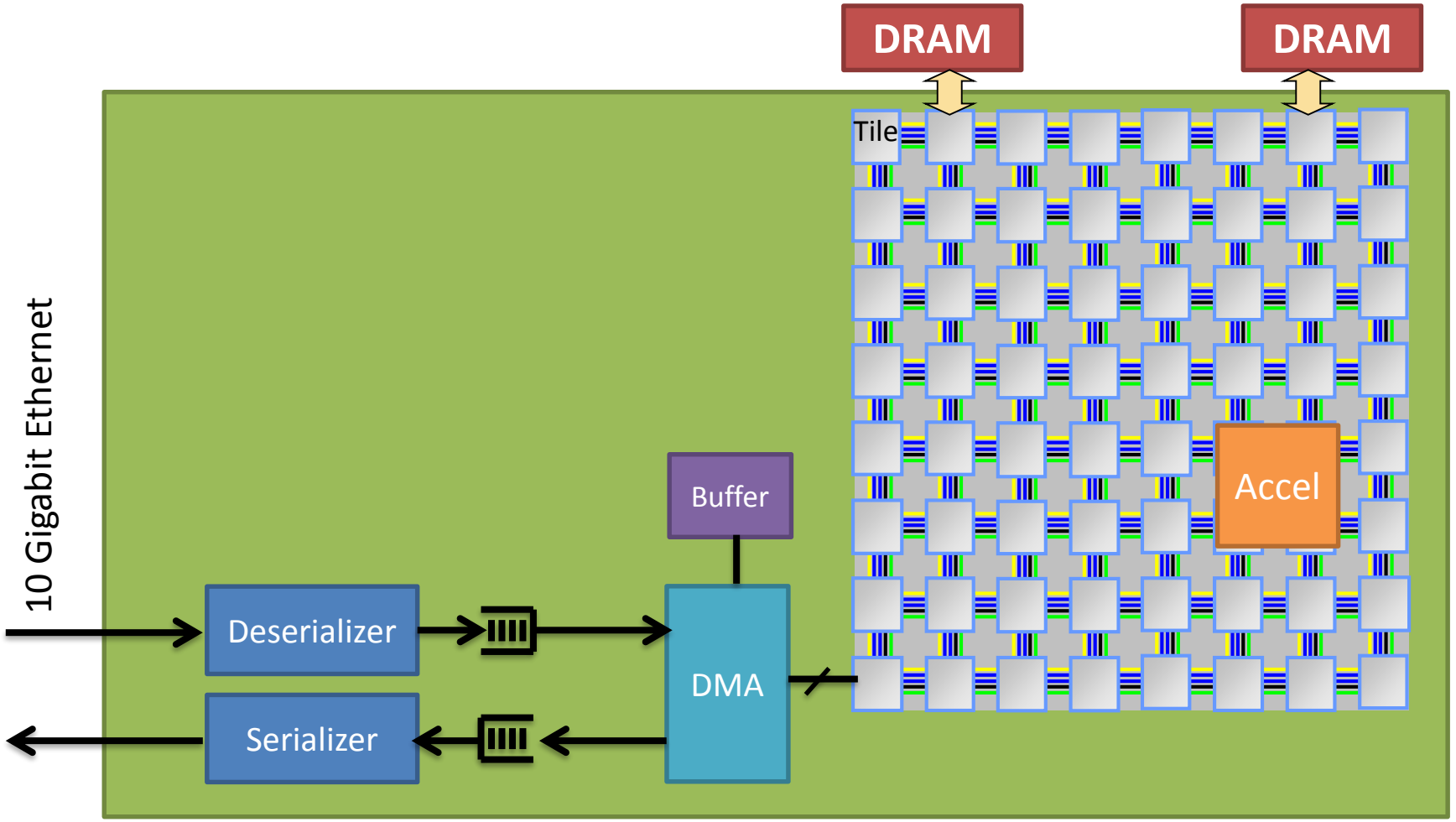
Getting to Work with OpenPiton

Princeton University

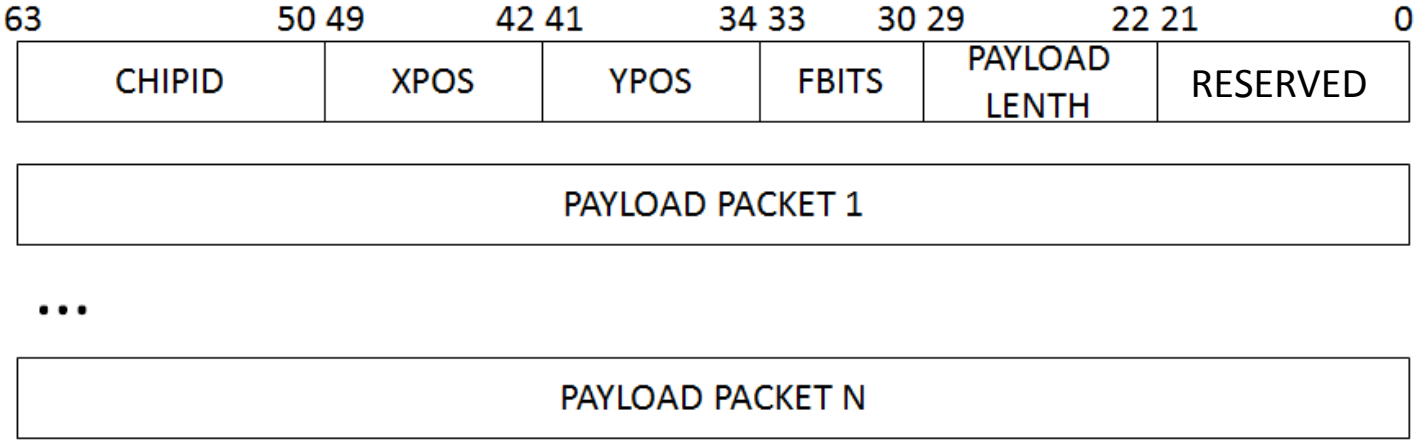
http://openpiton.org

Extension Using NoCs

NoC Connected I/O and Accelerators



NoC: packet format



CHIPID: Highest bits indicate whether the destination is on-chip or off-chip, the rest of the bits indicates the chip ID

XPOS: The position of the destination tile in the X dimension

YPOS: The position of the destination tile in the Y dimension

FBITS: The router output port to the destination

PAYLOAD LENGTH: The number of payload packets

RESERVED: Reserved Bits used by higher-level protocols.

NoC: .h files

piton/design/include/network_define.h

Defines the header flits b63-22

(all except messageid, tag, and options 1)

piton/design/include/define.vh

defines the rest

```
181   //Memory requests from L2 to DRAM
182   `define MSG_TYPE_LOAD_MEM          8'd19
183   `define MSG_TYPE_STORE_MEM         8'd20
```

```
196   //Memory acks from memory to L2
197   `define MSG_TYPE_LOAD_MEM_ACK      8'd24
198   `define MSG_TYPE_STORE_MEM_ACK     8'd25
199   `define MSG_TYPE_NC_LOAD_MEM_ACK   8'd26
200   `define MSG_TYPE_NC_STORE_MEM_ACK  8'd27
```

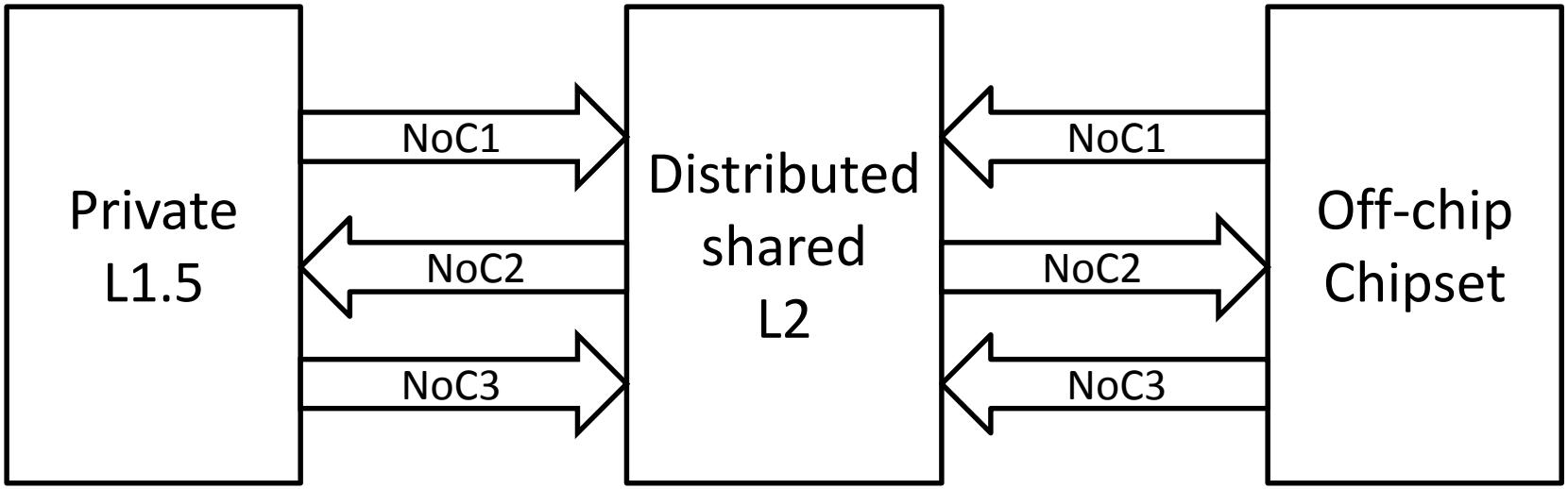
```
144   //Requests from L15 to L2
145   // Should always make #0 an error
146   `define MSG_TYPE_RESERVED          8'd0
147   `define MSG_TYPE_LOAD_REQ          8'd31
148   `define MSG_TYPE_PREFETCH_REQ      8'd1
149   `define MSG_TYPE_STORE_REQ         8'd2
150   `define MSG_TYPE_BLK_STORE_REQ     8'd3
151   `define MSG_TYPE_BLKINIT_STORE_REQ 8'd4
152   `define MSG_TYPE_CAS_REQ           8'd5
153   `define MSG_TYPE_CAS_P1_REQ        8'd6
154   //condition satisfied
155   `define MSG_TYPE_CAS_P2Y_REQ       8'd7
156   //condition not satisfied
157   `define MSG_TYPE_CAS_P2N_REQ       8'd8
158   //Both SWAP and LDSTUB are the same for L2
159   `define MSG_TYPE_SWAP_REQ          8'd9
160   `define MSG_TYPE_SWAP_P1_REQ       8'd10
161   `define MSG_TYPE_SWAP_P2_REQ       8'd11
162   `define MSG_TYPE_WB_REQ            8'd12
163   `define MSG_TYPE_WBGUARD_REQ       8'd13
164   `define MSG_TYPE_NC_LOAD_REQ       8'd14
165   `define MSG_TYPE_NC_STORE_REQ      8'd15
166   `define MSG_TYPE_INTERRUPT_FWD      8'd32
```

Cache Coherence Protocol

Directory-based MESI coherence Protocol

- Four-hop message communication (no direct communication between private L1.5 caches)
- Uses 3 physical NoCs with point-to-point ordering to avoid deadlock
- The directory and L2 are co-located but state information are maintained separately
- Silent eviction in E and S states
- No need for acknowledgement upon write-back of dirty lines from L1.5 to L2, but writeback guard needed in some cases.

Memory Hierarchy Datapath



NoC Messages

In order to avoid deadlock, NoC3 messages will never be blocked