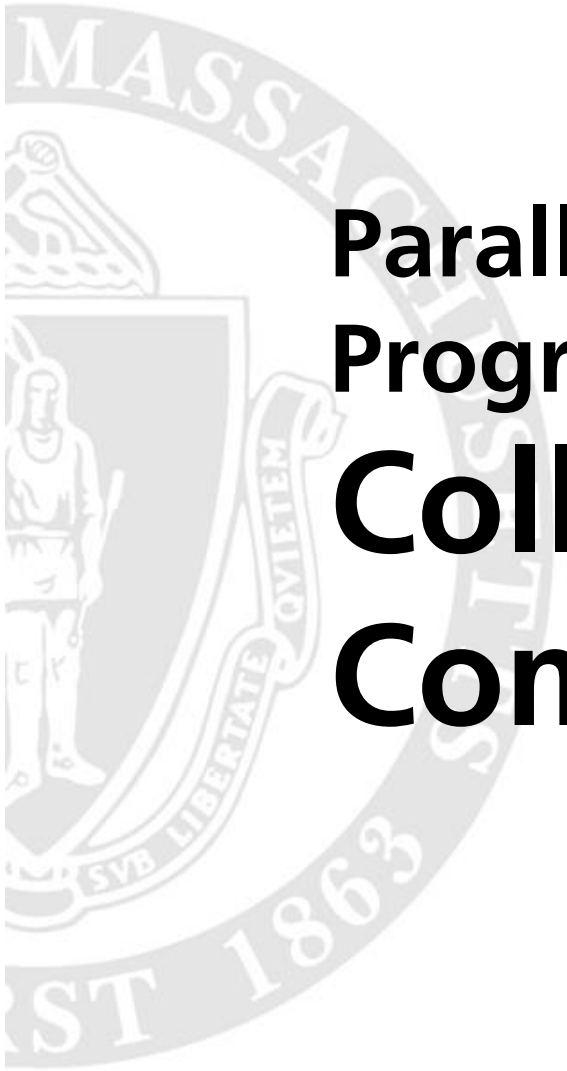


Parallel & Concurrent Programming: Collective Communication

Emery Berger

CMPSCI 691W

Spring 2006

Outline

- Last time:
  - Distributed parallel programming via **message-passing**
  - **MPI** – **library** approach
- Today:
  - Moving beyond point-to-point: **collective communication**

some slides from van de Geijn et al.

Collective Communication

- Instead of single target, **collective communication** operates on entire **communicator**
  - e.g., MPI_COMM_WORLD
- Advantages over point-to-point:
  - **Higher-level**
  - Typically **far more efficient**
    - e.g., O(n) $\Rightarrow$ O(log n)
    - **Pipelining**

Simple Analytical Model

- To send *n* bytes:
  - α = startup latency
  - β = per-byte cost
  - ⇒ $\alpha + n\beta$
- Naive broadcast (*p* processors):
  - $\alpha(p-1) + (p-1)n\beta$
- **Minimum spanning tree**:
  - $\alpha(\log p + p - 1) + 2(p-1)/p * n\beta$
  - Note: does not model **contention**
    - In practice, can avoid it

MPI Collective Ops

- MPI has rich set of collective communication operations (& duals):
  - **Synchronization**
    - Barrier
  - **Communication**
    - Broadcast
    - Scatter, Gather
    - Reduce, Scan
  - **Communicator management**

Barrier Synchronization

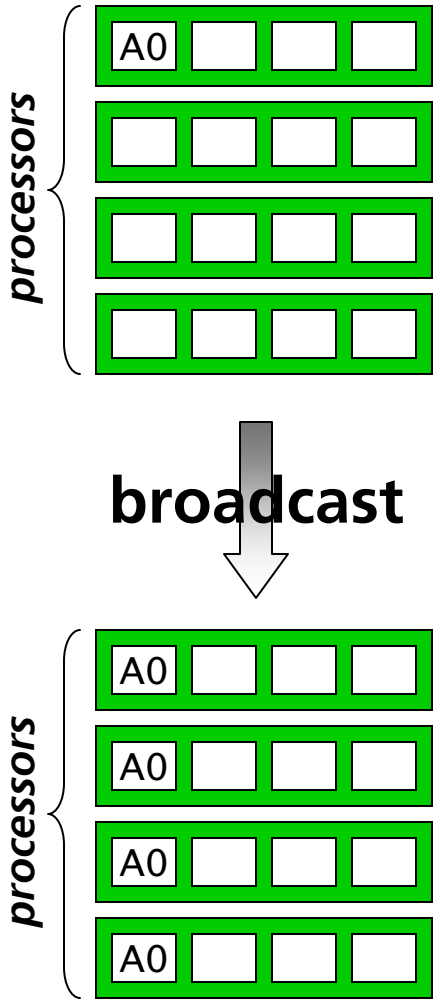
- **MPI_Barrier**:
  progress continues past barrier only after all processes have arrived

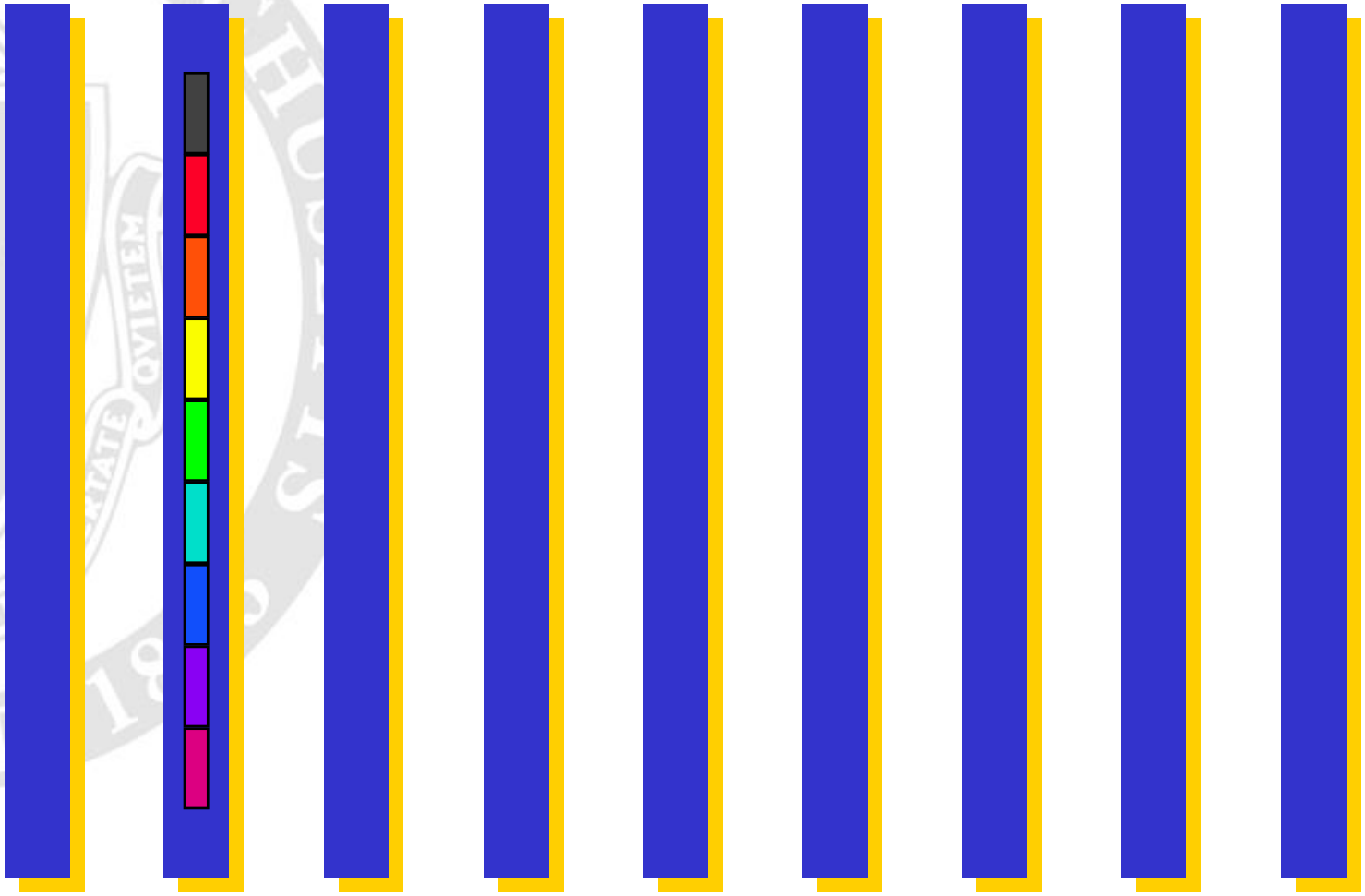
```
// perform computation

...

MPI_Barrier (MPI_COMM_WORLD);
// all done here
```

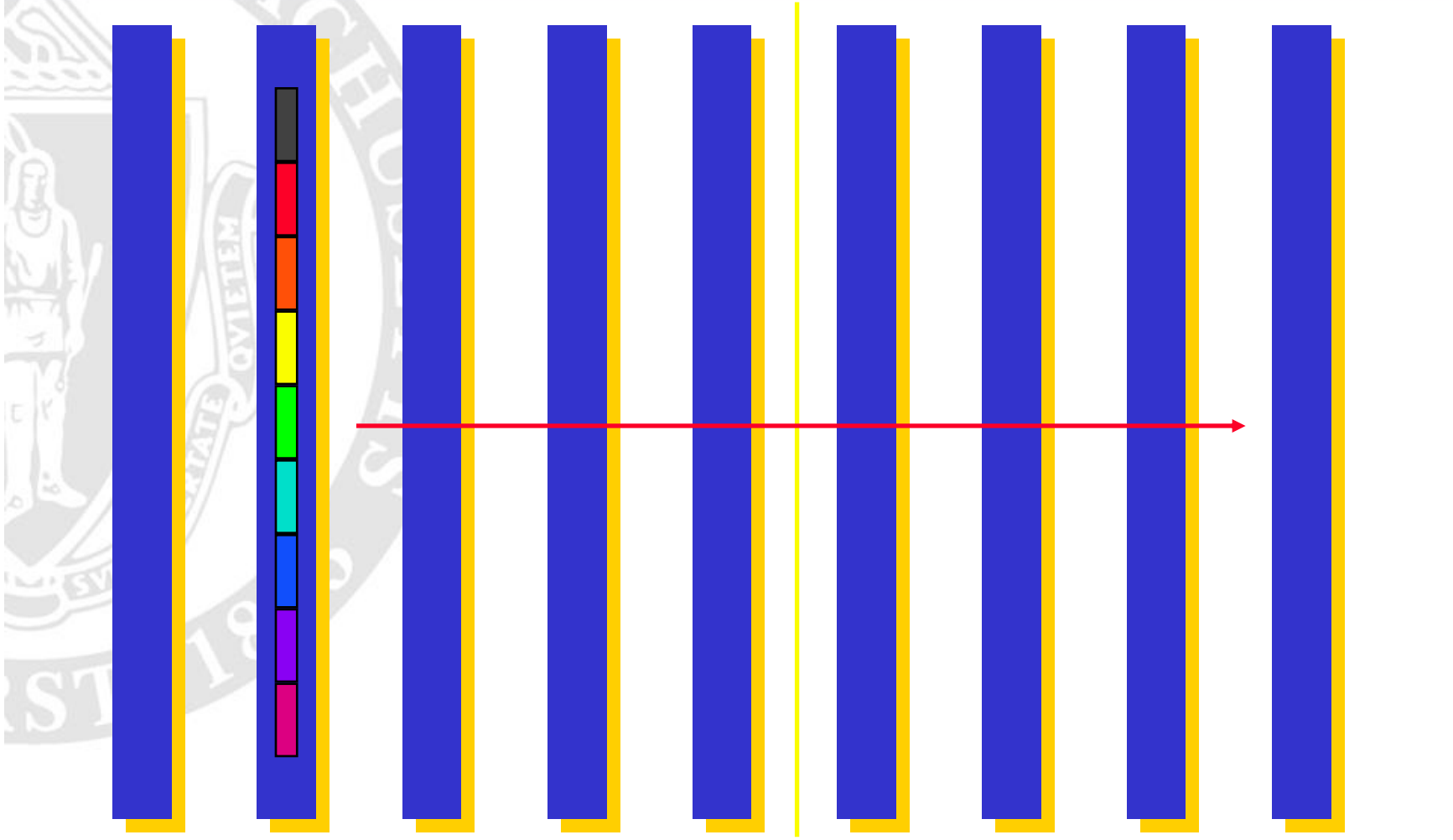
```
// perform computation (faster)



MPI_Barrier (MPI_COMM_WORLD);
// all done here
```

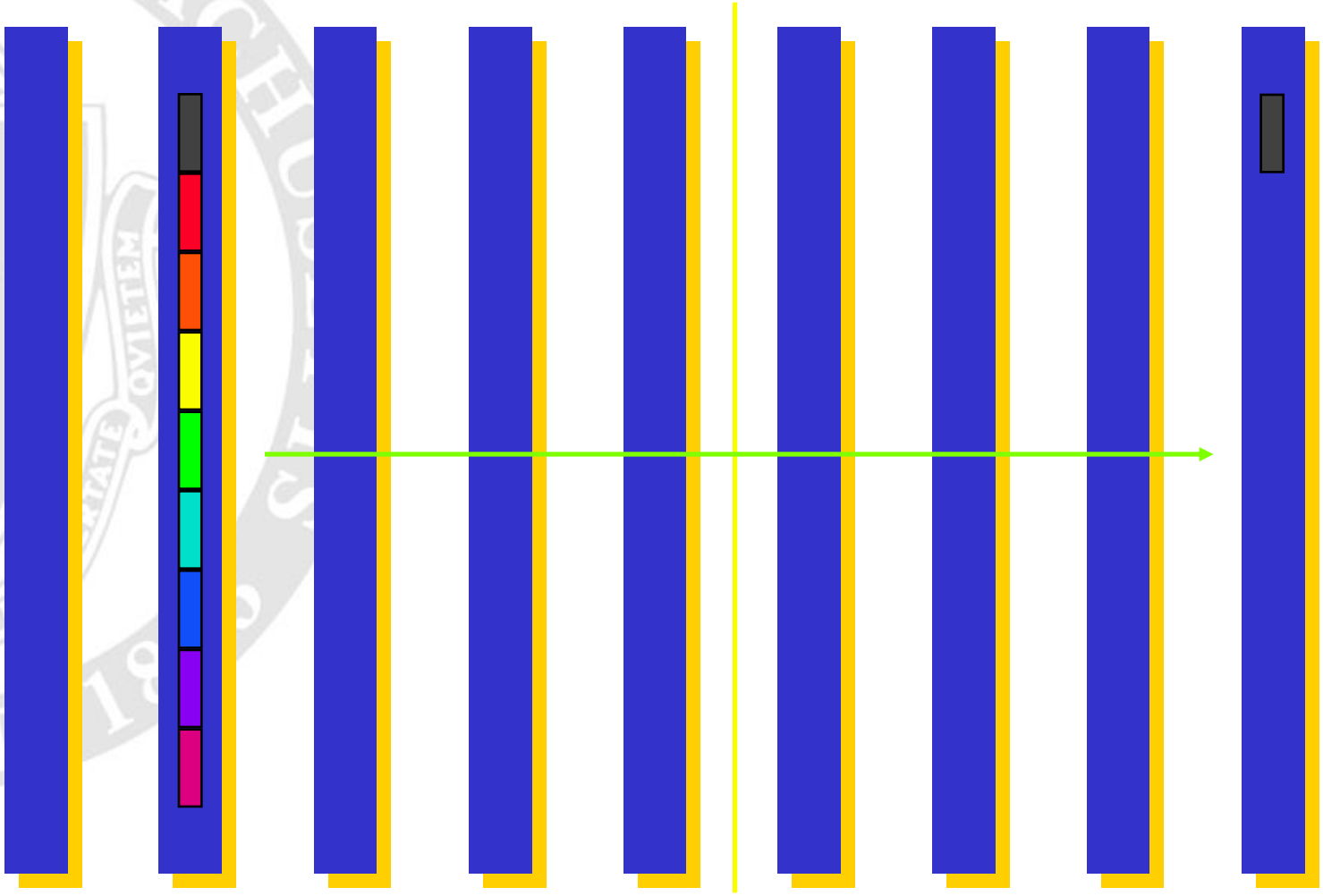
From root: Broadcast

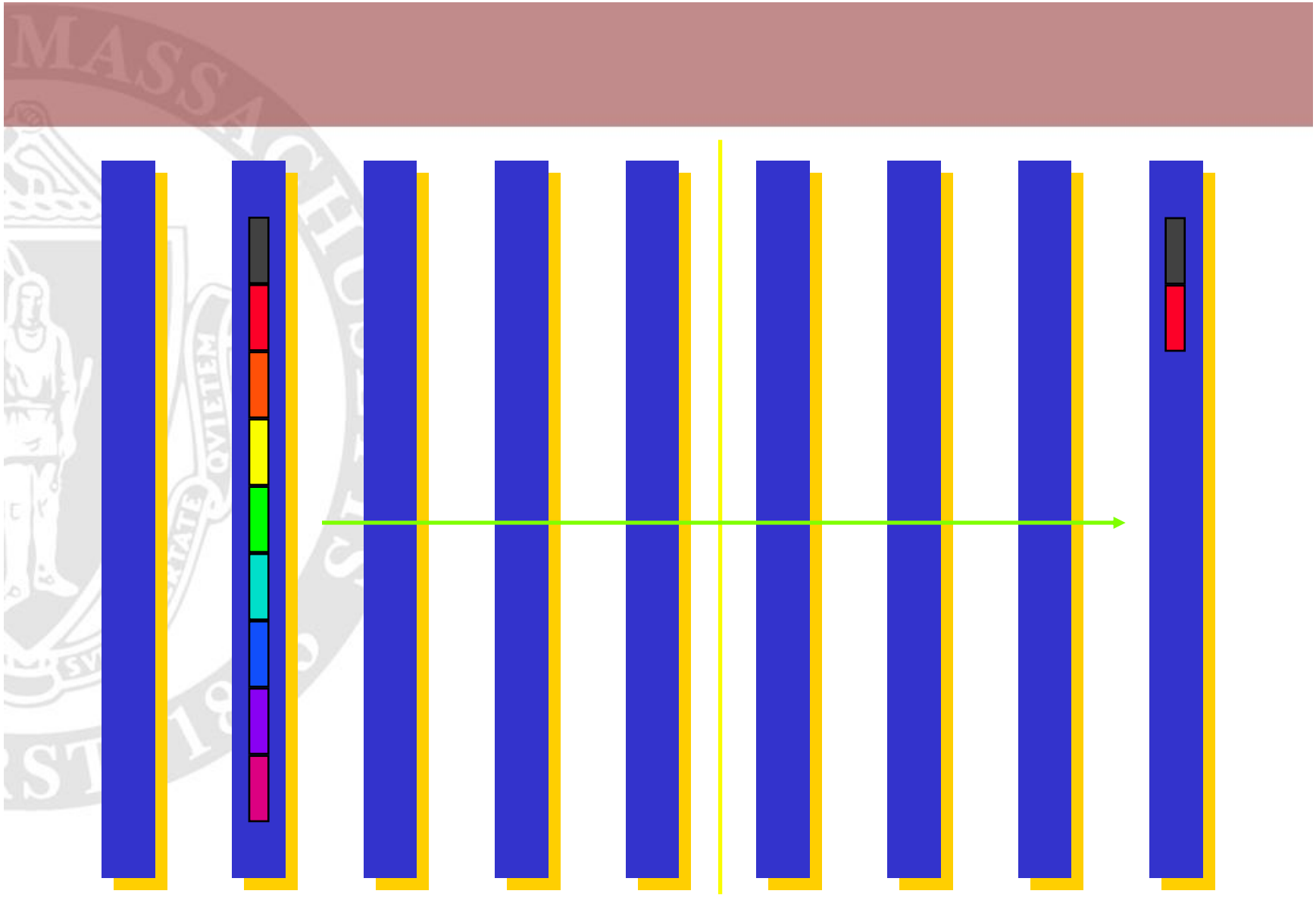
- **MPI_Bcast**: broadcast single datum across all processors

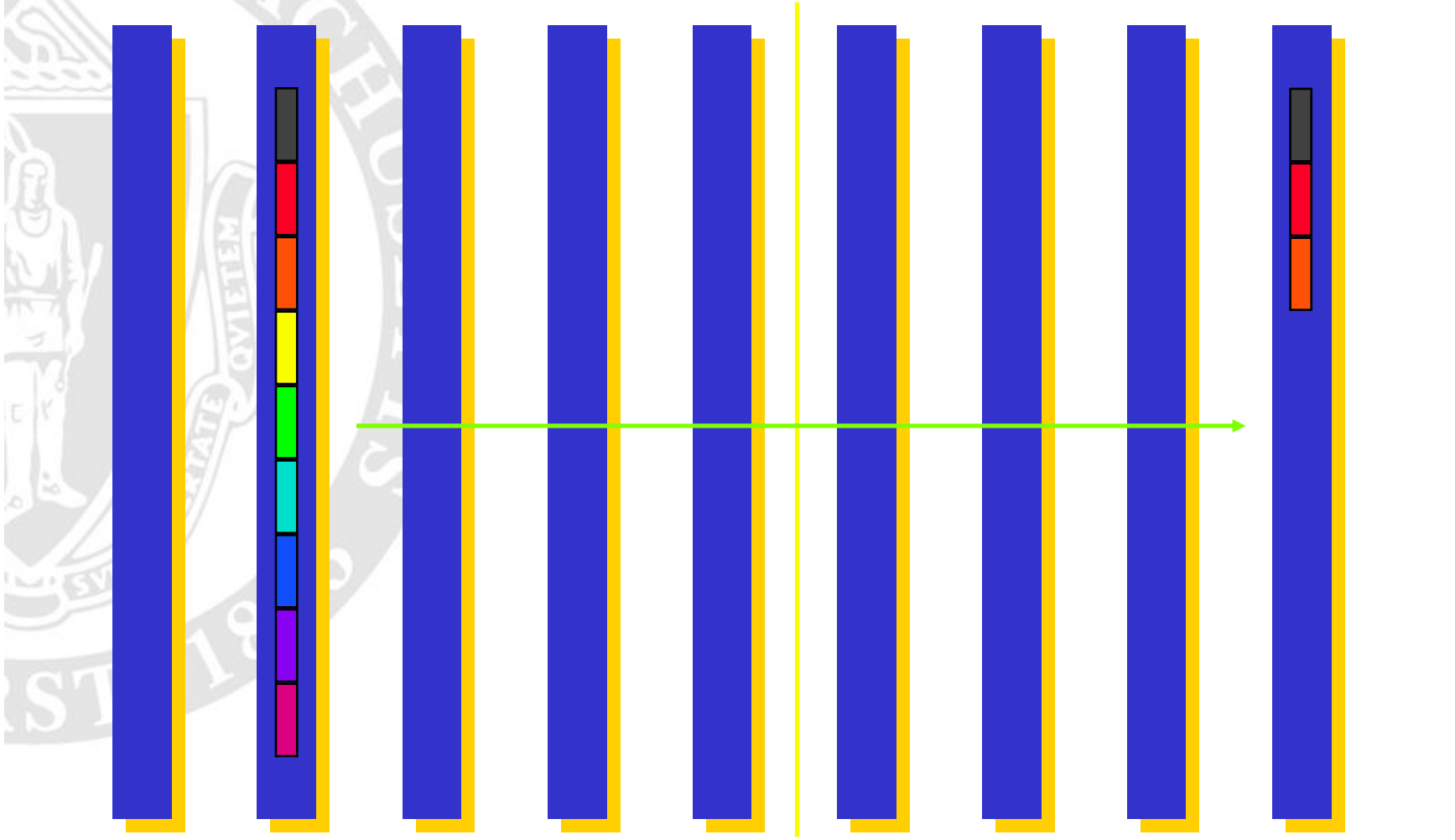


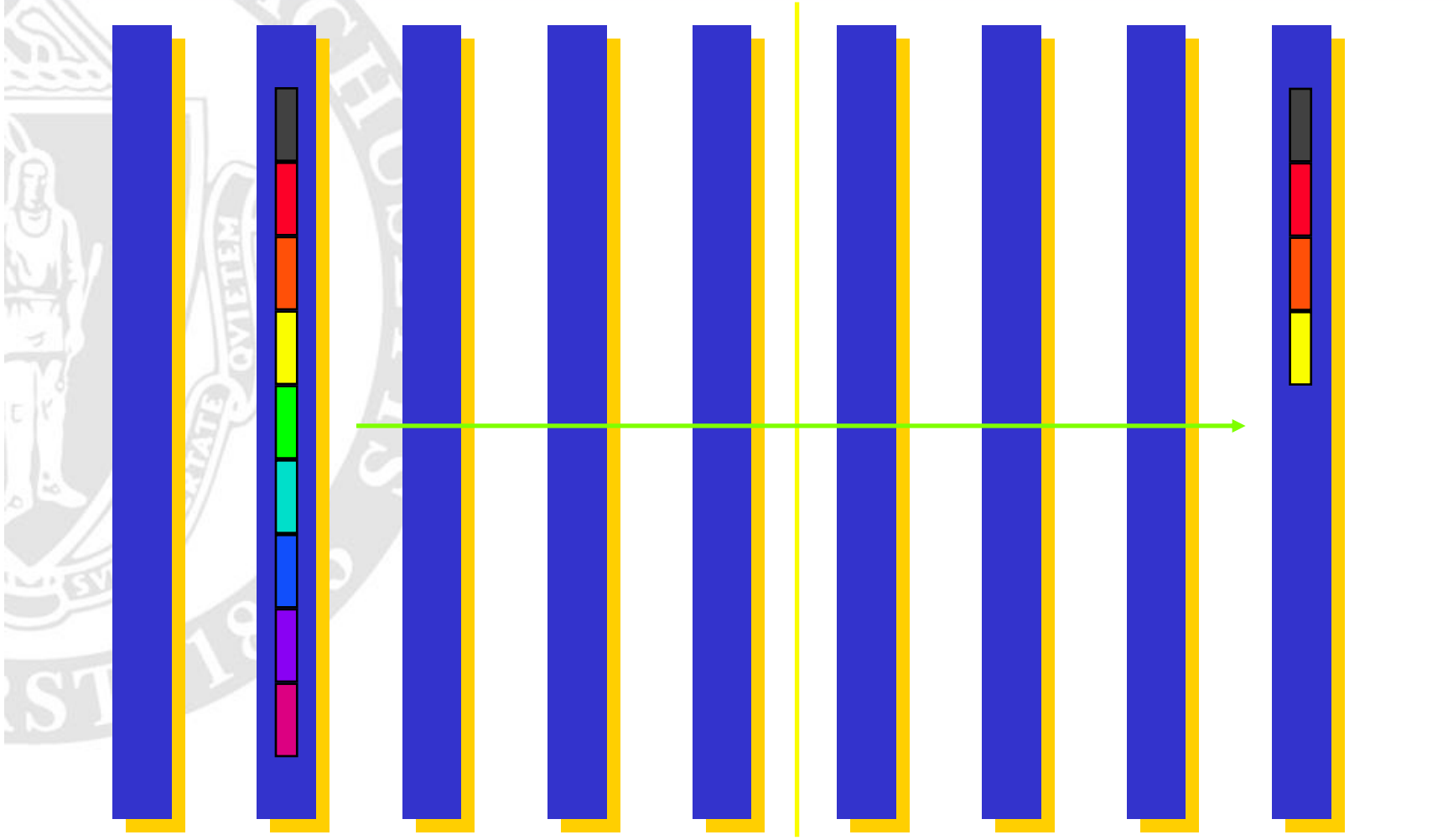


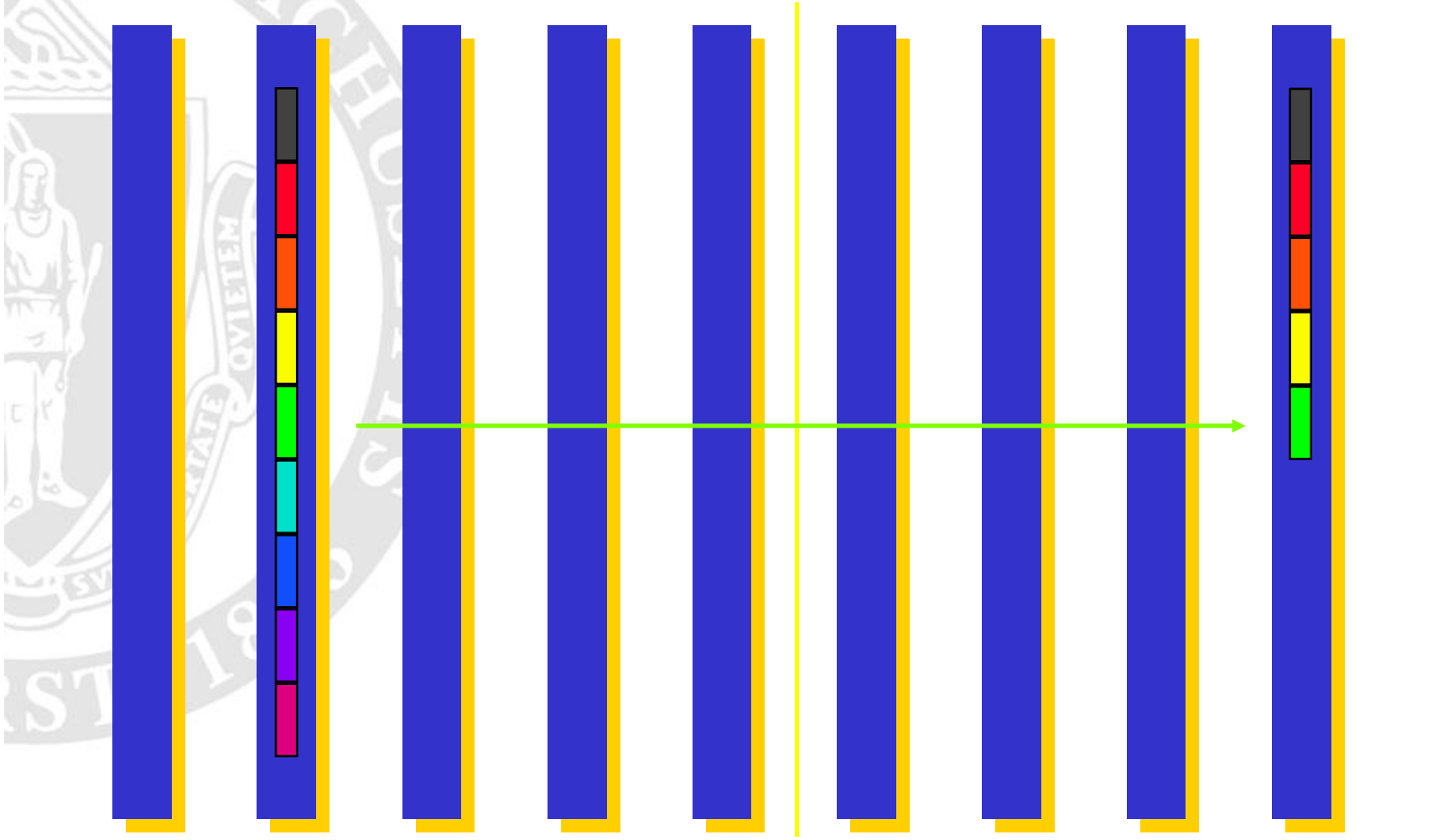


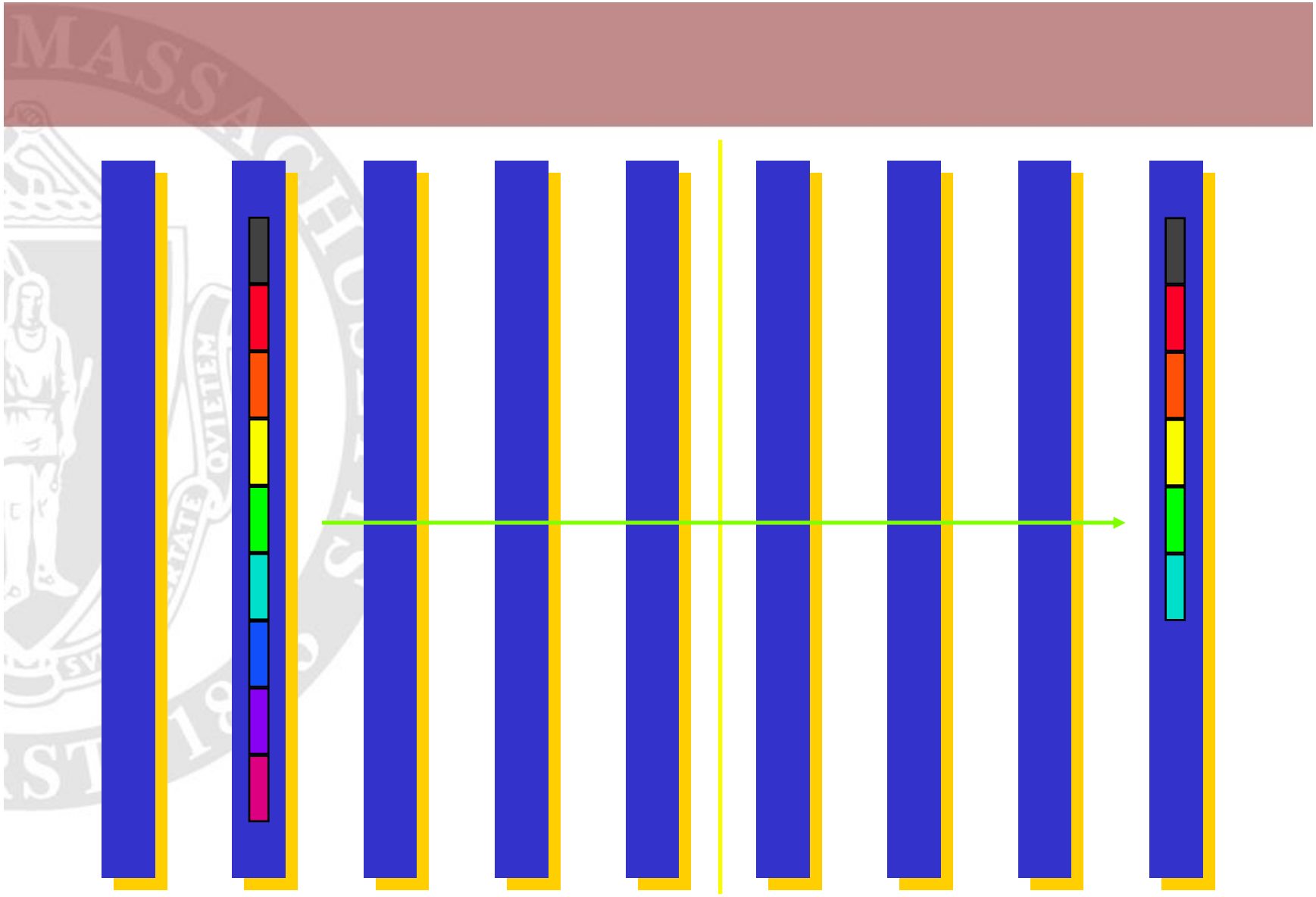


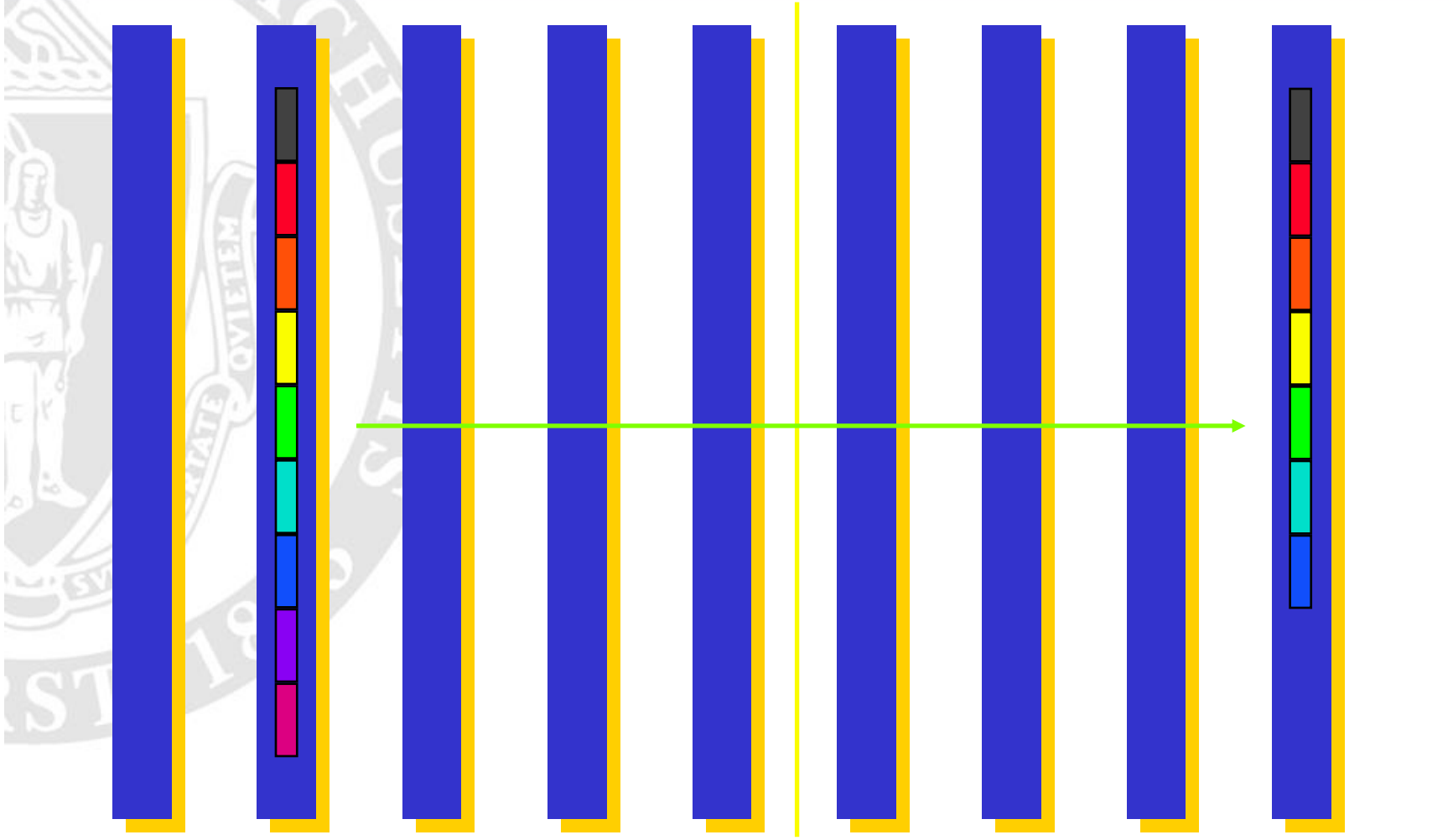


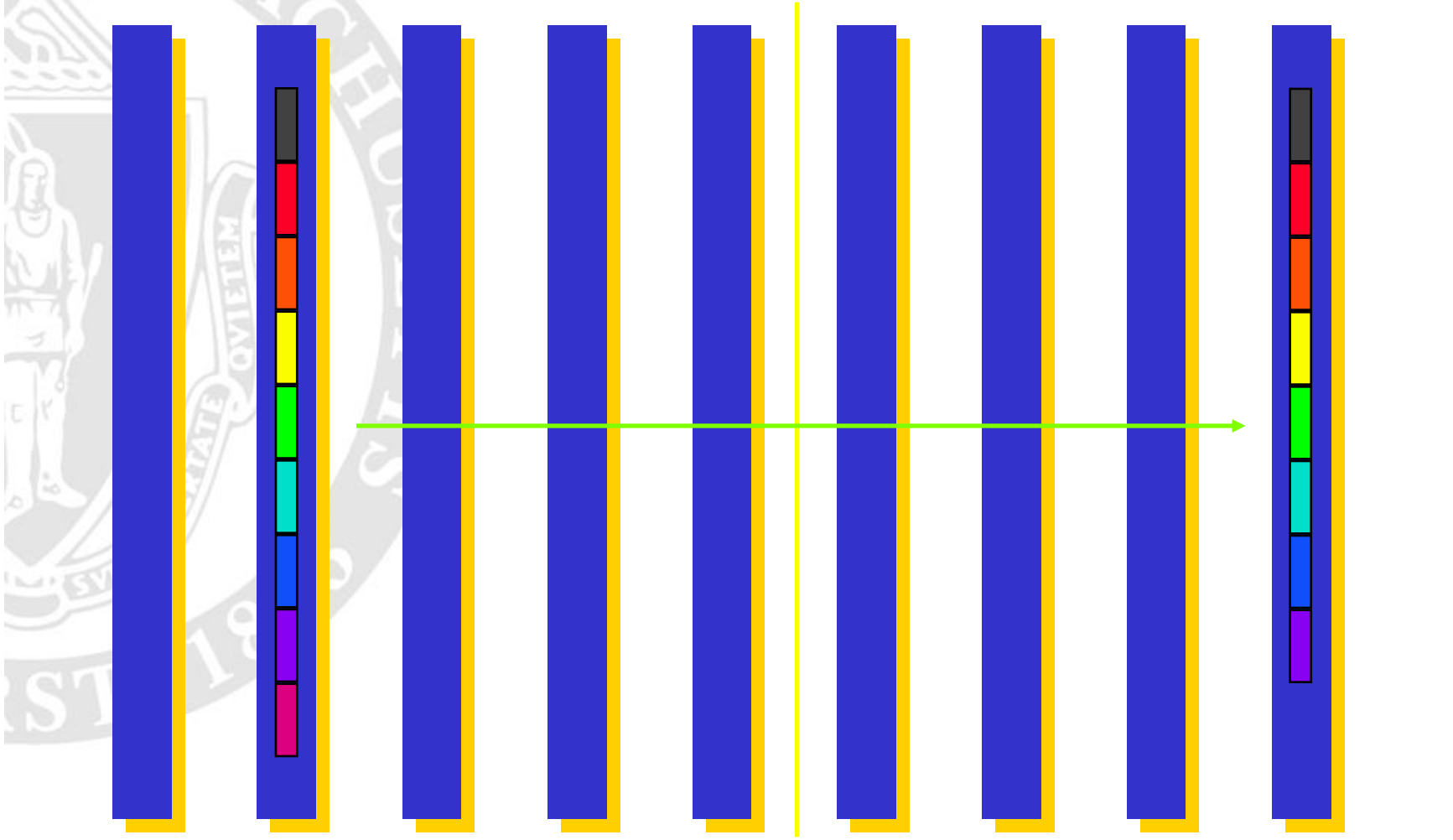


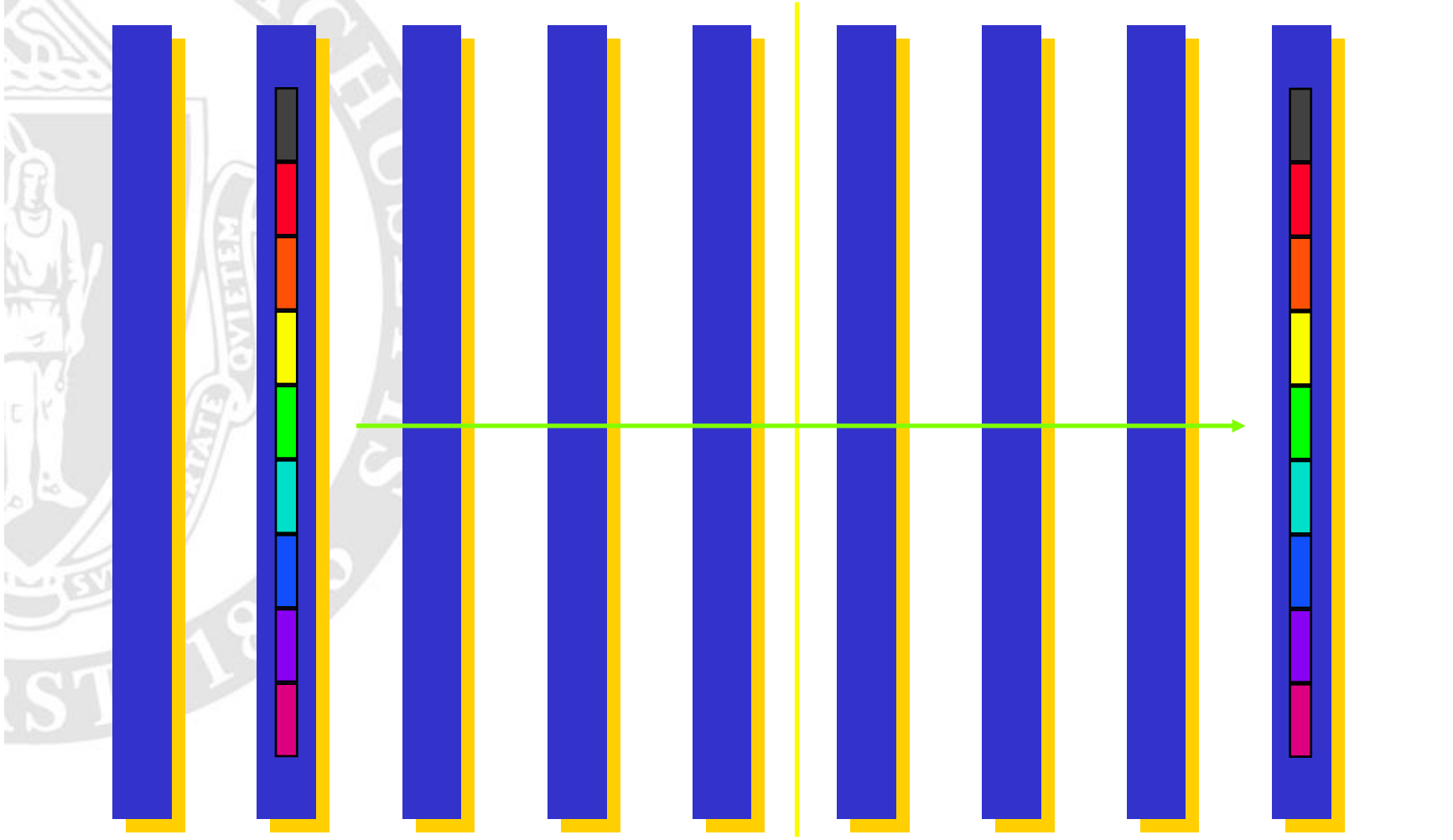


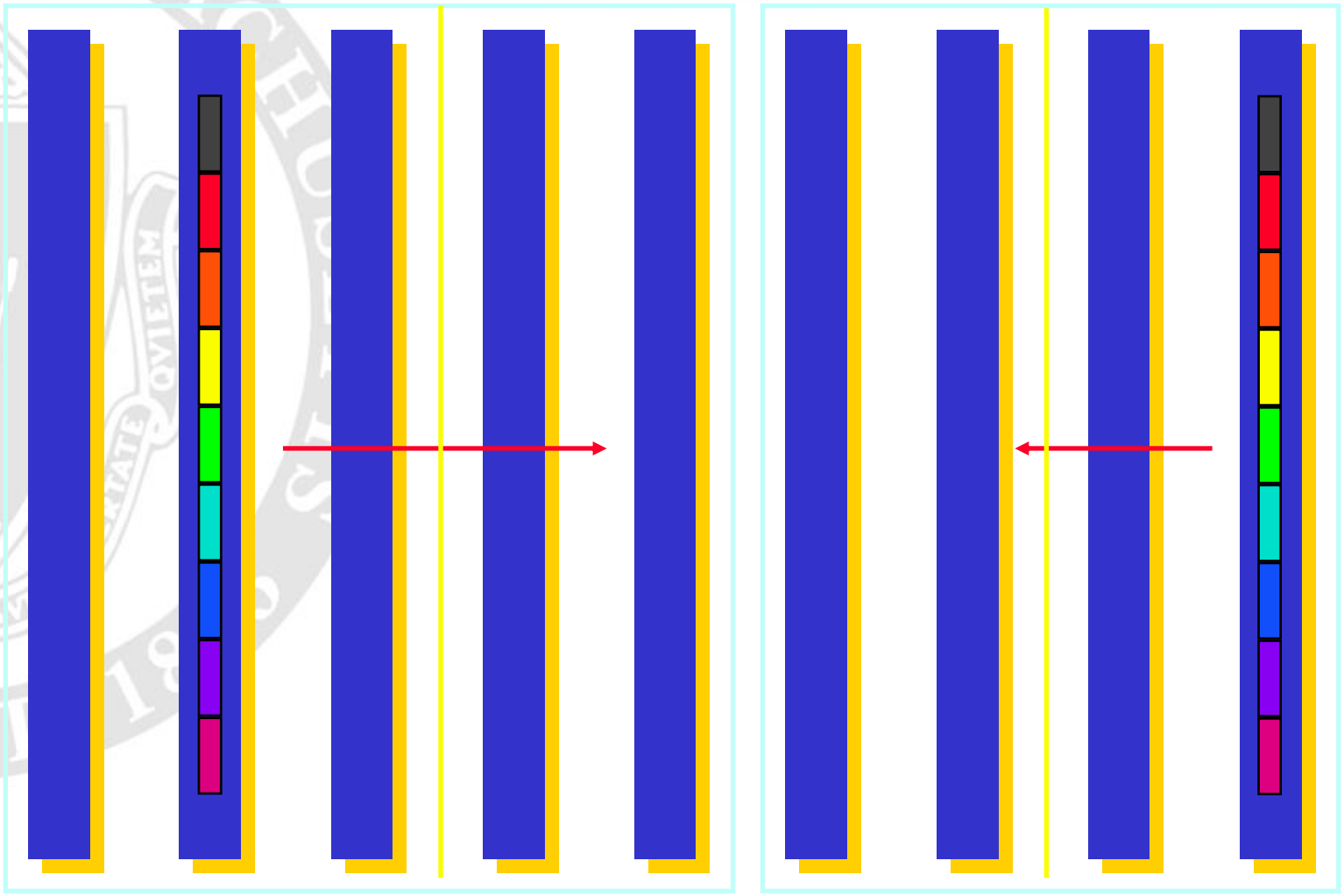


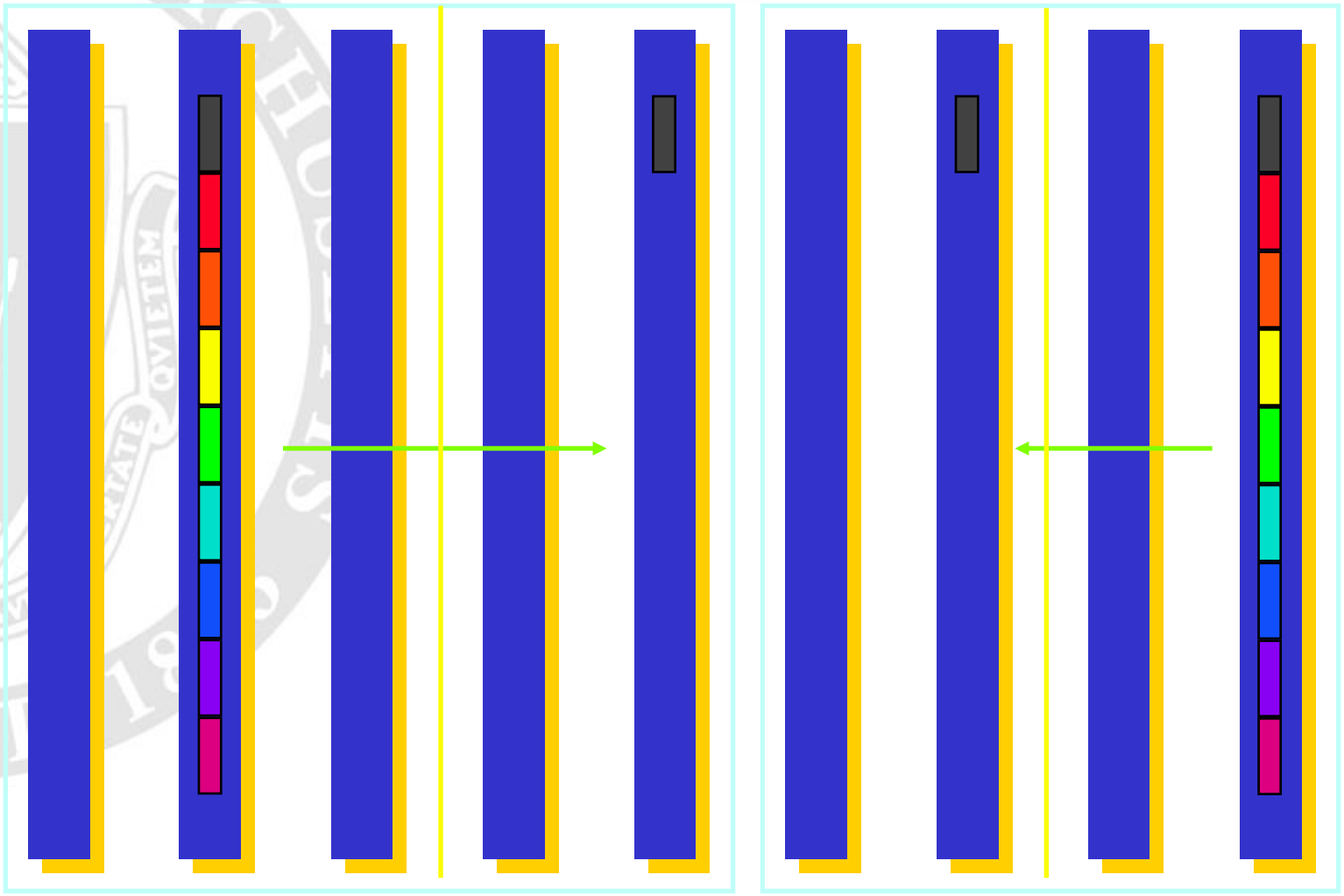


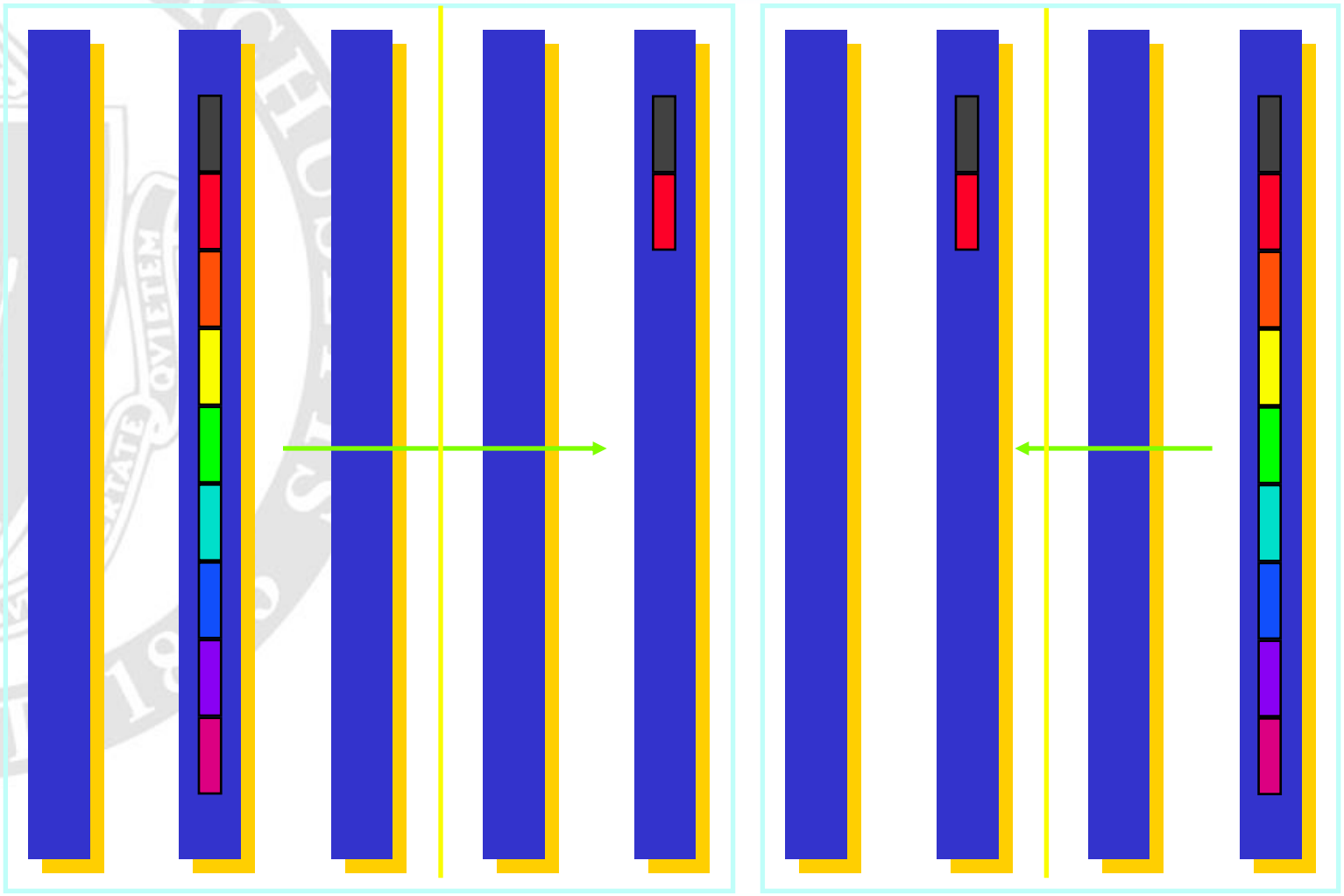


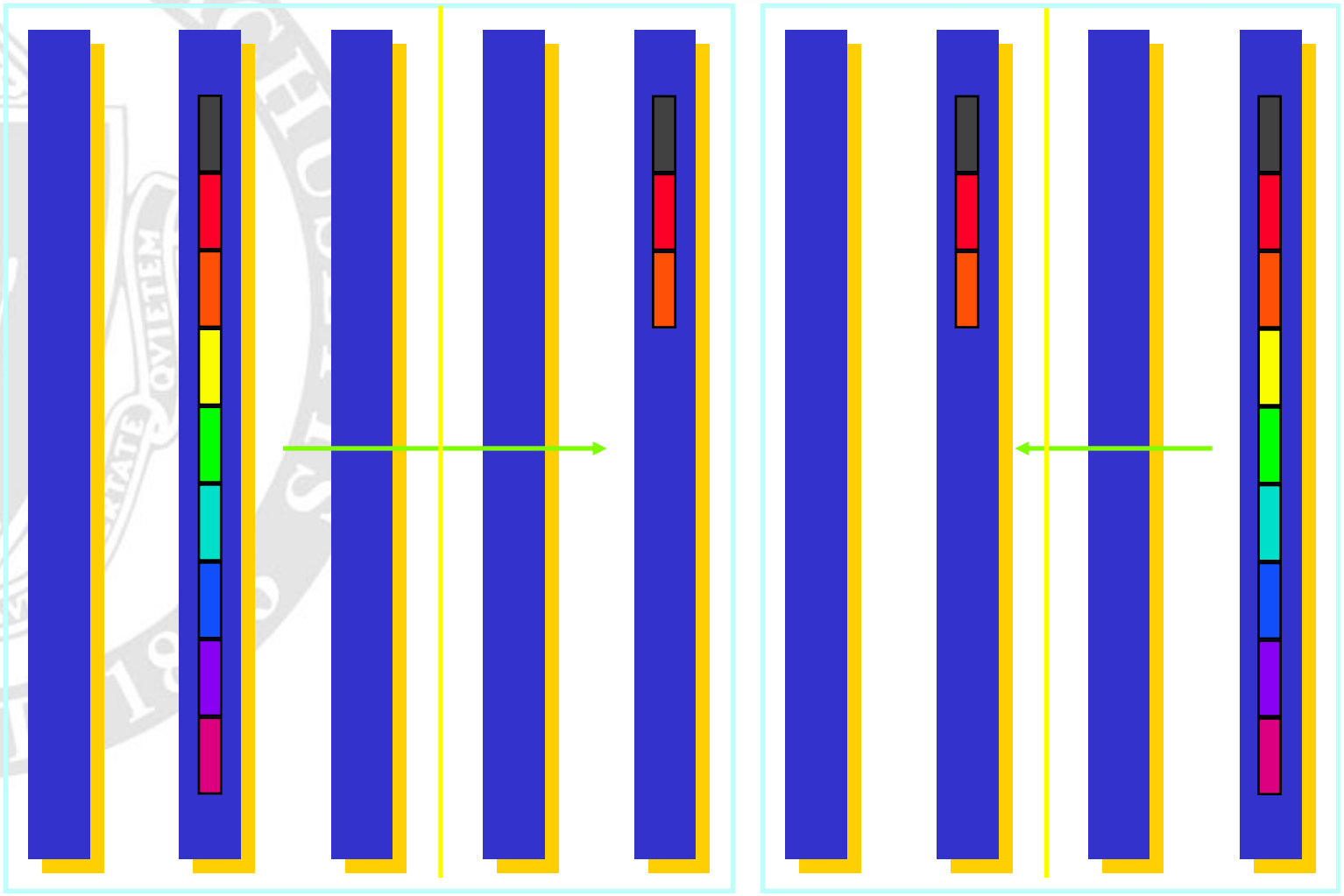


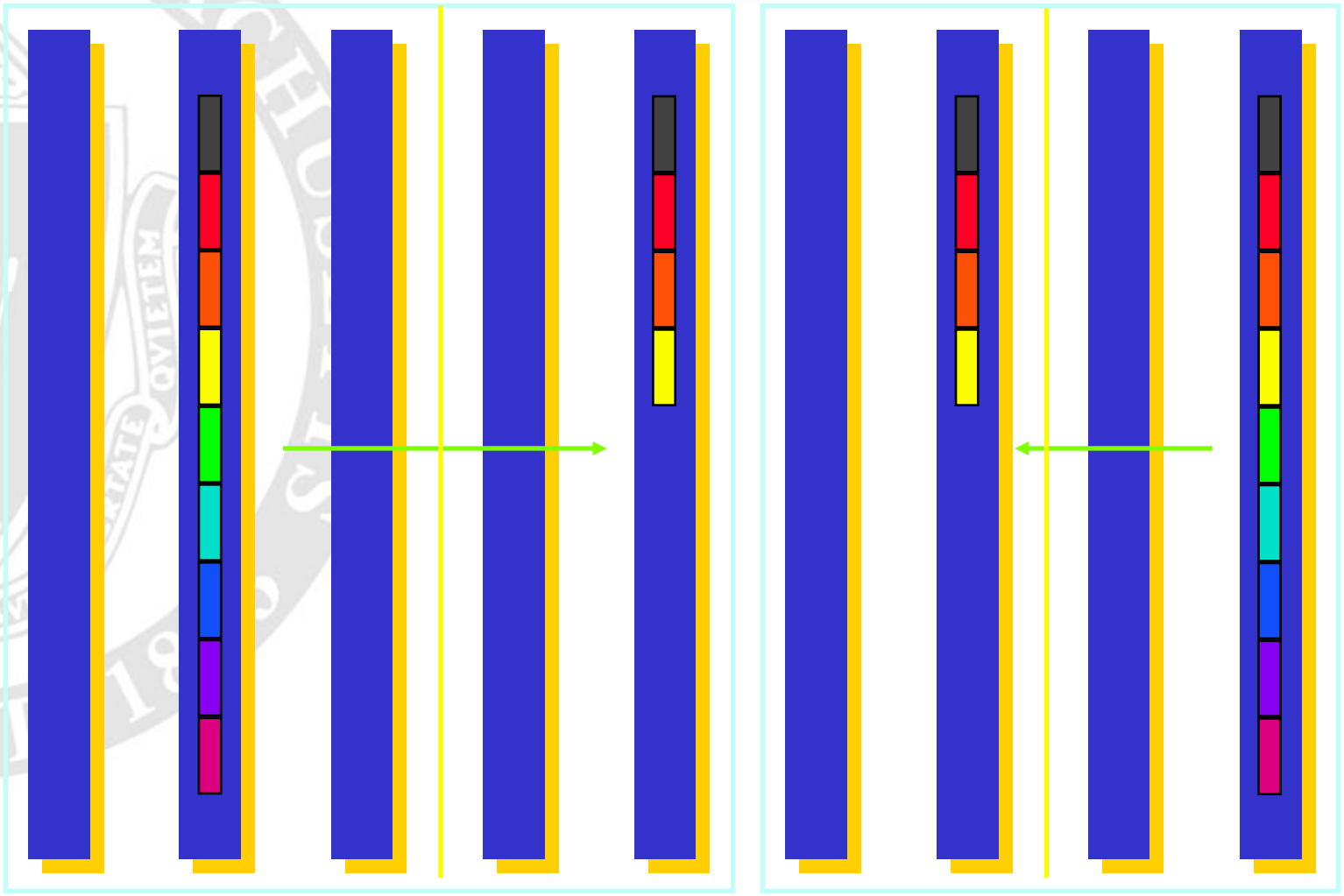


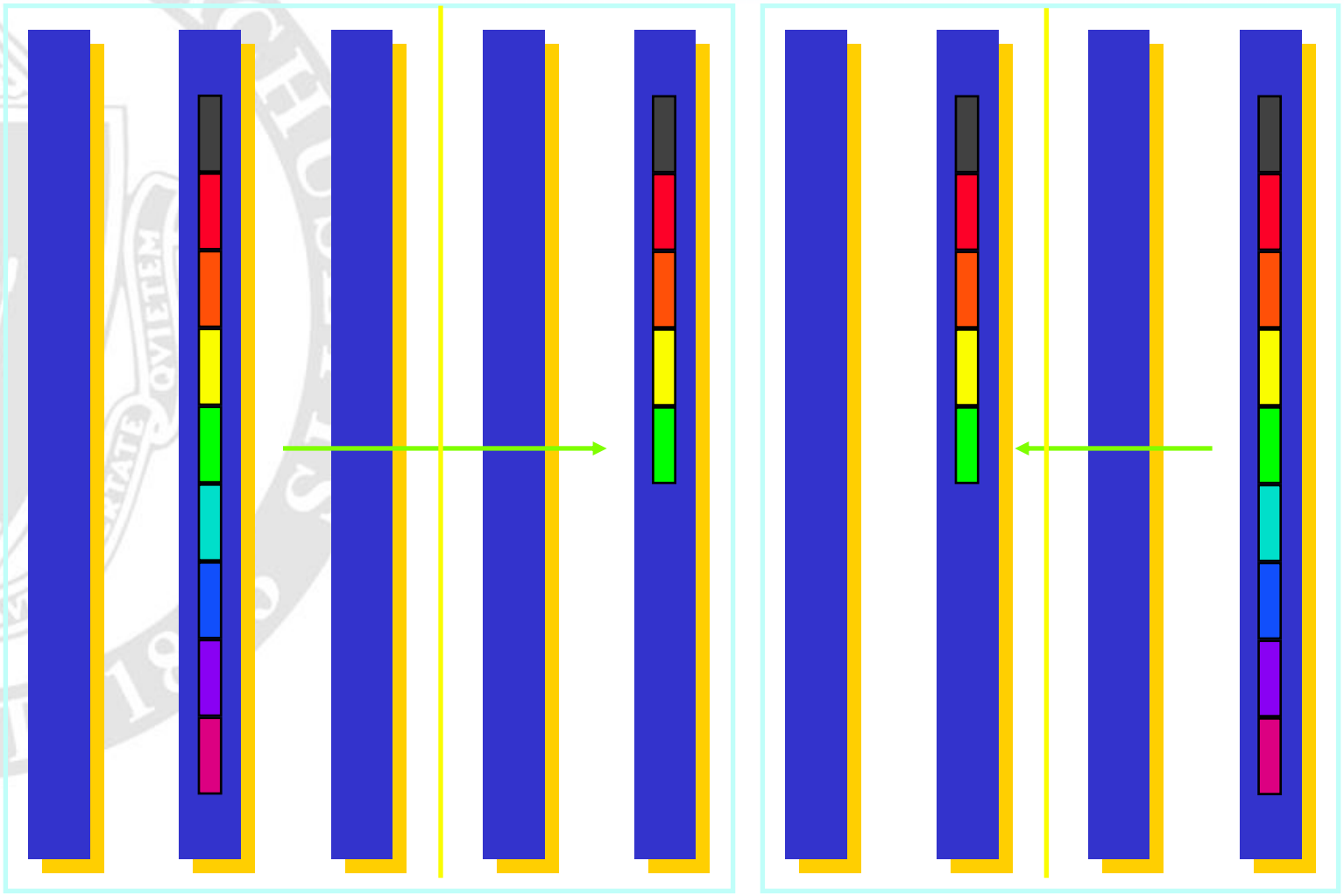


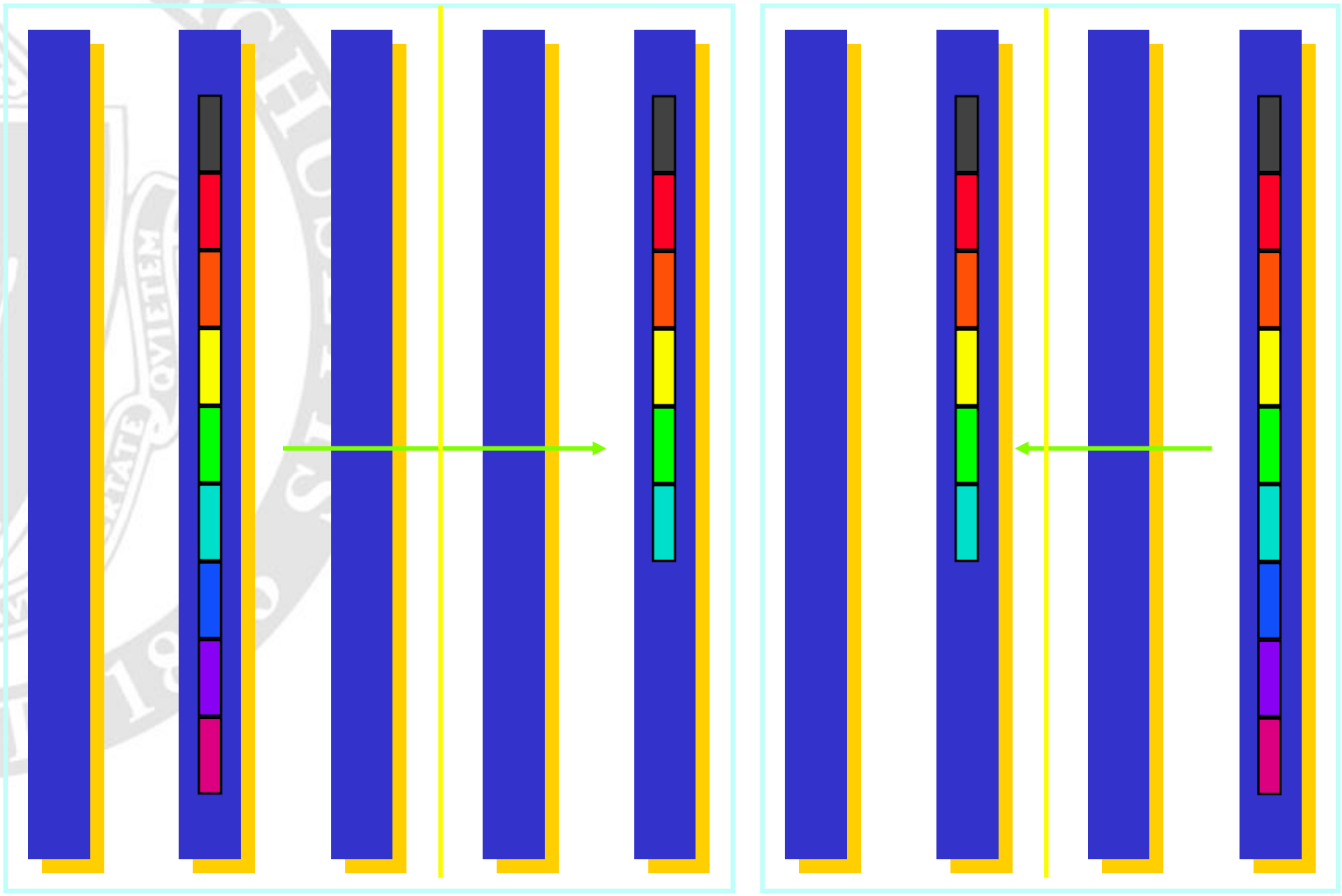


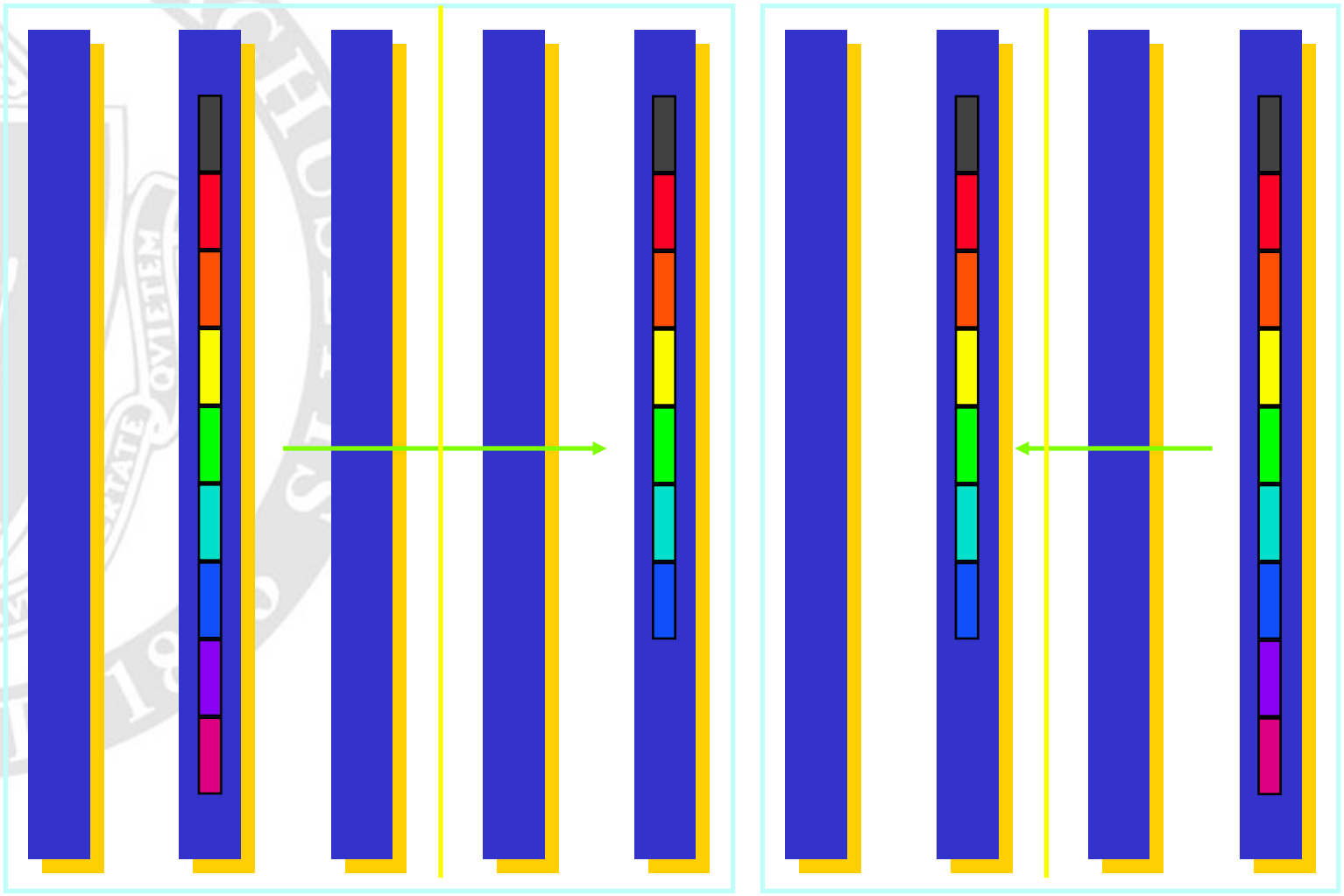


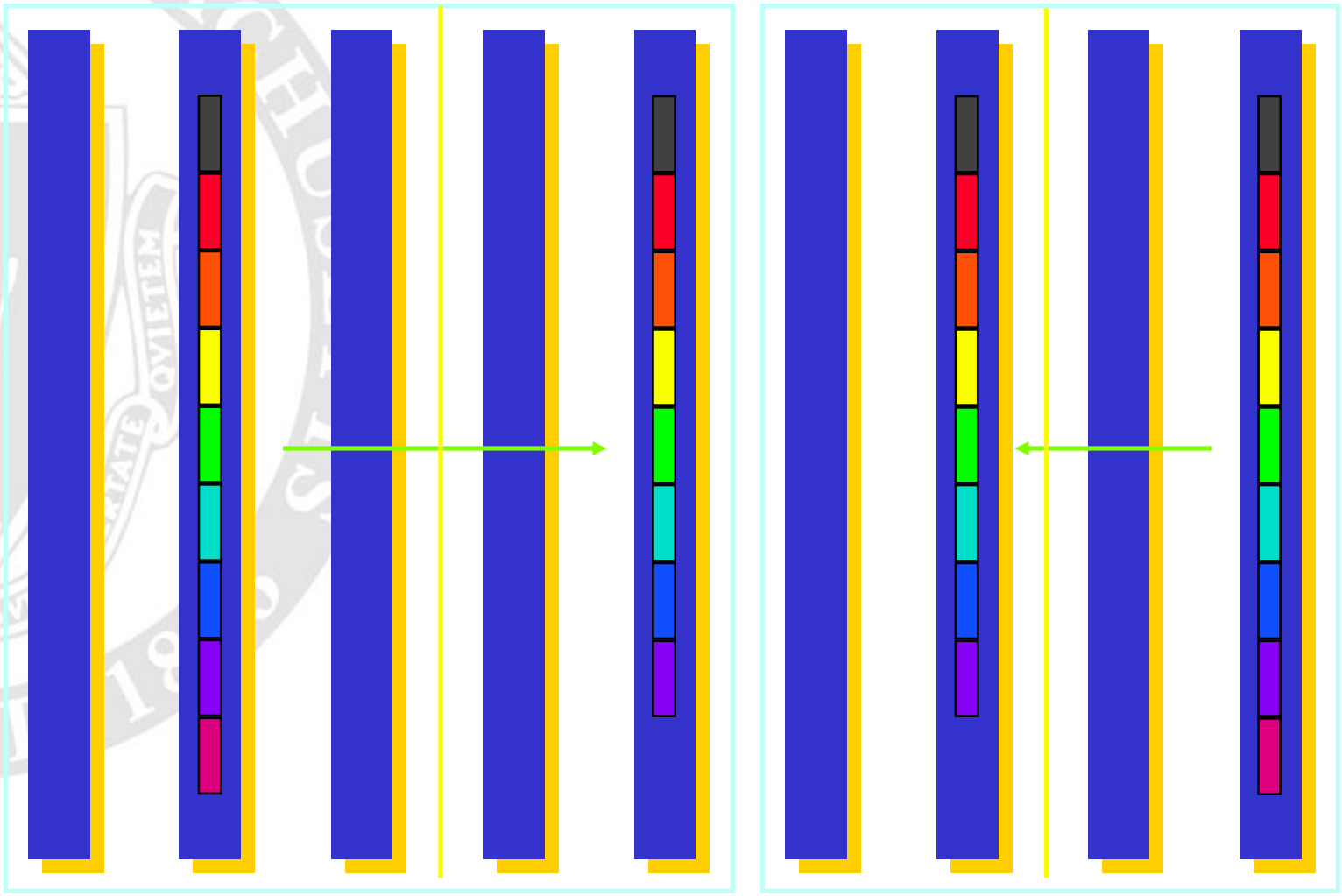


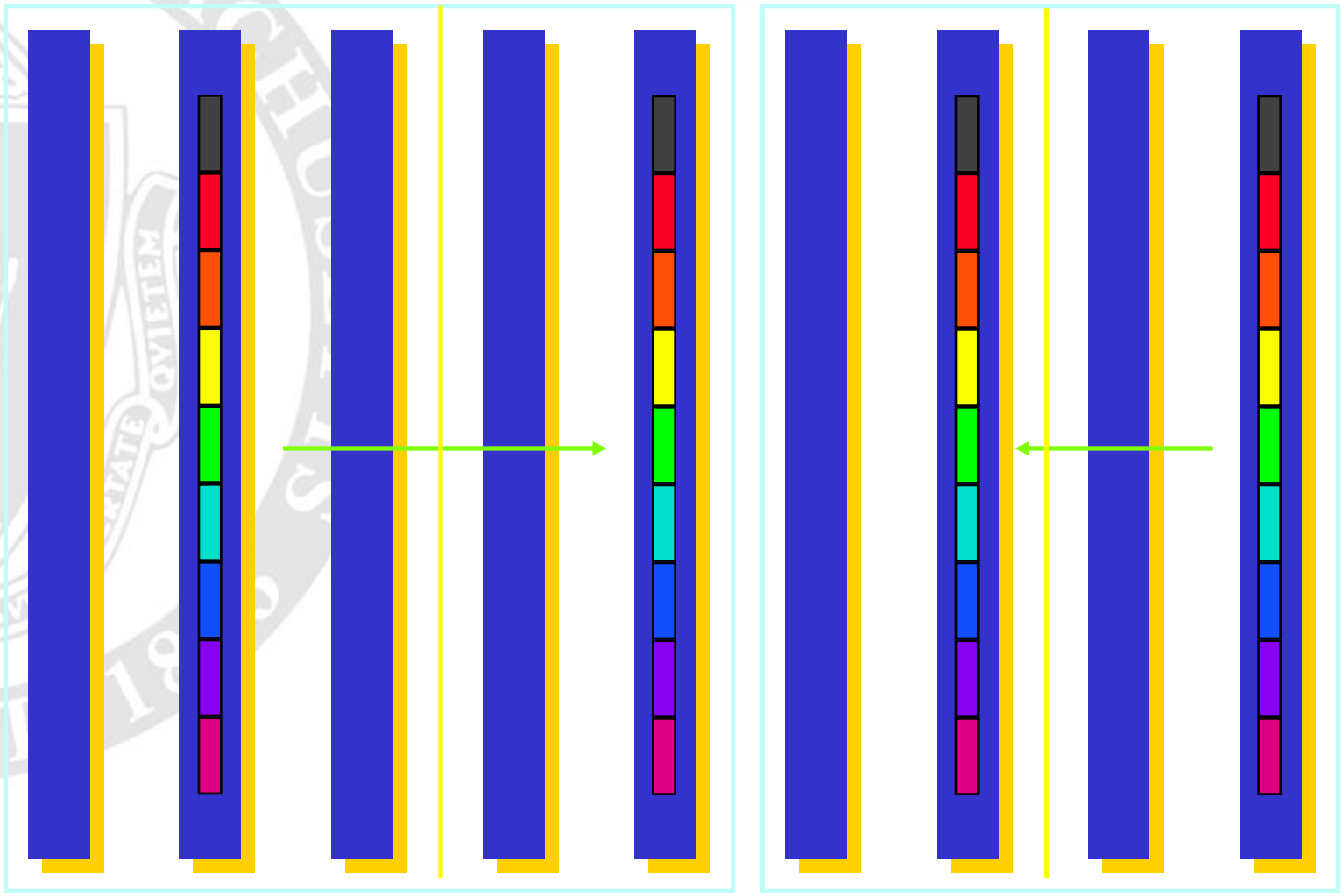


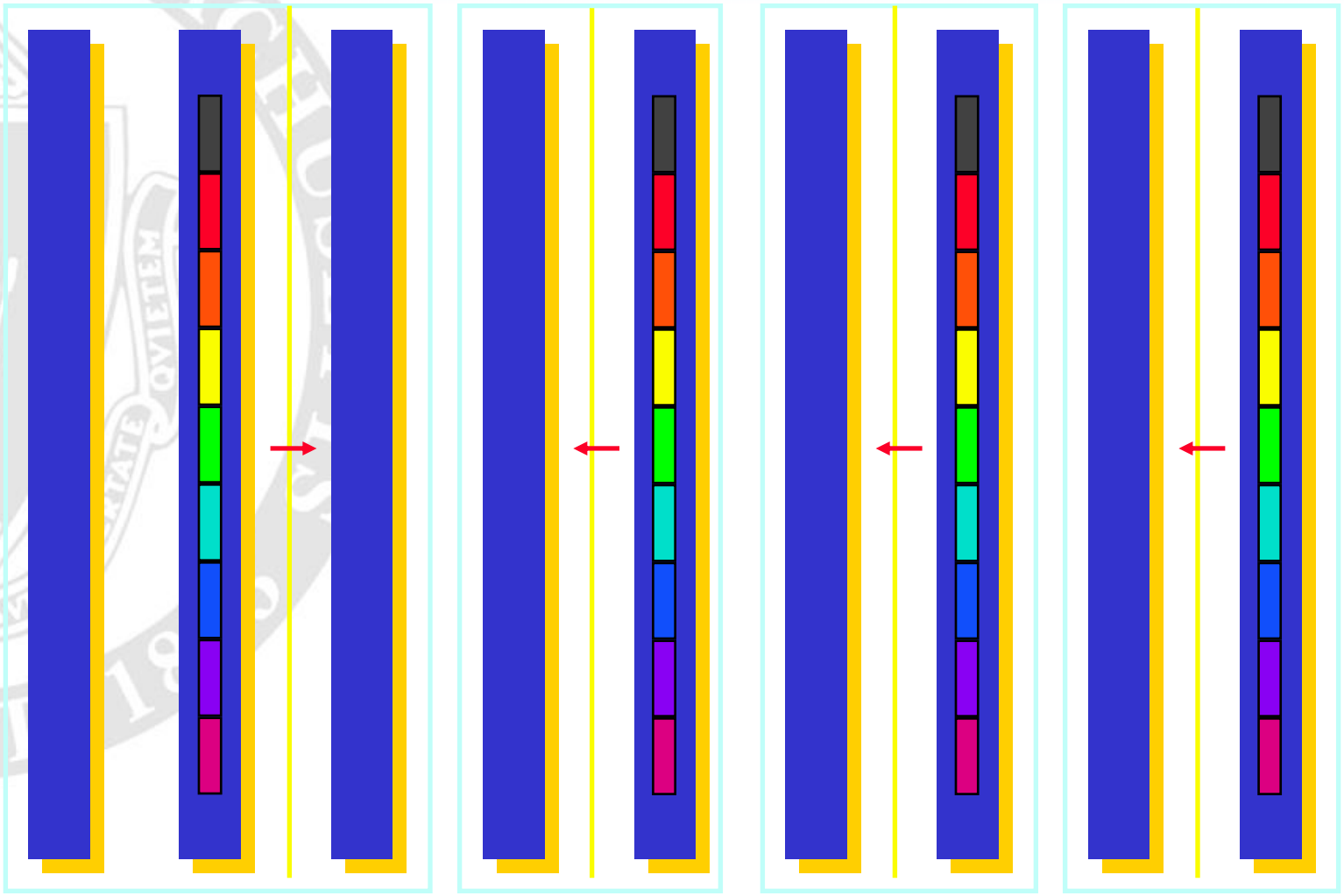


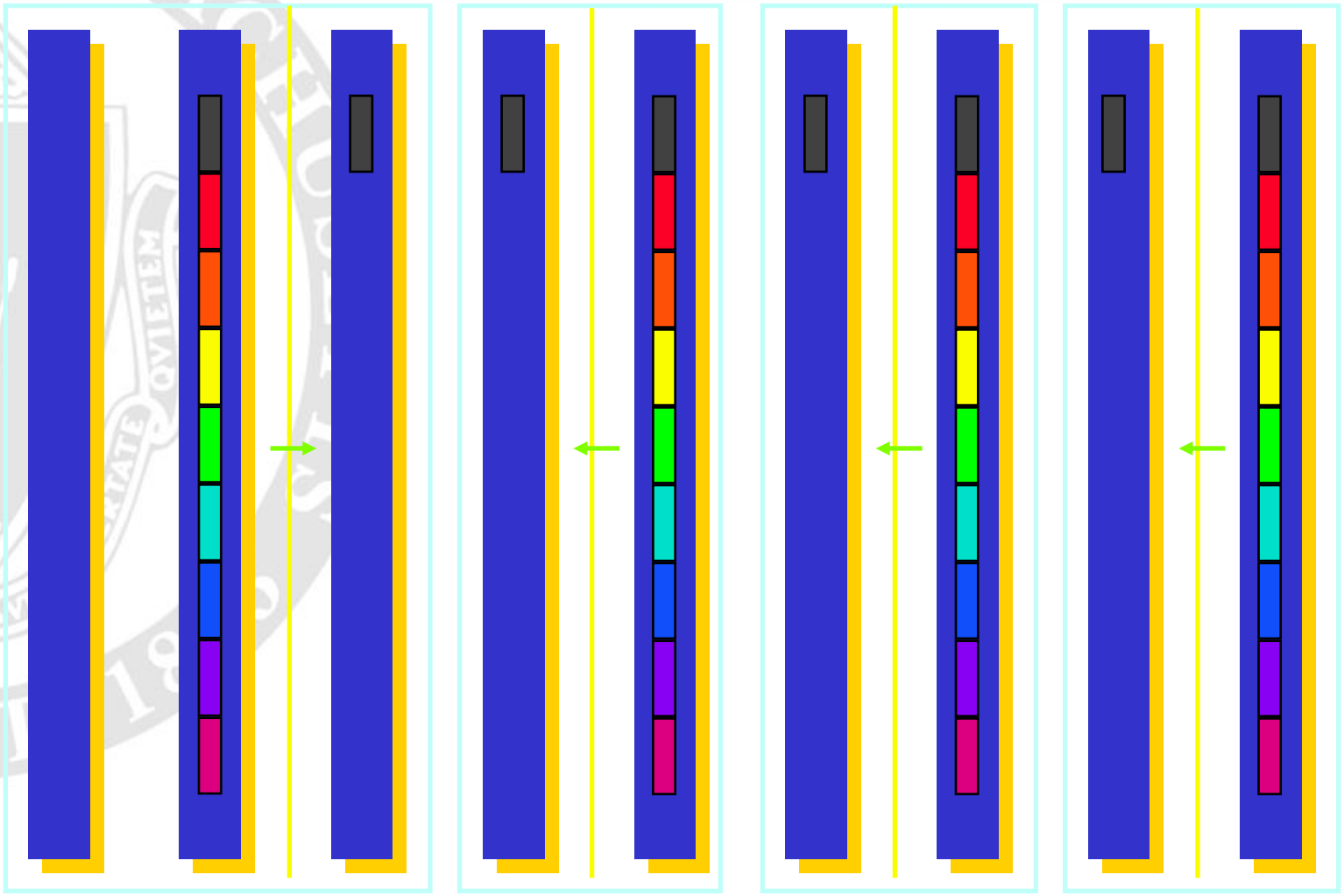


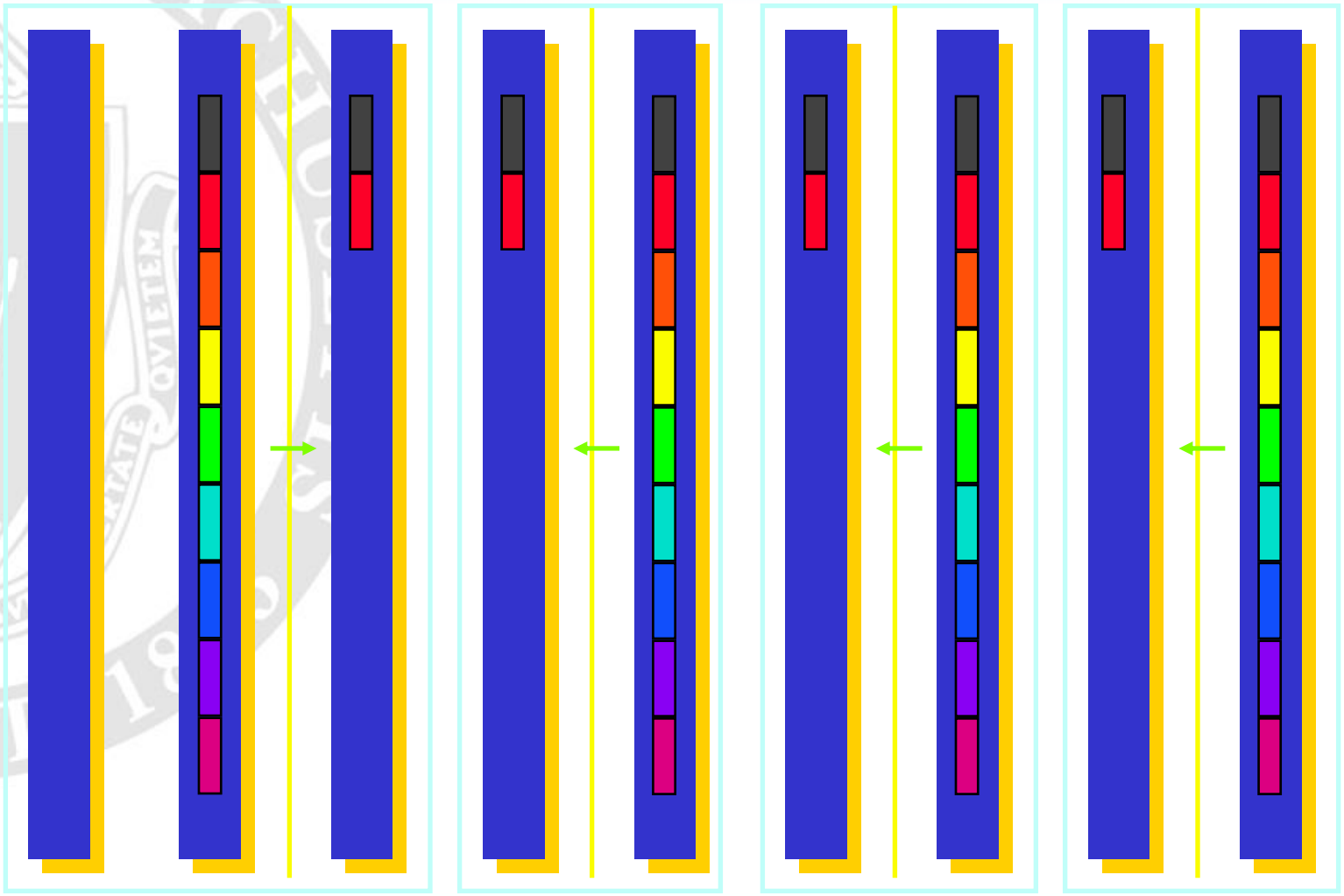


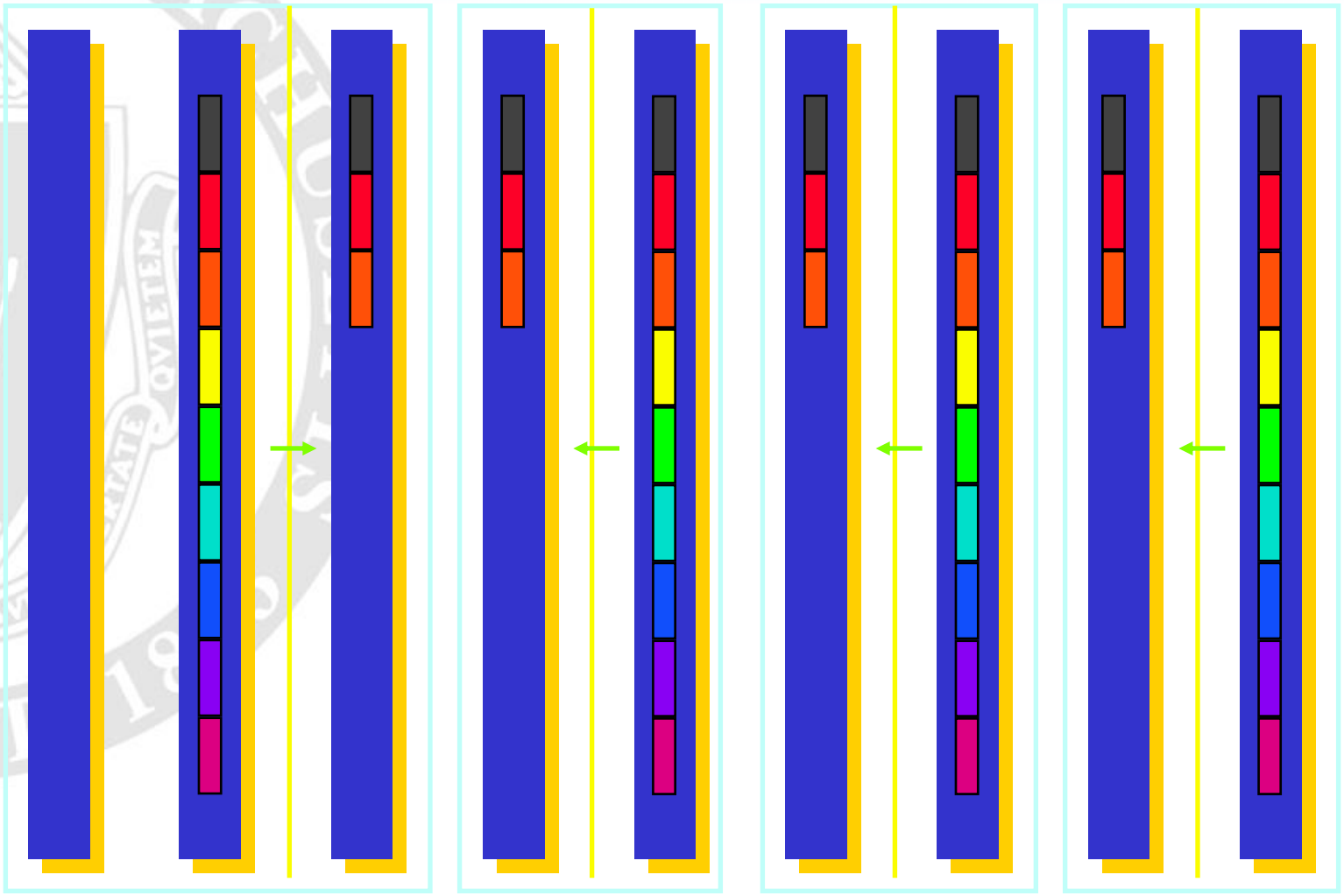


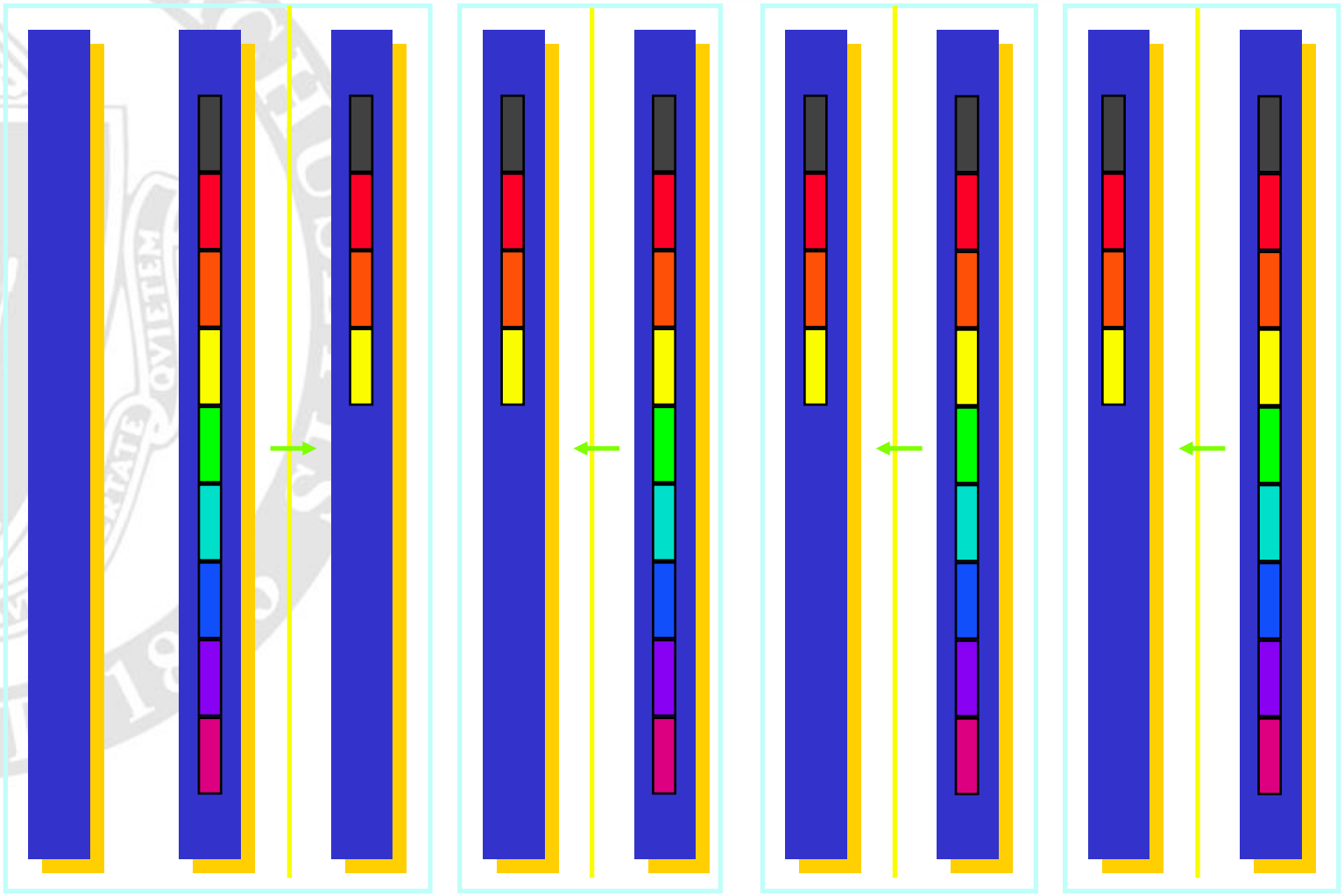


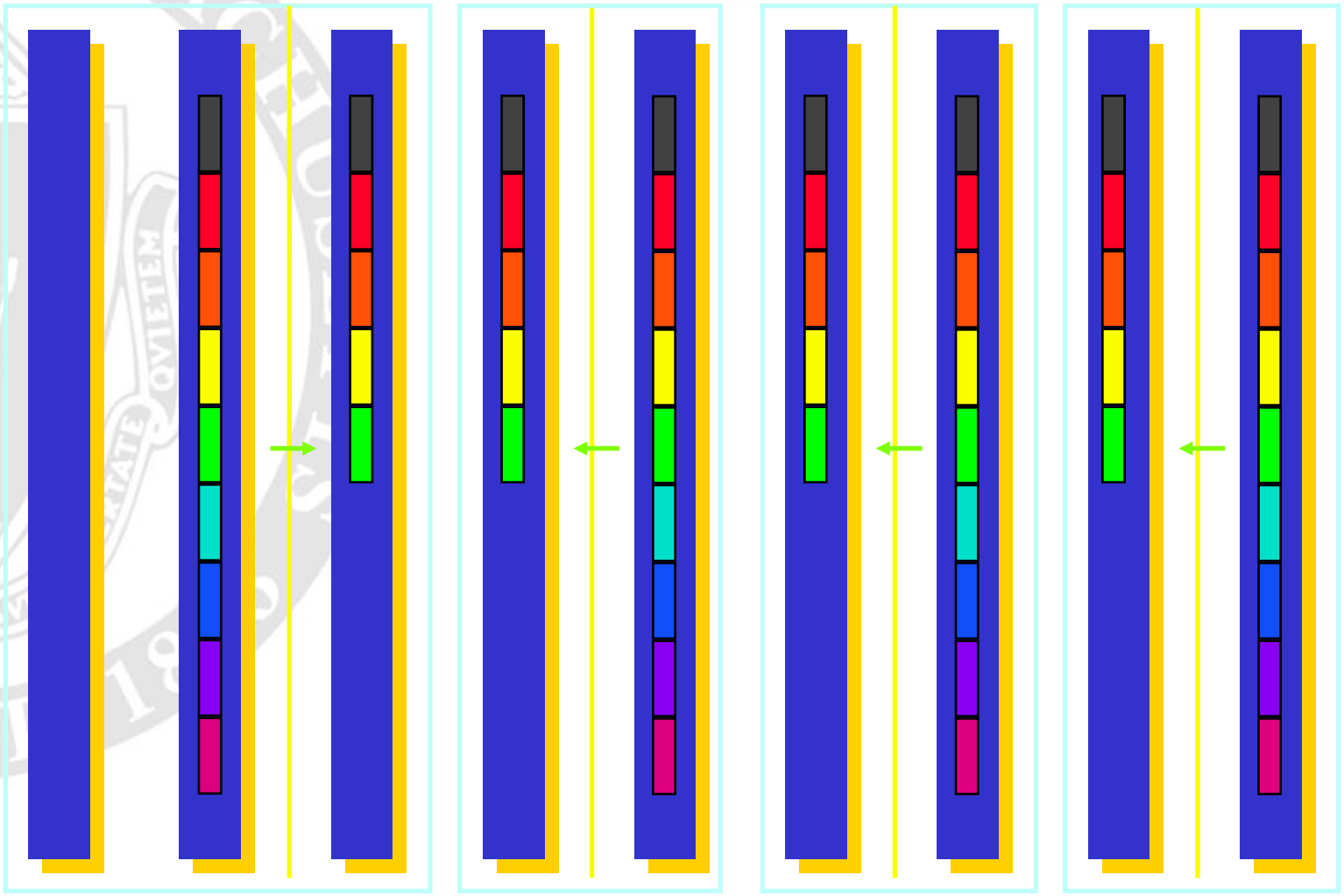


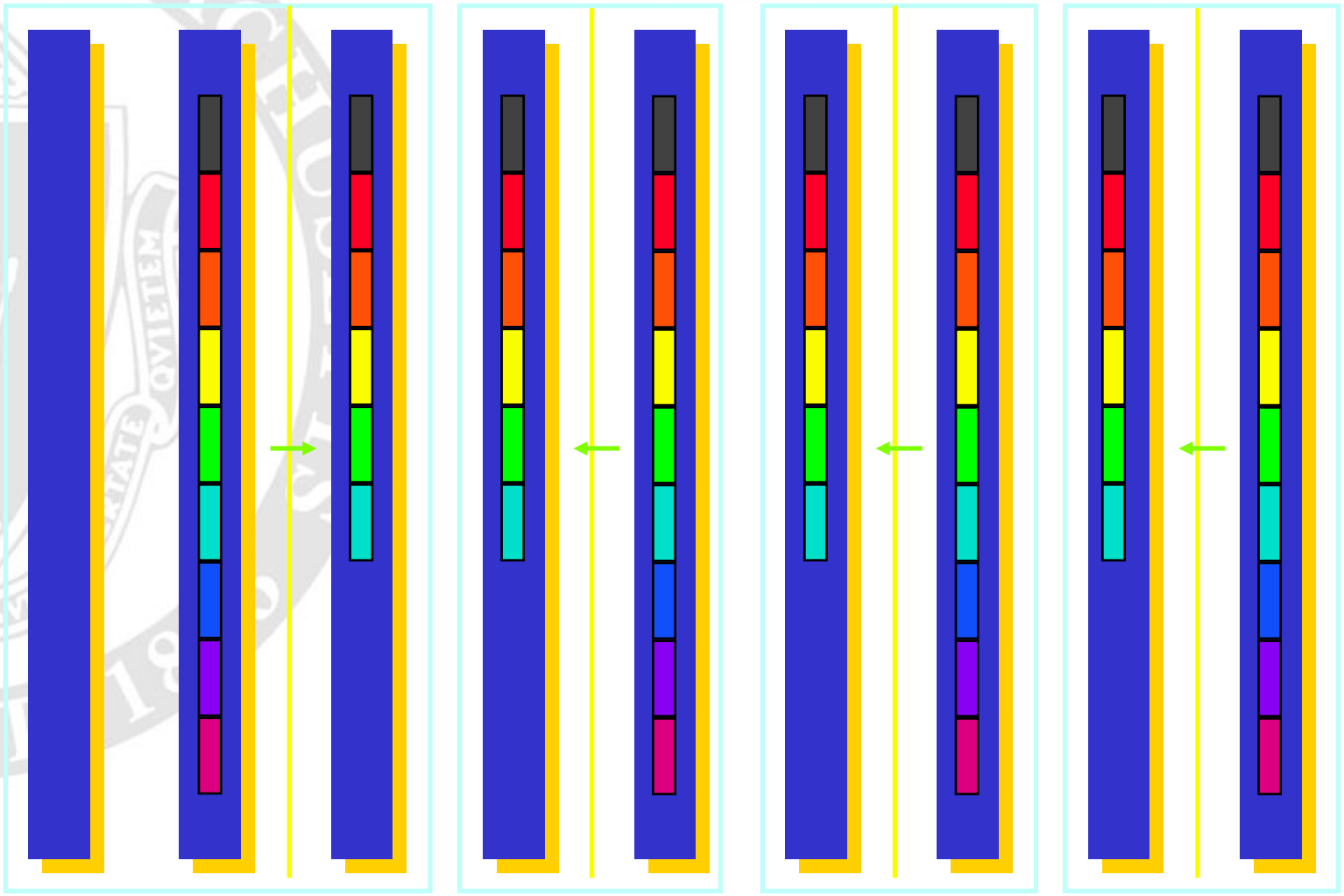


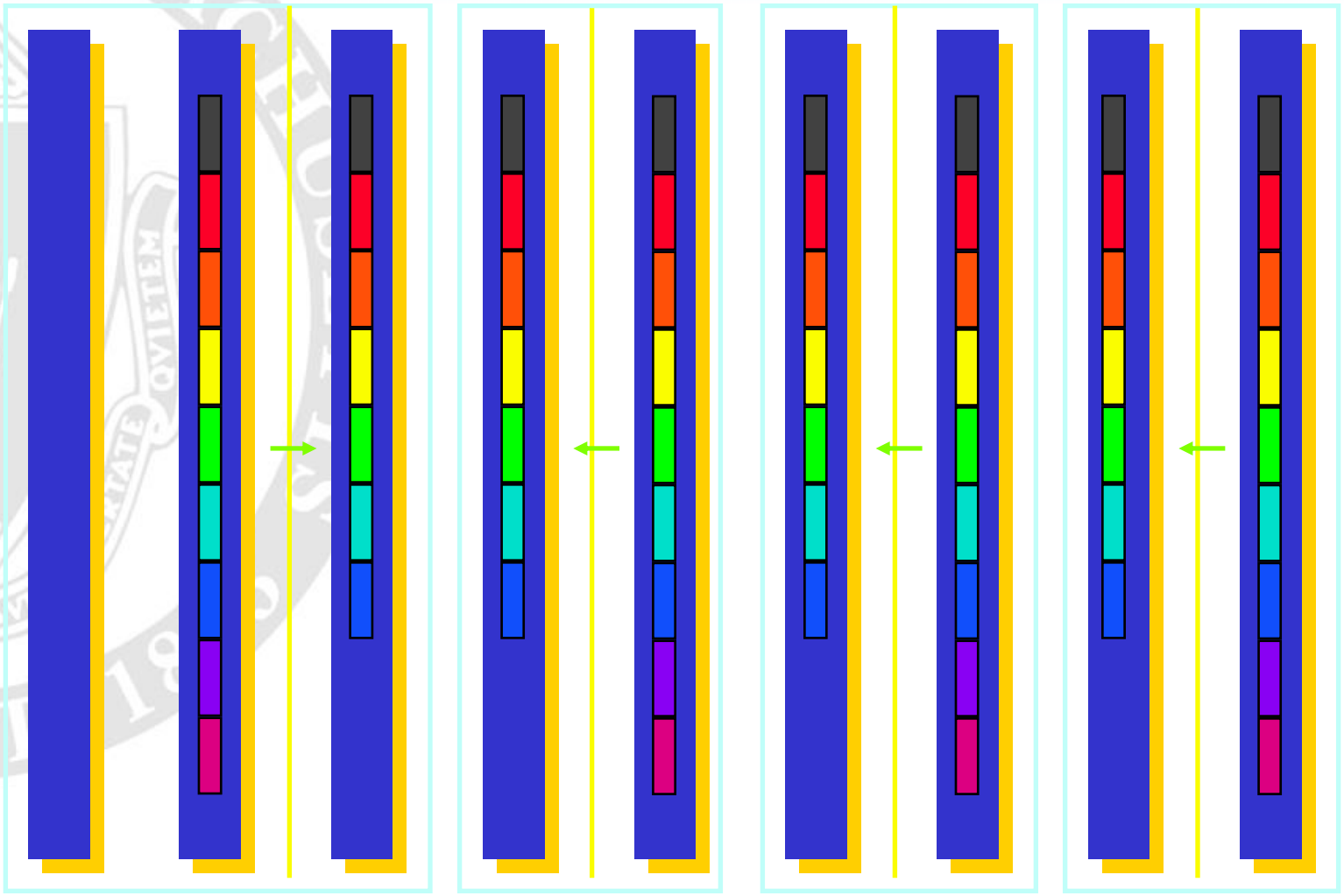


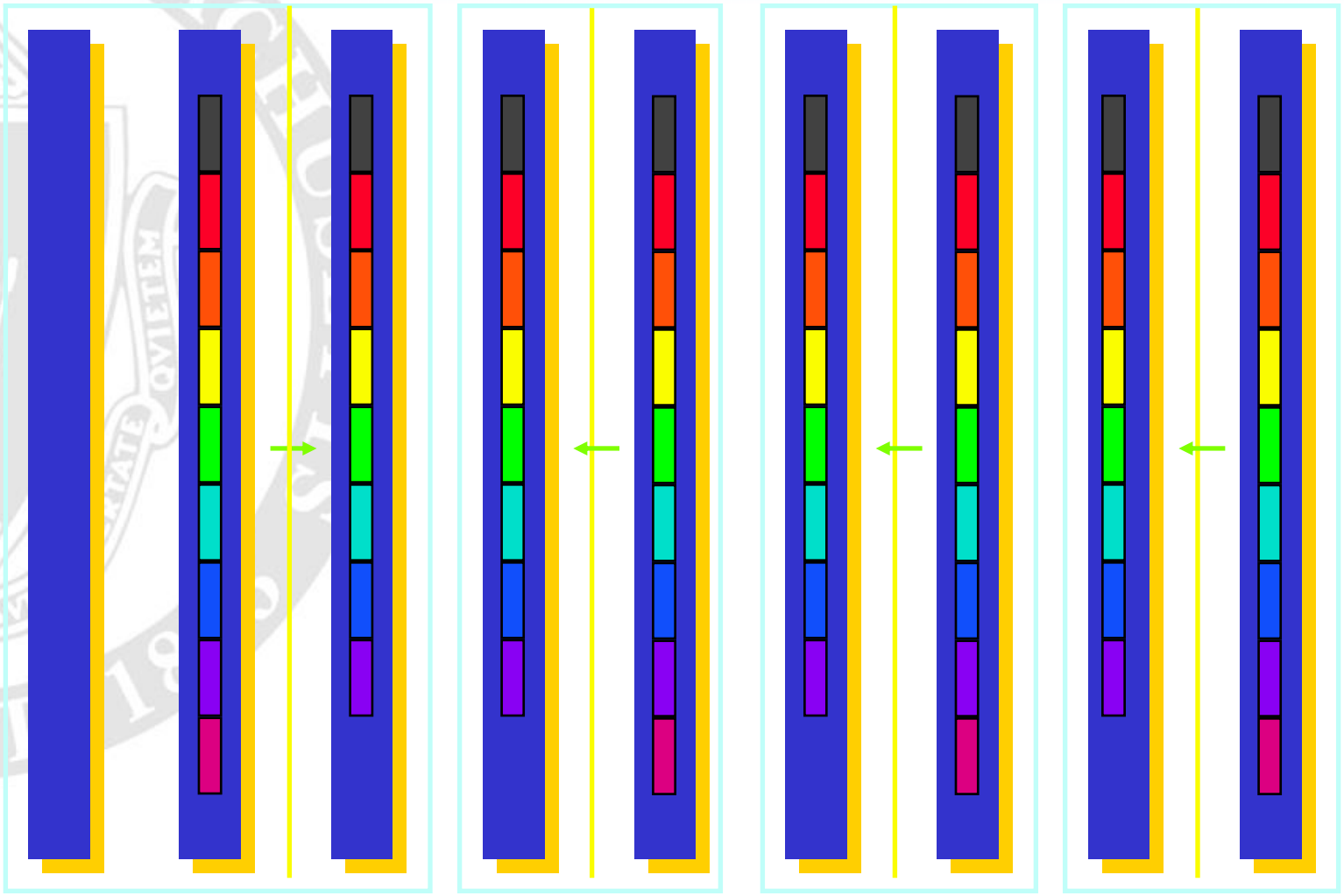


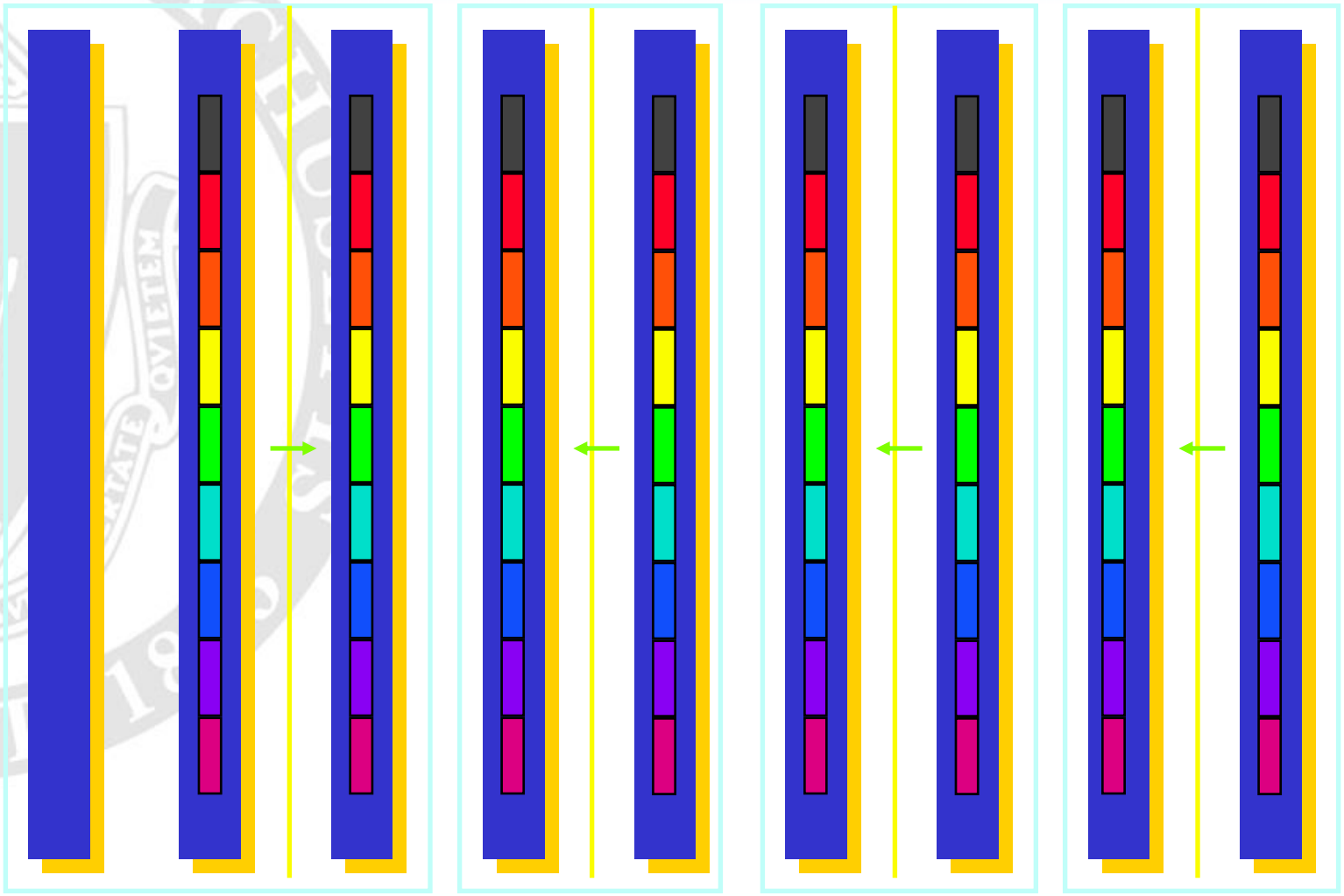


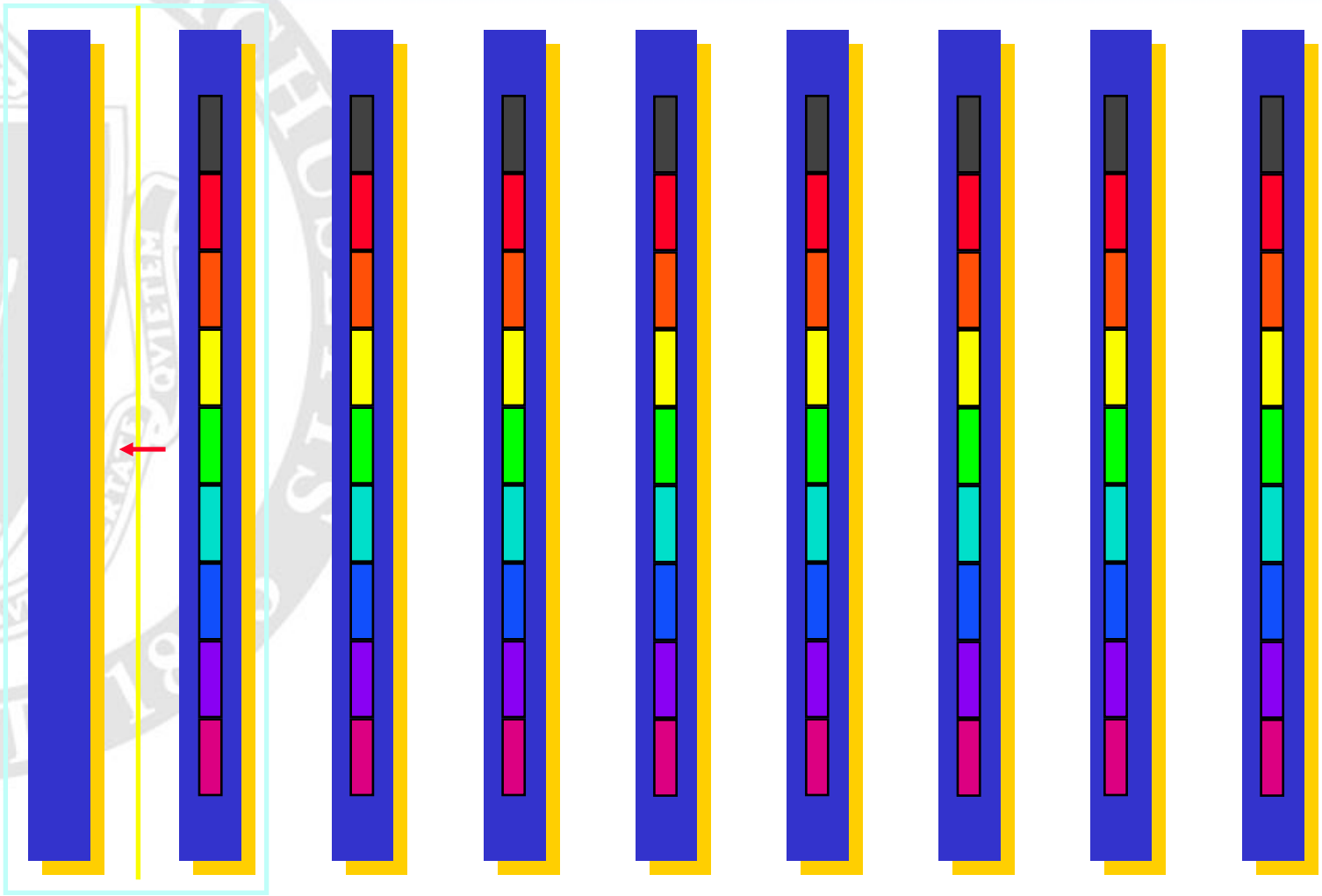


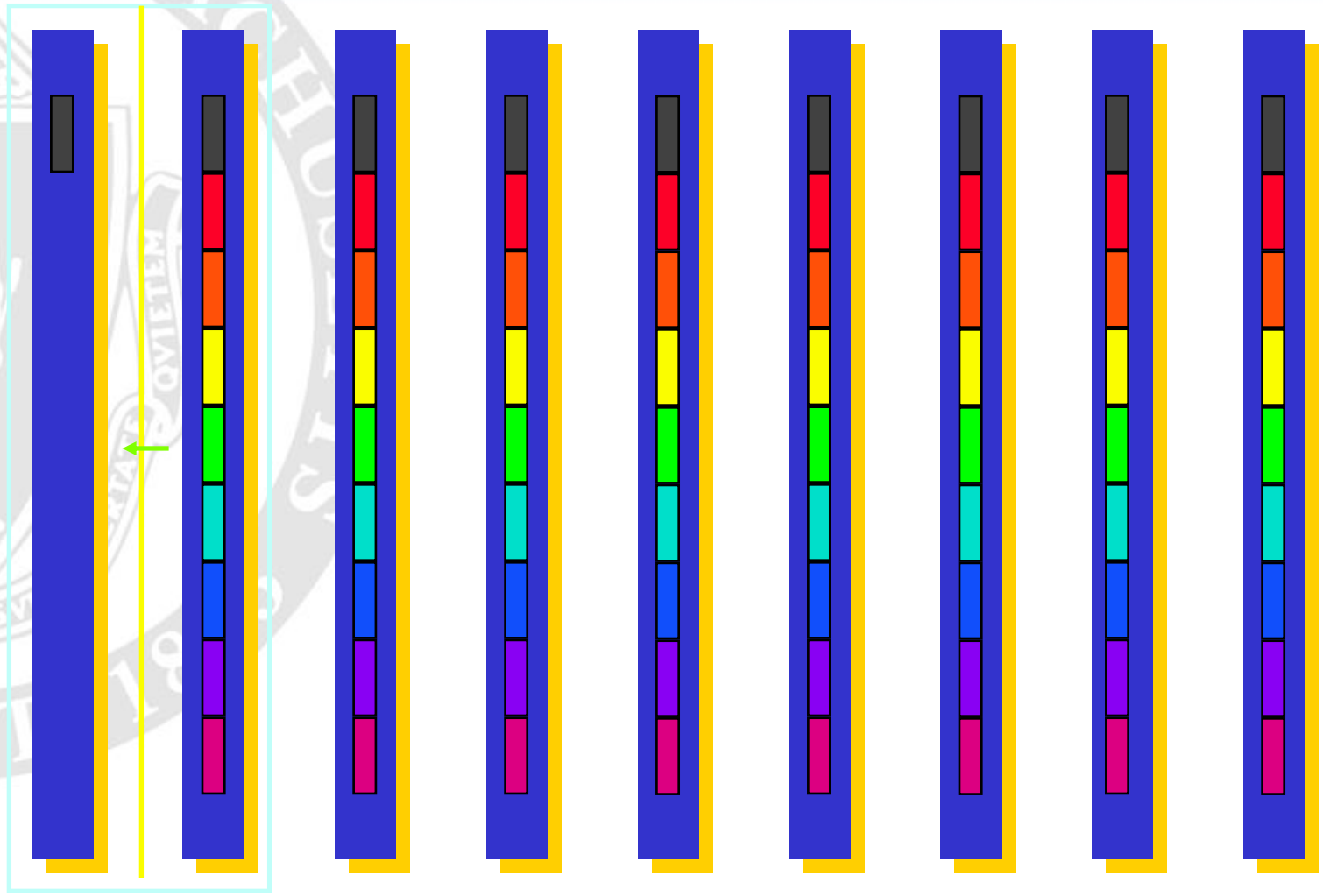


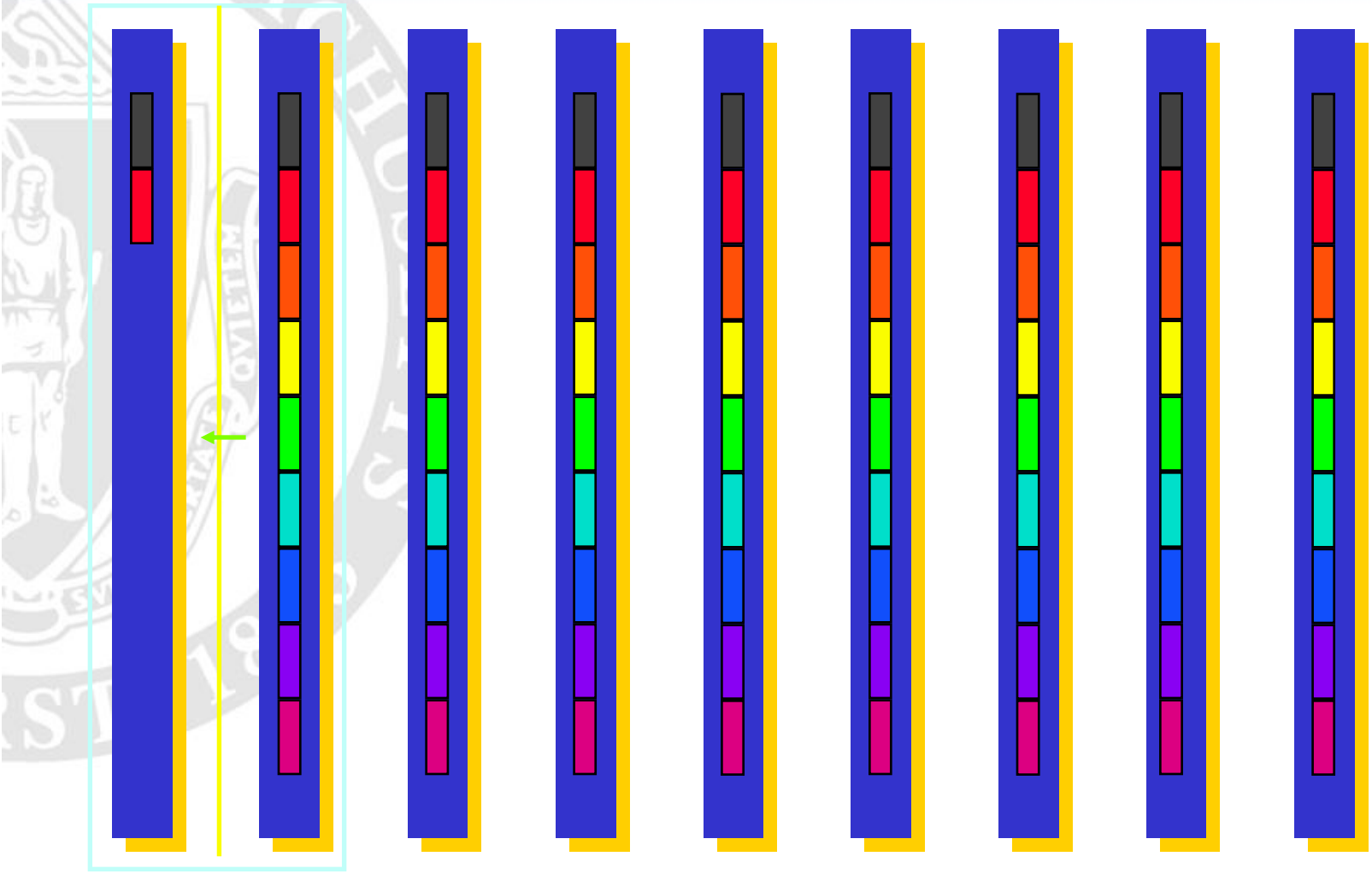


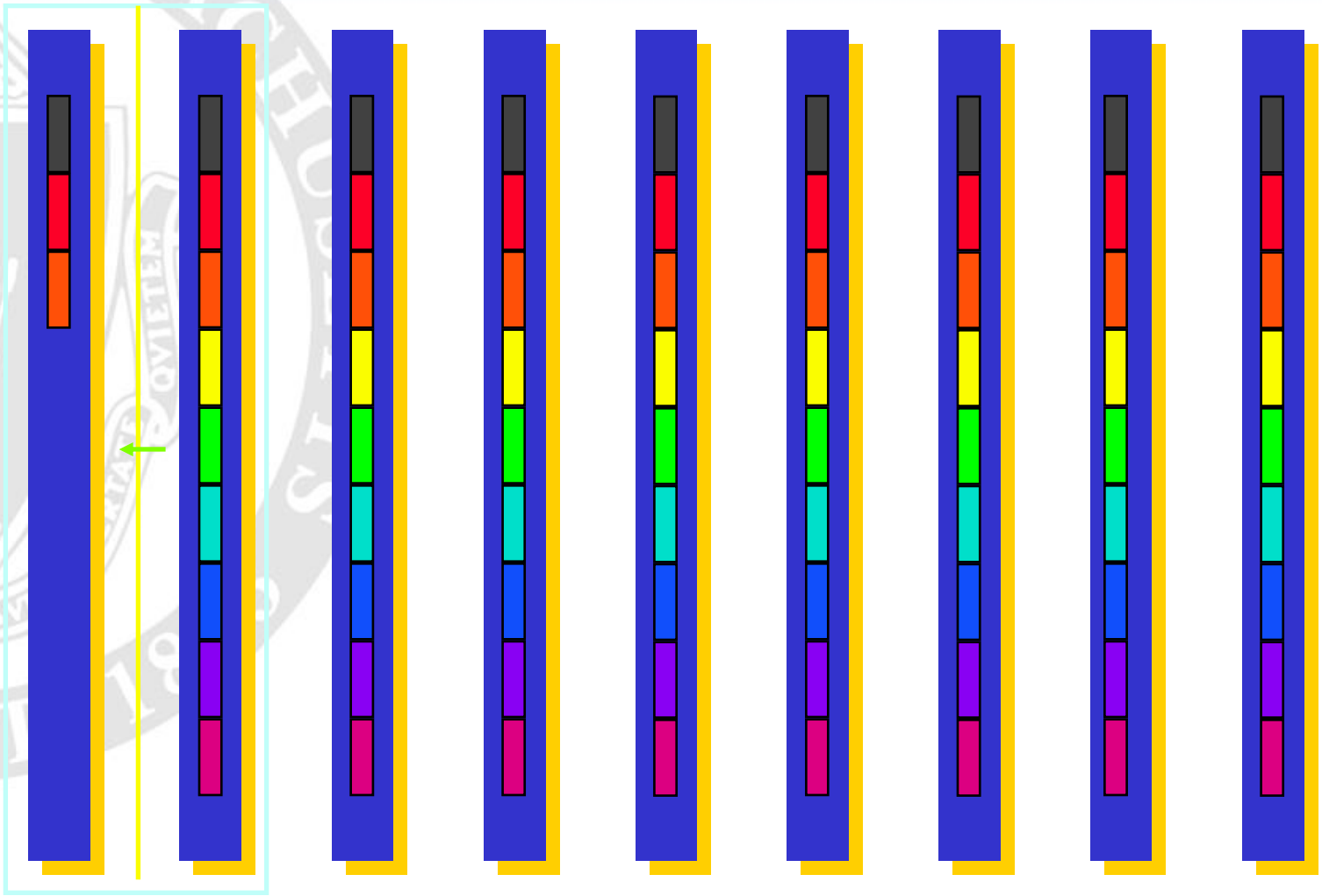


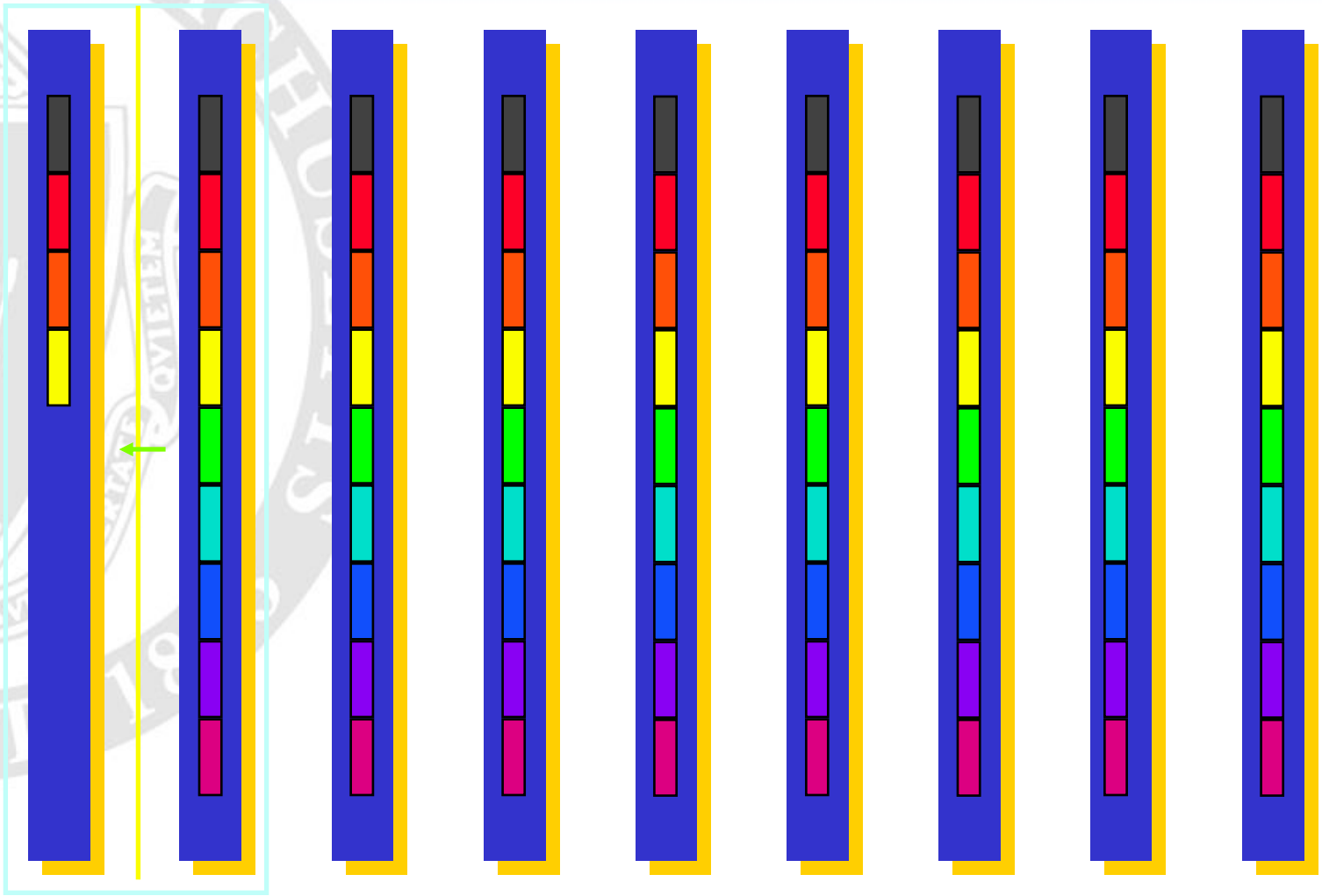


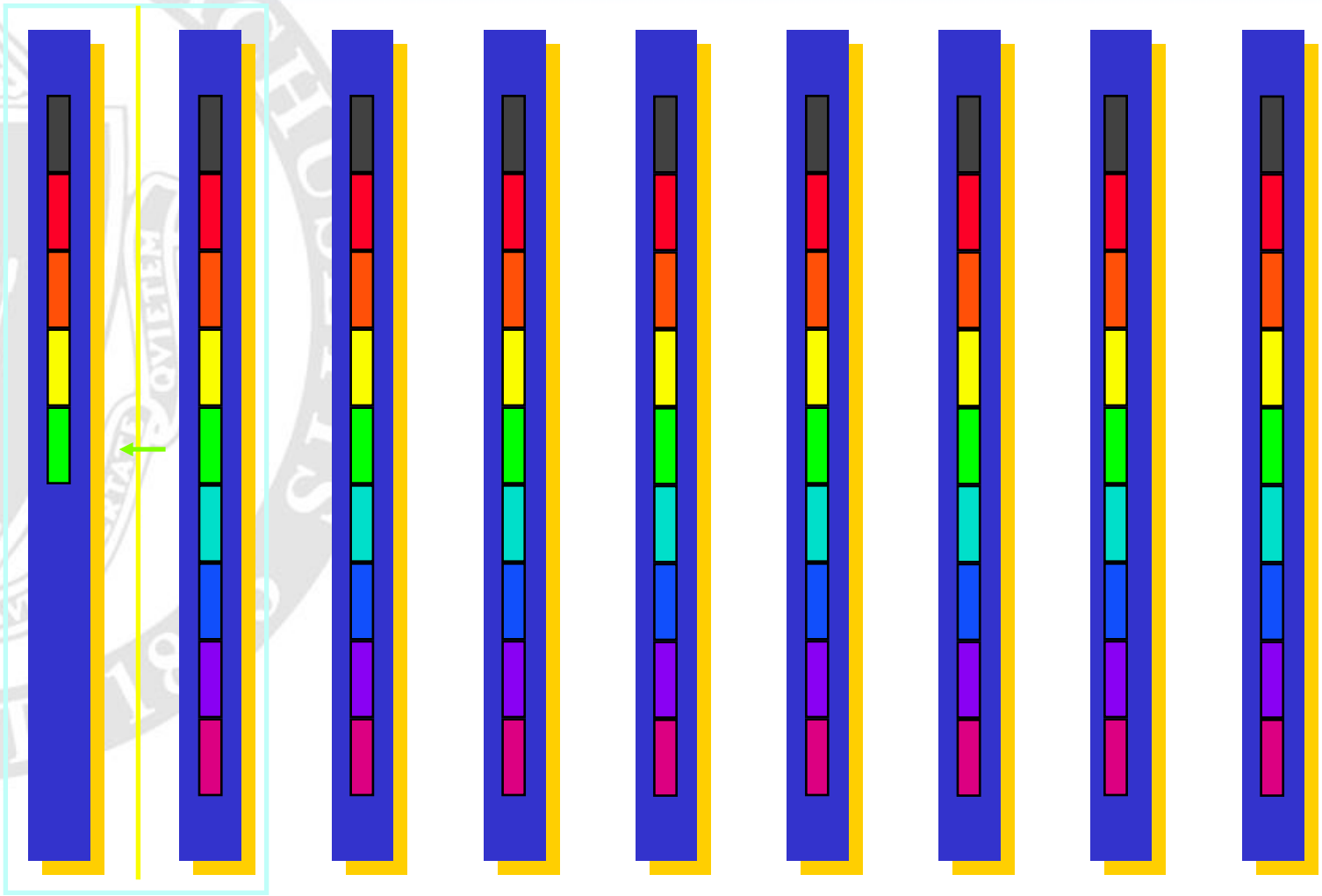


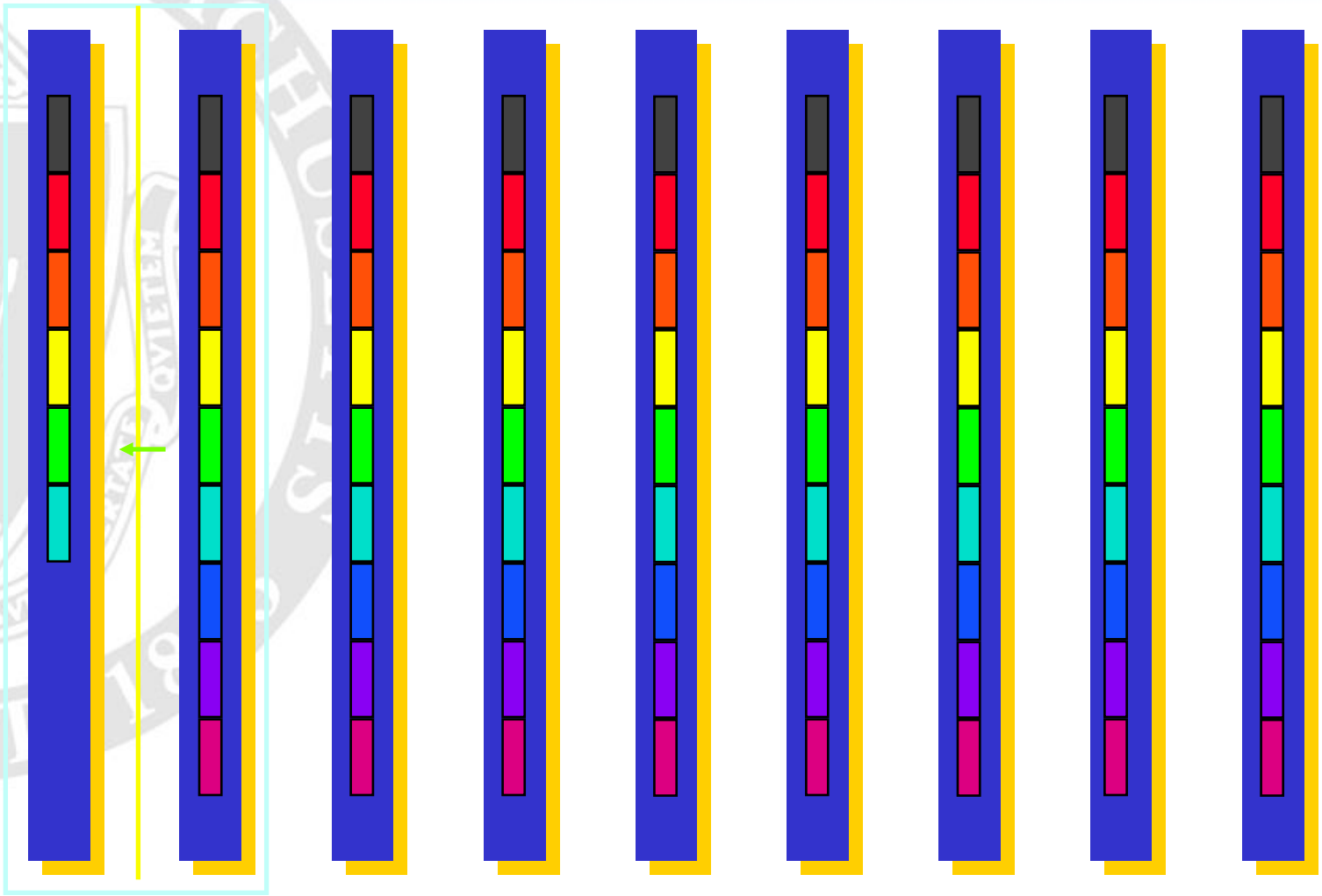


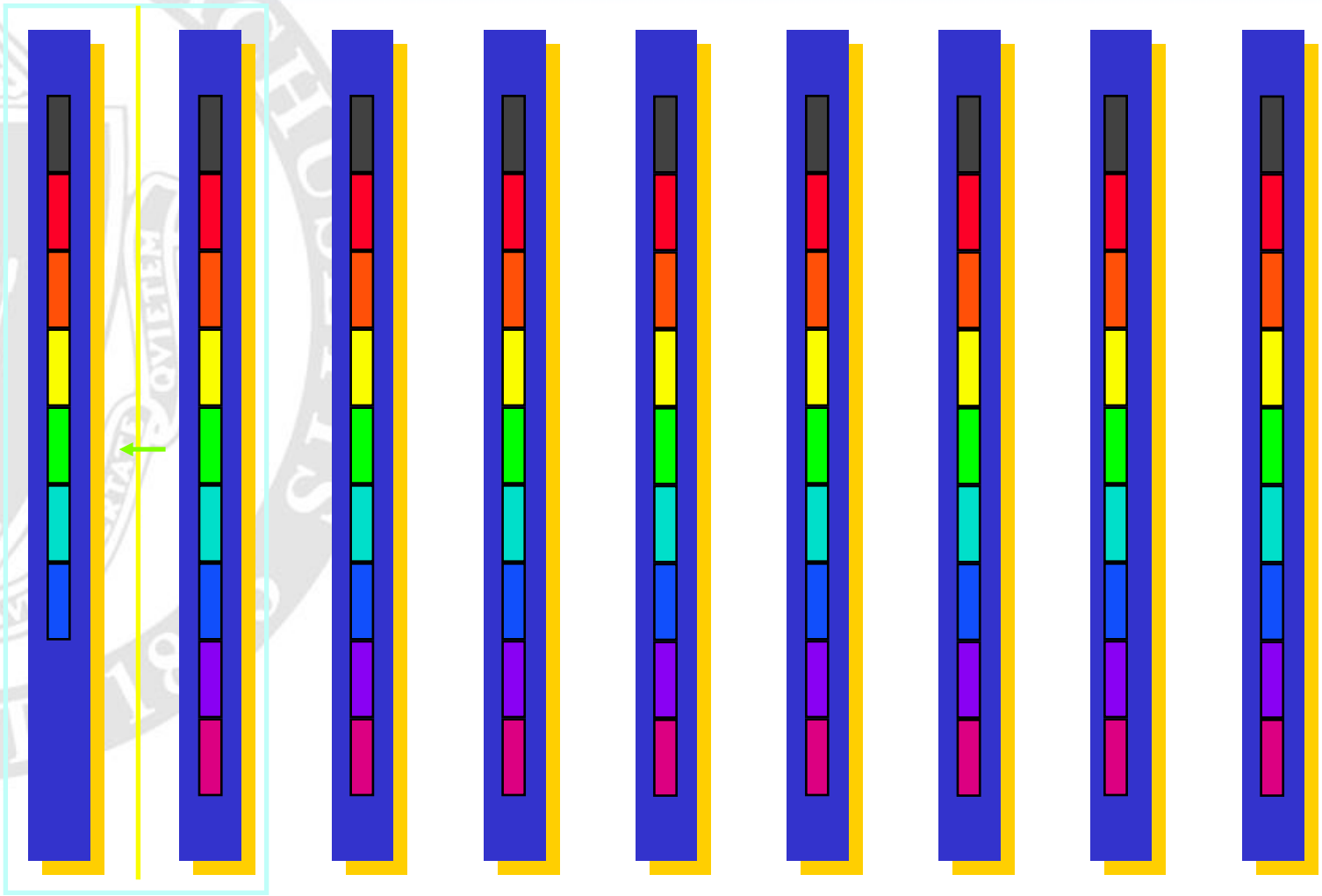


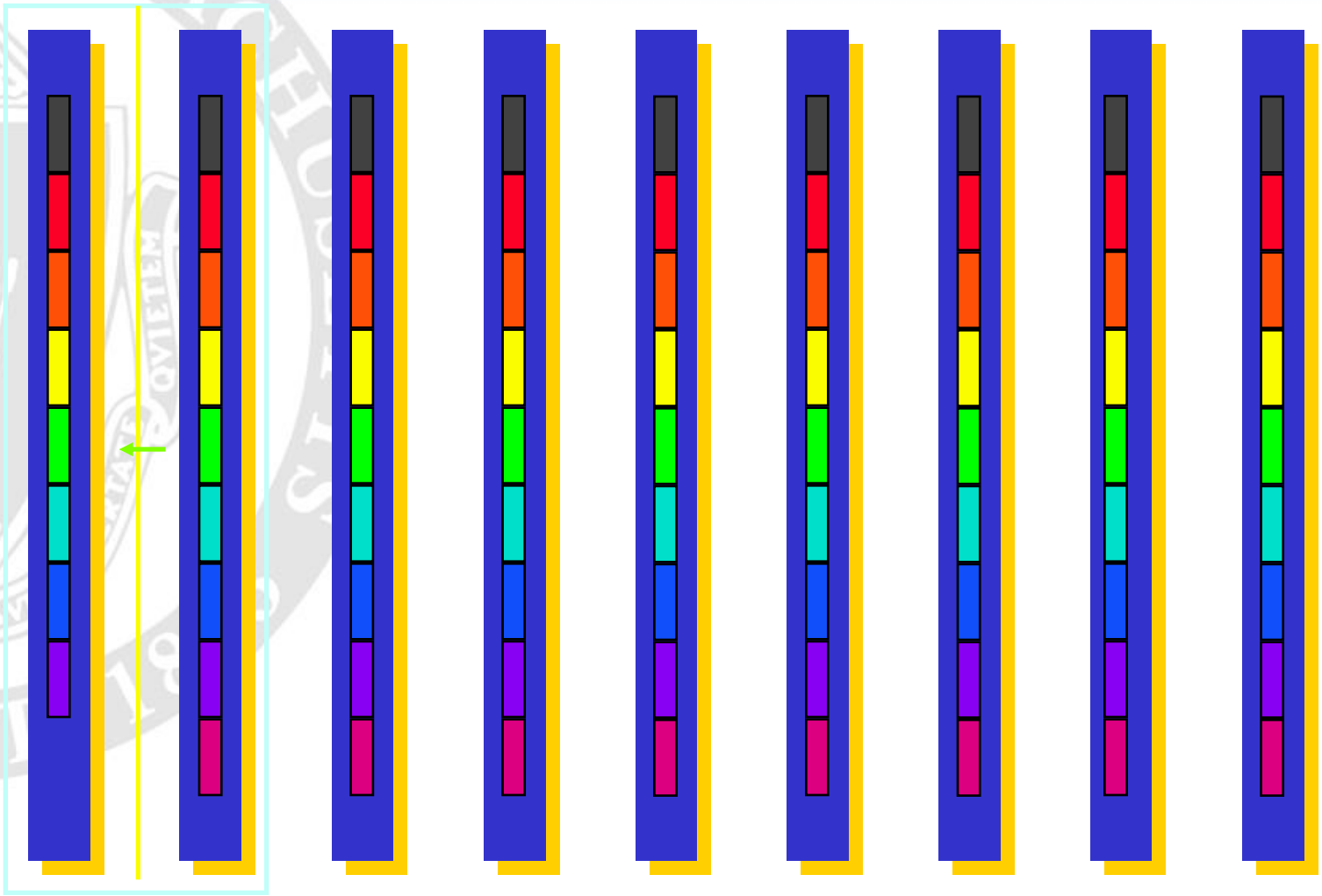


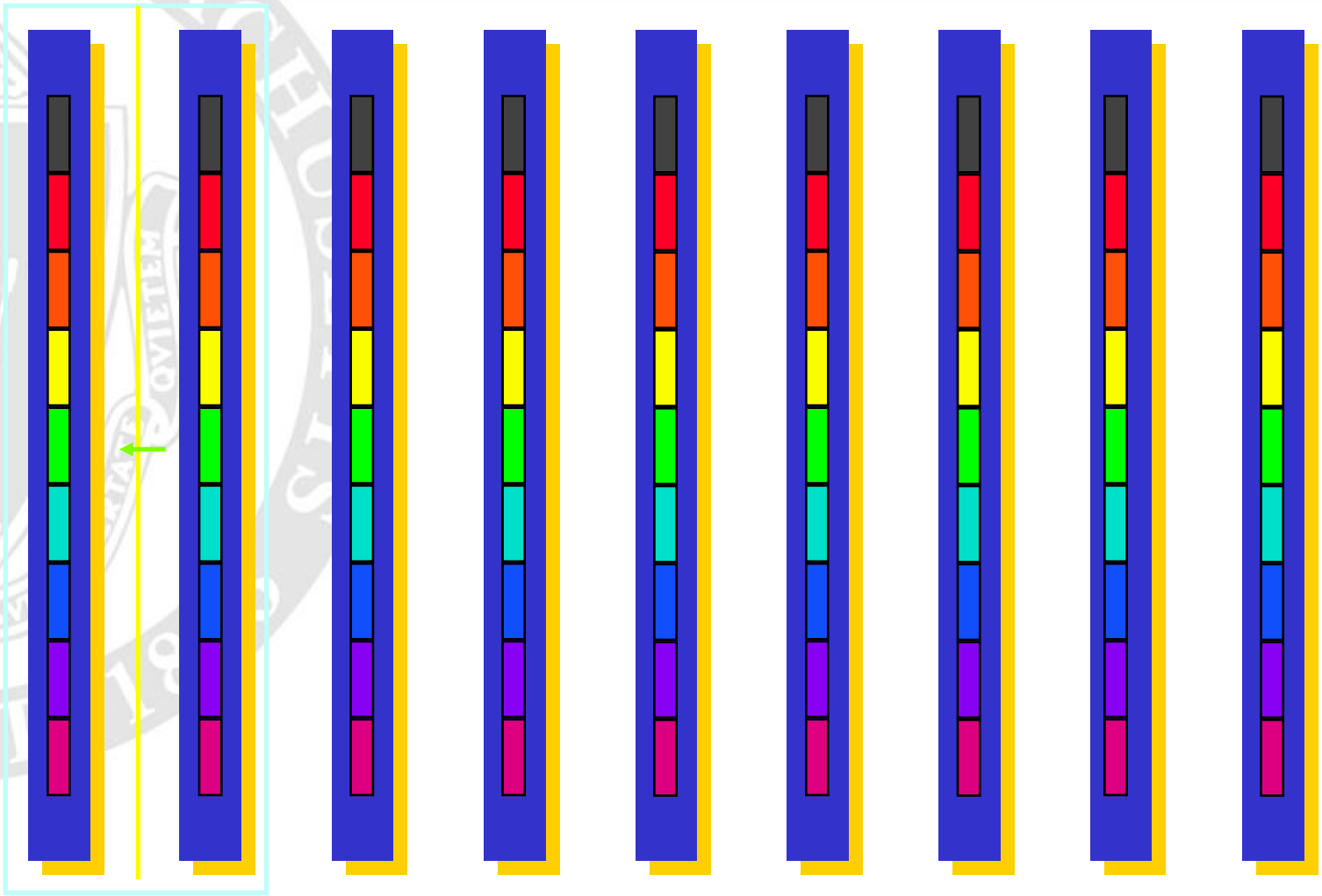


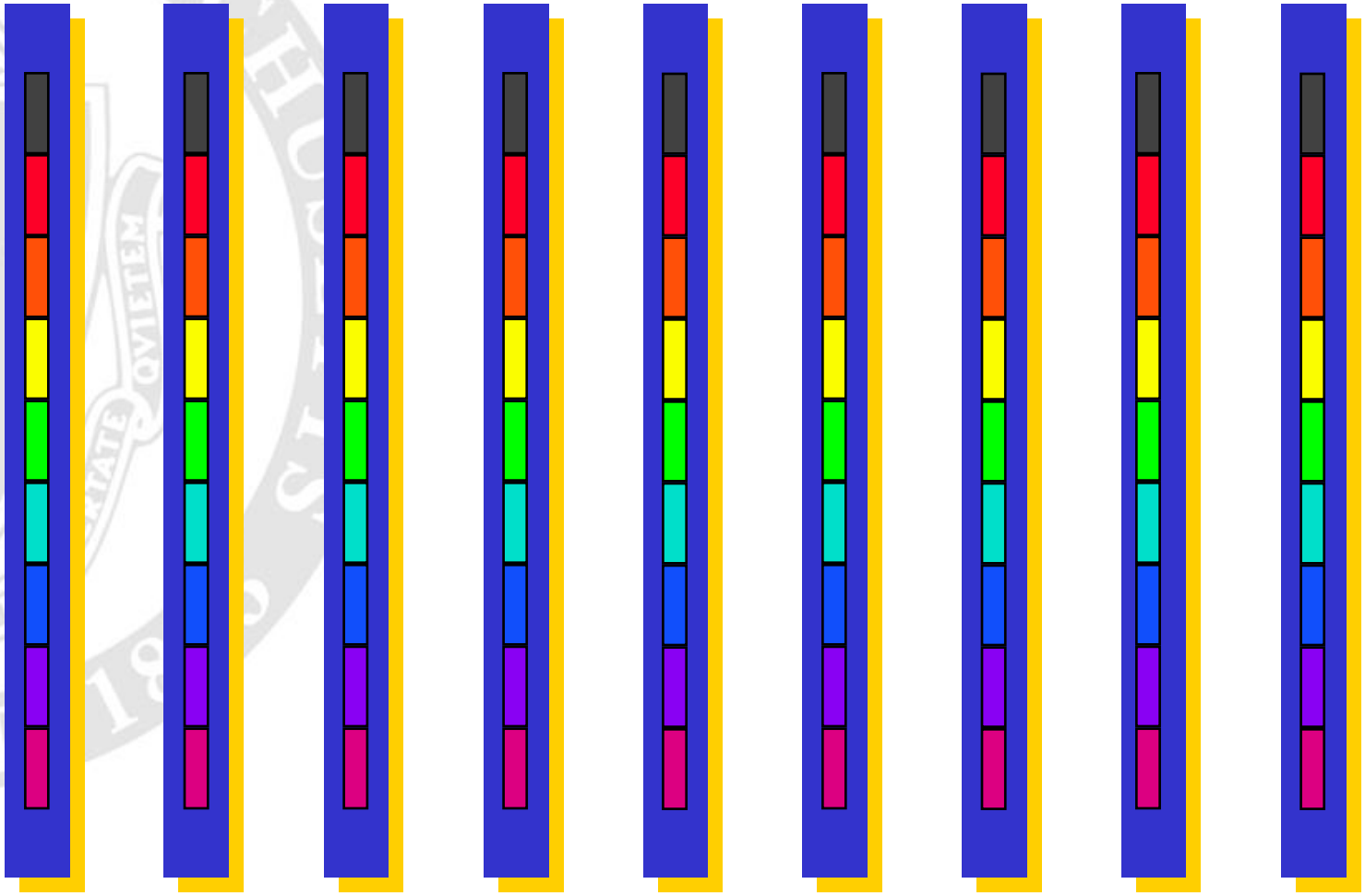






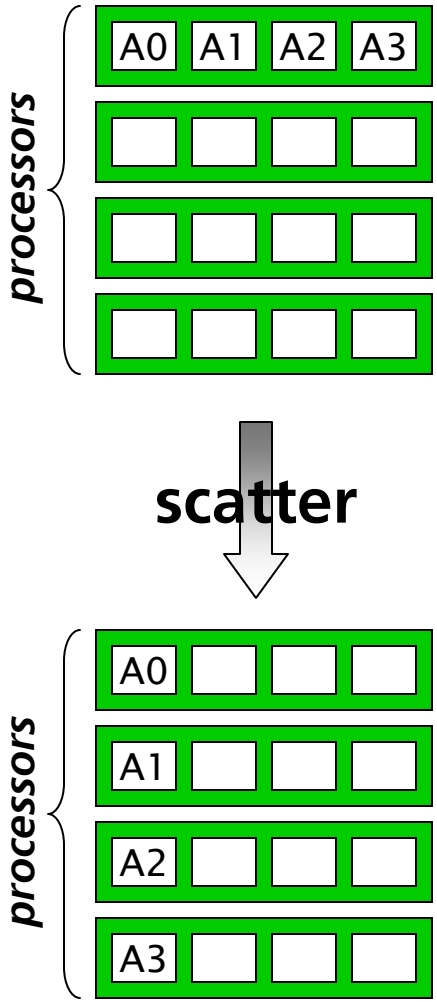




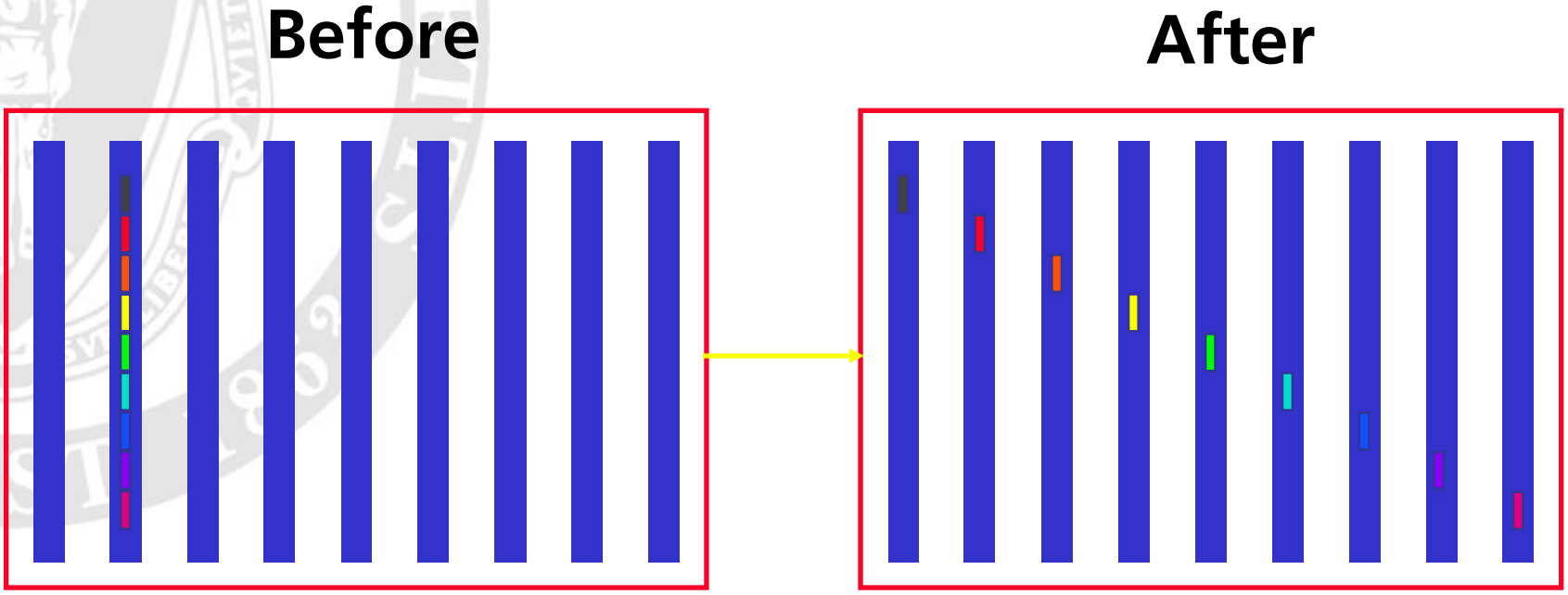


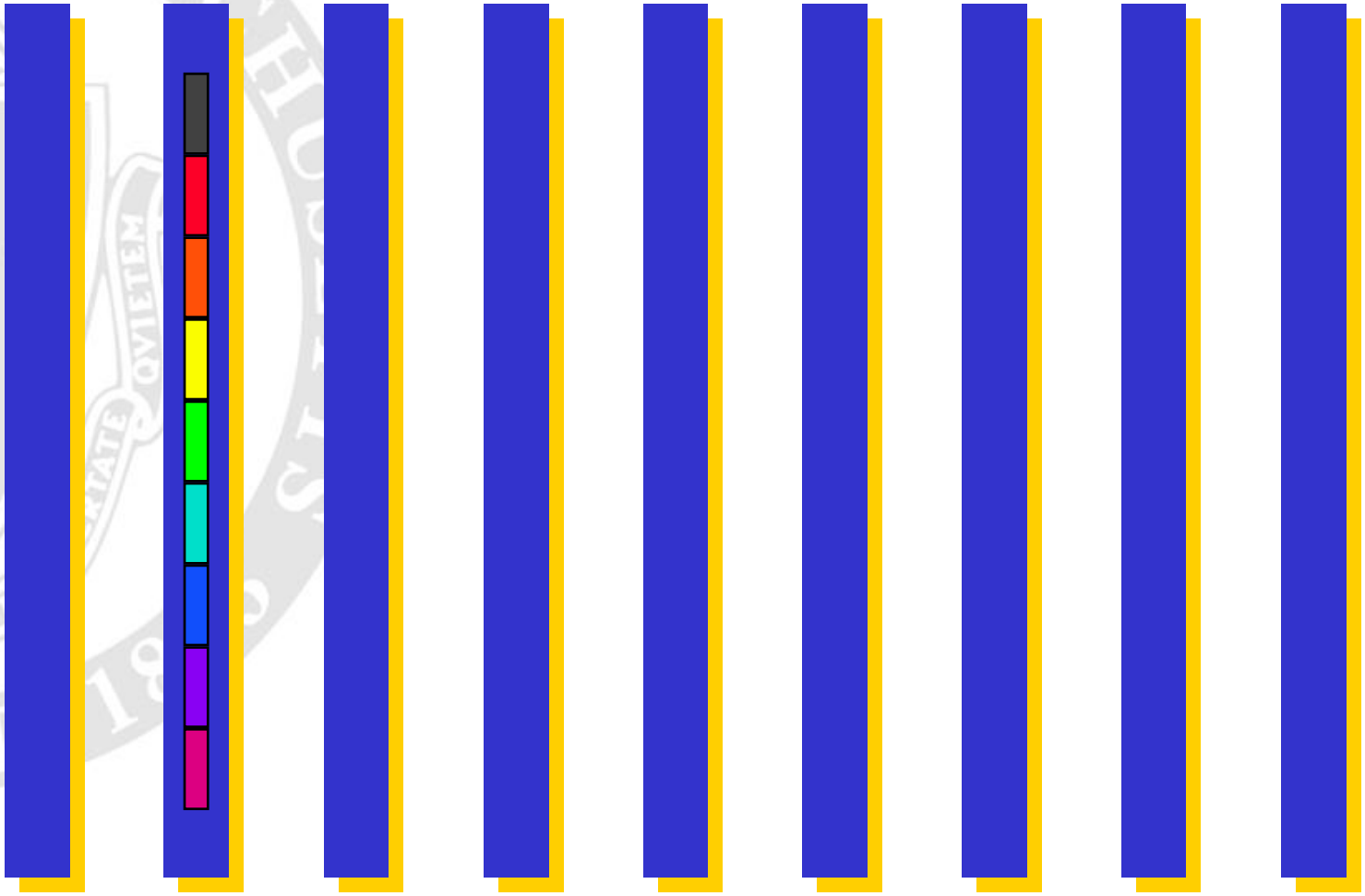
From root: Scatter

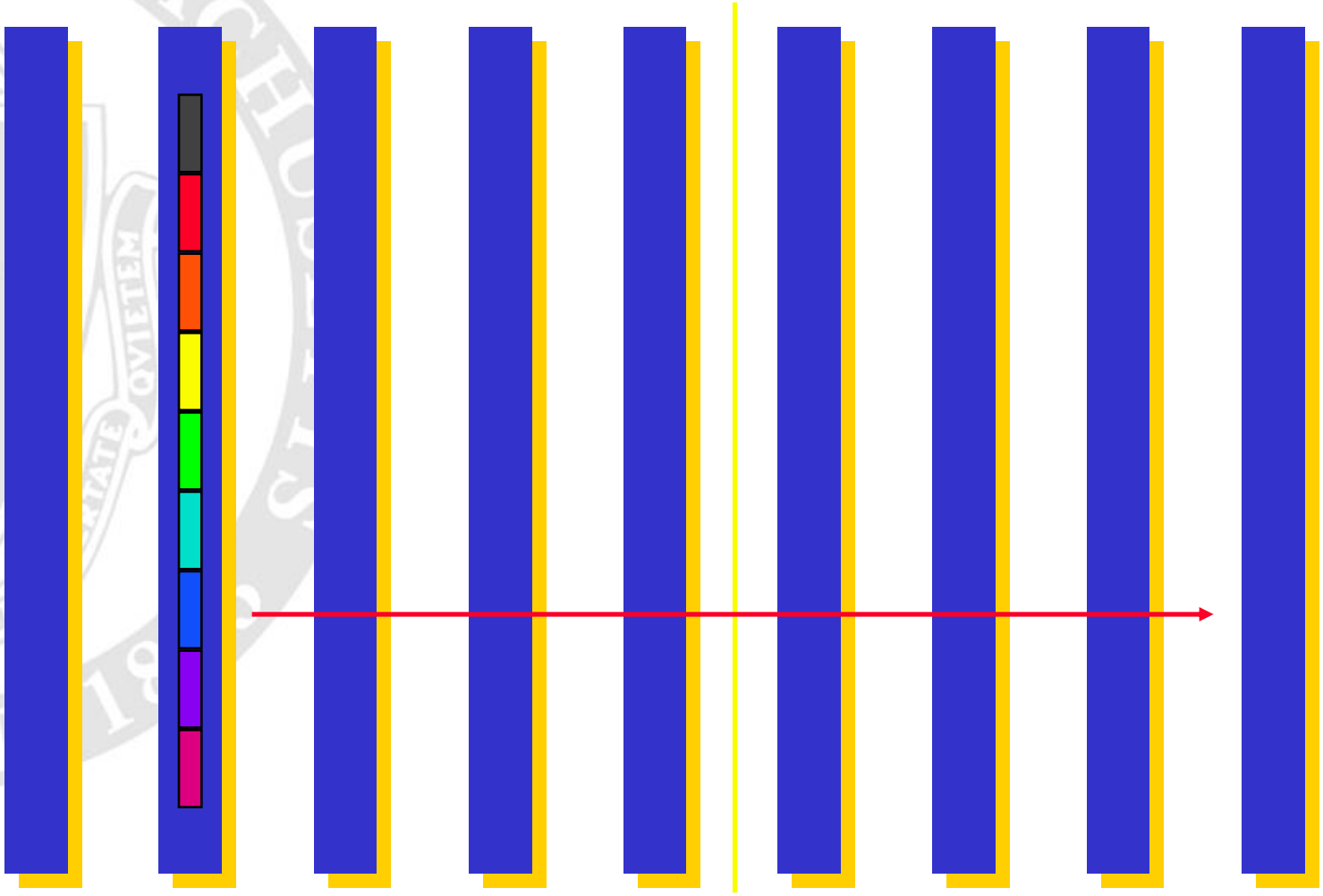
- **MPI_Scatter**: spread data across all processors
- *variant* – **MPI_Scatterv**: scatters buffer *in parts* (specified in array)

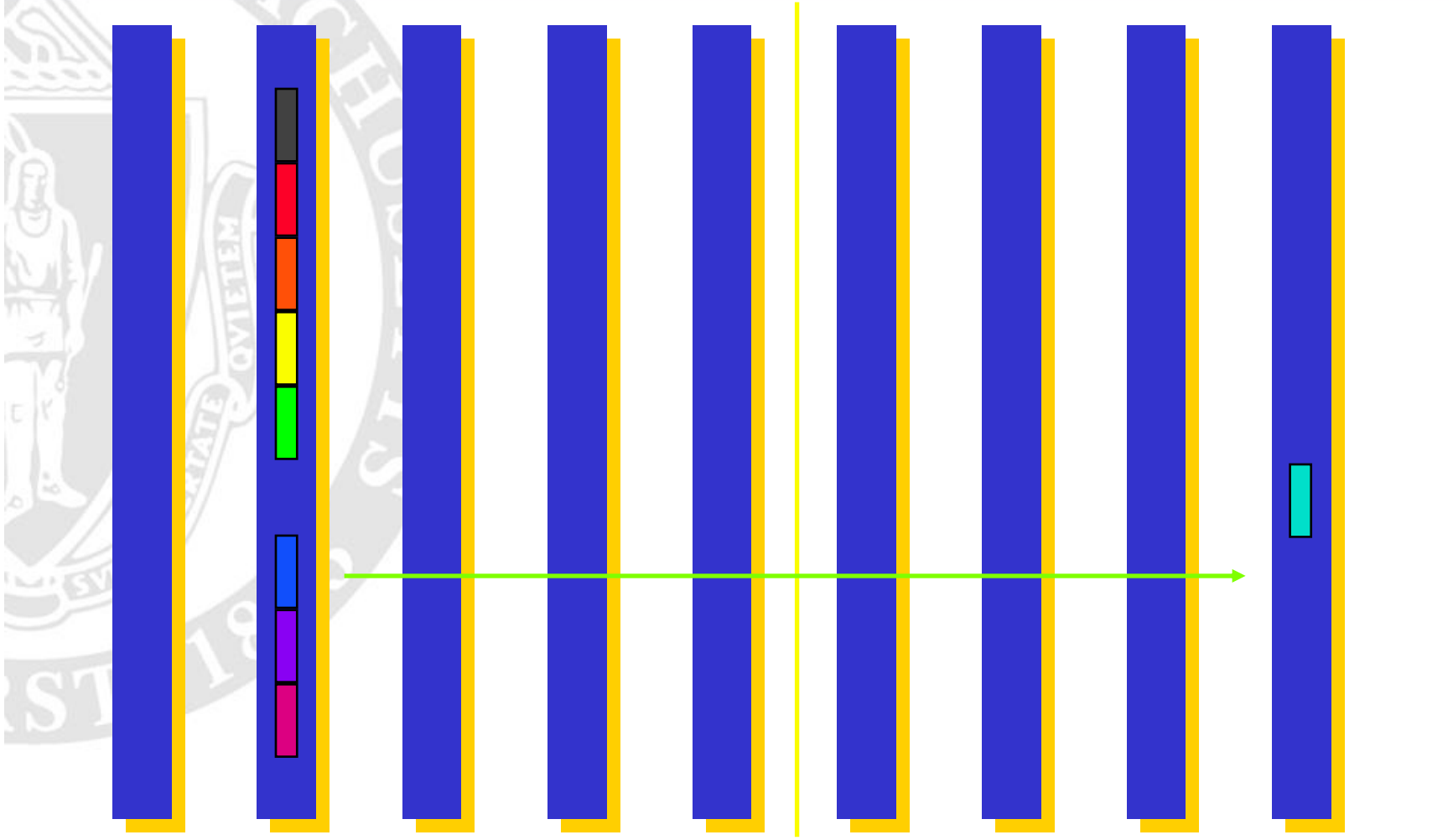


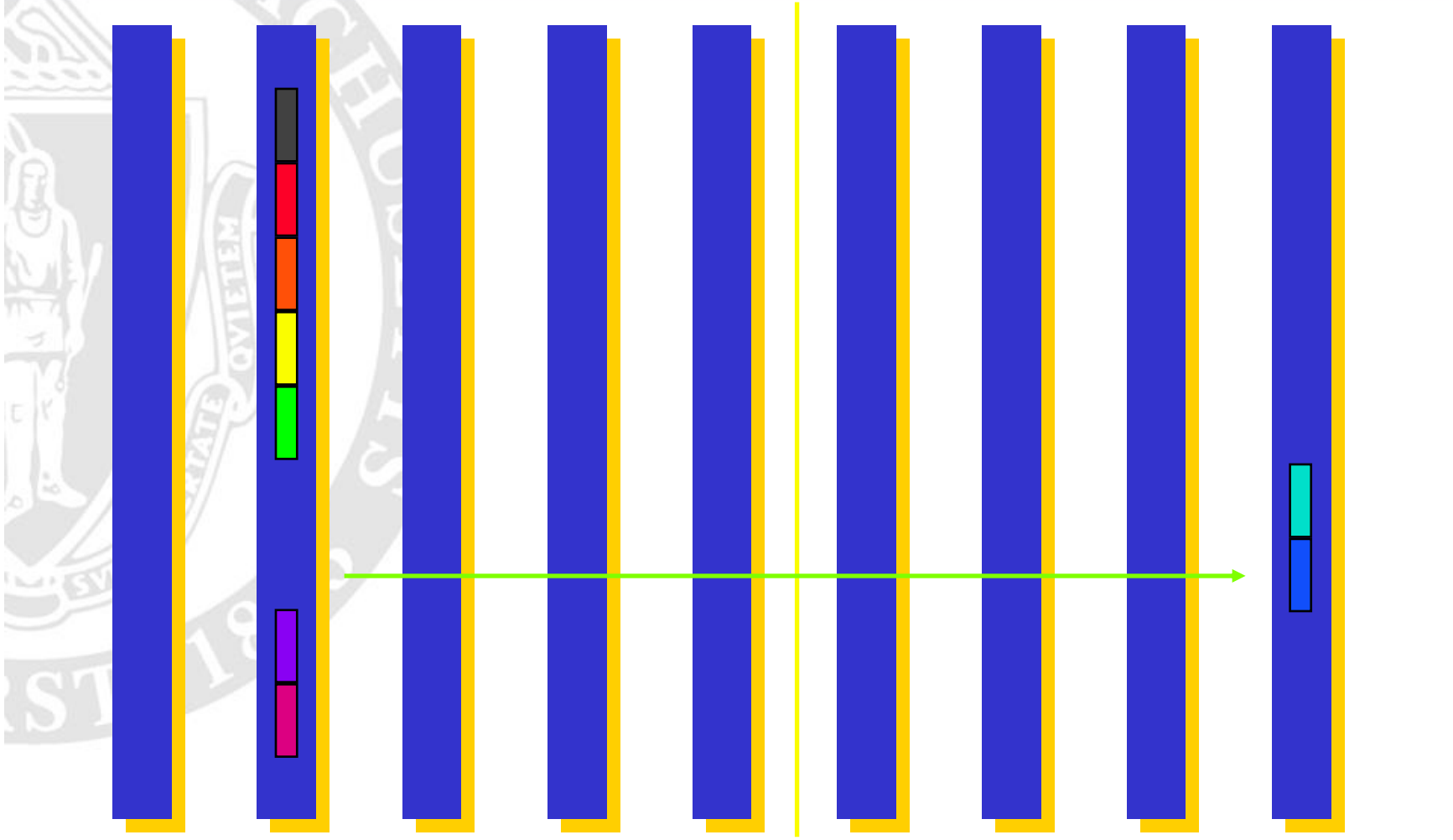
Scatter

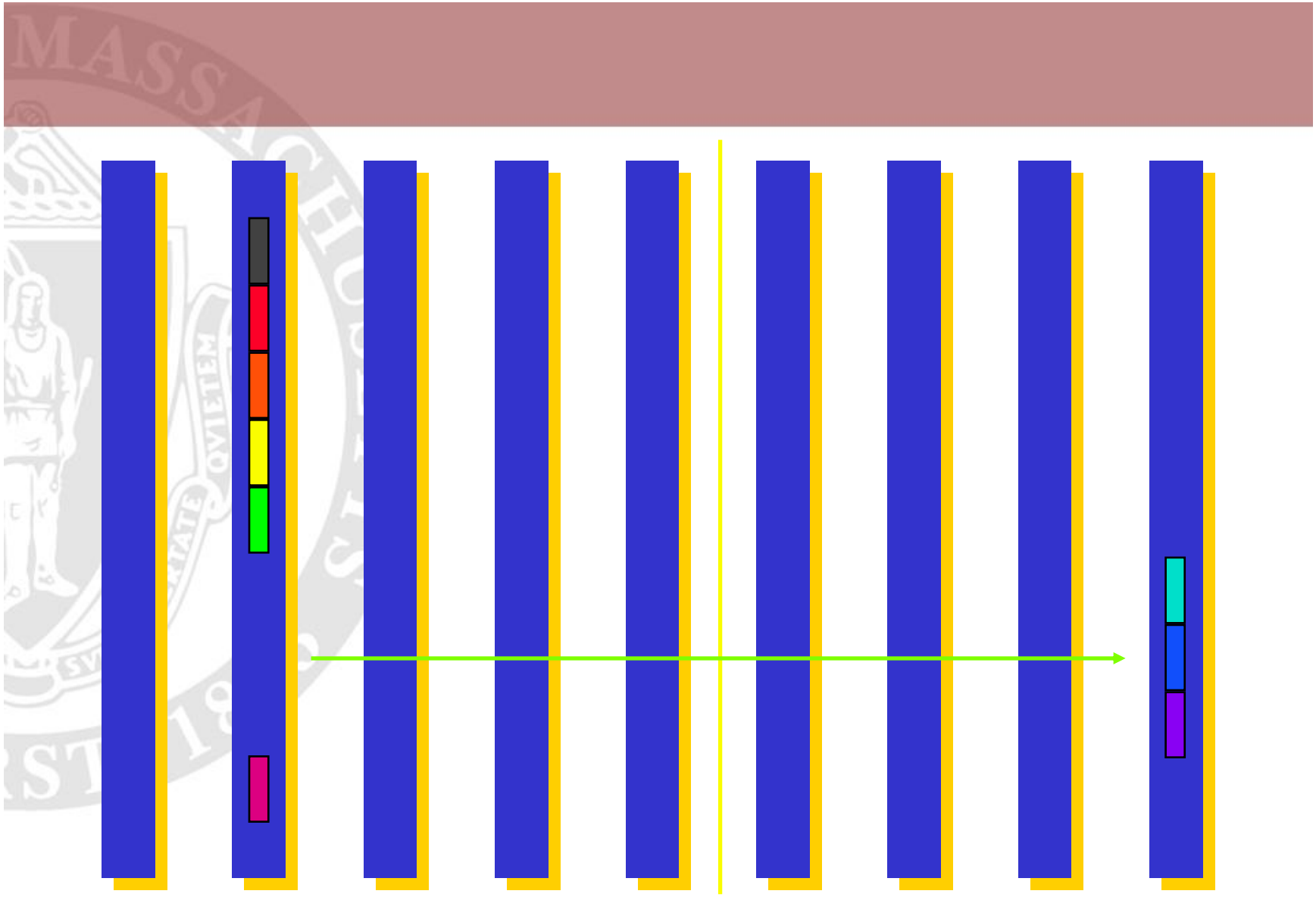


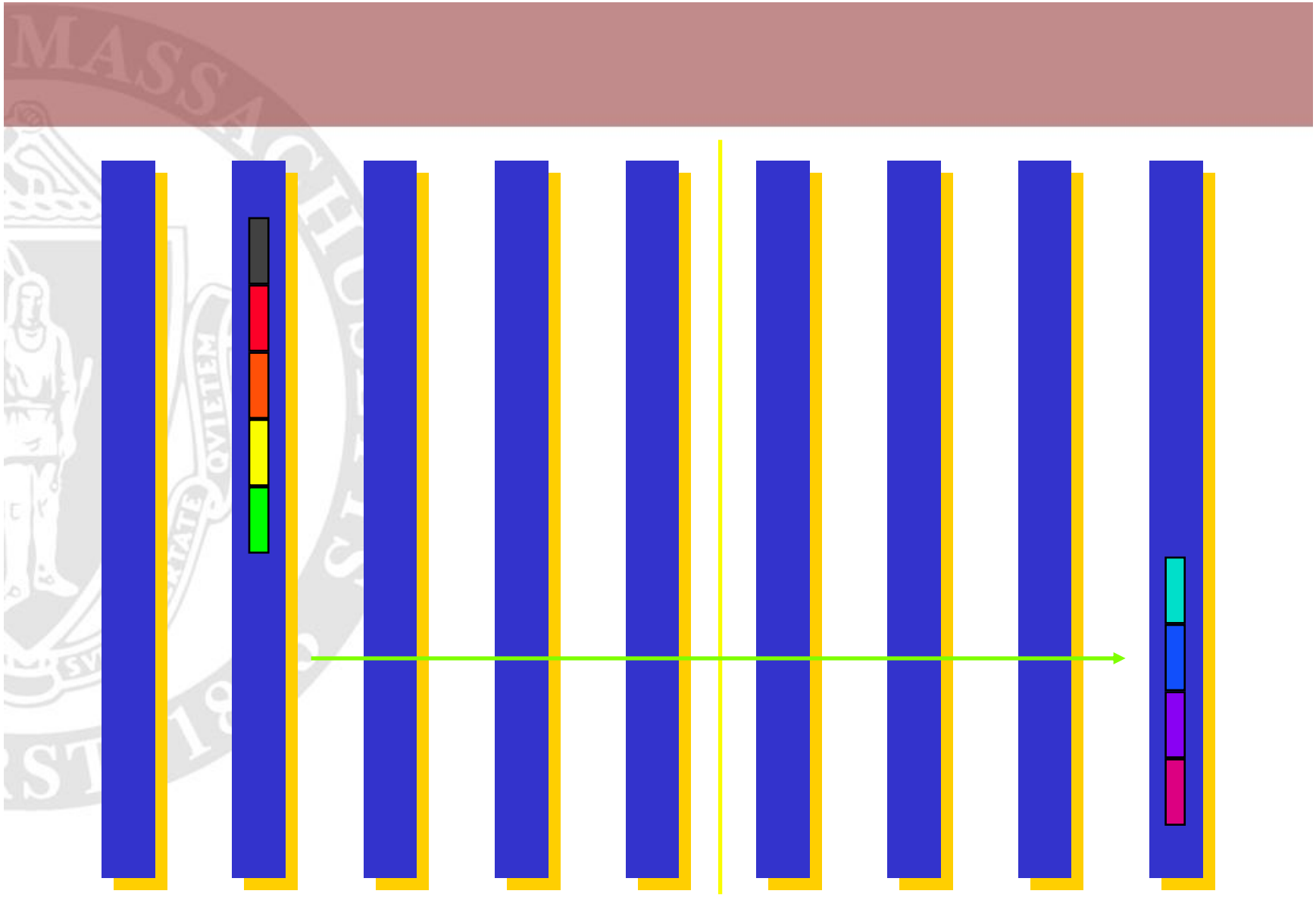


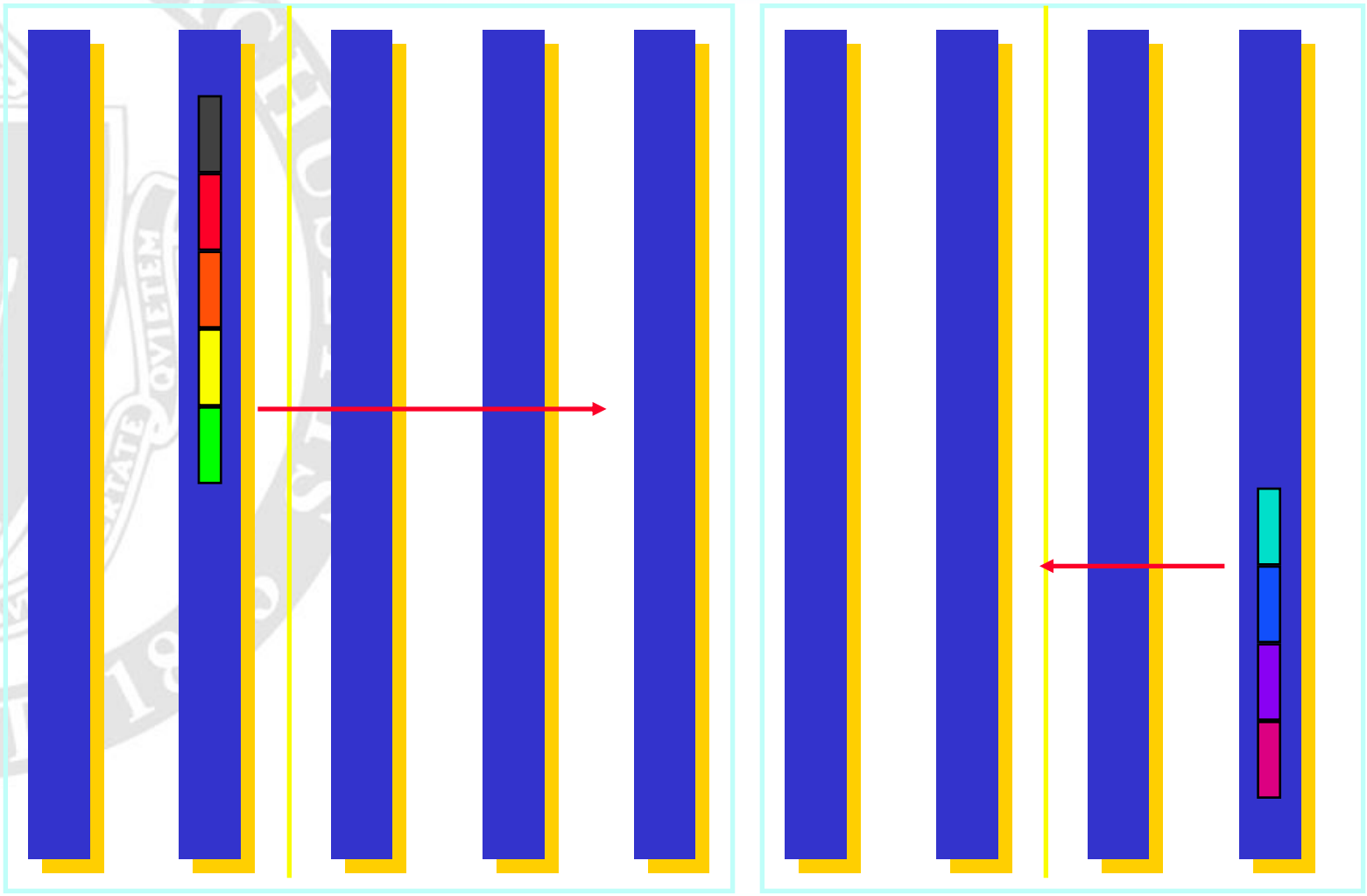


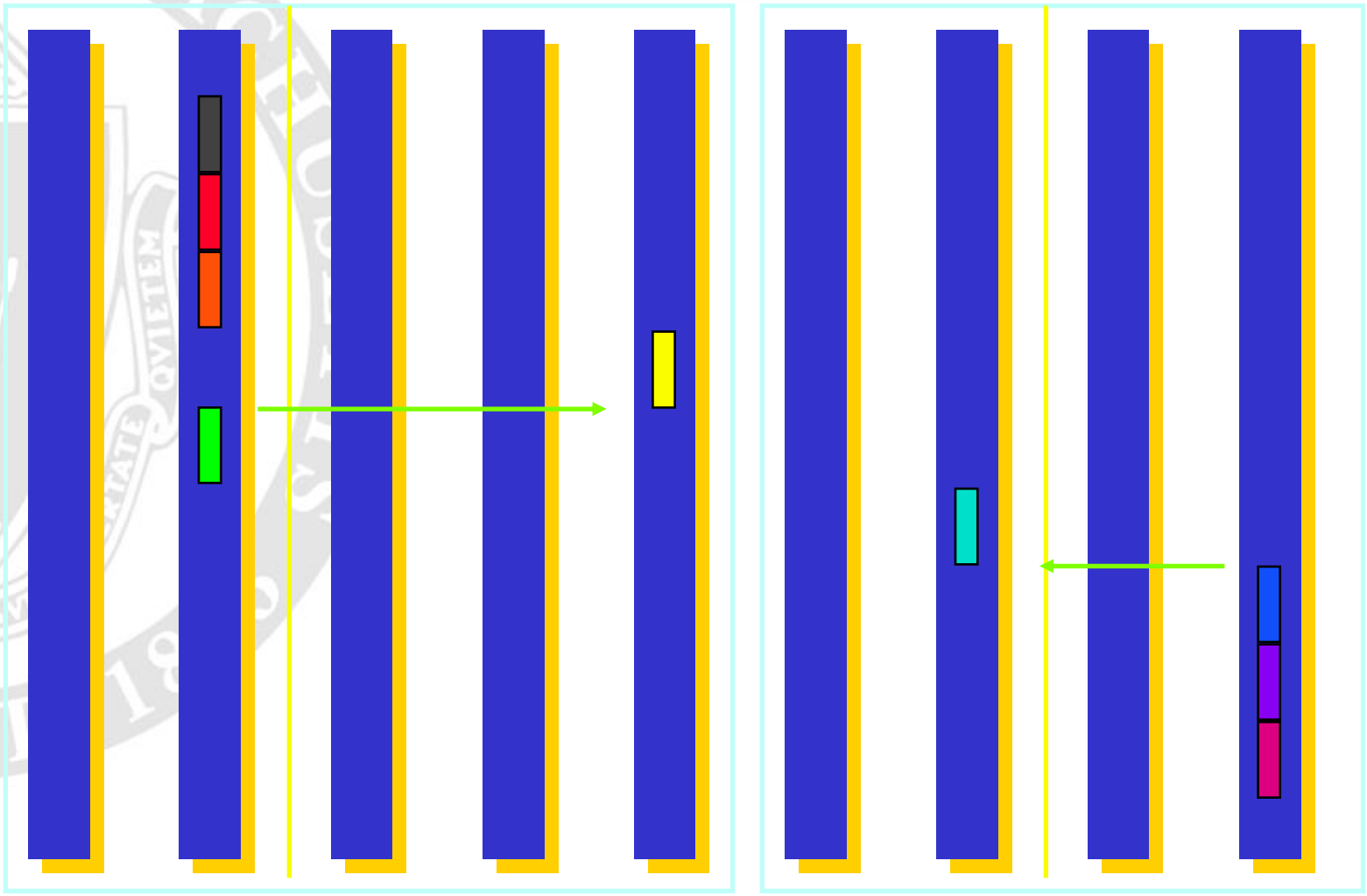


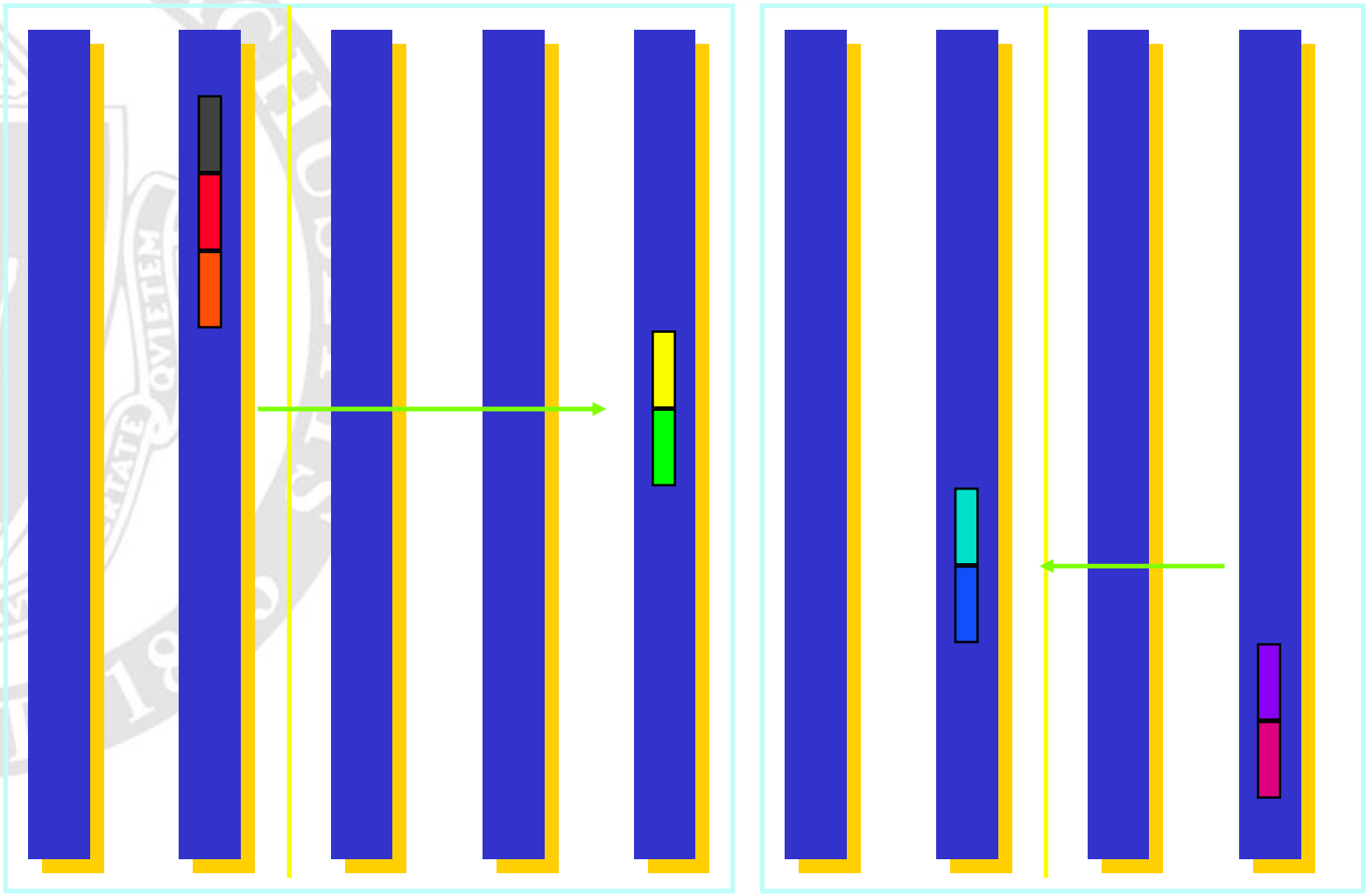


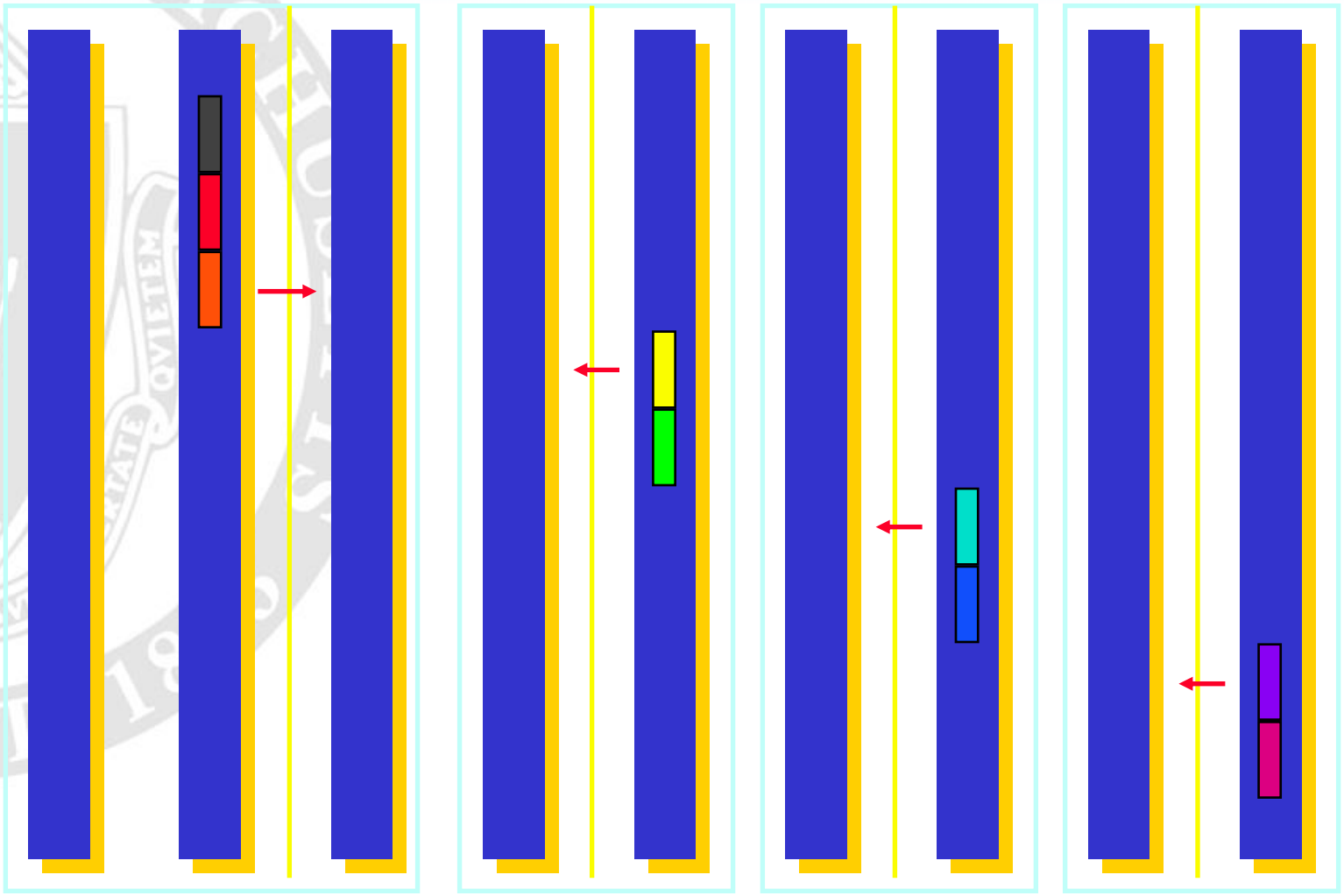


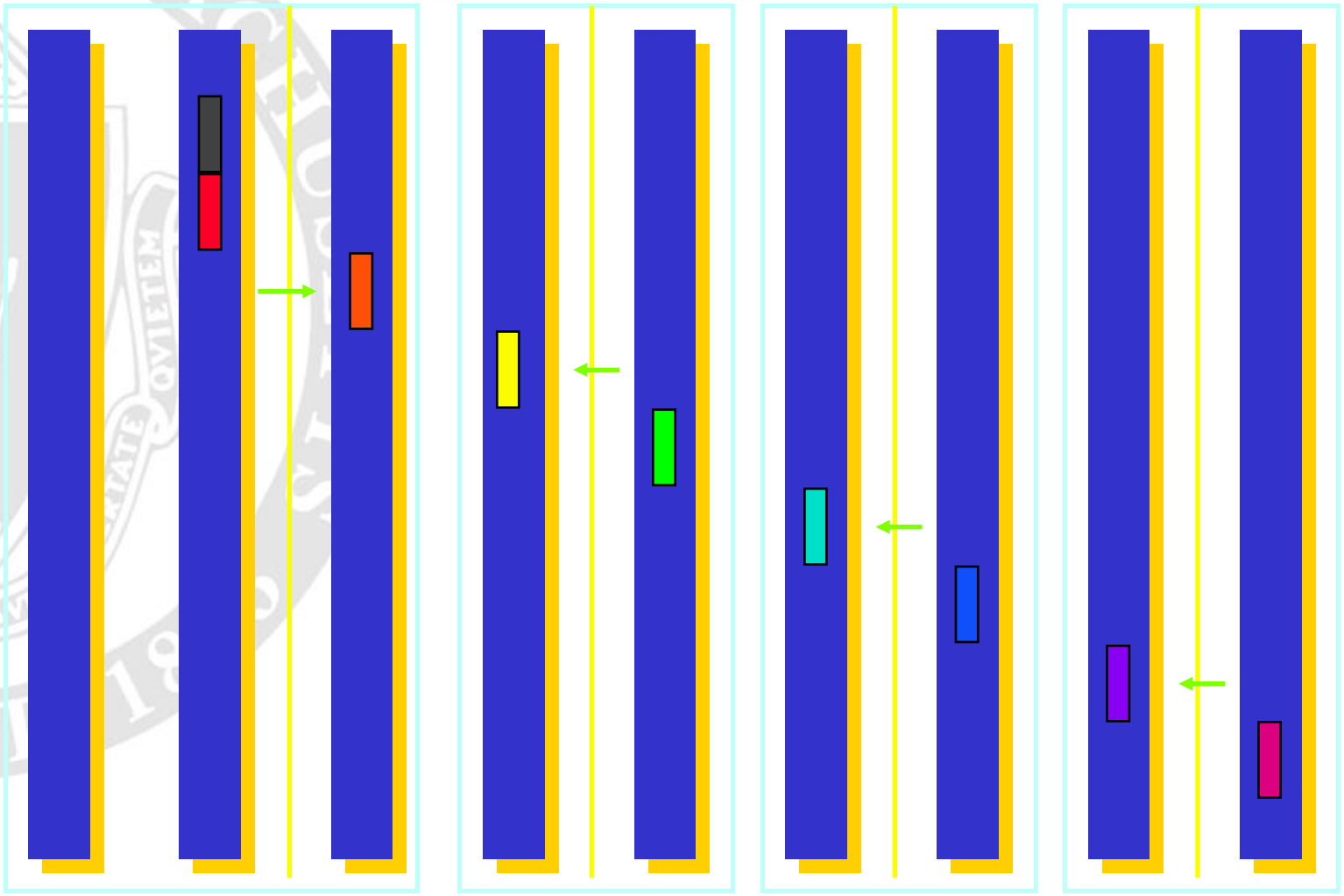


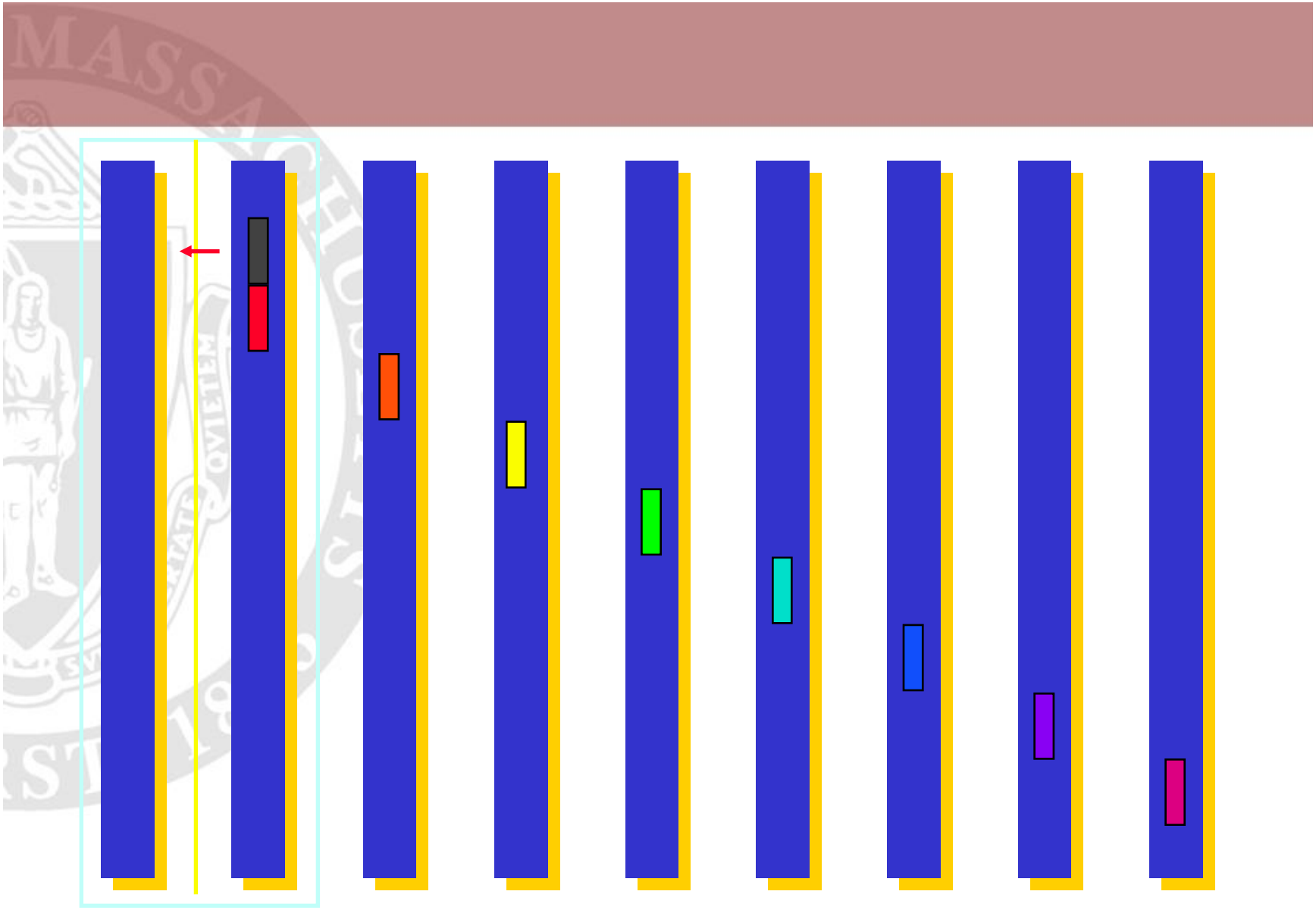


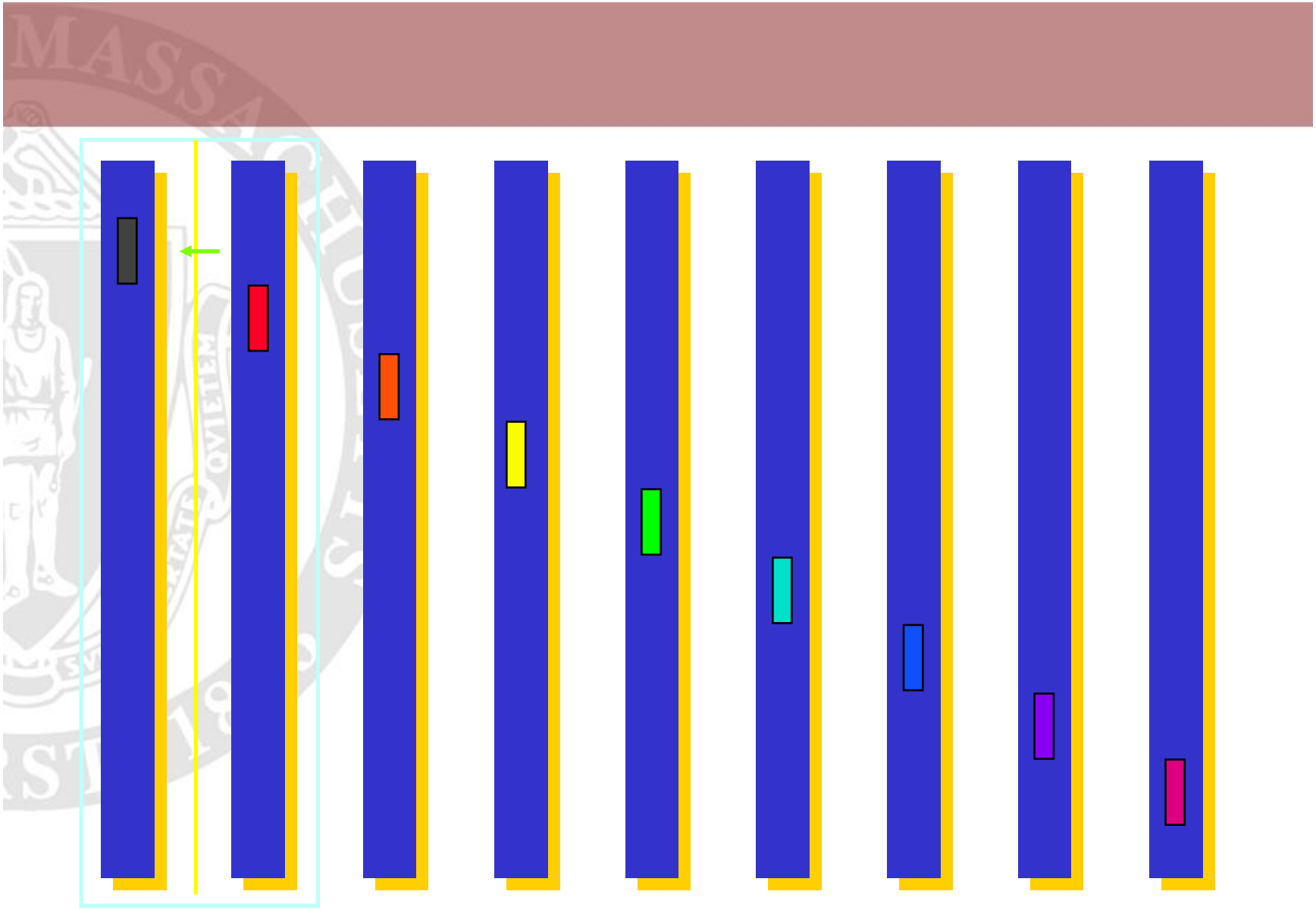


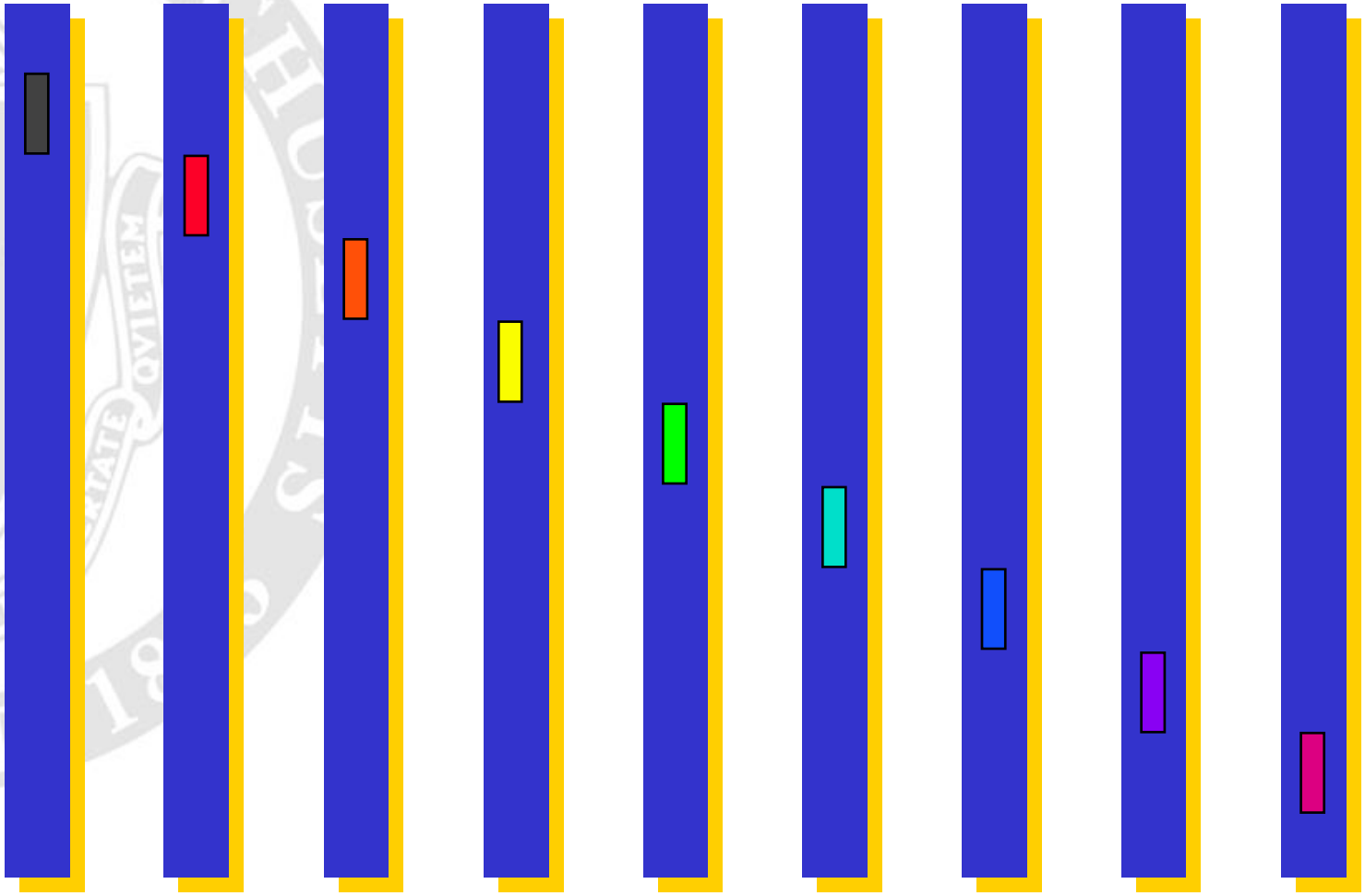






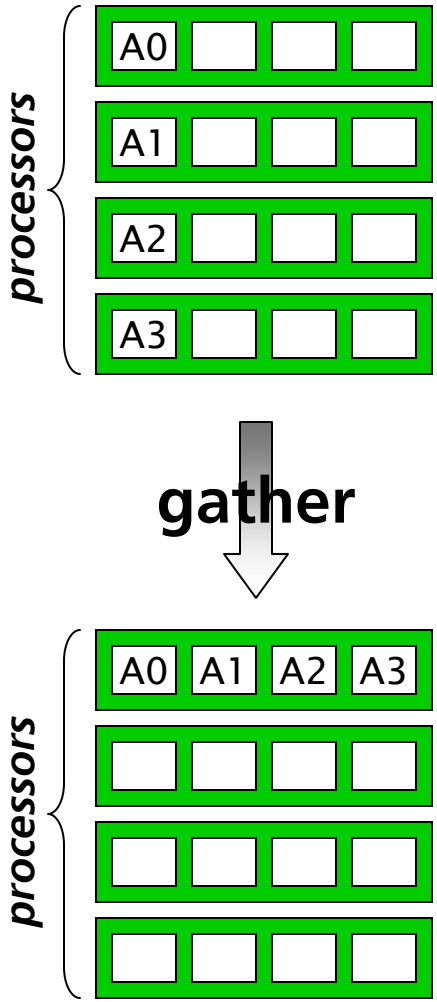


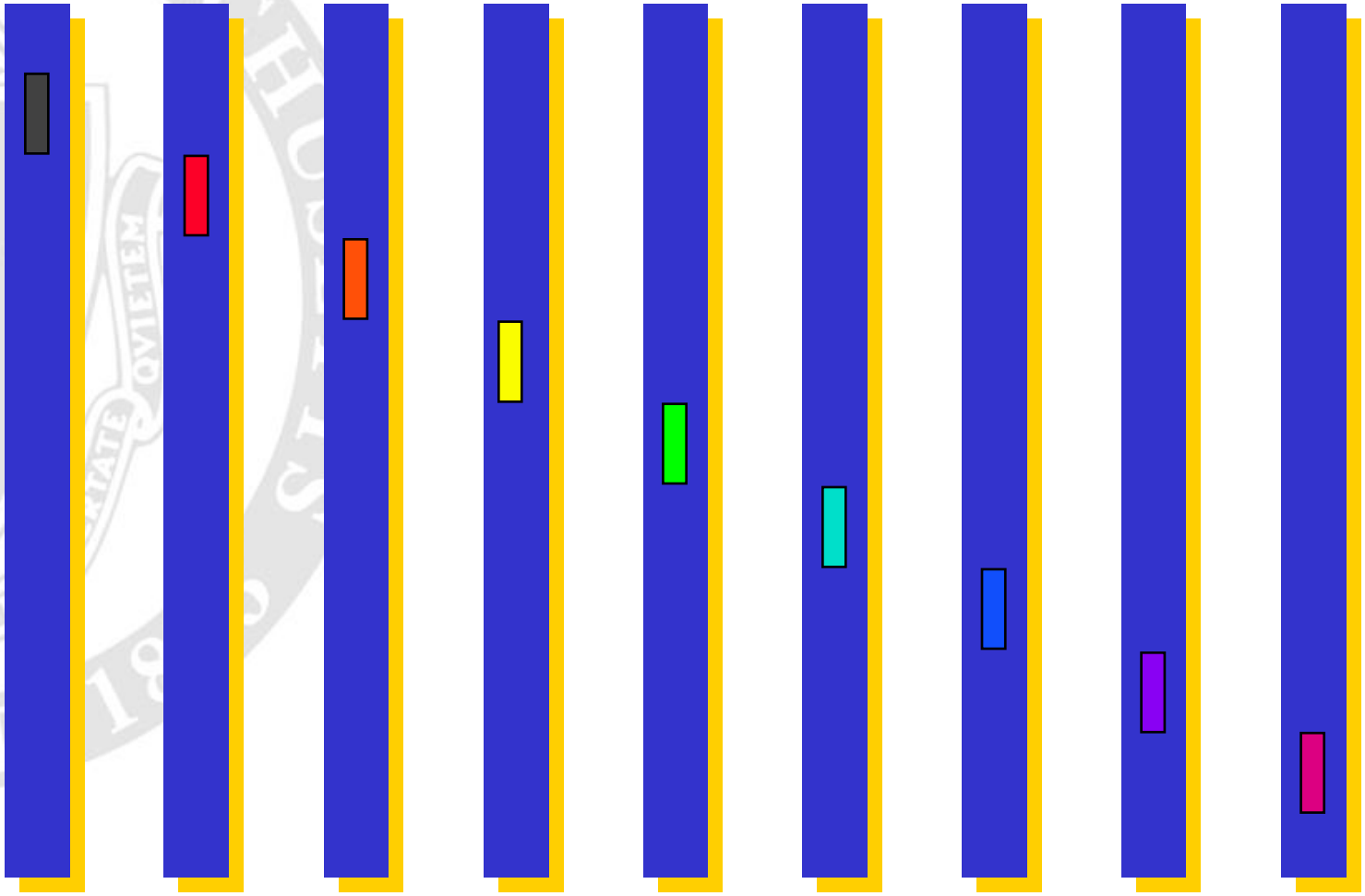


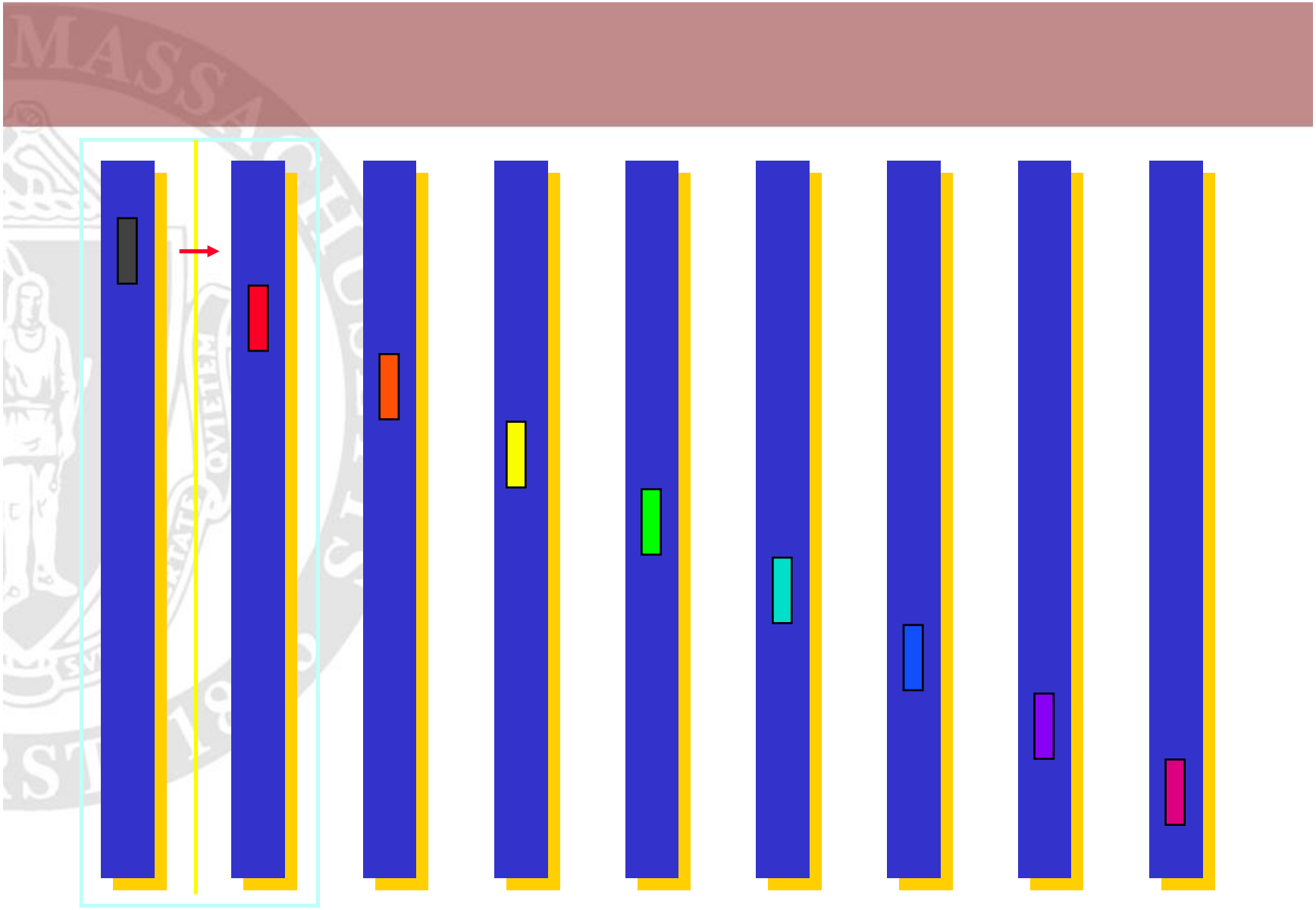


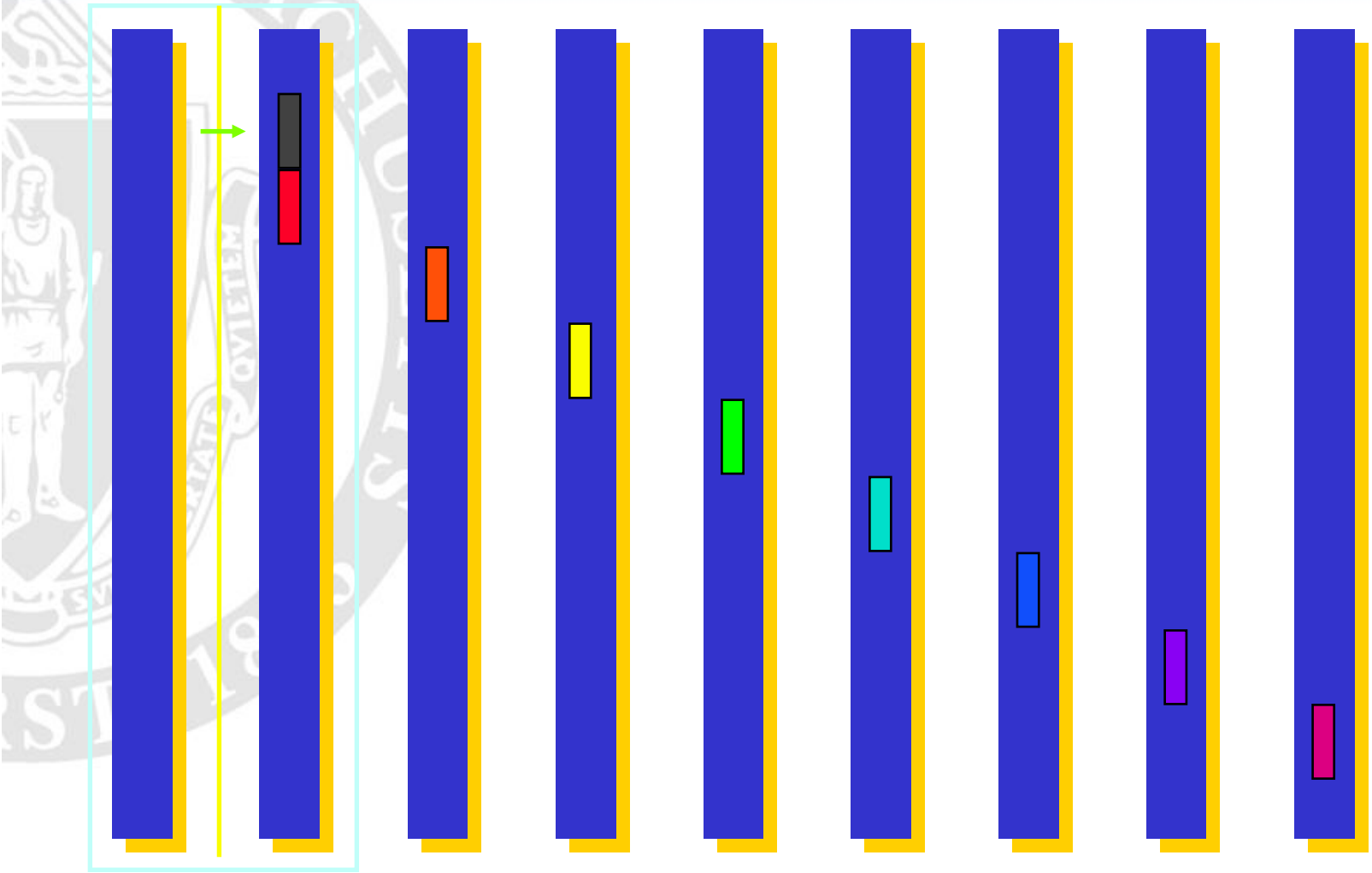
To root: Gather

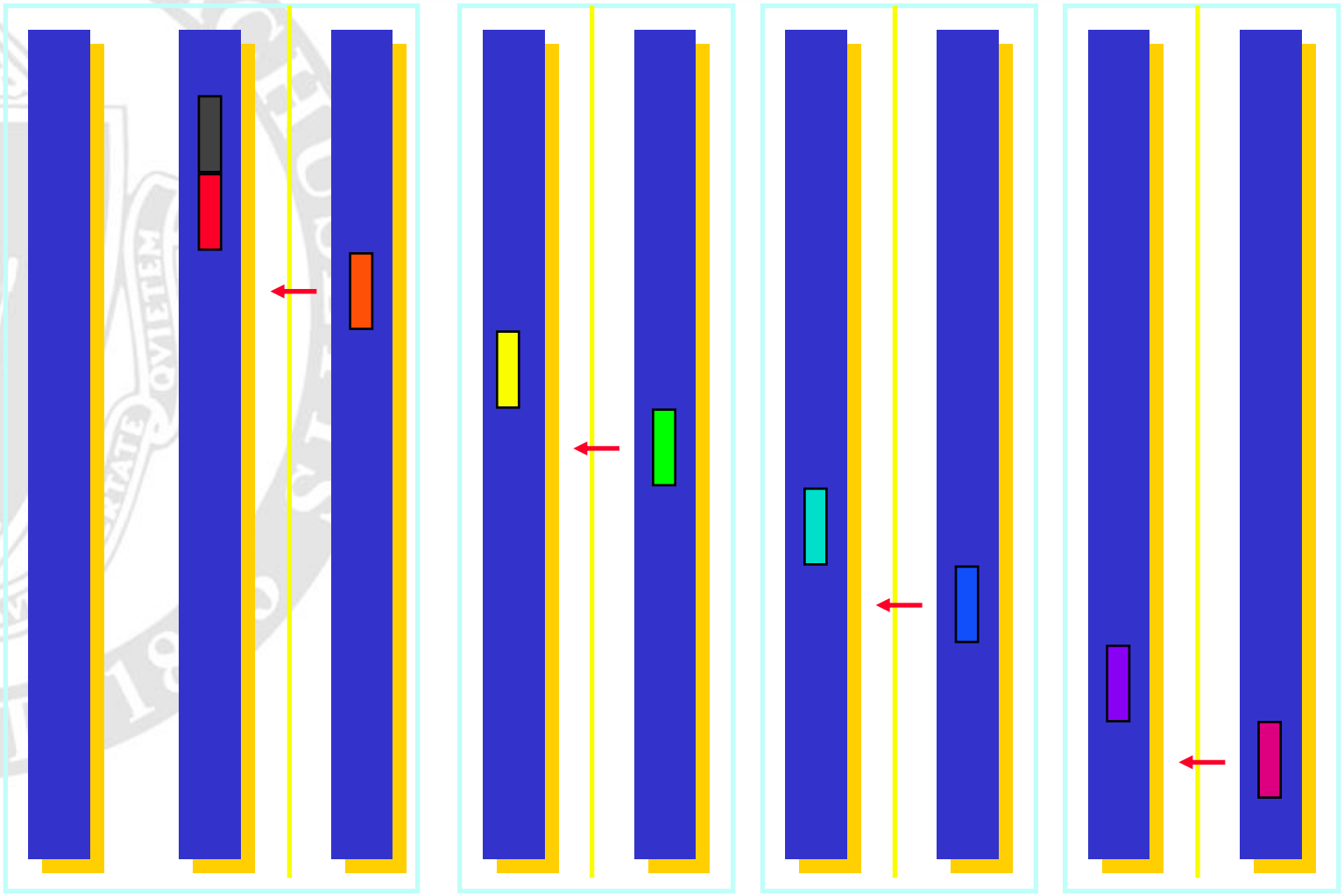
- **MPI_Gather**: gather data from all processors to one processor
- *variant* – **MPI_Gatherv**

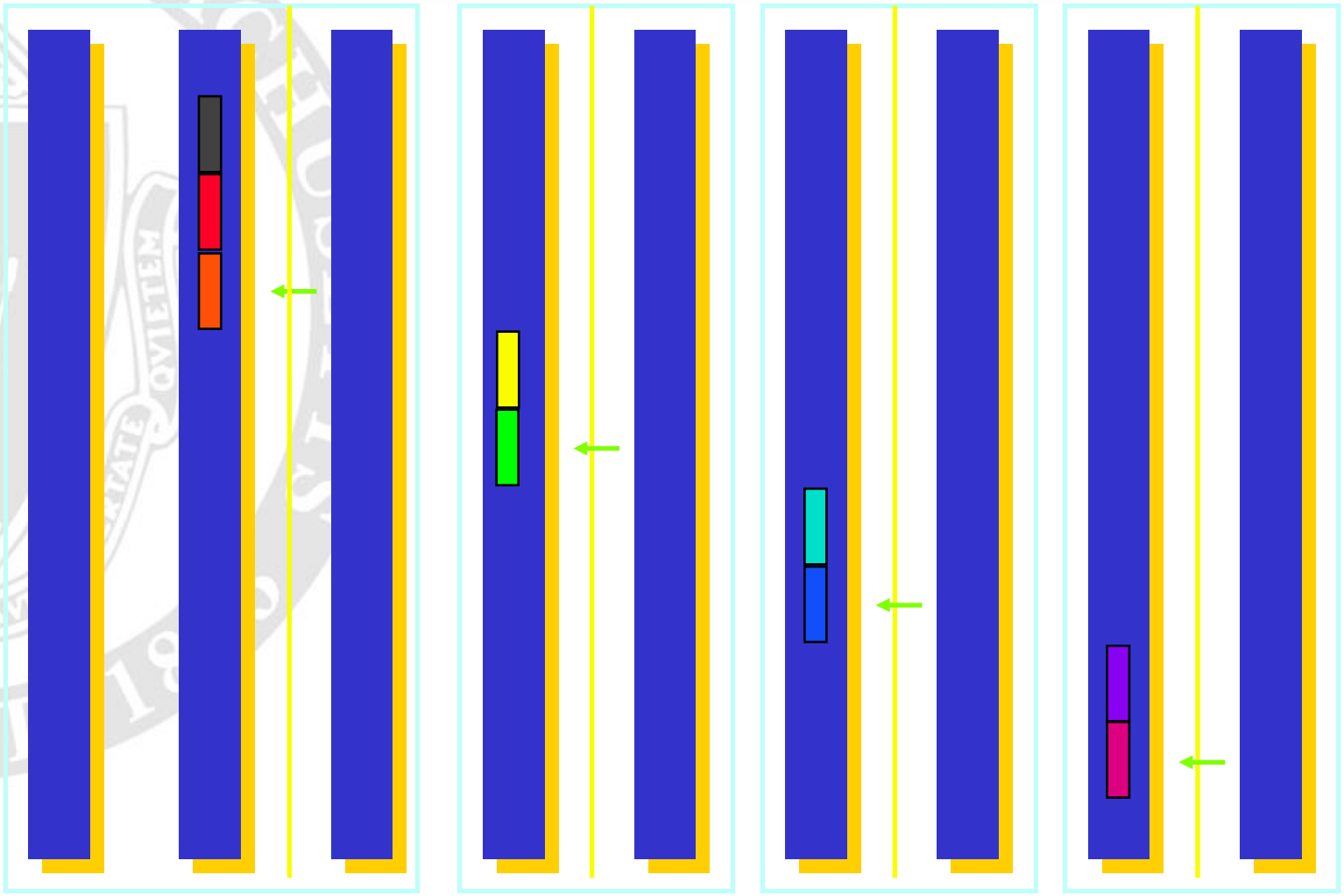


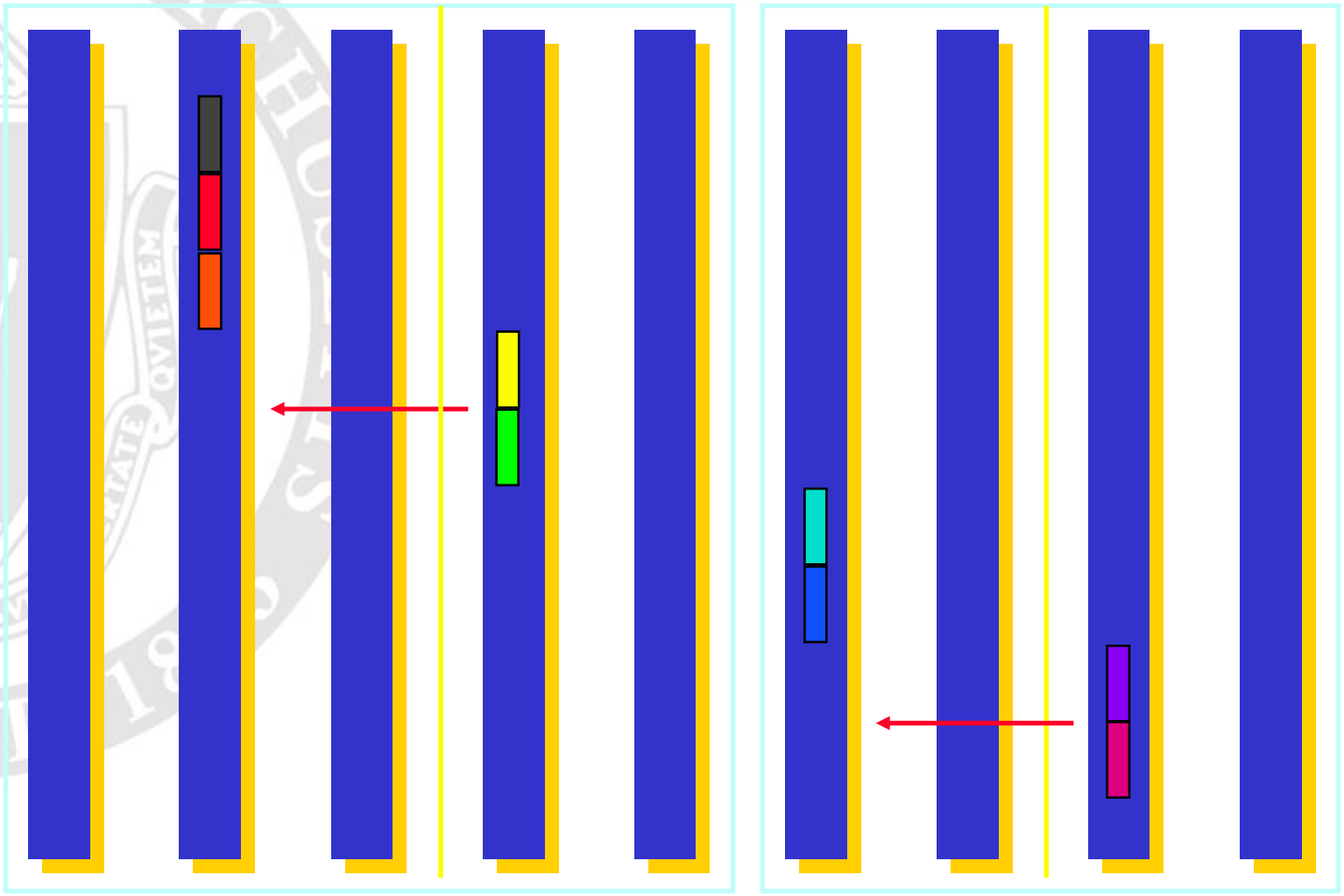


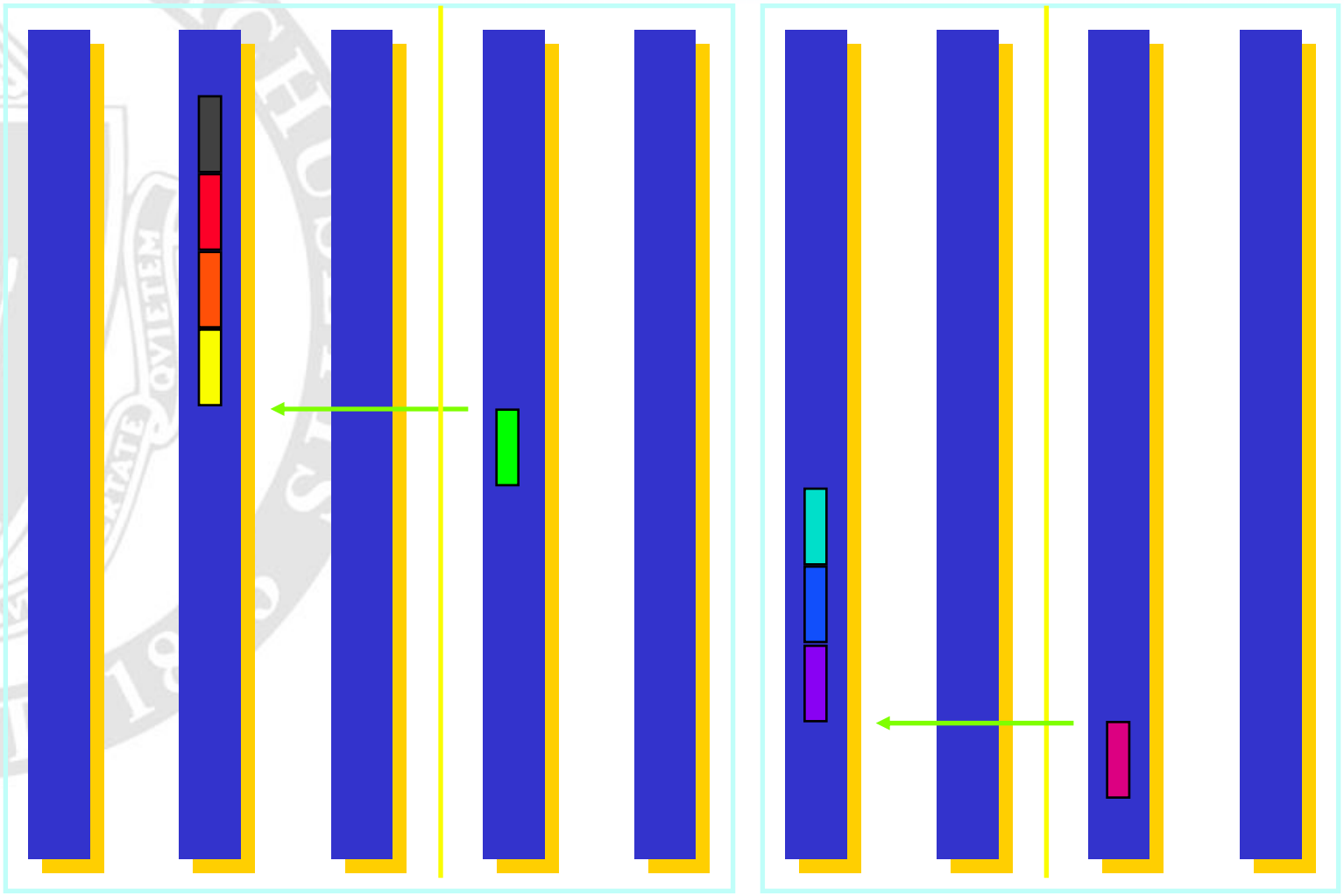


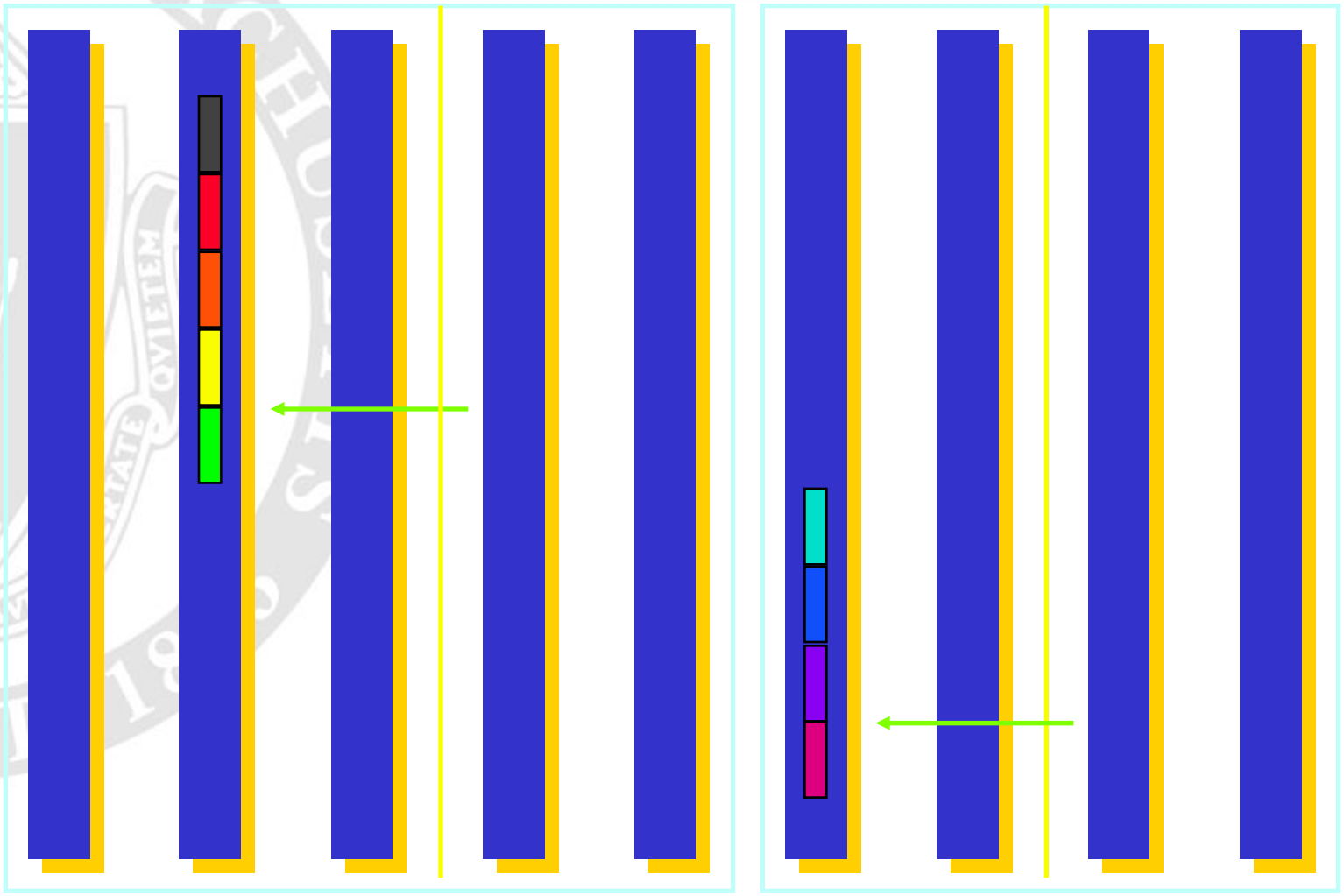


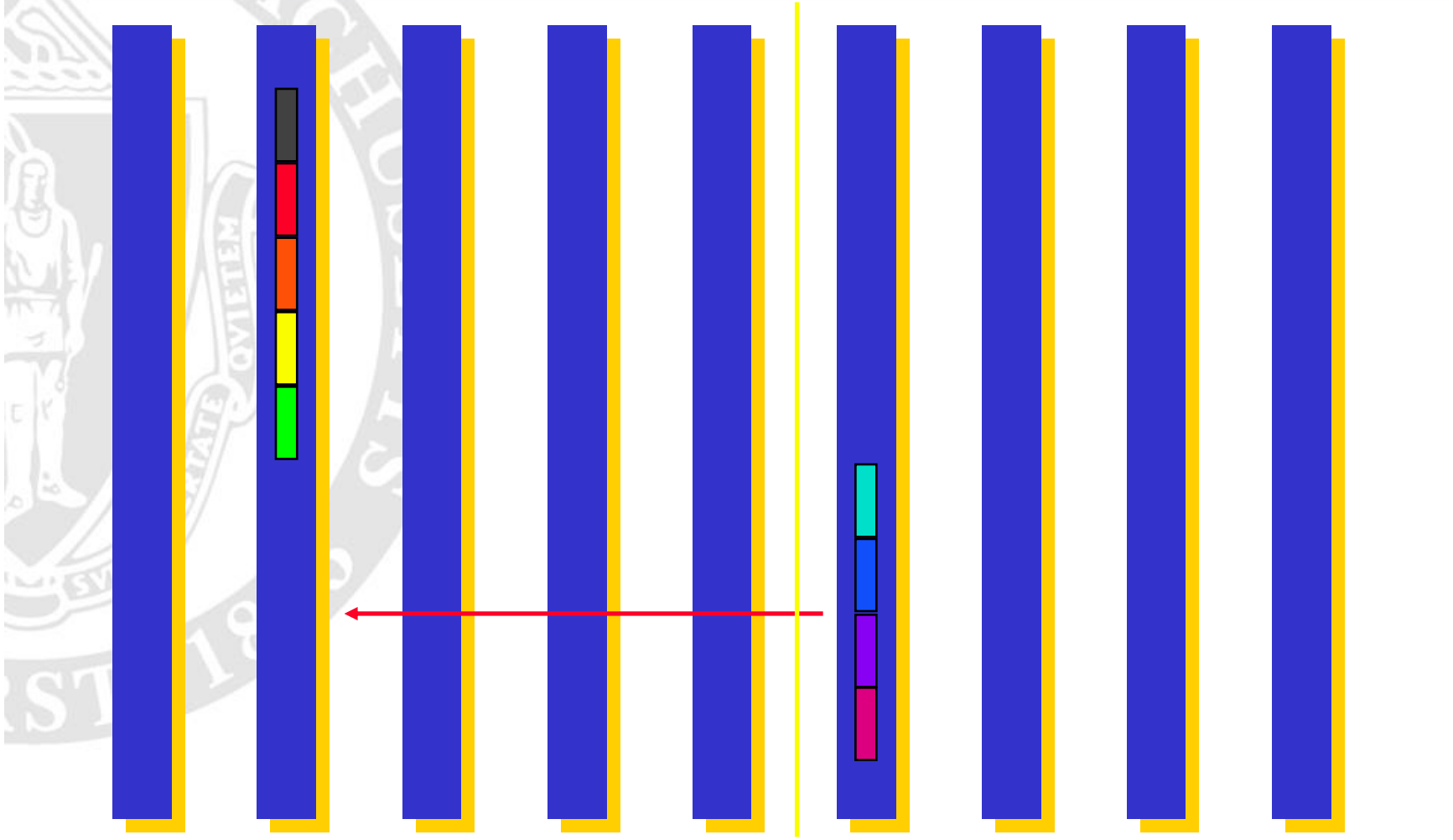


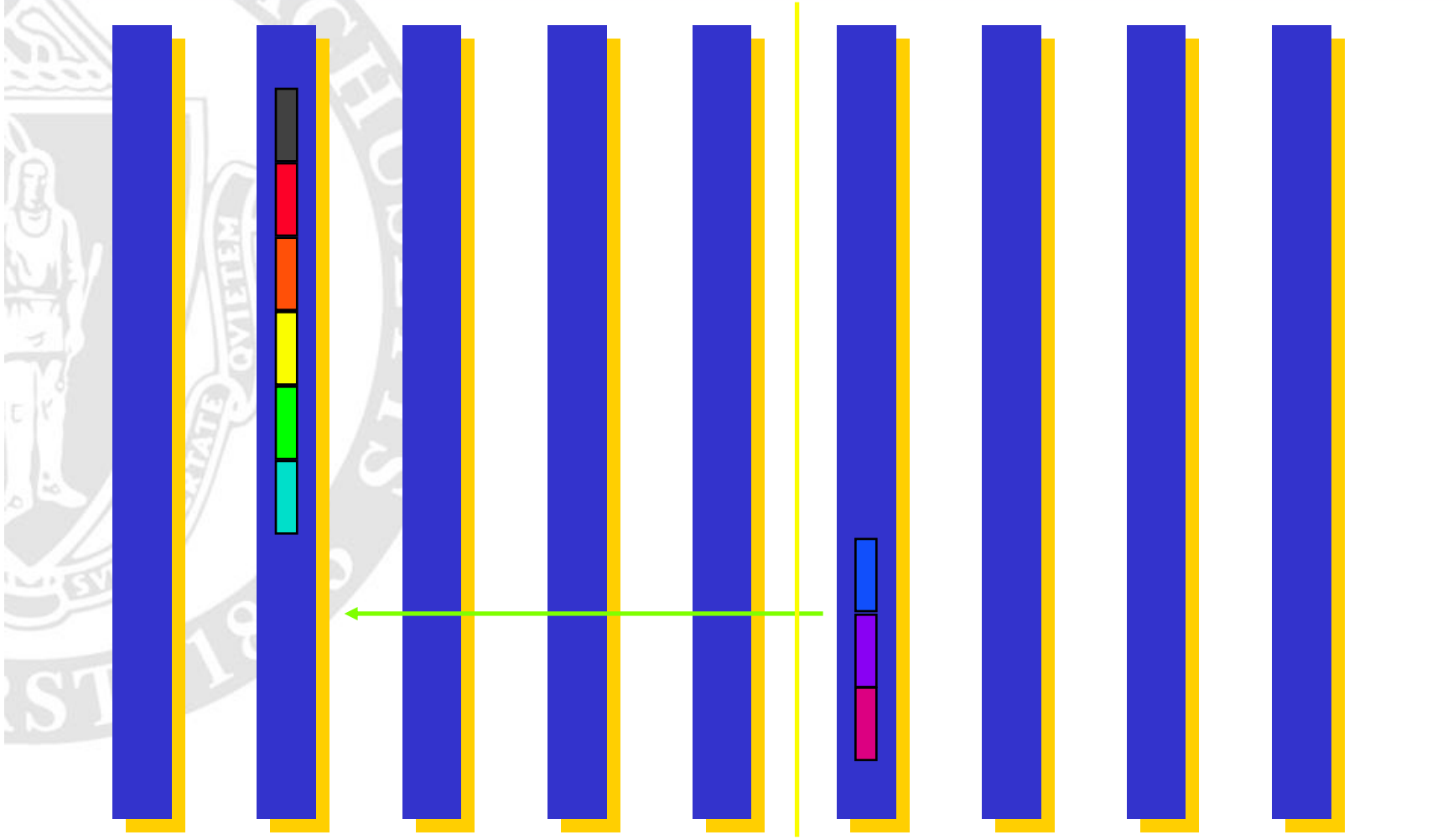


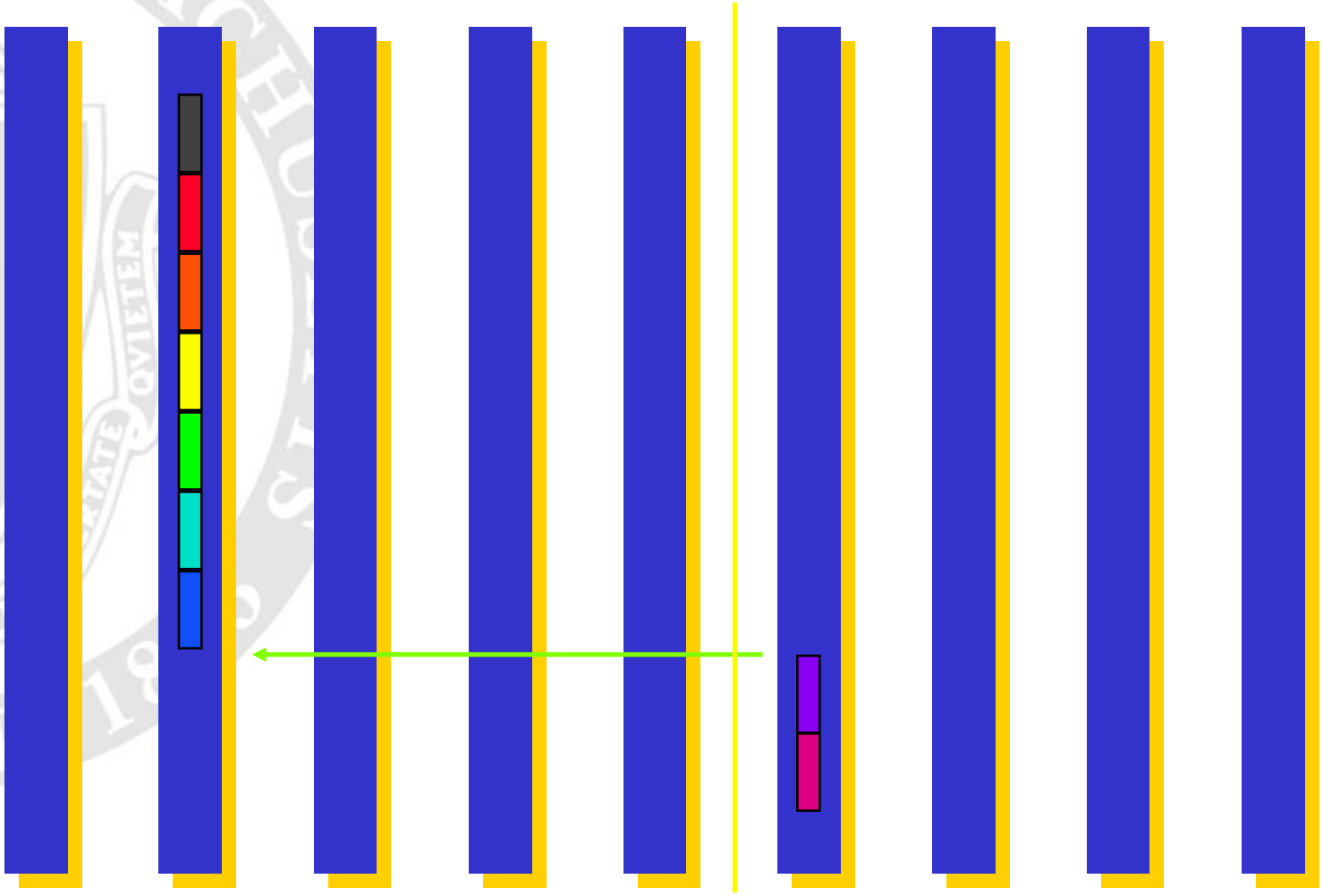


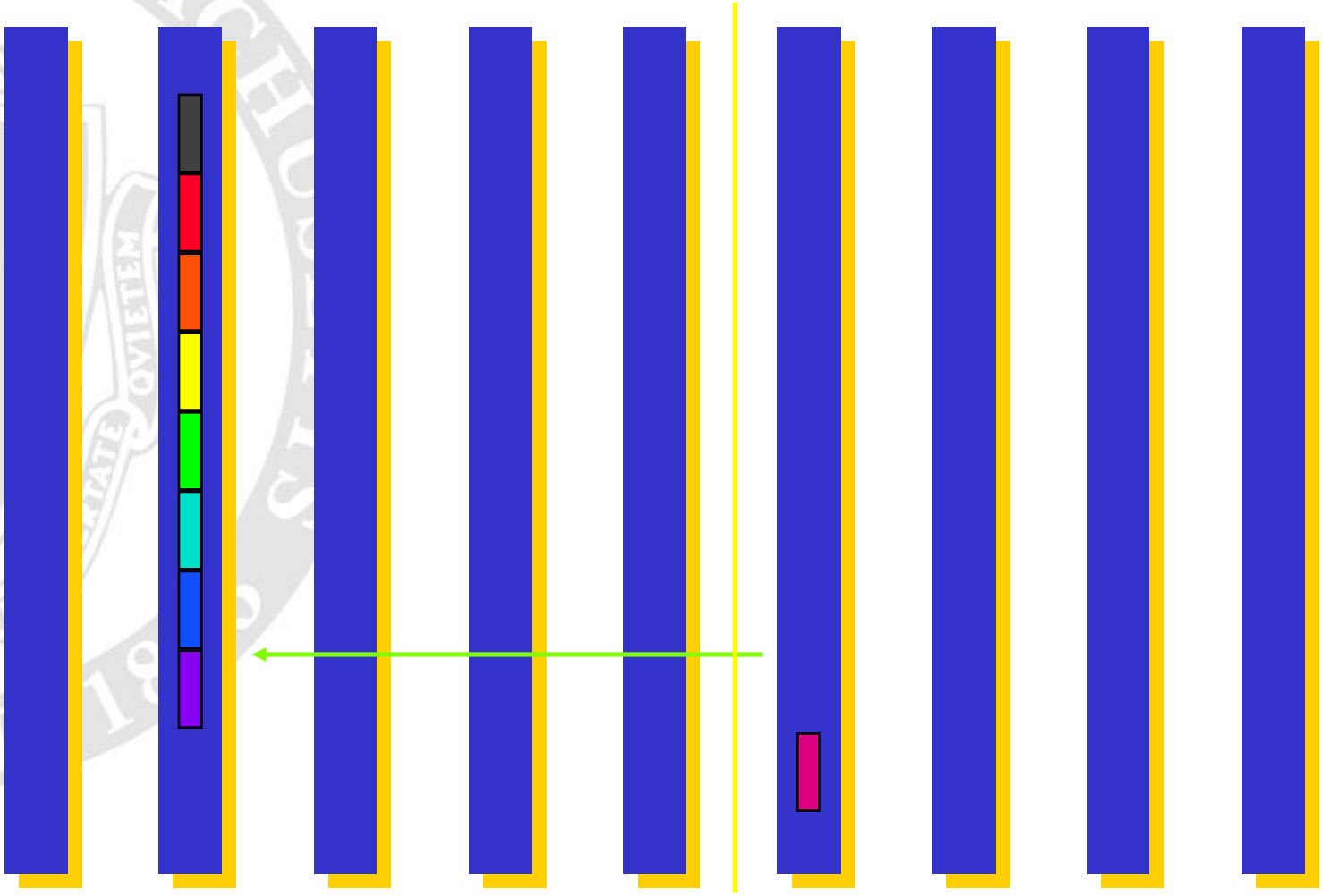


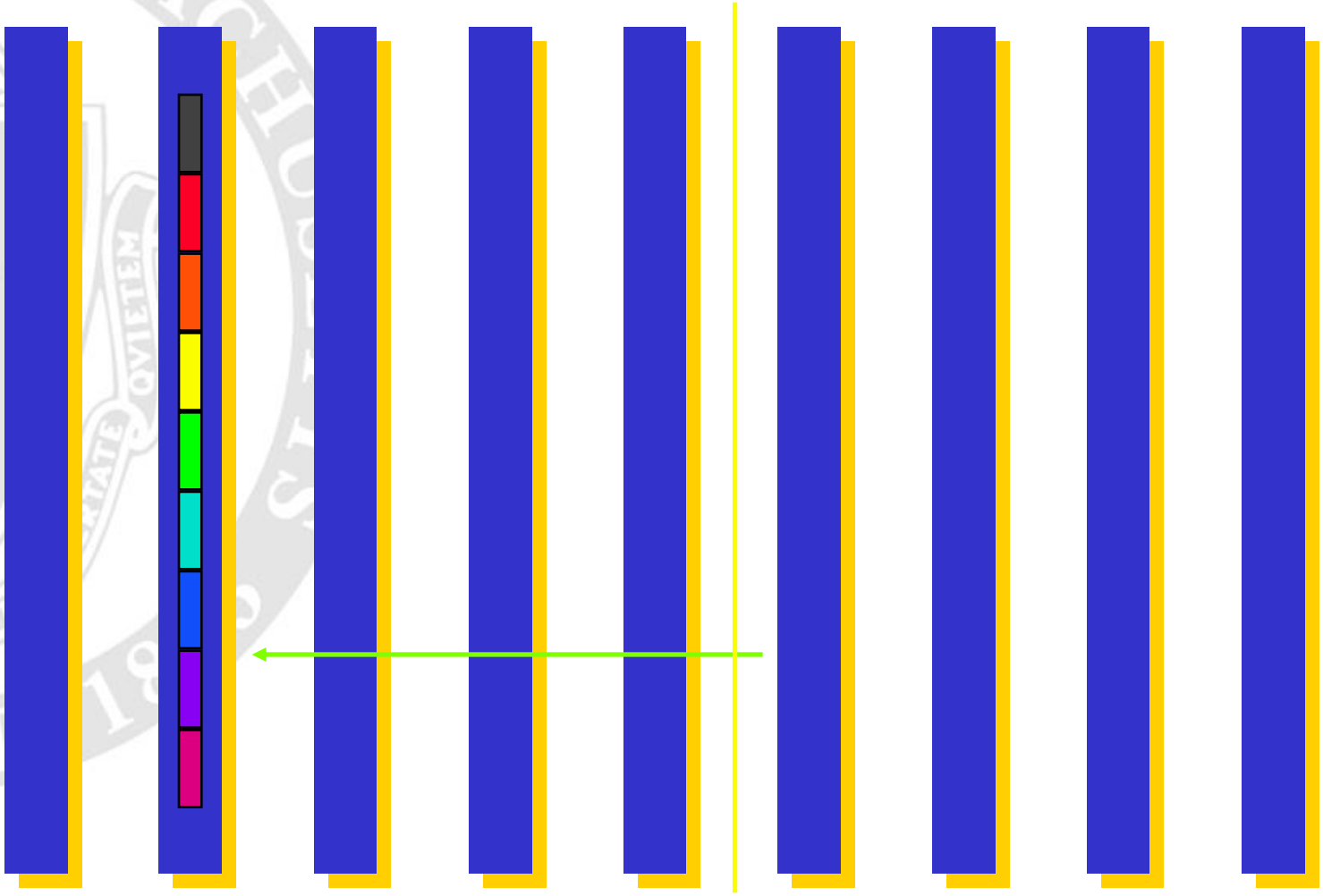


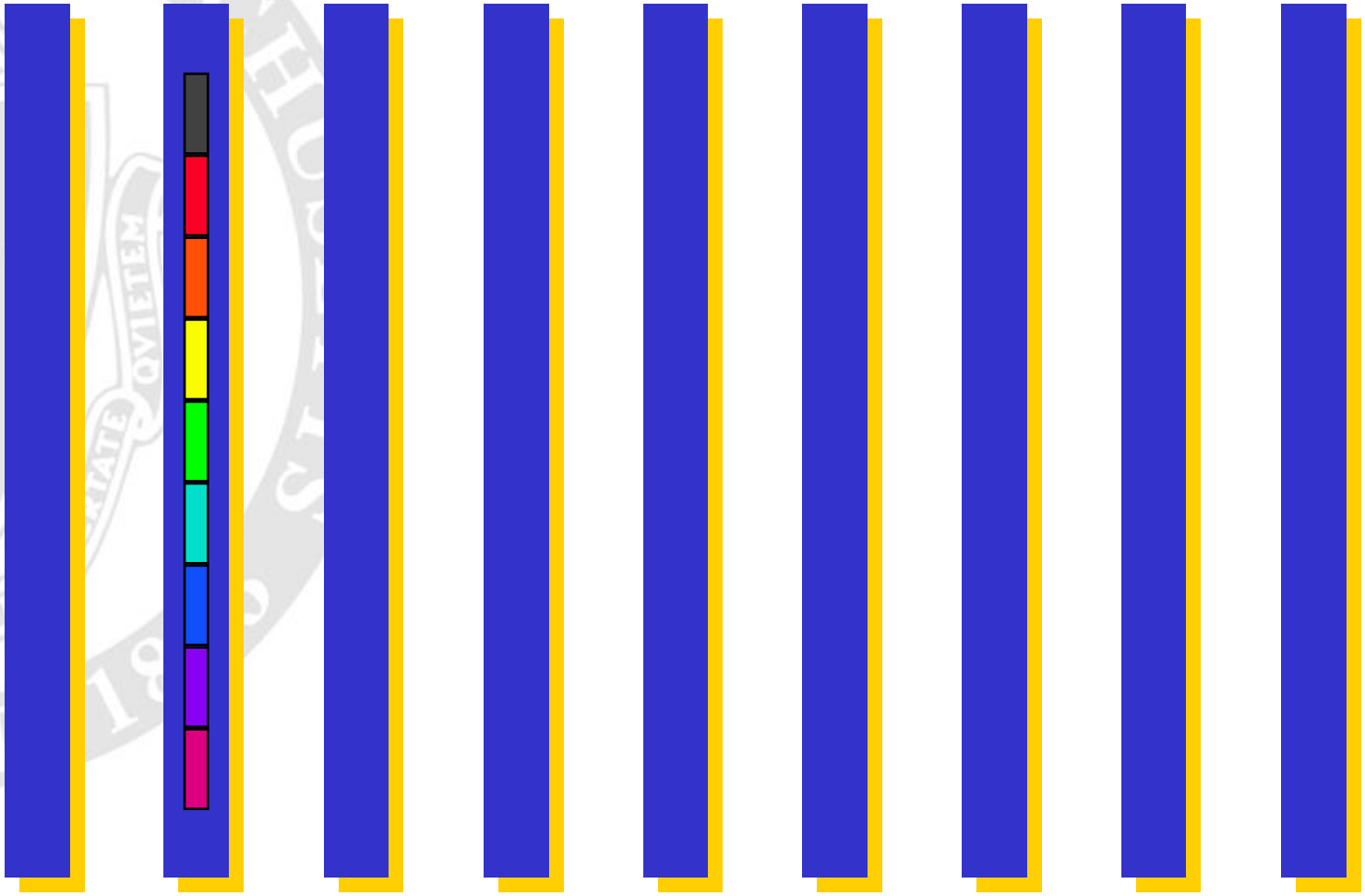






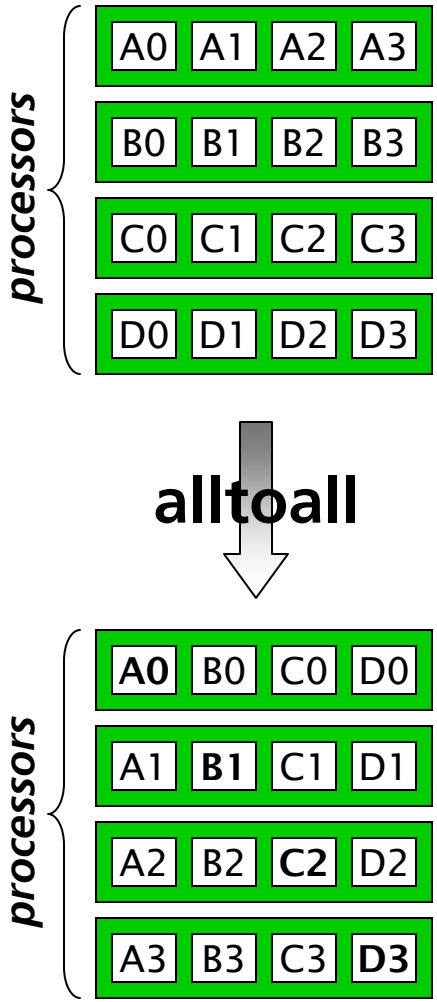






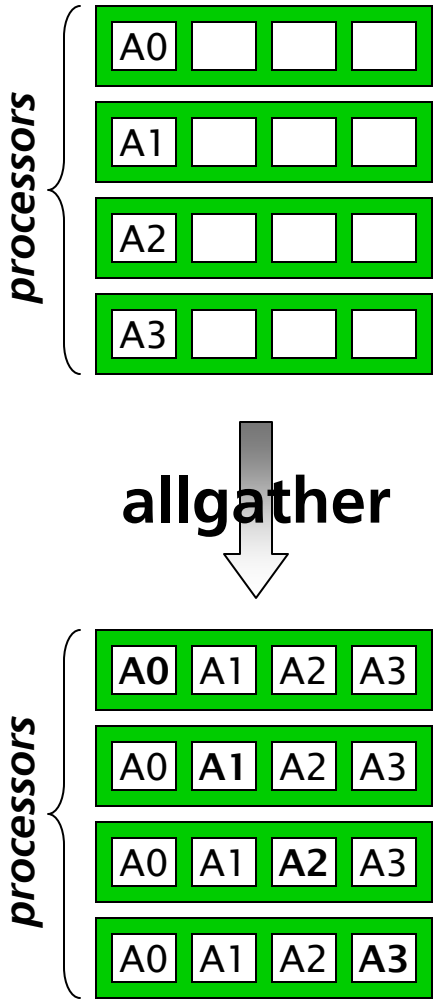
All: Alltoall

- **MPI_Alltoall**: send data from all processors to all processors



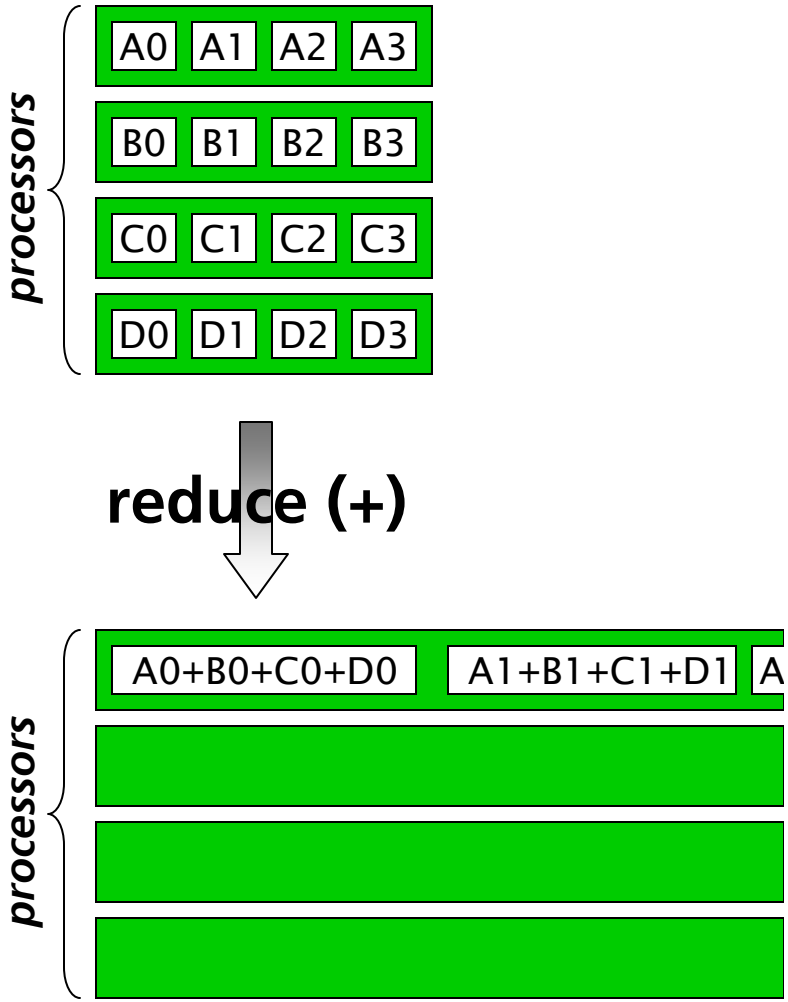
All: Allgather

- **MPI_Allgather**: send data from all processors to one vector across all processors



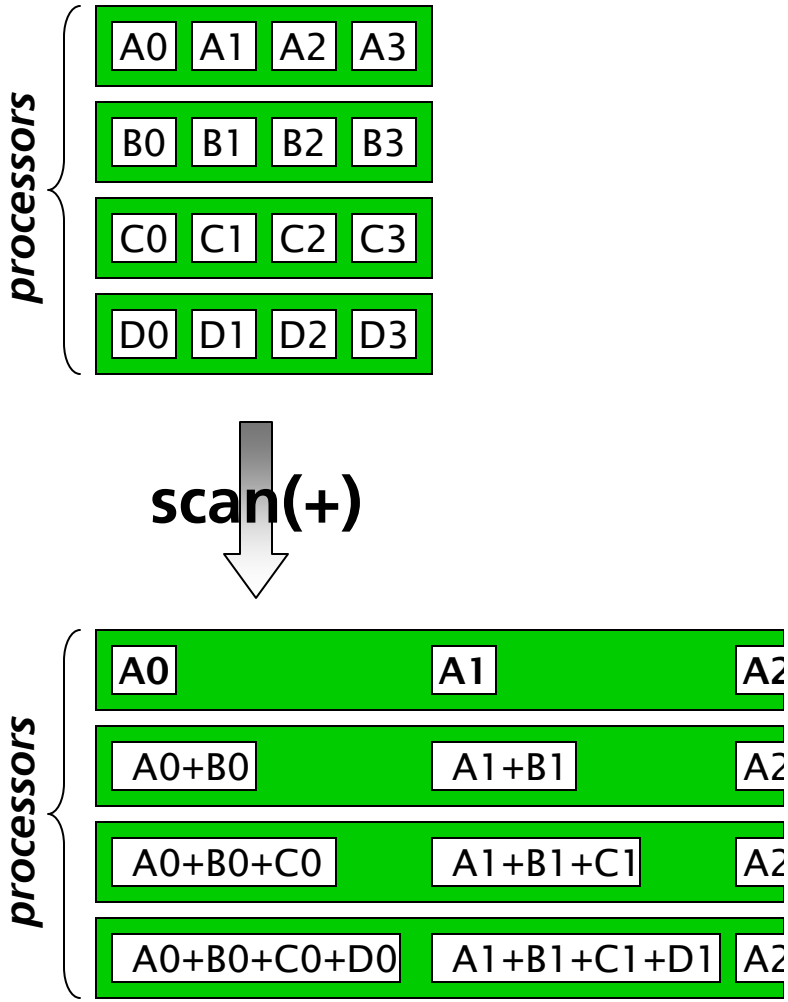
Reductions: Reduce

- **MPI_Reduce**: collect data from all processors, apply operator to each, put on one processor



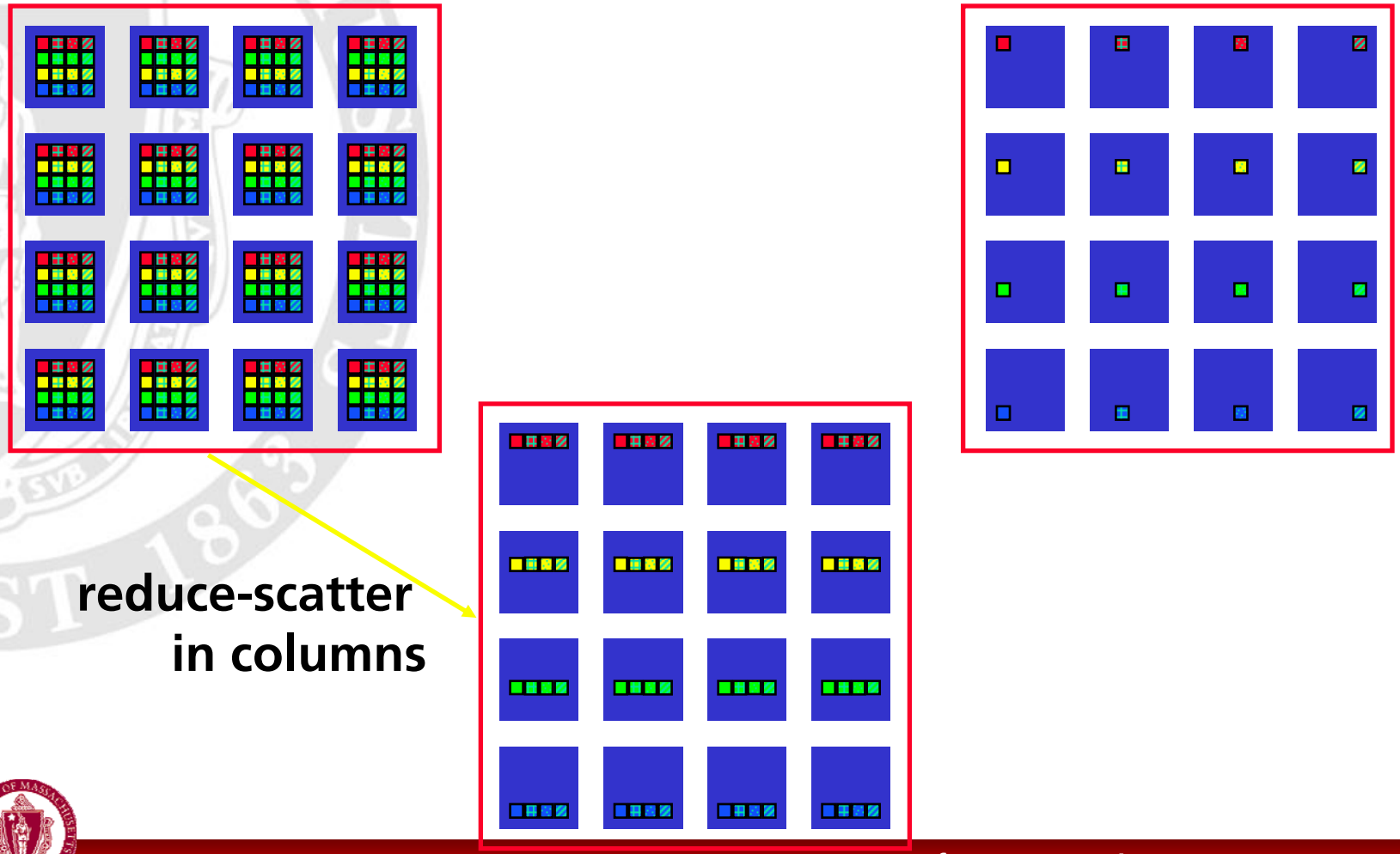
Reductions: Scan

- **MPI_Scan**: apply partial reductions
- Operation applied to increasingly-long prefixes

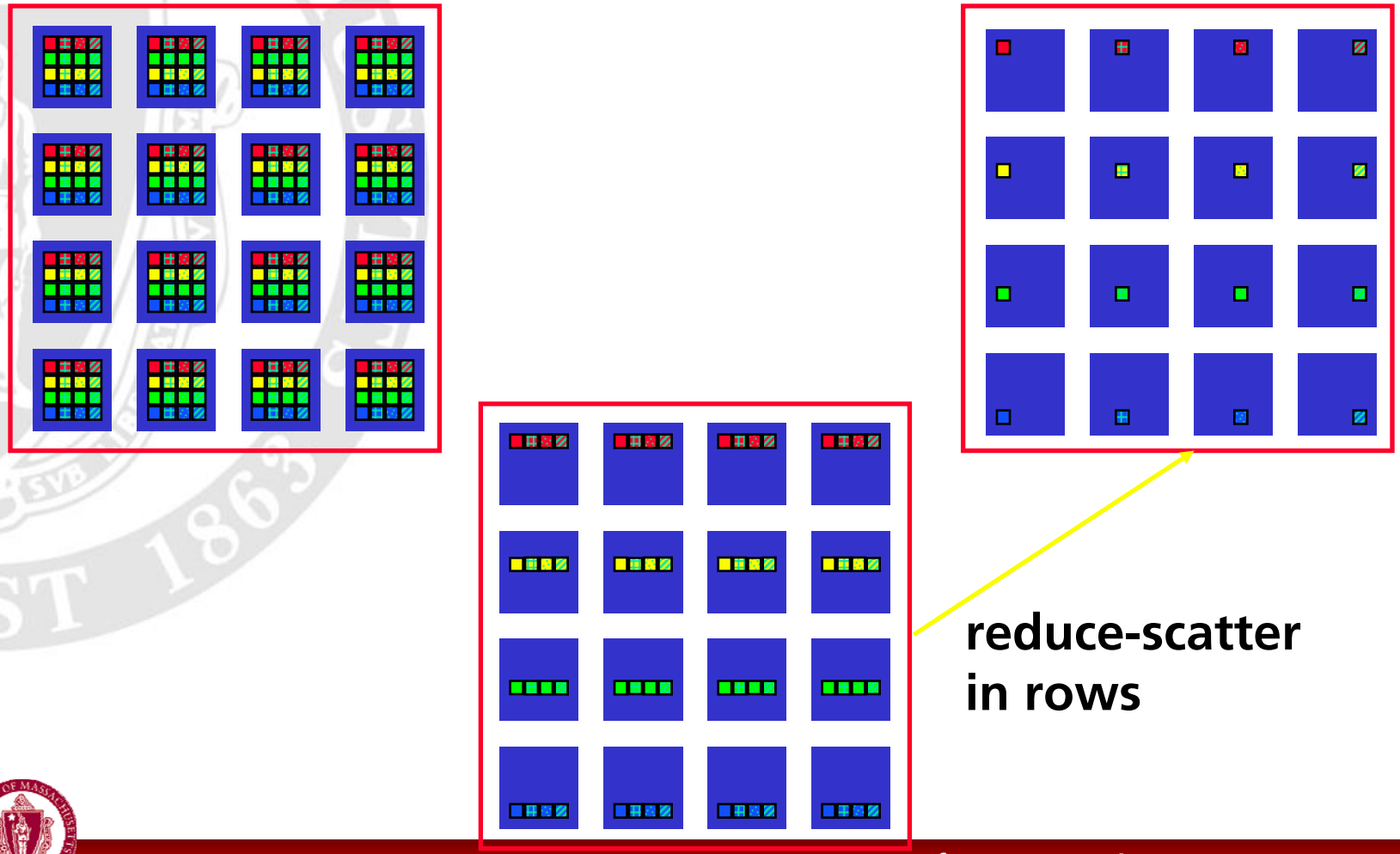


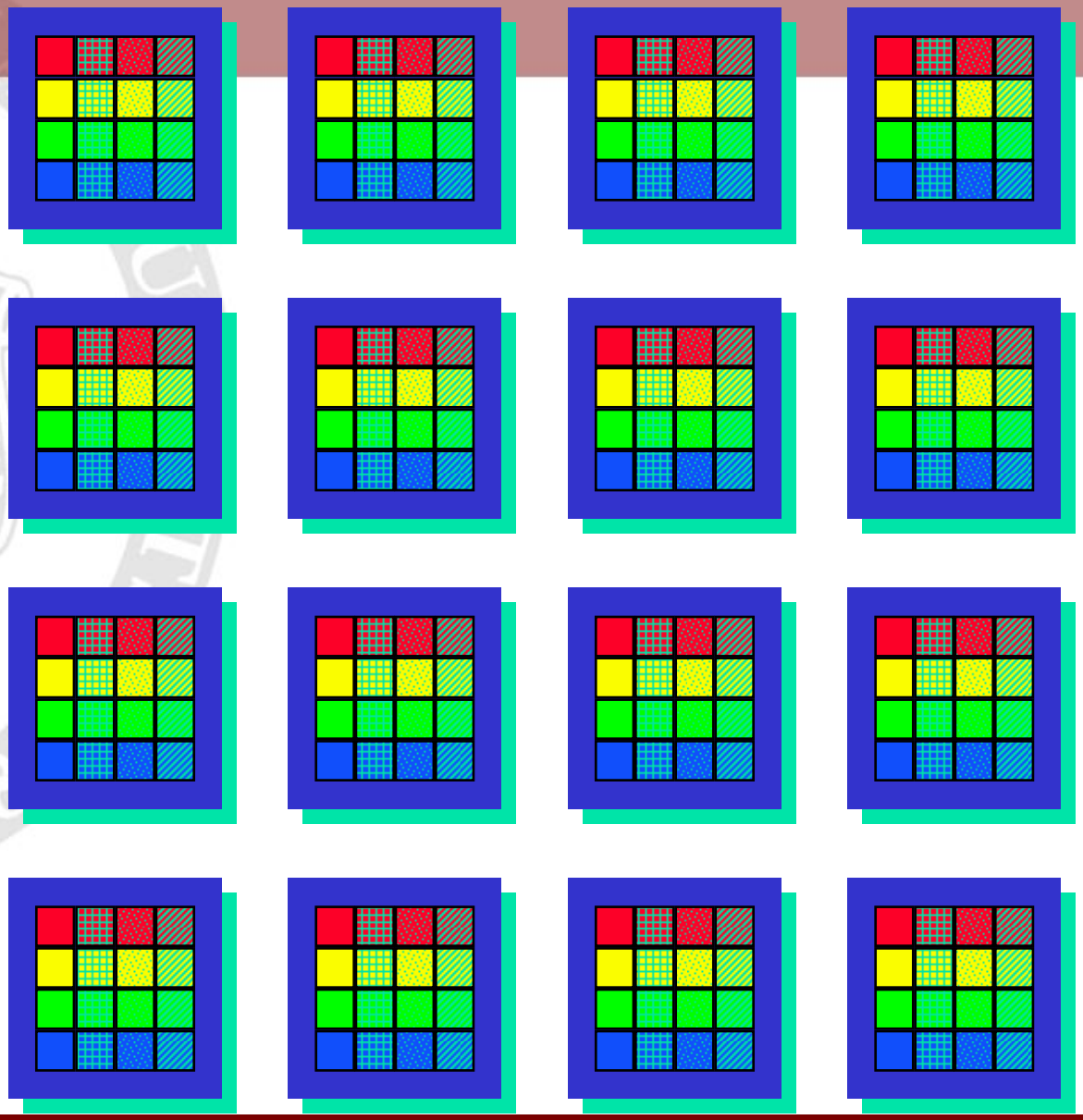
2D Reduce-scatter

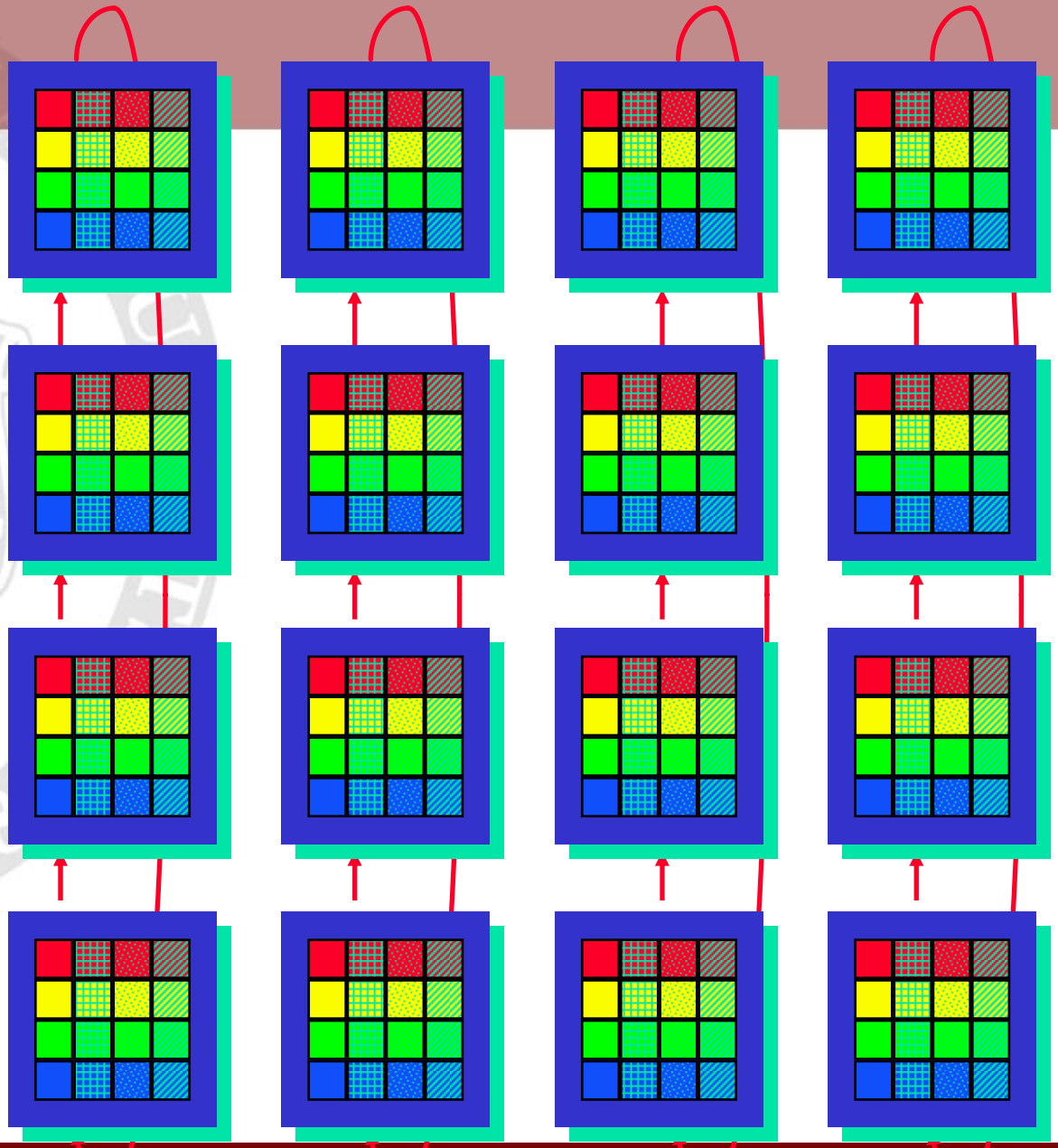
2D Reduce-scatter

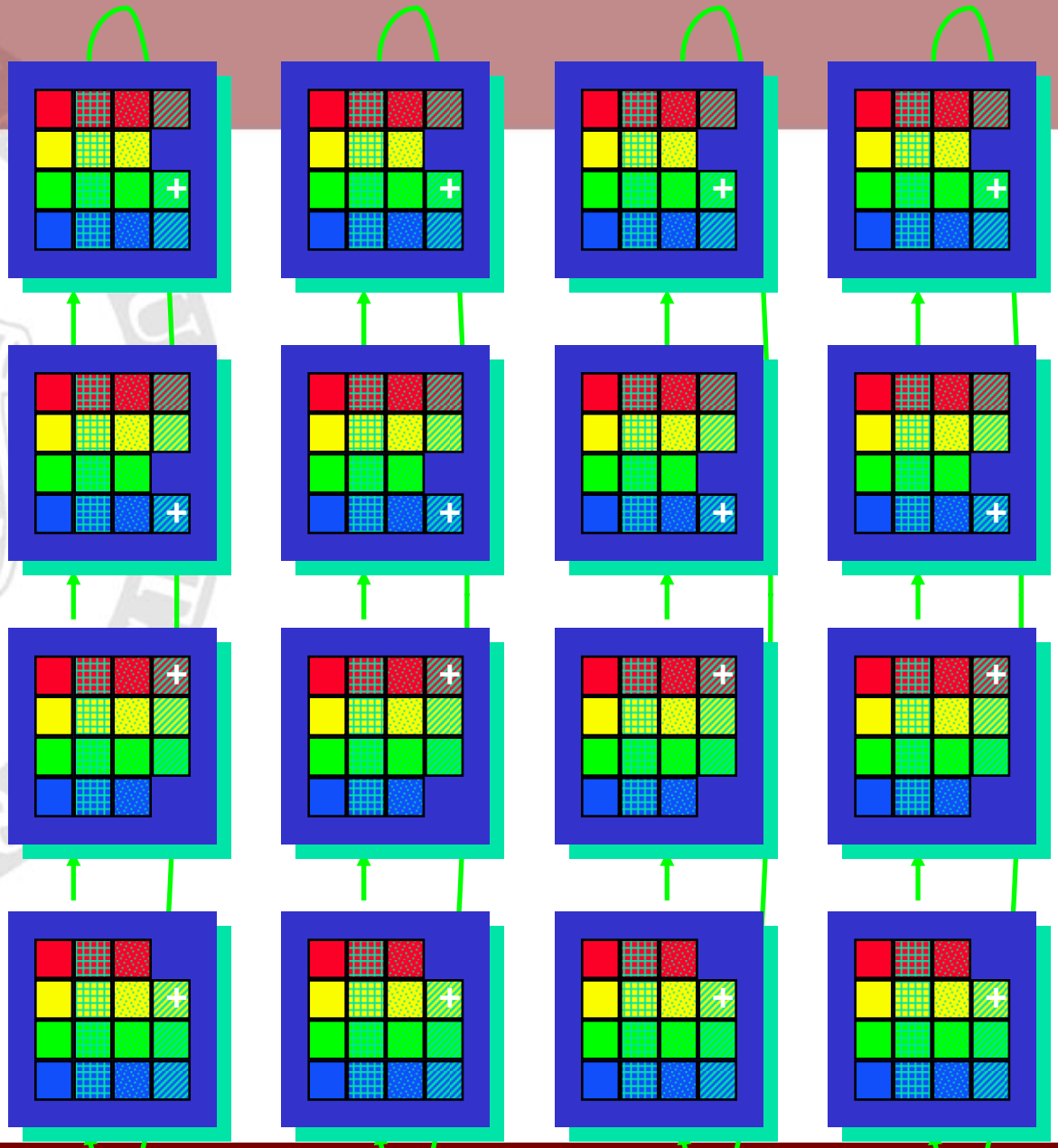


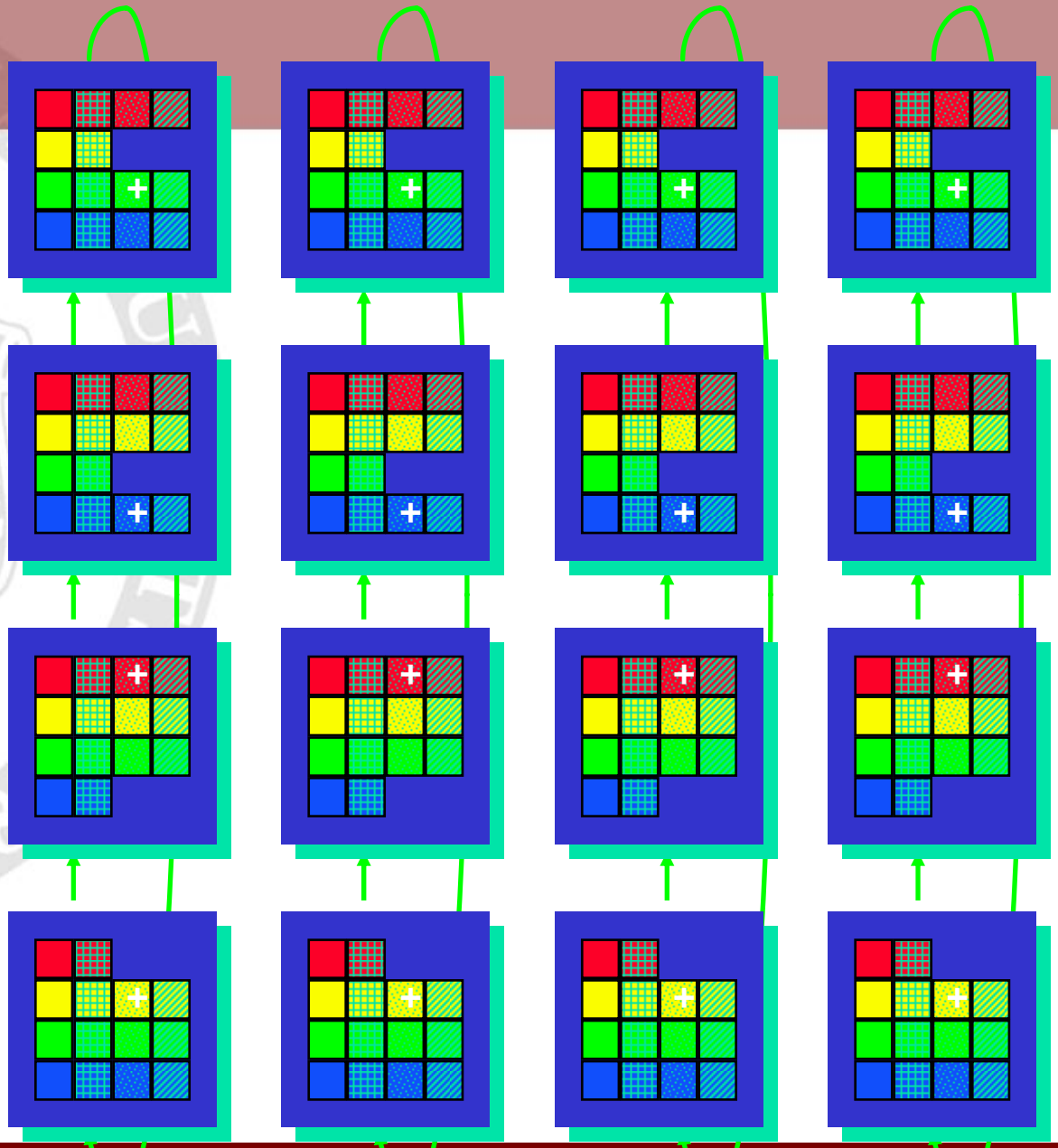
2D Reduce-scatter

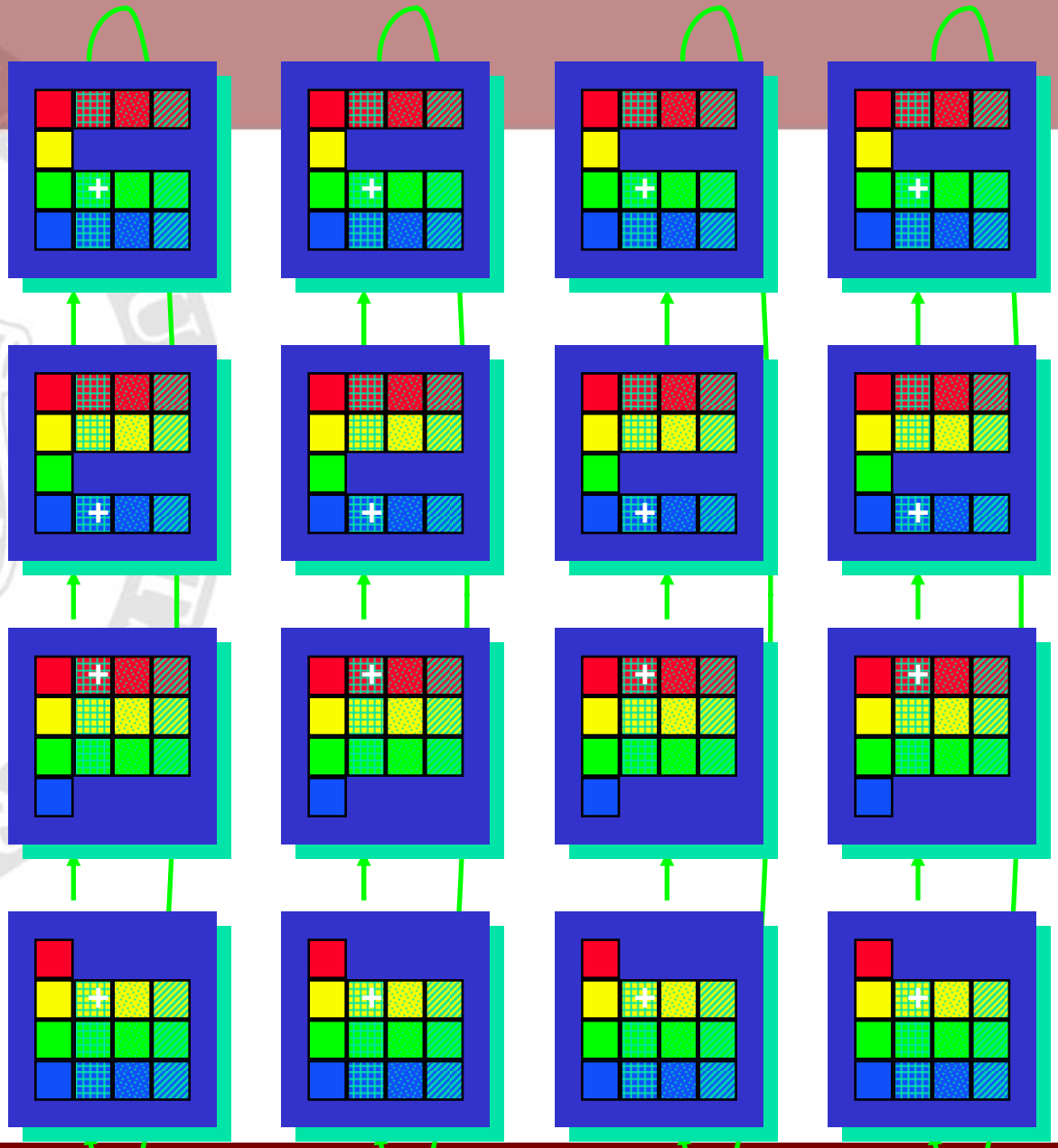


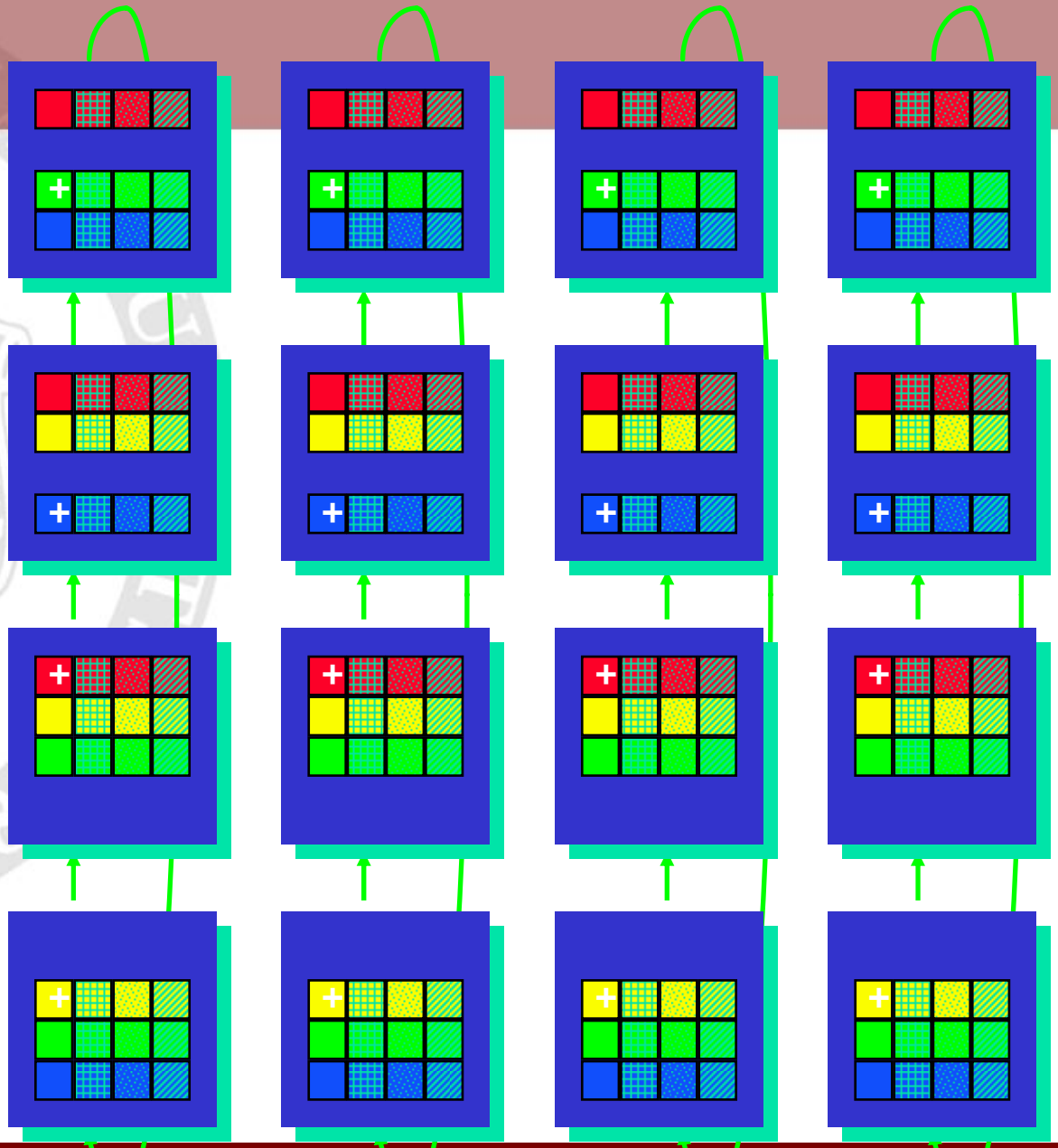


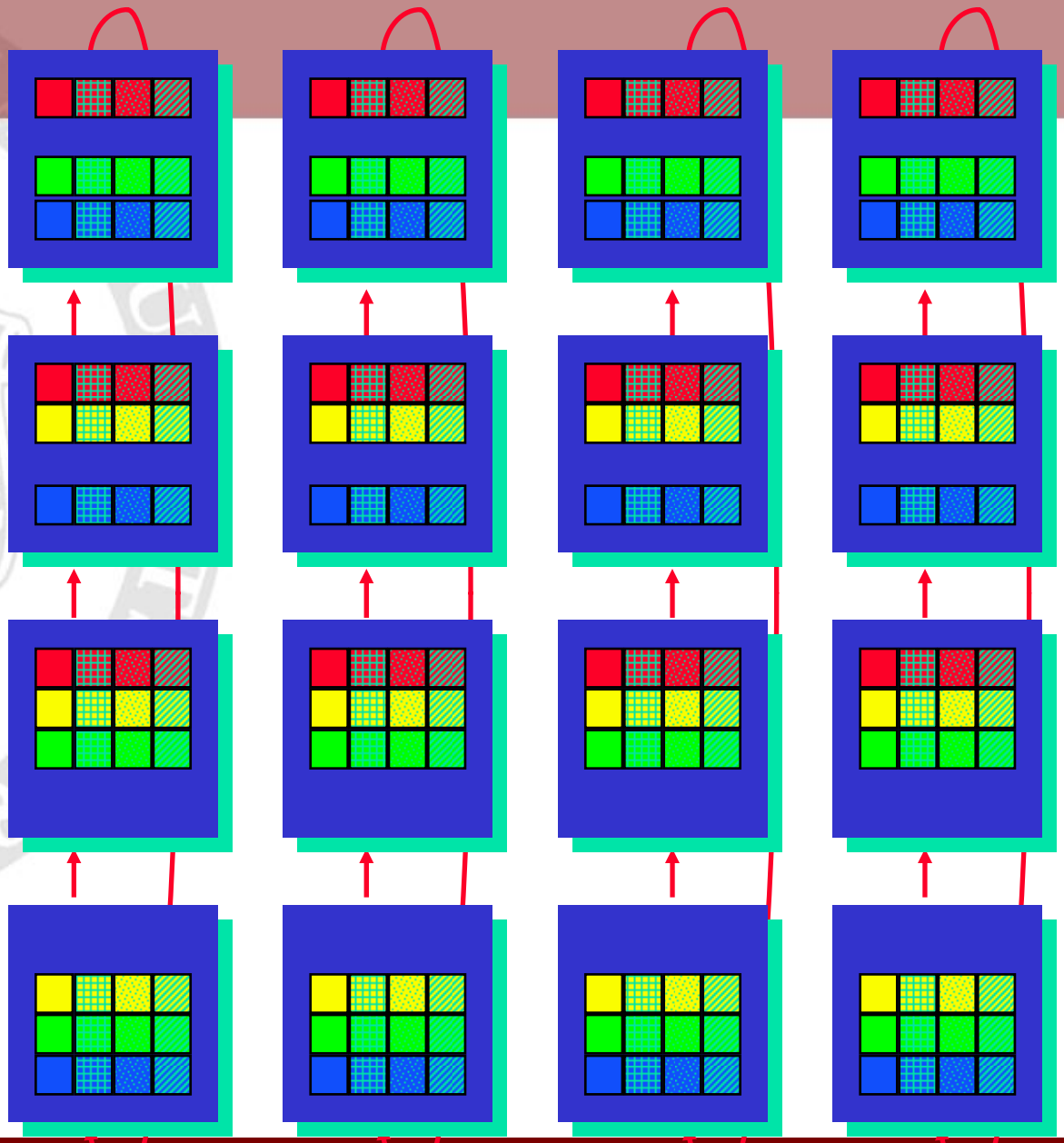


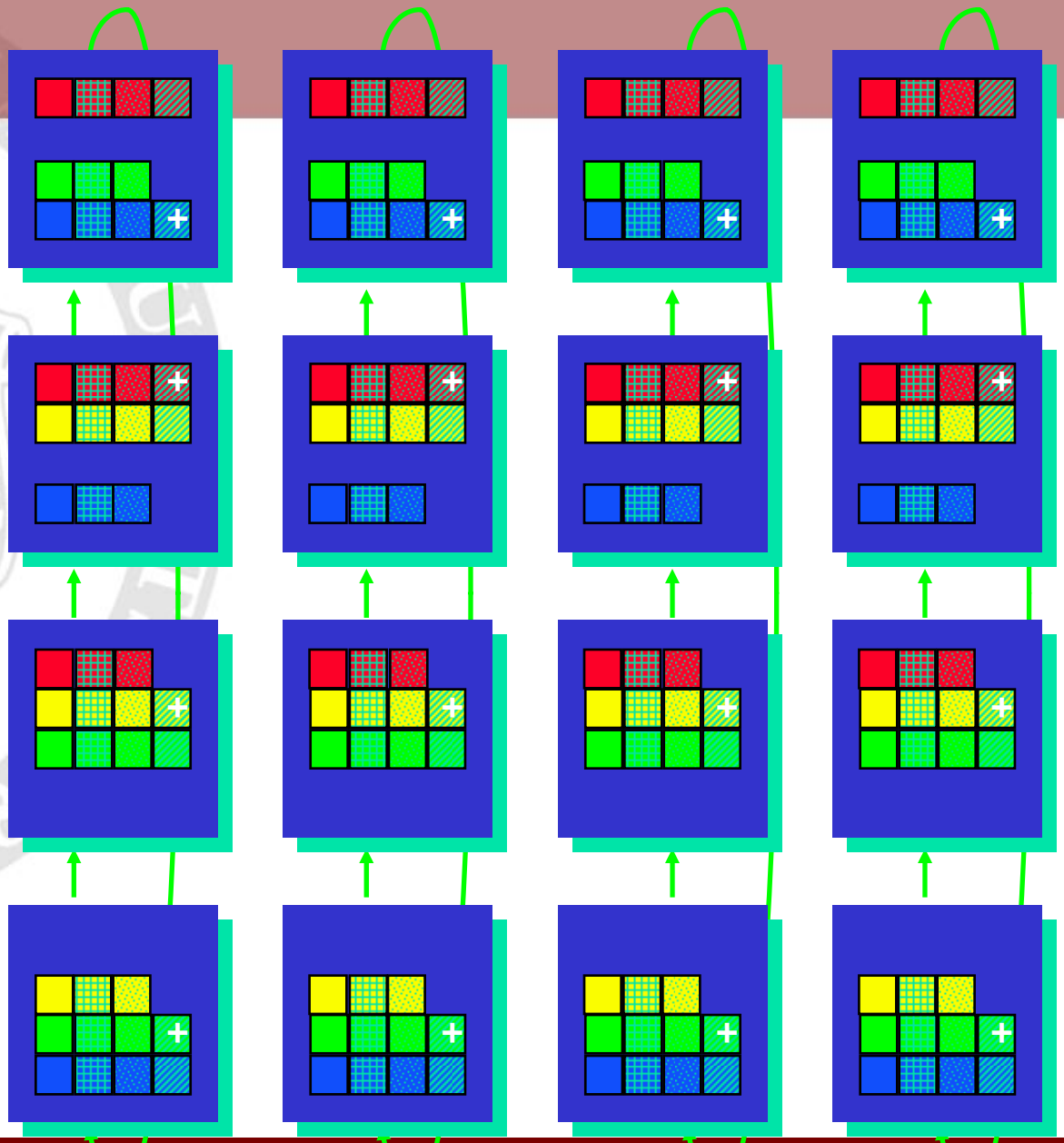


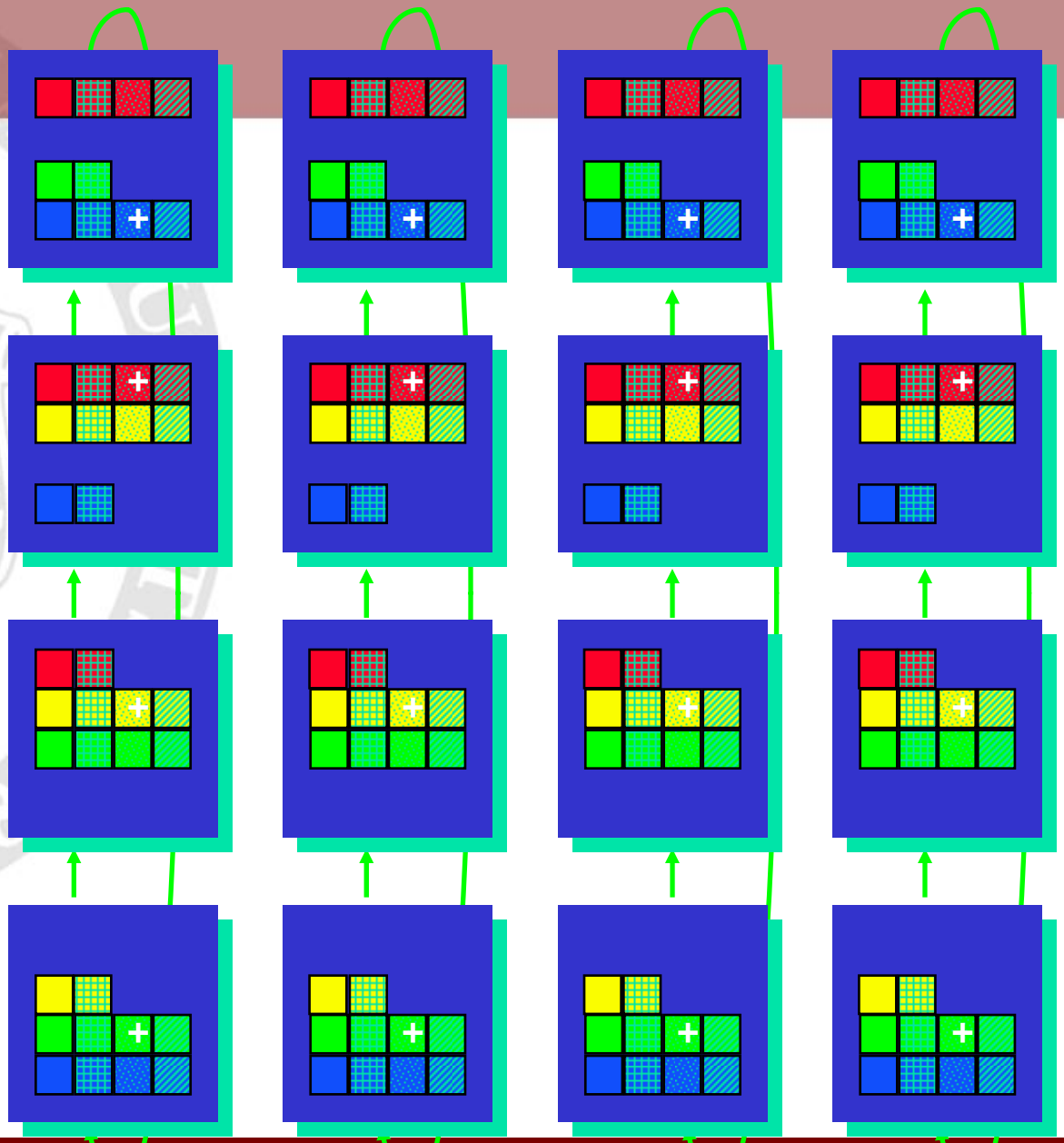


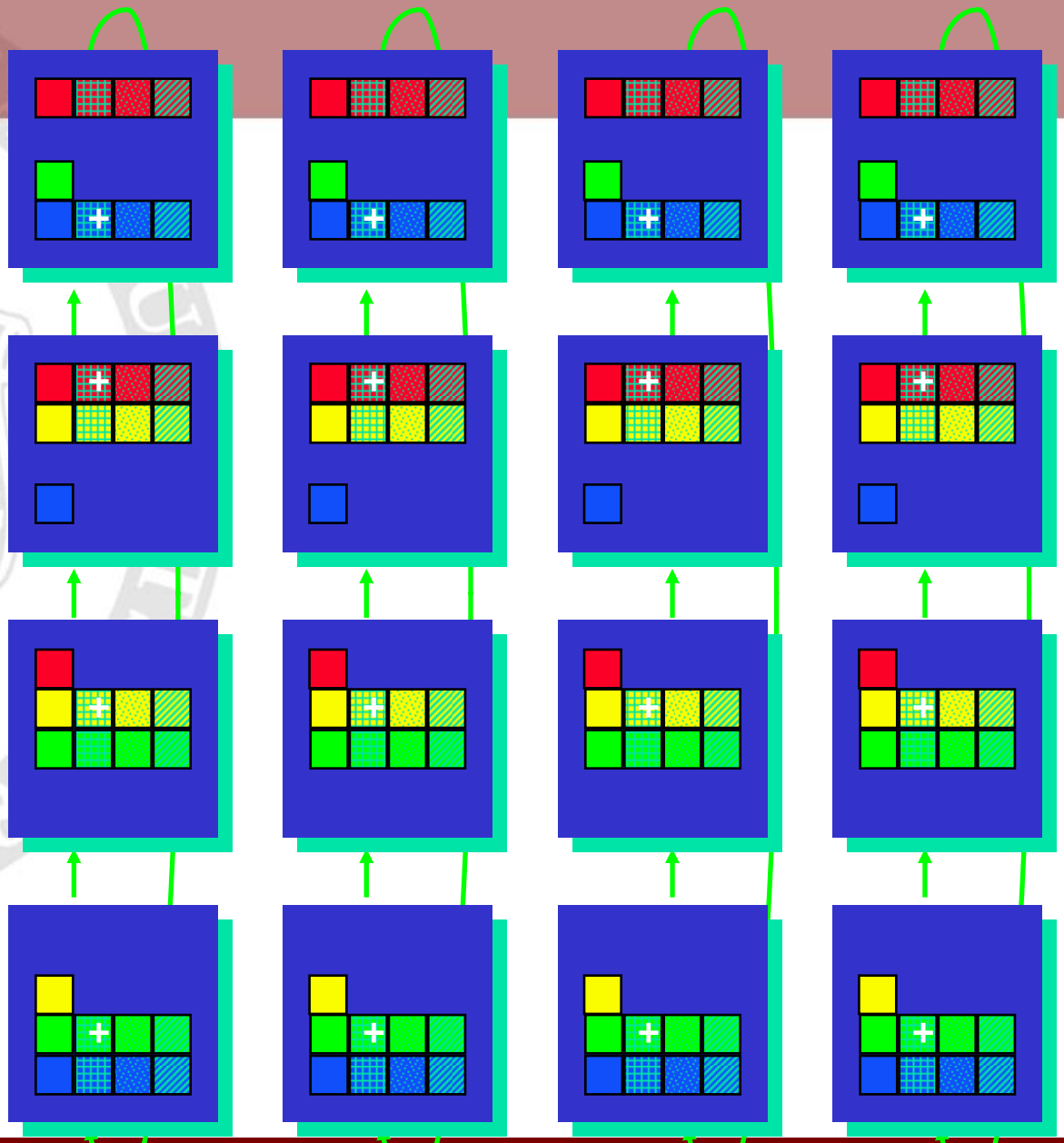


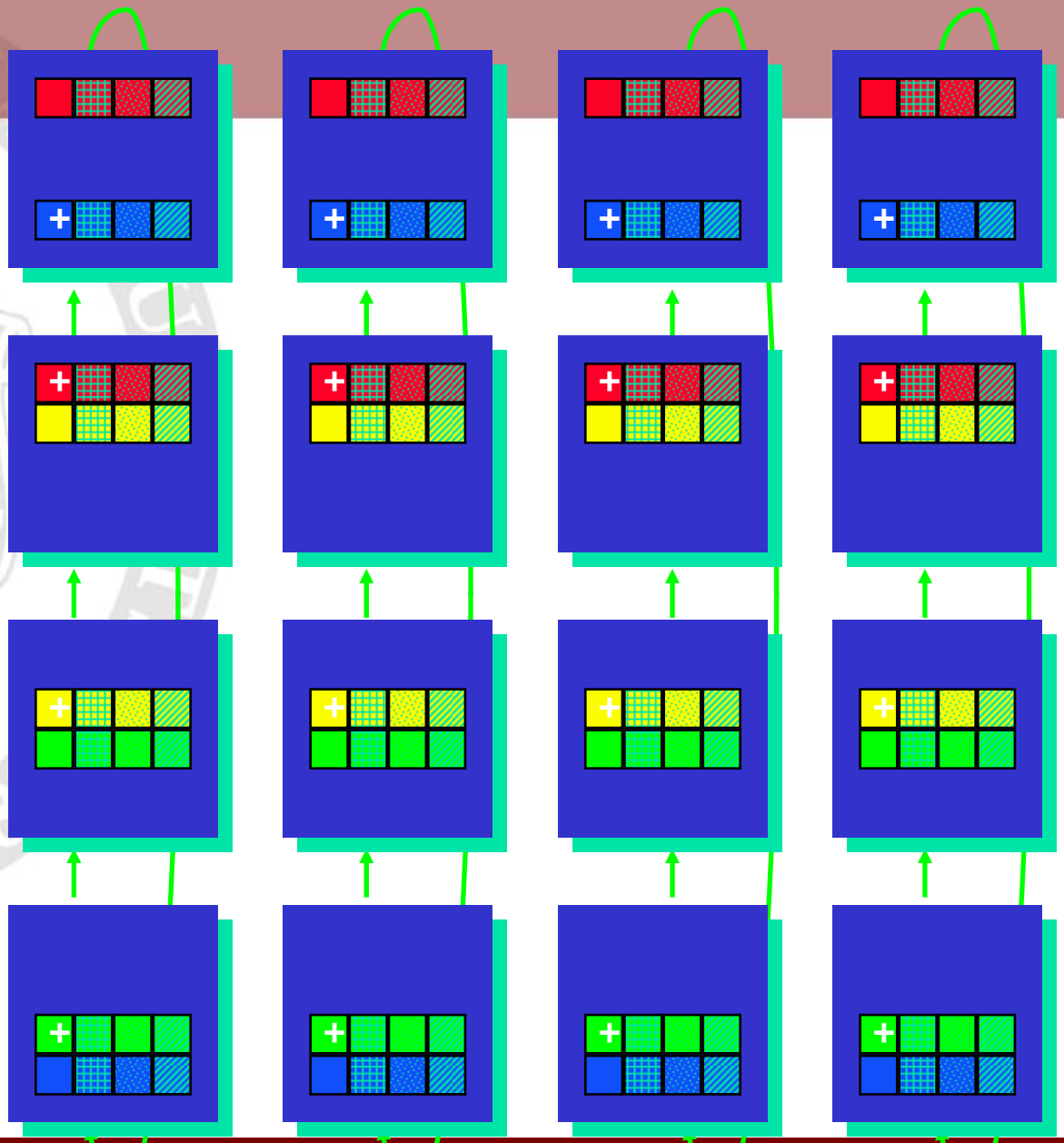


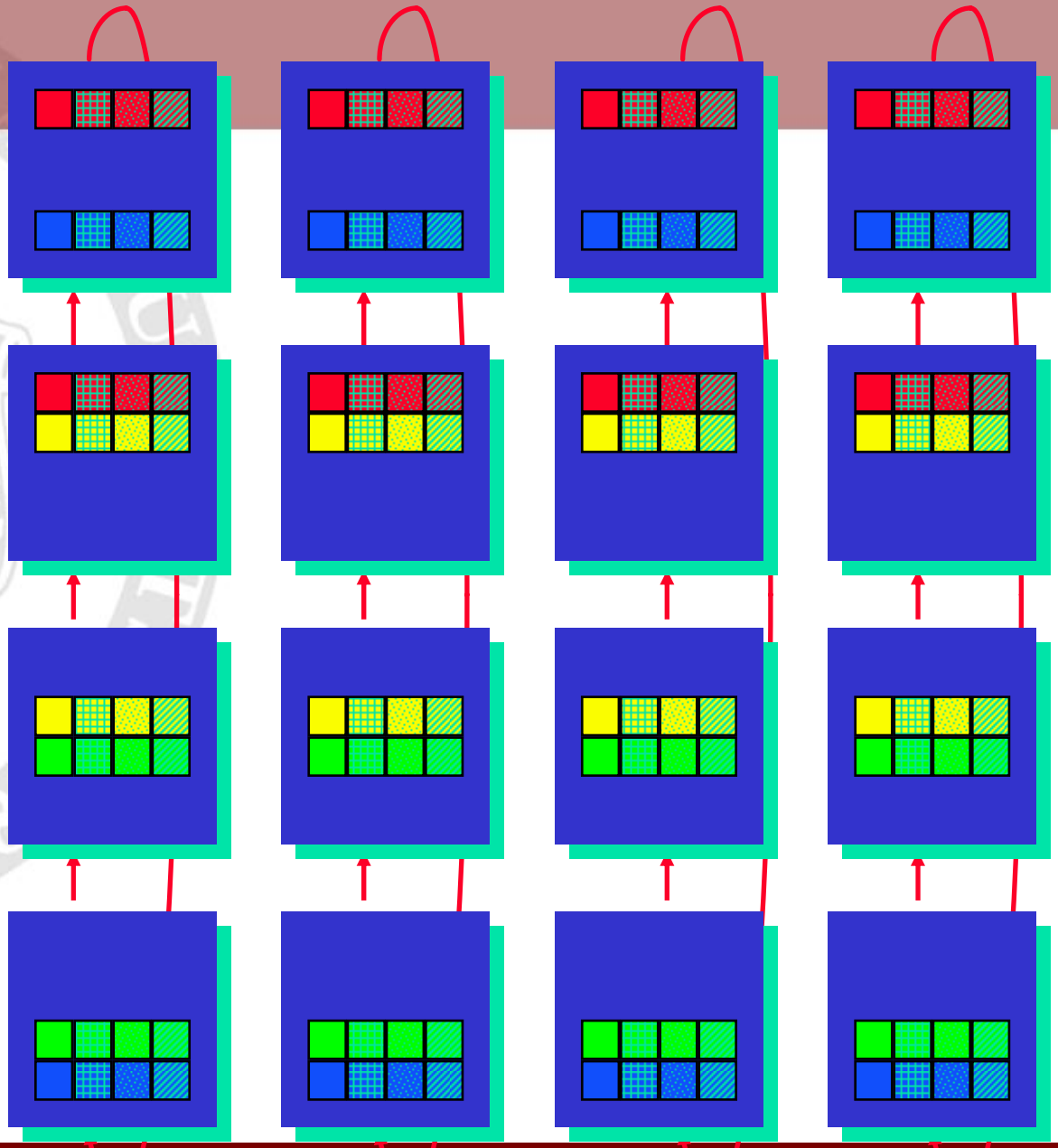


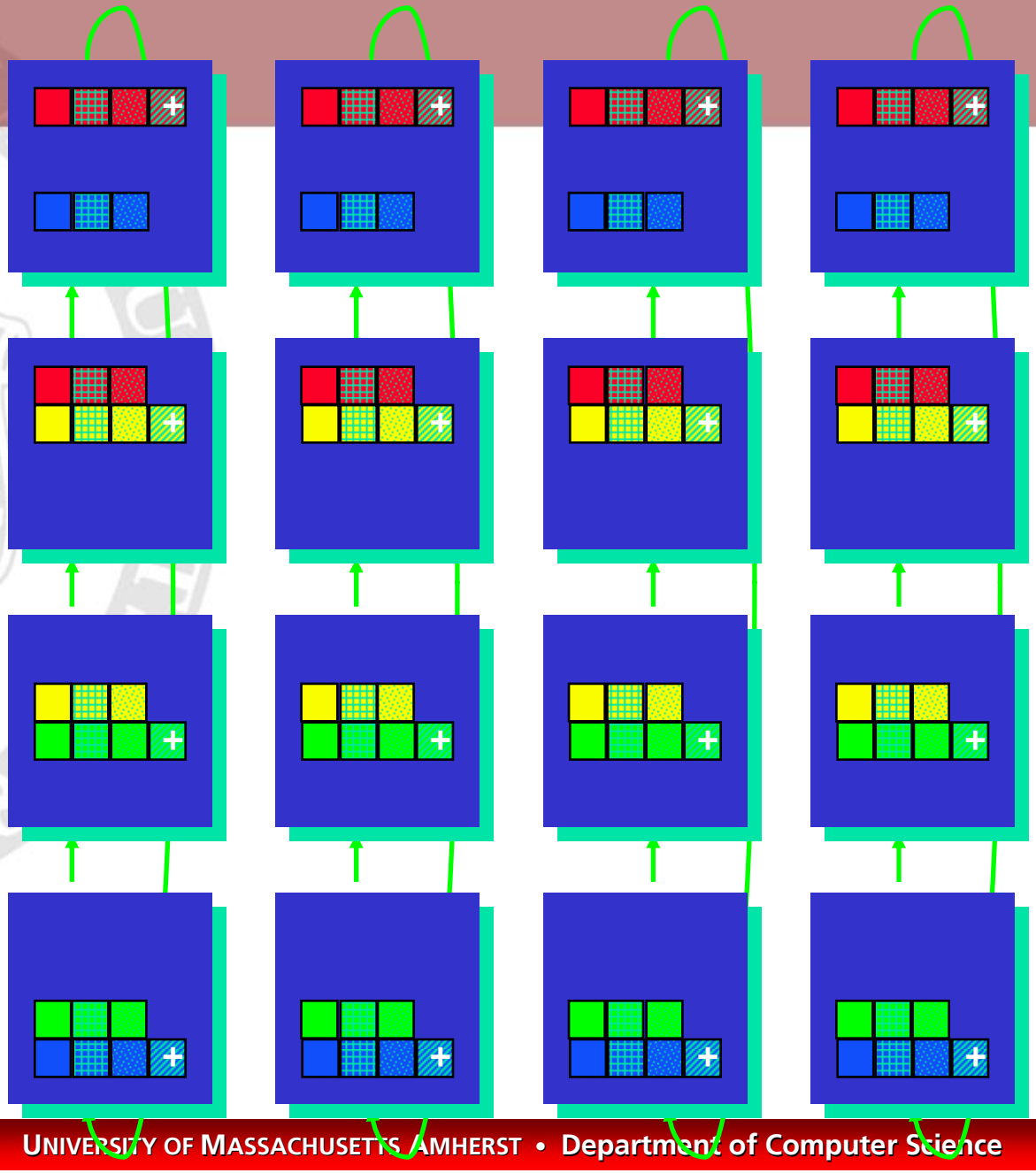


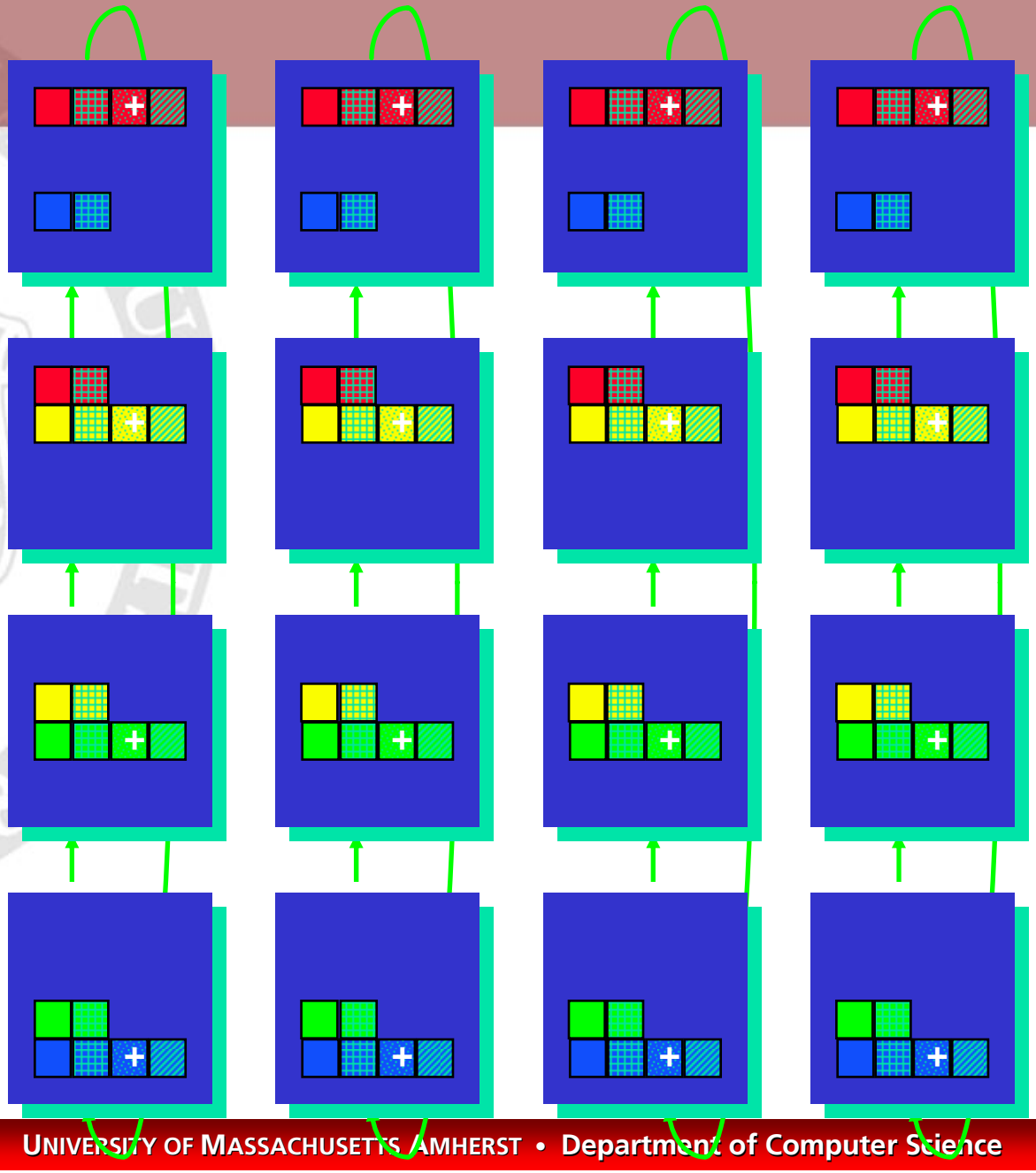


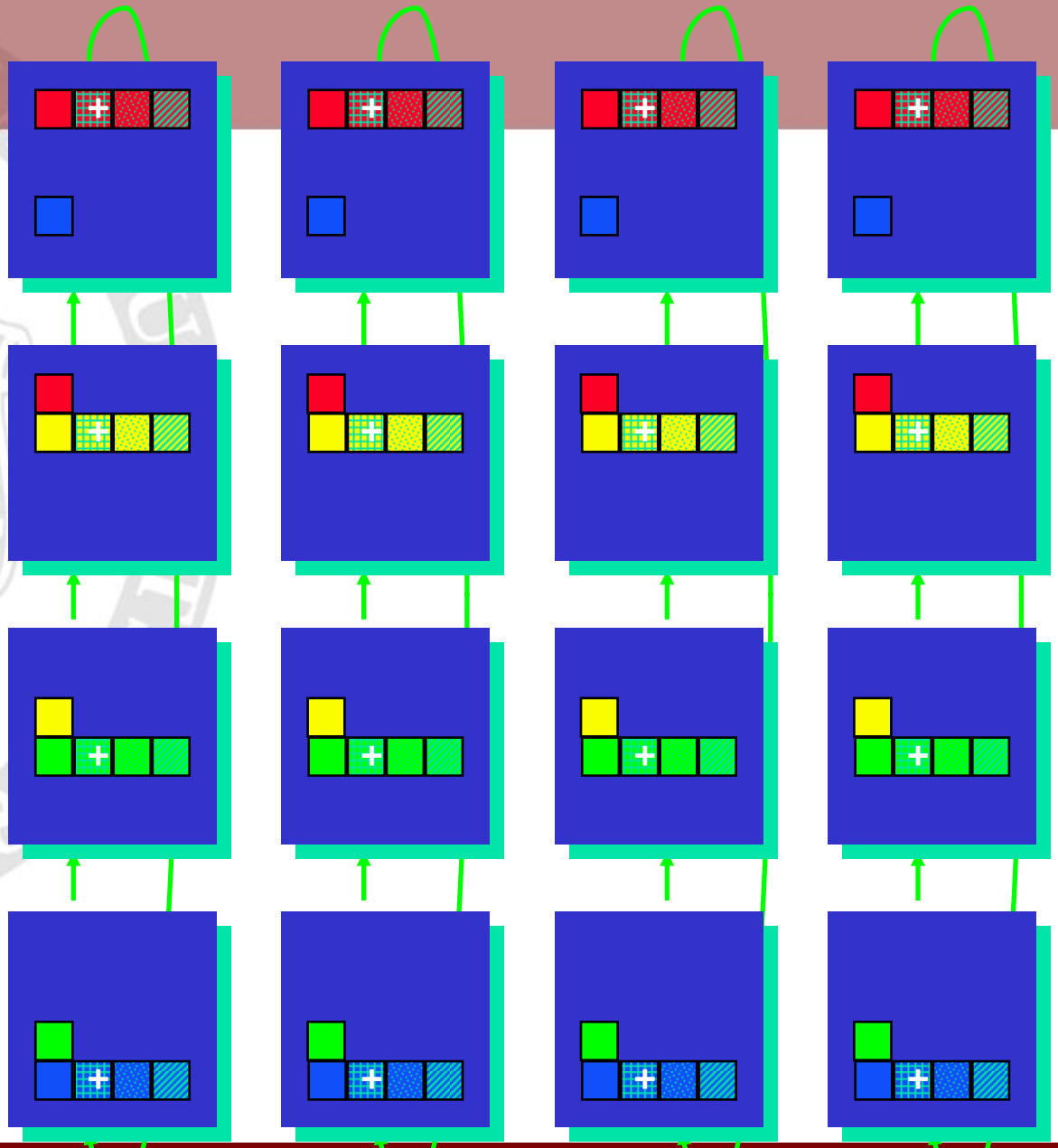


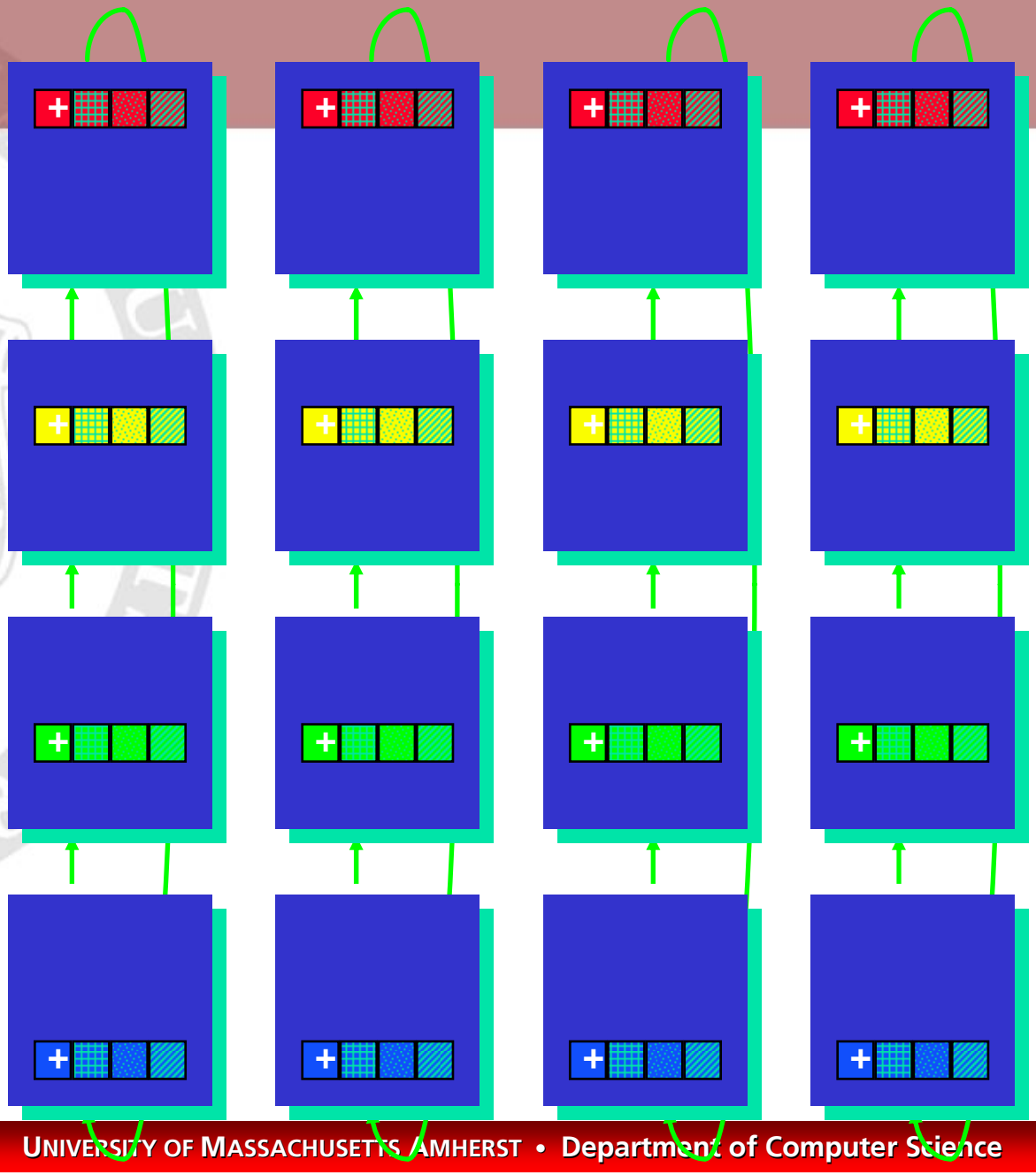


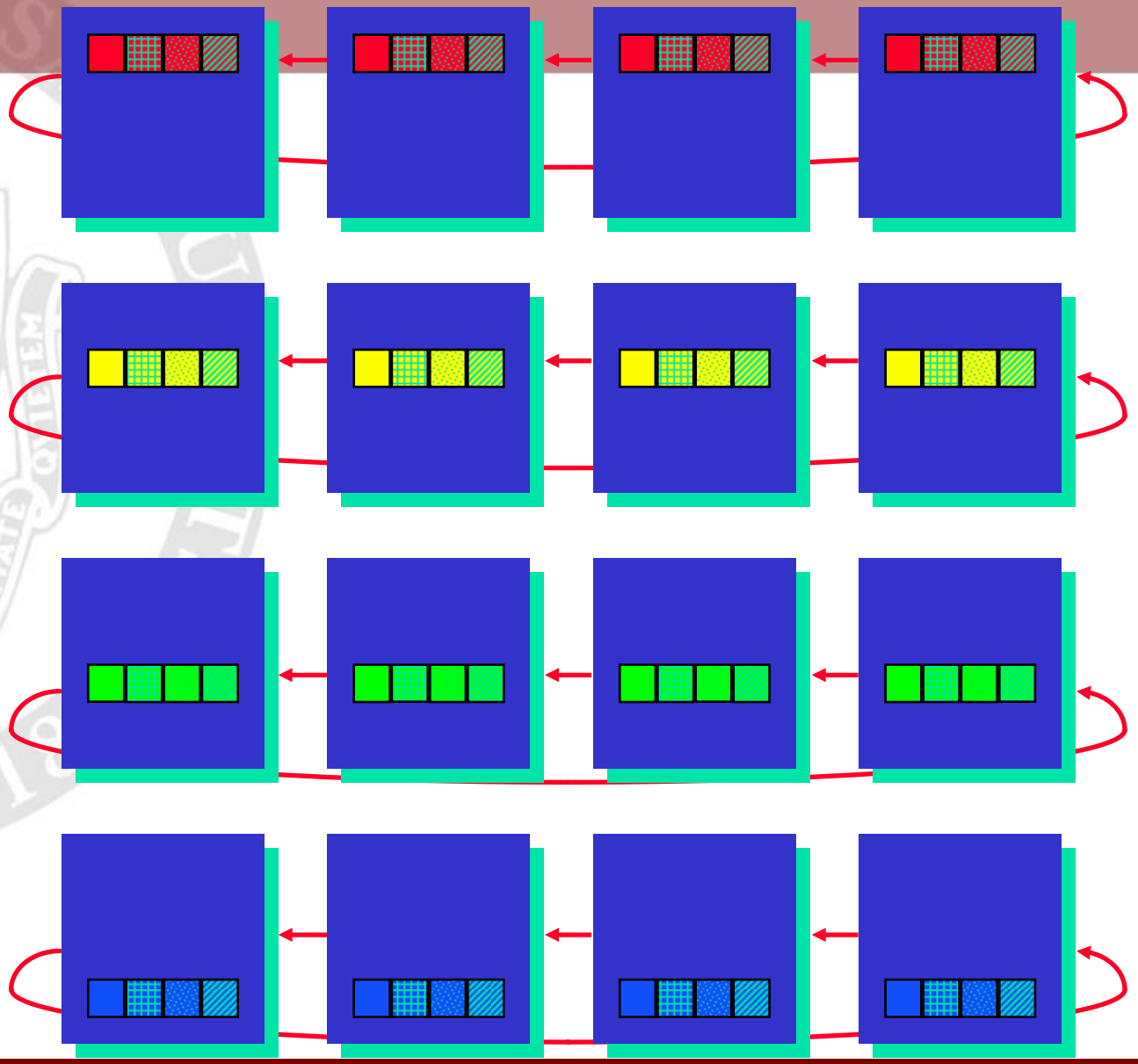


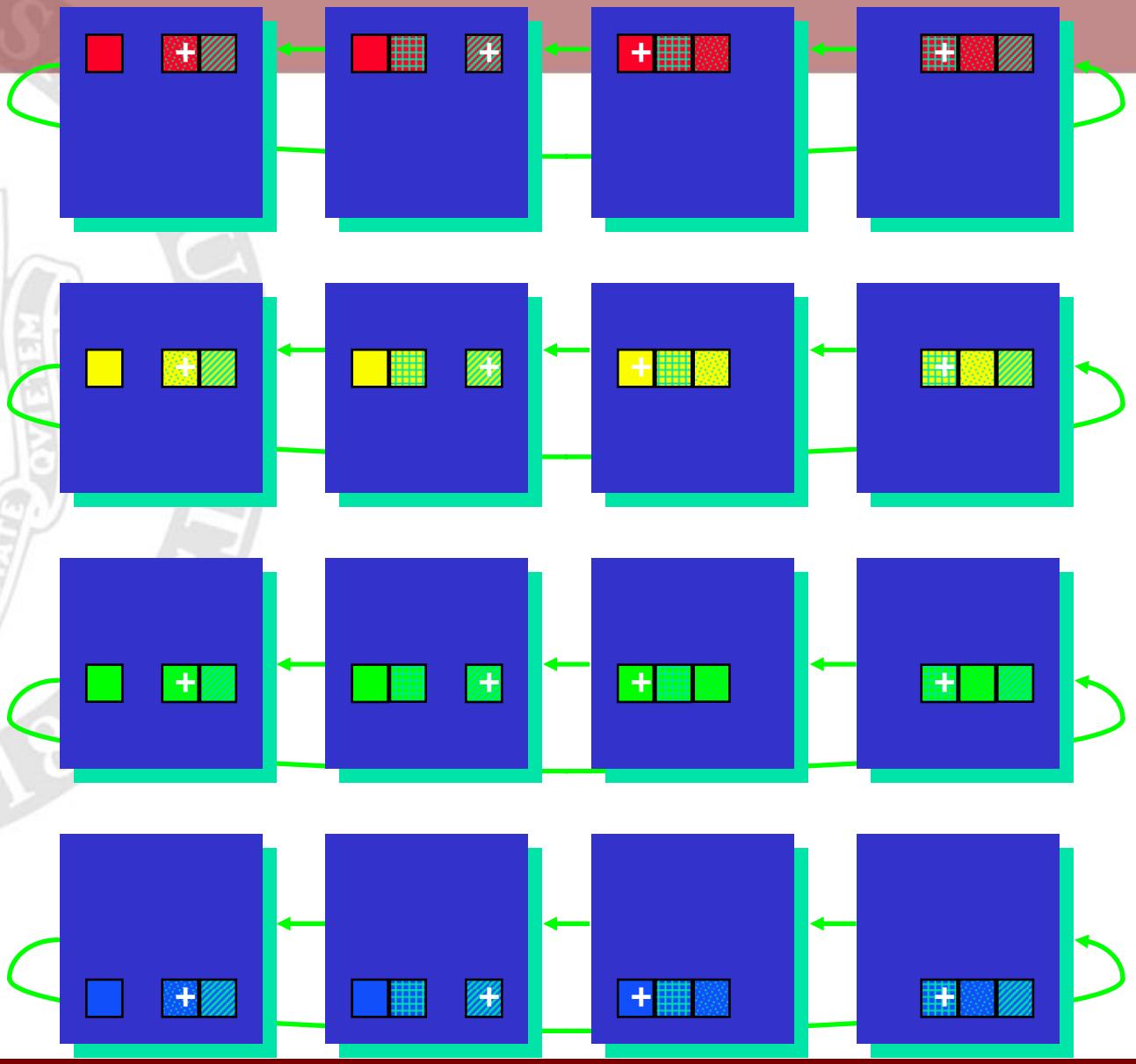


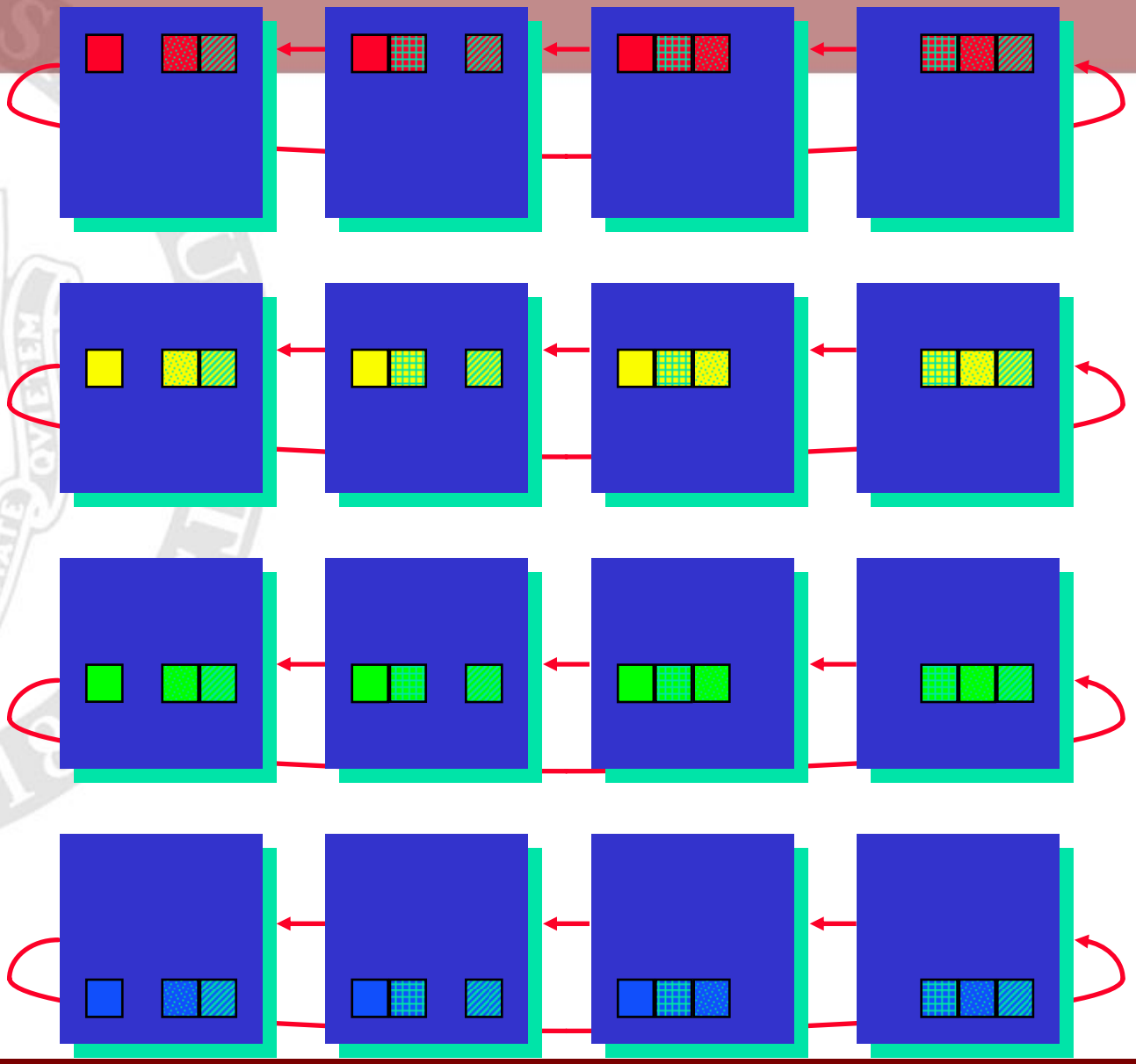


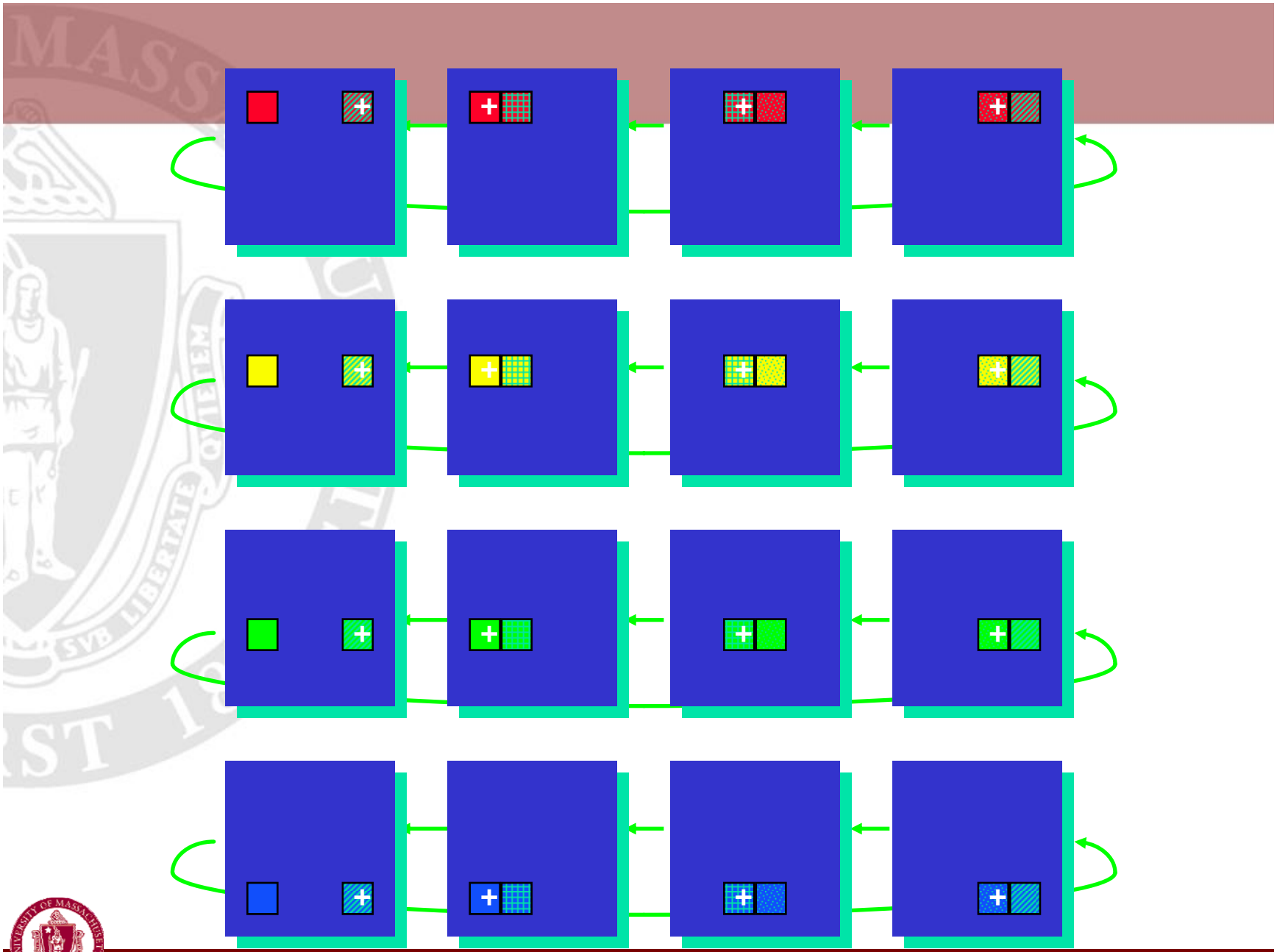


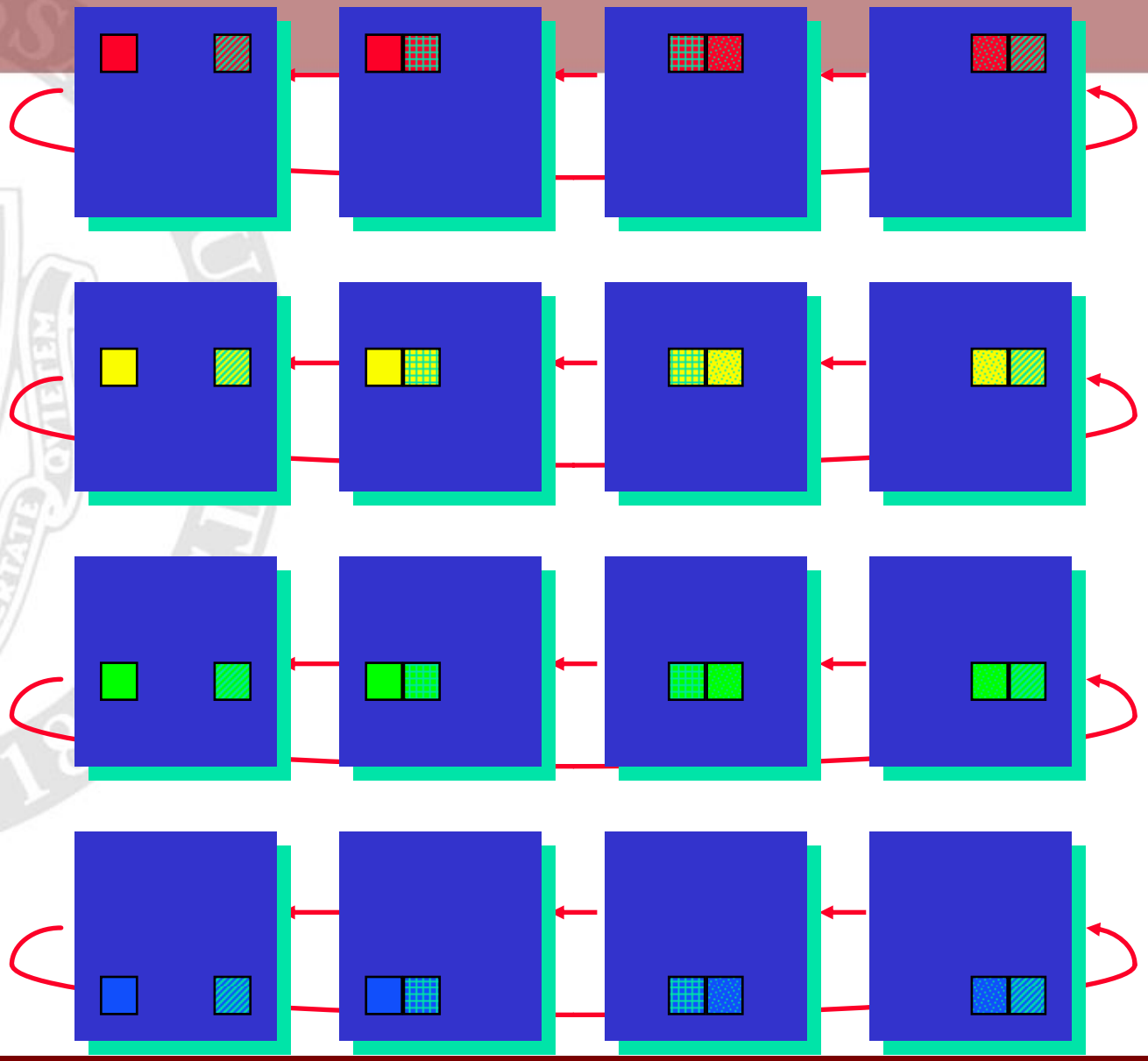


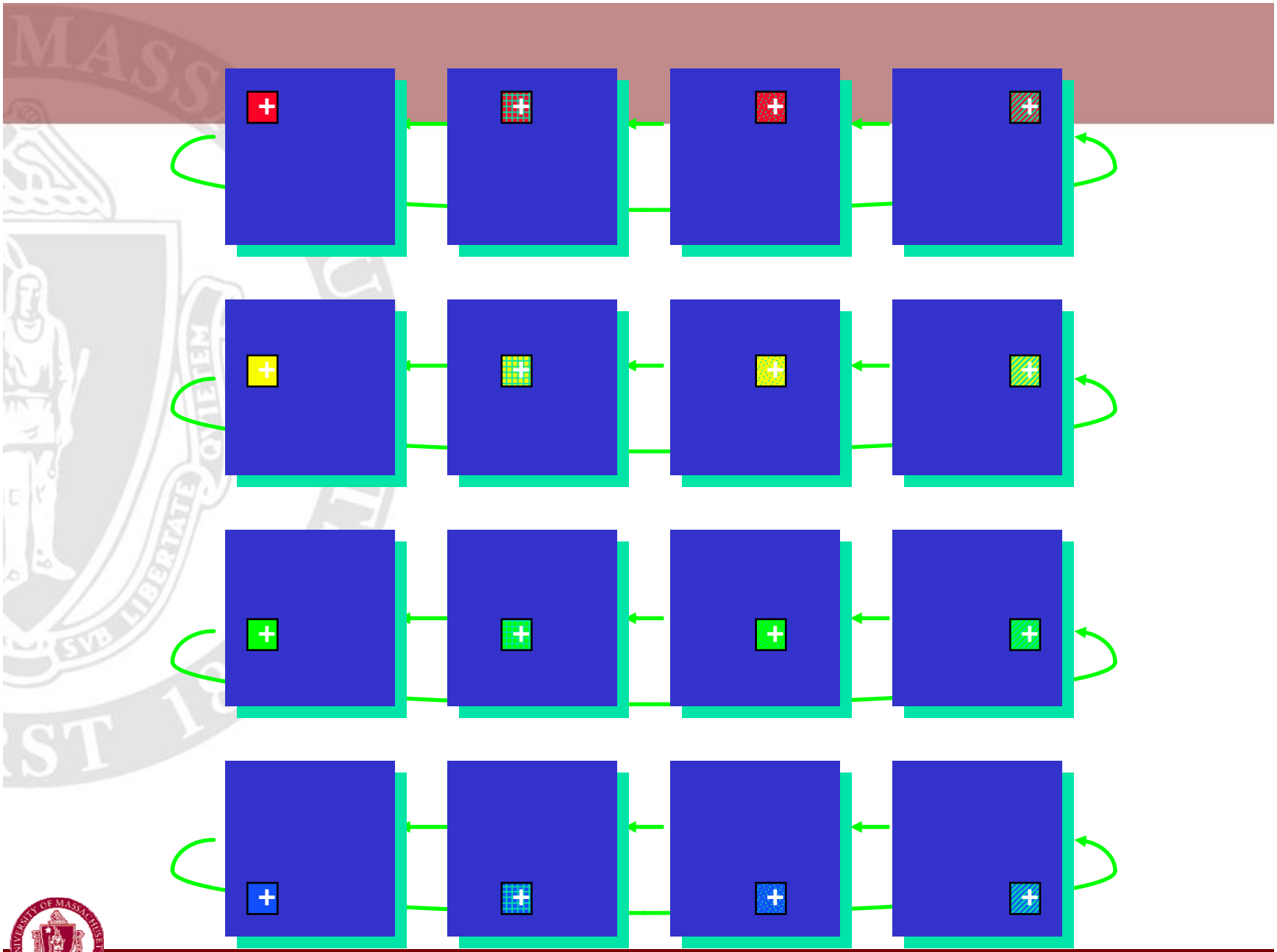


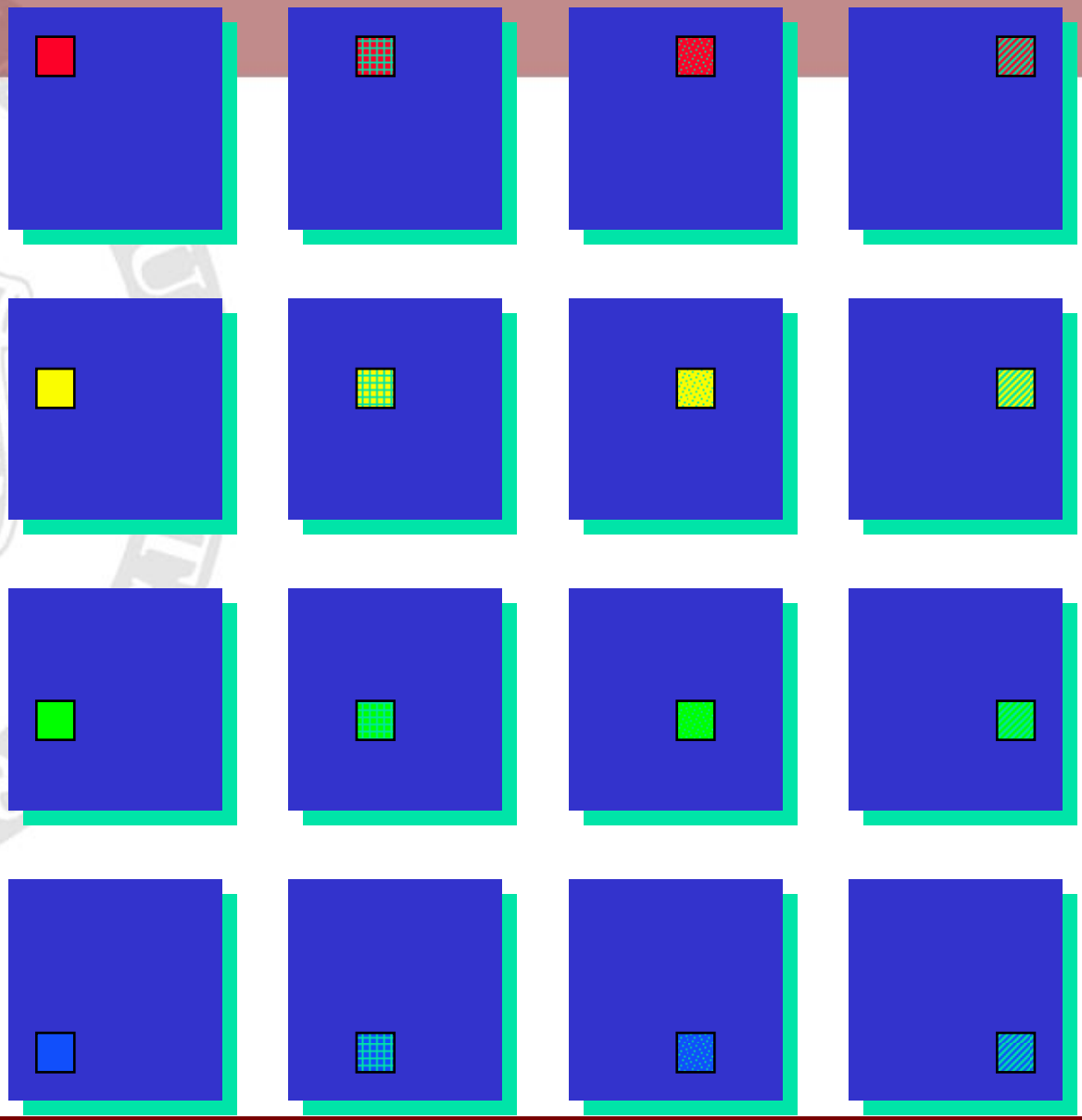












Summary

- **From root task to all others**
  - MPI_Bcast
    (buf, cnt, type, root, comm)
  - MPI_Scatter
    (sendbuf, cnt, type, recvbuf, recvcnt, type, root, comm)
- **From all tasks to root**
  - MPI_Gather
    (sendbuf, cnt, type, recvbuf, recvcnt, type, root, comm)
- **From all to all**
  - MPI_Alltoall
    (sendbuf, cnt, type, recvbuf, recvcnt, type, comm)
  - MPI_Allgather
    (sendbuf, cnt, type, recvbuf, recvcnt, type, comm)
- **Reductions**
  - MPI_Reduce (sendbuf, recvbuf, cnt, type, op, root, comm)
  - MPI_Scan (sendbuf, recvbuf, cnt, type, op, comm)

Communicators

- Can **duplicate**:
  - **MPI_Comm_dup** (old, &new)
- Can then **split** communicators:
  - **MPI_Comm_split** (old, **color**, **key**, &new)
  - Same color ⇒ same communicator
  - Key determines rank in new

	0	1	2	3	4
color	0	0	1	1	1
key	3	2	1	1	1
new rank	1	0	0	1	2

The End

- Next time:
  - OpenMP