

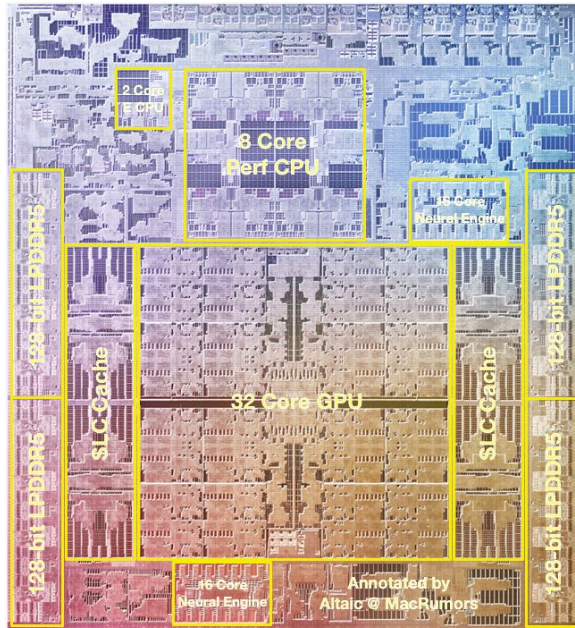
AKER

Safe and Secure SoC Access Control

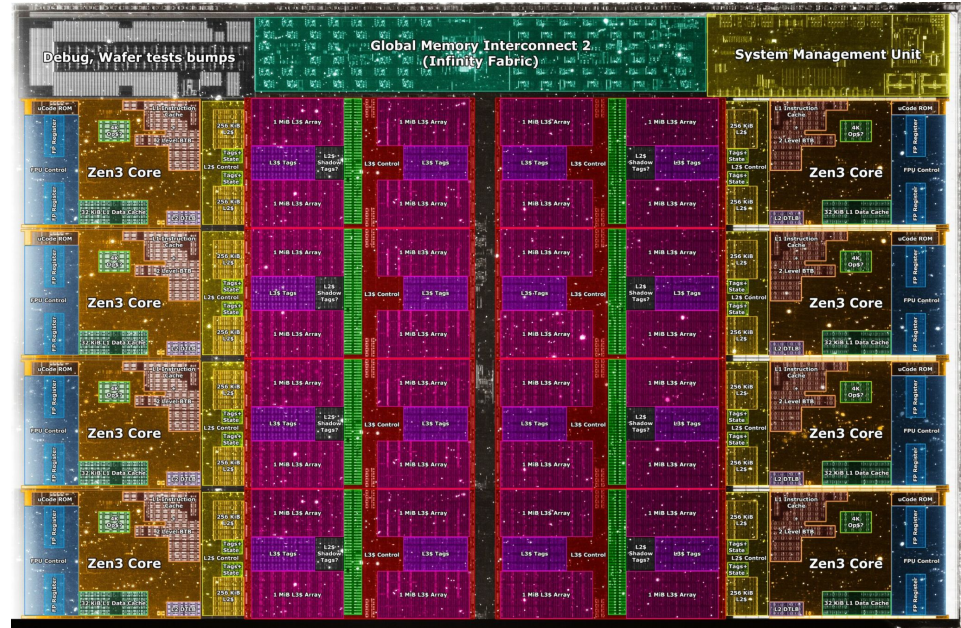
Chi Chow | Brandon Erickson | Hosein Yavarzadeh

One Last Time: SoCs and AKER's motivation

- What is an SoC?



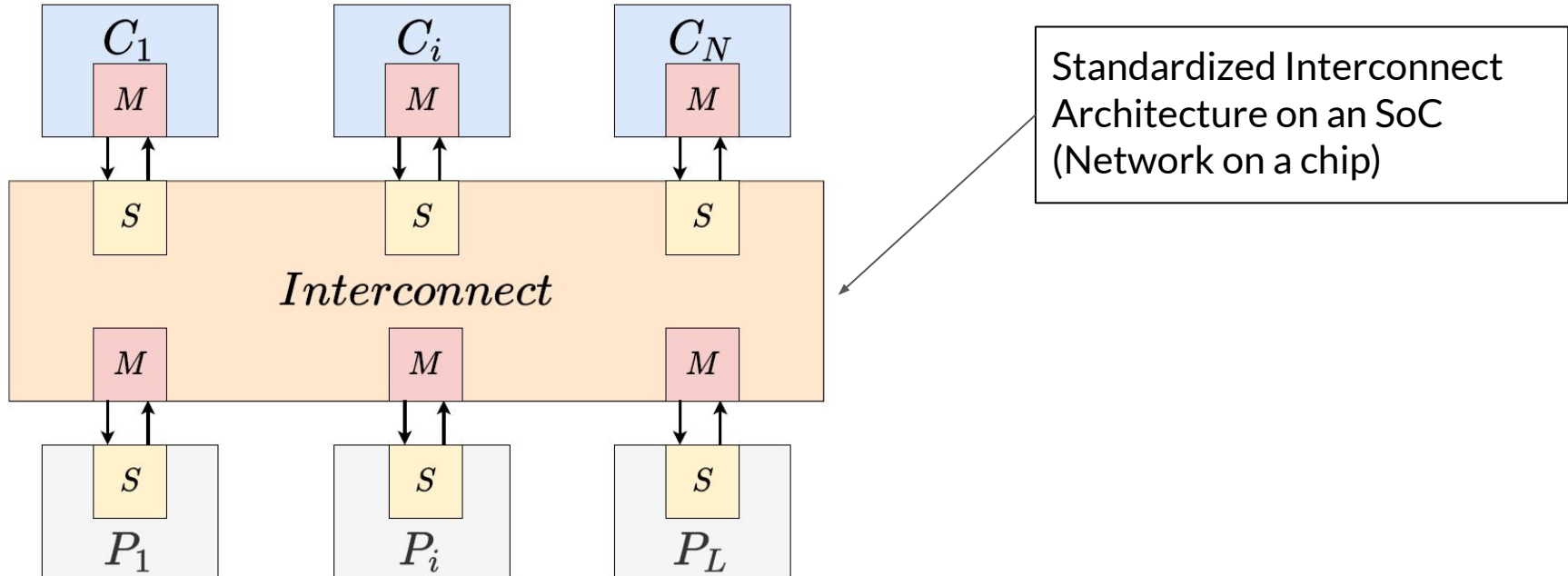
Apple M1 Max



AMD Ryzen 5 5600X

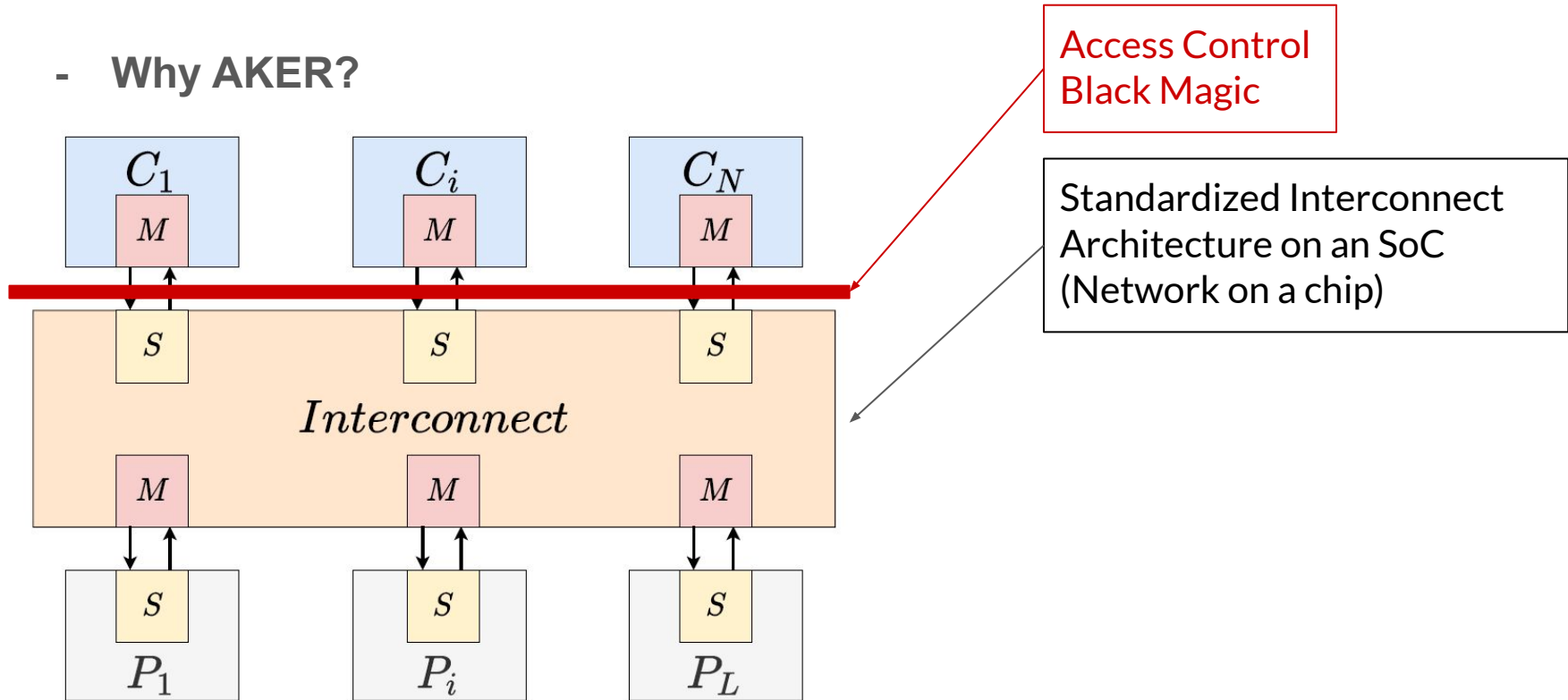
One Last Time: SoCs and AKER's motivation

- Why AKER?



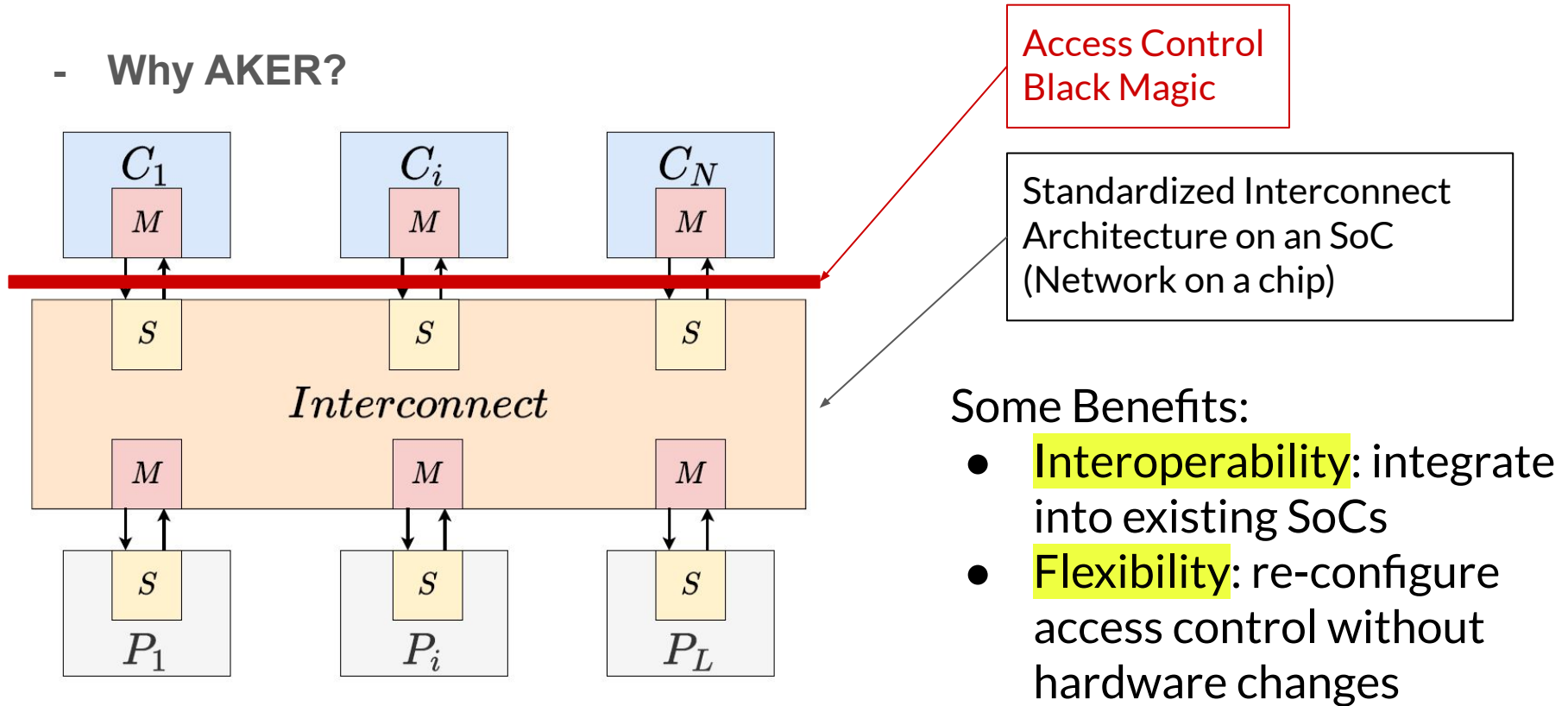
One Last Time: SoCs and AKER's motivation

- Why AKER?



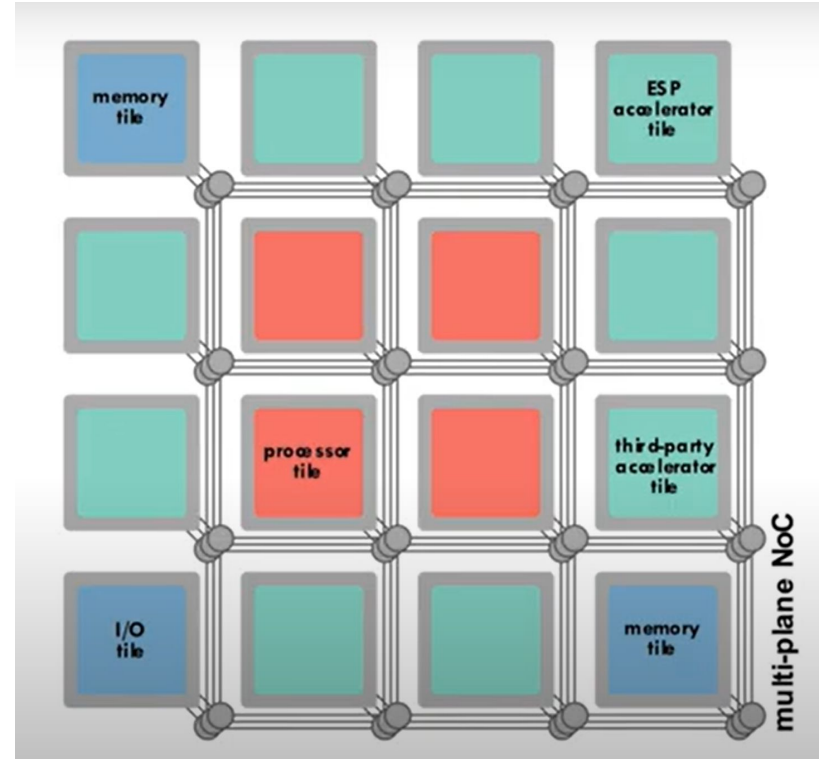
One Last Time: SoCs and AKER's motivation

- Why AKER?



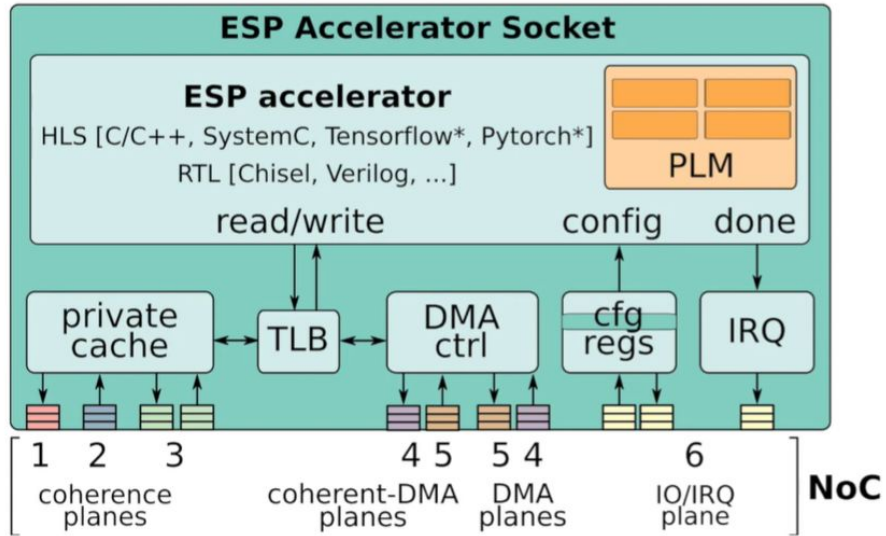
NoC Architecture Example: ESP

- An open-source platform for rapid SoC development.
- ESP **tiles** interface with the network via **sockets**.



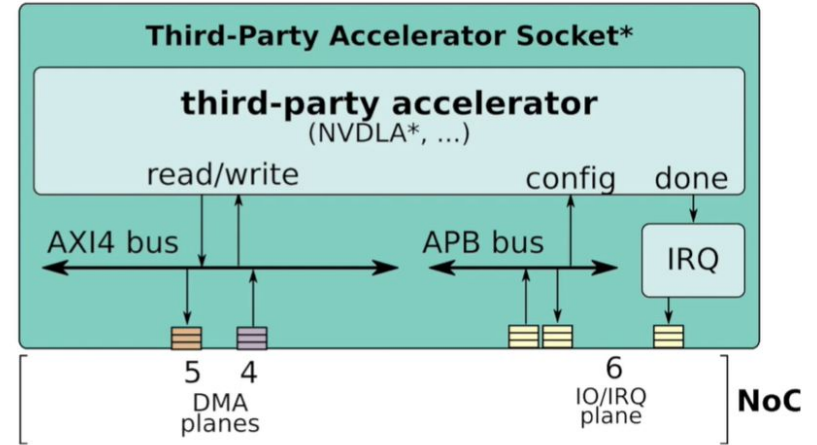
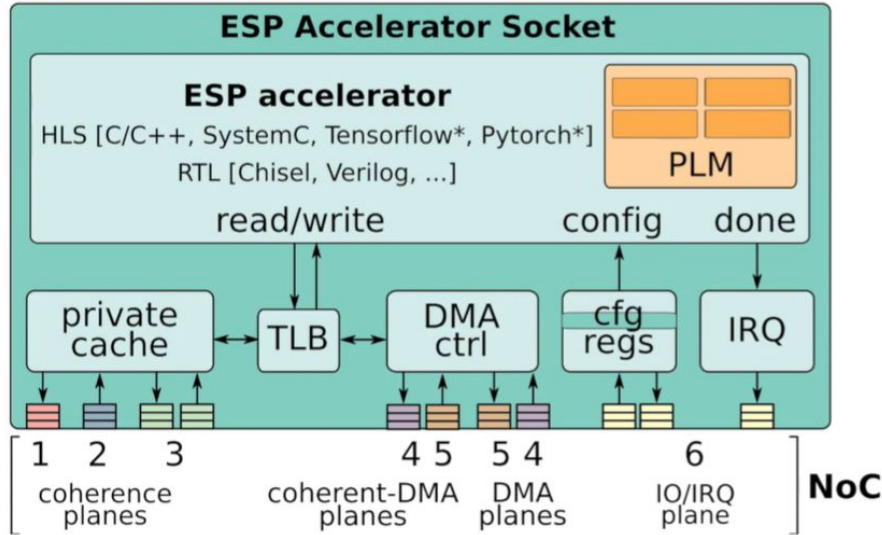
Source: <https://www.esp.cs.columbia.edu/docs/>

ESP: Socket Architecture



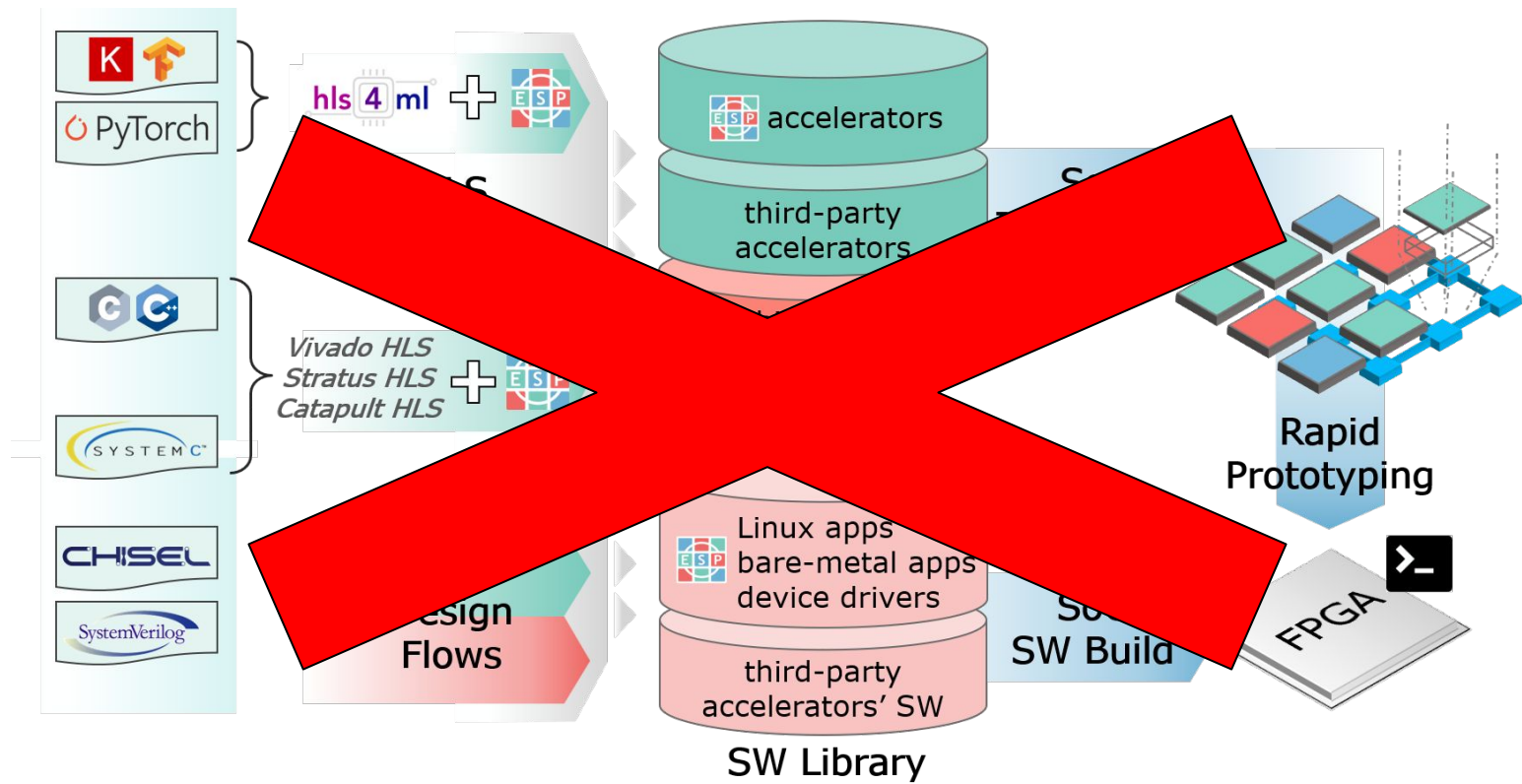
Source: <https://www.esp.cs.columbia.edu/docs/>

ESP: Socket Architecture



Source: <https://www.esp.cs.columbia.edu/docs/>

ESP Development Workflow



Actually very straightforward...

Step 1: Create Hardware Accelerator

- `esp/tools/accgen/accgen.sh`
- Configure resources
- For our use case, leave everything in default (we will see why later)
- Implement accelerator and import

Step 2: Configure SoC

ESP SoC Generator

General SoC configuration:
virtex7
SLD FPU
No JTAG
Eth (192.168.1.7)
Use SGMII
No SVGA
With synchronizers

CPU Architecture:
Core: leon3
Shared Local Memory:
KB per tile: 256
Data transfers:
☐ Bigphysical area
☒ Scatter/Gather

Cache Configuration:
Use Caches: ☒
Implementation: ESP RTL
L2 SETS: 512
L2 WAYS: 4
LLC SETS (per mem tile): 1024
LLC WAYS: 16
ACC L2 SETS: 512
ACC L2 WAYS: 4

NoC configuration
Rows: 2 Cols: 2
Config
☐ Monitor DDR bandwidth
☐ Monitor memory access
☐ Monitor injection rate
☐ Monitor router ports
☐ Monitor accelerator status
☐ Monitor L2 Hit/Miss
☐ Monitor LLC Hit/Miss
☐ Monitor DVFS
Num CPUs: 1
Num memory controllers: 1
Num local memory tiles using:
Num local memory tiles using:
Num IO tiles: 1
Num accelerators: 0
Num CLK regions: 1
Num CLKBUF: 0
VF points: 4

NoC Tile Configuration

(0,0) mem	(0,1) cpu
<input type="checkbox"/> Has cache <input type="checkbox"/> Has DDR Clk Reg: 0 <input type="checkbox"/> Has PLL <input type="checkbox"/> CLK BUF	<input checked="" type="checkbox"/> Has cache <input type="checkbox"/> Has DDR Clk Reg: 0 <input type="checkbox"/> Has PLL <input type="checkbox"/> CLK BUF
(1,0) empty	(1,1) IO
<input type="checkbox"/> Has cache <input type="checkbox"/> Has DDR Clk Reg: 0 <input type="checkbox"/> Has PLL <input type="checkbox"/> CLK BUF	<input type="checkbox"/> Has cache <input type="checkbox"/> Has DDR Clk Reg: 0 <input type="checkbox"/> Has PLL <input type="checkbox"/> CLK BUF

Generate SoC config

Step 2: Configure SoC

ESP SoC Generator

General SoC configuration:
virtex7
SLD FPU
No JTAG
Eth (192.168.1.7)
Use SGMII
No SVGA
With synchronizers

CPU Architecture:
Core: leon3
Shared Local Memory:
KB per tile: 256
Data transfers:
☐ Bigphysical area
☒ Scatter/Gather

Cache Configuration:
Use Caches: ☒
Implementation: ESP RTL
L2 SETS: 512
L2 WAYS: 4
LLC SETS (per mem tile): 1024
LLC WAYS: 16
ACC L2 SETS: 512
ACC L2 WAYS: 4

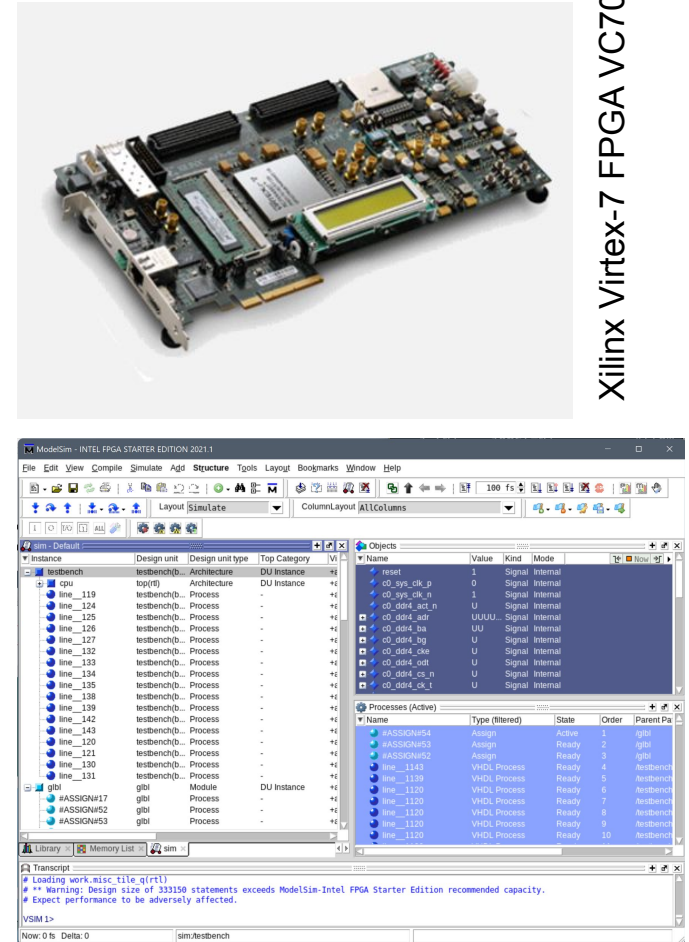
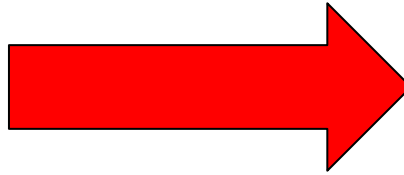
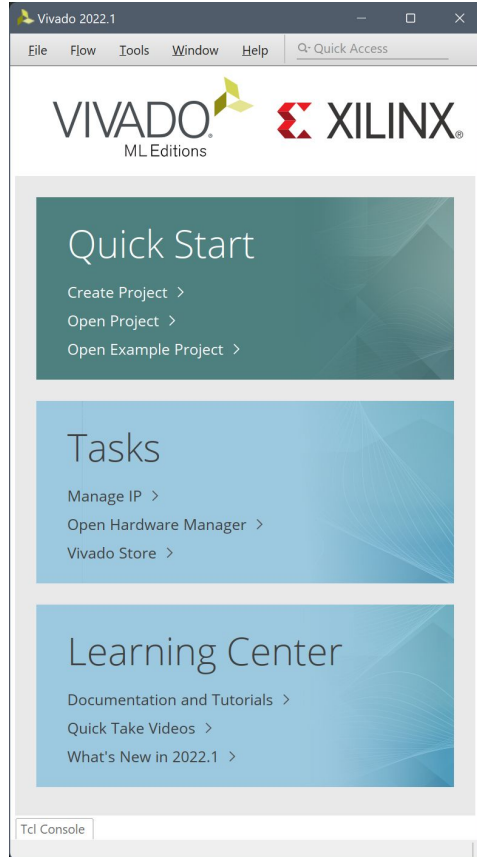
NoC configuration
Rows: 2 Cols: 2
Config
☐ Monitor DDR bandwidth
☐ Monitor memory access
☐ Monitor injection rate
☐ Monitor router ports
☐ Monitor accelerator statu
☐ Monitor L2 Hit/Miss
☐ Monitor LLC Hit/Miss
☐ Monitor DVFS
Num CPUs: 1
Num memory controllers: 1
Num local memory tiles using
Num local memory tiles using
Num IO tiles: 1
Num accelerators: 1
Num CLK regions: 1
Num CLKBUF: 0
VF points: 4

NoC Tile Configuration

(0,0) mem	(0,1) cpu
<input type="checkbox"/> Has cache <input type="checkbox"/> Has DDR Clk Reg: 0 <input type="checkbox"/> Has PLL <input type="checkbox"/> CLK BUF	<input checked="" type="checkbox"/> Has cache <input type="checkbox"/> Has DDR Clk Reg: 0 <input type="checkbox"/> Has PLL <input type="checkbox"/> CLK BUF
(1,0) DUMMY_RTL	(1,1) IO
<input type="checkbox"/> Has cache <input type="checkbox"/> Has DDR Clk Reg: 0 <input type="checkbox"/> Has PLL <input type="checkbox"/> CLK BUF	<input type="checkbox"/> Has cache <input type="checkbox"/> Has DDR Clk Reg: 0 <input type="checkbox"/> Has PLL <input type="checkbox"/> CLK BUF

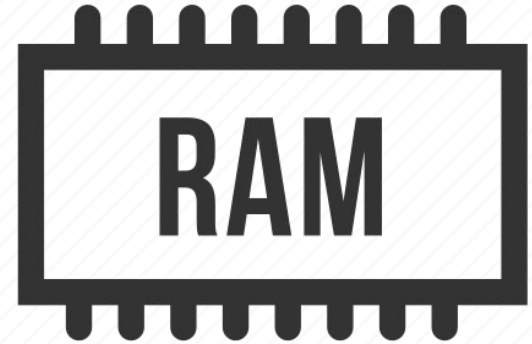
Generate SoC config

Step 3: Xilinx Vivado and Modelsim



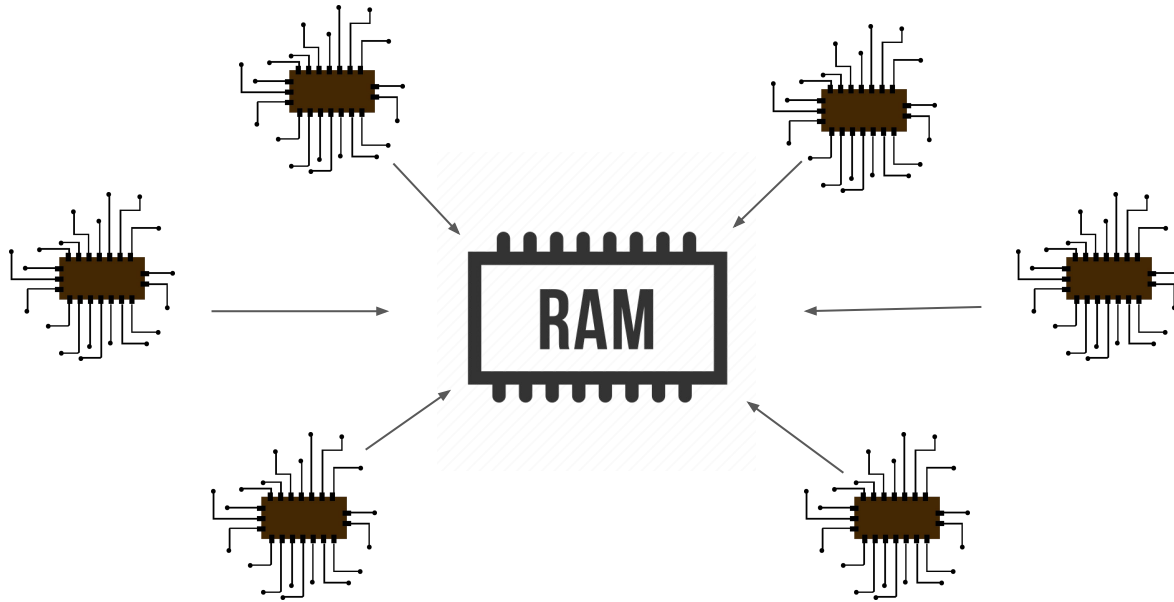
AKER: Safe and Secure SoC Access Control

Processing units store data in storage devices like RAMs.

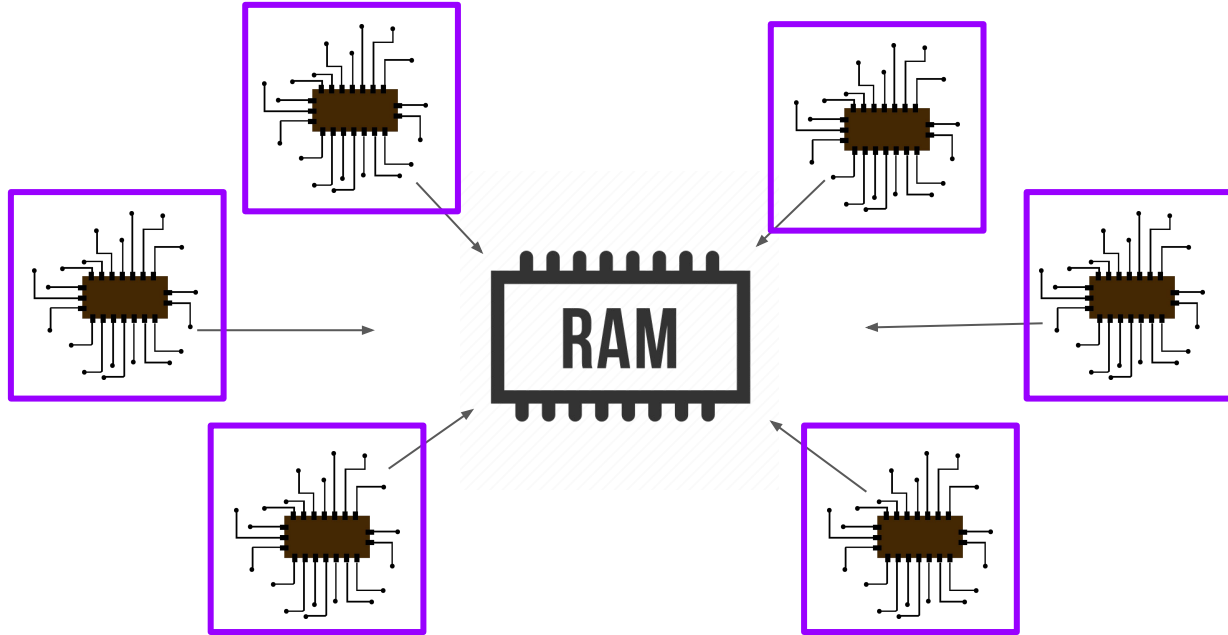


AKER

Storage Devices are shared between various processing units.



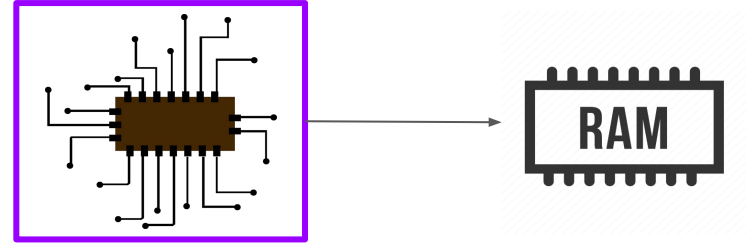
AKER: Access control wrapper



AKER: Access Control Wrapper (ACW)

Simplified version of the ACW:

```
1  Bool Addr_Check (int id, int address) {  
2      if ( !trust_id(id) ) {  
3          return false;  
4      }  
5      else {  
6          if ( address < boundary ){  
7              return true;  
8          }  
9          else {  
10             return false;  
11         }  
12     }  
13 }
```



Challenges:

- Implementation on top of ESP
- Consistency

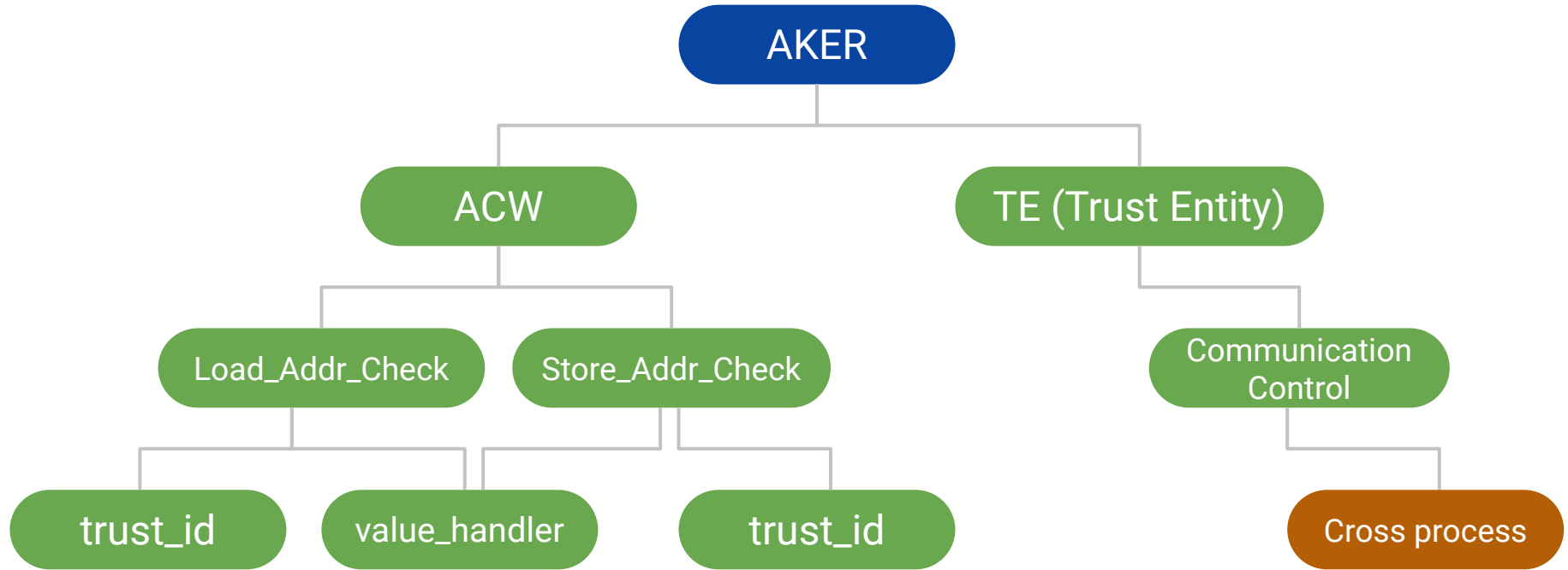
AKER: Access Control Wrapper (ACW)

Simplified version of the ACW:

```
17  Bool trust_id (int id) {  
18      if (id_decoder(id)!=privileged){  
19          return false;  
20      }  
21      return true;  
22  }
```

AKER on ESP! (what did we do)

Implemented Functions:



What's next?

- “Communication Control” part >> debug!
- Security Evaluation (ongoing)
- Performance Evaluation (ongoing)

Week	Goals	Deliverables
5	Implementation of AKER on the ESP platform (C, C++, Verilog, etc.) <ul style="list-style-type: none">• Build target ESP architecture with NoC, routing protocol• Develop compatible AKER access control module• Integrate AKER + ESP platform	
6		
7		Milestone Update Presentation/Report (5/17)
8		
9	Testing and Validation	
10		Final Video and Report (week of 6/5)

Last Question?

Are all vulnerabilities in SoCs eliminated by AKER?