data delivery to custom compute, *multi-layer AMBA®* (AHB interconnect generation)
- Arm® Cortex M0
- Choice of DMA
  - Low area PL230 for simple transactions
  - DMA350 for complex transaction and AXI stream support

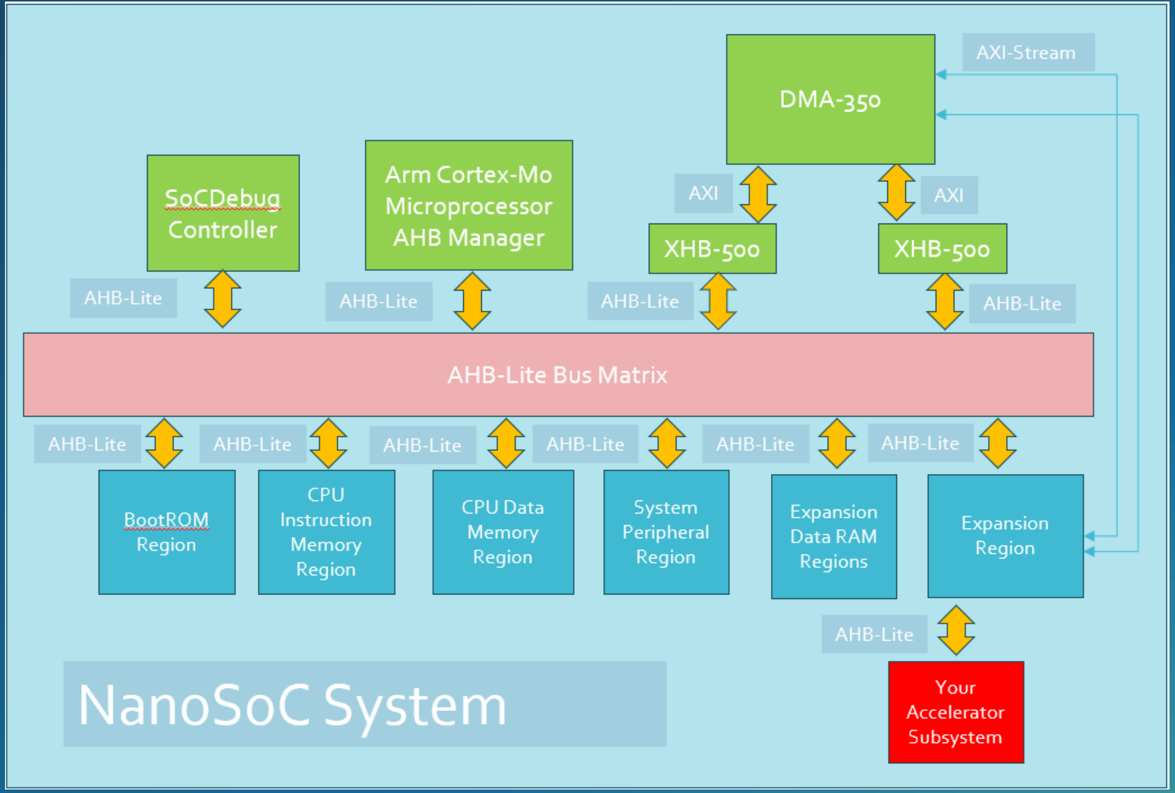


soc labs

the global community for arm-based projects

# nanoSoC + DMA-350

- Based around the nanoSoC system
- Using the DMA-350 in place of PL230
- Allows for more complex DMA transfers
- Also includes AXI stream port for hardware in DMA loop
- Uses 2 AXI-AHB masters, allows dedicated port for read and write
  - nearly doubles transfer rate

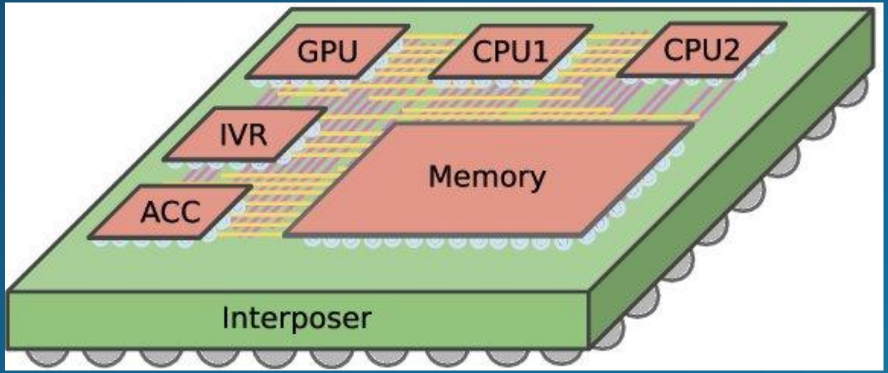


soc labs

the global community for arm-based projects

# A quick note: Chiplets

- We are starting to work on chiplet designs
- University of Southampton developing interposers
- Why Chiplets?:
    - Reduced costs, System Flexibility, Heterogenus integration, Improved PPA?
    - Most academics don't need 100 dies, so maximizing re-use and minimizing cost



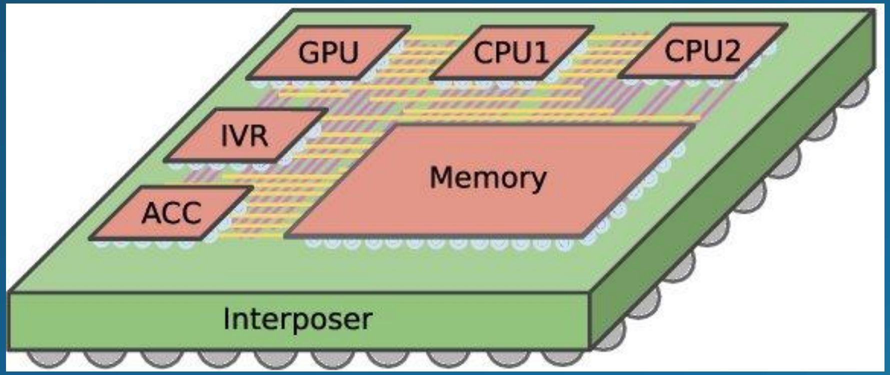
Interposer-Based Root of Trust: arXiv:2105.02917v1

soc labs

the global community for arm-based projects

# A quick note: Chiplets

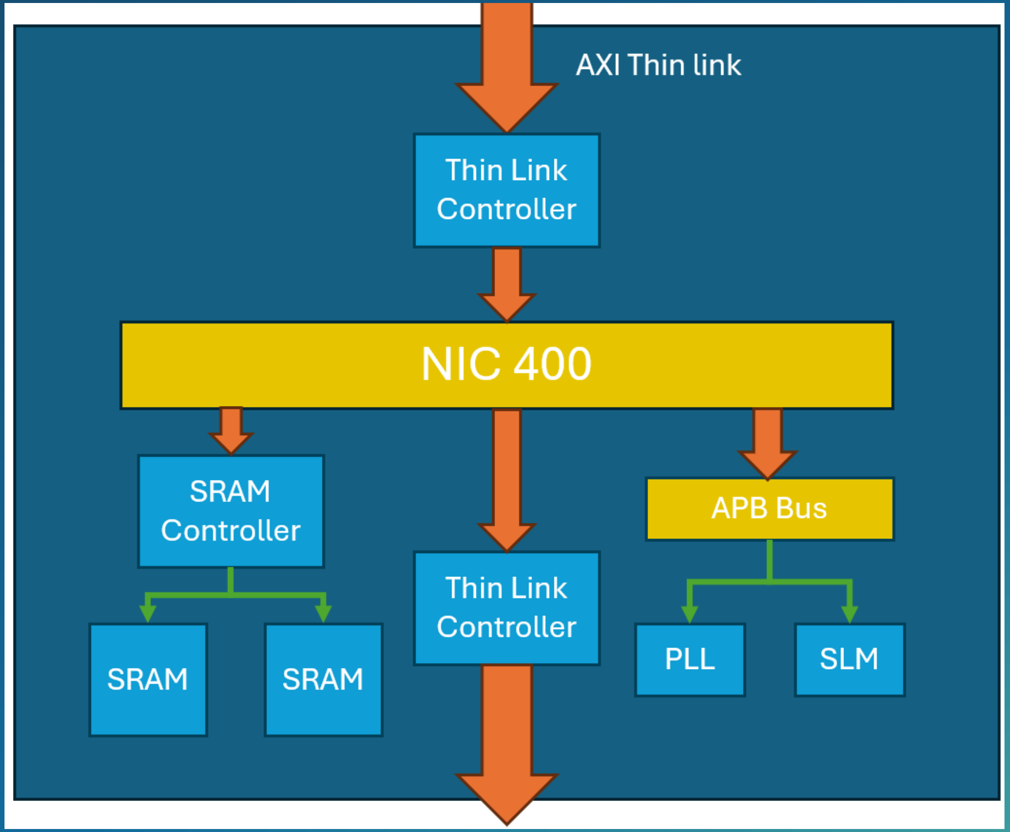
- Chiplet Challenges:
  - Not a lot of already developed IP in the open domain
  - Not fully standardized yet (UCIE, BoW, CCI)
  - Relatively high pin count per interface



Interposer-Based Root of Trust: arXiv:2105.02917v1

# A quick note: Chiplets

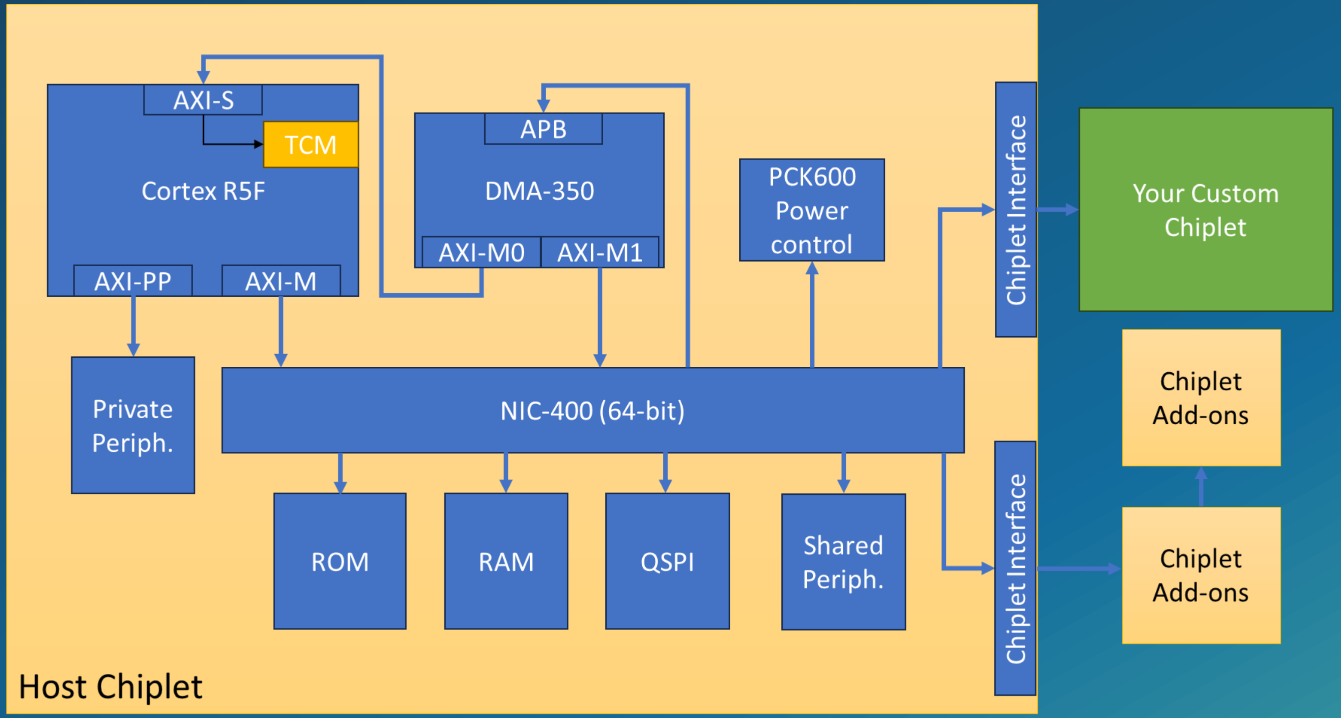
- SoCLabs SRAM Chiplet:
  - SRAM area is significant in ASICs
  - Particularly for bigger SoC where MBs of cache is needed
  - SRAM chiplet with 1MB SRAM plus daisy chaining to increase up to 16 MB
- Chiplet interface – Arm Thin Links
  - Converts an AXI or AHB interface to an AXI stream interface
  - Includes full addressing and channel control (size, burst, response etc.)



soc labs
the global community for arm-based projects

# milliSoC

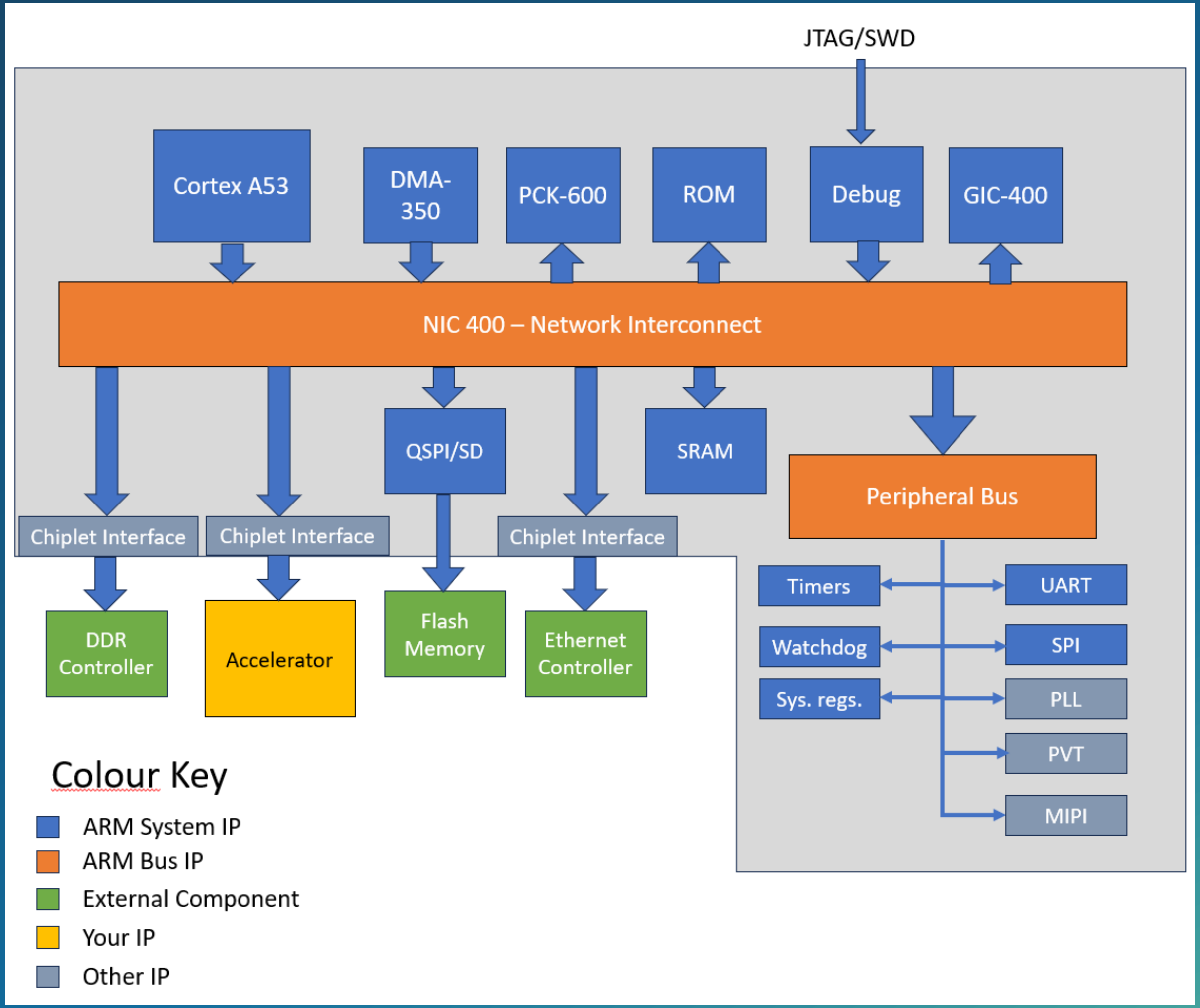
- Real time processor (Cortex R class)
- Tightly coupled memory
- Host-chiplet with 2 chiplet interfaces for:
    - Custom accelerator
    - Daisy chain of add-ons



Chiplet add-ons could be SRAM, ethernet, USB, DDR

# Request for Collaboration: ‘megasoc’

- Visibility from early soclabs collaborators
- Configurable DMA controller
  - (not in similar current Corstone platforms)
- Accelerator validation independently
  - integration test



# forming our shared “roadmap”

- driven by collaborating partners’ needs within Arm AAA provision…
- Cortex-M CPU, controller class
  - (picosoc ?) minimal infrastructure to host energy harvesting or mixed-signal
  - nanosoc – Cortex-M0 CPU + DMA230 (enhanced option) AHB DMA
  - (microsoc ?) CPU + AXI interconnect, wider memory, DMA350
  - (millisoc ??) CPU/DMA + asynchronous bridge to DVFS capable subsystems
    - PVT sensors
- Cortex-A CPU, virtual-memory Linux OS
  - kilo-/mega-soc(!) – bridge from Zynq FPGA prototyping platform

*lots more AAA IP to choose from…*

## about this session

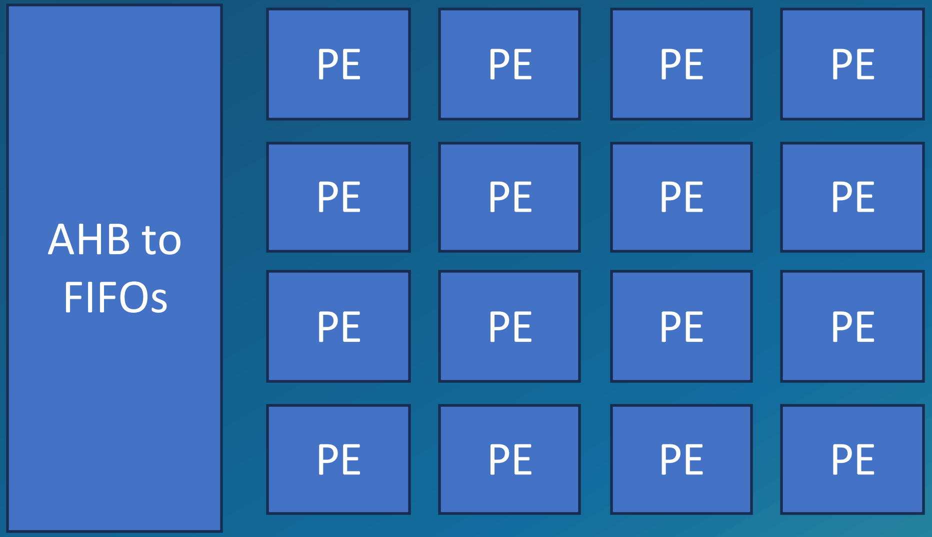
- **about:** SoC Labs, … now and going forward
- **flows:** building an ASIC
- **example flows:** idea -> fpga -> custom ASIC
- **technology:** design references
- **worked examples**
- **projects:** from initial open call design contest
- **get involved**

## Aside

- When implementating an AI/ML model you effectively have 3 choices
  - Completely general
    - Effectively a matrix multiplication engine, by tiling your matrices to fit the hardware you could run any model on this
    - Typically small area but requires continuous loading of tiles
  - Fixed architecture
    - Model architecture is fixed but weights can vary
    - Larger area, only requires loading of weights at startup
  - Fixed model
    - Model architecture and weights fixed
    - Slightly smaller area than fixed architecture (as tie high or tie low cells used instead of registers) no loading of weights

# Example 1: Gemm Engine

- General Matrix Multiply: $\boldsymbol{C} \leftarrow \alpha \boldsymbol{A}\boldsymbol{B} + \beta \boldsymbol{C}$
- 4x4 matrix multiplication (fixed point)
- What am I trying to achieve?
  - Verify in silicon – measure physical PPA
- Don’t need to run a full/large model
- 32 bit AHB bus – 16 bit words
- Model size: 10’s KiB
- Bandwidth: no real constraints

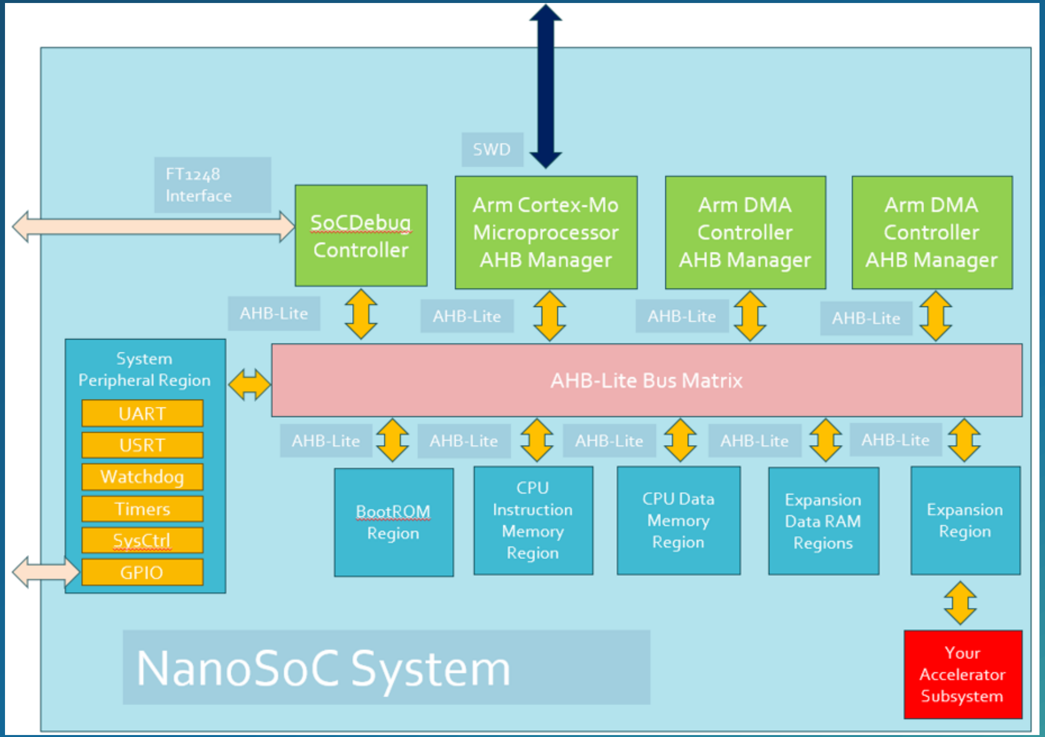


soc labs

the global community for arm-based projects

# Example 1: Gemm Engine

- What are the system requirements?
- 32 bit AHB bus
- CPU for pre/post processing data
- DMA for data transfer
- 10’s KiB on chip SRAM

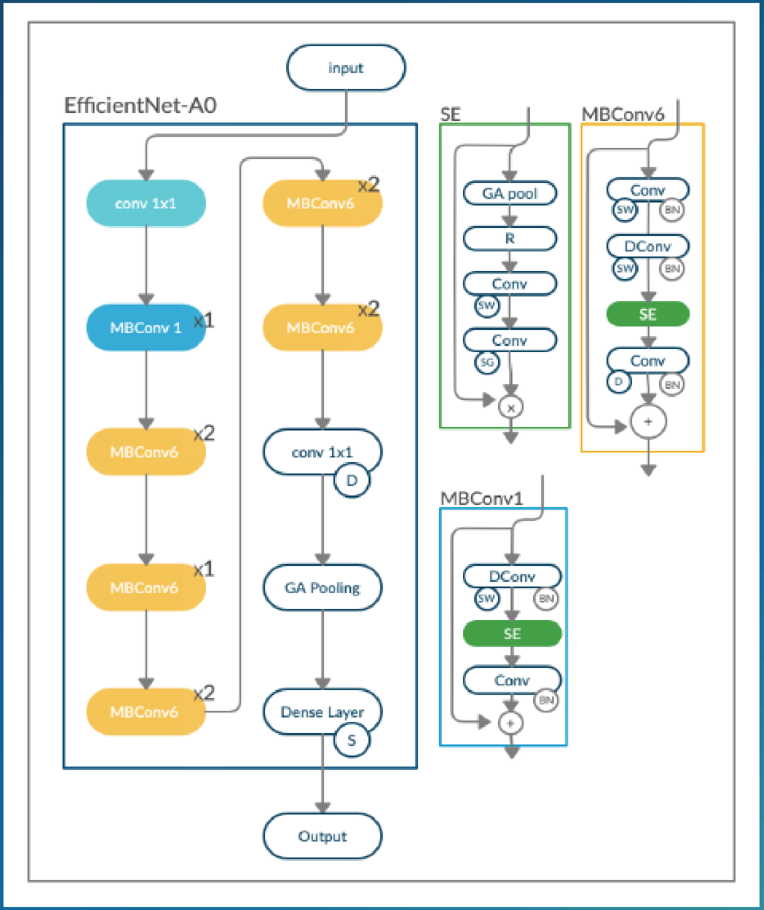


soc labs

the global community for arm-based projects

## Example 2: Voice Keyword detection

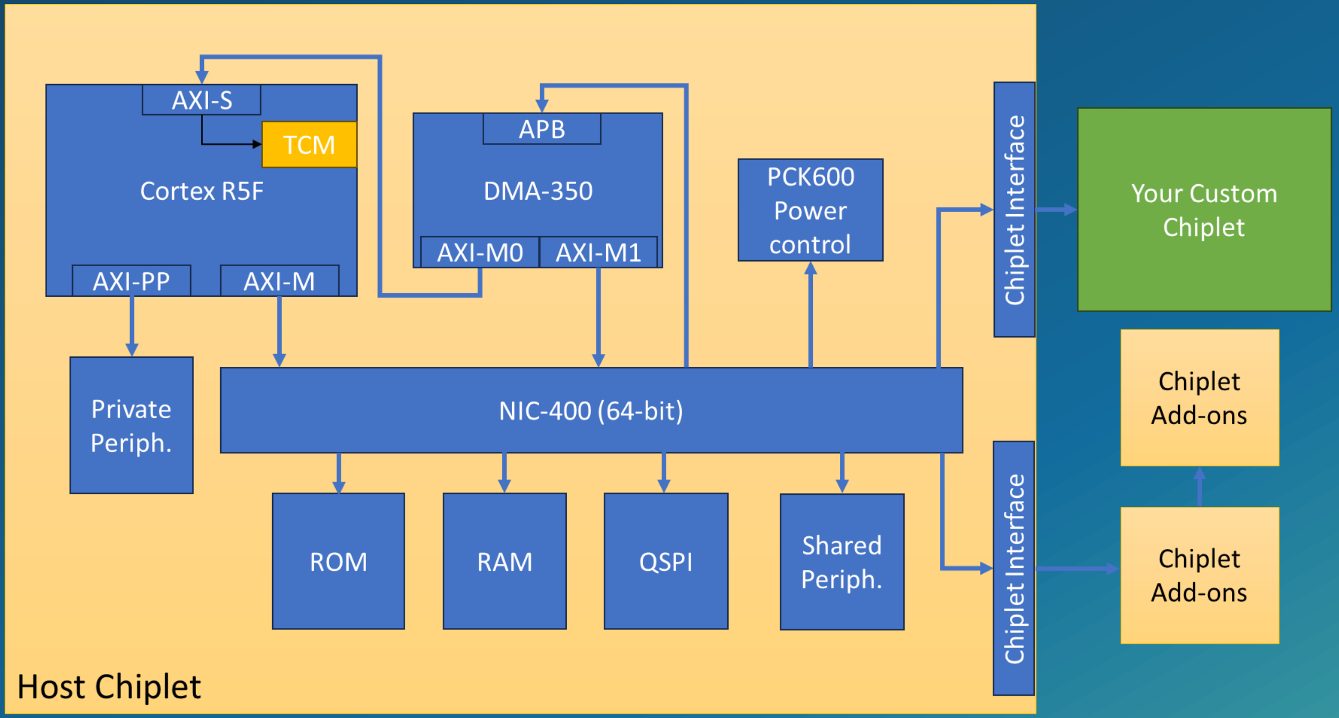
- CNN Model
- What am I trying to achieve?
  - Verify in silicon – measure physical PPA
  - **Deploy** with microphone
- Need to run full model
- 16 bit audio data
- Model size: 100 KiB
- Bandwidth:
  - Data – 16 bit 44.1 kHz
  - Model – 4 GBps (100 KiB x 44.1 kHz)



EFFICIENTNET-ABSOLUTE ZERO FOR CONTINUOUS SPEECH KEYWORD SPOTTING arXiv:2012.15695v1

# Example 2: Voice Keyword detection

- What are the system requirements?
- High bandwidth bus – AXI 64 bit @ 500 MHz
- 100 KiB storage (sram chiplet useful here)
- Real-time operation
  - Must complete before next audio sample



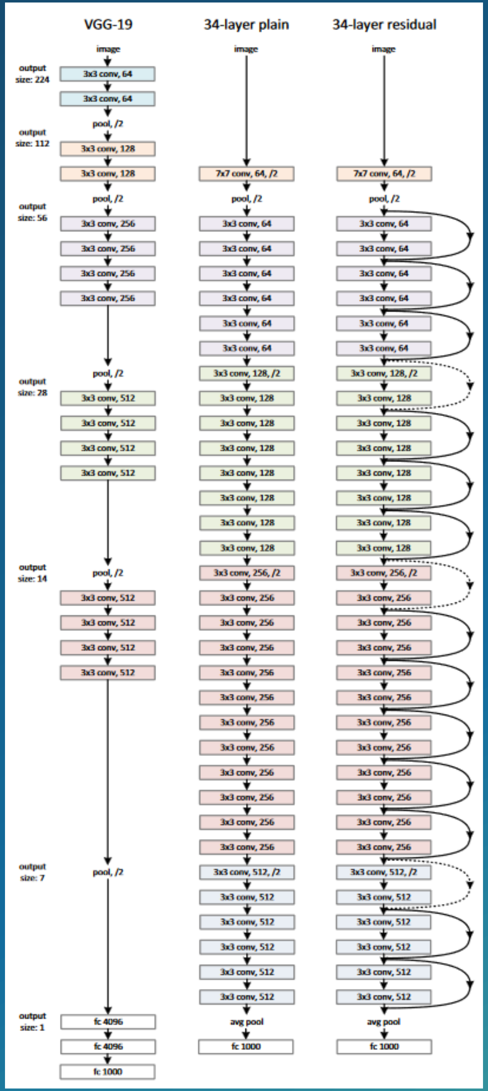
Chiplet add-ons could be SRAM, ethernet, USB, DDR

soc labs

the global community for arm-based projects

# Example 3: Vision Object detection

- Deep learning model
- What am I trying to achieve?
  - Verify in silicon – measure physical PPA
  - **Deploy** with camera
- Need to run full model
- 224x224x3 Video data (150 KiB/frame)
- Model size: 42 MiB
- Bandwidth:
  - Data – 28 Mbps (150 KiB @ 24fps)
  - Model – 8 Gbps (42 MiB x 24 fps)



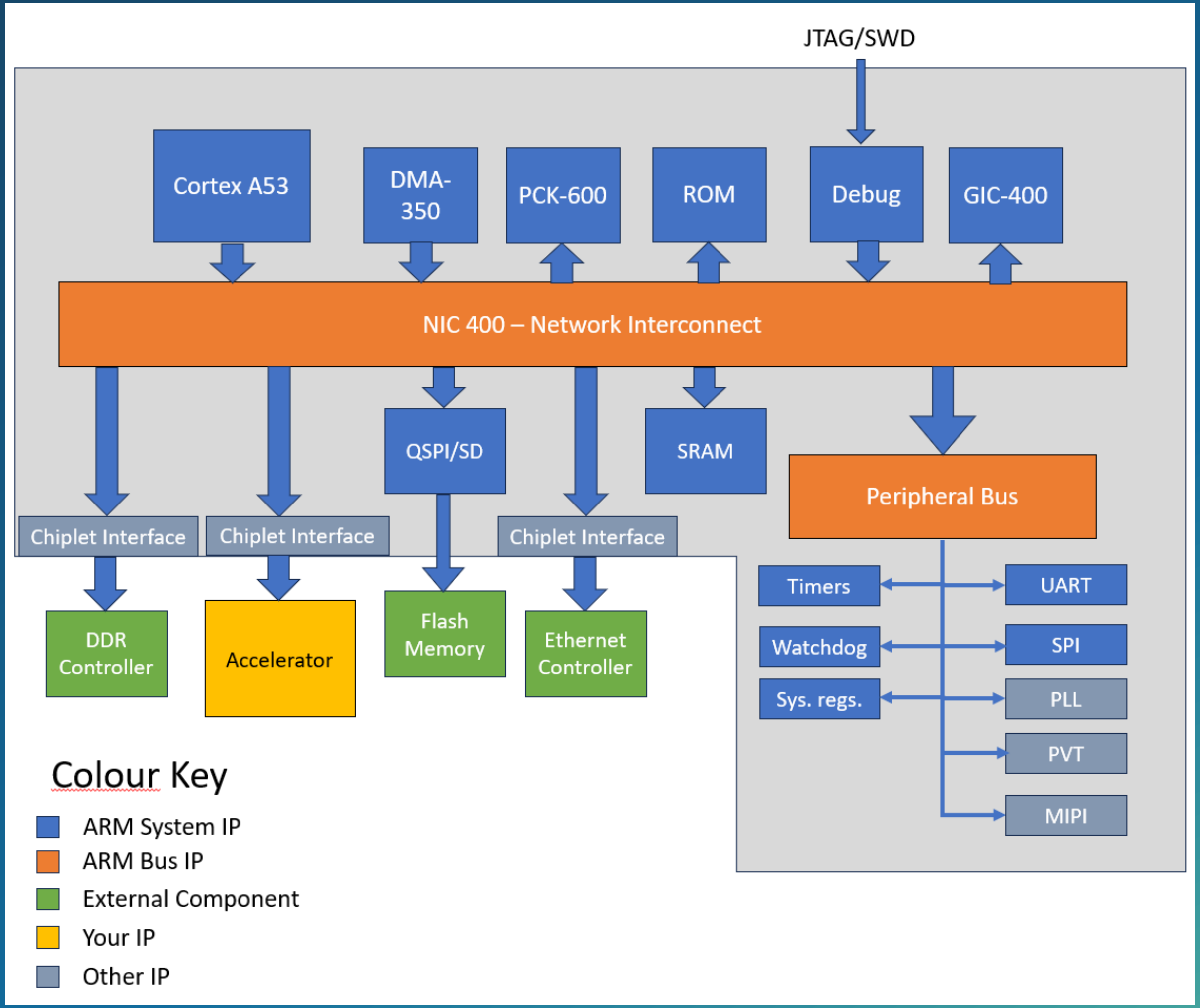
Deep Residual Learning for Image Recognition
arXiv:1512.03385v1

soc labs

the global community for arm-based projects

# Example 3: Vision Object detection

- What are the system requirements?
- High bandwidth bus – AXI 64 bit @ 1 GHz
- 42 MiB storage – SRAM Chiplet or DDR
- Real-time operation
  - Must complete before next video sample
- Full OS?



Colour Key

- 🟦 ARM System IP
- 🟧 ARM Bus IP
- 🟩 External Component
- 🟨 Your IP
- ⬜ Other IP

## about this session

- **about:** SoC Labs, … now and going forward
- **flows:** building an ASIC
- **example flows:** idea -> fpga -> custom ASIC
- **technology:** design references
- **worked examples**
- **projects:** from initial open call design contest
- **get involved**

# 2023/24 contest

- hardware Track:
  - BlackBear: Reconfigurable AI for large image (Jen-Chien Chang, NCKU)
  - DeepSoCFlow: Accelerate DNNs for Scientific Compute (Abarajithan Gnaneswaran UCSC/Moratuwa)
  - Real-Time Edge AI SoC: High-Speed Low Complexity Reconfigurable-Scalable Architecture for DNNs (Sai Dinesh Y V, IITH)
- education Track:
  - Hell Fire SoC: Configurable Systolic array processing (Srimanth Tenneti, Cincinnati)
  - Fast-kNN: Implementing a k-Nearest-Neighbour classifier (Epifanios Baikas, University of Southampton)

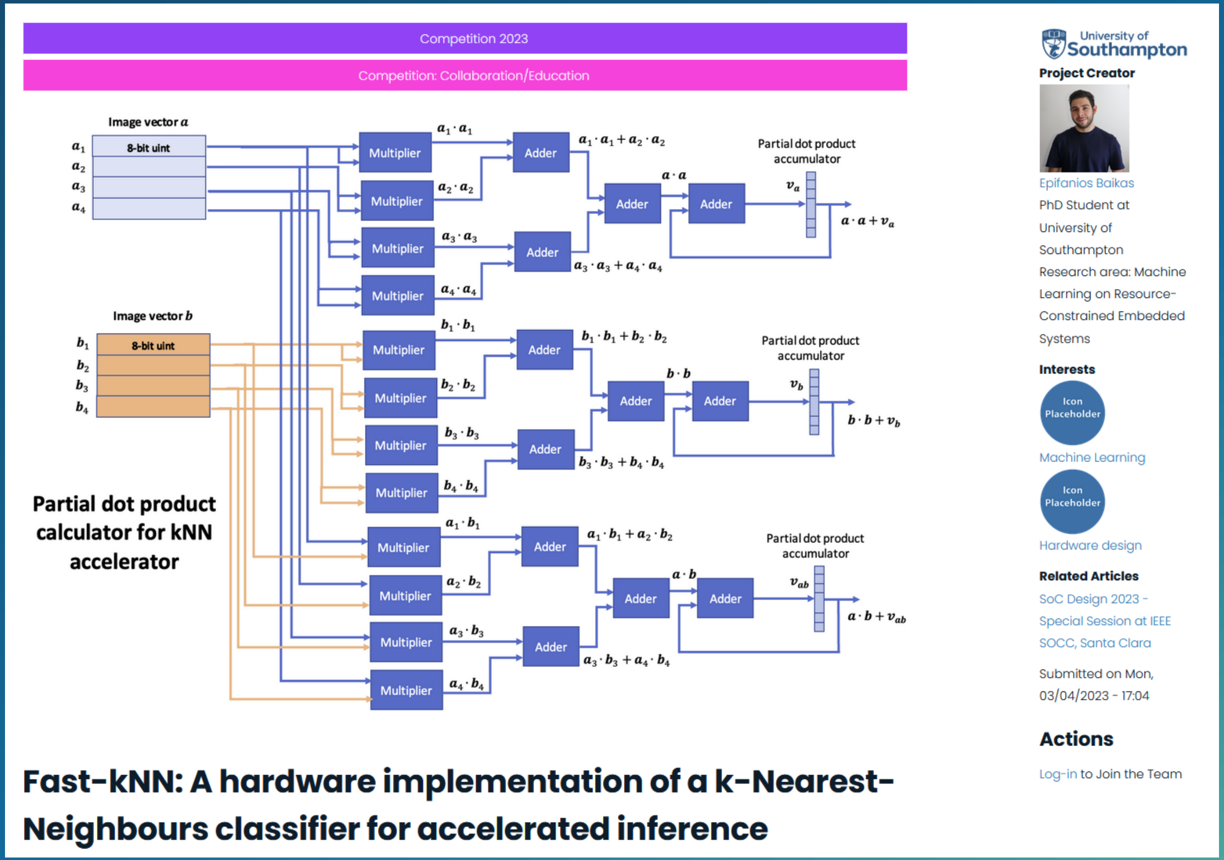
soc labs

the global community for arm-based projects

# Fast-kNN - education track

University of Southampton

- PhD student – Epifanios Baikus
- began with little experience in hardware design
- developed accelerator inside **nanoSoC** reference environment
  - No last-minute integration needed
- submitted for tape out on TSMC 65nm mini-ASIC shuttle



Competition 2023

Competition: Collaboration/Education

**Fast-kNN: A hardware implementation of a k-Nearest-Neighbours classifier for accelerated inference**

University of Southampton

**Project Creator**

Epifanios Baikas

PhD Student at University of Southampton

Research area: Machine Learning on Resource-Constrained Embedded Systems

**Interests**

Icon Placeholder

Machine Learning

Icon Placeholder

Hardware design

**Related Articles**

SoC Design 2023 - Special Session at IEEE SOCC, Santa Clara

Submitted on Mon, 03/04/2023 - 17:04

**Actions**

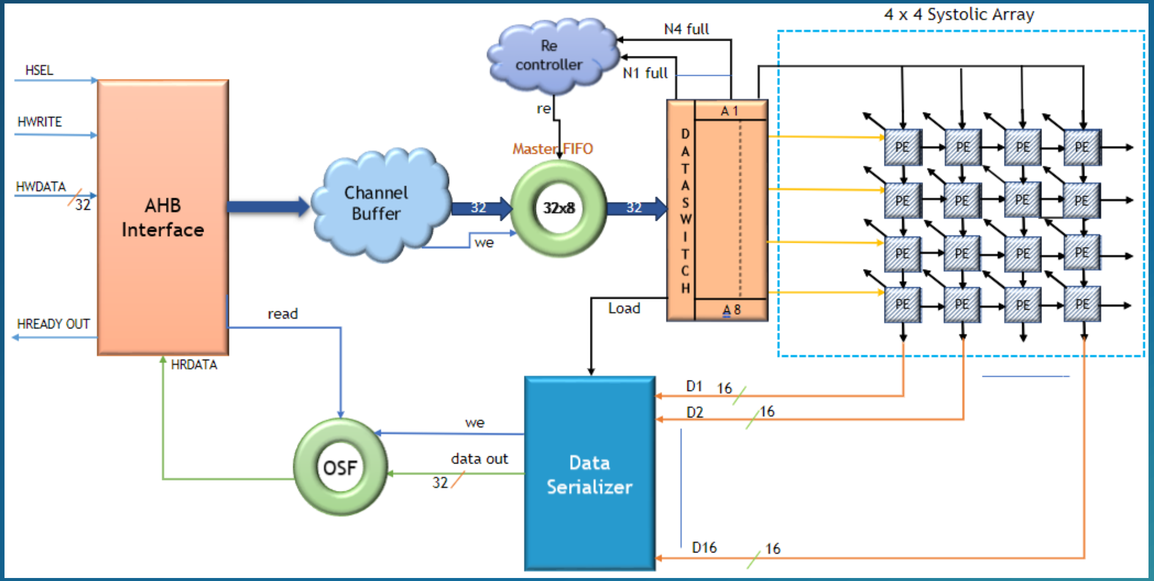
Log-in to Join the Team

soc labs
the global community for arm-based projects

## Hell fire SoC – education track

UNIVERSITY OF Cincinnati

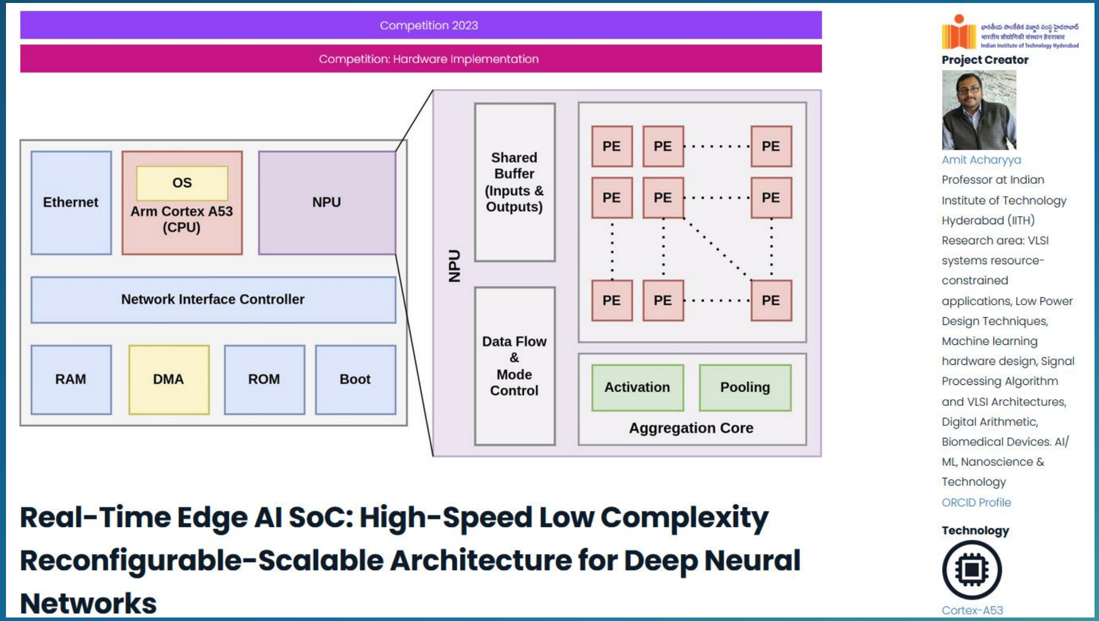
- Independent project
- systolic array with 4x4 processing elements
- submitted for tapeout on TSMC 65nm mini-ASIC shuttle
- design includes nanoSoC with DMA-350 instead of PL230
- also developed his own SoC based on Arm Design Start IP



# IITH – hardware track

భారతీయ సాంకేతిక విజ్ఞాన సంస్థ హైదరాబాద్
भारतीय प्रौद्योगिकी संस्थान हैदराबाद
Indian Institute of Technology Hyderabad

- edge AI SoC for image processing
- previously taped out as standalone NPU with FPGA
- SoC based on Arm's Corstone 1000 subsystem (SSE-710) + DMA-350
- currently in backend flow for tape out in May
- backend flow includes multiple power and clock domains



Competition 2023

Competition: Hardware Implementation

**Real-Time Edge AI SoC: High-Speed Low Complexity Reconfigurable-Scalable Architecture for Deep Neural Networks**

భారతీయ సాంకేతిక విజ్ఞాన సంస్థ హైదరాబాద్
भारतीय प्रौद्योगिकी संस्थान हैदराबाद
Indian Institute of Technology Hyderabad

**Project Creator**

Amit Acharyya
Professor at Indian Institute of Technology Hyderabad (IITH)
Research area: VLSI systems resource-constrained applications, Low Power Design Techniques, Machine learning hardware design, Signal Processing Algorithm and VLSI Architectures, Digital Arithmetic, Biomedical Devices. AI/ML, Nanoscience & Technology
ORCID Profile

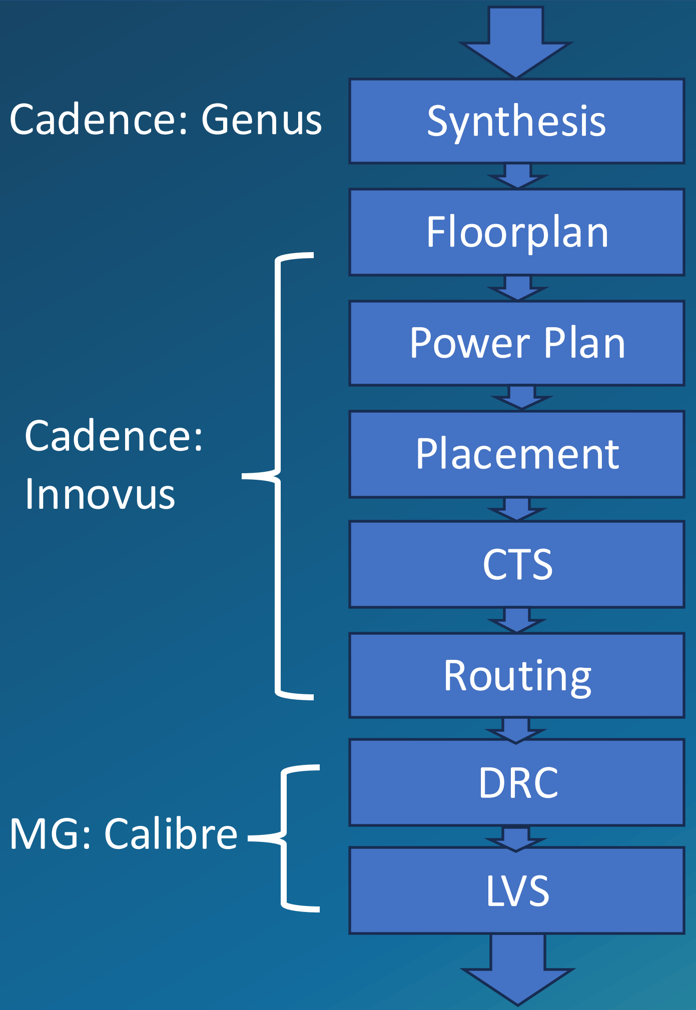
**Technology**

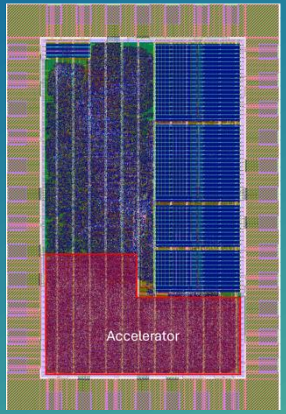
Cortex-A53

# NanoSoC Tapeout

- 2 Custom accelerators taped out with nanosoc reference design (more on the way)
- Both contestants from the 2023/24 contest in the education track
- Srimanth: Master student
  - Hell Fire SoC – a systolic array accelerator for AI/ML applications
- Fanis: Junior PhD student
  - Fast-kNN – hardware implementation of Euclidean distance algorithm for kNN image classification

| | |
|---|---|
| Process | TSMC 65nm LP |
| Metal Scheme | 9m 6x1z1u |
| Lib Corners | ss_1.08V_125C<br>tt_1.2V_25C<br>ff_1.32V_-40C |
| Chip area | 1x1.5mm (mini@SIC) |
| Instances | 2x 8kB Register file<br>2x 16kB register file |
| IO Pads | 38 total<br>16 GPIO |
| Clocks | 1x System clock 1x SWD clock |
| Max Frequency | 240 MHz System Clock |



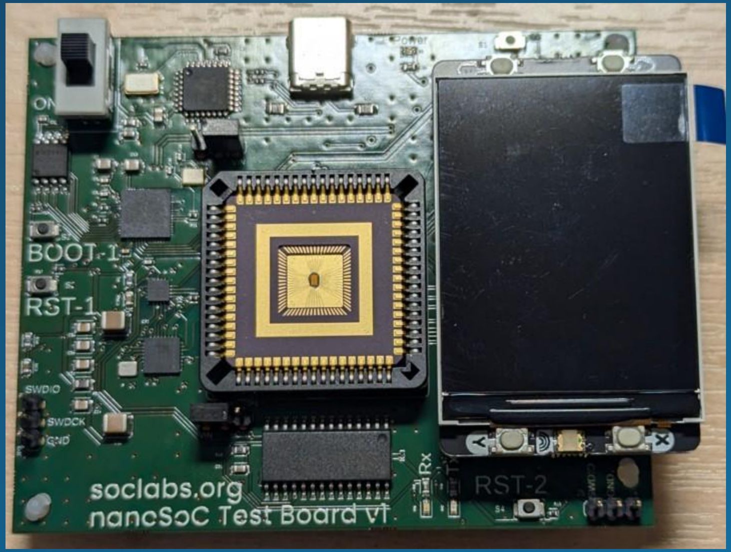


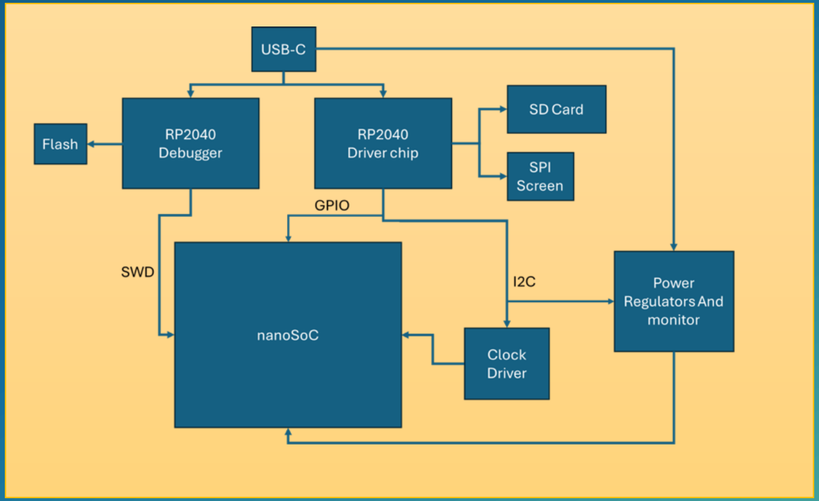
soc labs

the global community for arm-based projects

## NanoSoC Test-board: Hardware

- Low-cost test board for showcase and development on nanoSoC ASIC.
- Uses 2 RP2040 chips from Raspberry Pi (dual core Arm® Cortex M0+)
- Enables support for SD card, screen, SWD debugging, clock generation and power monitoring
- USB-C power and interface to both RP2040s





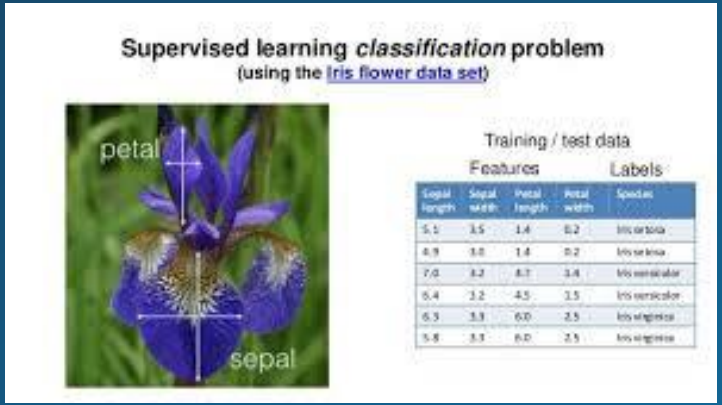
soc labs

the global community for arm-based projects

# NanoSoC Test-board: Software

- Hell Fire Demo – IRIS dataset classification

1. RP2040 driver sends program file + all data and weights for the neural network to nanosoc
2. Nanosoc computes the output of the neural network using the hell fire accelerator
3. Nanosoc handshakes the output back to the RP2040
4. RP2040 displays result on screen
5. Loop back to 2 until all calculations are complete
6. Displays the average power consumption


petal
sepal

Supervised learning *classification* problem

(using the Iris flower data set)

Training / test data

| Features | | | | Labels |
|---|---|---|---|---|
| Sepal length | Sepal width | Petal length | Petal width | Species |
| 5.1 | 3.5 | 1.4 | 0.2 | Iris setosa |
| 4.9 | 3.0 | 1.4 | 0.2 | Iris setosa |
| 7.0 | 3.2 | 4.7 | 1.4 | Iris versicolor |
| 6.4 | 3.2 | 4.5 | 1.5 | Iris versicolor |
| 6.3 | 3.3 | 6.0 | 2.5 | Iris virginica |
| 5.8 | 3.3 | 6.0 | 2.5 | Iris virginica |

- Fast-kNN Demo – Fashion MNIST classification

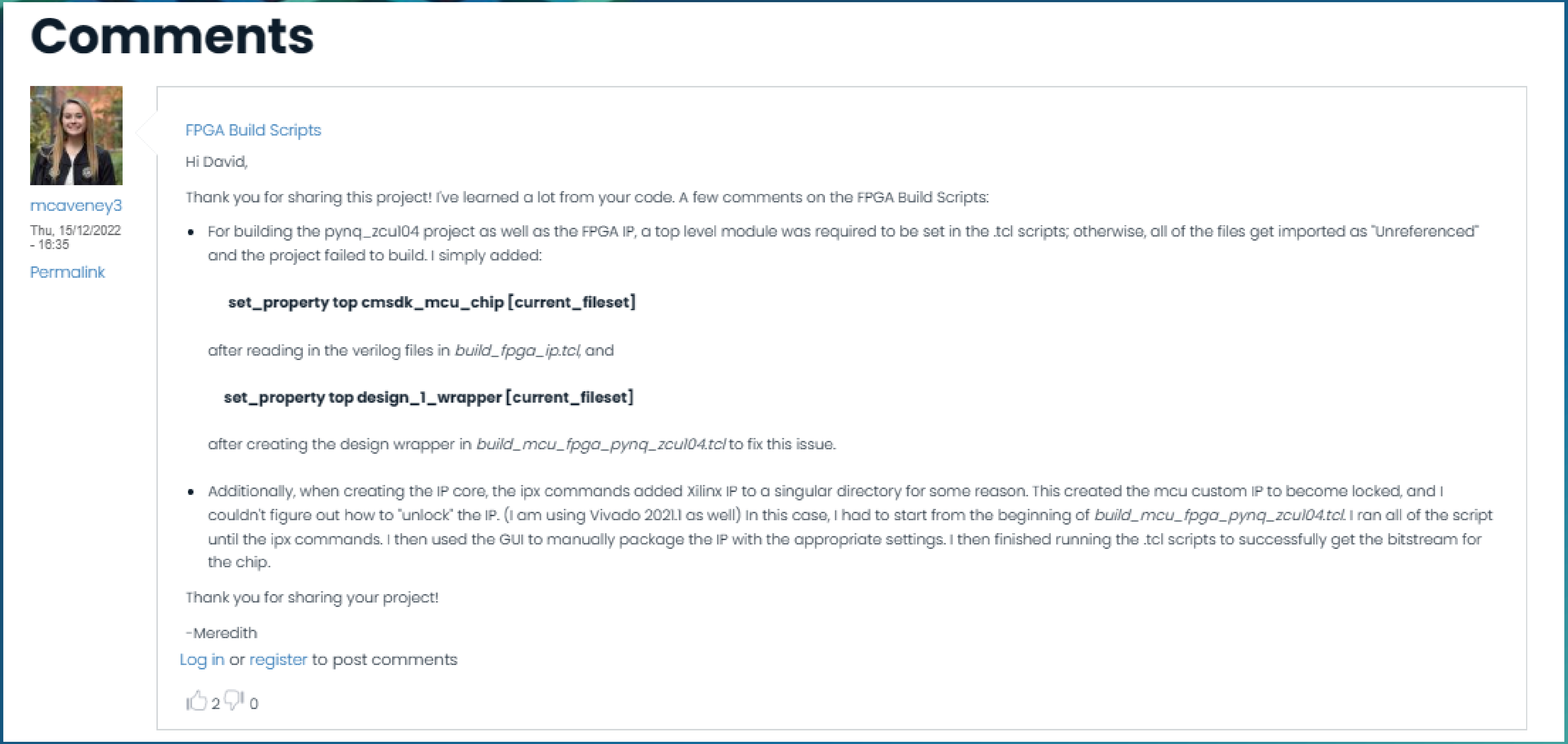
1. RP2040 loads all data from the SD card to RAM. Sends the program file
2. RP2040 sends the unlabelled image and 10 labelled images to nanosoc
3. Nanosoc runs a comparison of the images and handshakes the values of the comparison to RP2040
4. RP2040 sends next 10 labelled images until all 100 have been sent
5. Loop back to 2 until all unlabelled images are sent
6. Displays the results of the comparisons

## about this session

- **about:** SoC Labs, … now and going forward
- **flows:** building an ASIC
- **example flows:** idea -> fpga -> custom ASIC
- **technology:** design references
- **worked examples**
- **projects:** from initial open call design contest
- **get involved**

# about SoC Labs

it only works if we communication, share and collaborate…



## Comments

mcaveney3

Thu, 15/12/2022 - 16:35

Permalink

FPGA Build Scripts

Hi David,

Thank you for sharing this project! I've learned a lot from your code. A few comments on the FPGA Build Scripts:

- For building the pynq_zcu104 project as well as the FPGA IP, a top level module was required to be set in the .tcl scripts; otherwise, all of the files get imported as "Unreferenced" and the project failed to build. I simply added:

  **set_property top cmsdk_mcu_chip [current_fileset]**

  after reading in the verilog files in *build_fpga_ip.tcl*, and

  **set_property top design_1_wrapper [current_fileset]**

  after creating the design wrapper in *build_mcu_fpga_pynq_zcu104.tcl* to fix this issue.

- Additionally, when creating the IP core, the ipx commands added Xilinx IP to a singular directory for some reason. This created the mcu custom IP to become locked, and I couldn't figure out how to "unlock" the IP. (I am using Vivado 2021.1 as well) In this case, I had to start from the beginning of *build_mcu_fpga_pynq_zcu104.tcl*. I ran all of the script until the ipx commands. I then used the GUI to manually package the IP with the appropriate settings. I then finished running the .tcl scripts to successfully get the bitstream for the chip.

Thank you for sharing your project!

-Meredith

Log in or register to post comments

👍 2 👎 0

## Chiplet Contest

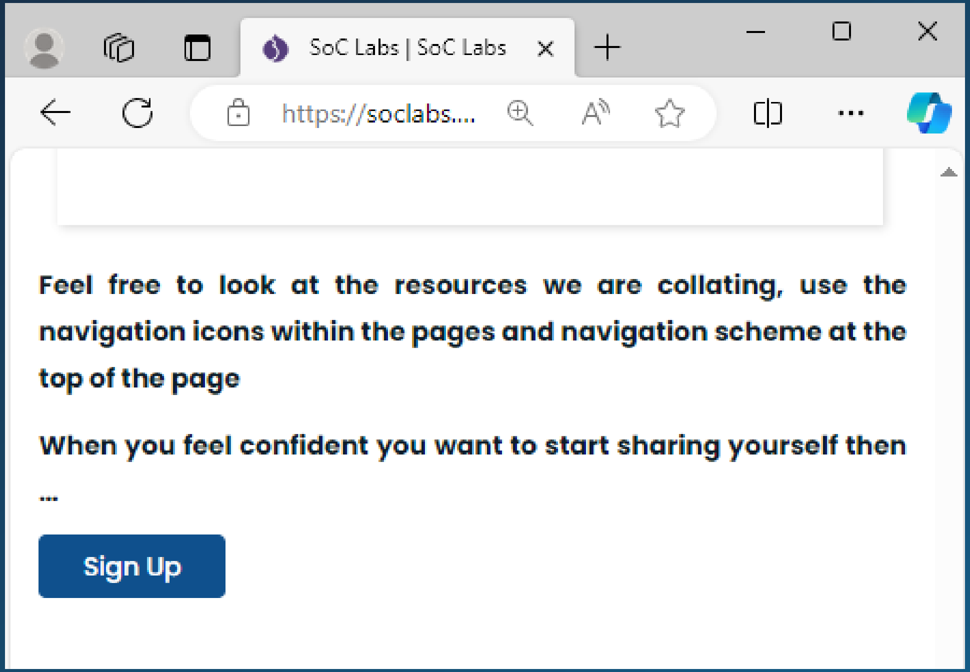
- Full details here: https://soclabs.org/article/design-contest-chiplet-based-soc-2025
- Announced this week at IEEE SOCC
- "contest for creation of an academic Chiplet based disaggregated SOC using the ARM ecosystem."
- SoC Labs will arrange for the winning design:
  - funding toward die fabrication costs for custom chiplets
  - fabrication of a custom interposer/package
  - design support during the year
  - subsidies for travel to the IEEE SOCC 2025 conference

## contest: entry

community centric hardware design

- individual and institutional skills development and collaboration
- building SoC design capability, sharing knowledge and experience (together we solve problems and learn faster)
- expand number of academics/institutions that produce SoCs
- no requirement for a novel solution
- reuse of existing design as important as creation of new design
- new application of a well know technique
- create shared resources especially **verification** efforts
- about the journey not the technology/IP

## contest: sign up

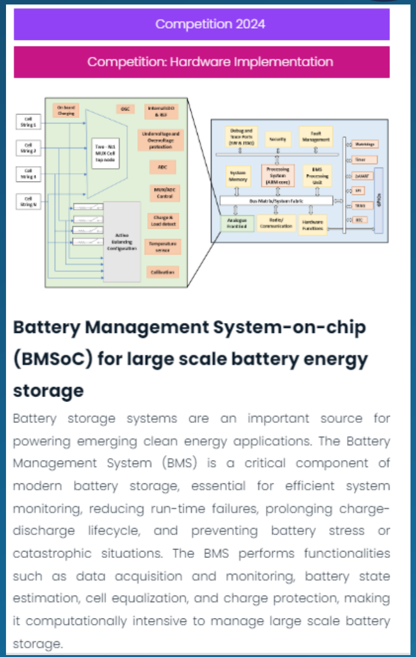


Feel free to look at the resources we are collating, use the navigation icons within the pages and navigation scheme at the top of the page

When you feel confident you want to start sharing yourself then ...

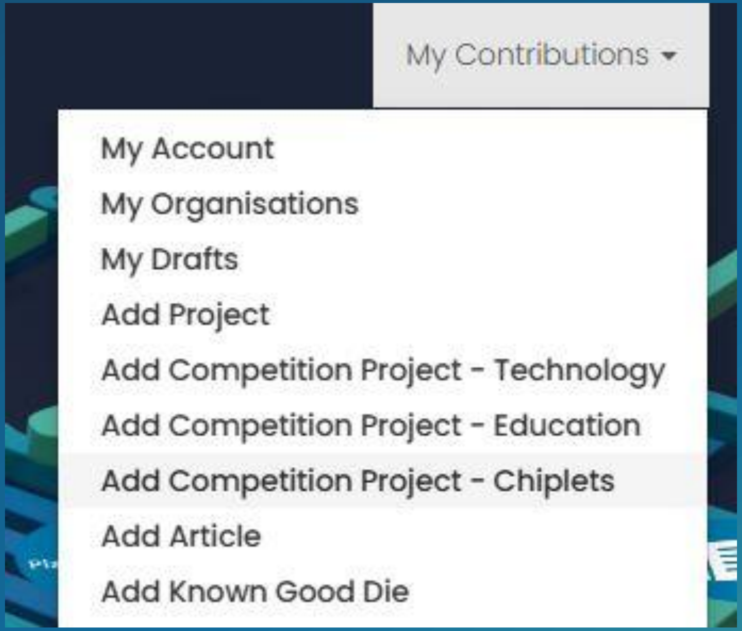
simply sign up on soclabs.org home page

develop project concept, an image and summary



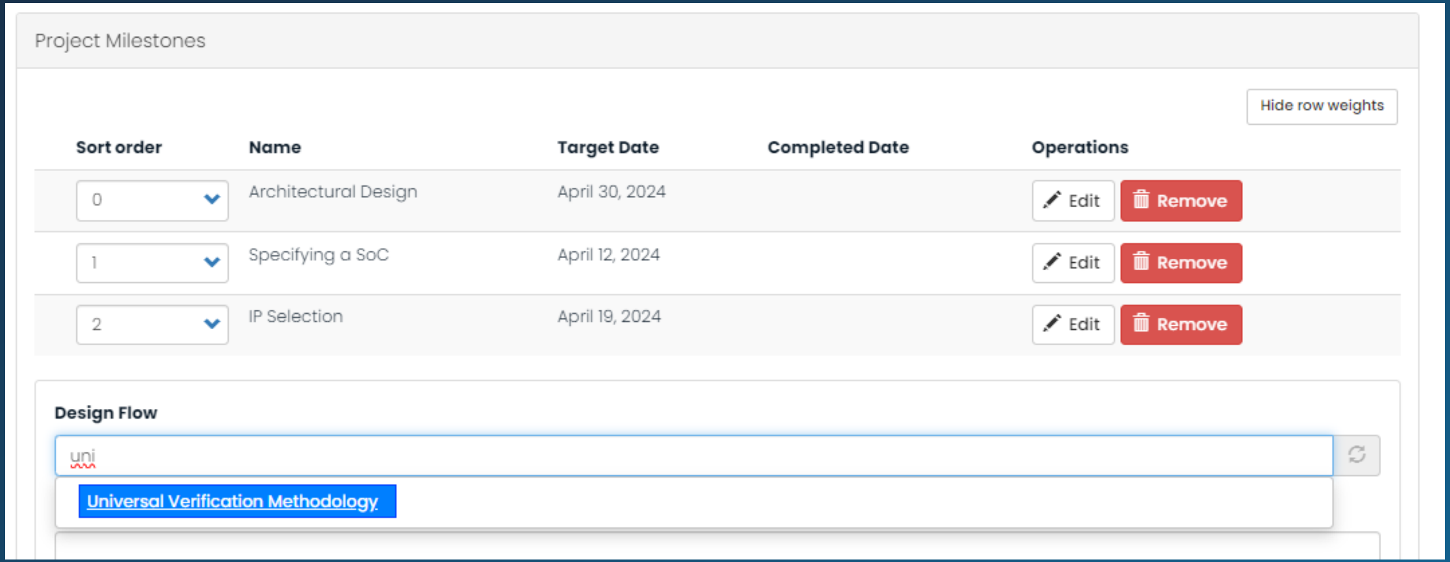
**Battery Management System-on-chip (BMSoC) for large scale battery energy storage**

Battery storage systems are an important source for powering emerging clean energy applications. The Battery Management System (BMS) is a critical component of modern battery storage, essential for efficient system monitoring, reducing run-time failures, prolonging charge-discharge lifecycle, and preventing battery stress or catastrophic situations. The BMS performs functionalities such as data acquisition and monitoring, battery state estimation, cell equalization, and charge protection, making it computationally intensive to manage large scale battery storage.


My Contributions
My Account
My Organisations
My Drafts
Add Project
Add Competition Project - Technology
Add Competition Project - Education
Add Competition Project - Chiplets
Add Article
Add Known Good Die

add your project to a chiplet via My Contributions

# contest: project progress



Project Milestones

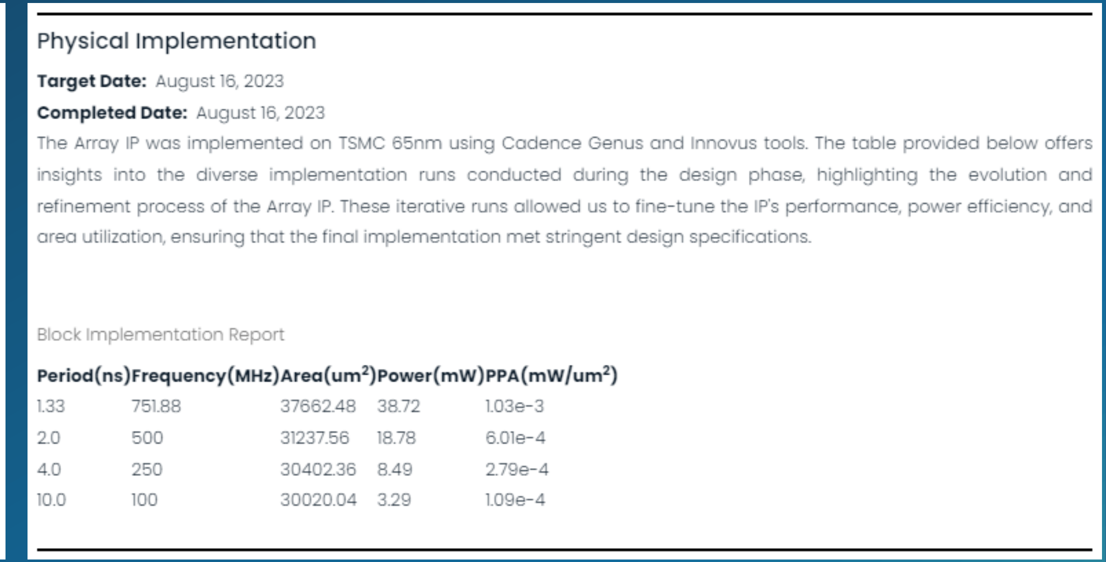
Hide row weights

| Sort order | Name | Target Date | Completed Date | Operations |
|---|---|---|---|---|
| 0 | Architectural Design | April 30, 2024 | | Edit Remove |
| 1 | Specifying a SoC | April 12, 2024 | | Edit Remove |
| 2 | IP Selection | April 19, 2024 | | Edit Remove |

Design Flow

uni

Universal Verification Methodology



Physical Implementation

**Target Date:** August 16, 2023

**Completed Date:** August 16, 2023

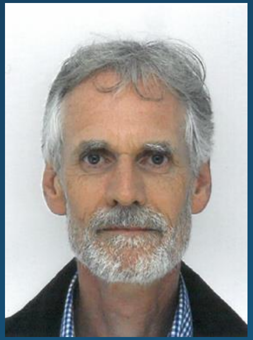
The Array IP was implemented on TSMC 65nm using Cadence Genus and Innovus tools. The table provided below offers insights into the diverse implementation runs conducted during the design phase, highlighting the evolution and refinement process of the Array IP. These iterative runs allowed us to fine-tune the IP's performance, power efficiency, and area utilization, ensuring that the final implementation met stringent design specifications.

Block Implementation Report

| Period(ns) | Frequency(MHz) | Area(um$^2$) | Power(mW) | PPA(mW/um$^2$) |
|---|---|---|---|---|
| 1.33 | 751.88 | 37662.48 | 38.72 | 1.03e-3 |
| 2.0 | 500 | 31237.56 | 18.78 | 6.01e-4 |
| 4.0 | 250 | 30402.36 | 8.49 | 2.79e-4 |
| 10.0 | 100 | 30020.04 | 3.29 | 1.09e-4 |

simply add milestones at any time, design flow steps can guide

add narrative describing your activities, especially in the education/collaboration track



Thank you for listening, questions?

we are here to help you on your journey