life.augmented

SoC Design & Architecture

Pushkar Sareen

Pushkar Sareen
SoC Design Architect
General purpose Microcontrollers (GPM) - MDG,
STMicroelectronics

Pushkar is currently engaged in Design Architecture activities for General Purpose Micro (GPM) products in ST, Greater Noida. Previously, he was working in Qualcomm in IP architecture team for Multimedia Systems and prior to that in Frontend and Performance Analysis team in NXP. He also has experience in SoC Verification activities during his time in Texas Instruments. His key interest is Performance Architecture and has driven many performance enhancement initiatives in various SoCs. He has 2 patents and 6 publications in the field on Security and SoC Performance Analysis.

Agenda

- # Case Study: Security Camera Optimizing for Performance and Area/Cost
  - # Building Blocks
    - # Camera, DDR
    - # NVM,SRAM, Peripherals
  - # Sample application
    - # Data flow evolution What-ifs

Architecture

- # Integration flow
  - # Documentation
    - # IP Selection Parametrization
    - # Integration
  - # Sign Offs

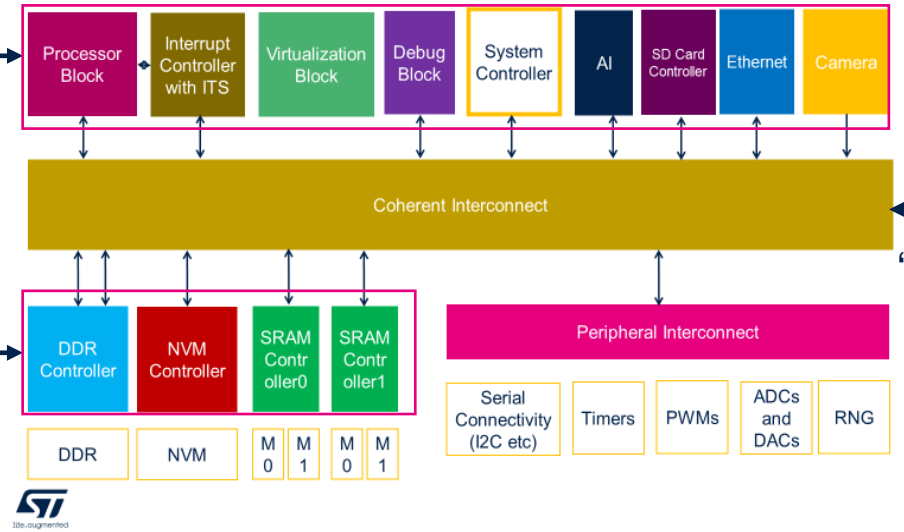
Frontend Design

life.augmented

Terms

- Initiators/Masters
  They ‘initiate’ transactions and wait for the data or response

- Slaves
  They receive the ‘commands’ of a transactions and respond with data or response

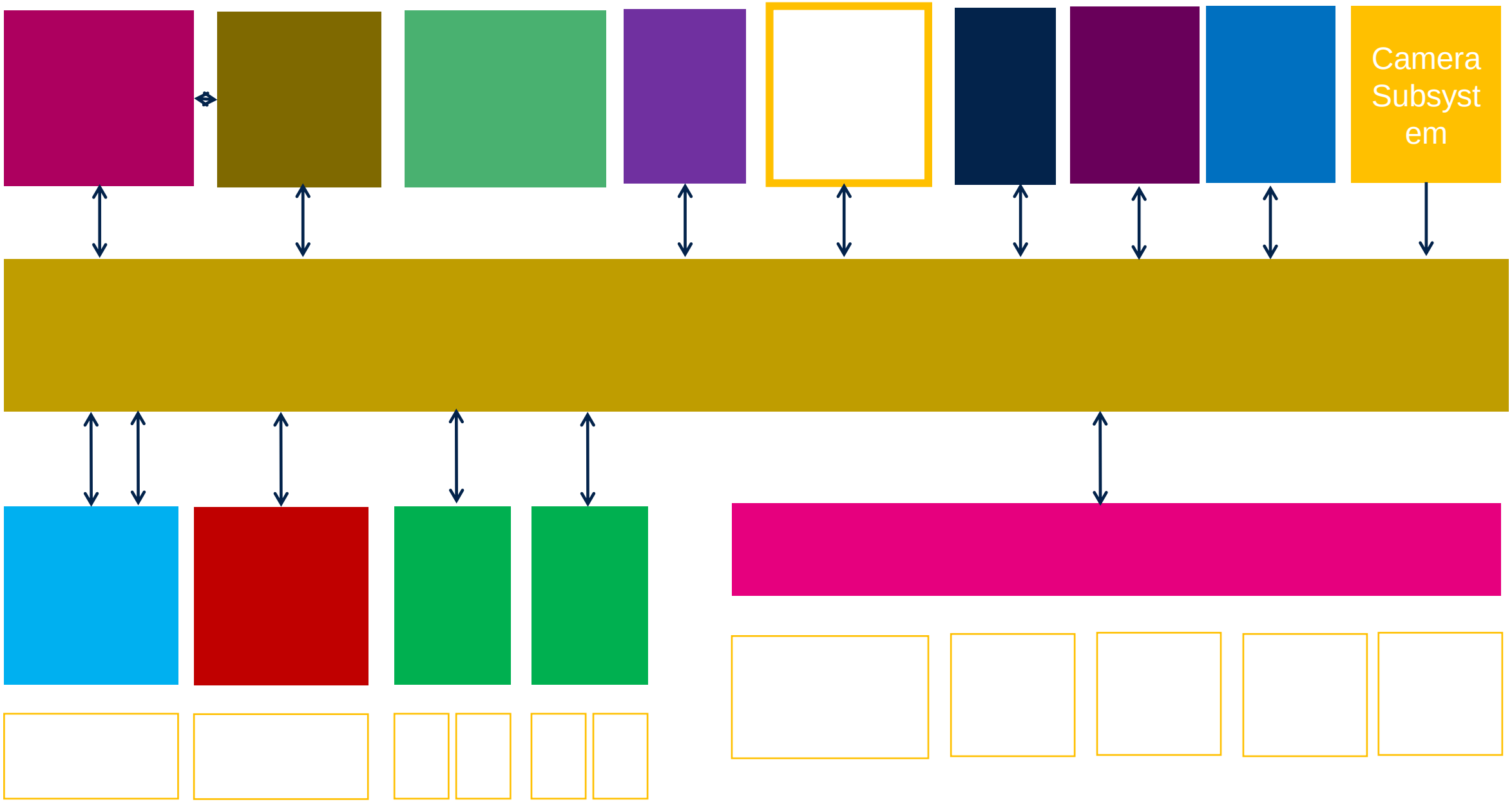


- Interconnect
  ‘Routes’ transaction **commands** Masters → Slaves

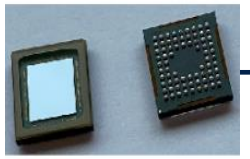
&

‘Routes’ transaction **data** or **response** Slaves → Masters

Transaction consists of a **Command** and a corresponding **Data or a Response**



life.augmented



Sensor

Camera Subsystem

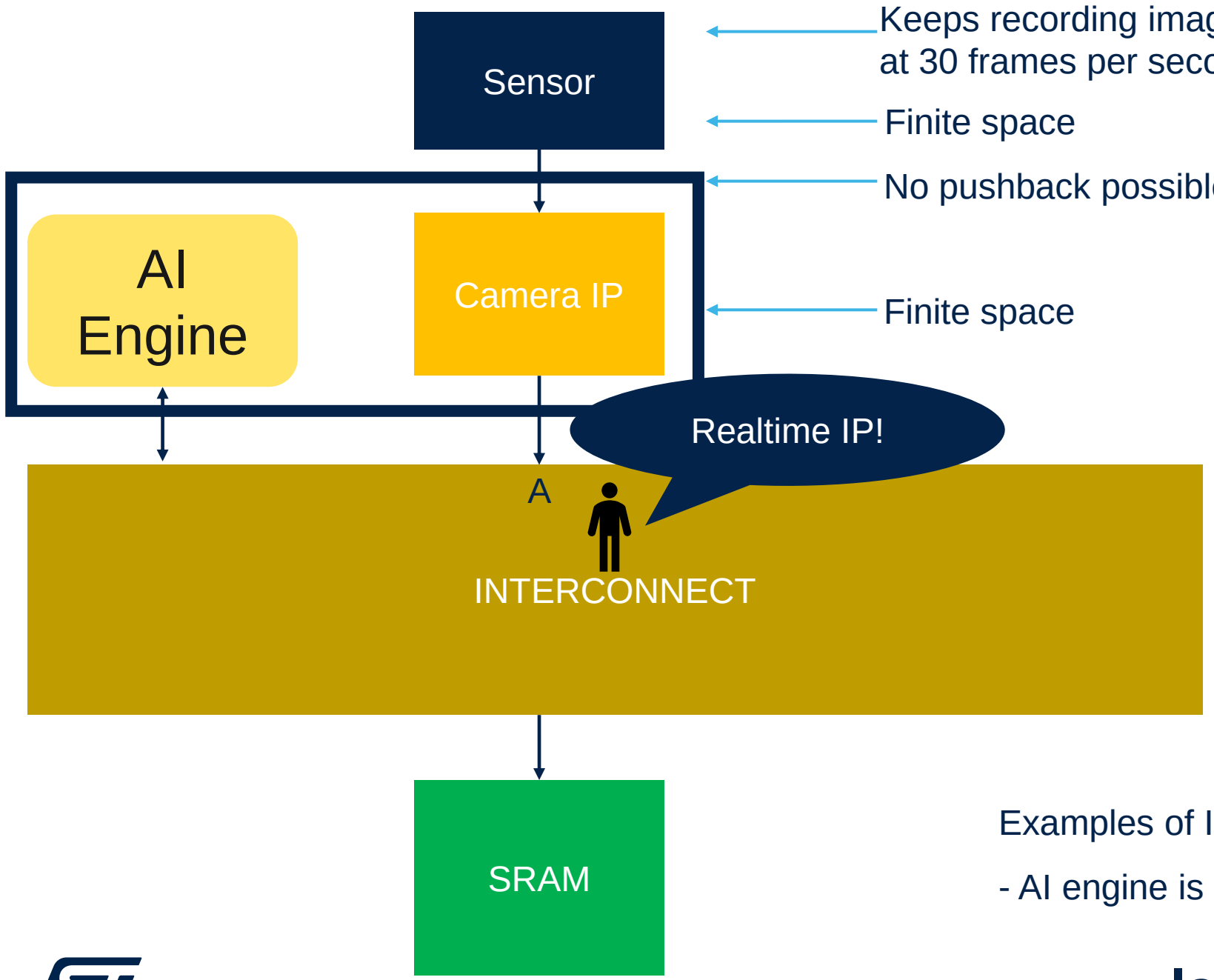
Camera IP

AI Engine

MEMORY

life.augmented

Camera IP



Features:

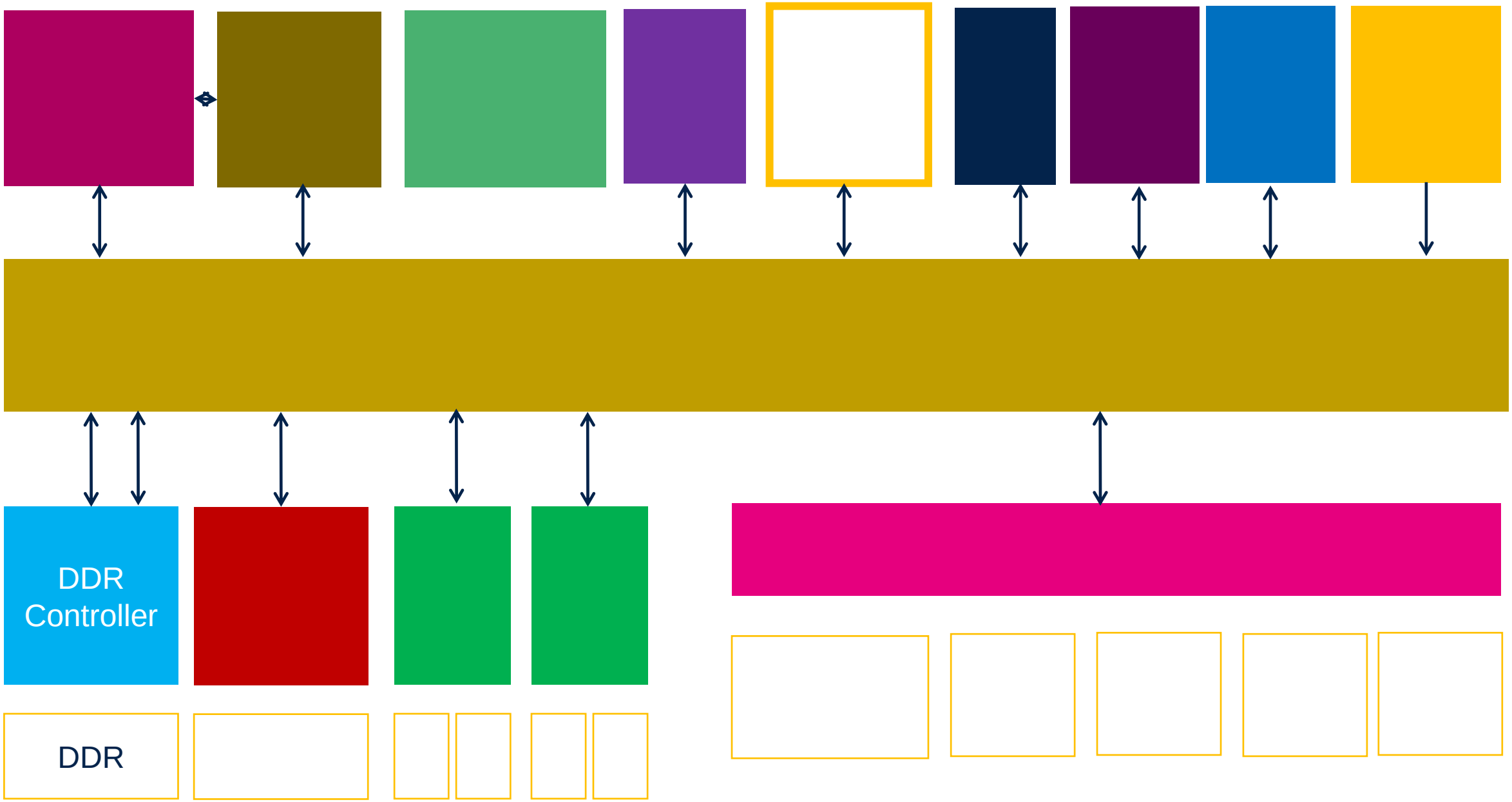
- Converts Bayer Pattern to RGB
- Pixel decompression
- Scaling
- Removes Noise

Examples of IPs which are **not** real time:

- AI engine is a **bandwidth critical** IP

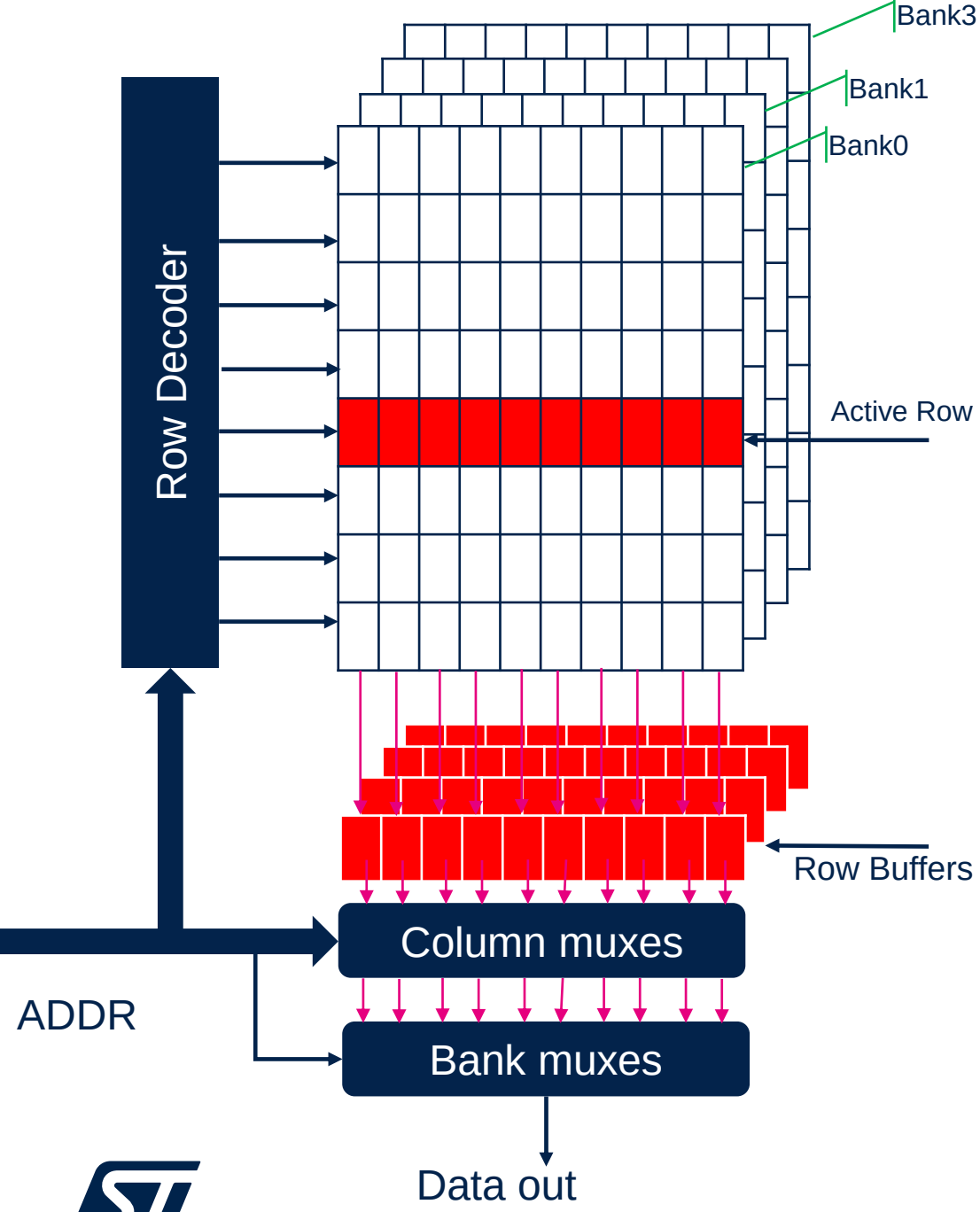
- CPU is a **latency critical** IP

life.augmented



life.augmented

DDR and its Controller
Performance aspects



DDR is a type of SDRAM (Synchronous Dynamic Random Access Memory)

A row usually can hold hundreds of transaction worth of data

READ requires two operations:

1) **open a row in a bank [if not already open]**
2) **Selecting data from row buffer**

WRITE requires three operations:

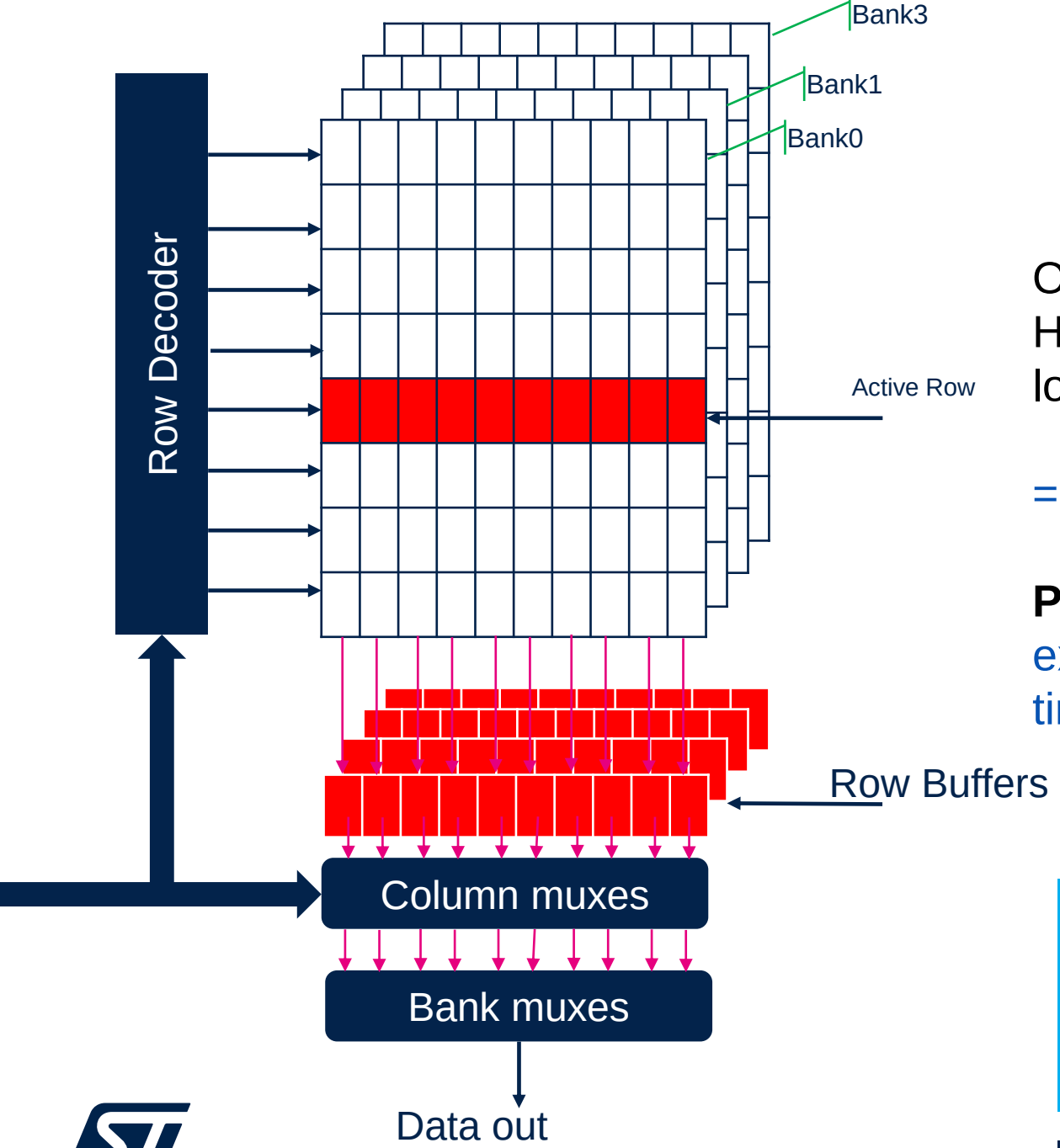
1) **open a row in a bank [if not already open]**
2) **Updating row buffer**
3) **later writing row buffer back to the bank (closing).**

House keeping:

Capacitors loose charge overtime because of leakage current. These cells have to be **refreshed** at a certain frequency

life.augmented

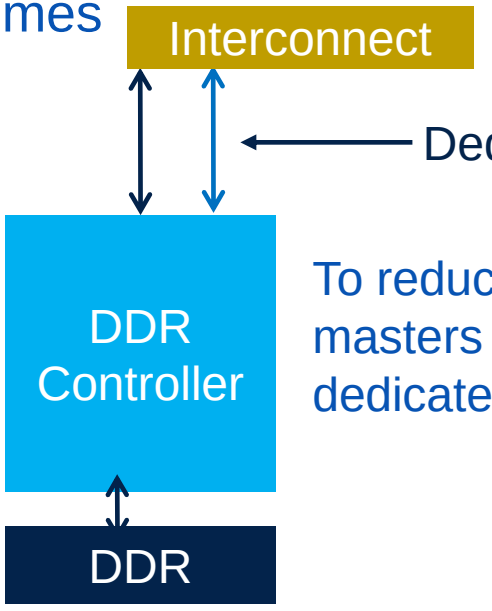
DDR and its Controller
Performance aspects



Opening a Row Buffer is a time consuming operation. Hence, DDR controllers want to keep the row open as long as they anticipate a request to the same row.

=> Requests to DDR will no longer be serviced in order

Performance Implication: Masters would see exceptionally large and exceptionally variable response times



To reduce latency, latency critical masters may be provided with dedicated transaction paths

life.augmented

DDR PowerPerfArea Summary

+ Much denser than SRAM hence saves

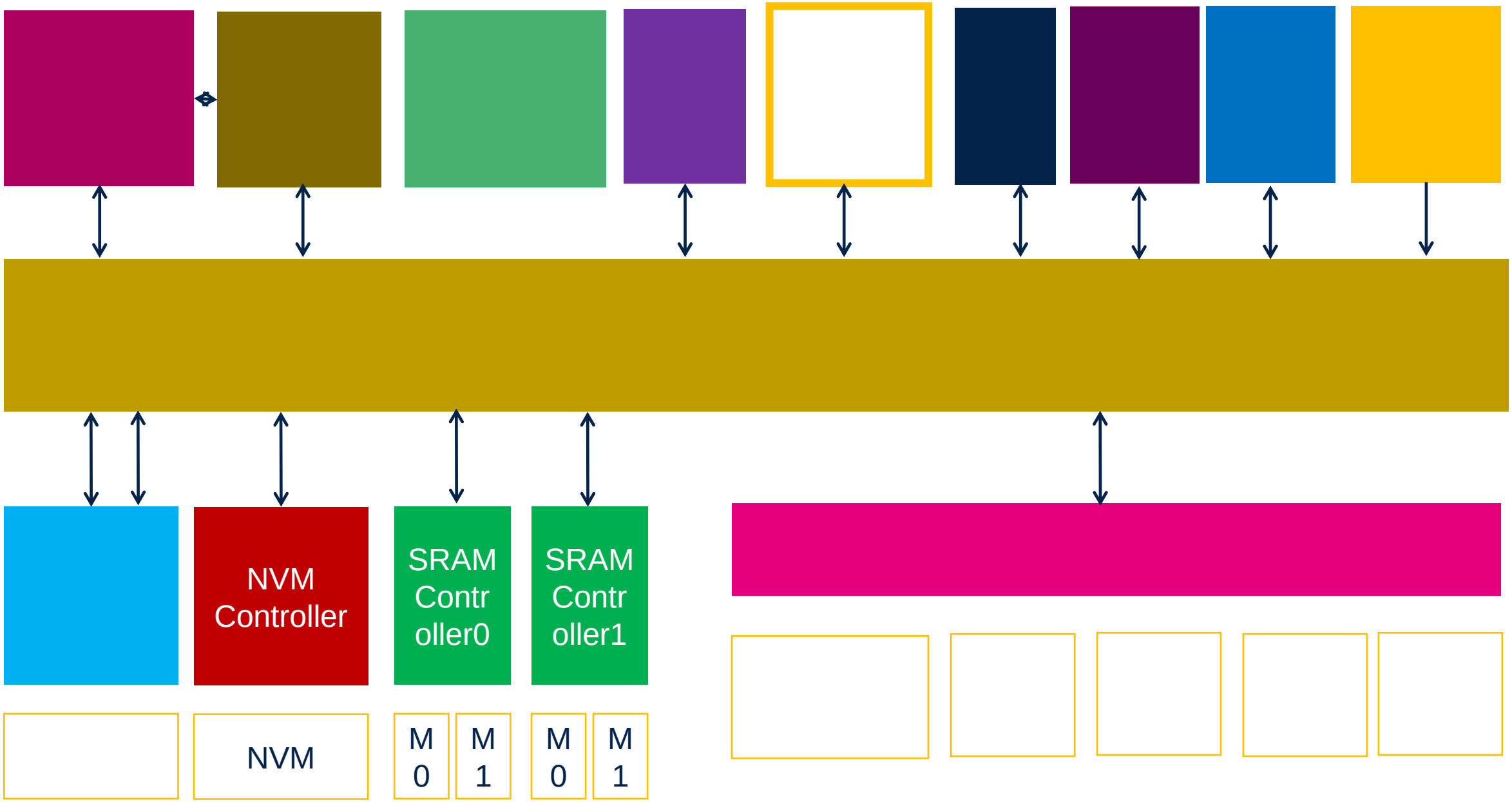
- Larger peak access times

- Consumes more power

- Lower available Bandwidth

Mr. Money

Mr. Performance



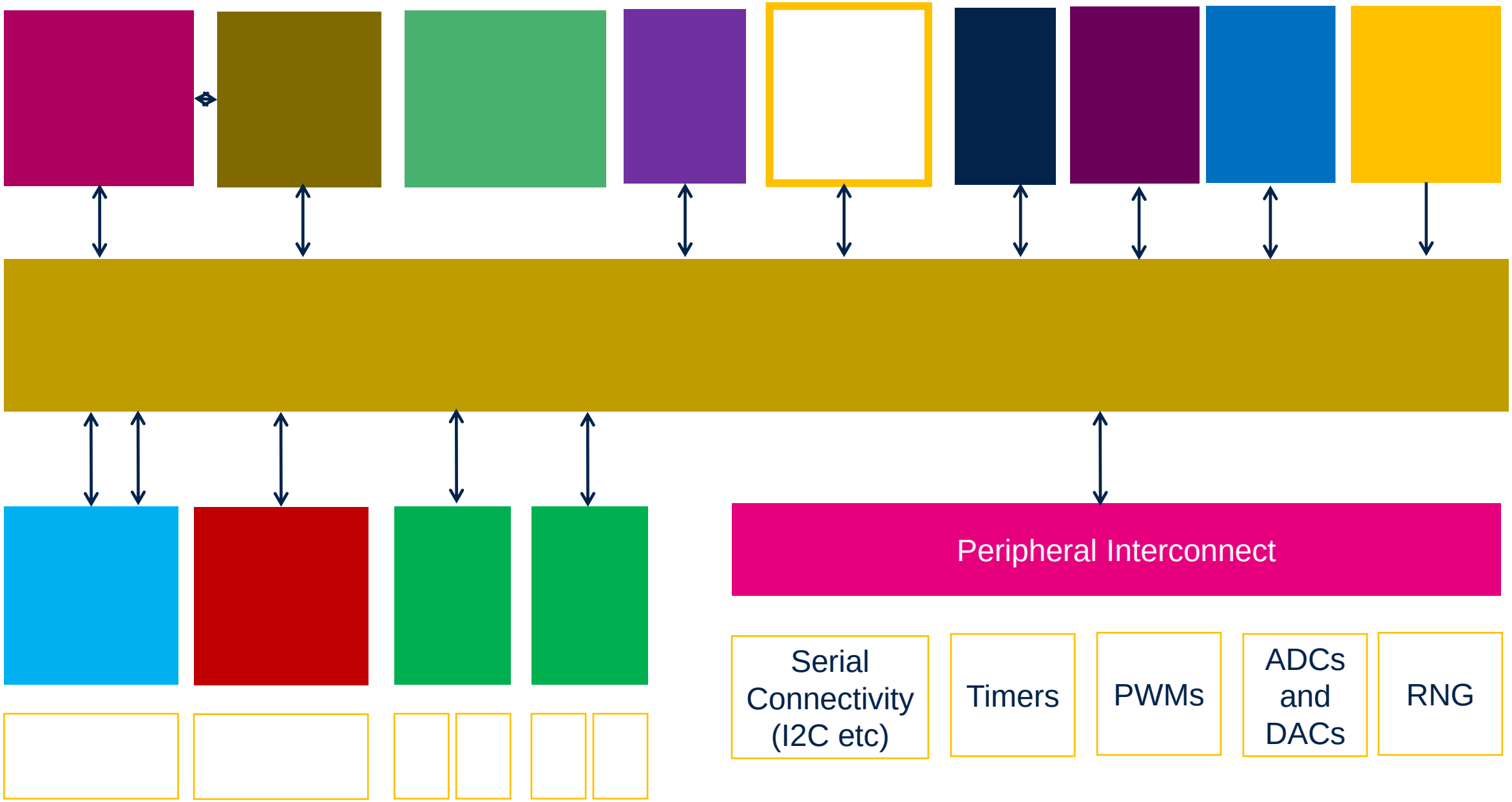
life.augmented

Non Volatile Memory

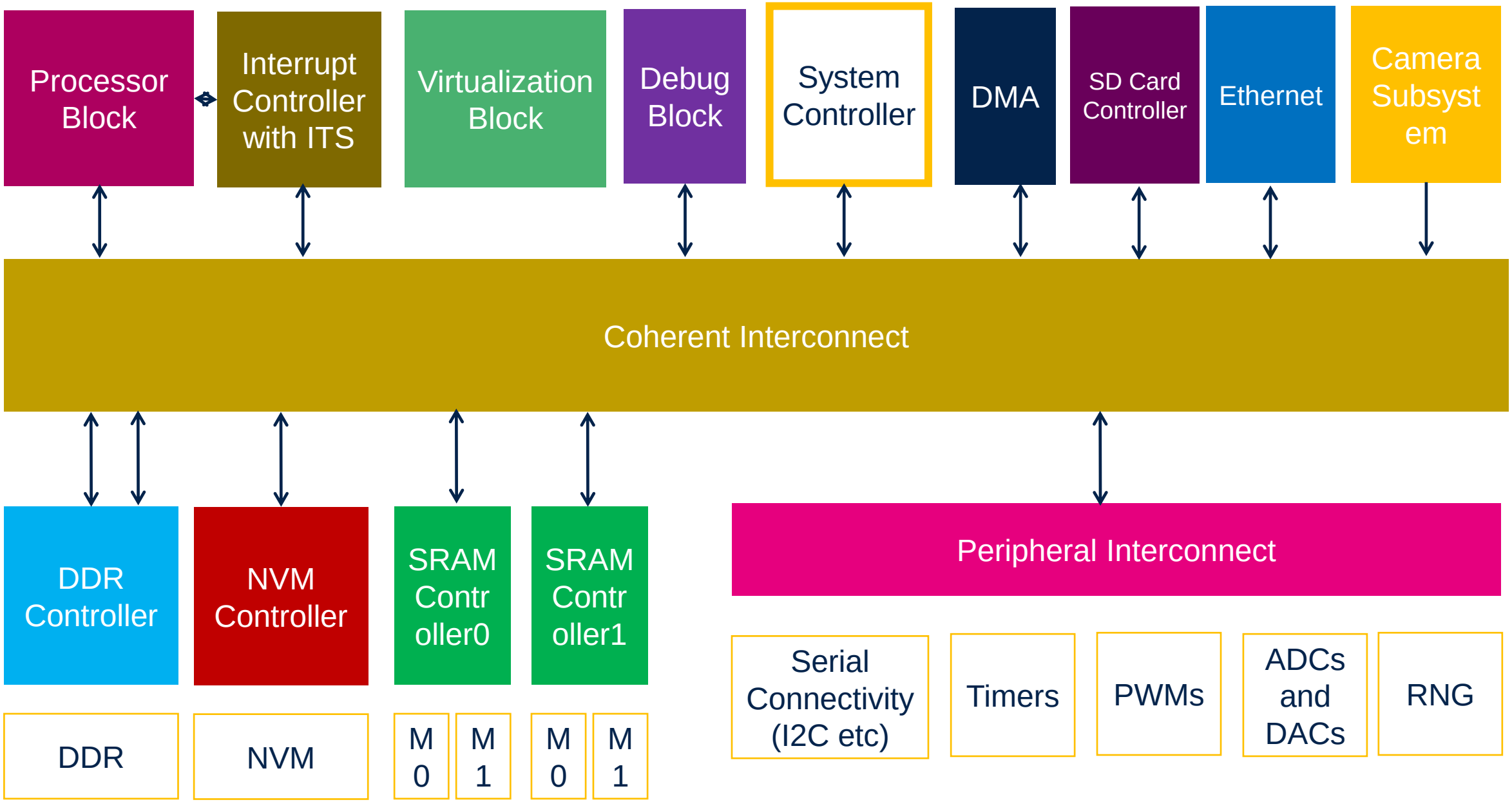
- Good to hold code and static data
- Various prevalent technologies are there : PCM, RRAM and MRAM
- Typically, EEPROM emulation support is there to store data which needs to be retained across power cycle
- Writes are slow, read is decent

Static Random Access Memory

- SRAM cell is made of made of Multi Transistor cross-coupled inverters
- Used in CPU Cache’s, IP and SoC internal scratchpad memories
- SRAM is faster and more expensive than DDR



life.augmented



life.augmented

Applications

Appliances

Metering

Robotics & Automation

Healthcare

Power Tools

Secure Locks

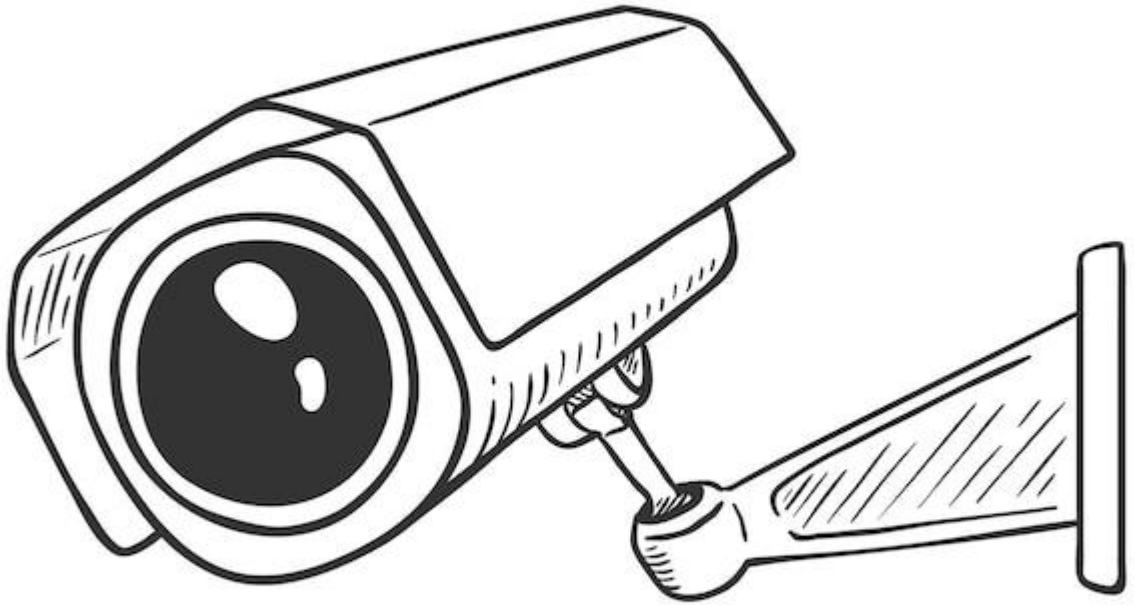
Surveillance

Smart Homes & Buildings

Drives, Pumps, Compressors

Lighting

Sample Application: Security Camera with Inbuilt battery



Marketing Requirement:

To design a SoC for a security camera

- Feature list:
  - Support RGB + IR camera
  - Illuminating Leds
  - Record video in external memory
  - Live Stream
  - Motor control
  - Face detections
  - Etc.

Negotiate

For this case study our **objective** is to reach a consensus between **Money** and **Performance**

- Developing part of the dataflow
- Identifying buffer/memory sizes
- Identifying CPU capabilities

Negotiate using the language of :
Key Performance Indicators and Cost Indicator

life.augmented

High Level Key Performance Indicator

- What rate are we able to process the frames?

At architectural level High level KPIs are typically derived from customer requirement and ‘realized’ or ‘verified’ using low level KPIs

Low Level KeyPerfIndicator

- **MIPS** : Million Instructions Per Second that core can/or did run

Requirement of MIPS get driven by algorithm complexity that runs on the CPU

- **Bandwidth** : How many transactions pass a point per second

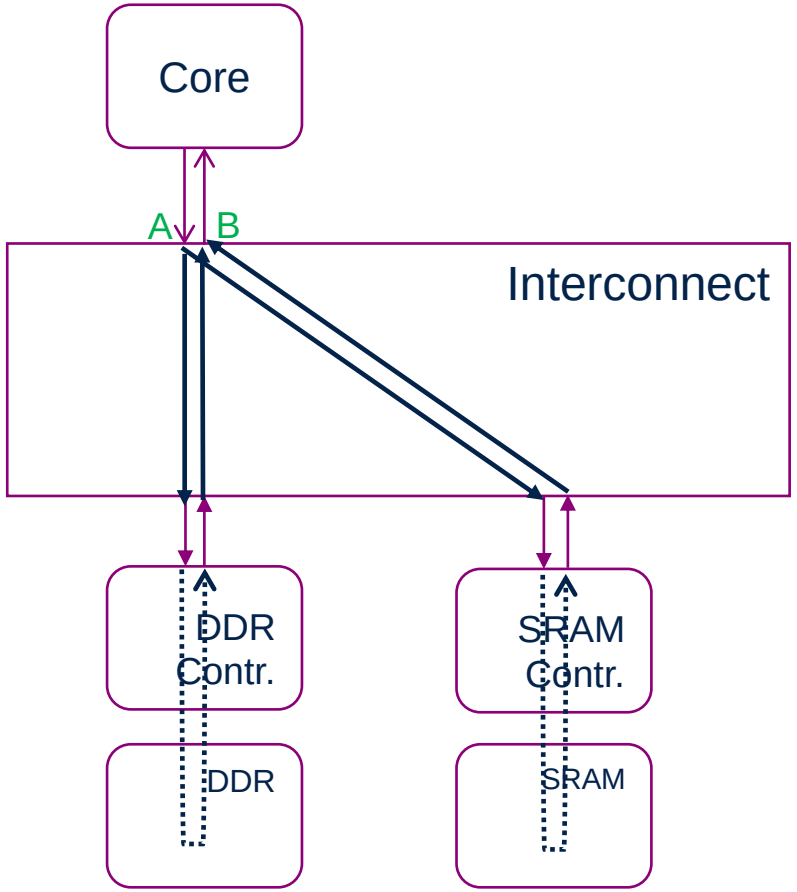
Requirement driven by amount of data movement that happens as per the dataflow

- **Throughput**: Generic term. How many jobs pass a point per second

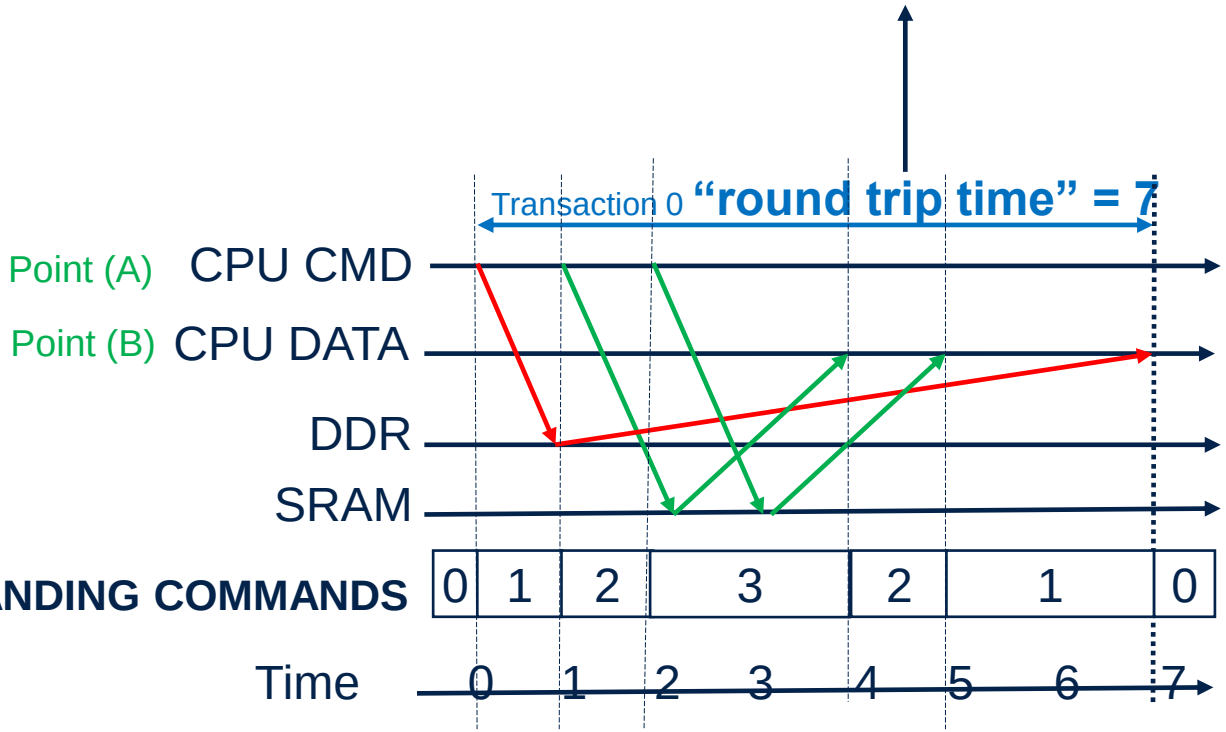
CPU performance may also be measured as Jobs/Second and maybe called throughput
Camera data bandwidth may also be called throughput

Low Level KeyPerfIndicator

- **Round Trip Time or Latency**



How much time it takes for a particular transactions data to reach back to the master after initiation of that transaction's command

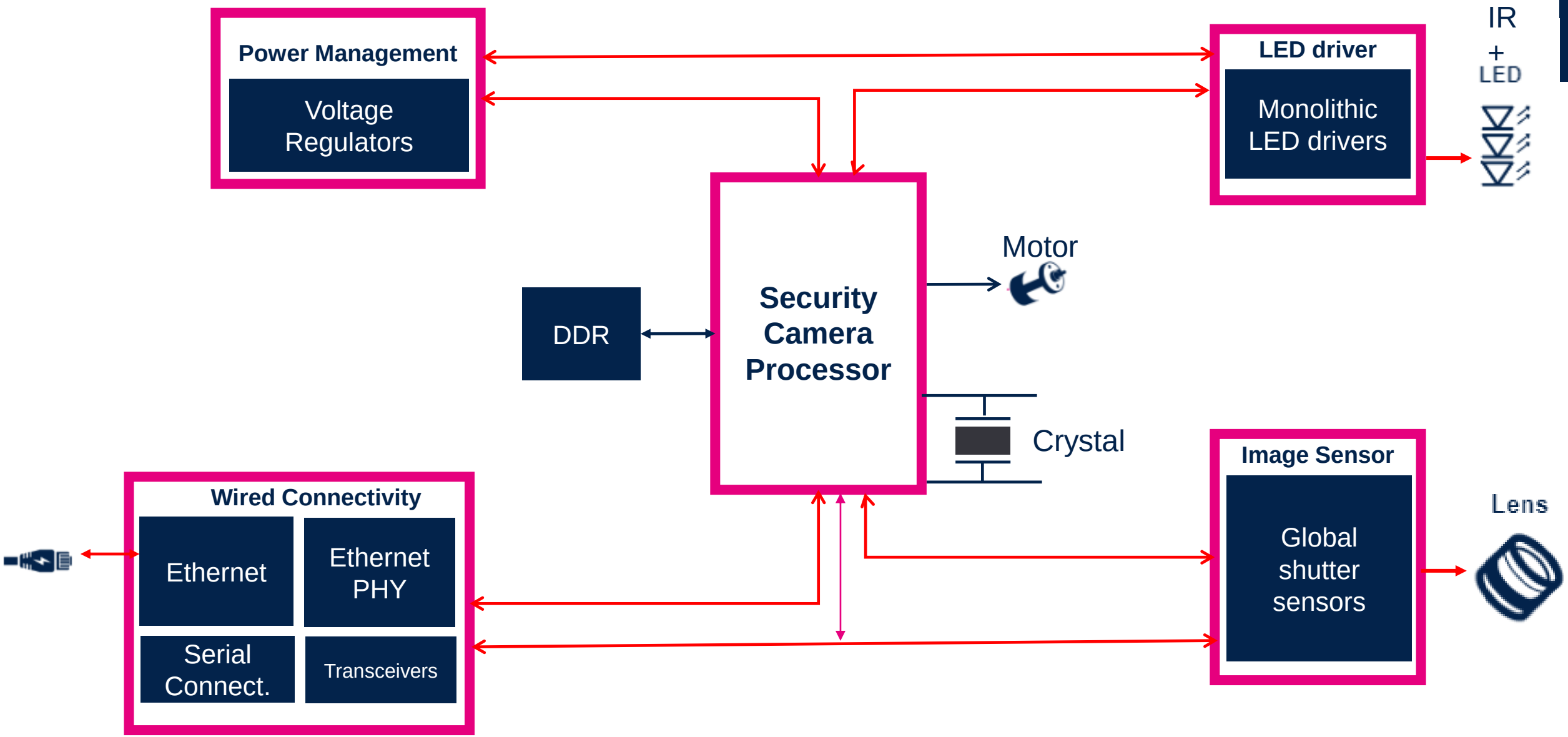


life.augmented

Cost Indicator

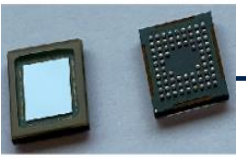
- **Lower Cost:**
  - Less Area: lesser the silicon area, lesser the cost
  - Lower Power:
    - Lower cost to manage the thermal
    - Cheaper packages
    - Smaller or no heat sinks
- **Competitive Advantage:**
  - Less Power: Less restrictions on where you can sell your chip
  - High Performance

Sample Application : Security Camera



life.augmented

Camera Dataflow



5.1MP
2560x1984 pix
(12-bit) @30fps
Sensor

How much SRAM will be needed?

Apart from SRAM needed for OS, drivers and data structures for other peripherals we will need 2560x1984x16bits to store 1 frame = 10 MB!.

How many frames do we need to store in SRAM?

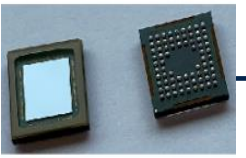
This is influenced by :
Algorithm: How many previous frames does the algorithm need?
CPU execution variability: OS, drivers/other services may introduce variability in execution, thereby forcing need for additional circular buffering. Typically, this may mean much more than 1 frame

Let's update the data flow to use DDR!

life.augmented

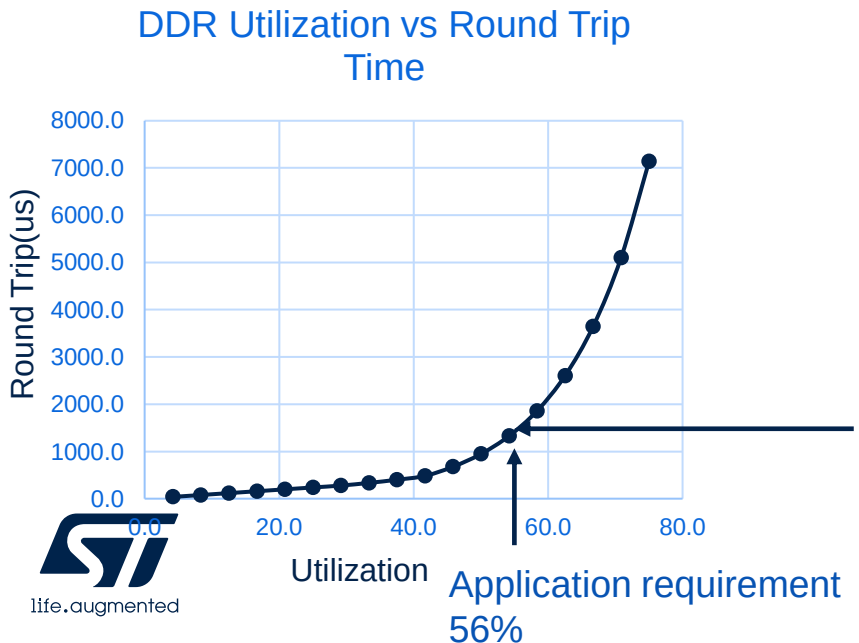
Image Sensor spec: https://www.st.com/en/imaging-and-photonics-solutions/vb1940.html#overview

Camera Dataflow



5.1MP
2560x1984 pix
(12-bit) @30fps
RGB/NIR
Sensor

DDRs although can provide decent Bandwidth, they come with occasional peak latency problem.
Camera is a real time IP hence it may be needed to re-evaluate buffering inside camera

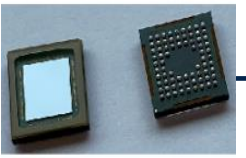


Peak anticipated latency = 1500 us

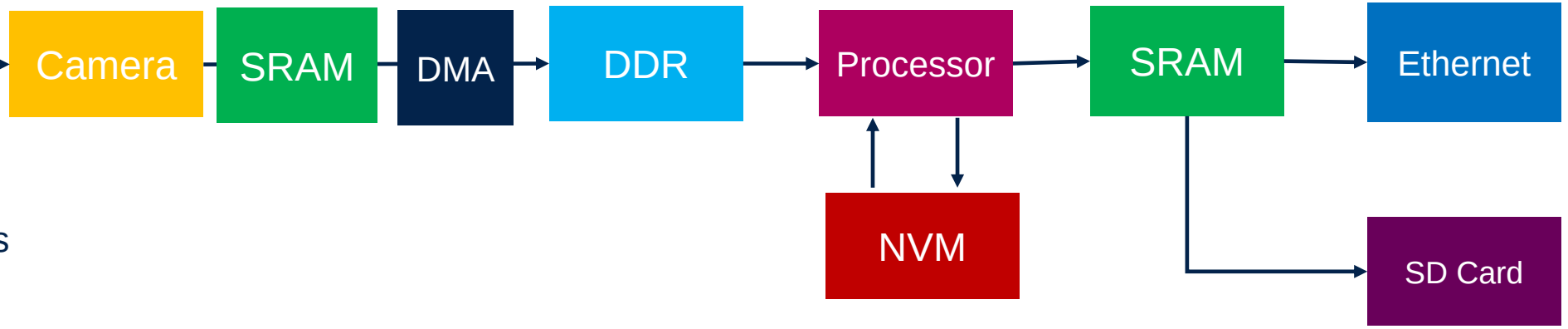
=> Buffer needed = 1500us x 10MB/(1/30sec) x 1.1(blanking)
= ~ **500 KB !**

This maybe too big to be dedicated just for camera.
So, we route the frame to DDR via SRAM using DMA, all the while storing only part of the frame in SRAM

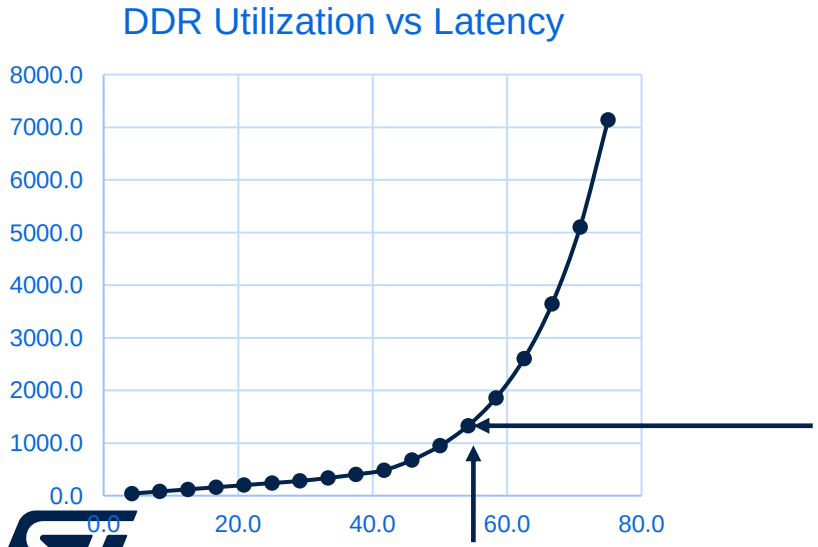
Camera Dataflow



5.1MP
2560x1984 pix
(12-bit) @30fps
RGB/NIR
Sensor



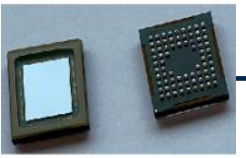
DDRs although can provide decent Bandwidth, they come with occasional peak latency problem.
Camera is a real time IP. peak latency Hence it may be needed to re-evaluate buffering inside camera



Application requires 56% DDR Bandwidth utilization
Peak latency = 1500 us
⇒ Buffer needed = 1500us x 10MB/(1/30sec) x 1.1(blanking) = ~ **500 KB**
This maybe too big to be dedicated just for camera. So, we put it in SRAM and use DMA to move the data to DDR

life.augmented

Camera Dataflow



Camera → SRAM → DMA → DDR → Processor → SRAM → Ethernet

SRAM → SD Card

5.1MP
2560x1984 pix
(12-bit) @30fps
RGB/NIR
Sensor

What kind of processor do we need to buy/build?

There can be many strategies here, some experts tell me we should split the Processor block into two parts. Image processing handled by an Application processor+extensions or DSP and analysis handled by AI engine

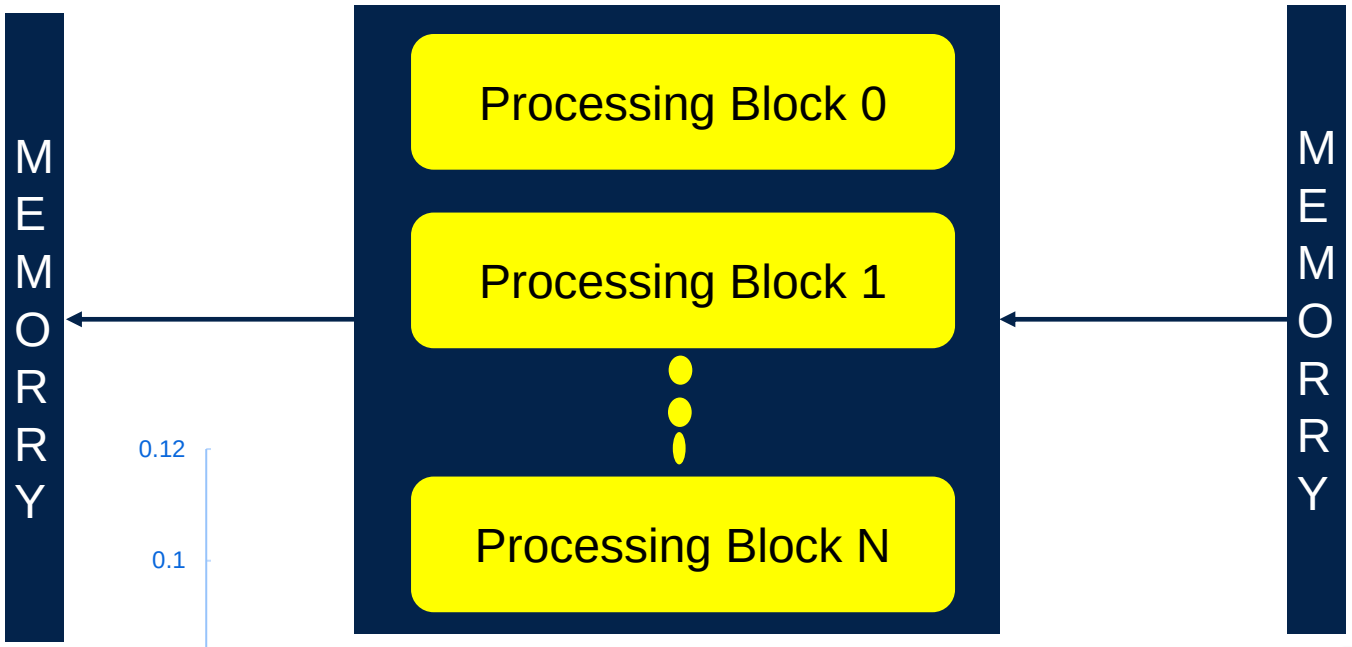
What capability(or parallel engines) do we need from the AI engine?

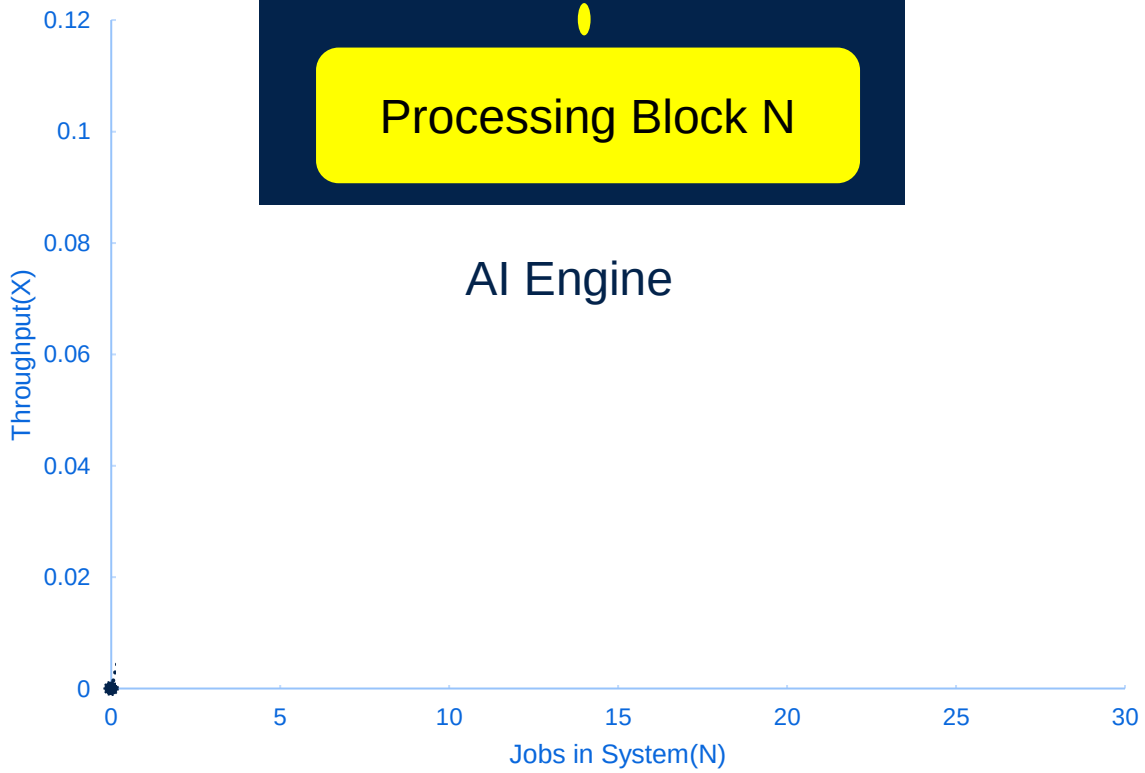
Give me a minute...err….days(maybe months?)… to figure it out..

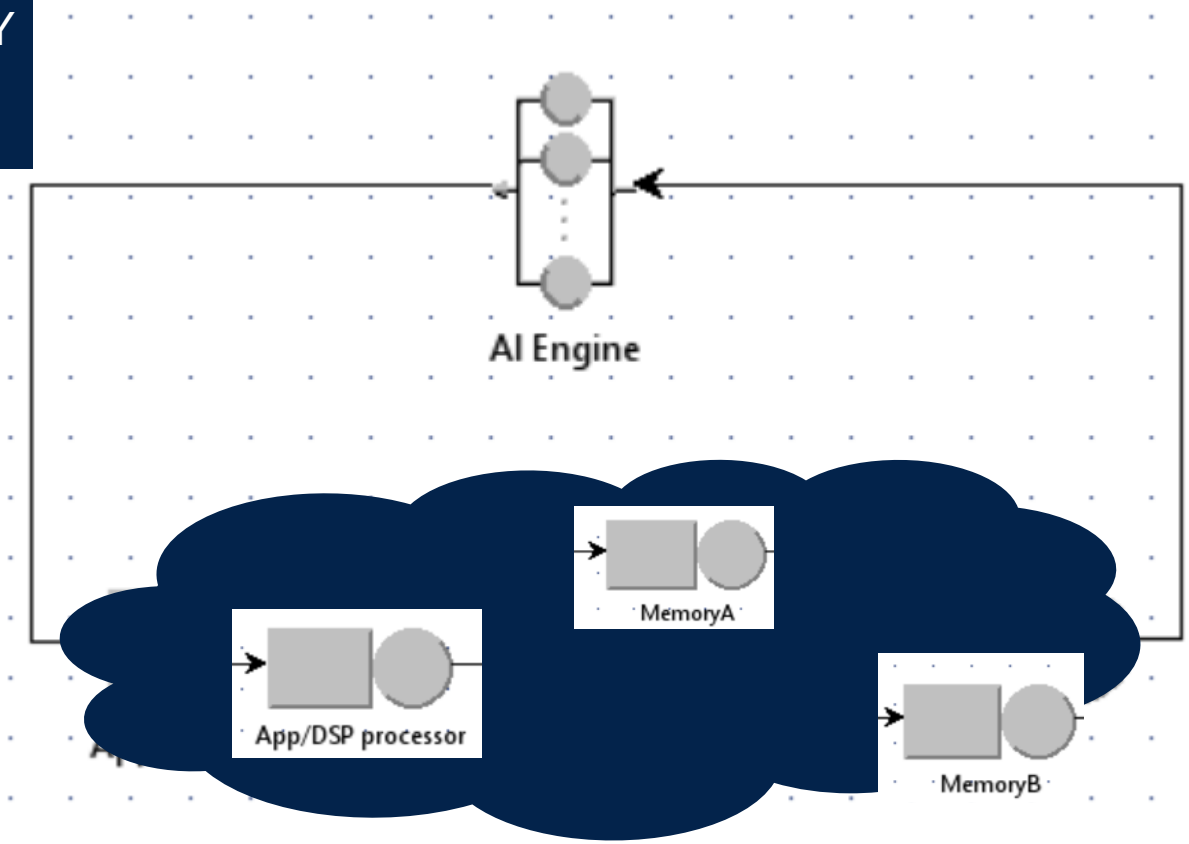
life.augmented

Analysis Phase

E[AI Engine Think time] = 18 sec
E[DSP Service Time] = 9 sec
E[MemoryA Service Time] = 5 sec
E[MemoryB Service Time] = 4 sec



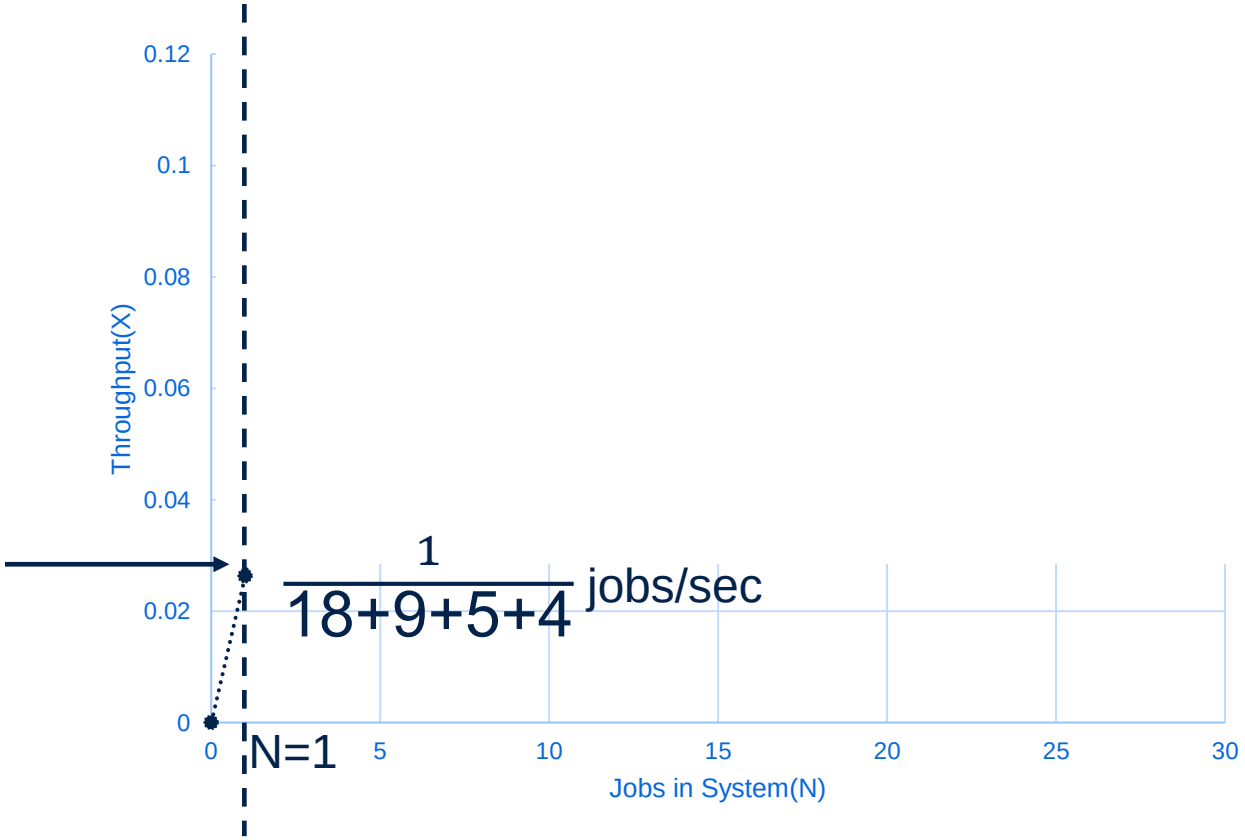


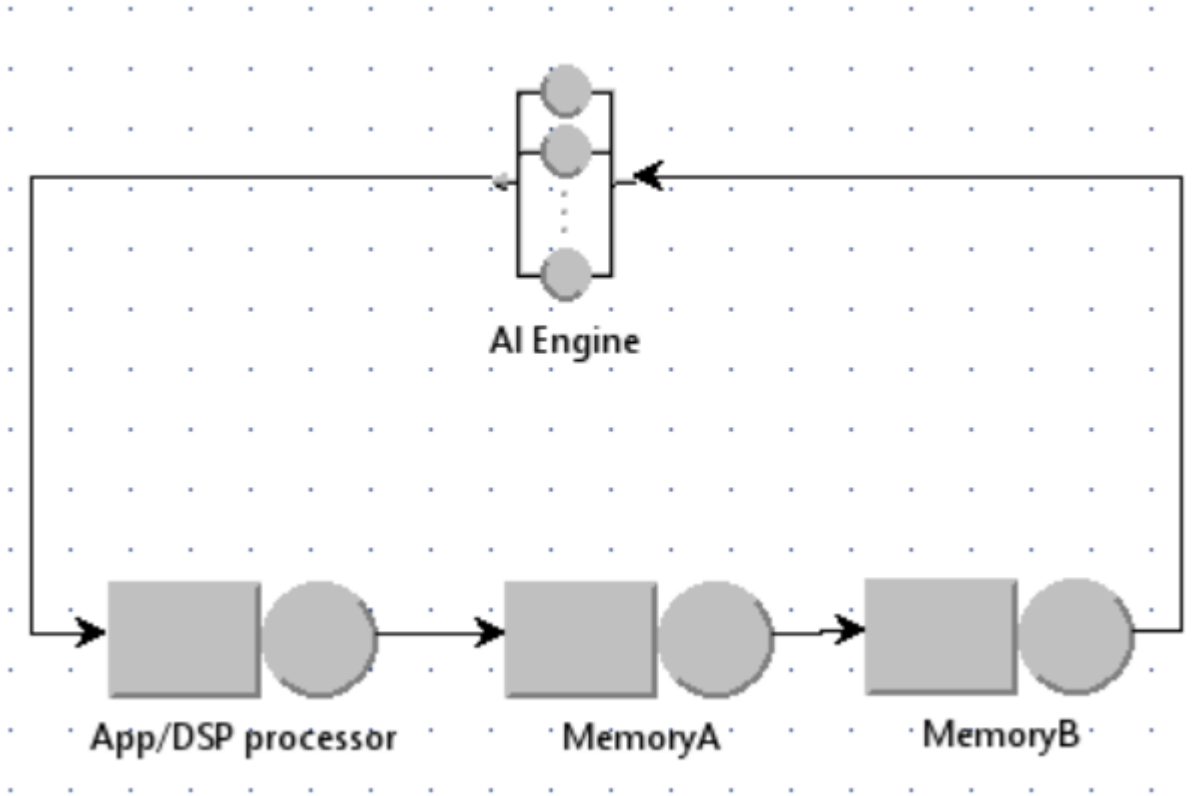


life.augmented

Analysis Phase

E[AI Engine Think time] = 18 sec
E[DSP Service Time] = 9 sec
E[MemoryA Service Time] = 5 sec
E[MemoryB Service Time] = 4 sec

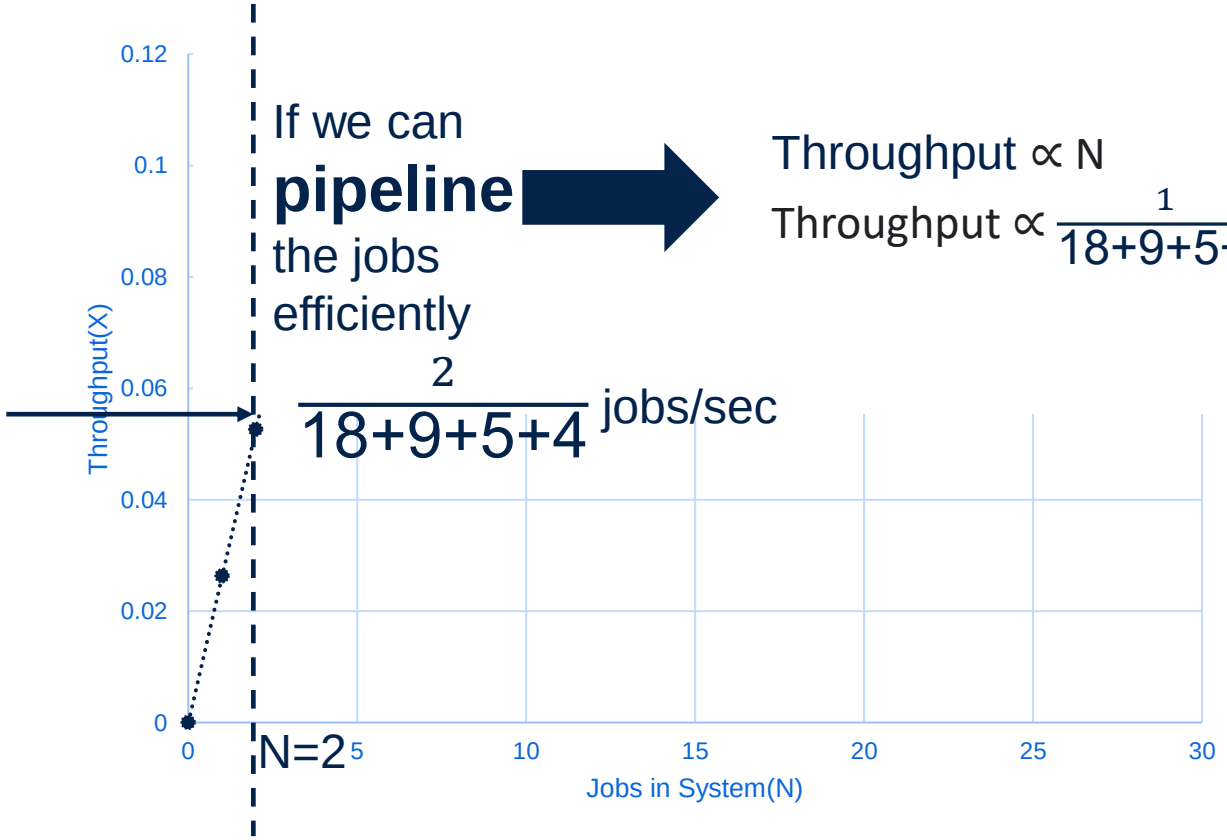


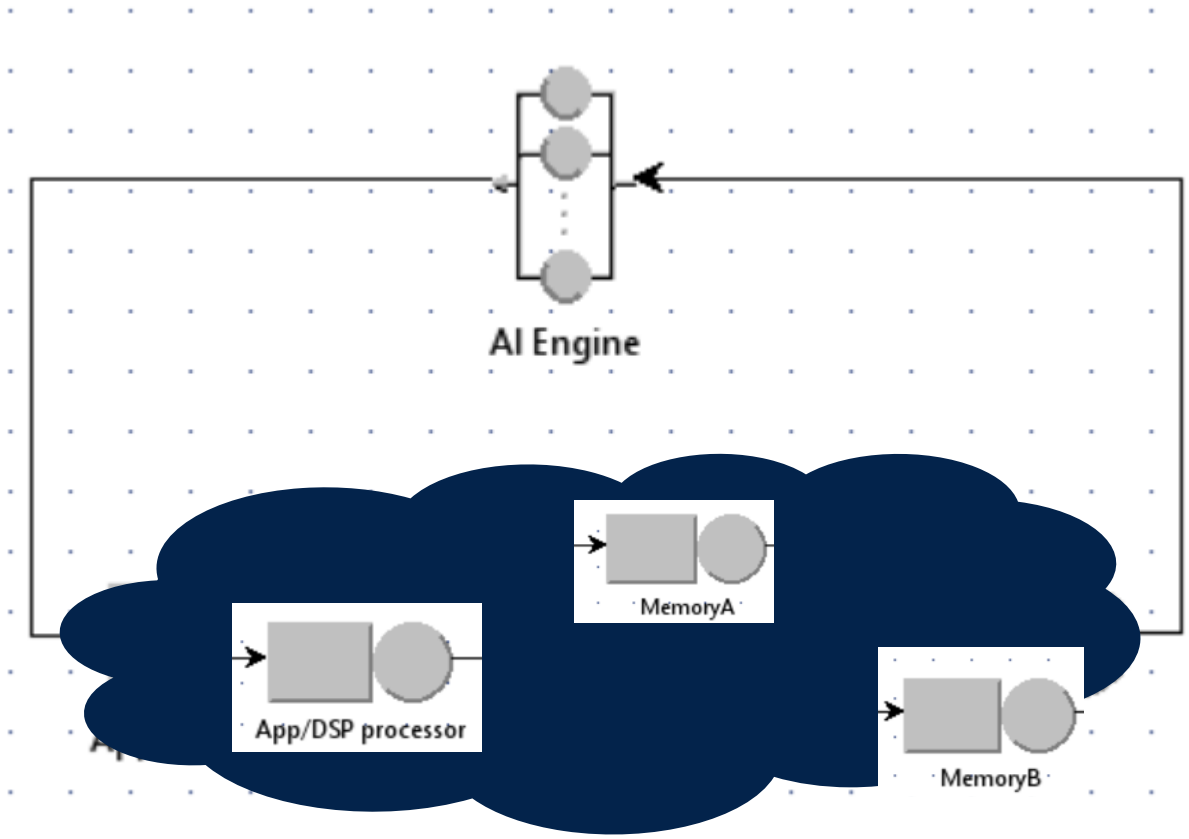


life.augmented

Analysis Phase

E[AI Engine Think time] = 18 sec
E[DSP Service Time] = 9 sec
E[MemoryA Service Time] = 5 sec
E[MemoryB Service Time] = 4 sec





life.augmented

Analysis Phase

X(throughput) is proportional to **N**(jobs in the system)

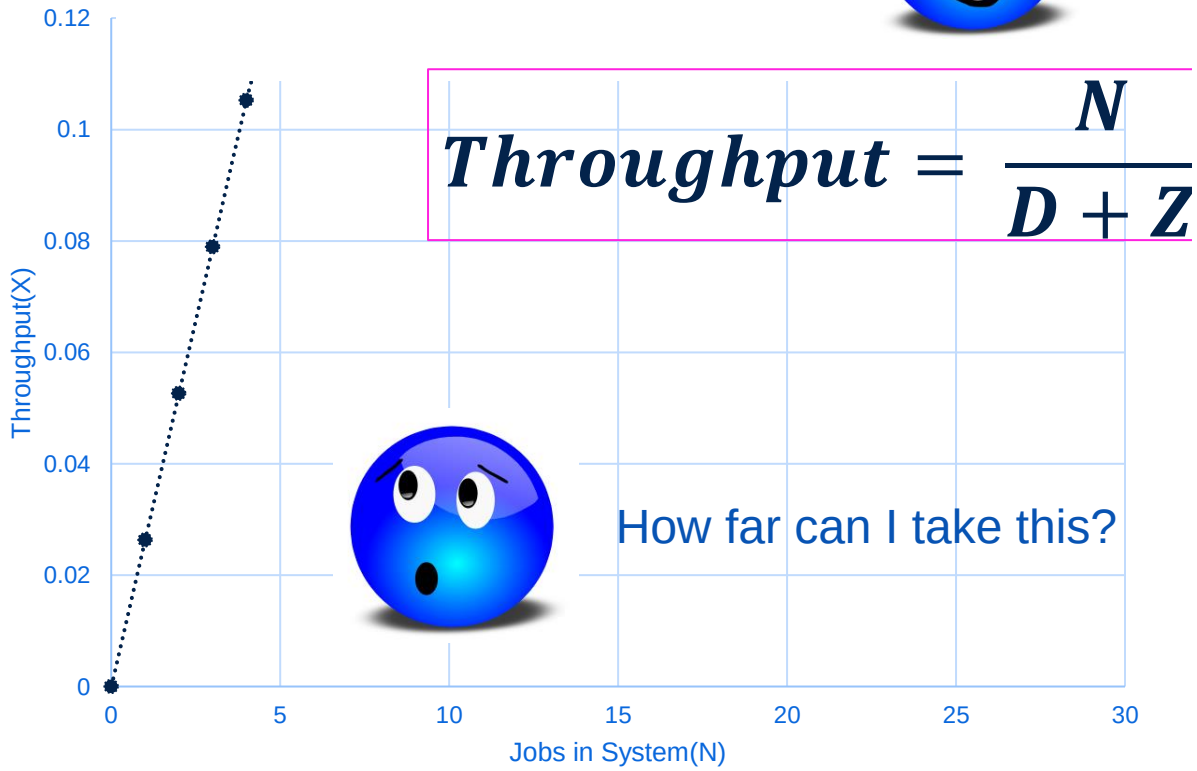
Time spent in system by a job =

Think time of AI Engine **(Z)**

+

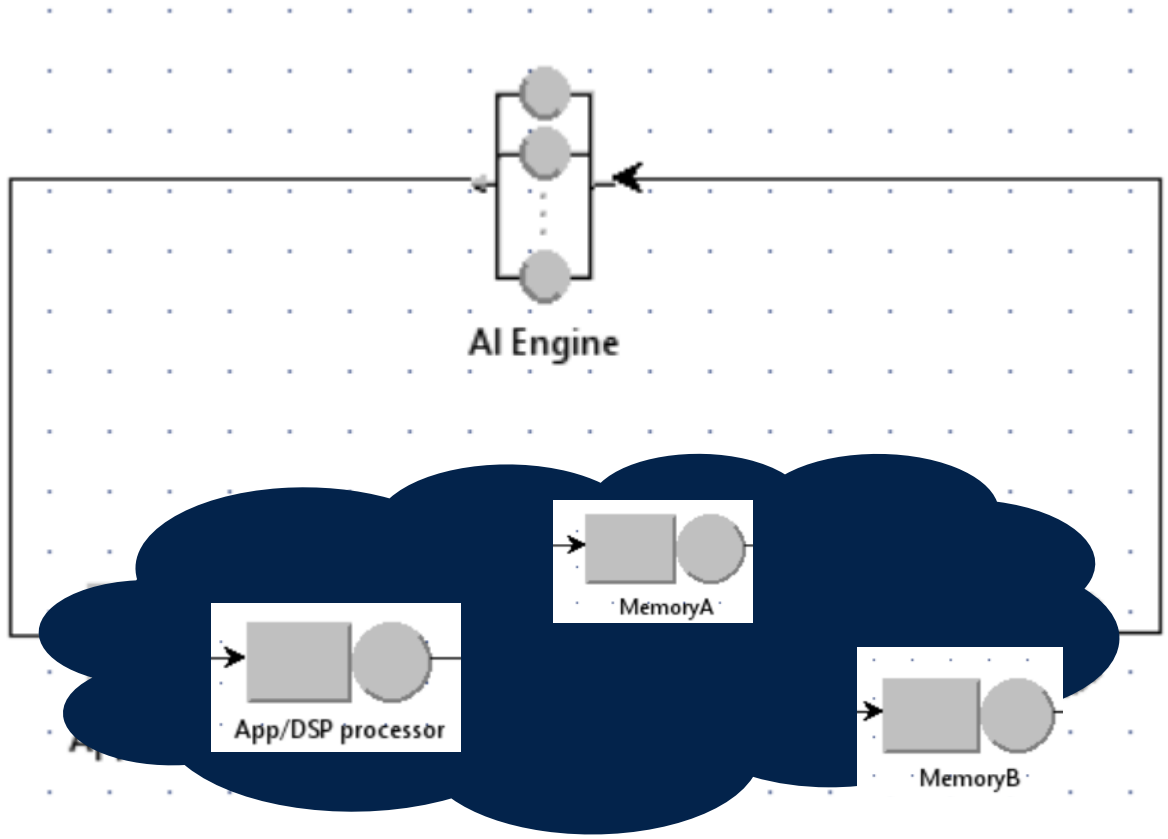
Total Service time of job on every node (D)

E[AI Engine Think time] = 18 sec
E[DSP Service Time] = 9 sec
E[MemoryA Service Time] = 5 sec
E[MemoryB Service Time] = 4 sec



$$Throughput = \frac{N}{D + Z}$$

How far can I take this?



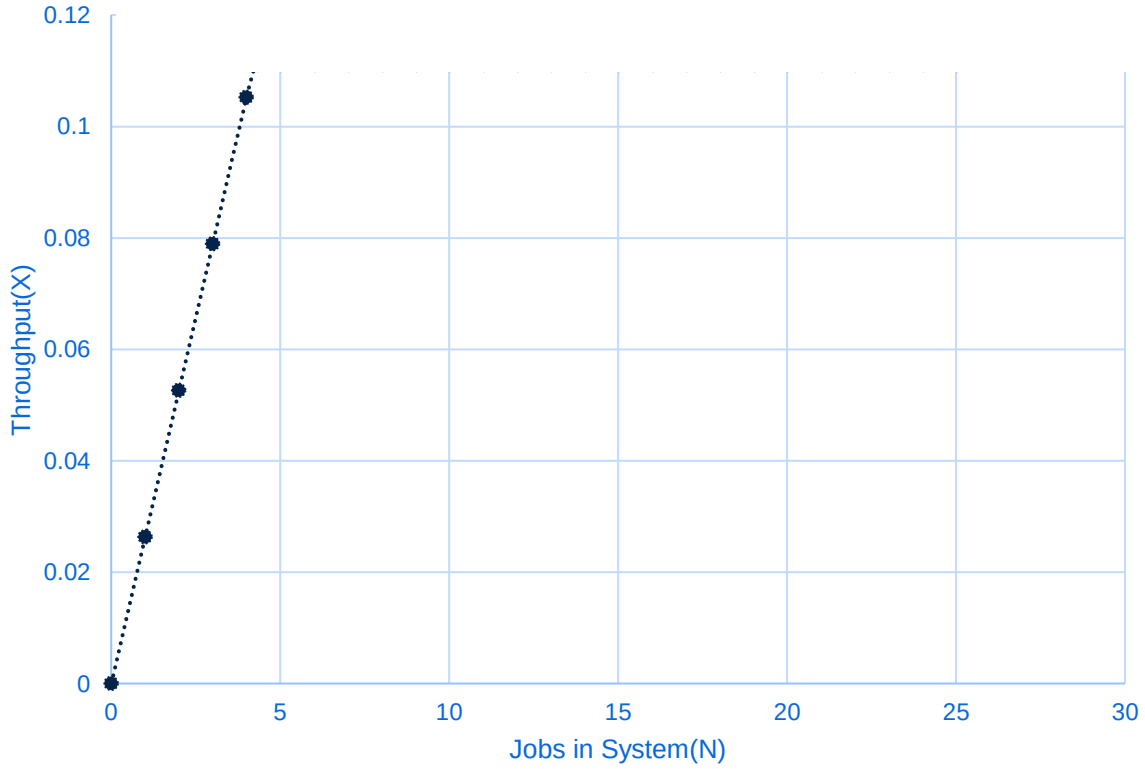
life.augmented

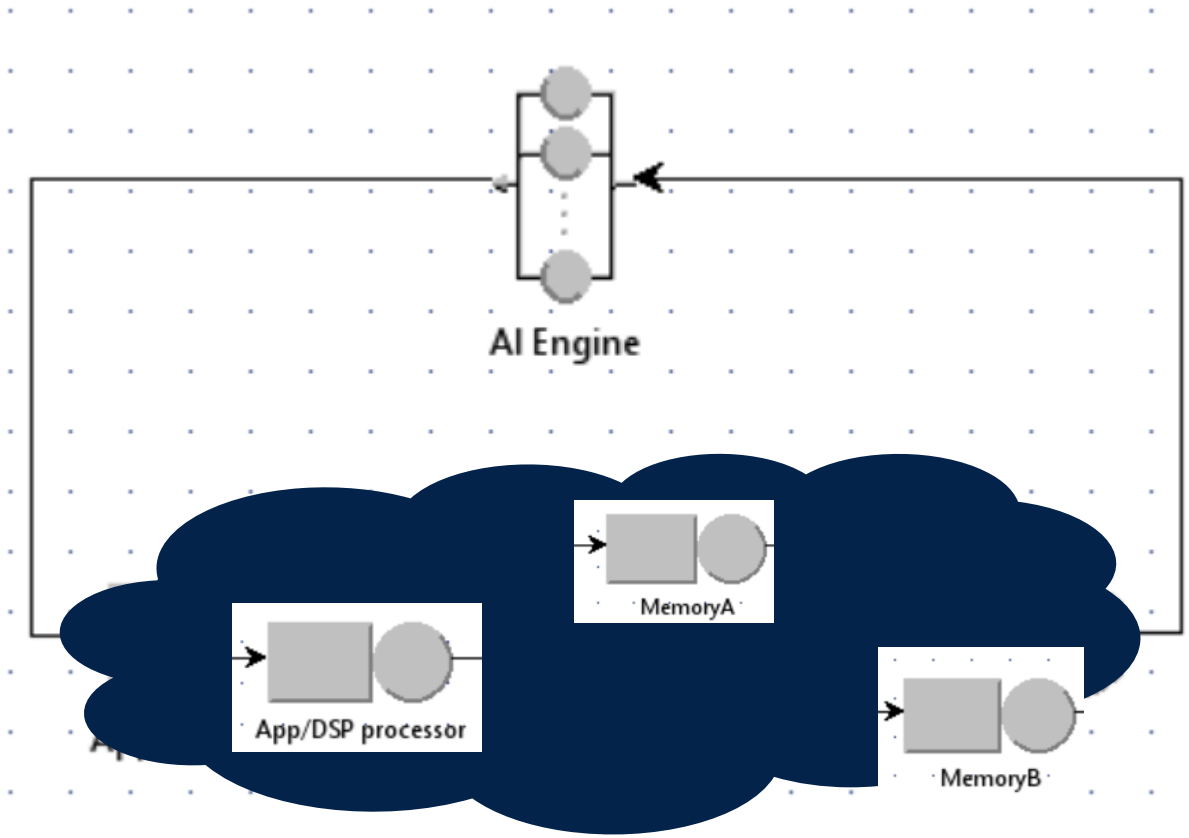
Analysis Phase

"Slowest step will determine the maximum speed"

1/Dmax = 1/9 = 0.111 jobs per second

E[AI Engine Think time] = 18 sec
E[DSP Service Time] = 9 sec
E[MemoryA Service Time] = 5 sec
E[MemoryB Service Time] = 4 sec





life.augmented

Extension

Number of Jobs/Transactions in the system

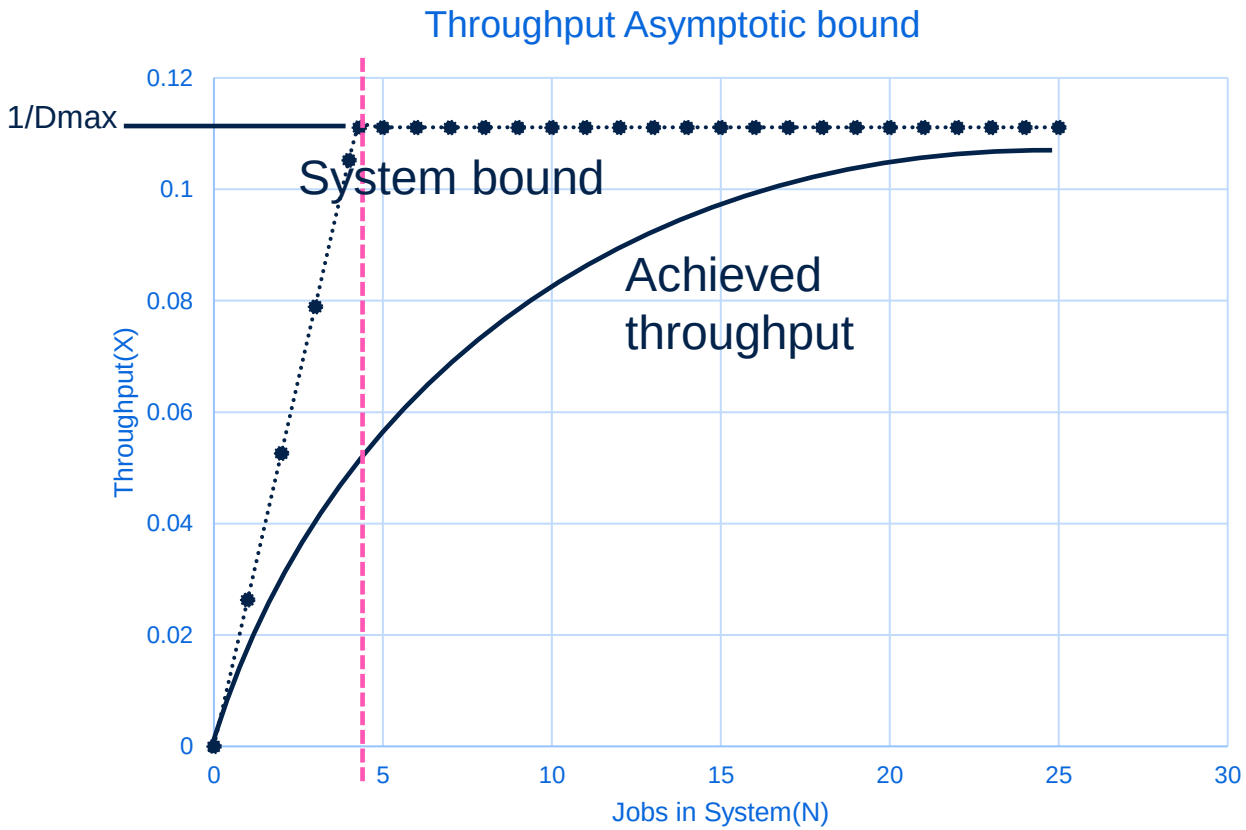
$$X \leq \min\left(\frac{N}{D+E[Z]}, \frac{1}{D_{max}}\right)$$

Total Service Demand at all nodes (time)

Think time

Maximum Service

E[AI Engine Think time] = 18 sec
E[DSP Service Time] = 9 sec
E[MemoryA Service Time] = 5 sec
E[MemoryB Service Time] = 4 sec



life.augmented

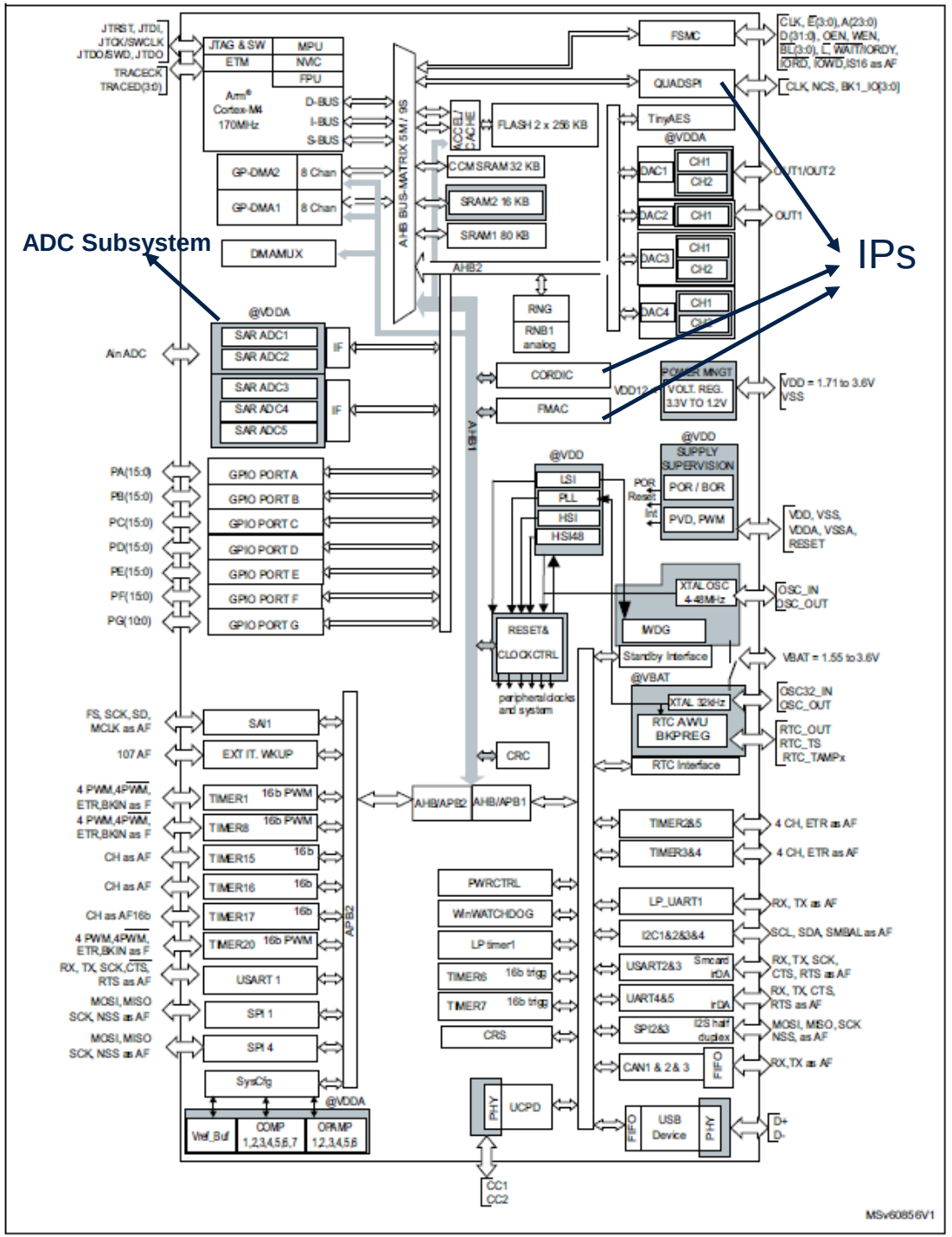
Key Question

What should be the capability of the AI Engine?

Answer:
Think time to stay at 18sec while throughput is to maintained at 0.111 jobs/sec
So, we need an AI Engine which can do atleast 2 jobs every 18sec **or**
We need 2 Engine which can do atleast 1 job every 18 sec

Which of these is better? ←

System-on-Chip (SoC) Design



SoC Design includes:

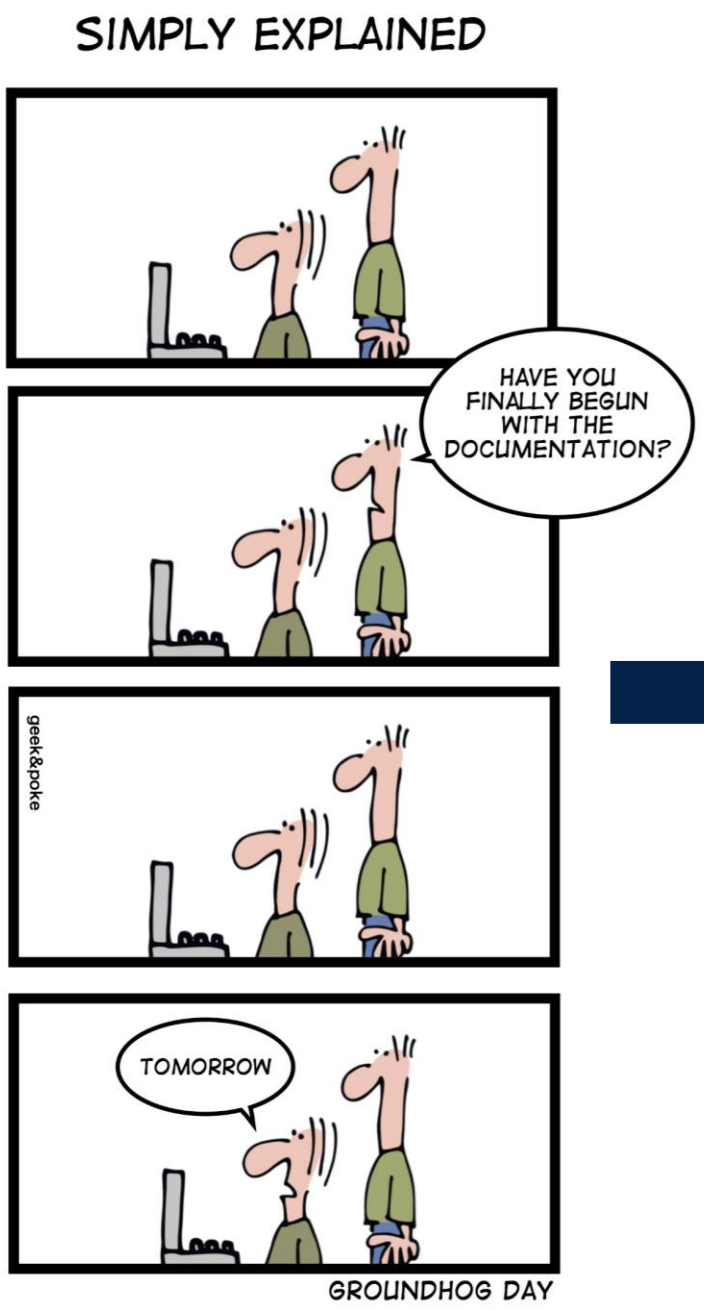
- Integrating multiple IP blocks and subsystems

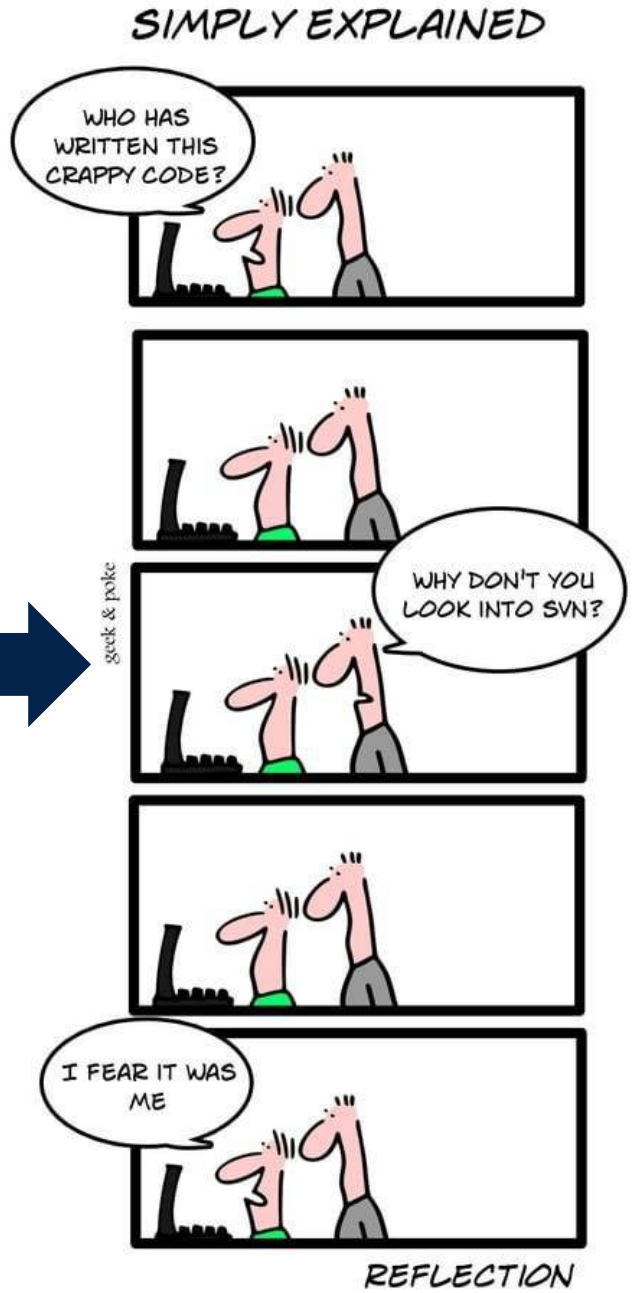
Frontend integration is a collection of various activities:

- Documentation:
  - IP selection , parameterization and integration
  - Bus architecture and interconnect design
  - Clock and power distribution

- Actual RTL – instantiation and Connection

- Quality Checks: CDC, RDC, and Lint.





Documentation

Documentation First!

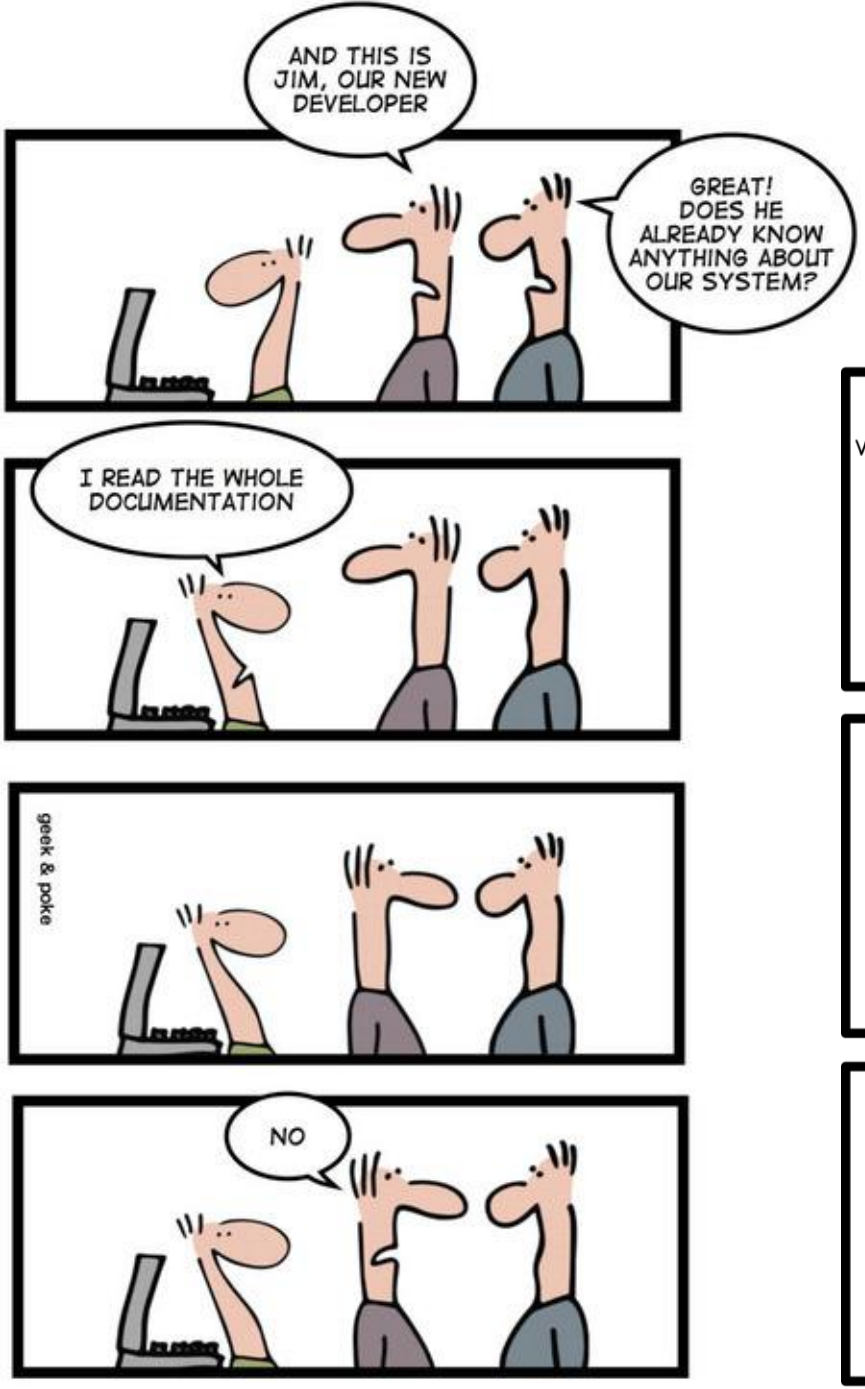
- Arguably it is the most important step of Design Implementation
- **A tool to THINK** and WORK!
- Lets others to review your work

Following documents are created:

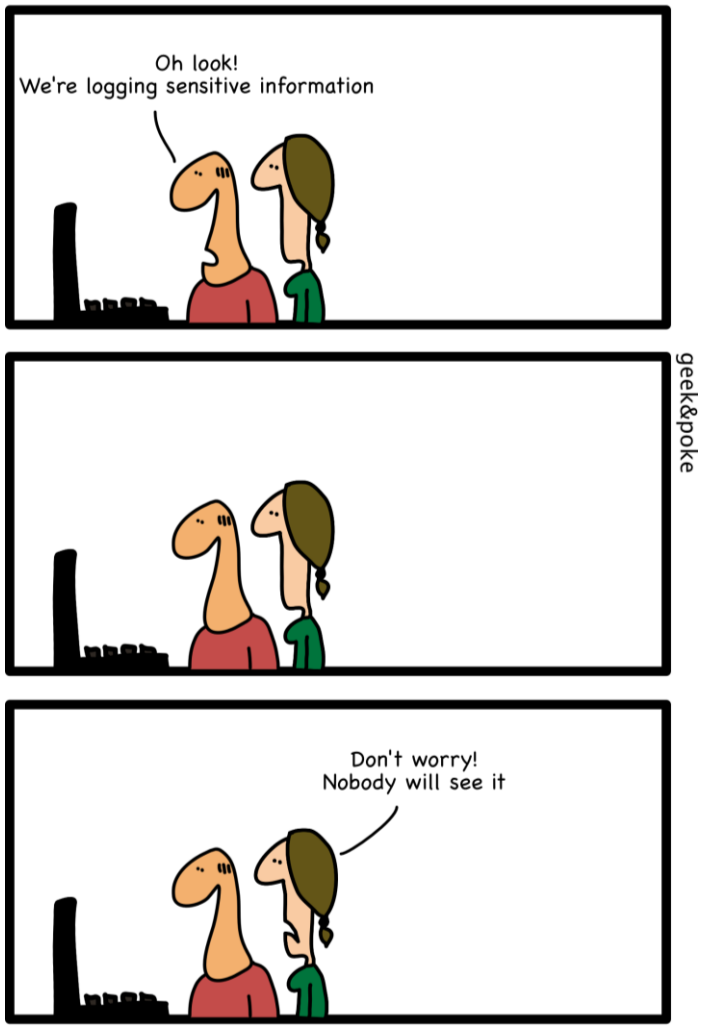
1. **Design specifications**: Co-owned with Arch.
   Describe the high-level requirements and specifications of the SoC design, such as the target application, performance requirements, power requirements, and other design constraints.

2. **Design documents**:
   Provide a detailed description of the SoC design, including the IP blocks, bus architecture, clock and power distribution, and other design details. May include design constraints, such as timing constraints, power constraints, and area constraints.

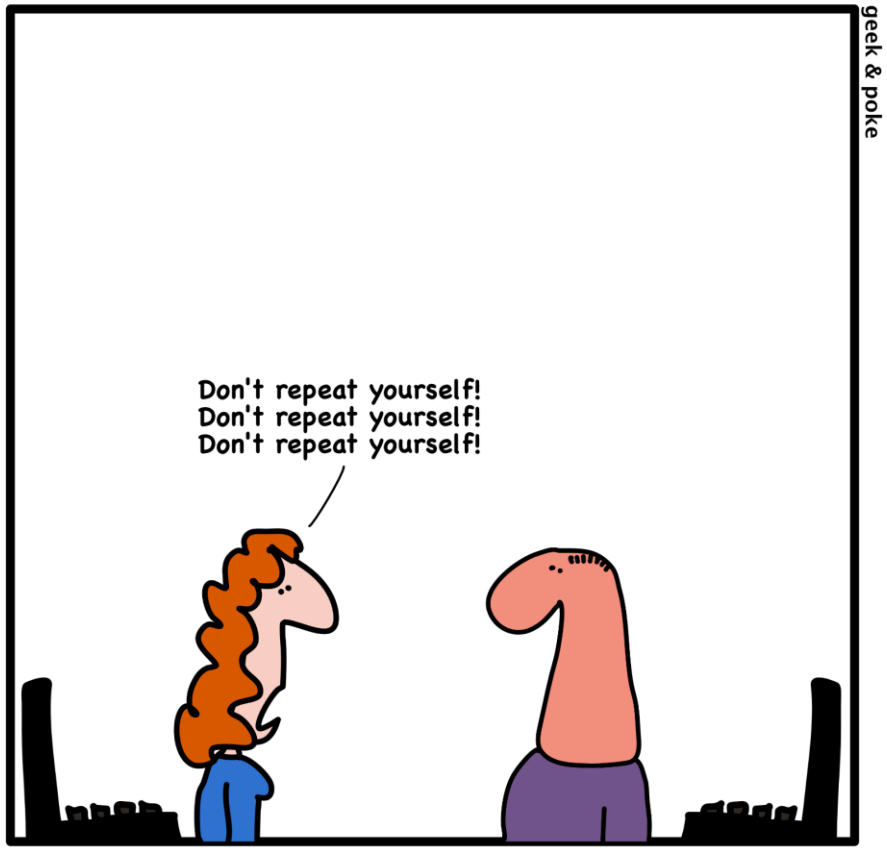
Documentation: Do’s and Dont’s



Be relevant and verbose



Classify



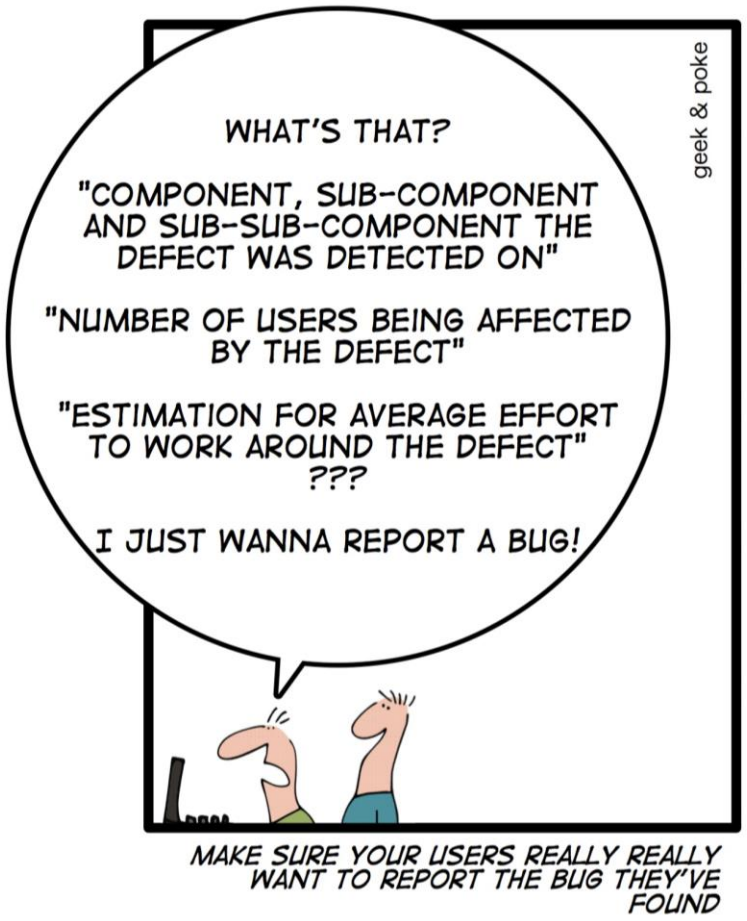
Single Source

Frontend: IP Selection, Parametrization and Integration

- Selecting the right IP for the job
  - CPU selection:
    - Application – Real Time/Non Realtime/ Data Intensive or Control Intensive/ Floating point or Fixed Point/Low power or high power/low or high performance etc.
  - Memories:
    - What should be the width of SRAM? ECC requirements?
    - Retention
  - Suppression of features – Parametrization

- Integration: Generate the RTL!
  - Clock flexibility/Control
  - Reset flexibility – Reset retentions
  - Safety requirements
  - Power domains
  - Compatibility

Quality Checks/Signoffs



Why Quality Checks in Design Cycle:

- Cost and Effort of handling a bug grows exponentially with time!

Major Frontend Design Signoffs

- LINT
- CDC
- RDC
- …

life.augmented

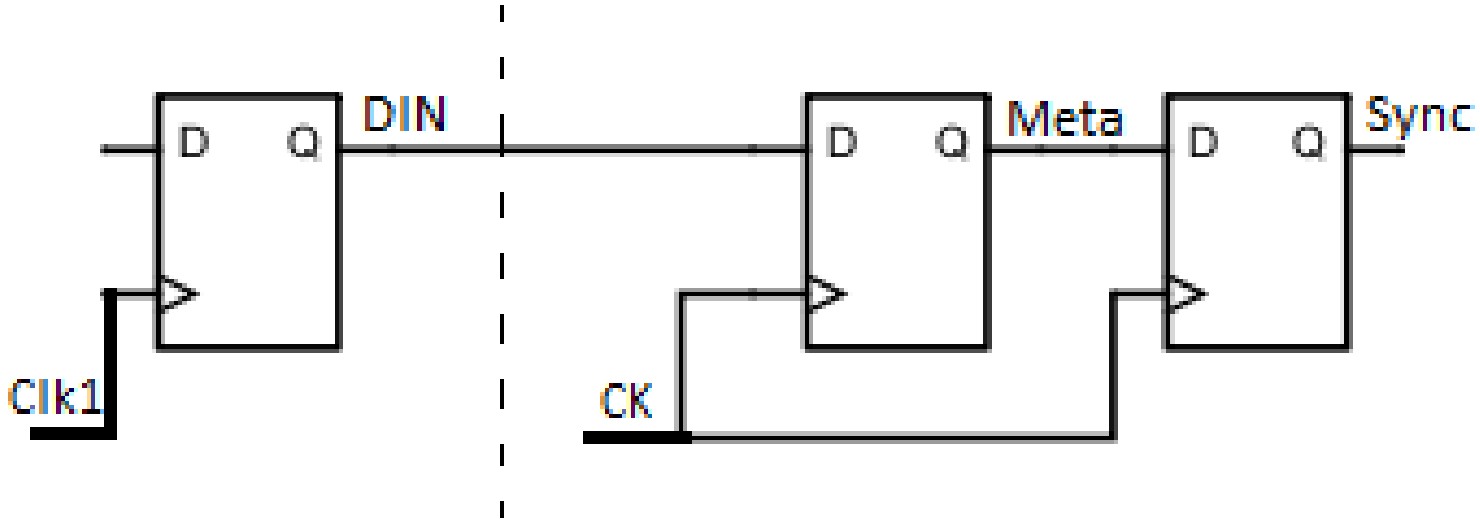
Quality Checks(Lint)

Lint signoff ensures that the SoC design meets the coding guidelines and does not have any coding errors or issues. Lint signoff involves analyzing the design code and ensuring that it meets the coding guidelines and standards.

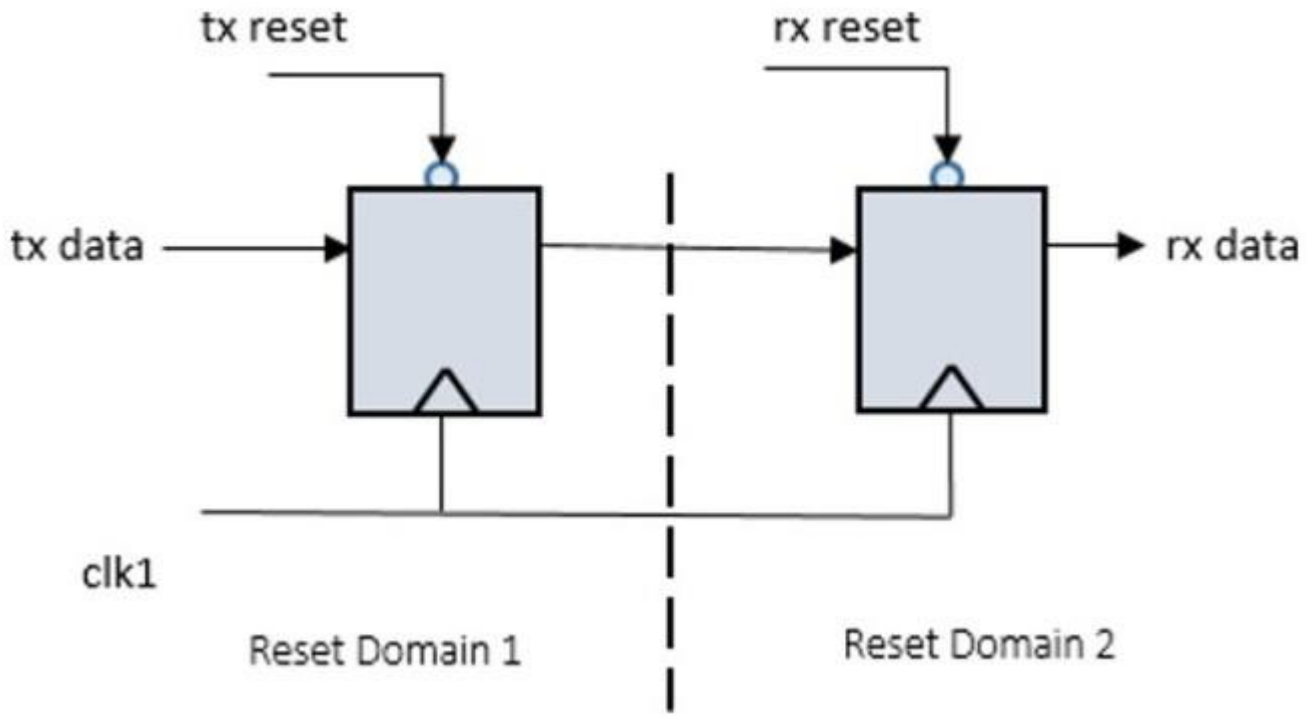
“Intent matches implementation”

Quality Checks (CDC)

CDC signoff: CDC (Clock Domain Crossing) signoff ensures that the data transfer between different clock domains is reliable and does not result in any data loss or corruption. CDC signoff involves analyzing the clock domain crossing paths and ensuring that they meet the timing constraints.

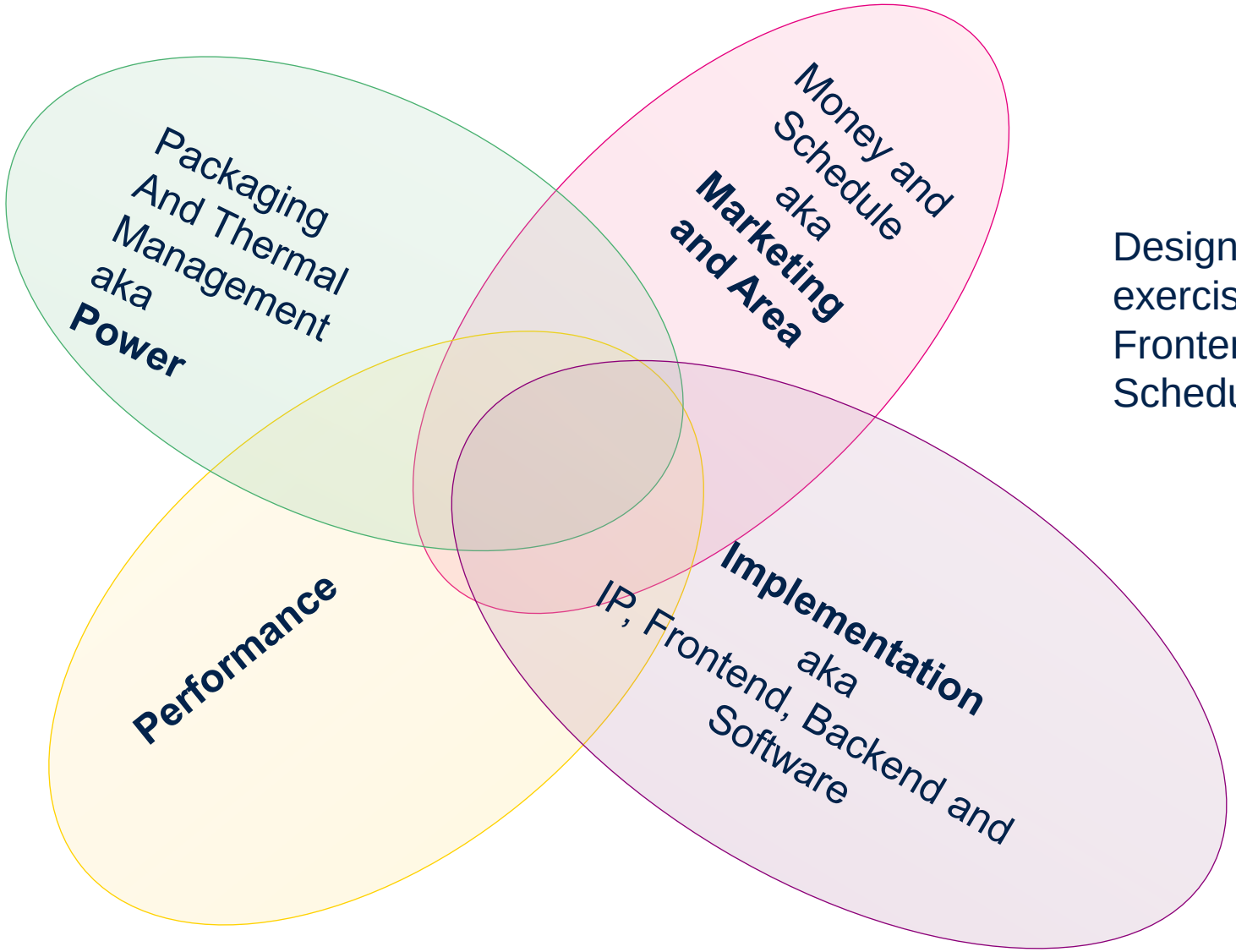


Quality Checks(RDC)

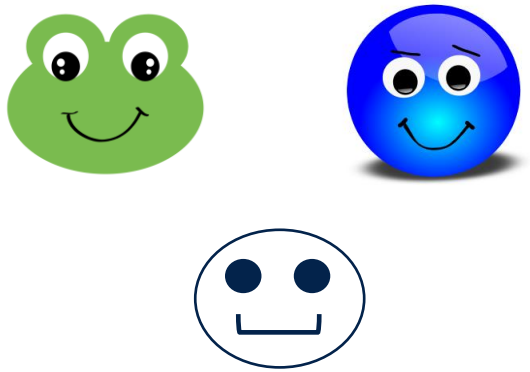


RDC signoff: RDC (Reset Domain Crossing) signoff ensures that the reset signals are correctly synchronized between different reset domains. RDC signoff involves analyzing the reset domain crossing paths and ensuring that they meet the timing constraints.

Conclusion



Design and Architecture is essentially a tradeoff exercise between Power, Performance, Area, IP, Frontend and Backend Implementation, Schedule etc. etc.



Our technology starts with You

Find out more at www.st.com

life.augmented