Reconfigurable & Computationally Efficient Architectures for Multi-Armed Bandit Algorithms

S. V. Sai Santosh & Sumit J. Darak
{siripurapu17197, sumit}@iiitd.ac.in

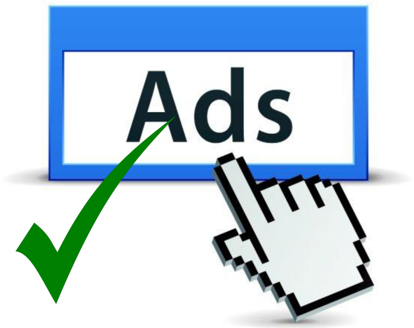
Department of Electronics & Communication, IIIT-Delhi, India-110020

2020 IEEE International Symposium on Circuits & Systems
Virtual, October 10-21, 2020

A2A
Algorithms to Architecture

Exploration Vs. Exploitation: The Multi-Armed Bandit Problem

- Imagine you **own a website**.
- A company gives you a bunch of ads that they want you to **display to a user** whenever he/she connects to your page.
- You'll **get paid** if the customer clicks the ad that's displayed to him.
- You find a **great ad** that's getting clicked by a lot of users. You might want to **exploit** that ad by displaying the same to all users.
- Good chance that there's a **better ad** in the bunch.
- At the same time, the chances of you encountering a **bad ad** are also pretty high.



What is a Multi-Armed Bandit problem?

Each ad has its own probability distribution of **success**.

- **Success:** User clicks the ad
- **Failure:** User doesn'

Displaying any one of the
reward of either **1 (for s**
failure).

Task is to find a strategy that will give you the **highest rewards** in the long run **without the prior knowledge** of the probability distribution.

$$maximize \sum_{k=1}^{K} \sum_{n=1}^{N} r_k(n)$$

r of ads in the bunch.

connects to the
akes a round **n**. We
have a total of **N** rounds.

- Ad **k** gives reward $r_i(n)$ where $\mathbf{r_k(n)}$ = **1** if the user clicked it or **0** if the user didn't.

One idea can be to apply the idea discussed in the previous slide by using an optimal strategy that exploits better performing ads while also exploring other ads.

This is what Upper Confidence Bound Algorithms try to do!

The Upper Confidence Bound!

- At the start, all ads have the **same observed** average value & **confidence bound Q**.
- Then the algorithm creates **confidence bound** for each ad. It **randomly picks** any of the ads.
- **Two** things can happen– the user **clicks** the ad or **doesn't**.
- If the user **doesn't click** the ad, the observed average of the ad **will go down**.
- If **he/she clicks**, the observed average **goes up** and the confidence bound also **goes up**.
- Exploiting the best ad **decreases** its confidence bound.

$$Q(k,n) = \frac{X(k,n)}{T(k,n)} + \sqrt{\frac{\alpha \log(n)}{T(k,n)}}$$

- **X(k,n)** is the sum of rewards of ad **k** upto round **n**.
- **T(k,n)** is the number of times ad **k** was selected upto round **n**.
- ***α*** is the exploration factor.

The Process!

Step 1 At each round consider two numbers for each ad **k**:

- **T(k,n)** – number of times the ad **k** was selected upto round n.
- **X(k,n)** – sum of rewards of ad **k** upto round n.

Step 2 From these two numbers we compute:

- **Average reward** of ad **k** upto round n:

$$\overline{r_k(n))} = \frac{X(k,n)}{T(k,n)}$$

- **Confidence interval** at round n:

$$\triangle_k(n) = \sqrt{\frac{\alpha \log(n)}{T(k,n)}}$$

Step 3 Select the ad which has the maximum upper confidence bound **UCB**.

$$Q(k,n) = \overline{r_k(n)} + \triangle_k(n)$$

Some more Upper Confidence Bounds!

The Basic Upper Confidence Bound!

$$Q(i,n) = \frac{X(i,n)}{T(i,n)} + \sqrt{\frac{\alpha \log(n)}{T(i,n)}}$$

Upper Confidence Bound with Variance Estimation!

$$Q_v(i,n) = \frac{X(i,n)}{T(i,n)} + \sqrt{\frac{\alpha_1 \log(n) \cdot V(i,n)}{T(i,n)}} + \frac{\alpha_2 \log(n)}{T(i,n)}$$

$$where,\ V(i,n) = \frac{Y(i,n)}{T(i,n)} - \left(\frac{X(i,n)}{T(i,n)}\right)^2$$

Upper Confidence Bound with Constant Tuning!

$$Q_t(i,n) = \frac{Y(i,n)}{T(i,n)} - \left(\frac{X(i,n)}{T(i,n)}\right)^2 + \sqrt{\frac{\alpha \log(n)}{T(i,n)}}$$

$$where,\ Y(k,n) = Y(k,n-1) + (R_n)^2 \cdot \mathbf{1}_{\{I_n == k\}}$$

Our contribution!

Problem 1: Robotics, IoT and wireless applications with strict latency constraints demand tight integration of MAB with the Physical layer (PHY).

Solution: Solved by integrating the UCBs with hardware in a resource-efficient & power-efficient manner, without compromising performance.

Problem 2: Various works have shown that a single UCB may not guarantee optimal performance under various constraints. The Velcro approach of parallel implementation of all algorithms for all machines is extremely inefficient.

Solution: Solved by the proposed dynamically reconfigurable architecture such that the type of UCB algorithm, can be changed on-the-fly.

Problem 3: The number of machines i.e., **K** required in different settings may vary. And, we want the flexibility to vary **K** without compromising on resource utilisation & power consumption.

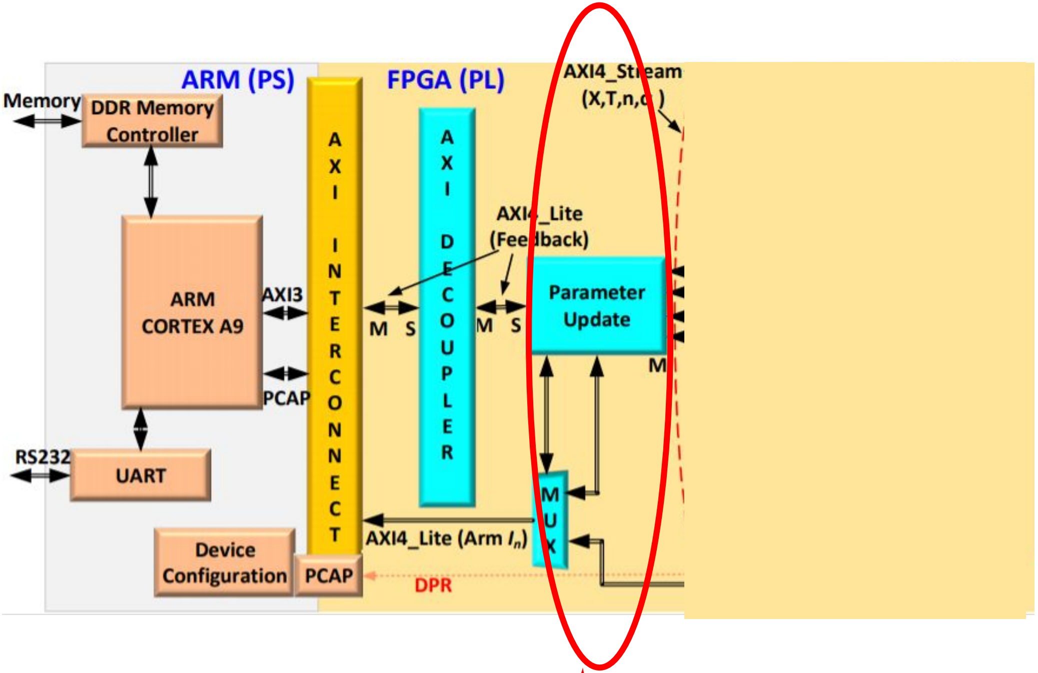
Solution: Solved by the proposed dynamically reconfigurable architecture such that the number of arms can be changed on-the-fly.

Proposed Architecture!

For the realization of a single round of the MAB problem, we need three blocks, as discussed in one of the previous slides:

- **Initialization & Parameter Update,** to initialize machines with same bounds, & update the parameters ***X, T, N*** for each machine.
- **Confidence Bound Calculation,** according to the chosen UCB algorithm.
- **Selection of the machine with the highest Confidence Bound,** for use in the next round.

Initialization & Parameter Update Block



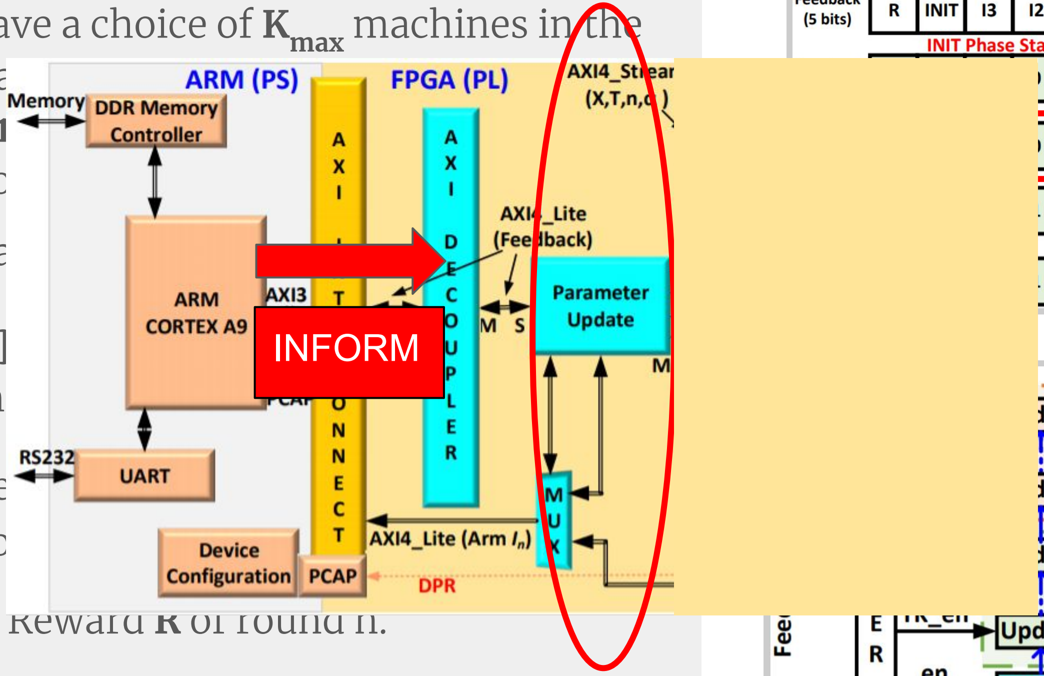
Initialization &
Parameter Update Block

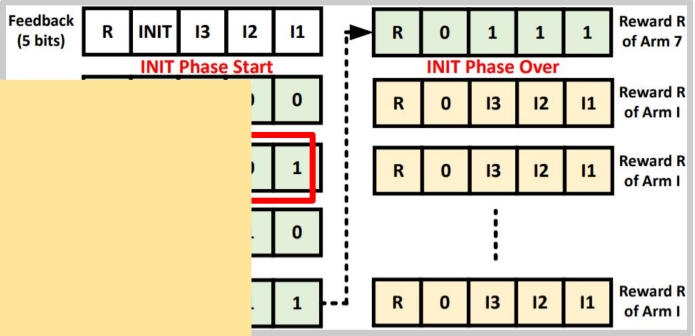
Inside Initialization & Parameter Update Block

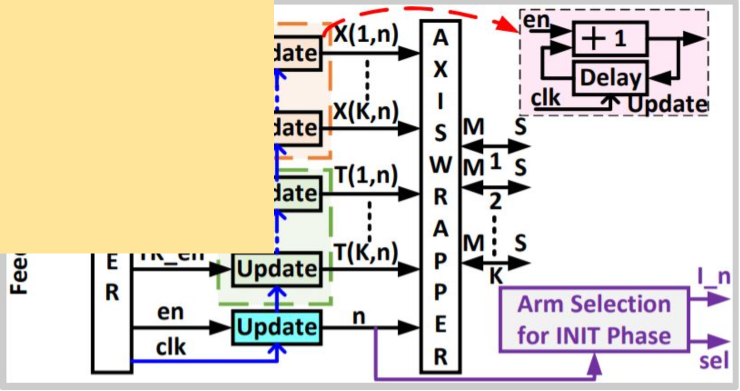
Assuming we have a choice of K_{max} machines in the
Multi-Armed Ba
ceil[log$_2$(K_{max}+1
architecture abo

If you have **7** ma

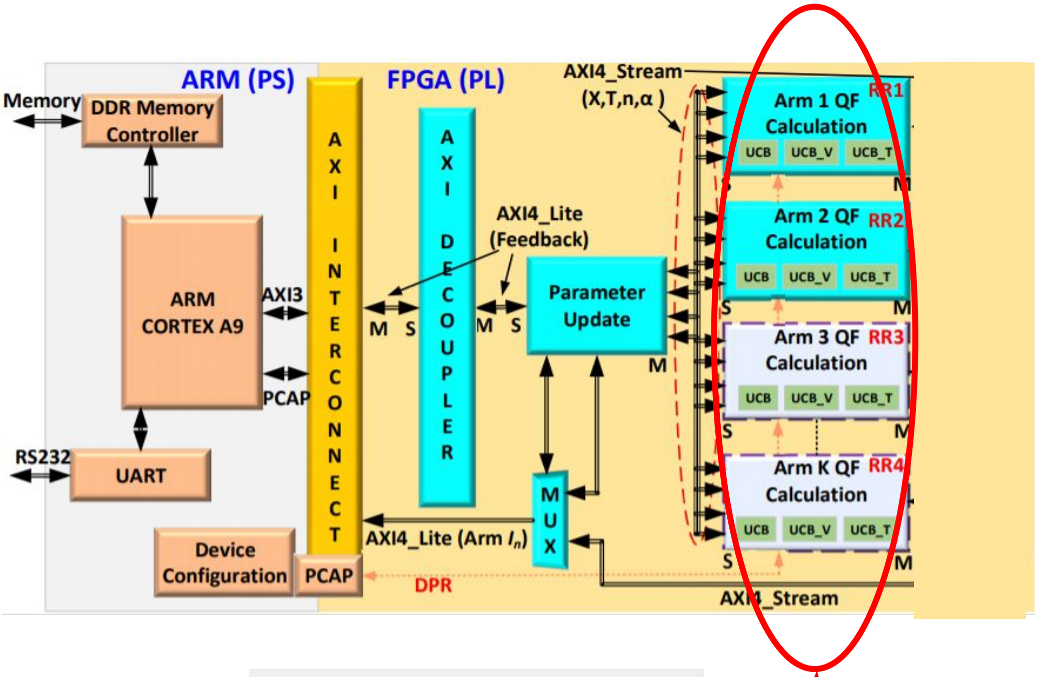
- **Bit [A-1:0]**
  selected in
- **Bit [A]**- Re
  problem to
- **Bit [A+1]**- Reward **R** of round n.







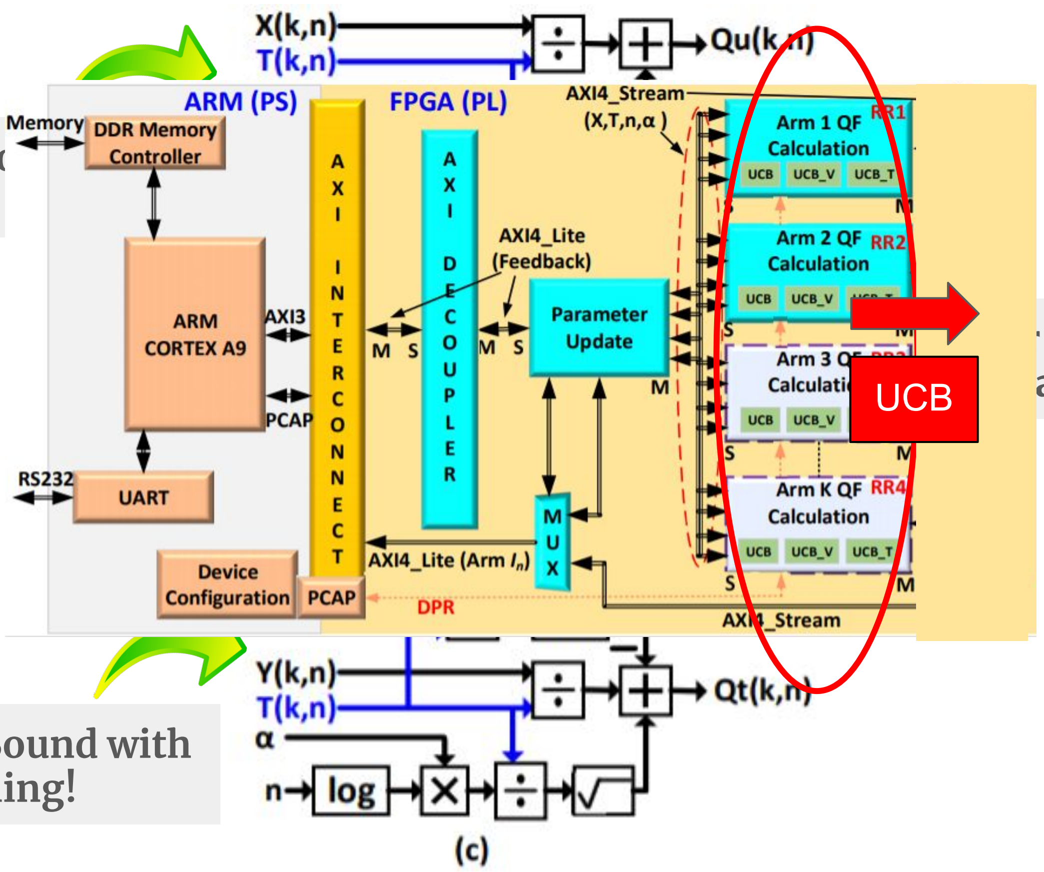
Confidence Bound Calculation Blocks



Confidence Bound Calculation Block for **at most 4** machines.

Inside each Confidence Bound Calculation Block

The Basic Upper Co
Bound!

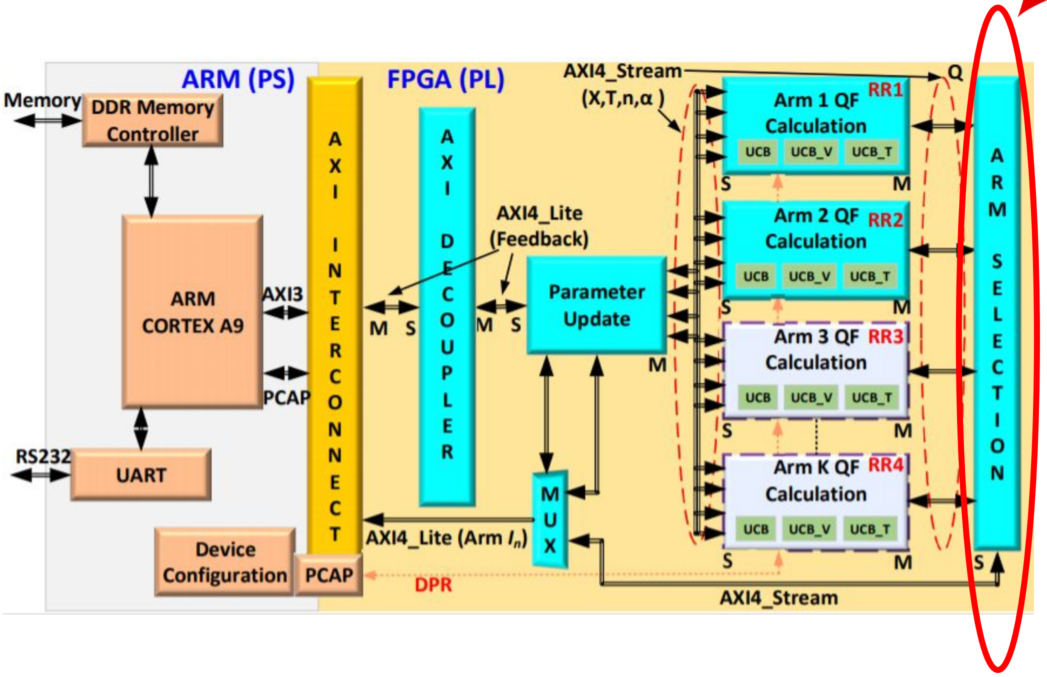


$$\frac{X(i,n)}{T(i,n)} + \sqrt{\frac{\alpha \log(n)}{T(i,n)}}$$

Confidence Bound
ariance Estimation!

Upper Confidence Bound with Constant Tuning!

Machine Selection Block

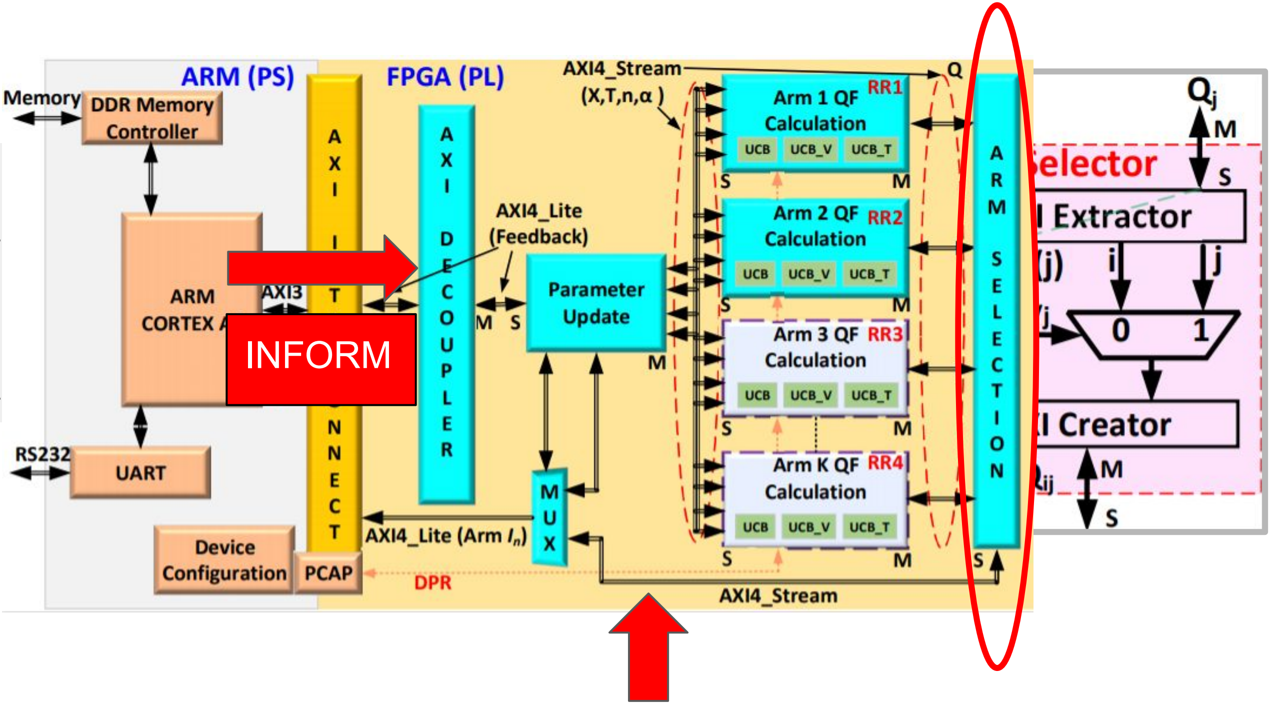


Machine Selection Block

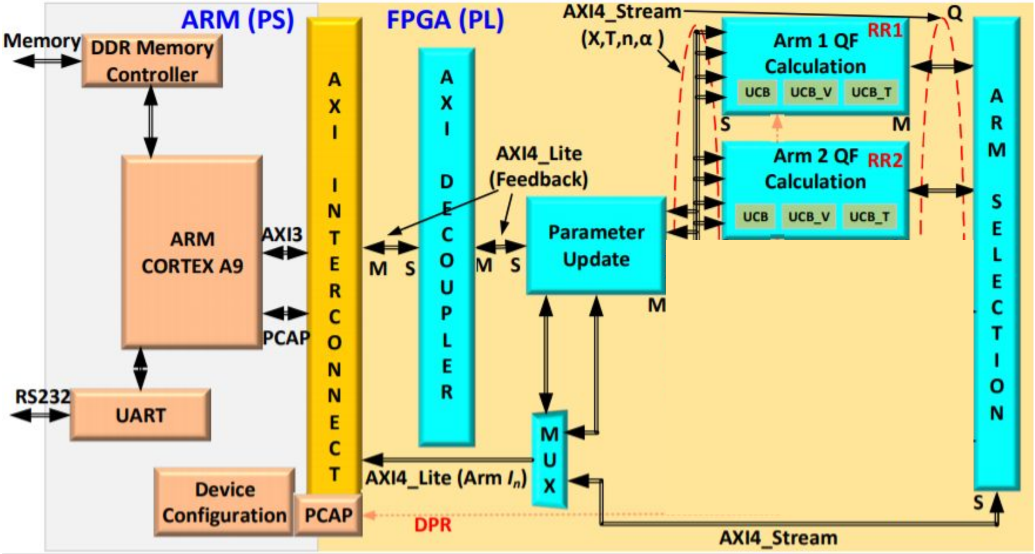
Inside the Machine Selection Block

The Arm Selection
machine having th
Confidence Bound

M denotes Master
Slave Port using in



On-the-fly reconfigurability!



The Confidence Bound Calculation Block is **partially reconfigurable** such that the **number of blocks** active at any instant as well as the **UCB algorithm they use** can be reconfigured **on-the-fly**.

Velcro Approach vs Proposed Approach!

Assuming we have a maximum of **4** machines in the problem setting, & a **single** algorithm option available:

Case I: All machines active

- No. of active CB Calculation blocks in **Velcro** = **4**
- No. of active CB calculation blocks in **Proposed** = **4**

Case II: Only **3** machines active

- No. of active CB Calculation blocks in **Velcro** = **4**
- No. of active CB calculation blocks in **Proposed** = **3**

Assuming we have a maximum of **4** machines in the problem setting, & **three** algorithm options available:

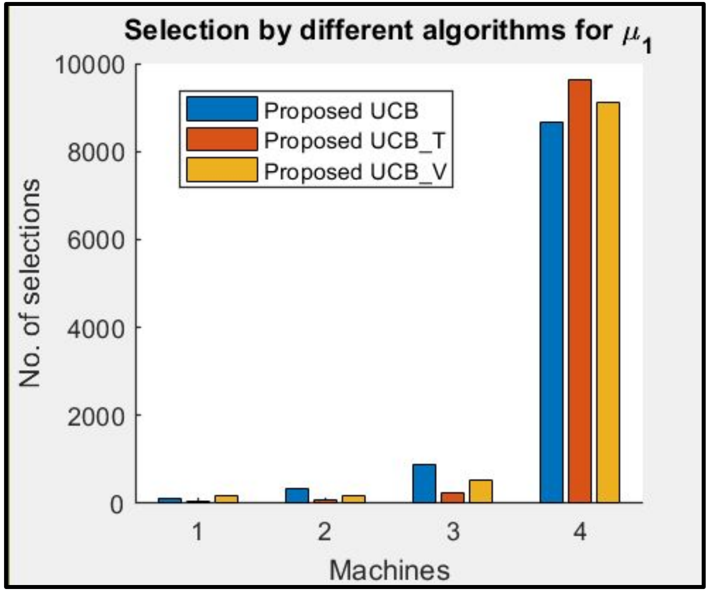
Case I: All machines active

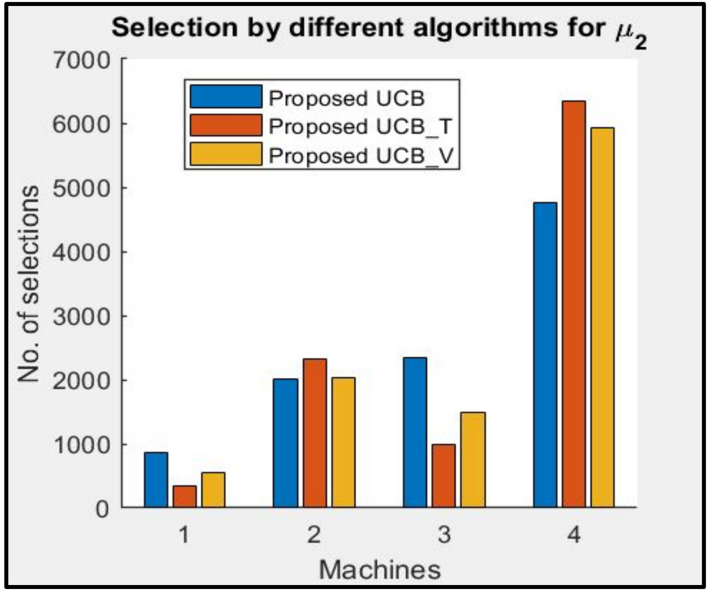
- No. of active CB Calculation blocks in **Velcro** = **12**
- No. of active CB calculation blocks in **Proposed** = **4**

Case II: Only **3** machines active

- No. of active CB Calculation blocks in **Velcro** = **12**
- No. of active CB calculation blocks in **Proposed** = **3**

Proposed - Functional Analysis

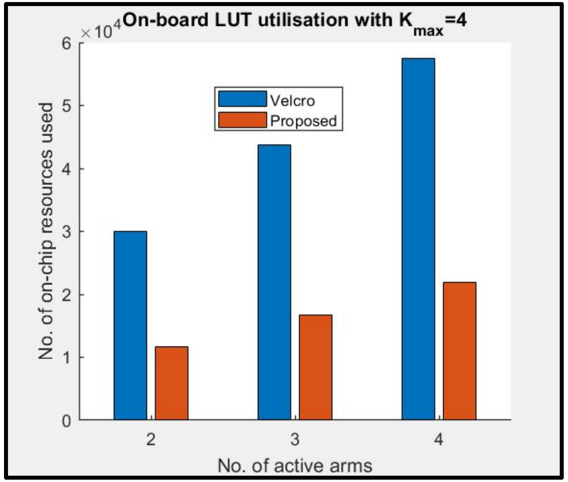


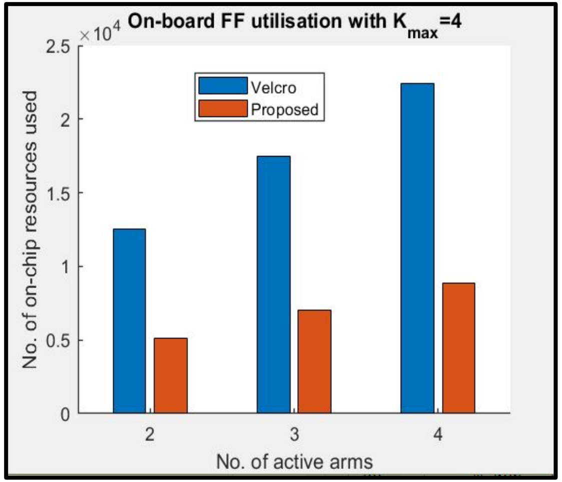


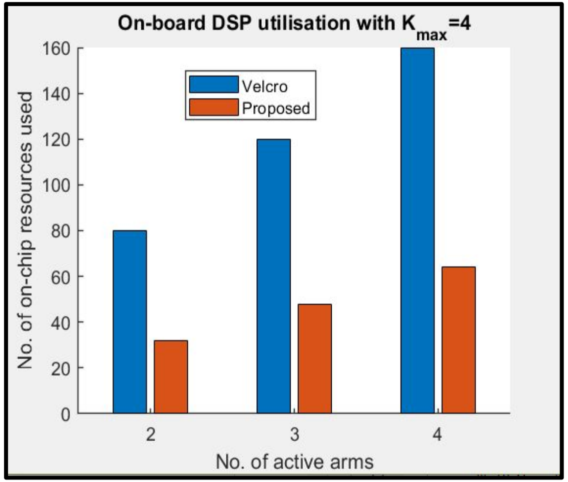
Considering probability distributions with four machines each:
(1) $\mu_1 = \{0.4, 0.5, 0.6, 0.7\}$ (2) $\mu_2 = \{0.78, 0.83, 0.81, 0.85\}$

As can be seen, the proposed architecture consistently **selects the best machine** from both the distributions without the prior knowledge of the probability of success for each of them.

Proposed vs Velcro - Resource Utilisation



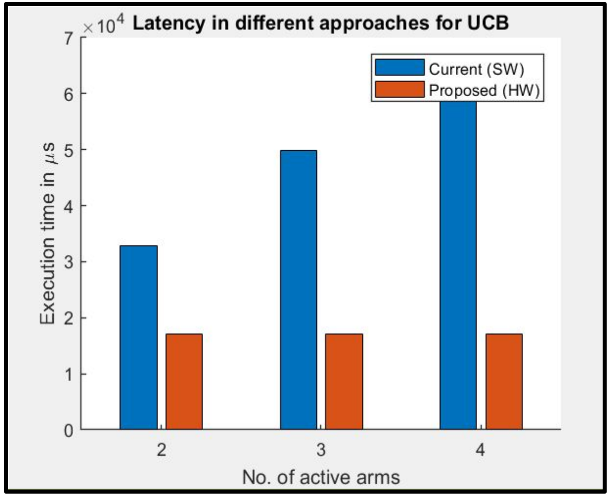


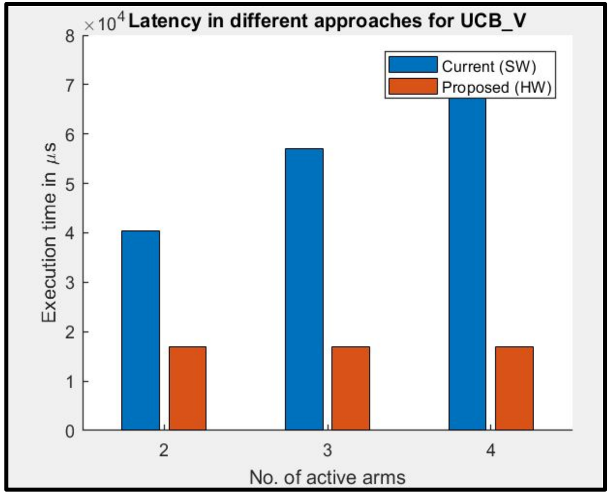


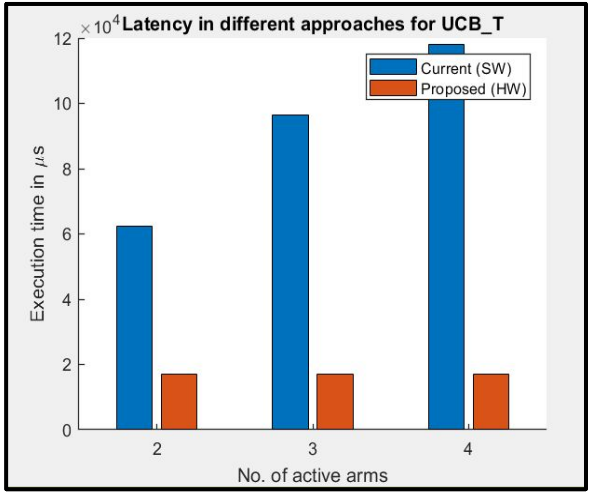
Considering an MAB setting with four machines & a choice of three algorithm options available & we want the ability to switch between algorithms, the options being:

As can be seen, the proposed architecture results in **huge savings** in terms of **resources** used & hence the **dynamic power** consumed by the setting. This is the case with $\mathbf{K_{max}}$**=4**, the savings will increase drastically in real-life settings with $\mathbf{K_{max}}$ **> 20**.

Proposed vs Current - Latency



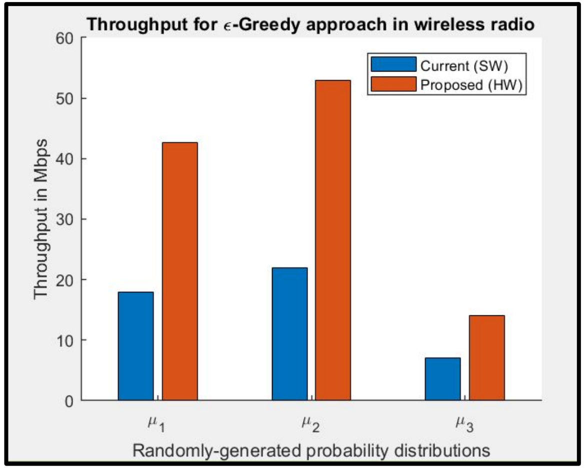


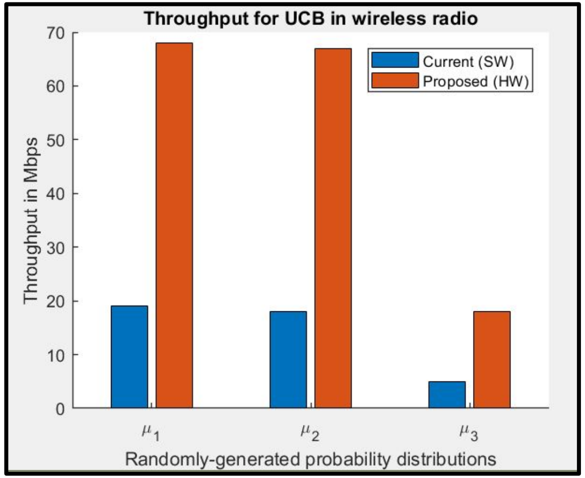


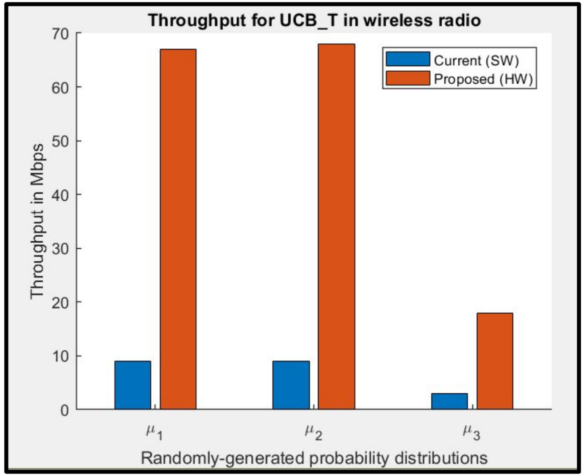
Considering an MAB setting with four machines

As can be seen, the proposed architecture results in **huge savings** in terms of **latency incurred** for CB calculation because of the parallel calculation in the proposed approach. This is the case with $\mathbf{K_{max}=4}$, the savings will increase drastically in real-life settings with $\mathbf{K_{max} > 20}$.

Power of MAB coupled with PHY - Throughput







Considering randomly-generated probability distributions $\boldsymbol{\mu_1}, \boldsymbol{\mu_2}, \boldsymbol{\mu_3}$. We use the proposed architecture in **cognitive ad-hoc wireless radio** for **MAB-based optimum channel selection** for throughput maximization.

- **ϵ-Greedy** is one of the heuristic algorithms popularly used in wireless networks.

The proposed architecture significantly **higher throughput** as compared to the current approach.