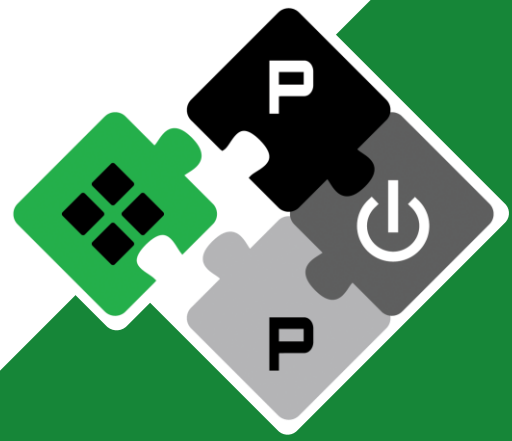
ETH zürich ALMA MATER STUDIORUM UNIVERSITÀ DI BOLOGNA

# CachePool: Many-core cluster of customizable, lightweight scalar-vector PEs for irregular L2 data-plane workloads


Integrated Systems Laboratory (ETH Zürich)

**Zexin Fu, Diyou Shen** zexifu, dishen@iis.ee.ethz.ch

**Alessandro Vanelli-Coralli** avanelli@iis.ee.ethz.ch
**Luca Benini** lbenini@iis.ee.ethz.ch

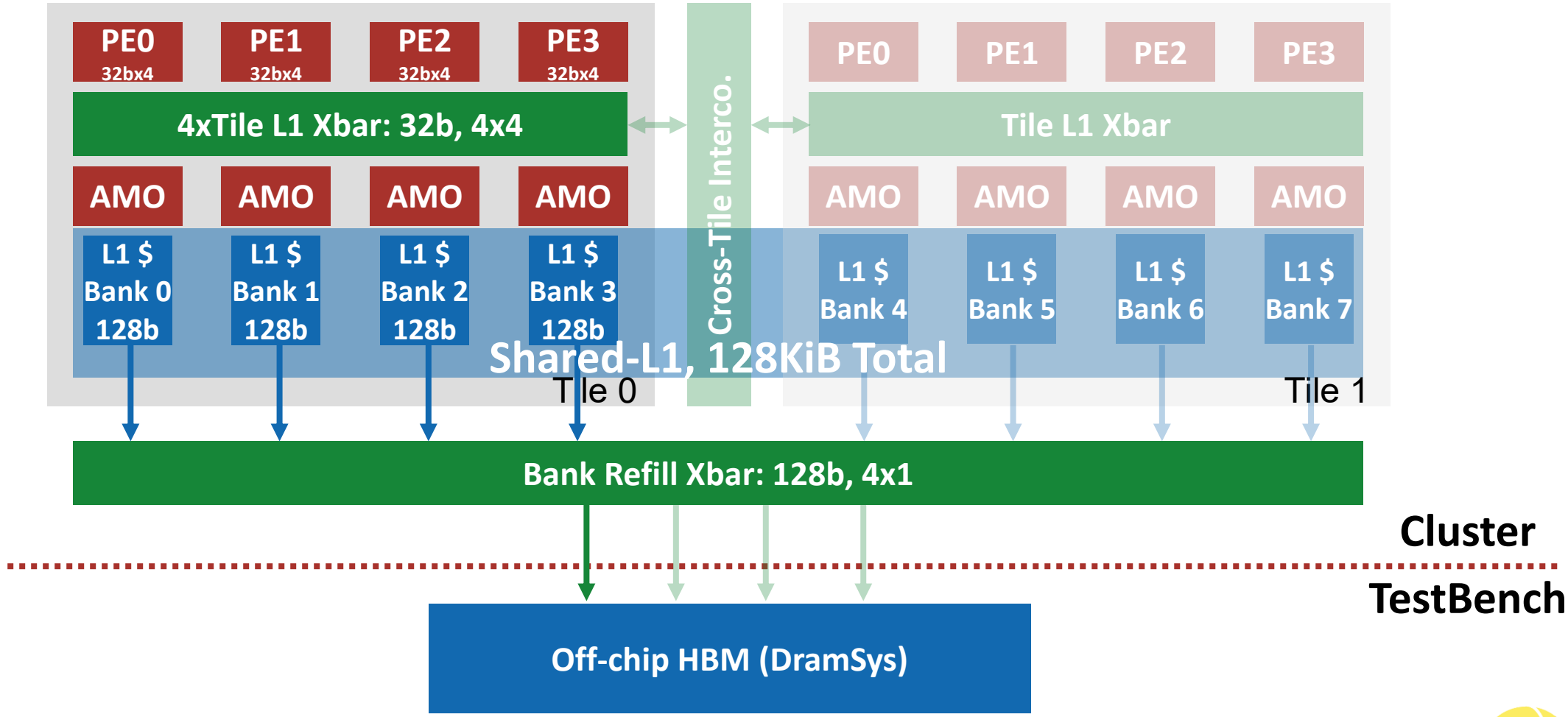
@pulp_platform

pulp-platform.org

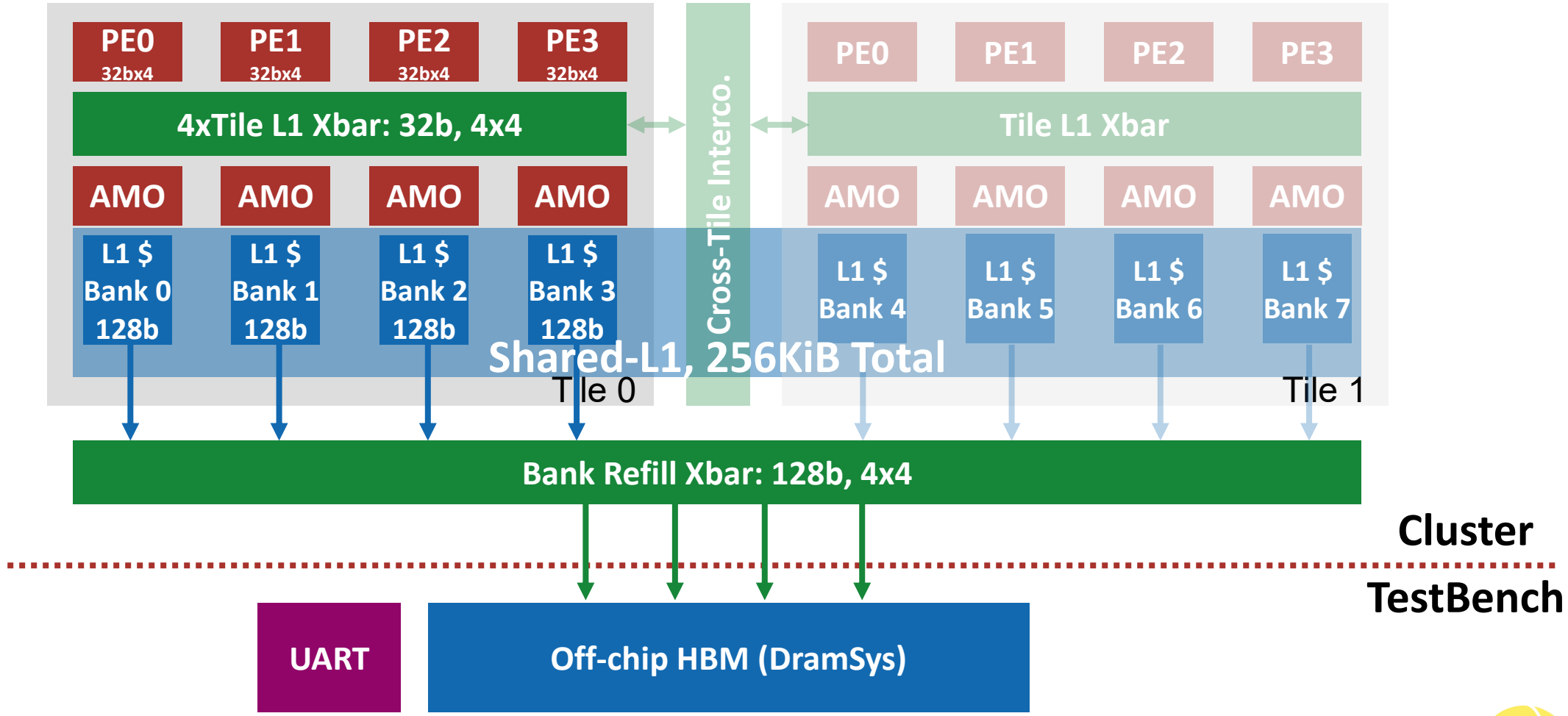
youtube.com/pulp_platform

**PULP Platform**
Open Source Hardware, the way it should be!

# Hardware Development



# Hardware Development

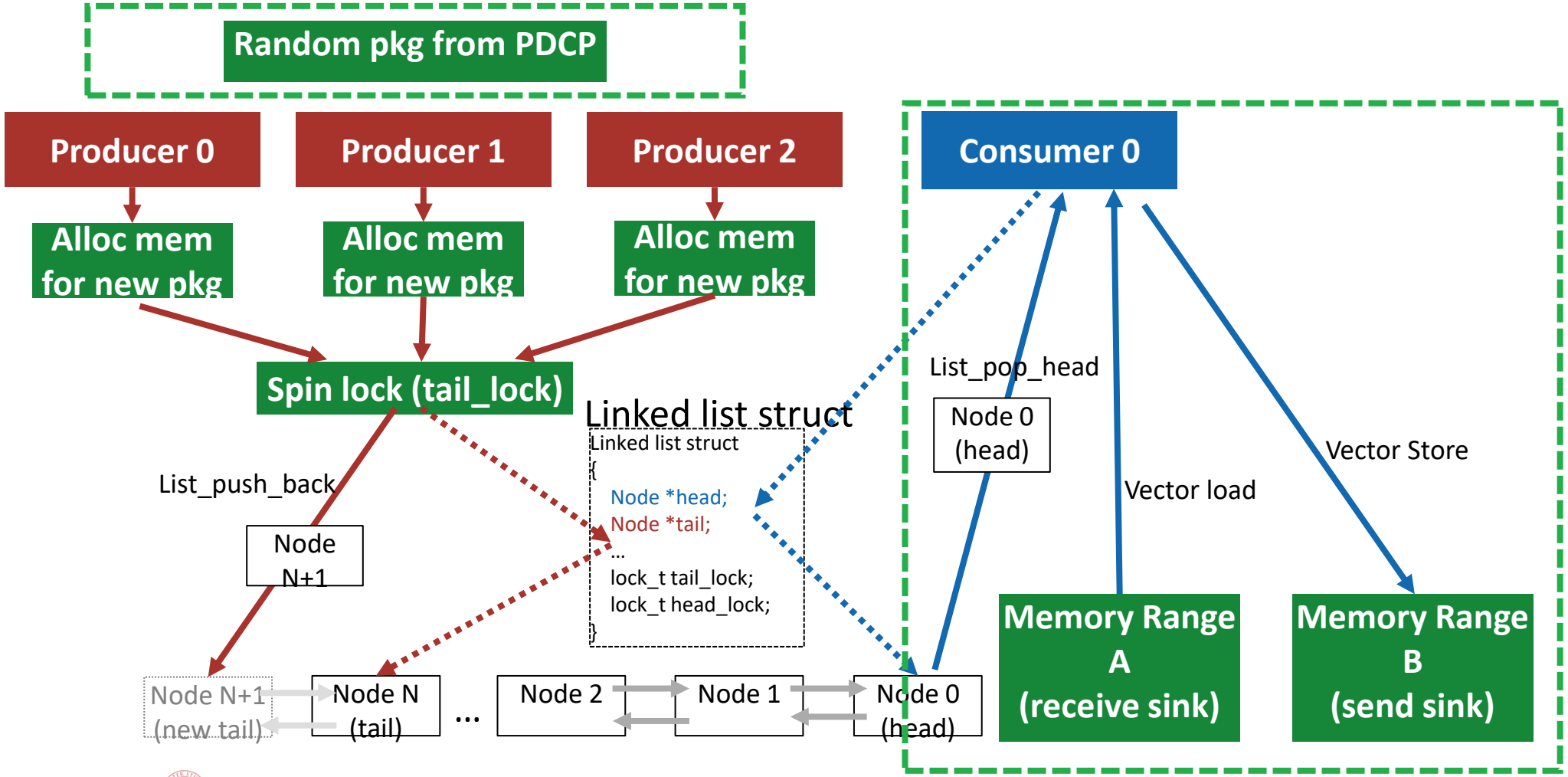


# Hardware Development

- **Complete**
  - Integration of DramSys Simulator (100%)
  - Cache-Refill Xbar
  - UART module and printing support
- **In Progress**
  - Code Cleanup: Remove the remaining modules from old cluster
    - DMA, SPM interconnection, stack modification
  - PPA Analysis: Start with the existing 12nm flow
    - Faster setup
    - Easier area comparison with previously collected data
  - Transit to 7nm flow for delivery
- **TODO**
  - Partition support

# Software Development

## Use Spatz VLSU to move data, simulate the RLC pkg sending process



# Random pkg from PDCP

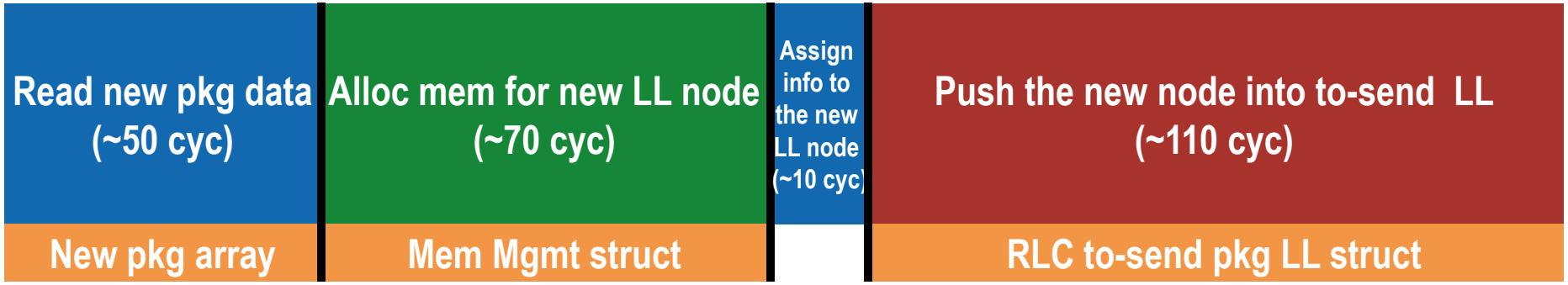
- **Pre generated by a python script**
  - Random for the user id, src and tgt address
- **Configurable by a json file**

```
{
  "active_user_number": 1,
  "pkg_length": 1350,
  "pdcp_header_length": 6,
  "src_addr": "0xA0000000",
  "src_length": 268435456, // 268435456 bytes = 256 MiB
  "tgt_addr": "0xB0000000",
  "tgt_length": 268435456, // 268435456 bytes = 256 MiB
  "total_pkg_number": 1000 // max 198400 packages =
  268435456 bytes / (1350 + 3) bytes per package
}
```

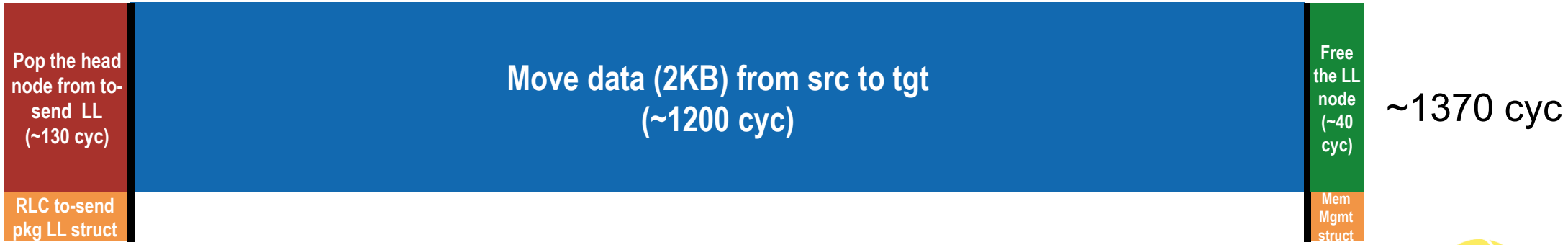
```
typedef struct {
    int          user_id;
    unsigned int src_addr;
    unsigned int tgt_addr;
    unsigned int pkg_length;
} pdcp_pkg_t;

#define NUM_SRC_SLOTS 131072
#define PDU_SIZE      2048
#define NUM_PKGS      10

static const pdcp_pkg_t __attribute__((section(".pdcp_info"))) pdcp_pkgs[NUM_PKGS] = {
    { 0, 0xA2735000, 0xB2735000, 2048 },
    { 0, 0xA4EDB800, 0xB4EDB800, 2048 },
    { 0, 0xA3EE2000, 0xB3EE2000, 2048 },
    { 0, 0xAB0F6000, 0xBB0F6000, 2048 },
    { 0, 0xA0E8B800, 0xB0E8B800, 2048 },
    { 0, 0xA5B24000, 0xB5B24000, 2048 },
    { 0, 0xAF899800, 0xBF899800, 2048 },
    { 0, 0xA1CFC800, 0xB1CFC800, 2048 },
    { 0, 0xA3652800, 0xB3652800, 2048 },
    { 0, 0xAE3DC000, 0xBE3DC000, 2048 },
};

static const uint8_t __attribute__((section(".pdcp_src"))) pdcp_src_data[NUM_SRC_SLOTS][PDU_SIZE] = {
    [7447] = { 0x00, 0x00, 0x00, 0x00, 0x17, 0x17, 0x17, 0x17, 0x17, 0x17, 0x17, 0x17, 0x17, 0x17, 0x1
    [14841] = { 0x00, 0x00, 0x00, 0x00, 0xF9, 0xF9, 0xF9, 0xF9, 0xF9, 0xF9, 0xF9, 0xF9, 0xF9, 0xF9, 0x
    [20074] = { 0x00, 0x00, 0x00, 0x00, 0x6A, 0x6A, 0x6A, 0x6A, 0x6A, 0x6A, 0x6A, 0x6A, 0x6A, 0x6A, 0x
    [27813] = { 0x00, 0x00, 0x00, 0x00, 0xA5, 0xA5, 0xA5, 0xA5, 0xA5, 0xA5, 0xA5, 0xA5, 0xA5, 0xA5, 0x
    [32196] = { 0x00, 0x00, 0x00, 0x00, 0xC4, 0xC4, 0xC4, 0xC4, 0xC4, 0xC4, 0xC4, 0xC4, 0xC4, 0xC4, 0x
    [40375] = { 0x00, 0x00, 0x00, 0x00, 0xB7, 0xB7, 0xB7, 0xB7, 0xB7, 0xB7, 0xB7, 0xB7, 0xB7, 0xB7, 0x
    [46664] = { 0x00, 0x00, 0x00, 0x00, 0x48, 0x48, 0x48, 0x48, 0x48, 0x48, 0x48, 0x48, 0x48, 0x48, 0x
    [90604] = { 0x00, 0x00, 0x00, 0x00, 0xEC, 0xEC, 0xEC, 0xEC, 0xEC, 0xEC, 0xEC, 0xEC, 0xEC, 0xEC, 0x
    [116664] = { 0x00, 0x00, 0x00, 0x00, 0xB8, 0xB8, 0xB8, 0xB8, 0xB8, 0xB8, 0xB8, 0xB8, 0xB8, 0xB8, 0
    [127283] = { 0x00, 0x00, 0x00, 0x00, 0x33, 0x33, 0x33, 0x33, 0x33, 0x33, 0x33, 0x33, 0x33, 0x33, 0
};
```

# Packet Receiving Task Preliminary Timing Breakdown

- **Producer: Receive new pkg from PDCP layer, add its info into to-send LL**

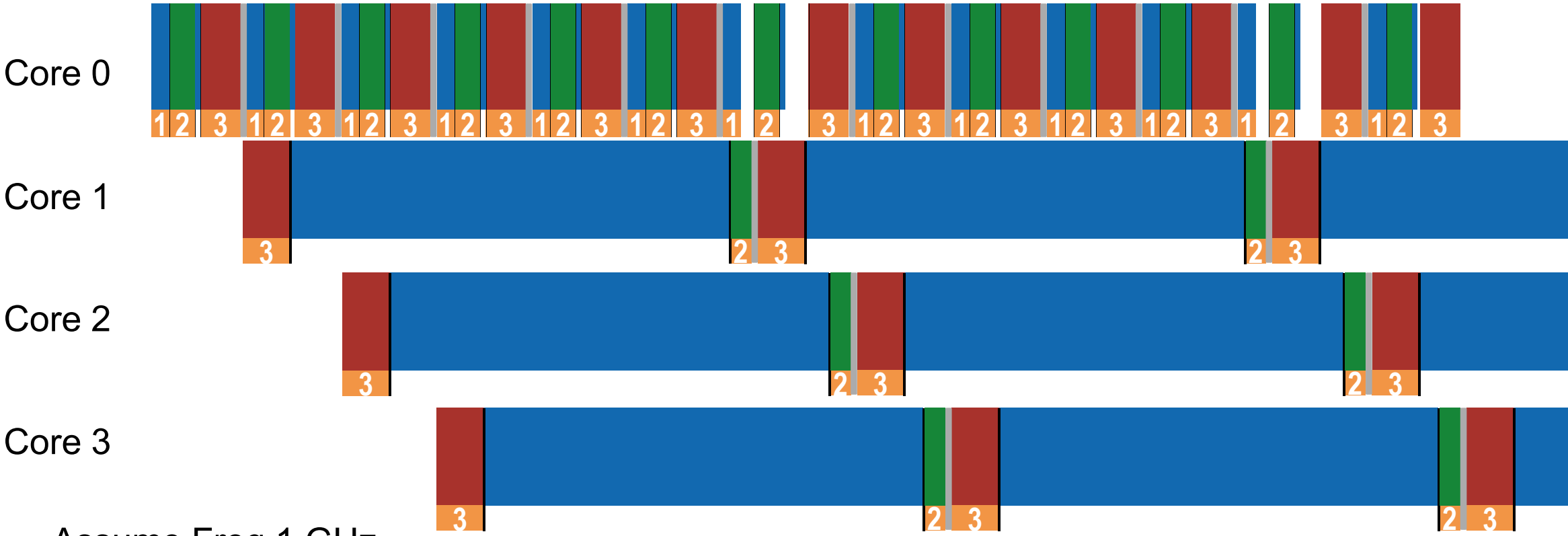



- **Consumer: Pop the pkg info from the LL, move pkg data (transmit to UE)**



# Packet Receiving Task Preliminary Timing Breakdown

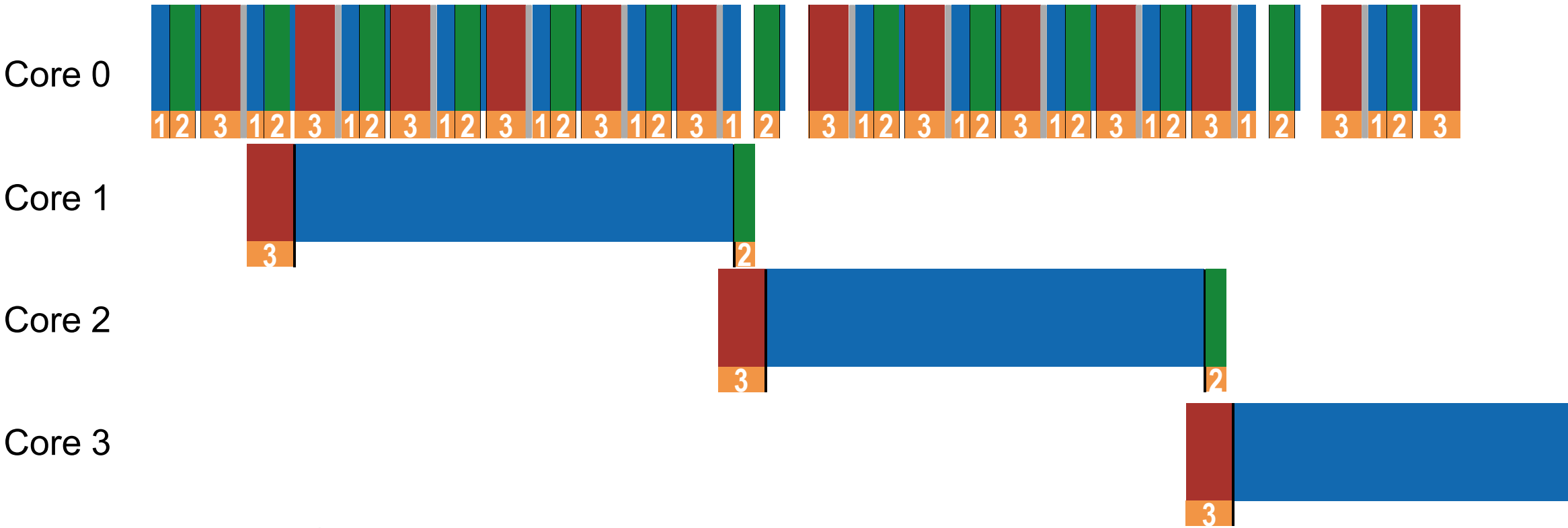
- **One tile: 4 core complex (1 snitch + 1 spatz)**




- Assume Freq 1 GHz
- BW 2142857pkg/s * 2KB = 4GBps (for now not enough, challenge: data movement latency)

# Packet Receiving Task Preliminary Timing Breakdown

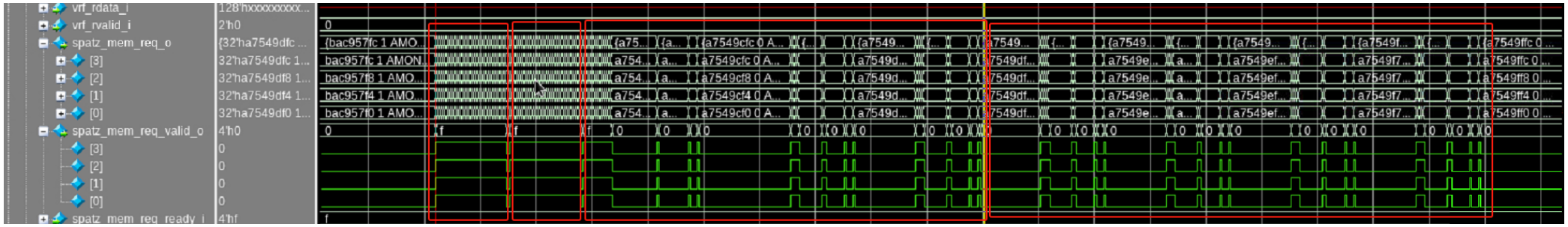
- **One tile: 4 core complex (1 snitch + 1 spatz)**




- Assume Freq 1 GHz
- BW 2142857pkg/s * 2KB = 4GBps (for now not enough, challenge: data movement latency)

# Memory Movement Challenge

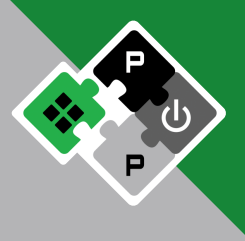
- **Each Spatz can only achieve 2 byte/cycle (max bw is 16 byte/cycle)**
  - Bottleneck: The size of Spatz VLSU ROB can handle only 256 – 512 byte in-flight load
    - Once the ROB is full, the VLSU has to wait until the previous load data back.




  - Possible solution
    - Redesign the ROB of the VLSU, make it more scalable and can handle more in-flight load
    - Offload the memory moving operations to an extra controller (prefetcher/dma/cache controller …)

# Software Development

- **Complete**
  - Random generated pkg from PDCP layer
  - Design and evaluate the RLC data management kernel with VLSU for data movement
- **In Progress**
  - GEMM and GEMV kernel performance calibration: HW is not stable for now
  - The sent linked-list for the RLC data management kernel
- **TODO**
  - Explore the way to fully utilize the memory BW for RLC pkg data movement



# Thank you!