Neuromorphic Computing

5. Communication Networks for Neuromorphic Systems (Part – I and II)

Ben Abdallah Abderazek, Khanh N. Dang
E-mail: {benab, khanh}@u-aizu.ac.jp

This lecture is based on the book ''Neuromorphic Computing Principles and Organization,'' Publisher: Springer; 2022 edition, ISBN-10 : 3030925242, ISBN-13 : 978-3030925246, by Abderazek Ben Abdallah, Khanh N. Dang

Lecture Contents

1. Introduction

- The brain connectivity is generally described at several levels of scale:
  - Synaptic connections that link individual at the **microscale**,
  - Networks connecting neuronal populations at the **mesoscale**,
  - Brain regions linked by fiber pathways at the **macroscale**
- Designing communication needs:
  - **High bandwidth** for exchanging massive amount of spikes
  - **Low-latency** to ensure the correctness of arrival time of spikes

1. Introduction
Type of connections

- Synaptic connections at the **microscale:**
  - **Spike vector:** in each time-step, a vector of firing (0 for non firing, 1 for firing) is sent to each neuron.
  - **Address Event Representation:** sending the address of firing neuron whenever it fires.
- Networks connecting neuronal populations at the **mesoscale and** Brain regions linked by fiber pathways at the **macroscale:**
  - On-chip communications: Bus, Point-to-Point, On-Chip Network.
  - Off-chip communications: direct off-chip link, LAN (via adapter), extneding on-chip communications

1. Introduction
AER: Address Event Representation

- One of the first chip design for neuromorphic system using so-called "Address Event Representation" (AER) for the off-chip communication.
    - The content of the package is the address of the firing neuron.
    - Only send the package once a neuron fired
    - For an N axonal fiber, with one active at a time, AER replaces regular wire with (1+log N) wires

1. Introduction
Overview of AER

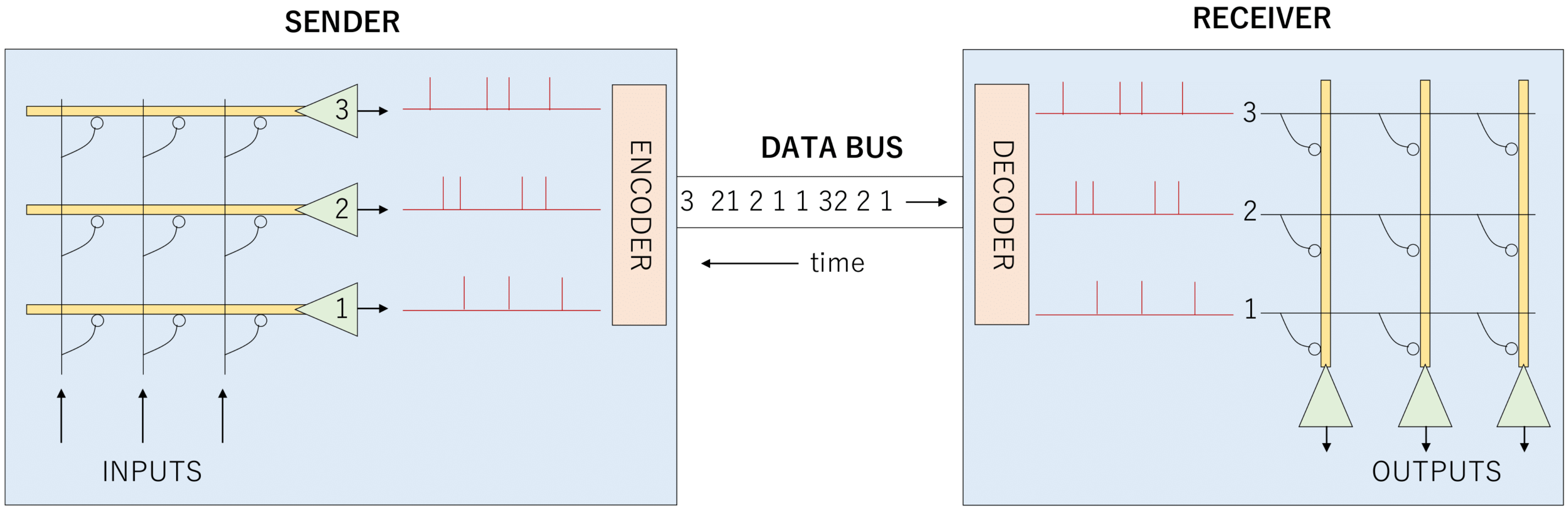
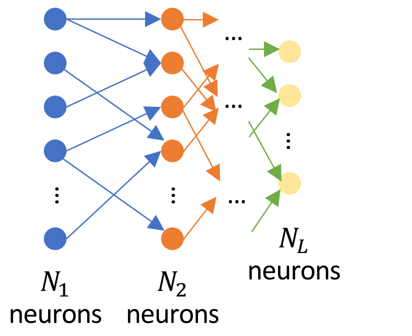


Fig. 5.1: Address-Event Representation (AER) Protocol.

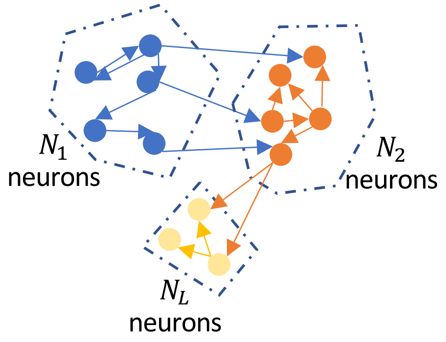
- Instead of sending spike vector, address of the firing neuron is sent

1. Introduction
Spike Vector vs AER



$N = \sum_{i=1...L} N_i$ neurons in total

Feedforward network



$N = \sum_{i=1...L} N_i$ neurons in total

Liquid State Network

Spike Vector:

- With N neurons, we need N-bit vector for each time step:
  - 1: fired
  - 0: not fired
- Group-wise vector:
  - $\log_2(L)$ bits for group index
  - $\max(N_i)$ bits for group vector

AER:

- With N neurons, we need $(\log_2(N) + 1)$-bit vector for each time step:
  - First bit: 1: has spike; 0: no spike
- Group-wise AER:
  - $\log_2(L)$ bits for group index
  - $\log_2 (\max(N_i))$ bits for group vector

Lecture Contents

2. Neural Communication
Spike and model

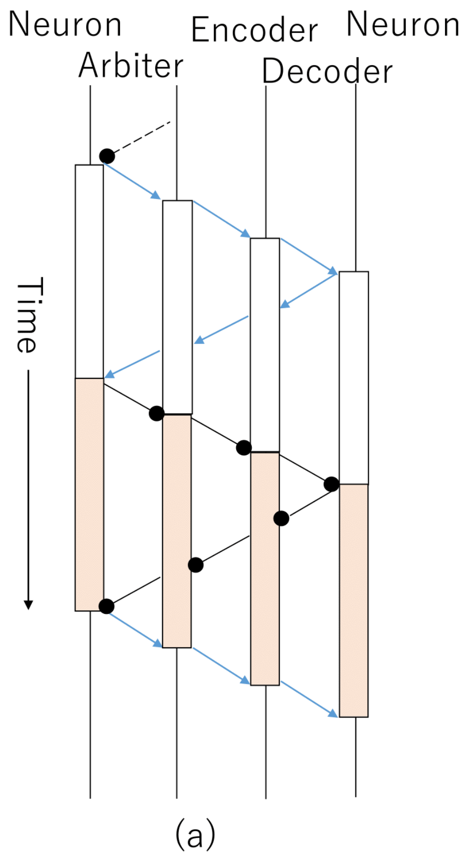
- Biological neurons communicate predominantly via an electrochemical **impulse** known as an **action potential** or **spike**
- Silicon neurons usually follow the **‘point neuron model’:**
  - the details of dendrite structures are ignored,
  - assume all inputs effectively arrive at the neuron.
- Usually, there is no global clock, signals are sent **asynchronously**.
  - With digital-base system, clock could be implemented for synchronization if neccessary

2. Neural Communication
Handshaking

- To communicate between two chips or two neuron clusters, the request and acknowledgment protocols are typically use
    1. a request signal is sent
    2. AER signals are transmitted
    3. AER signals is received and stored
    4. Acknowledgement signals are sent back
- Pipelining can be used for shorter execution time.

2. Neural Communication

Handshaking



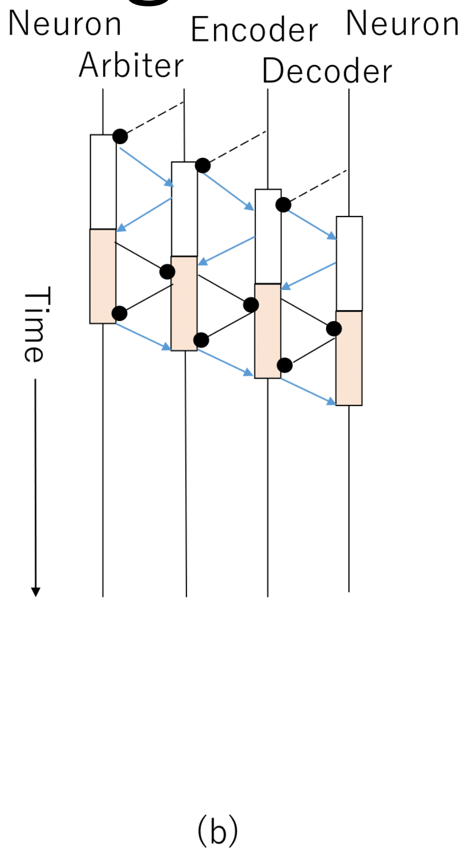
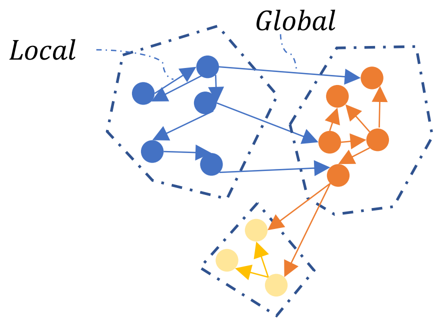


Fig. 5.2: Control signal flow starting from a neuron through the arbiter and encoder on on the transmitting side, to the decoder and a neuron on the receiving side (left to right). (a) Completion of the spike transmission for the originating neuron before the handshaking is completed in the acknowledge phase. (b) Pipelining reduces the overall handshaking time by allowing the signals to propagate forward in the set phase without waiting for the acknowledge signal.

2. Neural Communication
Global and Local Communication

- There are two major types of neural communication
  - Local: within the cluster of neurons
    - Consisting of short range routing path
  - Global: between the cluster of neurons
    - Consisting of long range routing path



Lecture Contents

3. Interconnects for Neural Communication

Overview

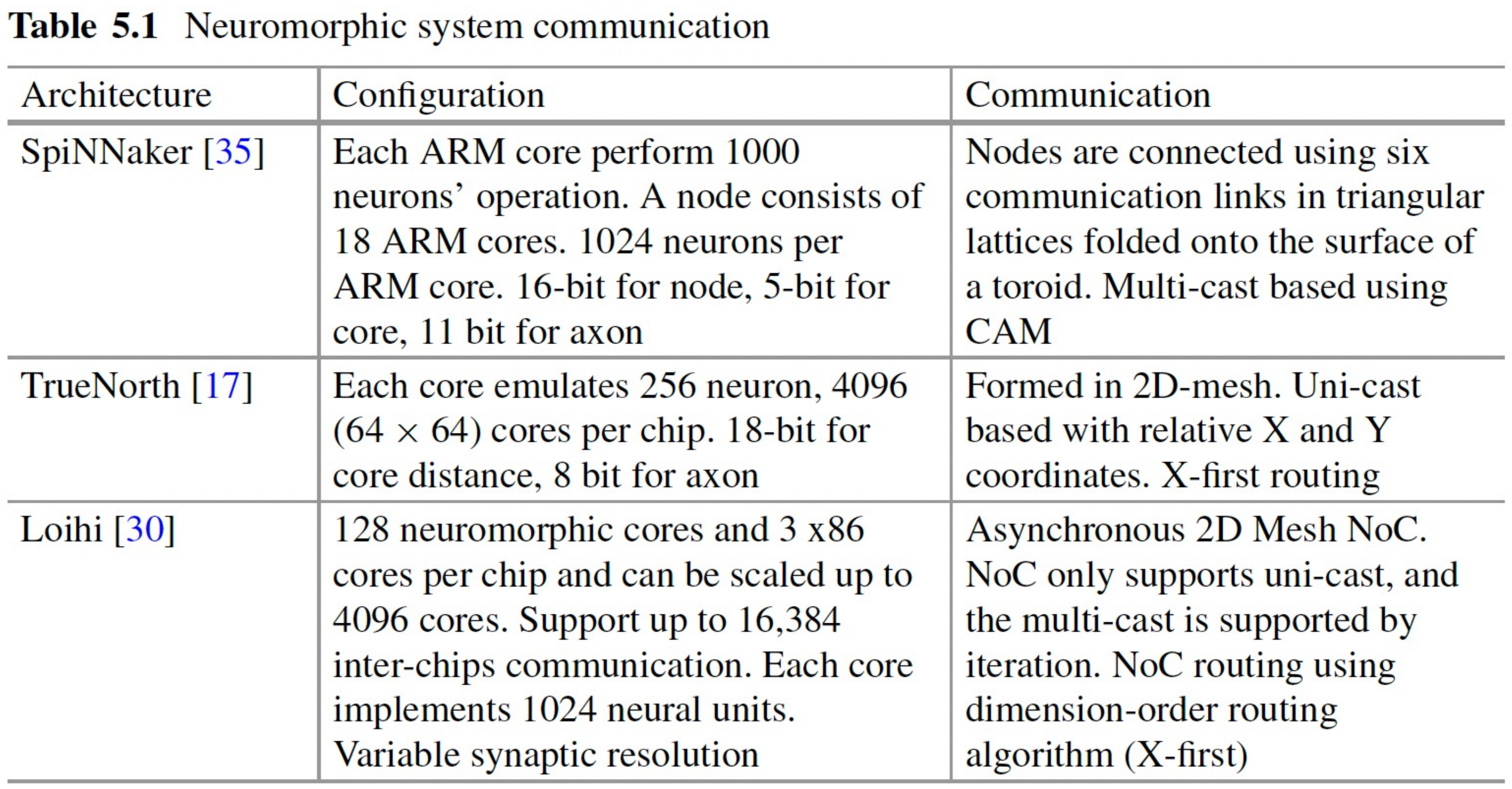


Table 5.1 Neuromorphic system communication

Architecture	Configuration	Communication
SpiNNaker [35]	Each ARM core perform 1000 neurons' operation. A node consists of 18 ARM cores. 1024 neurons per ARM core. 16-bit for node, 5-bit for core, 11 bit for axon	Nodes are connected using six communication links in triangular lattices folded onto the surface of a toroid. Multi-cast based using CAM
TrueNorth [17]	Each core emulates 256 neuron, 4096 (64 × 64) cores per chip. 18-bit for core distance, 8 bit for axon	Formed in 2D-mesh. Uni-cast based with relative X and Y coordinates. X-first routing
Loihi [30]	128 neuromorphic cores and 3 x86 cores per chip and can be scaled up to 4096 cores. Support up to 16,384 inter-chips communication. Each core implements 1024 neural units. Variable synaptic resolution	Asynchronous 2D Mesh NoC. NoC only supports uni-cast, and the multi-cast is supported by iteration. NoC routing using dimension-order routing algorithm (X-first)

3. Interconnects for Neural Communication

SpiNNaker

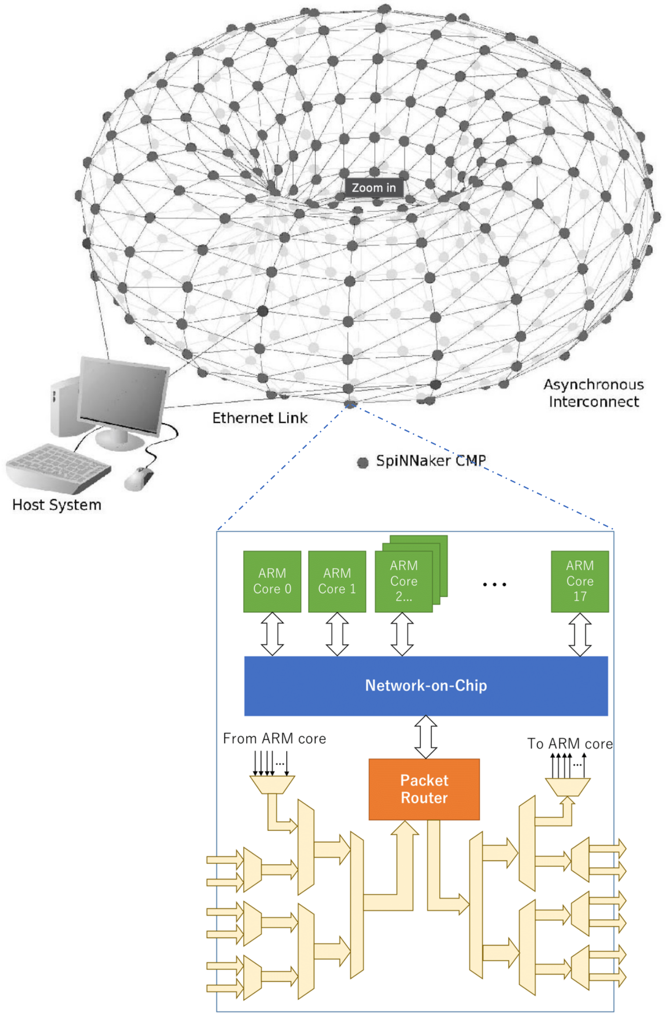
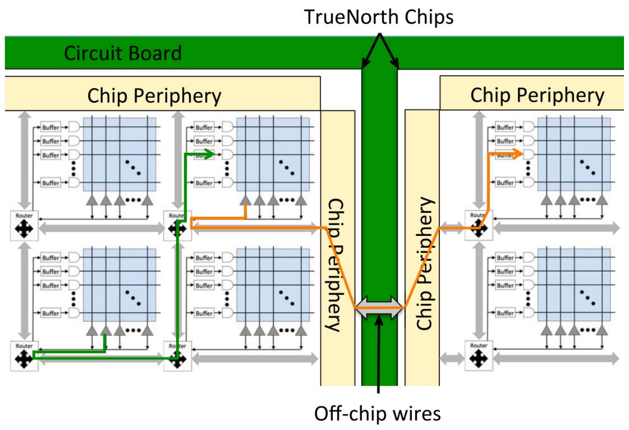


Fig. 5.3: SpiNNaker

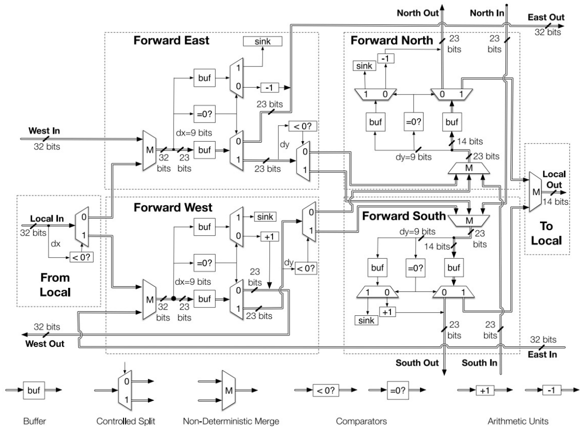
- Each node has 18 ARM cores connected via Network-on-Chip
- Off-chip communication using packet-router modules
- Scale up to 57,600 processing node (1 million cores)
- Adopts the AER protocol as the central idea
- Four types of packets:
  - Nearest neighbors
  - Point-to-Point
  - Neural event multi-cast
  - Fixed route

3. Interconnects for Neural Communication
TrueNorth

- Each TrueNorth chip consists of 4096 cores connected via Network-on-Chip (asynchronous 2D Mesh)
- Each neuron has 256 programmable synapses that emulate the strength between two neurons
- TrueNorth uses the X-first routing algorithm where the relative distance in X-coordinate (dx) value is increased or decreased first until it gets to zero.
  - After dx became zero, routing in Y-dimension is used.
- Off-chip connection is considered as of extending the Mesh Network-on-Chip



On-chip and off-chip interconnect

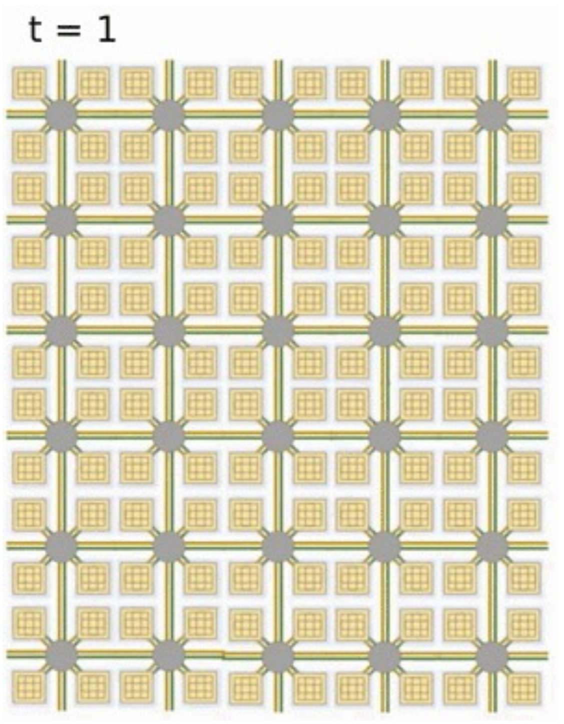


Router Architecture

F. Akopyan et al., "TrueNorth: Design and Tool Flow of a 65 mW 1 Million Neuron Programmable Neurosynaptic Chip," in IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, vol. 34, no. 10, pp. 1537-1557, Oct. 2015, doi: 10.1109/TCAD.2015.2474396.

3. Interconnects for Neural Communication
Loihi-1

- Loihi-1 chip consists of 3 x86 cores and 128 neuromorphic cores connected via asynchronous 2D NoC
- Each neuron has 256-1024 programmable synapses that emulate the strength between two neurons
- Off-chip connection is considered as of extending the Mesh Network-on-Chip



Loihi-1 NoC

https://en.wikichip.org/wiki/intel/loihi

Lecture Contents

4. Interconnect Design Principle
Major principles

When designing interconnect, followings are the major design choices:

- **Network topology:** For example, *SpiNNaker* uses a folded triangle lattice topology, *TrueNorth* and *Loihi-1* use Mesh topology.
- **Classification of the communications:** beside inter-neural communication (i.e. spikes), other communication such as data transaction or debugging is needed.

4. Interconnect Design Principle
Major principles (cnt)

- **Network support for communication:**
  - *Multi-cast*
  - *Uni-cast*
  - *Broadcast*
- **Time constraints**
  - Spikes in neuromorphic systems must arrive before a predefine time (as synchronization)
  - There two type: local and long-range communication. Long-range communication take long time to arrive which may violate the time constraint.
- **Mapping**
- **Fault-tolerance**

4. Interconnect Design Principle
OSI Model

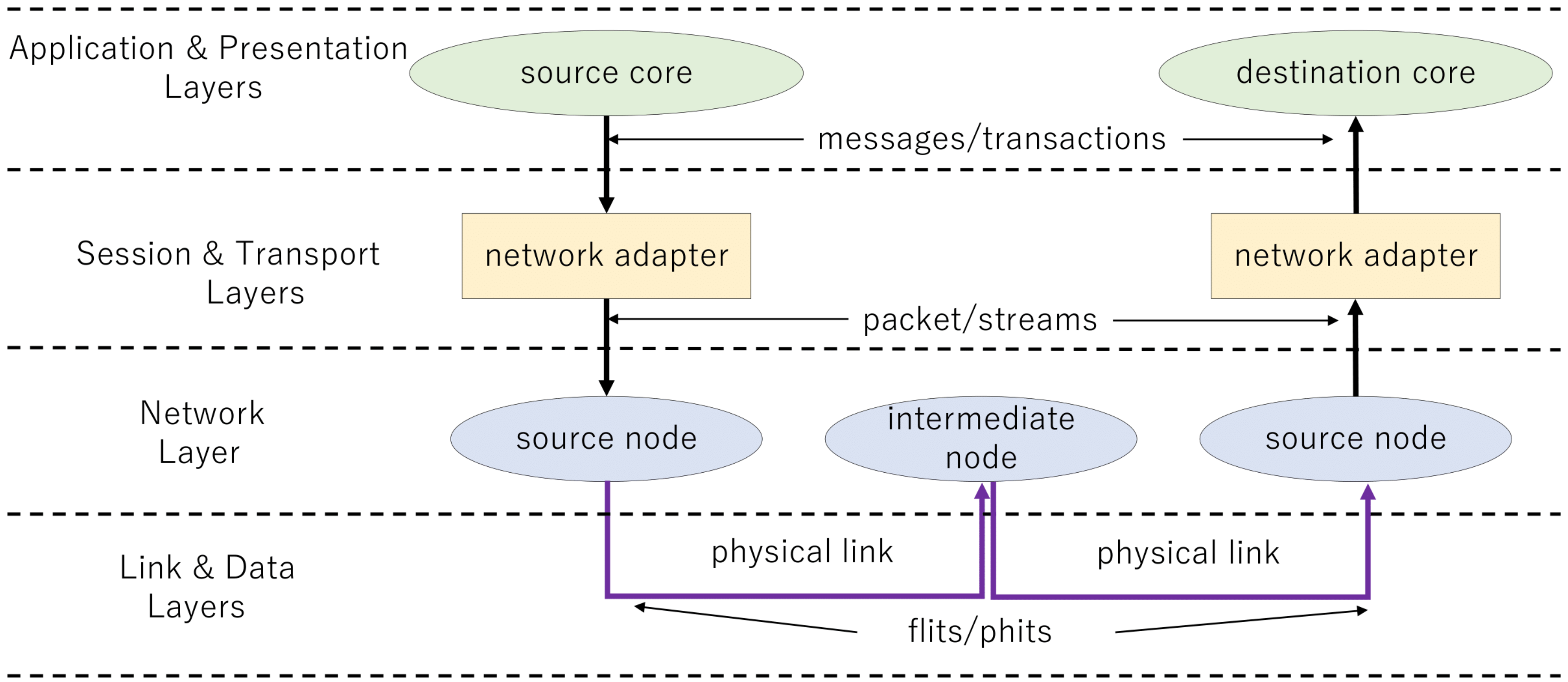


Fig. 5.5: OSI reference model for Network-on-Chip.

4. Interconnect Design Principle

Mapping Issue

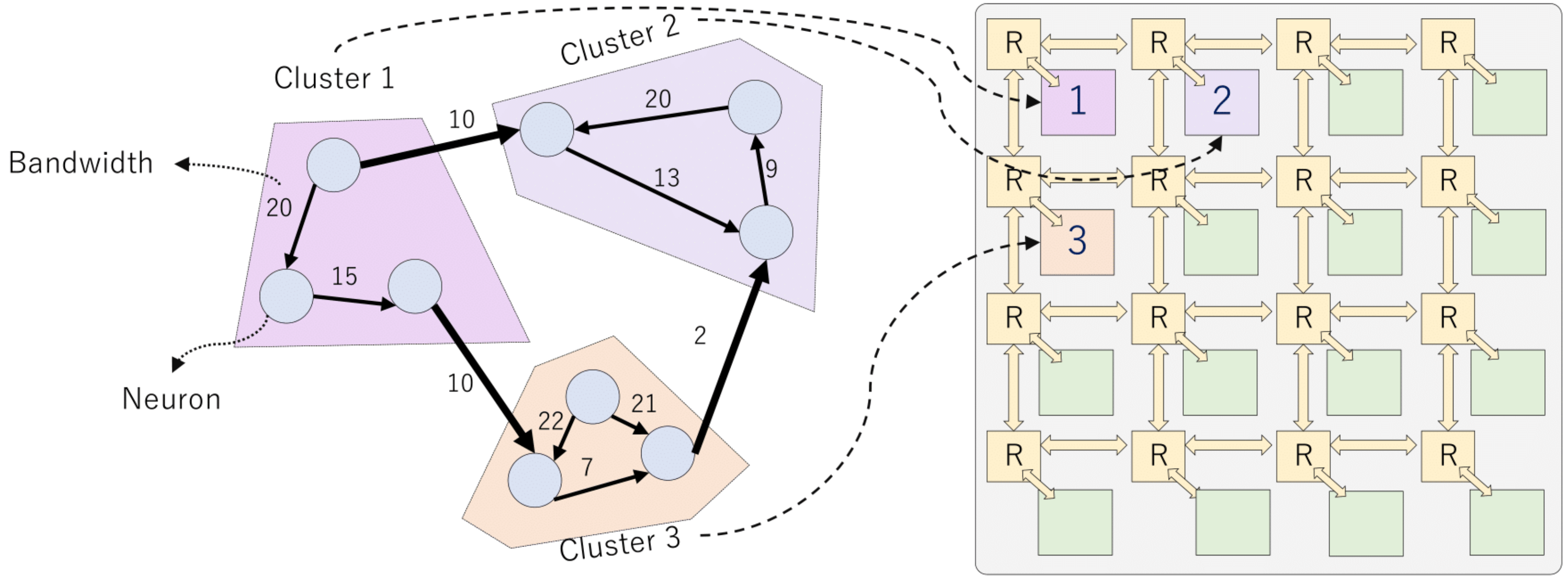


Fig. 5.8: Mapping neuromorphic systems in two phases: partitioning and placing.

Lecture Contents

5. Network Topology
Topology: 2D Mesh

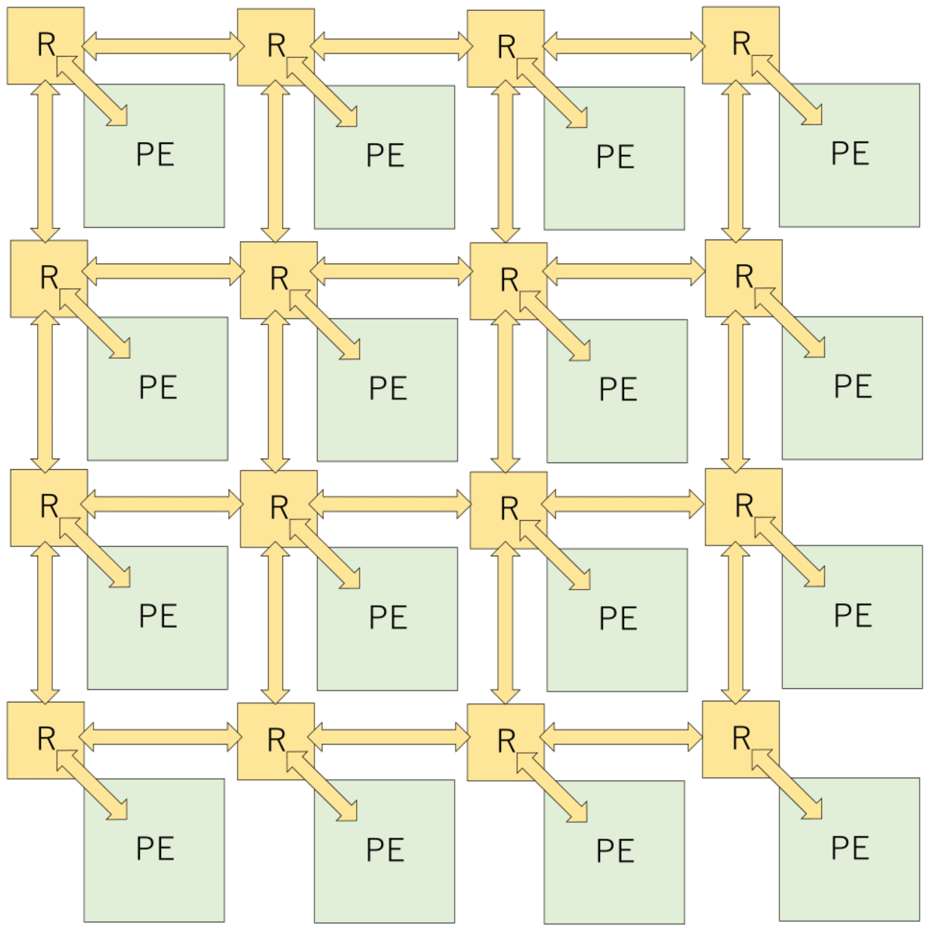


Fig. 5.6: 2D Mesh topology.

- MxN mesh of routers (or switches) interconnecting Processing Element (PE) placed near the router
- Routing can be done via X or Y direction
- Each router (except the one on border) can connect up to 4 neighboring routers and its attached PE.
- In multi-chip system, bordering routers can connect to "neighboring chips"
    - Routing stay unchanged in multi-chip system

5. Network Topology

Topology: 3D Mesh

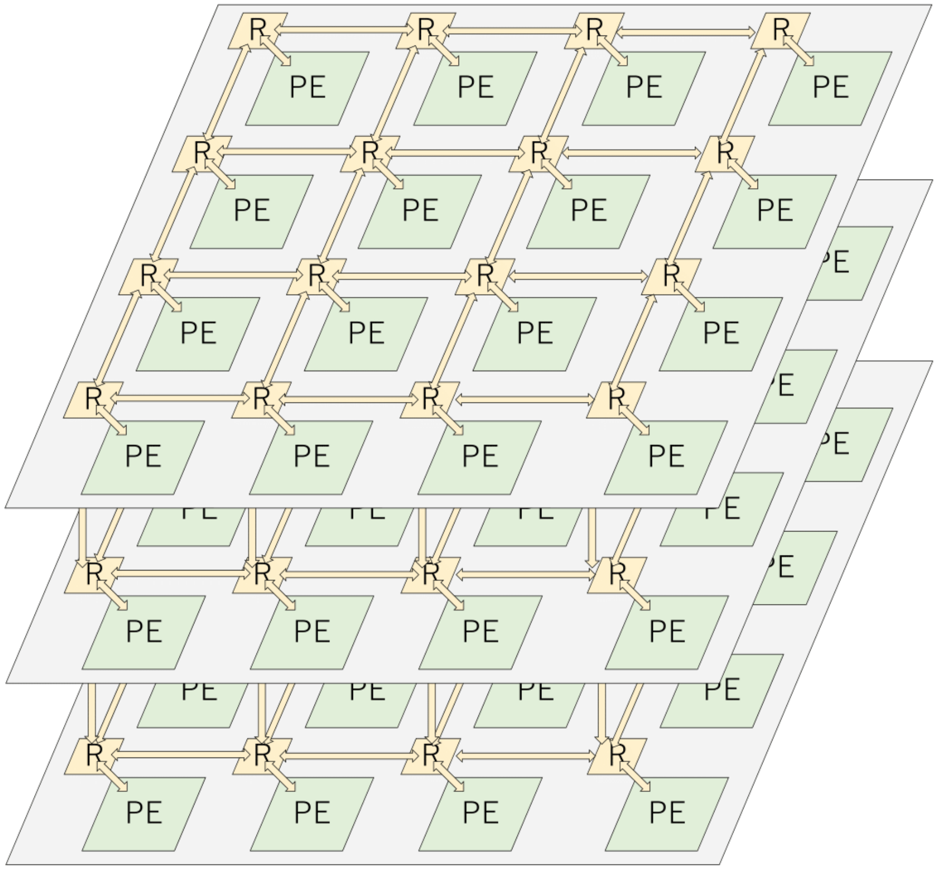
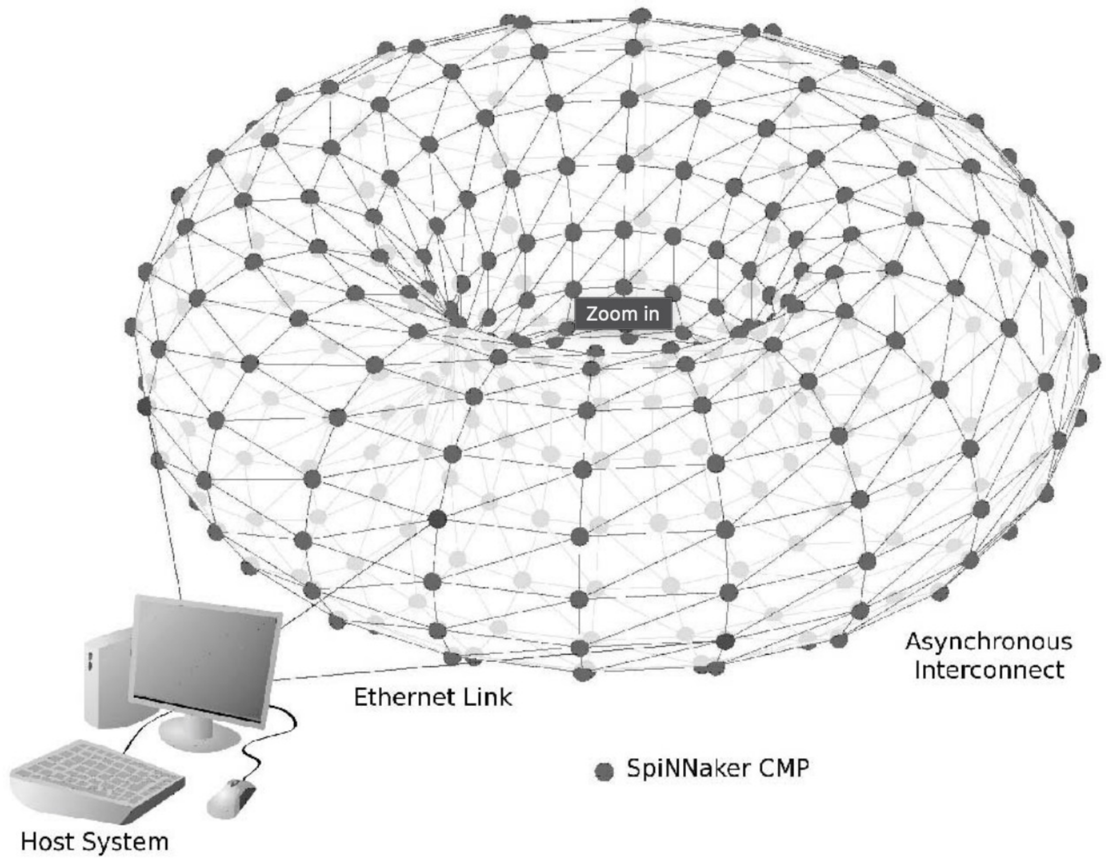


Fig. 5.7: 3D Mesh topology.

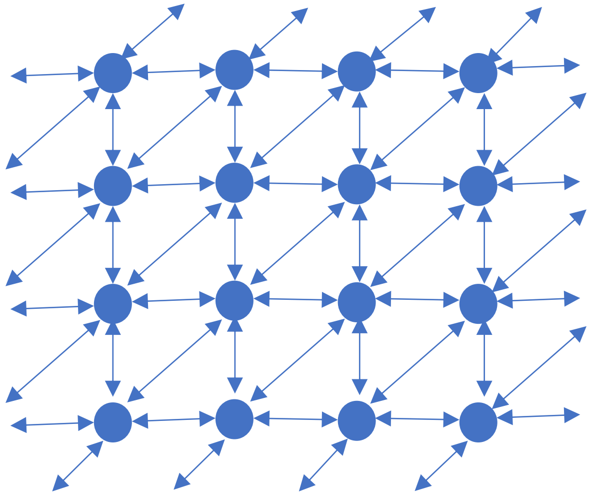
- MxNxL mesh of routers (or switches) interconnecting Processing Element (PE) placed near the router
- Routing can be done via X, Y or Z direction
- Each router (except the one on border) can connect up to 6 neighboring routers and its attached PE.
- In multi-chip system, bordering routers can connect to "neighboring chips"
    - Routing stay unchanged in multi-chip system

5. Network Topology

Topology: Triangular Lattice (SpiNNaker)



- The six communications links are used to connect the nodes in a triangular lattice
- The lattice is then folded onto the surface of a toroid



S. B. Furber et al., "Overview of the SpiNNaker System Architecture," in IEEE Transactions on Computers, vol. 62, no. 12, pp. 2454-2467, Dec. 2013, doi: 10.1109/TC.2012.142.

Lecture Contents

6. Communication Architecture
Overview

- On the communication, there are some parts must be decided:
  - Switching techniques: packet switching, circuit switch or hybrid.
    - Within the switching, there are several models. For instance, with packet switching, there are store-and-forward, virtual-cut-through and wormhole.
  - Packet routing:
    - Deterministic or nondeterministic routing
    - Minimal or non-minimal routing
    - Deadlock/livelock awareness

6. Communication Architecture
Overview

- On the communication, there are some parts must be decided:
  - Flow-control:
    - NACK/ACK: issue signal whether sucesfully receive data.
    - STALL/GO: stop the sender by using stall signal
    - CREDIT: issue the index of the received flit to the sender
  - QoS:
    - Best Effort: do not reserve any resource and the flit arrive without any constraint. Package dropping is possible
    - Guarantee Service: pre-allocating resource for delivering the package.

6. Communication Architecture

Generic Router

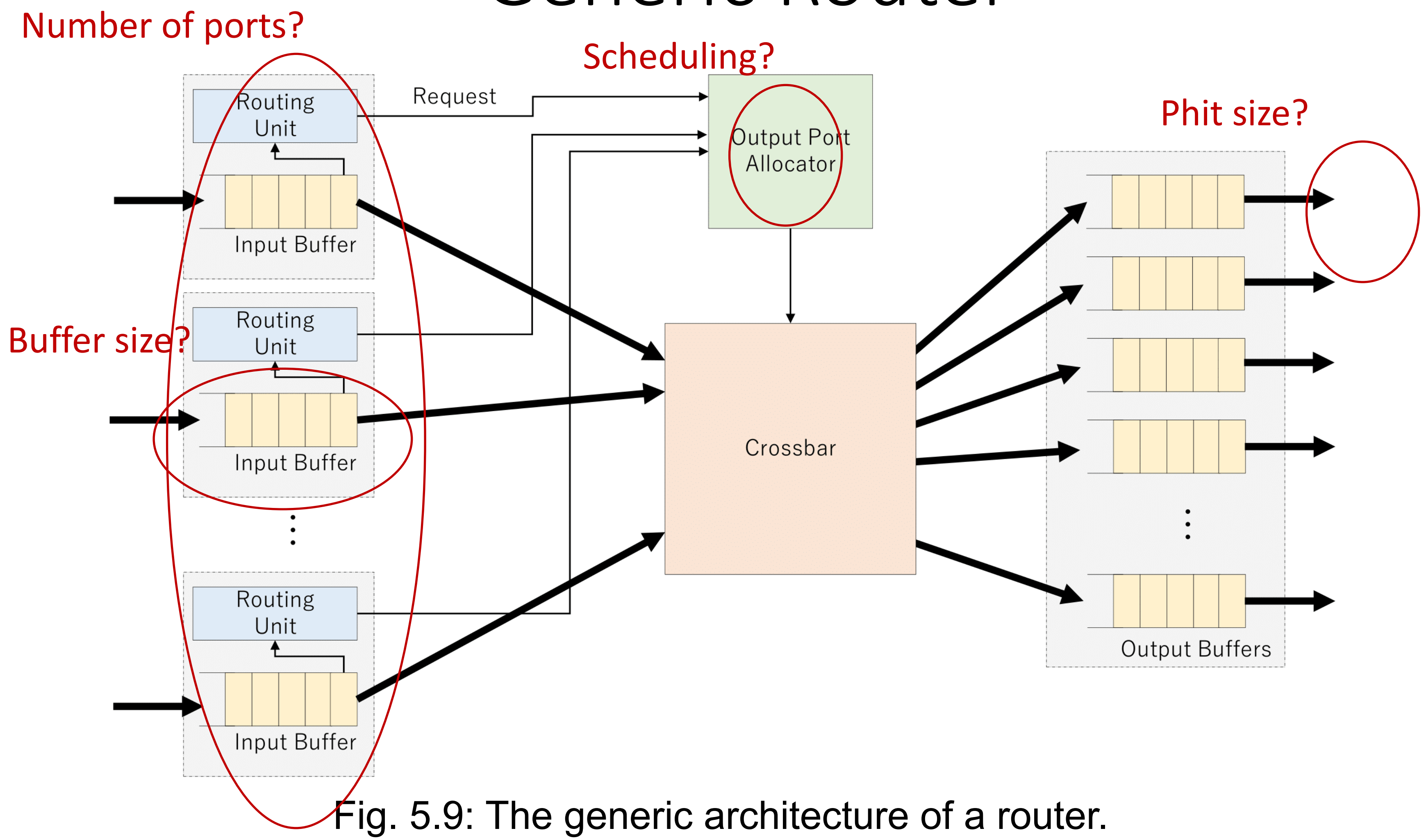


Fig. 5.9: The generic architecture of a router.

Lecture Contents

7. Advanced Design

Router

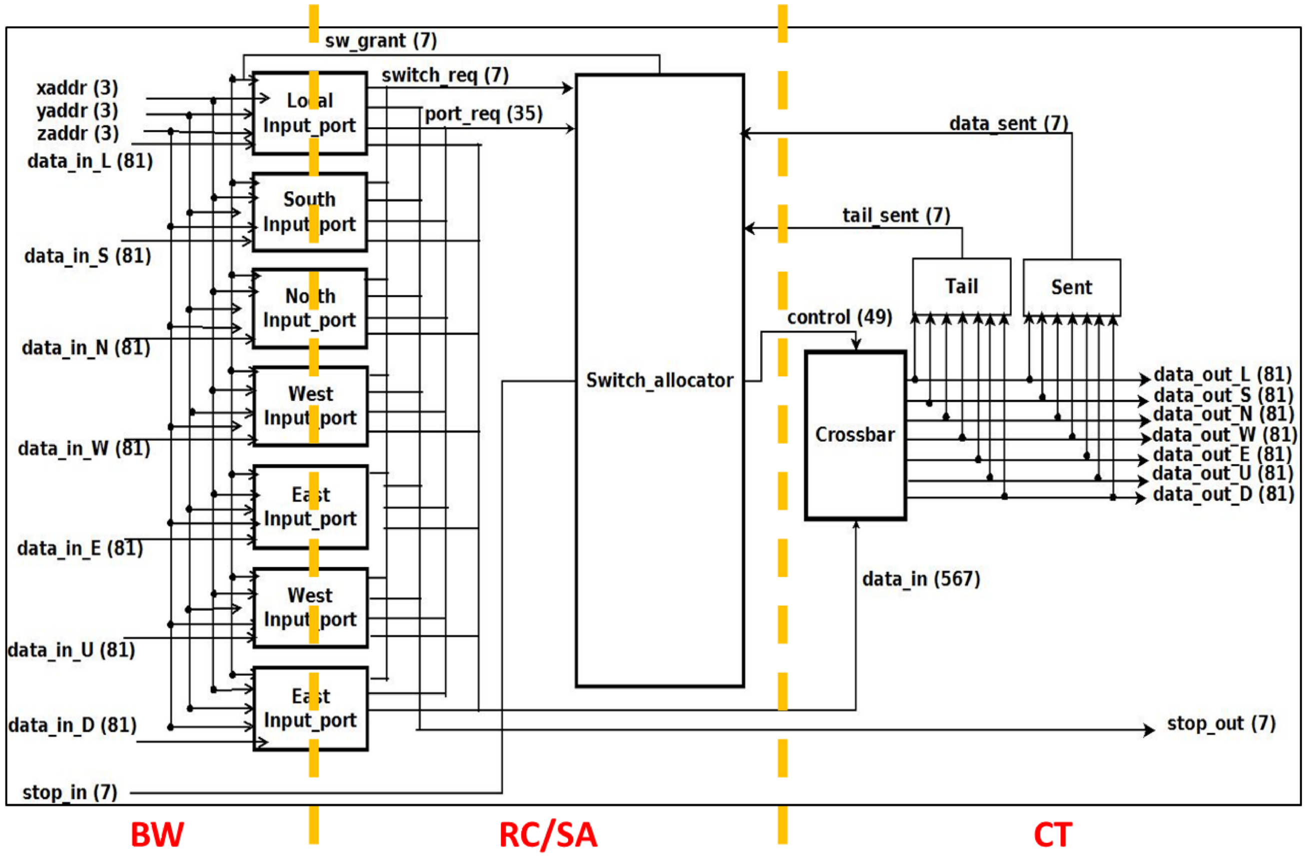


Fig. 5.12: 3D-ONoC pipeline stages: Buffer writing (BW), Routing Calculation and Switch Allocation (RC/SA) and Crossbar Traversal stage (CT).

7. Advanced Design Router

One router has **three** important elements

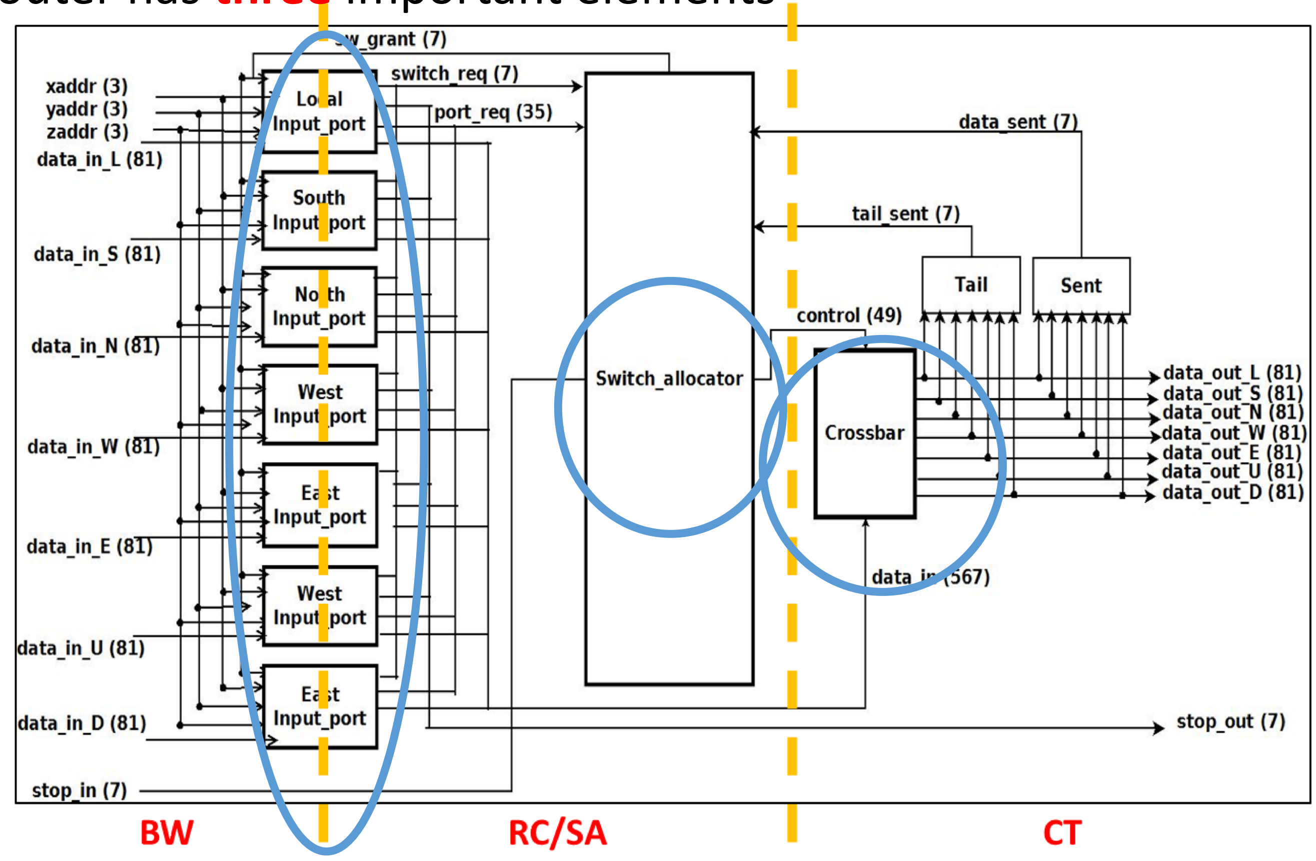


Fig. 5.12: 3D-ONoC pipeline stages: Buffer writing (BW), Routing Calculation and Switch Allocation (RC/SA) and Crossbar Traversal stage (CT).

5. Communication Architecture

Router

One router has **three** important elements

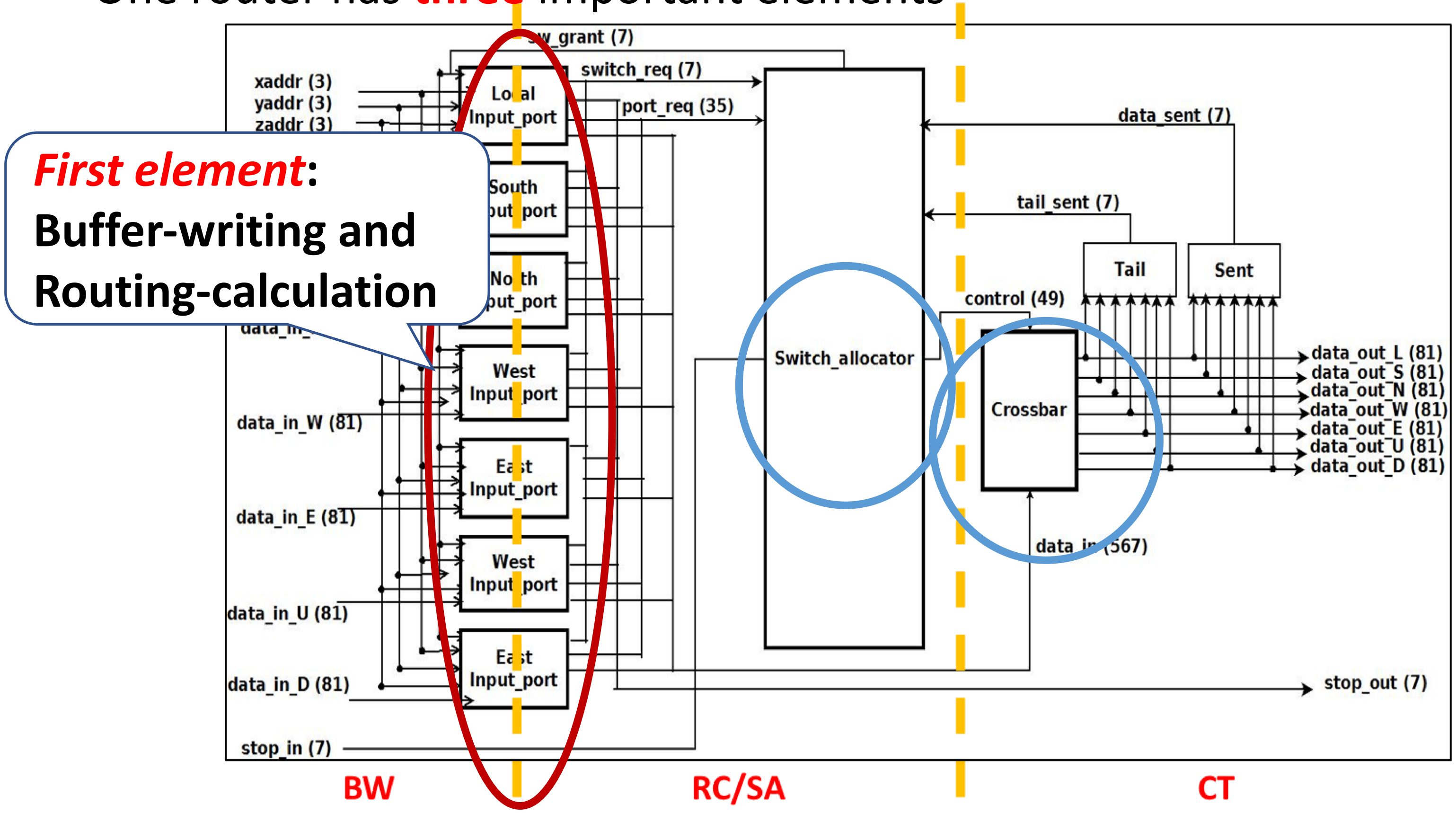
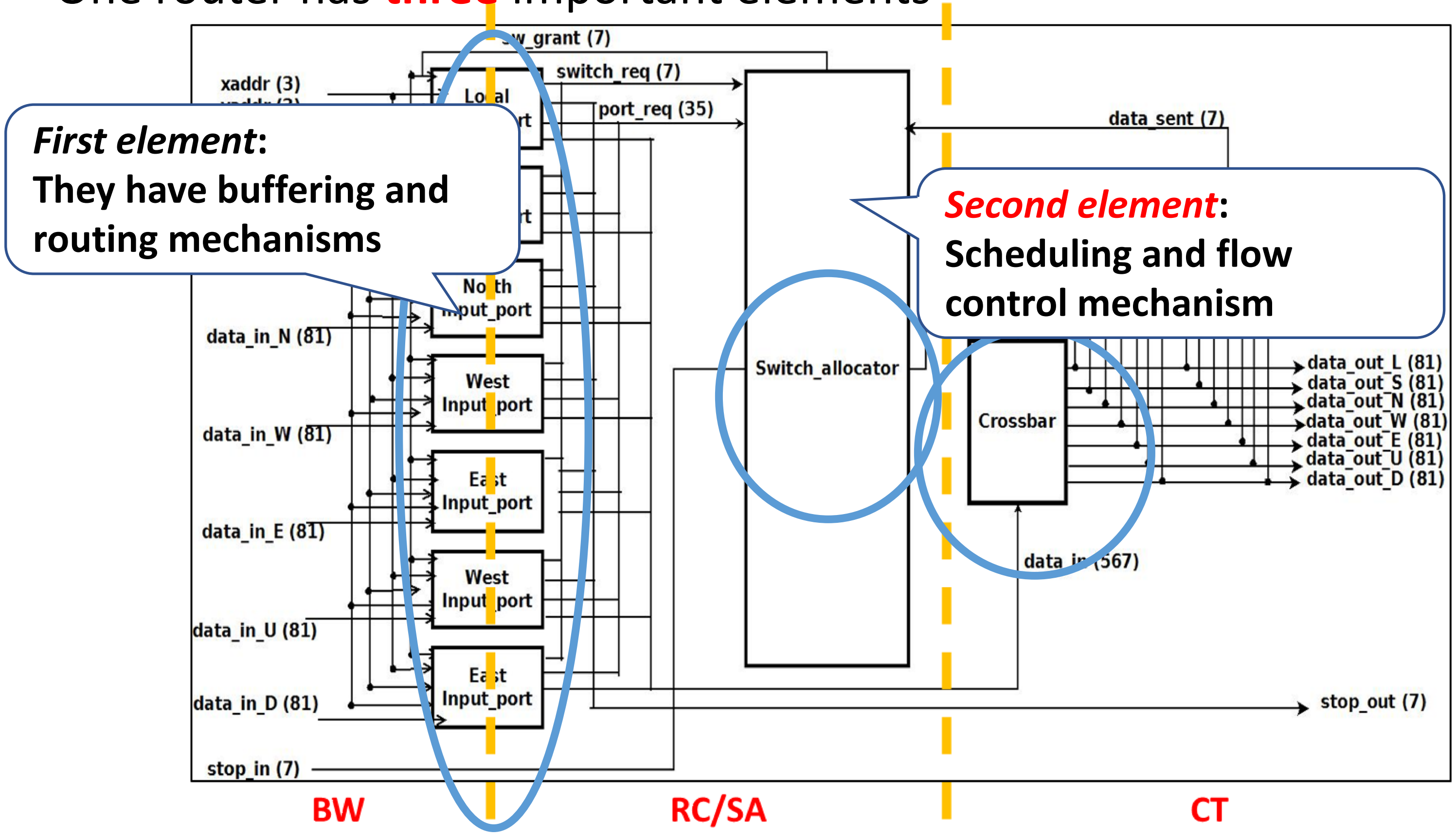


Fig. 5.12: 3D-ONoC pipeline stages: Buffer writing (BW), Routing Calculation and Switch Allocation (RC/SA) and Crossbar Traversal stage (CT).

7. Advanced Design Router

One router has **three** important elements



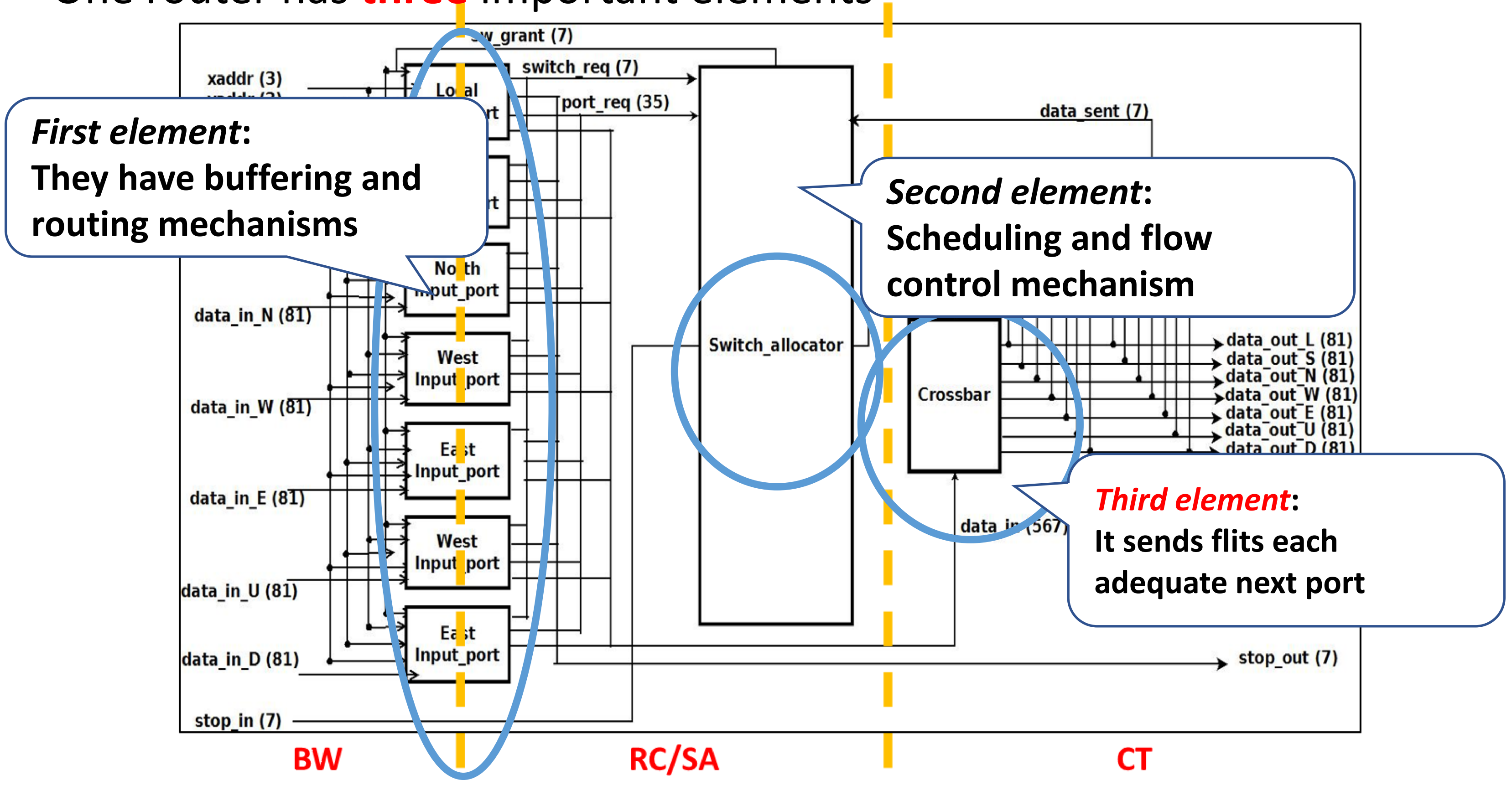
***First element*:**
They have buffering and routing mechanisms

***Second element*:**
Scheduling and flow control mechanism

Fig. 5.12: 3D-ONoC pipeline stages: Buffer writing (BW), Routing Calculation and Switch Allocation (RC/SA) and Crossbar Traversal stage (CT).

7. Advanced Design Router

One router has **three** important elements



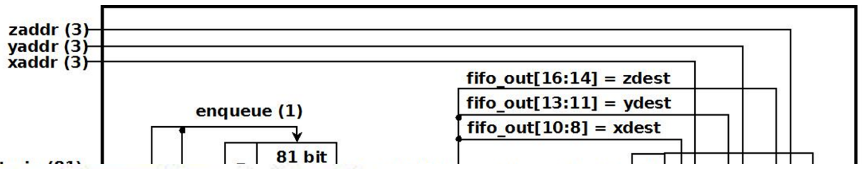
***First element*:**
They have buffering and routing mechanisms

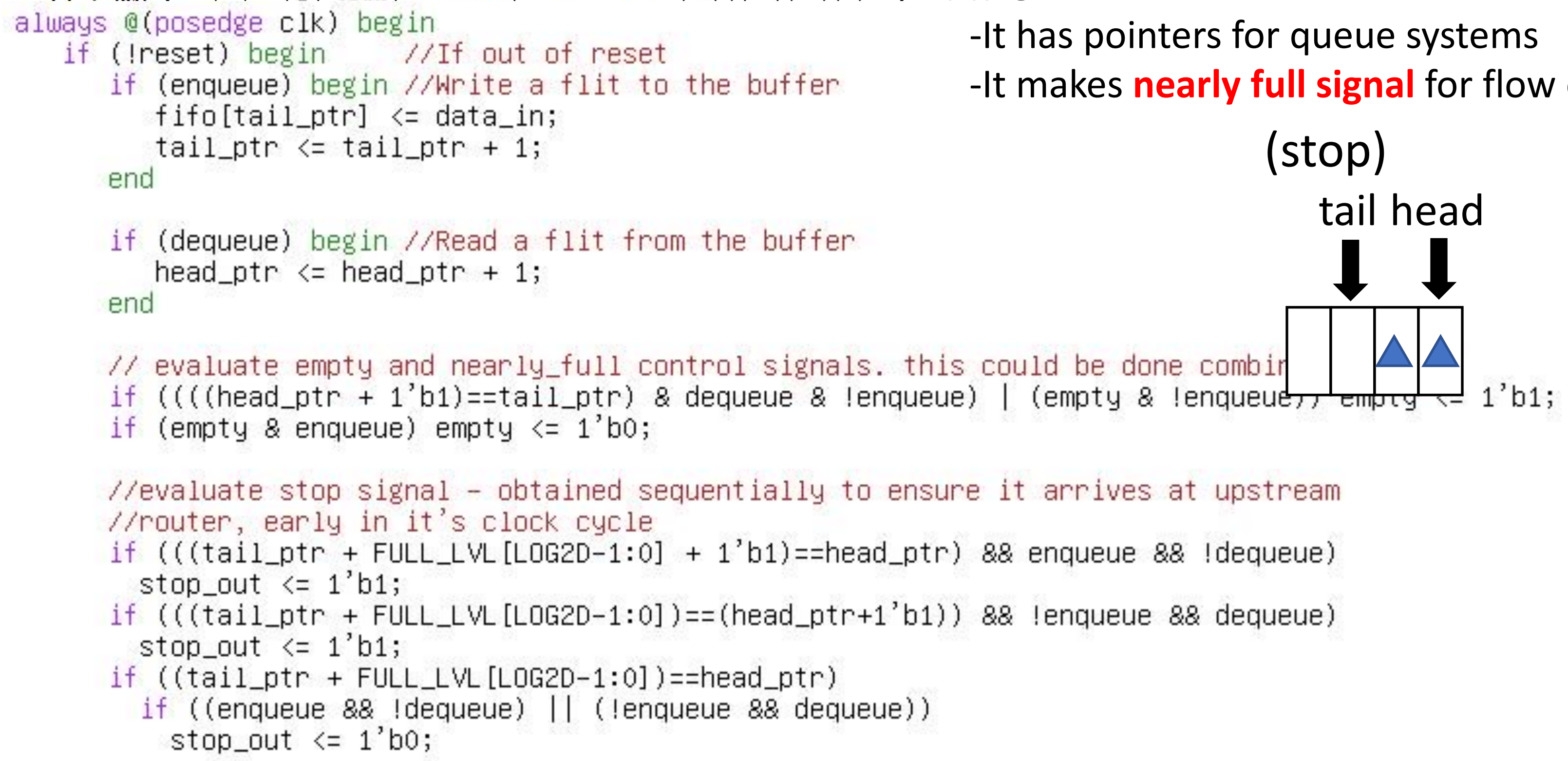
***Second element*:**
Scheduling and flow control mechanism

***Third element*:**
It sends flits each adequate next port

Fig. 5.12: 3D-ONoC pipeline stages: Buffer writing (BW), Routing Calculation and Switch Allocation (RC/SA) and Crossbar Traversal stage (CT).

7. Advanced Design
Input-Port





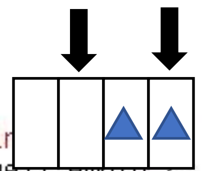
```
always @(posedge clk) begin
  if (!reset) begin     //If out of reset
    if (enqueue) begin //Write a flit to the buffer
      fifo[tail_ptr] <= data_in;
      tail_ptr <= tail_ptr + 1;
    end

    if (dequeue) begin //Read a flit from the buffer
      head_ptr <= head_ptr + 1;
    end

    // evaluate empty and nearly_full control signals. this could be done combir
    if ((((head_ptr + 1'b1)==tail_ptr) & dequeue & !enqueue) | (empty & !enqueue)) empty <= 1'b1;
    if (empty & enqueue) empty <= 1'b0;

    //evaluate stop signal - obtained sequentially to ensure it arrives at upstream
    //router, early in it's clock cycle
    if (((tail_ptr + FULL_LVL[LOG2D-1:0] + 1'b1)==head_ptr) && enqueue && !dequeue)
     stop_out <= 1'b1;
    if (((tail_ptr + FULL_LVL[LOG2D-1:0])==(head_ptr+1'b1)) && !enqueue && dequeue)
     stop_out <= 1'b1;
    if ((tail_ptr + FULL_LVL[LOG2D-1:0])==head_ptr)
      if ((enqueue && !dequeue) || (!enqueue && dequeue))
       stop_out <= 1'b0;
```

*It has **FIFO** and routing modules*

- FIFO
  - -It has pointers for queue systems
  - -It makes **nearly full signal** for flow control

(stop)

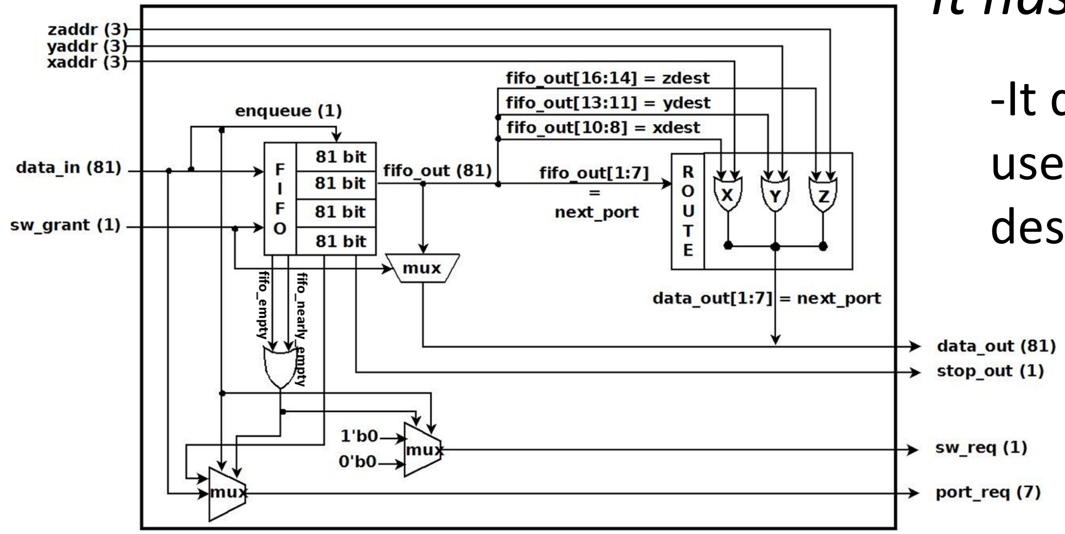
tail head



7. Advanced Design
Input-Port

*It has FIFO and **routing** modules*

-It decides transaction direction to use the current address and destination address



```
//assign next addresses
if (nextport == `EAST) next_xaddr = xaddr + 1'b1;
  else if (nextport == `WEST) next_xaddr = xaddr - 1'b1;
    else next_xaddr = xaddr;

if (nextport == `NORTH) next_yaddr = yaddr + 1'b1;
  else if (nextport == `SOUTH) next_yaddr = yaddr - 1'b1;
     else next_yaddr = yaddr;
```

```
//evaluate next port
if (next_xaddr == xdest) begin
   if (next_yaddr == ydest) route = `SELF;
     else if(next_yaddr < ydest) route = `NORTH;
       else route = `SOUTH;
end else begin
   if (next_xaddr < xdest) route = `EAST;
     else route = `WEST;
end
```

First, next router's address is computed

Next-port is decided by using next address

7. Advanced Design

Arbiter

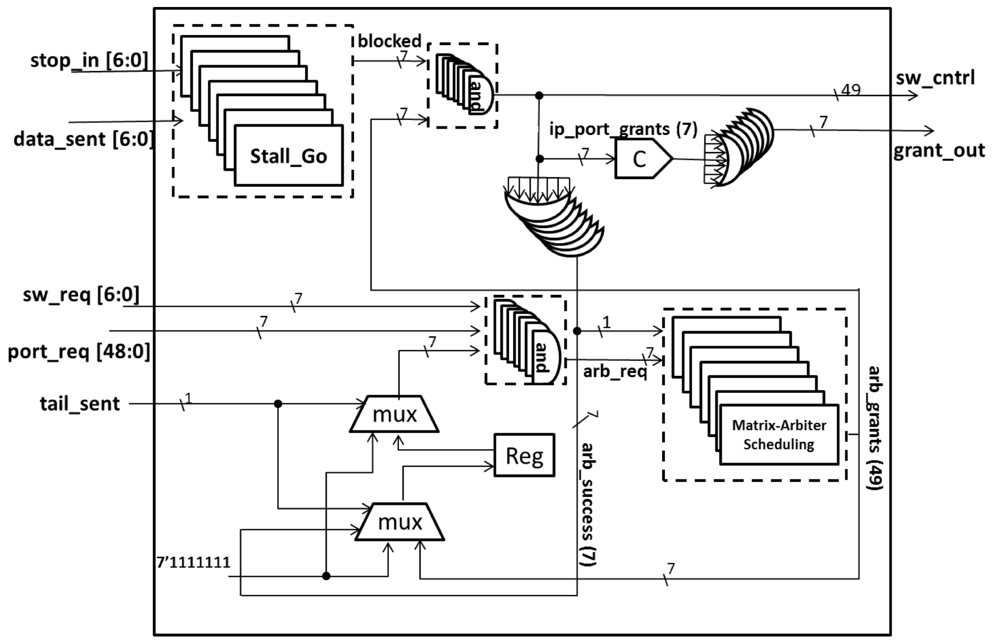


Fig. 5.14: Switch allocator architecture.

- Output bandwidth is limited for one flit data size per output port
- Therefore, scheduling is needed
- Matrix-Arbiter Scheduling will decide which input port can send flit to output port

7. Advanced Design

Arbiter: Matrix arbiter

$$\begin{pmatrix} X & P_{12} & P_{13} & P_{14} \\ P_{21} & X & P_{23} & P_{24} \\ P_{31} & P_{32} & X & P_{34} \\ P_{41} & P_{42} & P_{43} & X \end{pmatrix}$$

When the priority i > j, P(i,j) becomes 1 and P(j, i) become 0

$$\begin{array}{r}\text{highest}\\ \\ \\ \\ \end{array}\begin{pmatrix} X & 1 & 1 & 1 \\ \hline 0 & X & 0 & 0 \\ 0 & 1 & X & 1 \\ \hline 0 & 1 & 0 & X \end{pmatrix} \begin{array}{l} \\ \\ \text{highest} \\ \\ \end{array} \Rightarrow \begin{pmatrix} X & 0 & 0 & 0 \\ \hline 1 & X & 0 & 0 \\ 1 & 1 & X & 1 \\ \hline 1 & 1 & 0 & X \end{pmatrix}$$

(a) (b)

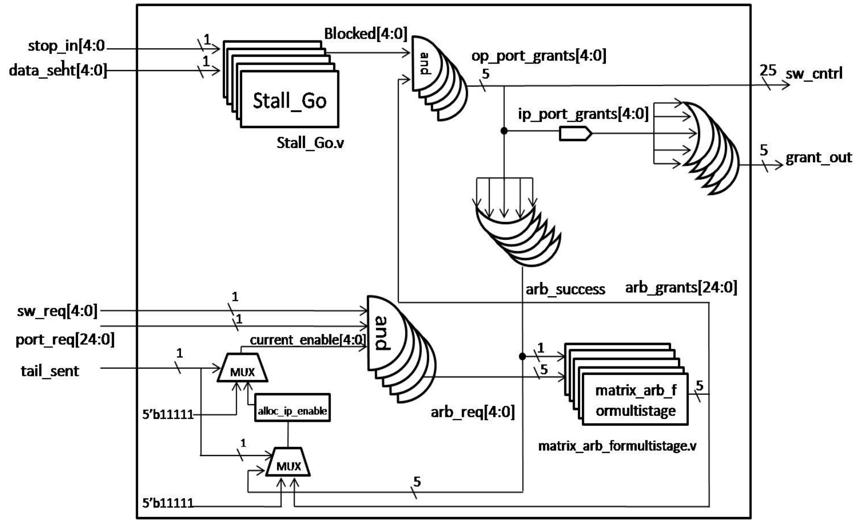
For instance)

- **Request 1** has the highest priority
- Second highest is **request 3**
- Third is **request 4**
- The lowest is request 2

Request 1 can be granted a resource

The priority will be change to the lowest

7. Advanced Design
Arbiter: Matrix arbiter



```
//
// generate grants
//
generate
for (i=0; i<SIZE; i=i+1) begin:ol1
   // generate grant i
   for (j=0; j<SIZE; j=j+1) begin:il1
  if (j==i)
    // request i wins if requesting and....
    assign pri[i][j]=request[i];
  else
    // ....no other request with higher priority
    if (j>i)
      // j beats i
      assign pri[i][j]=!(request[j]&&state[j*SIZE+i]);
    else
      // !(i beats j)
      assign pri[i][j]=!(request[j]&&!state[i*SIZE+j]);
   end

   assign grant[i]=&pri[i];
end
endgenerate
```

```
// update state
//
generate
for (i=0; i<SIZE; i=i+1) begin:ol2
   for (j=0; j<SIZE; j=j+1) begin:il2
  assign new_state[j*SIZE+i]=(success&&((state[j*SIZE+i]&&!grant[j])
                          ||(grant[i])))||(!success&&state[j*SIZE+i]);
   end
end
endgenerate
```

- Right code indicates comparison of priority between current transmitting input port and other routers which send request to arbiter

7. Advanced Design
Arbiter: Stall and Go

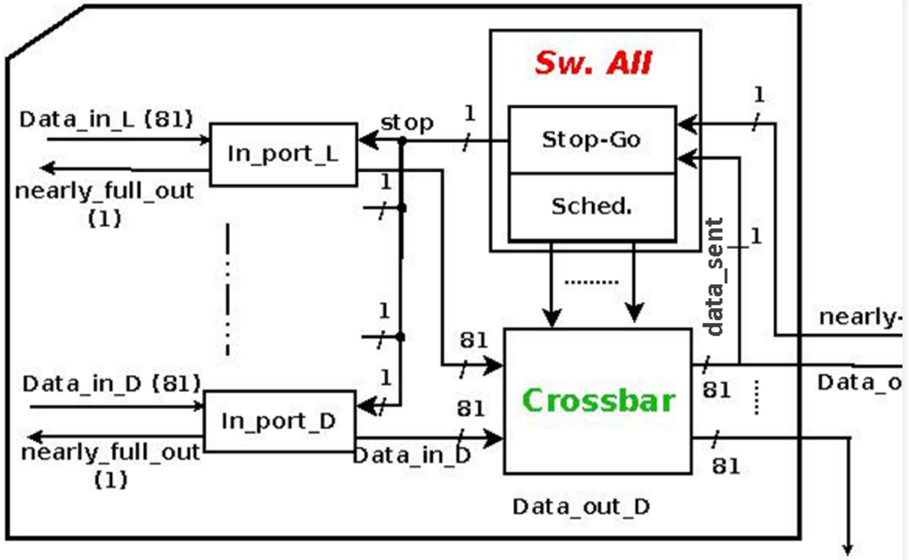


Fig. 5.15: Stall-Go flow c

```
module stall_go(clk, reset, data_sent,
                stop_in, blocked);

  input clk, reset;
  input data_sent, stop_in;

  output blocked;

  reg [1:0] state;

  always @(posedge clk) begin
     if (!reset) begin

        if ((state=='GO) && stop_in && data_sent)
          state <= 'SENT1;

        if (state=='SENT1) begin
           if (stop_in && !data_sent)
             state <= 'GO;
           if (!stop_in && data_sent)
             state <= 'STOP;
        end

        if ((state=='STOP) && stop_in)
          state <= 'GO;

     end else
       state <= 'GO;
  end
```

7. Advanced Design
Arbiter: Stall and Go (Cnt.)

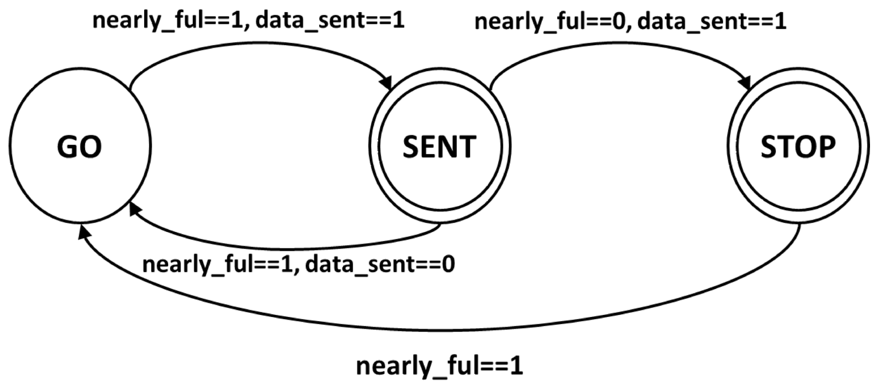
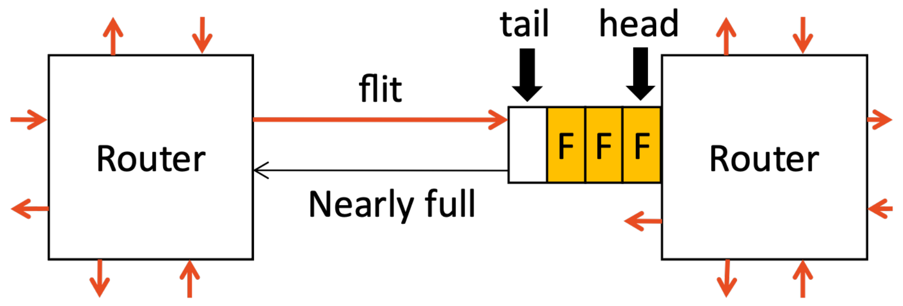

nearly_ful==1, data_sent==1
nearly_ful==0, data_sent==1
GO
SENT
STOP
nearly_ful==1, data_sent==0
nearly_ful==1

Fig. 5.16: Stall-Go flow control Finite State Machine.

Nearly full means…
it has one flit space



stall_go treats flow control

- It has state machine to decide when it issues stall signal

Lecture Contents

8. Conclusion

- In this lecture, we have reviewed the communication network for neuromorphic computing.
  - Address Event Representation vs Spike Vector
  - Overview of the SOTA architectures
  - Network-on-Chip design and principle
- We also discuss about the detailed design of the OASIS Network on Chip

9. Exercises

OASIS NoC is available at: https://github.com/benabdallah-dang-lab/MigSpike with 2D NoC (9 cores, 3x3)

1. Reconfigure OASIS NoC to 3x3x3
2. Generate Random Input/Output configuration
3. Evaluate the overall latency of the system
4. Indicate which is the longest routing path
5. Indicate the most congestion path