Hardware Architectures for Post Quantum Security

A Silicon Root-of-trust Approach

Guided by Professor Dr. Jawar Singh

Presented By:
Abhraneel Saha (2101EE96)
ParthaSarathi Dutta (2101EE50)

Recap

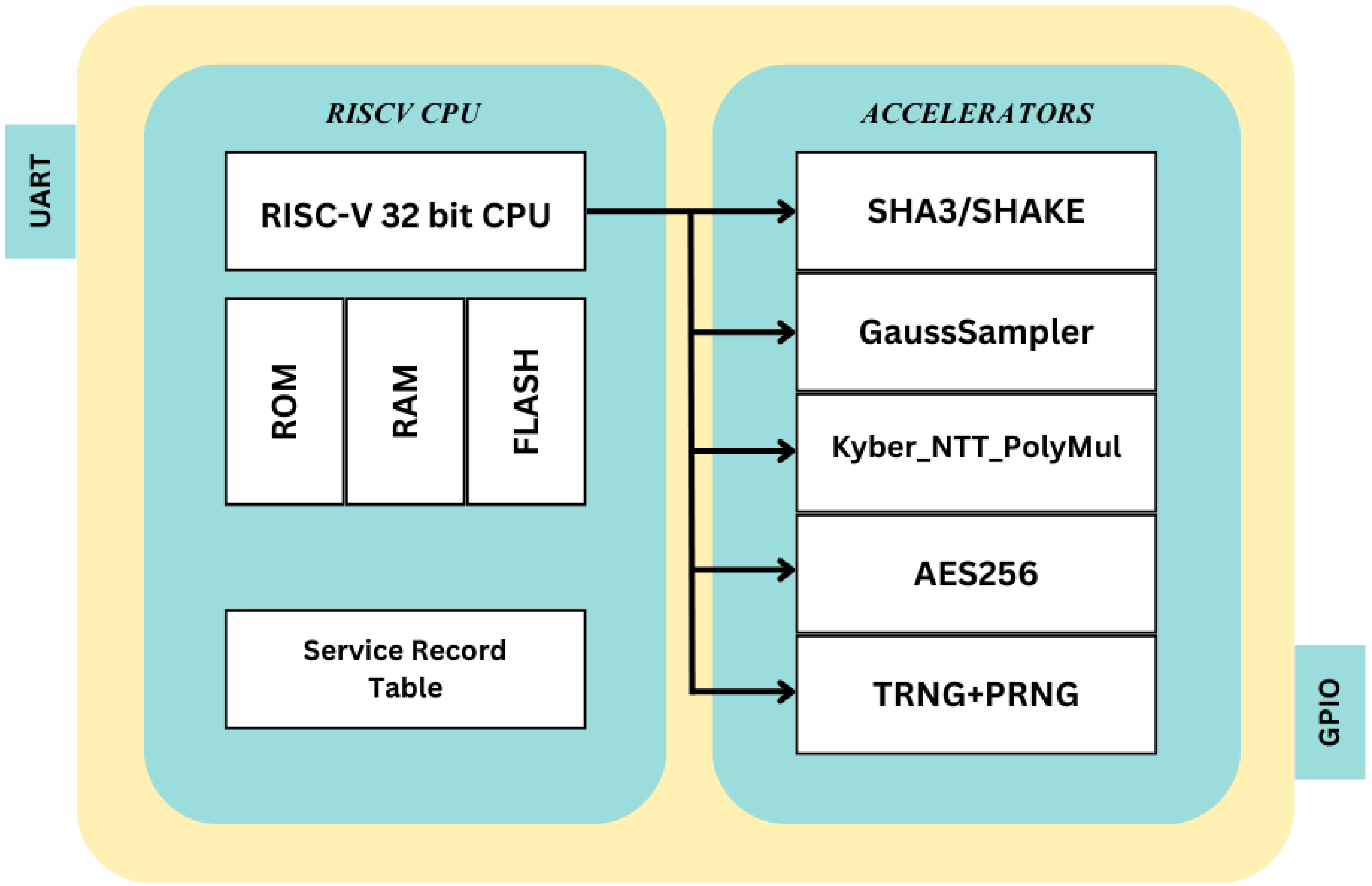


Figure 1 : Silicon Root of Trust

We adopted a software-hardware co-design approach to develop a robust silicon root of trust, delivering a secure and high-performance environment for executing cryptographic and network operations seamlessly.

Recap

We have created three levels of abstraction to implement CRYSTALS-KYBER algorithm. This ensures clearity in code and security

Custom data-structure “Rq” which enables us to operate in the “Ring” Polynomial domain

Kyber.cpp takes care of generating Public and secret keys. Helps to perform sofware control over hardware accelerators

KyberProcesses.cpp takes care of managing encapsulation, decapsulation and network management of keys (transmitting, recieving, and storing)

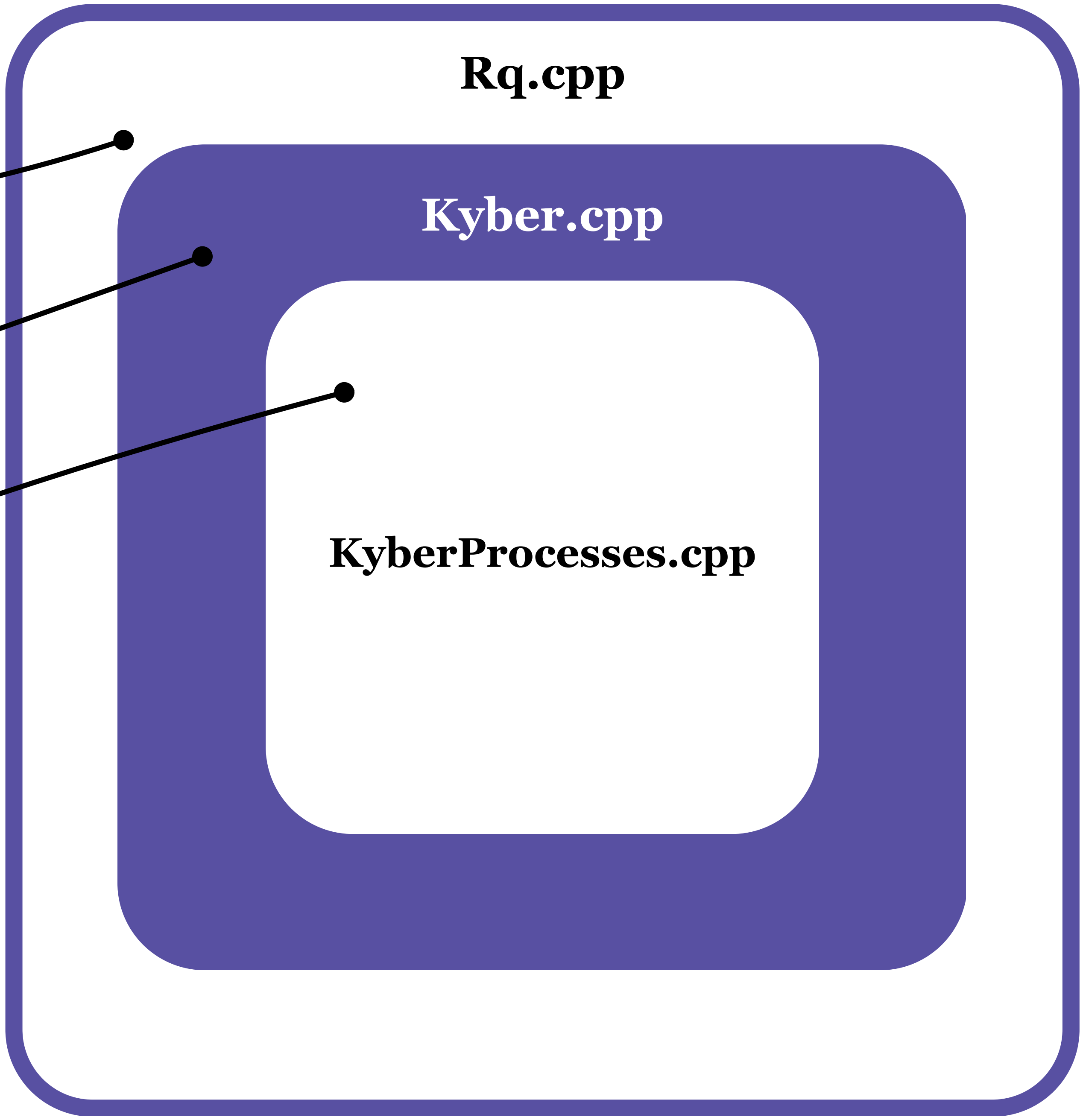
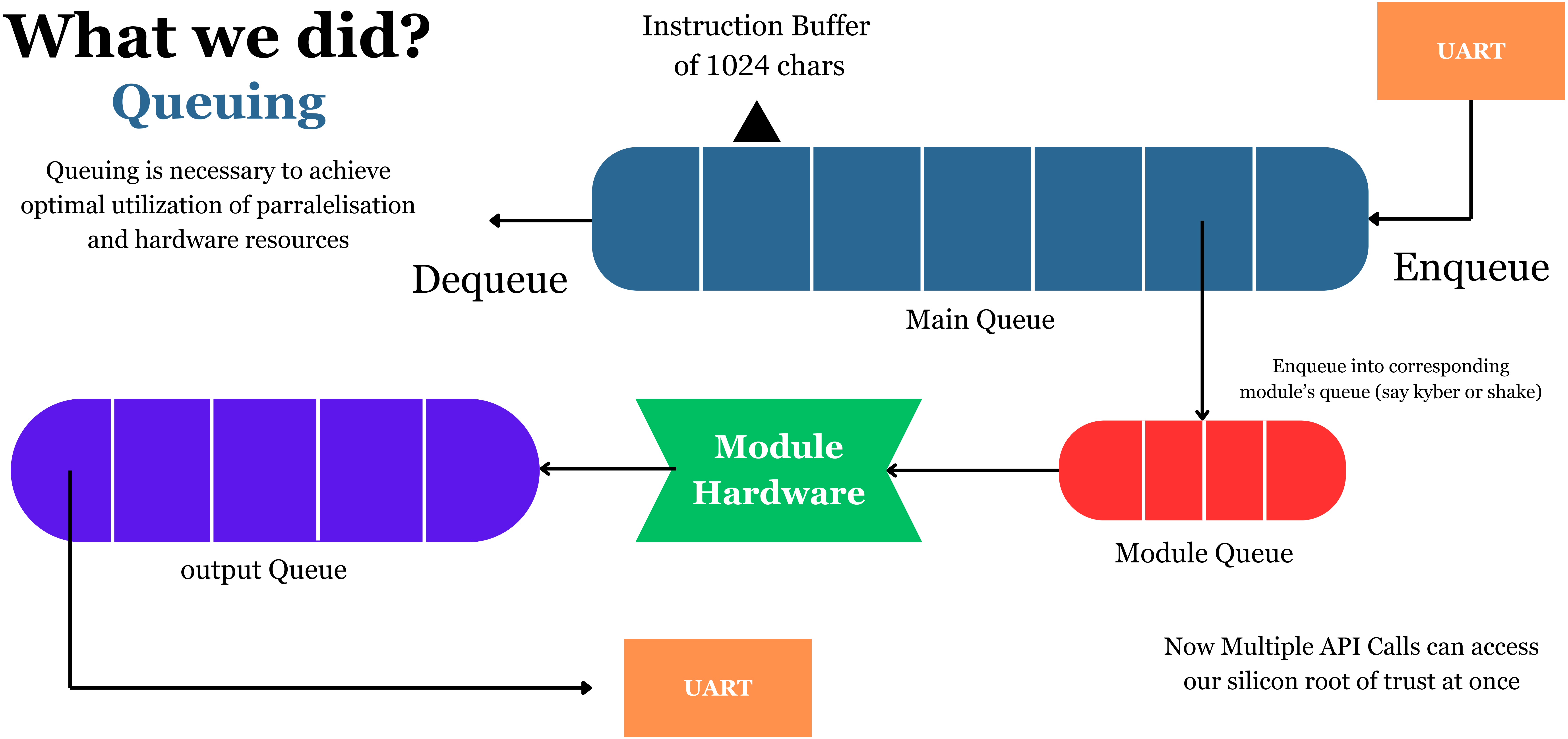


Figure 2: Kyber Software Abstraction

What we did?
Queuing

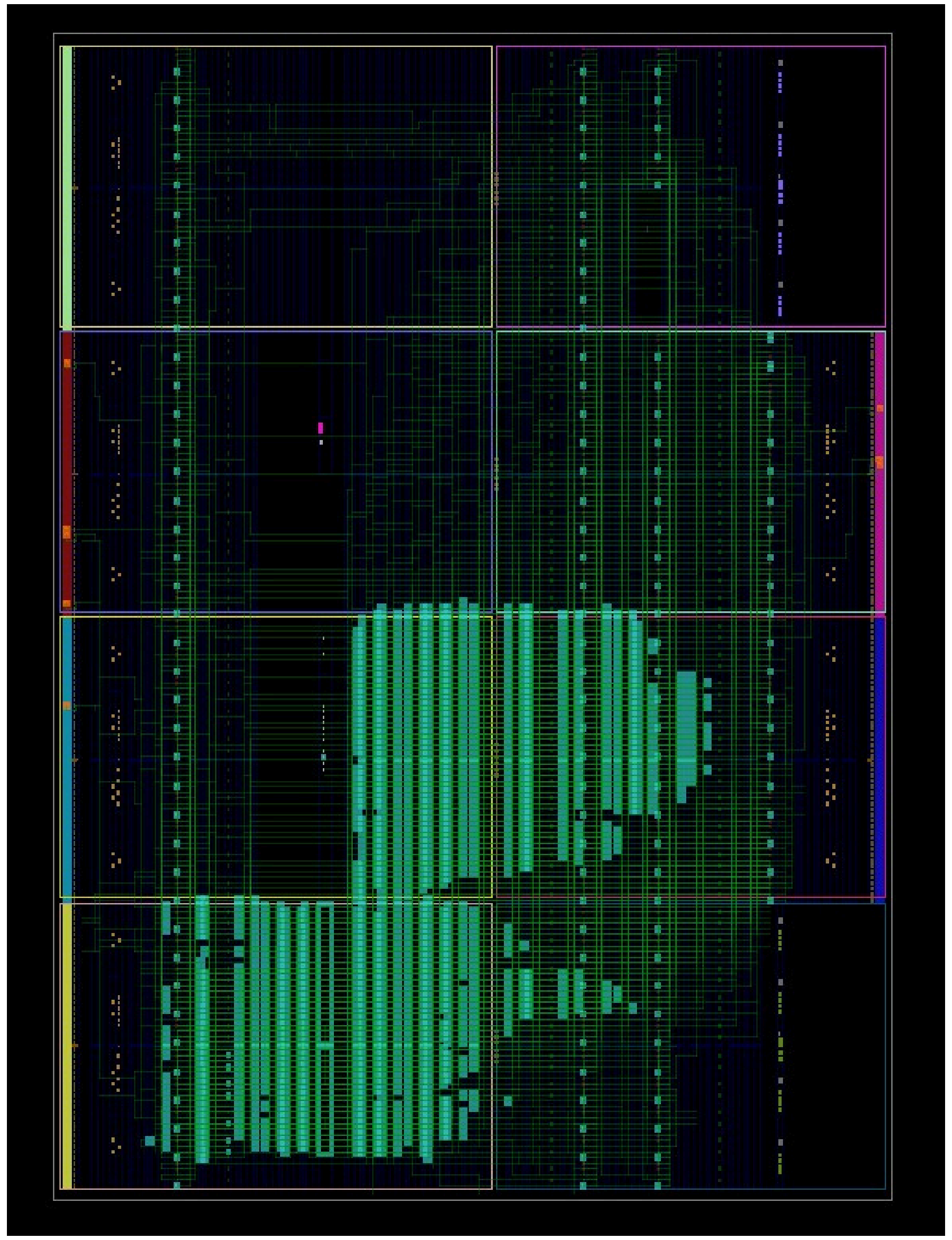
Queuing is necessary to achieve optimal utilization of parralelisation and hardware resources

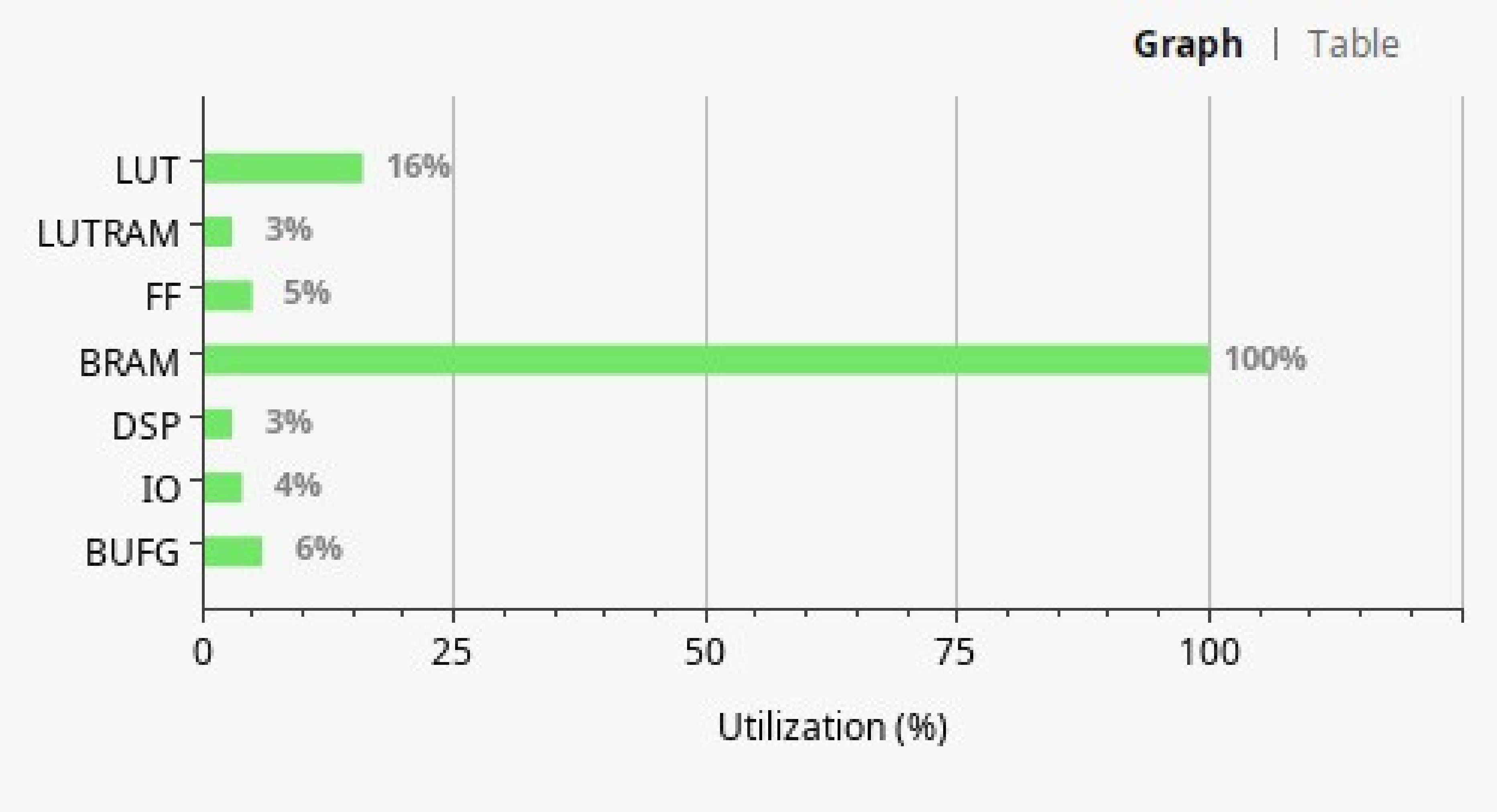


Now Multiple API Calls can access our silicon root of trust at once

What we did?

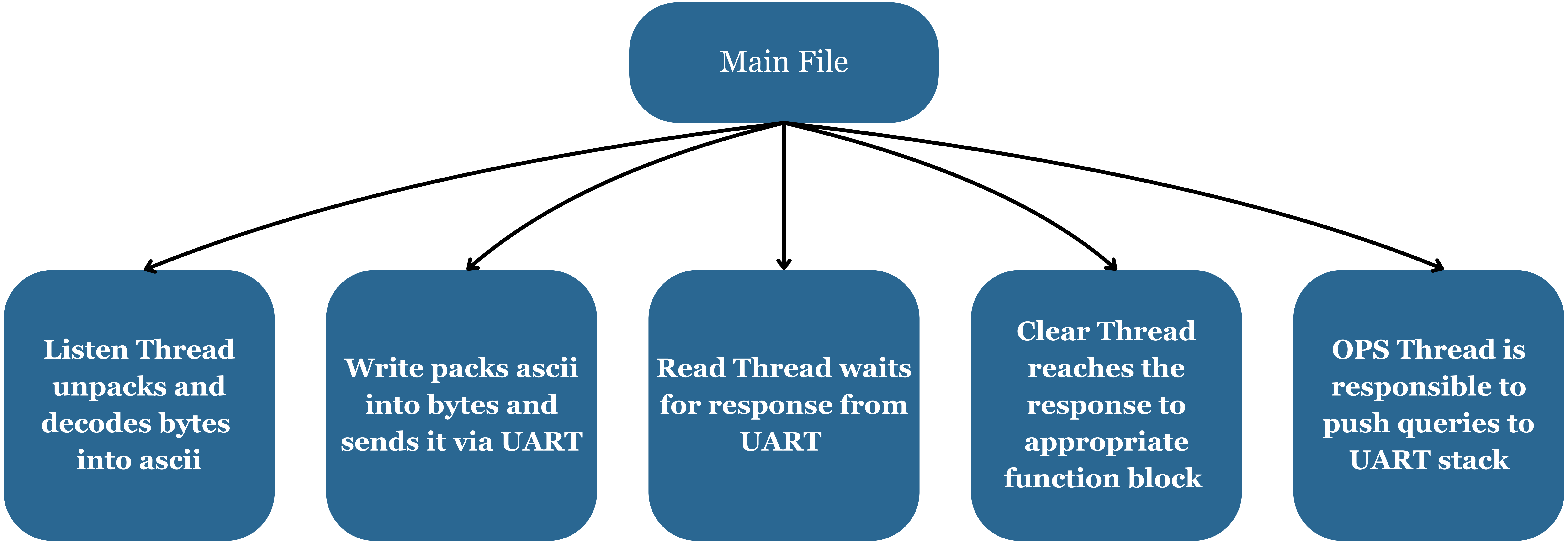
Final RISC-V CPU Resource consumption





Memory region	Used Size	Region Size	%age Used
RAM:	242280 B	400 KB	59.15%

Main.py file (API)



For every call to main file, a session is created

Demo.....

How can we increase speed? (Further work)

Faster Clocking

Hardware is operating at 100MHz, design can be optimised for till 1GHz

Introducing threading in hardware

By Using RTos for scheduling

Stacking SiRTs

Multiple SiRTs can be used for faster operations

Use a faster language for API dev (like C++)

Python is a slow language

How can we increase speed? (Further work)

on is a slow language

Faster Cloc

NOTE

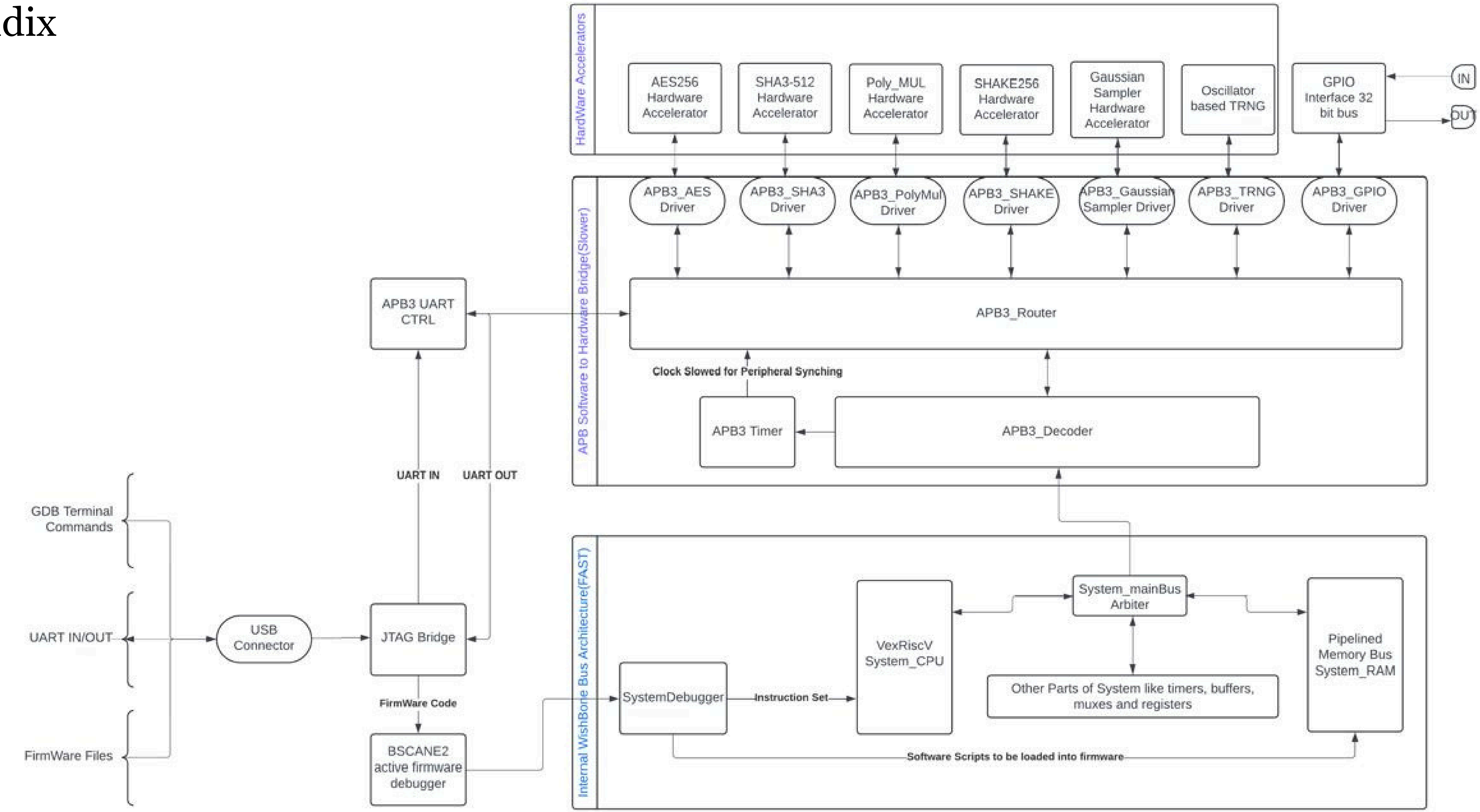
Assuming constant software optimization and other factors, the relationship between clock speed and operation time would be inversely proportional. Hence, a operation time of 3.5sec at 100MHz is proportional to 70ms at 5GHz (benchmark of kyber768 keygen is 294ms

se a faster language
or API dev (like C++)

Hardware is operating at
design can be optimised f

Thank You

Appendix



HardWare Accelerators
AES256 Hardware Accelerator
SHA3-512 Hardware Accelerator
Poly_MUL Hardware Accelerator
SHAKE256 Hardware Accelerator
Gaussian Sampler Hardware Accelerator
Oscillator based TRNG
GPIO Interface 32 bit bus
IN
OUT
APB Software to Hardware Bridge(Slower)
APB3_AES Driver
APB3_SHA3 Driver
APB3_PolyMul Driver
APB3_SHAKE Driver
APB3_Gaussian Sampler Driver
APB3_TRNG Driver
APB3_GPIO Driver
APB3_Router
Clock Slowed for Peripheral Synching
APB3 Timer
APB3_Decoder
APB3 UART CTRL
UART IN
UART OUT
GDB Terminal Commands
UART IN/OUT
FirmWare Files
USB Connector
JTAG Bridge
FirmWare Code
BSCANE2 active firmware debugger
Internal WishBone Bus Architecture(FAST)
SystemDebugger
Instruction Set
VexRiscV System_CPU
System_mainBus Arbiter
Other Parts of System like timers, buffers, muxes and registers
Pipelined Memory Bus System_RAM
Software Scripts to be loaded into firmware