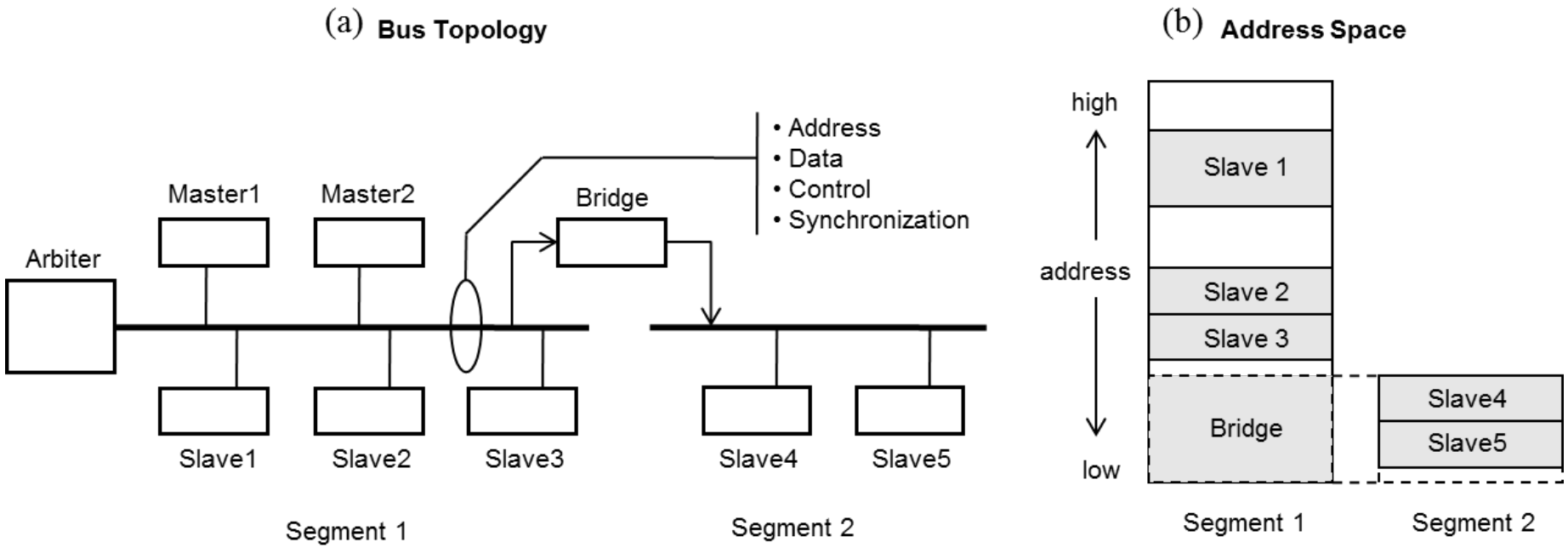
System-on-Chip Design
On-Chip Buses

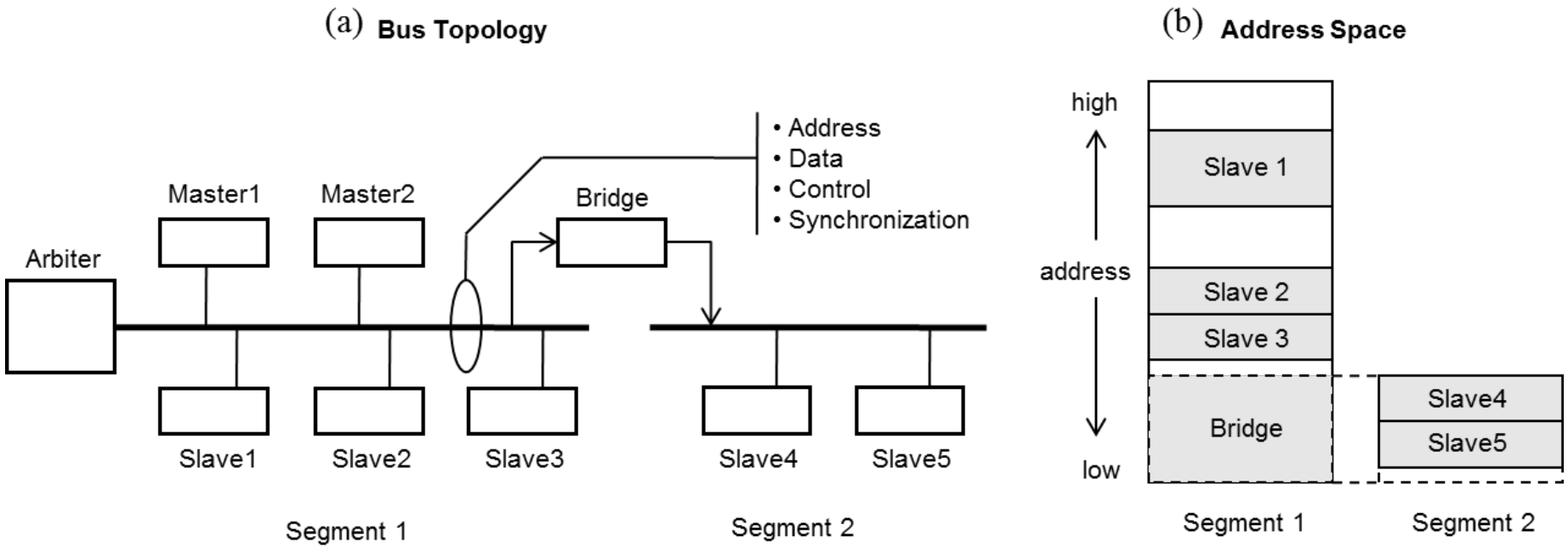
Hao Zheng
Comp Sci & Eng
U of South Florida

Elements of a Shared Bus



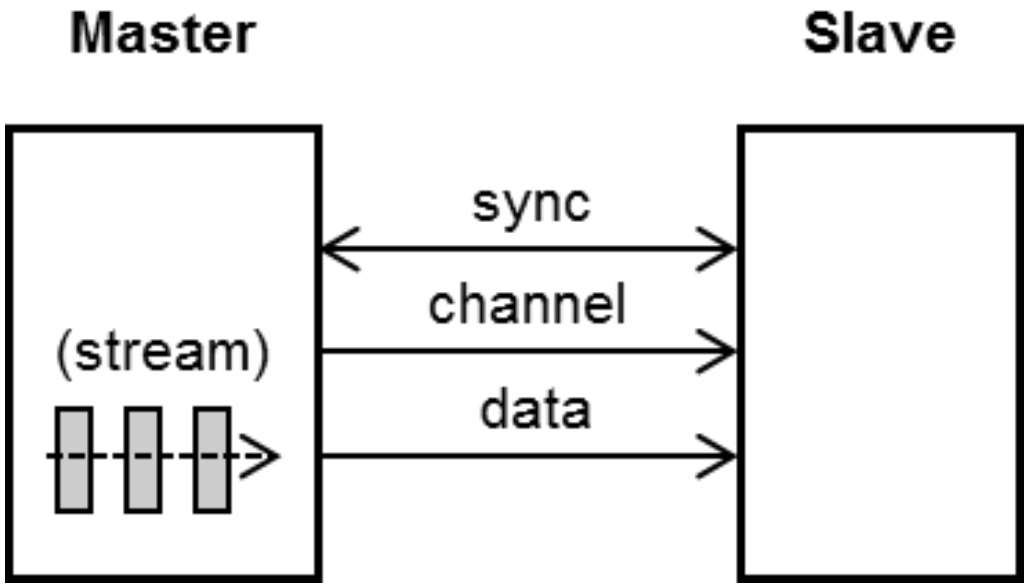
- Segments connected by bridges.
- **Bridges** convert transactions on one segment to transactions on another segment
- **Masters** initiate transactions that **slaves** respond.
- **Arbiter** selects a master to control the bus fairly.
- Address space assigns an unique address to each device.

Elements of a Shared Bus



- **Address** wires carry memory addresses of the target slaves.
- **Data** wires carry data to or from slaves.
- **Command** wires carry operations to be performed by slaves.
- **Synchronization** wires are used for synchronization.

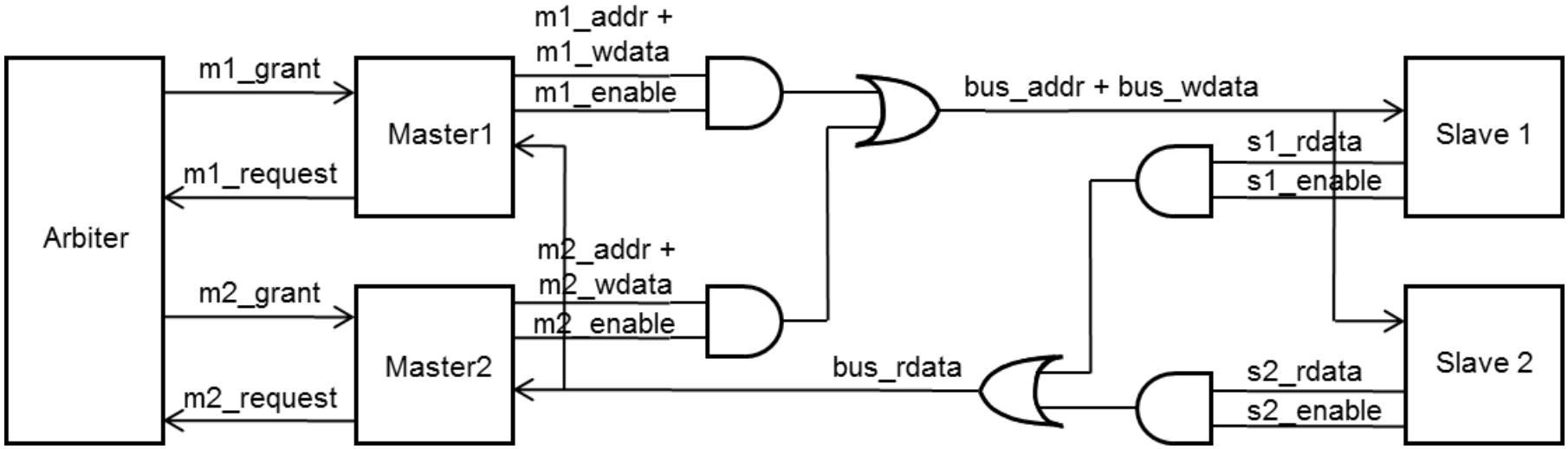
Elements of a P2P Bus



Channels allow simulation of multiple ports using a single port.

No need for address wires.

Physical Connection of Buses



Write: data flow from a master to a slave.

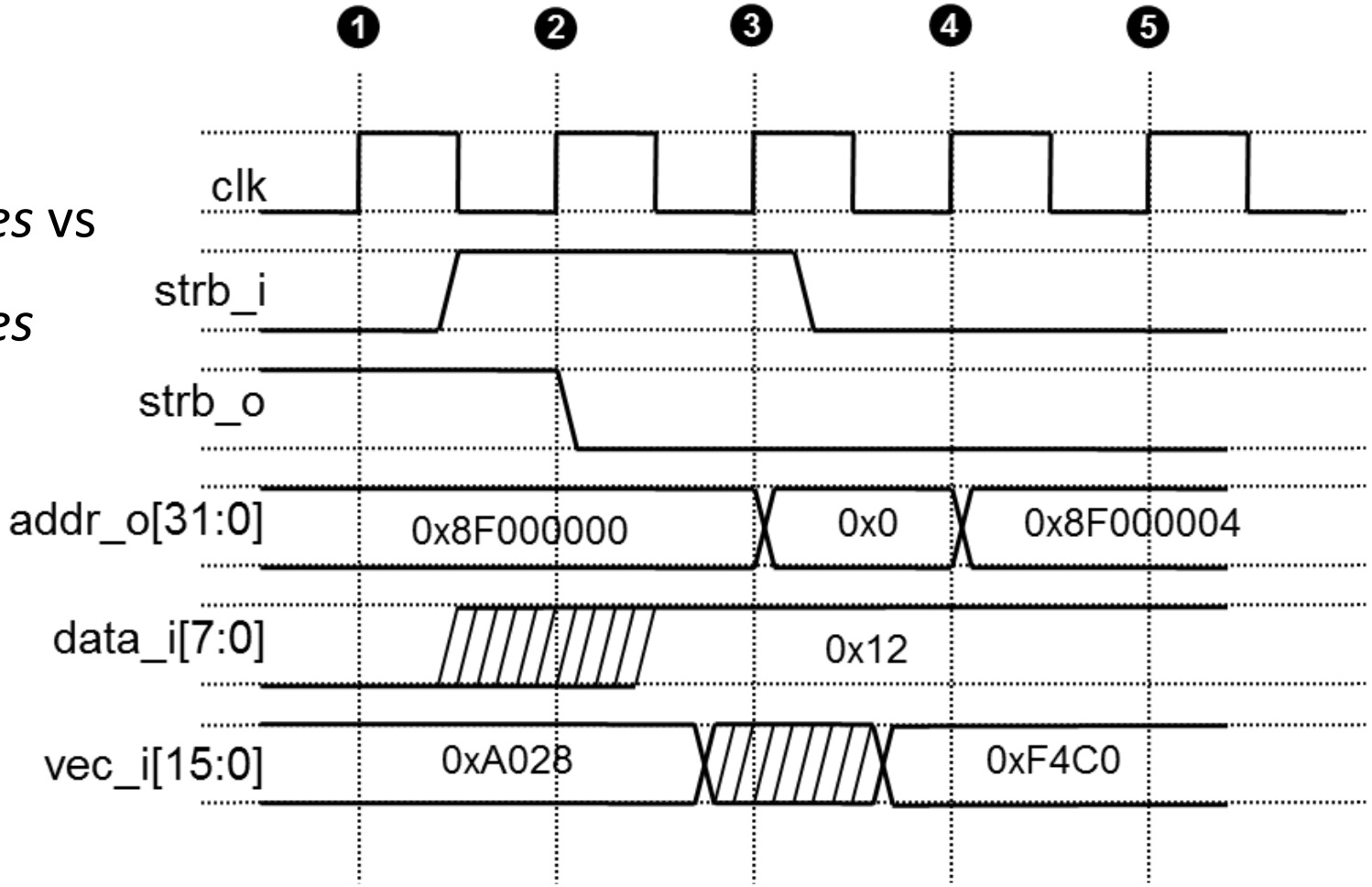
Read: data flow from a slave to a master.

Separate read & write channels allow concurrent operations.

Bus Timing Diagrams

clock edges vs

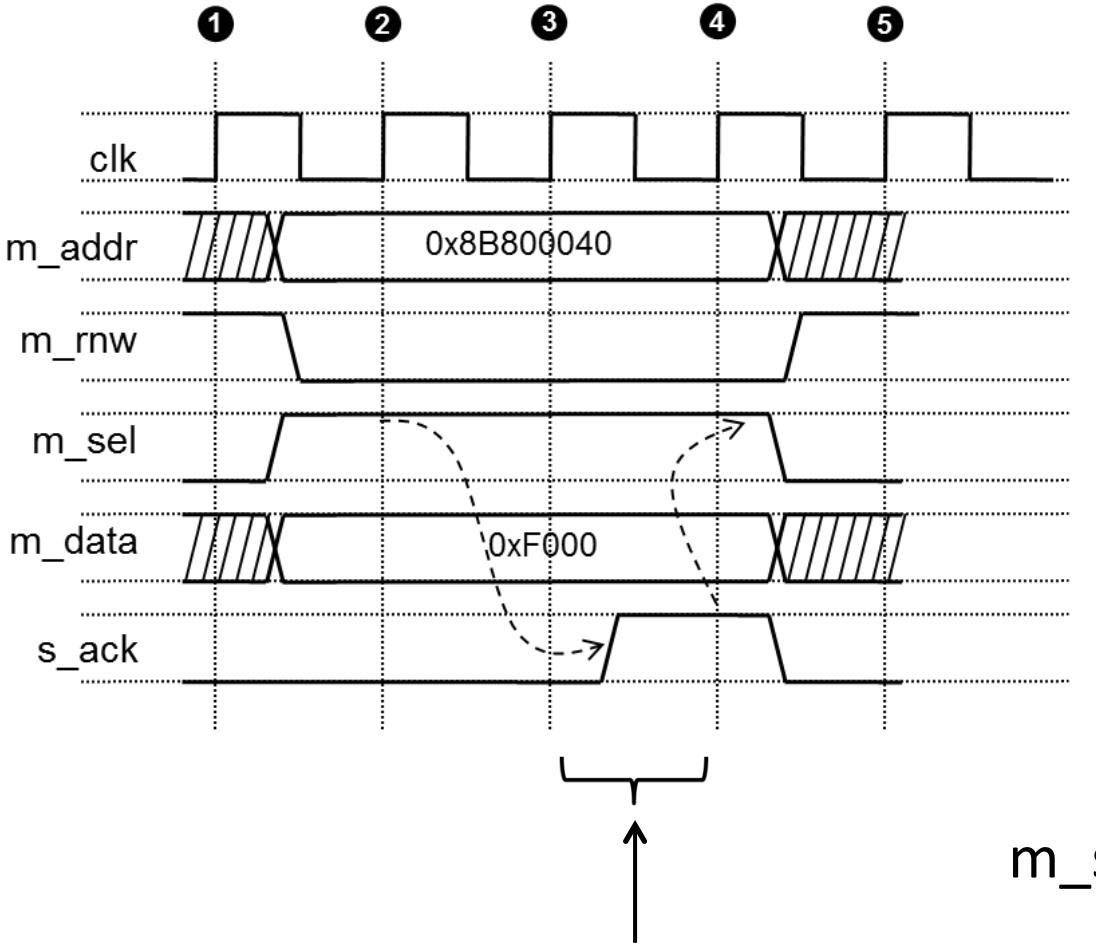
clock cycles

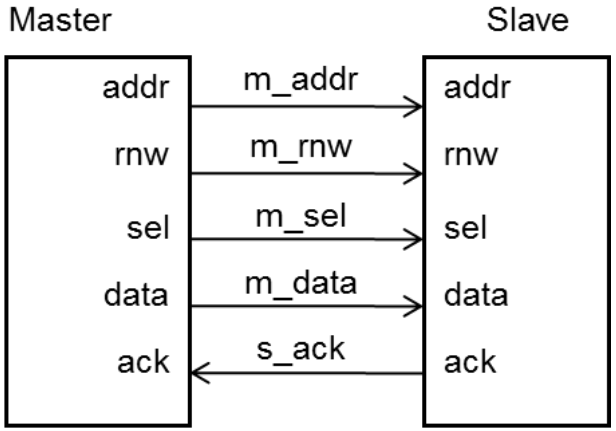


For input *i*, it's high in cycle *n* if *i* is high before the clock edge *n*.

For output *o*, it's low in cycle *n* if *o* is low after the clock edge *n*.

Basic Write Transfers (10.2.1)



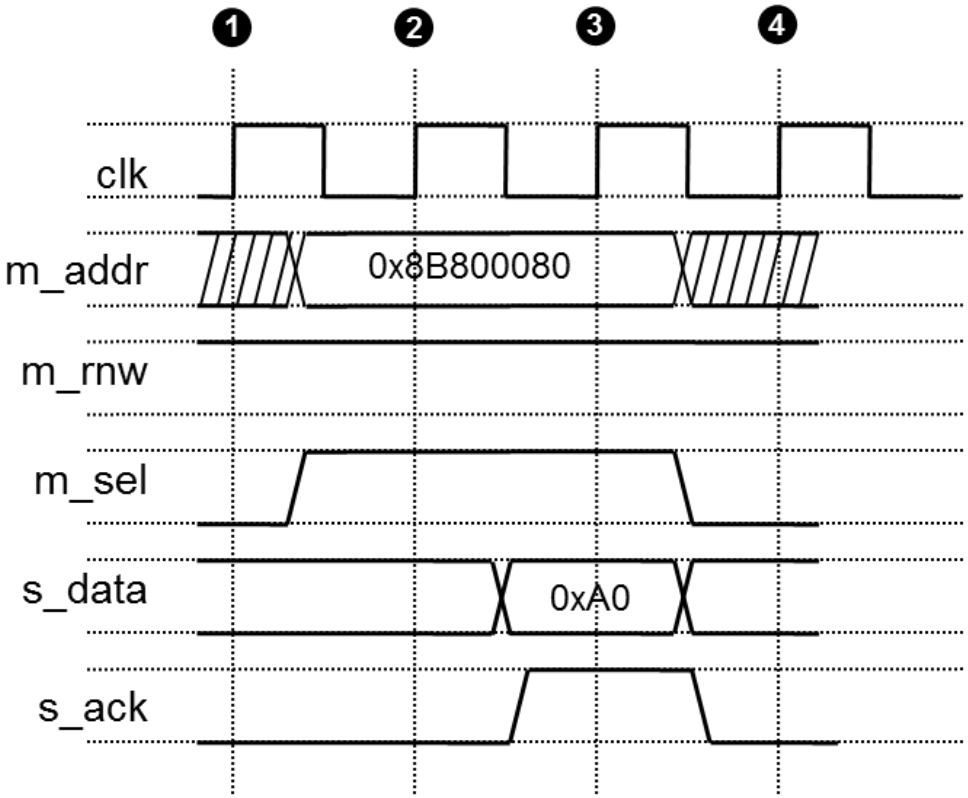


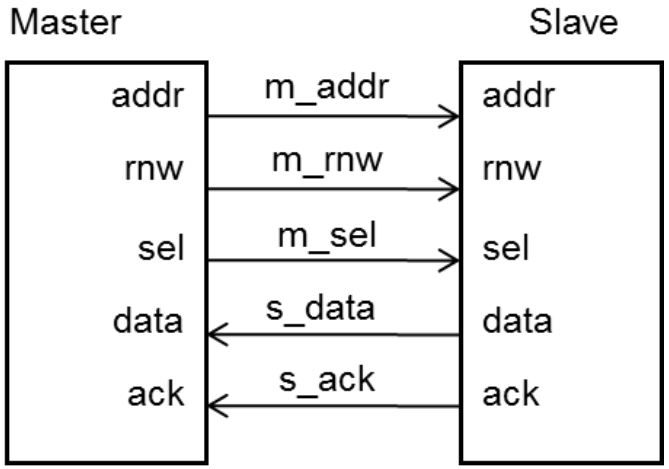
m_sel: transfer validity signal

wait state: hurt bus performance

Time-out is needed for slow slaves.

Basic Read Transfers (10.2.1)

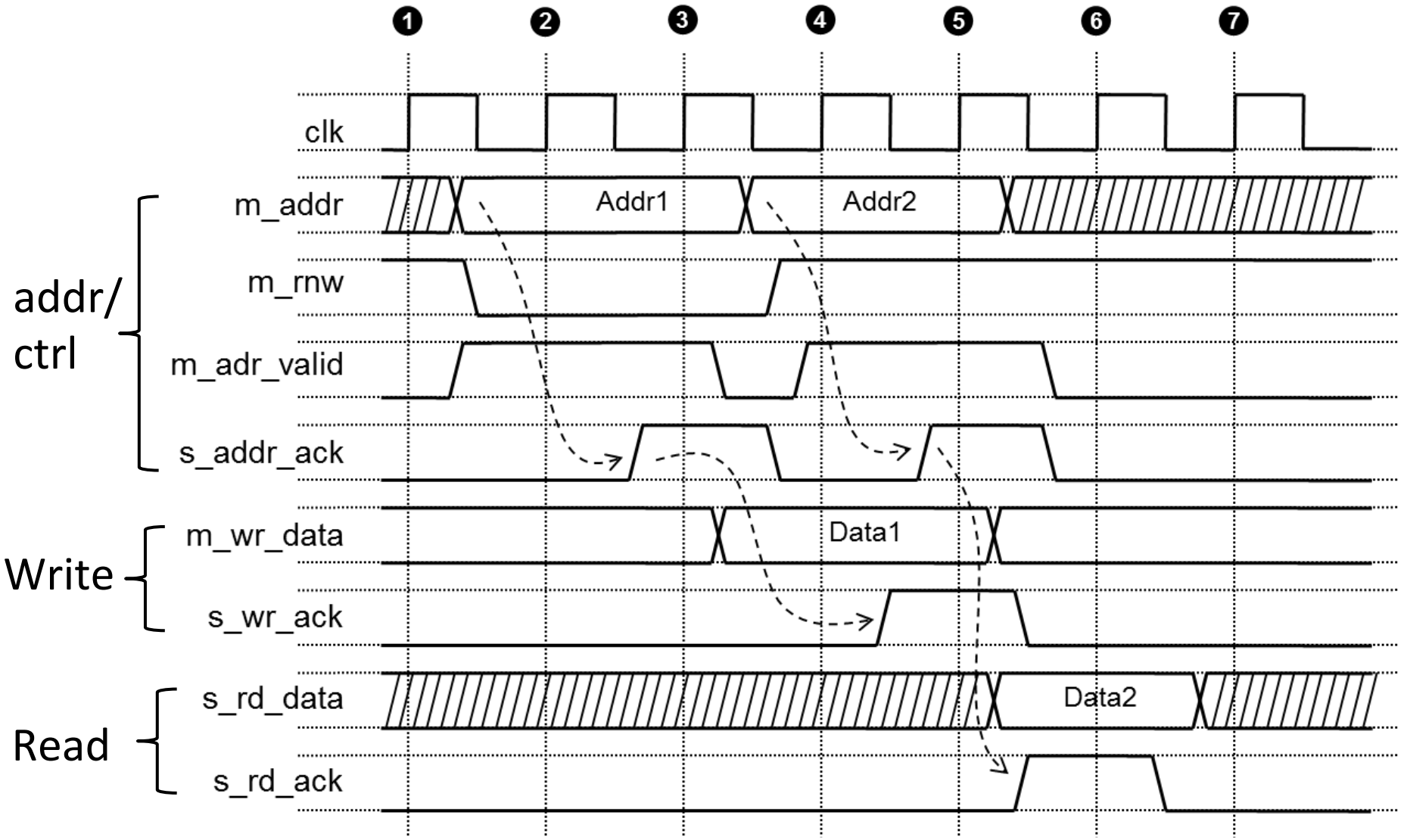




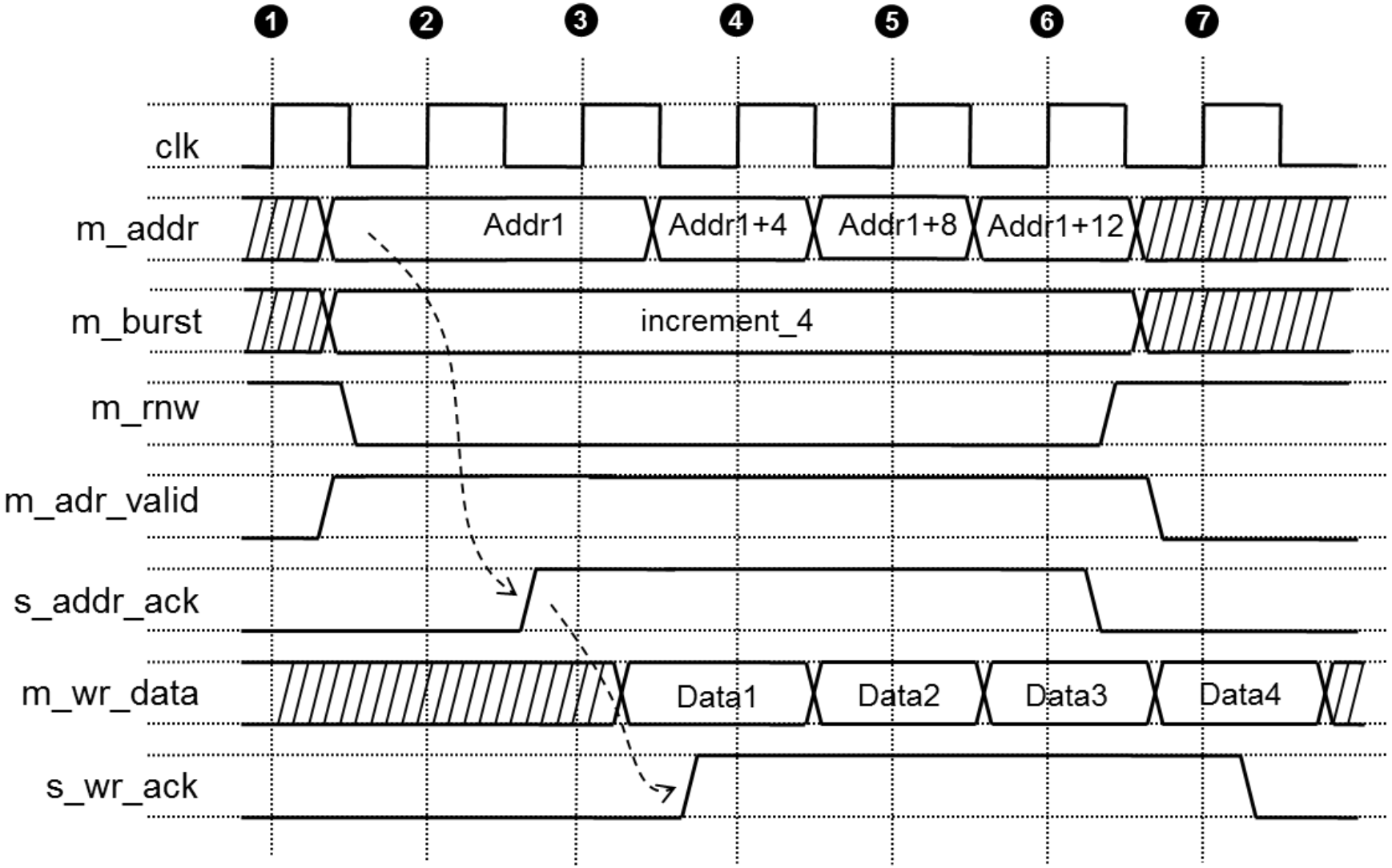
Improved Bus Transfers (10.2.3)

- Each data transfer has multiple phases in sequence.
  - Master gets bus access by negotiating with bus arbiter.
  - Master issues address/data/command/control.
  - Slave acknowledges the transfer.
  - Master releases the bus.
- Optimizations:
  - *Transaction splitting and pipelining transfers*
  - *Burst-mode operation*

Transaction Splitting and Pipelining Transfers

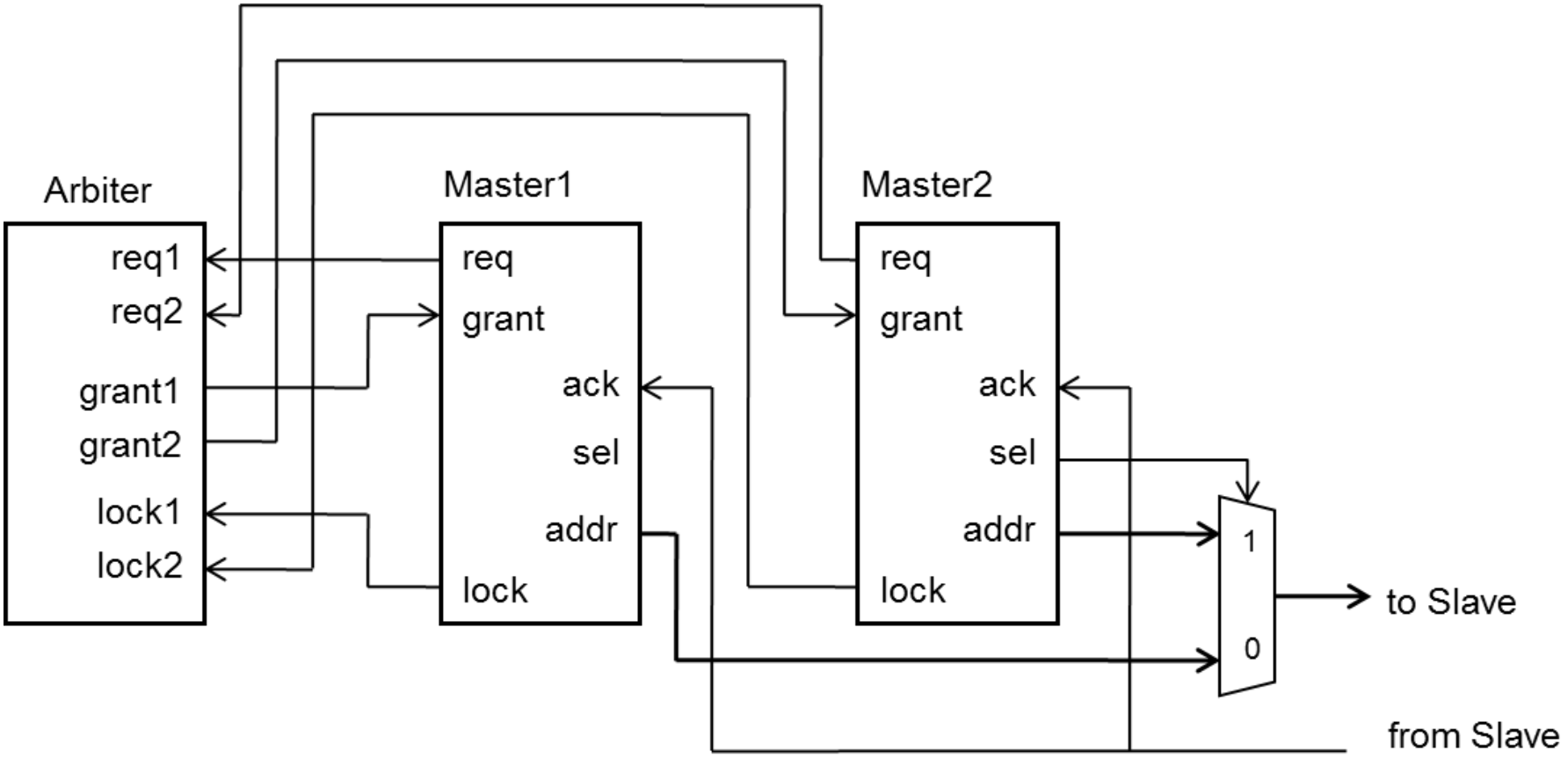


Burst-Mode Transfers



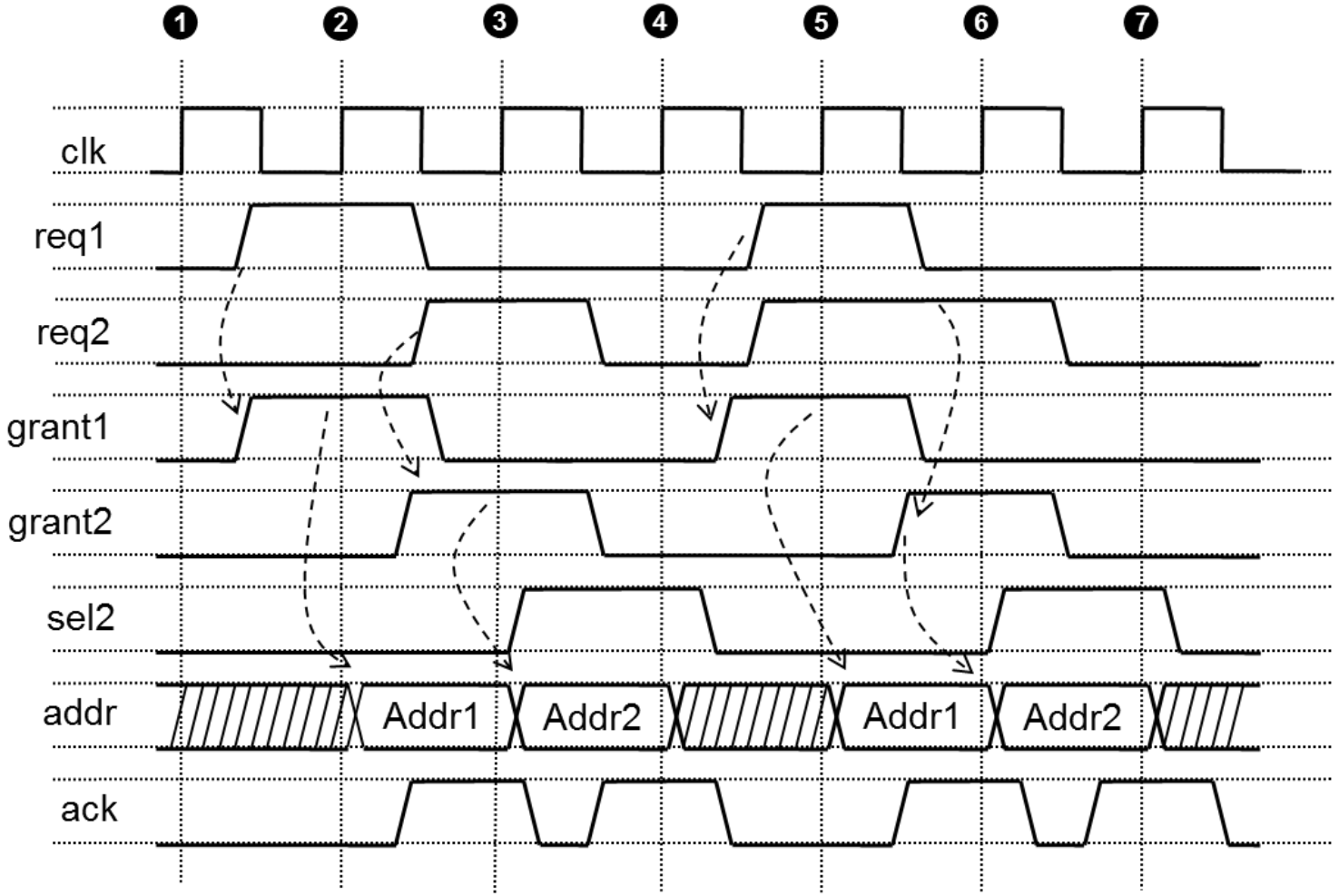
One communication, multiple data transfers, reduced overhead.

Multi-Master Bus Systems (10.3)



A master must talk to the arbiter first before it can communicate with a slave.

Multi-Master Bus Systems: Timing



Bus priority: should prevent starvation.

Multi-Master Bus Systems: Bus Locking

- Locking ensures exclusive access of bus for certain duration of time.
  - Transfer of low priority master cannot be interrupted by the request from a high priority master.
  - Need of an atomic sequence of transfers.
  - Ensure latency requirements.

Multi-Master Bus Systems: Bus Locking

```
int *mutex = (int*) 0x8000;

int test_and_set() {
  int a;
  lock_bus();
  a = *mutex;
  *mutex = 1;
  unlock_bus();
  return a;
}
```

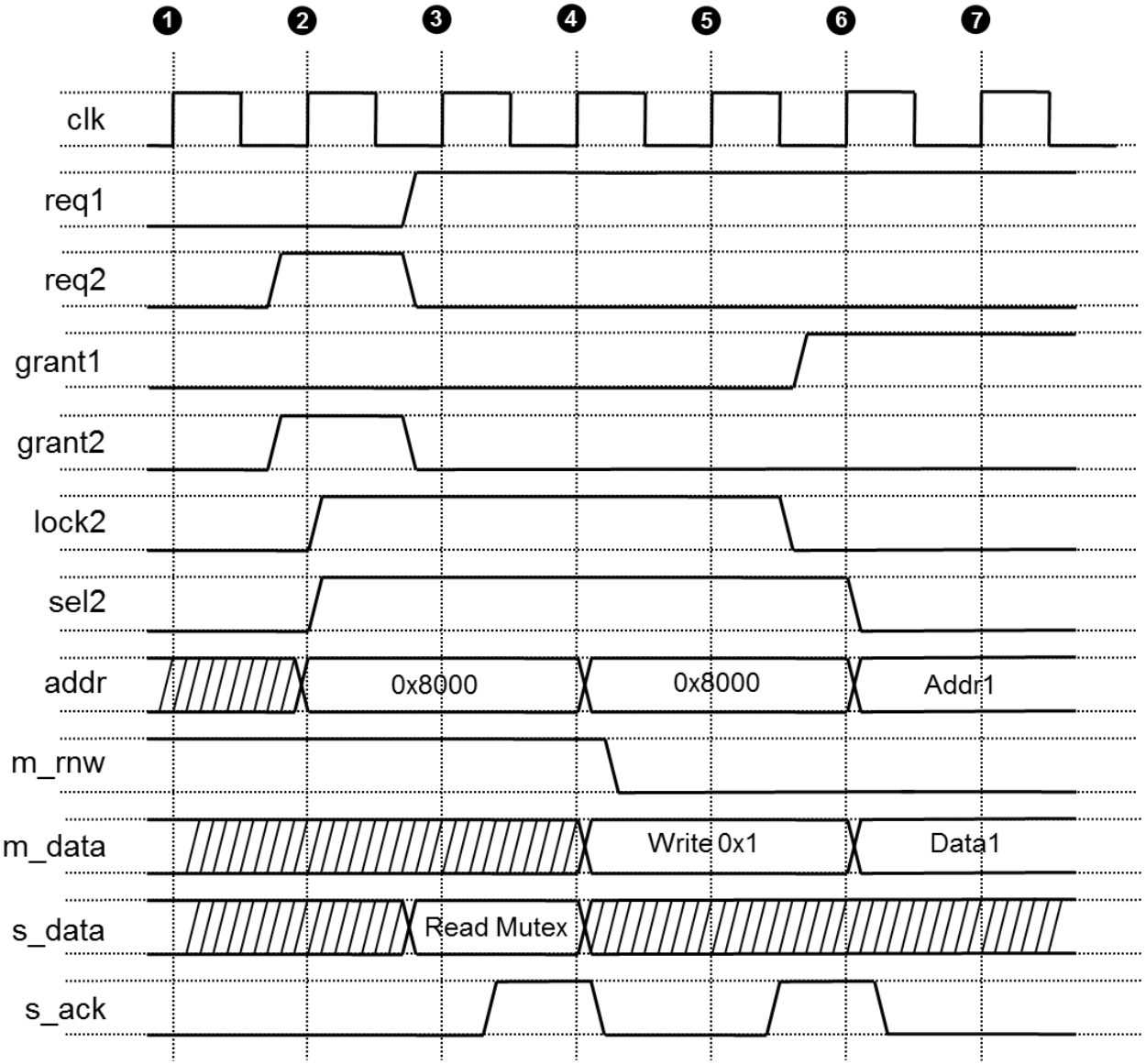
```
void leave() {
  *mutex = 0;
}

void enter()
  while (test_and_set());
}
```

test_and_set(): lock mutex

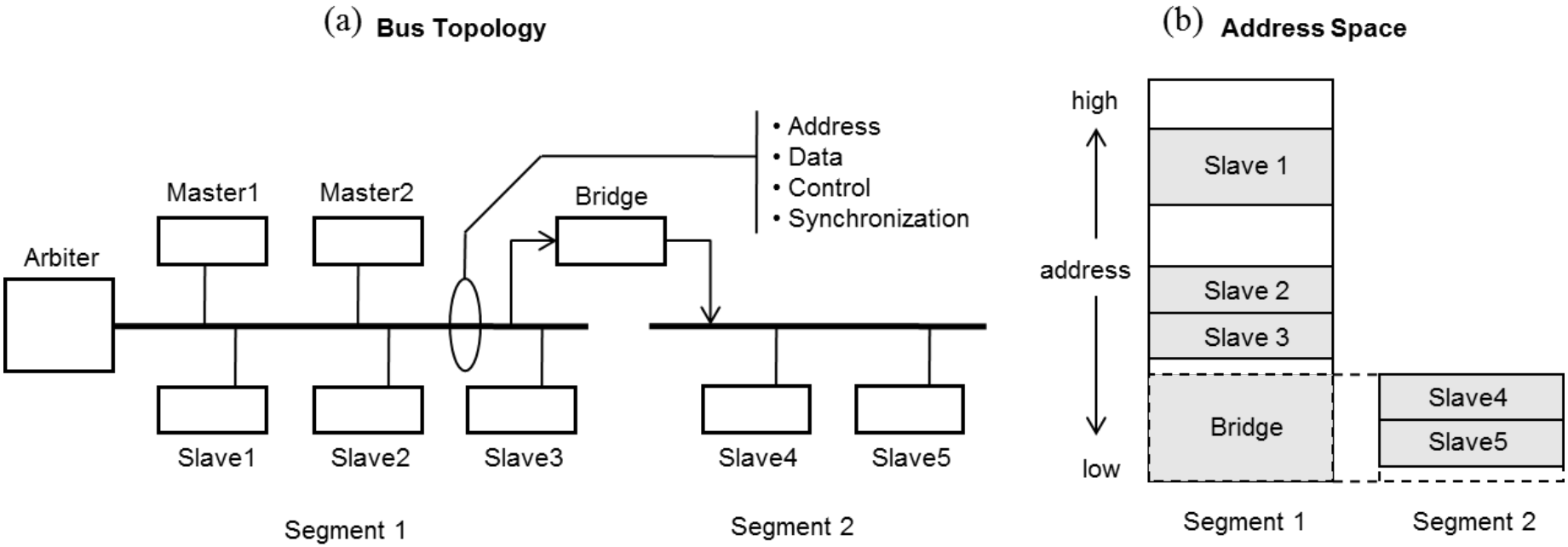
leave(): unlock mutex

Multi-Master Bus Systems: Bus Locking

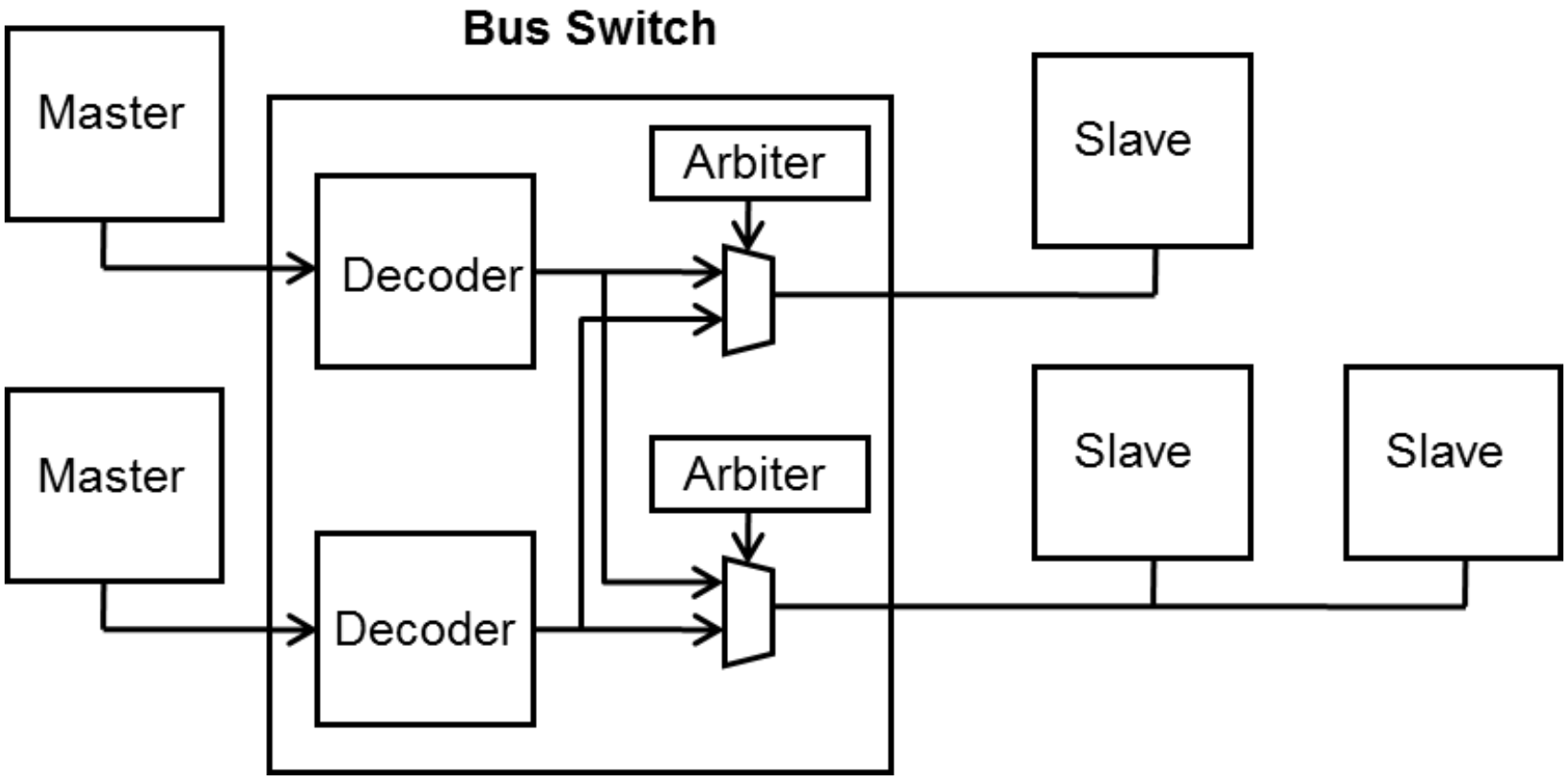


Bus Topologies (10.4)

- Organization of bus components and their connections.
- Parallel transfers on a bus must be sequentialized.
- Bus segments cannot be too long

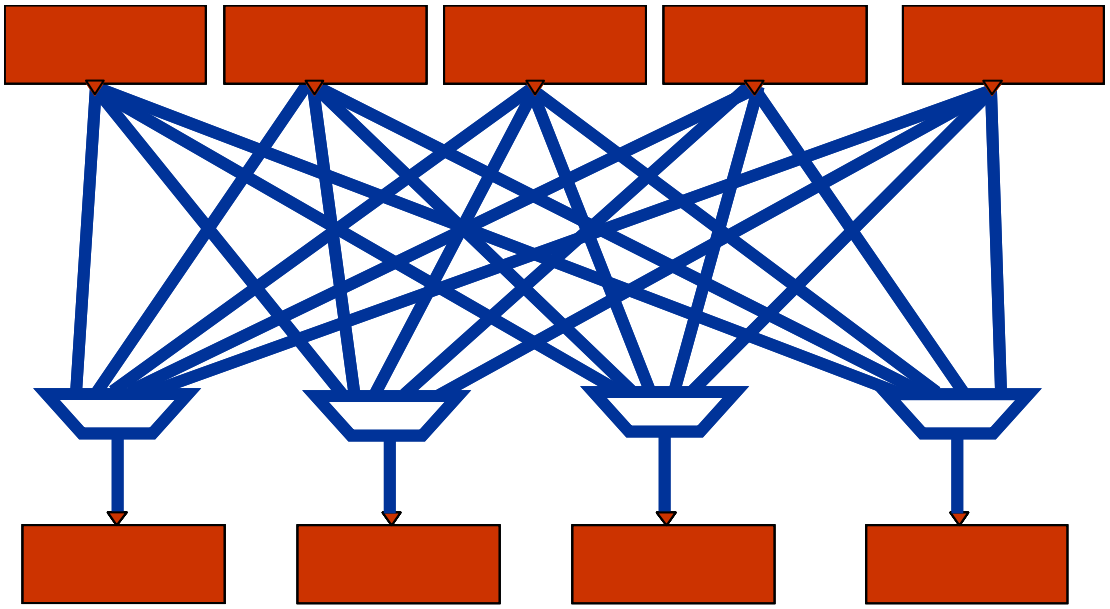


Bus Topologies: Switches



- Masters can transfer to different slaves concurrently.
- Transfers to the same slaves are sequentialized.

Bus Topologies: Crossbar



- Highly parallel.
- expensive to implement,
- Not scalable.

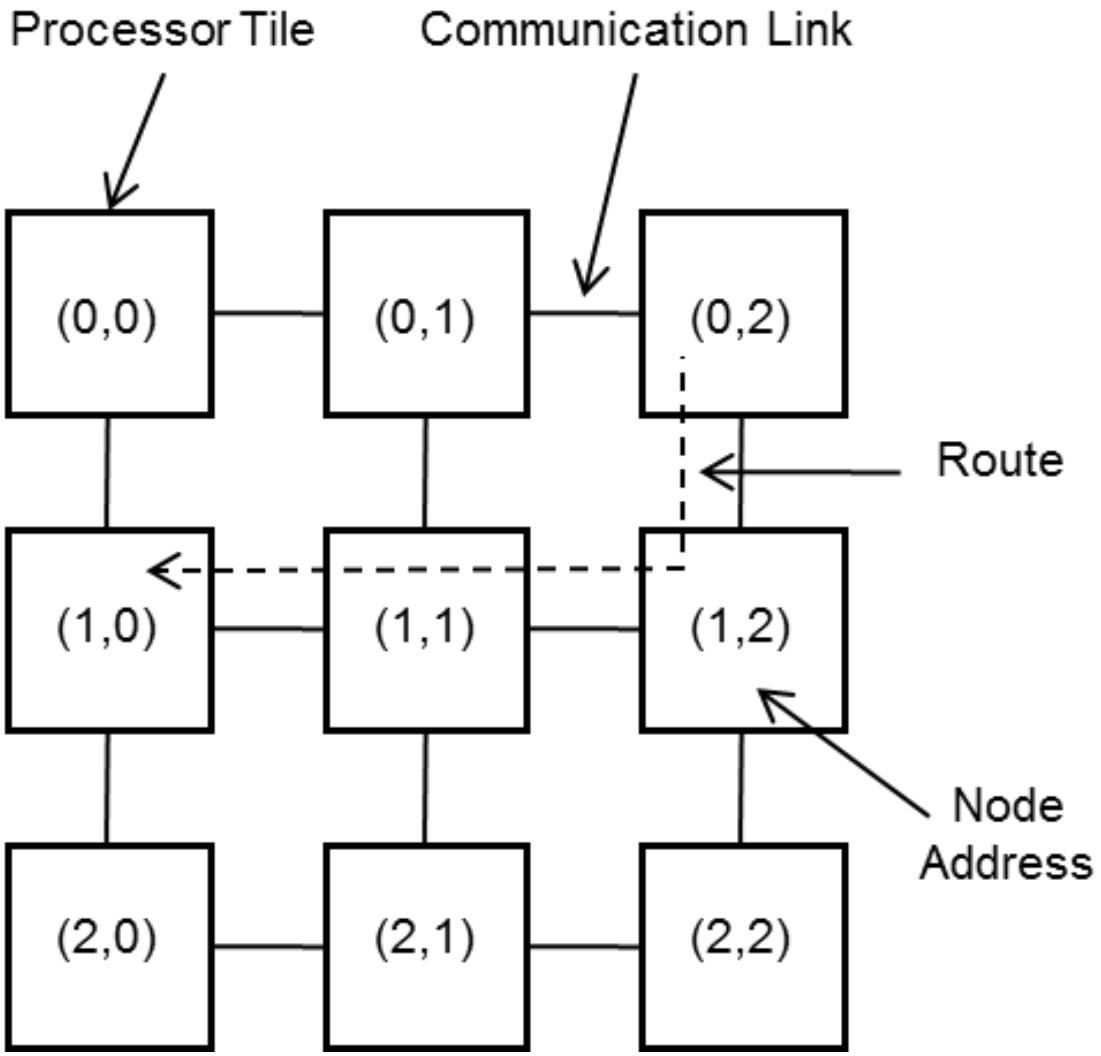
Bus Topologies: Network-on-Chip

The route between nodes are not unique.

Transfer delay less predictable.

Each node implements a routing algorithm to find such a route and reduce congestion.

Much more scalable and parallel.



Reading Guide

- Chapter 10, the *CoDesign* book.
  - Skip 10.2.2