
Digital Design: A Systems Approach

Lecture 15: Interfaces, Interconnect, and Memory

Reading

- L15: Chapters 22, 24, & 25
- L16: Course Review

Today

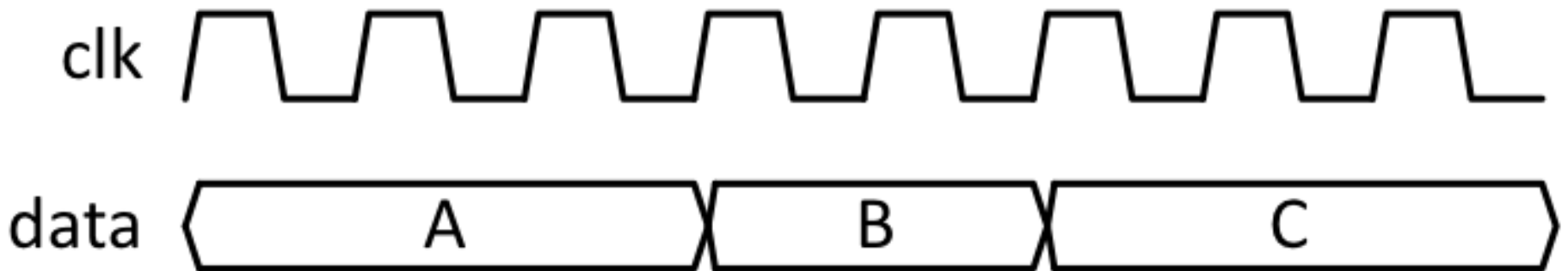
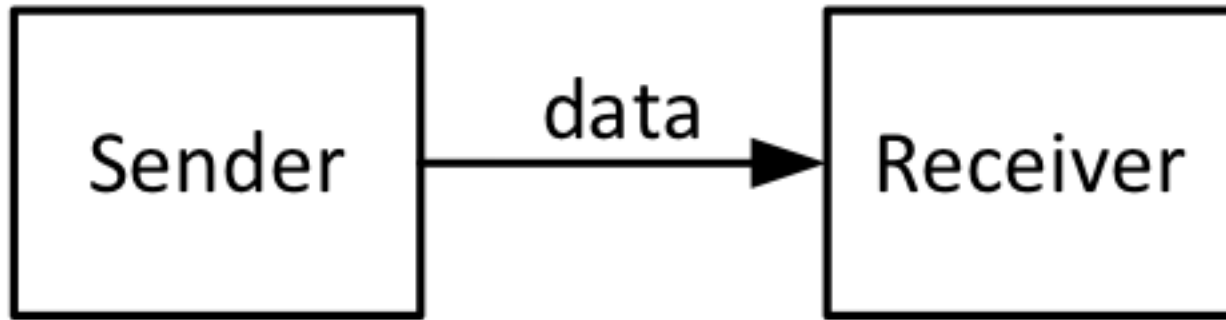
- Interface Timing
- Interconnect
- Memory

Interface Timing

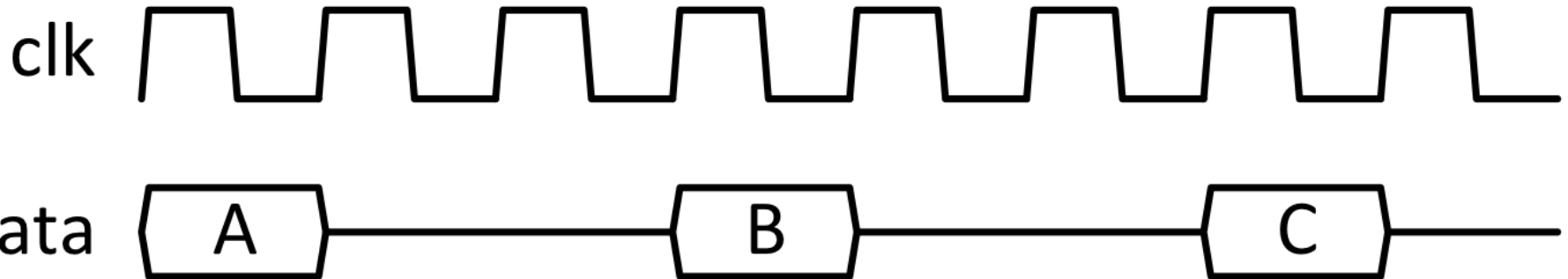
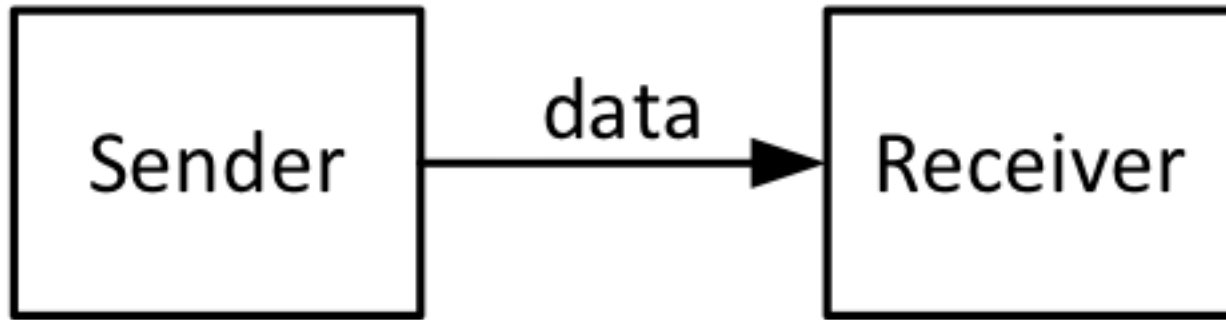
- How do you pass data from one module to another?
 - Open loop
 - Flow control
 - Serialized



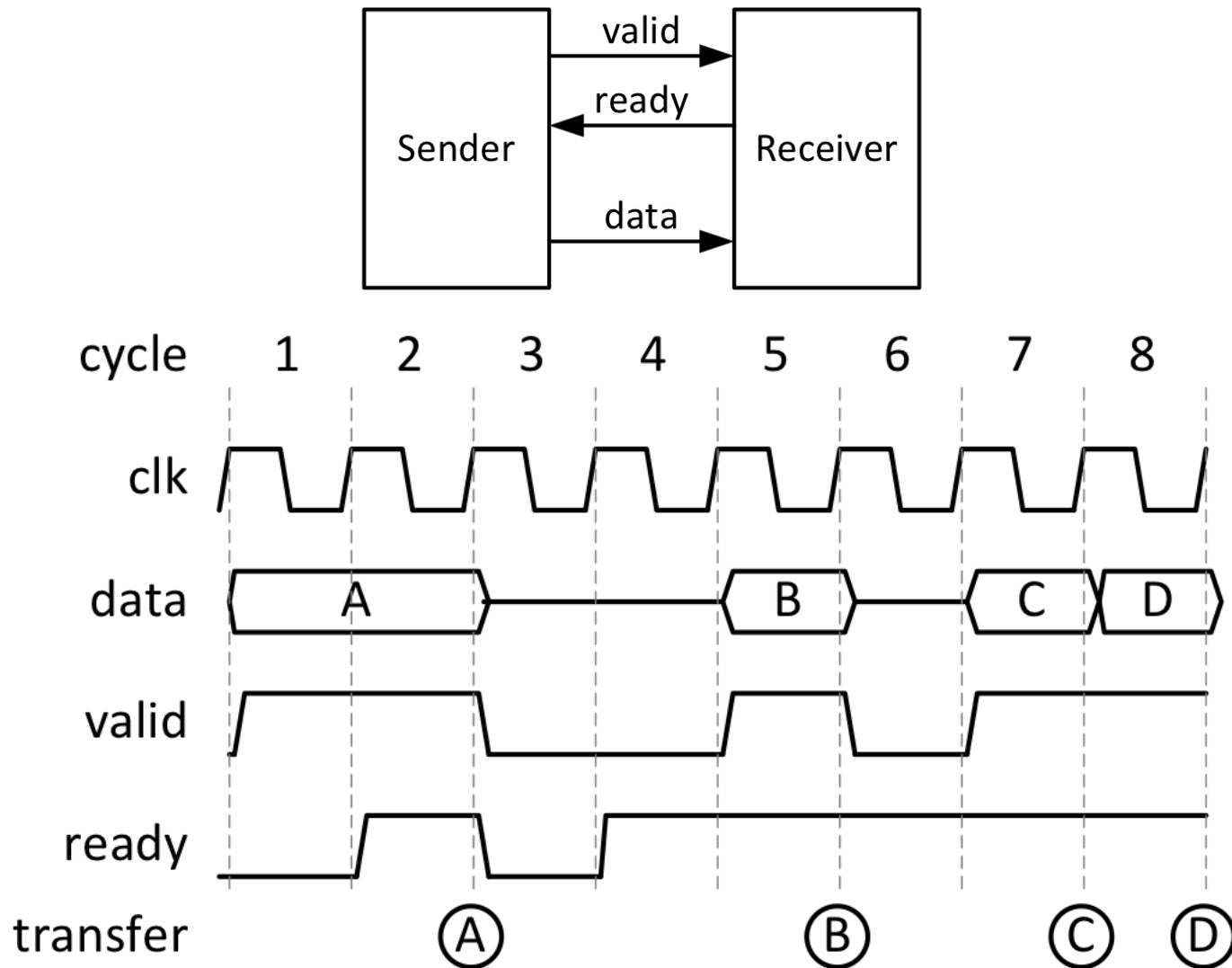
Always Valid Timing



Periodically Valid Timing

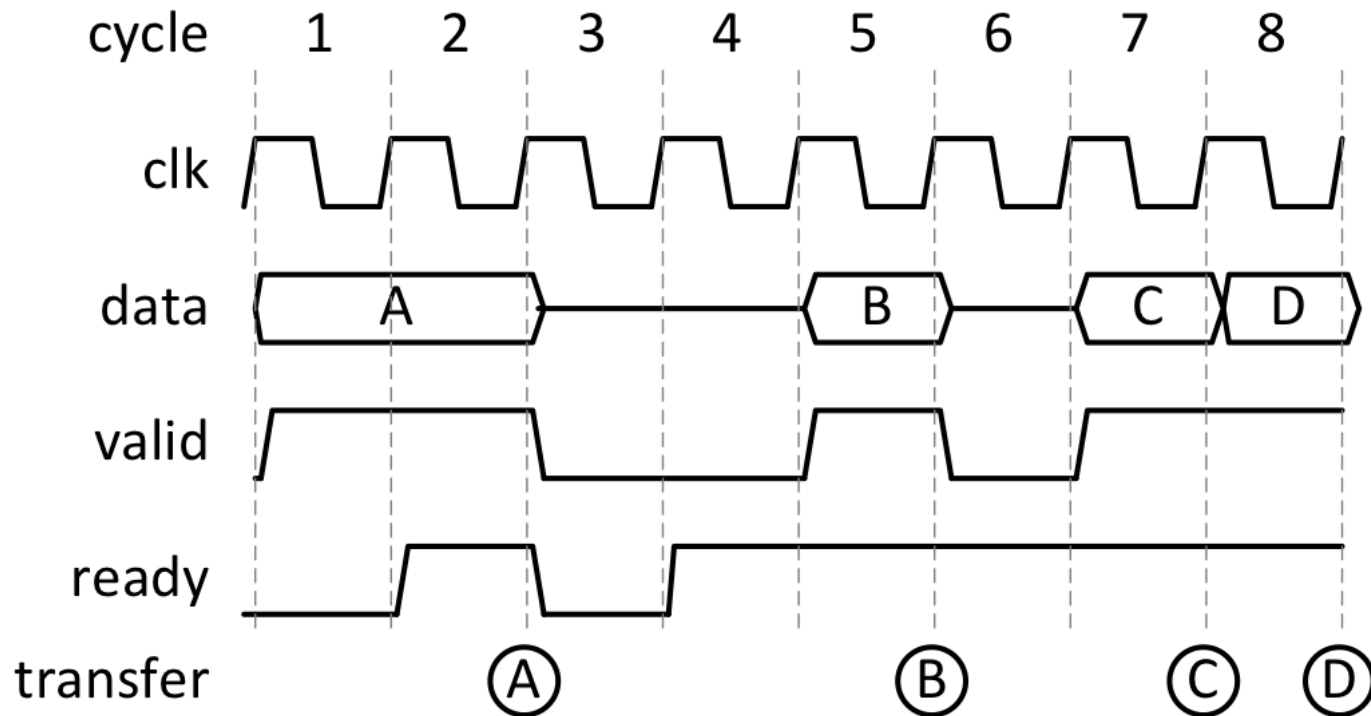


Flow Control

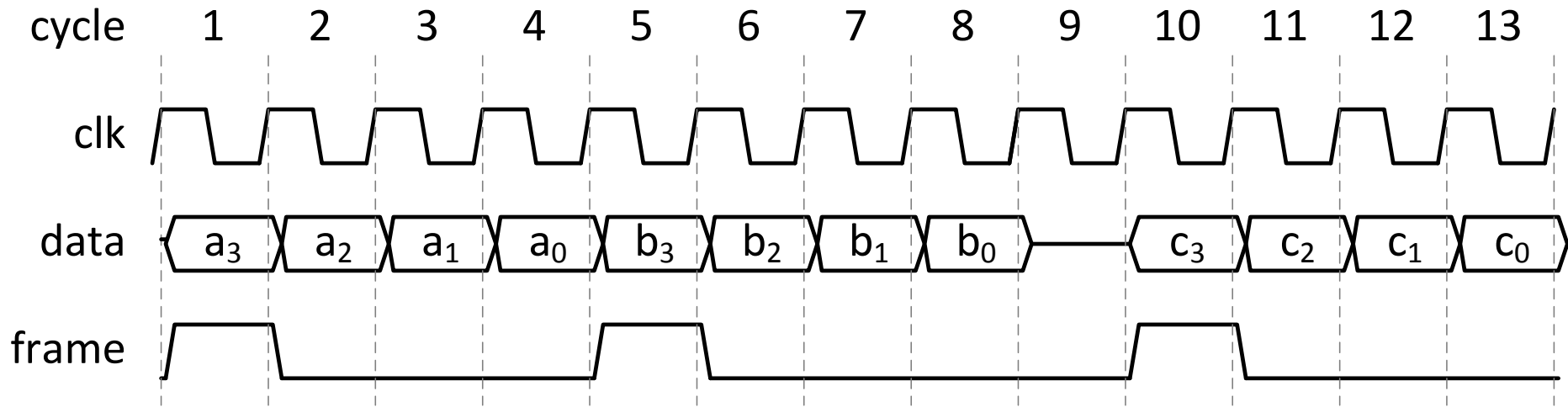


Flow-control

- Valid – Tx has data available
- Ready – Rx able to take data
- Push flow control – assume Rx always Ready
- Pull flow control – assume Tx always Valid



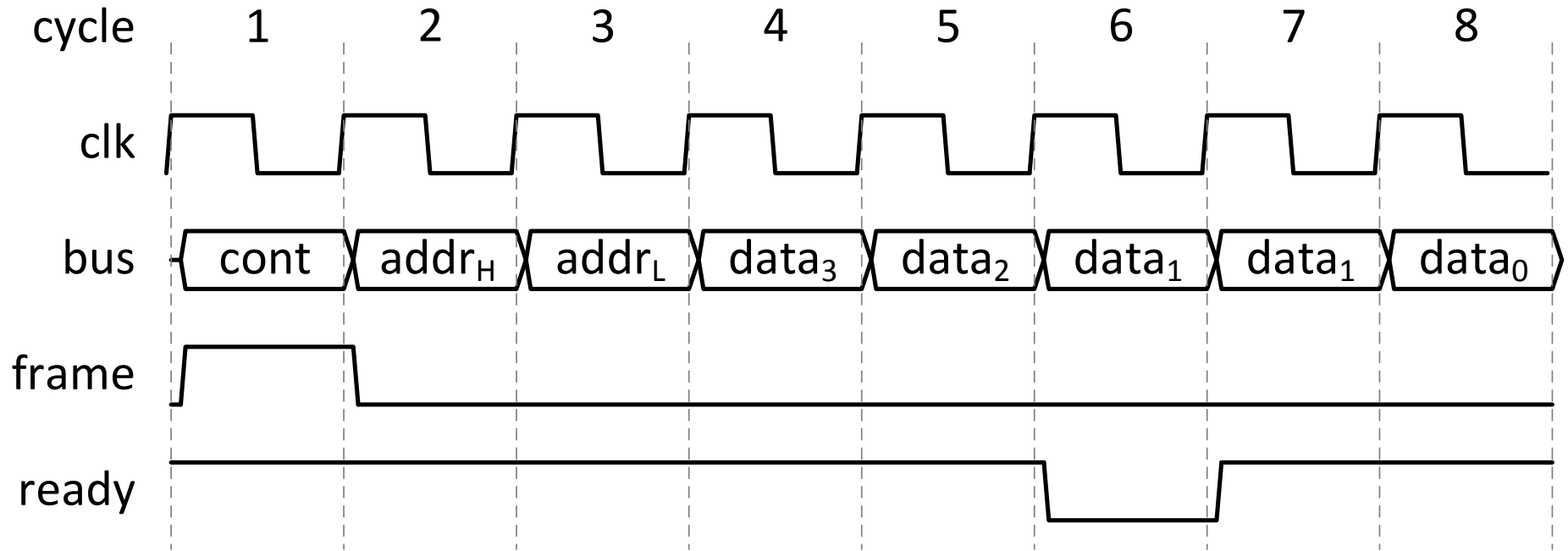
Serialization



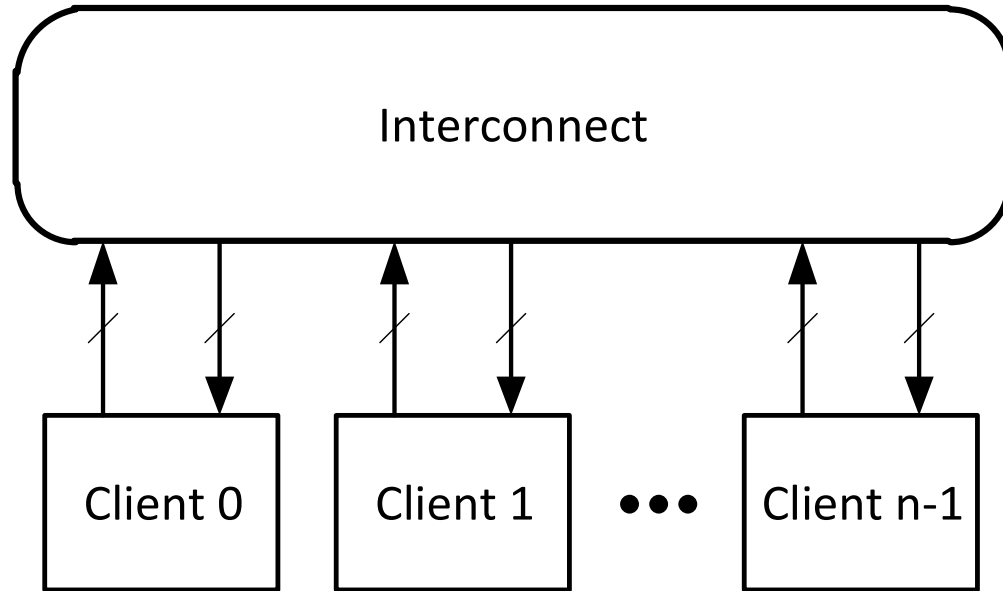
Frame signals start of new serial frame (in this case 4 words)

Flow control can be at frame granularity or word granularity

Serialization with word granularity FC

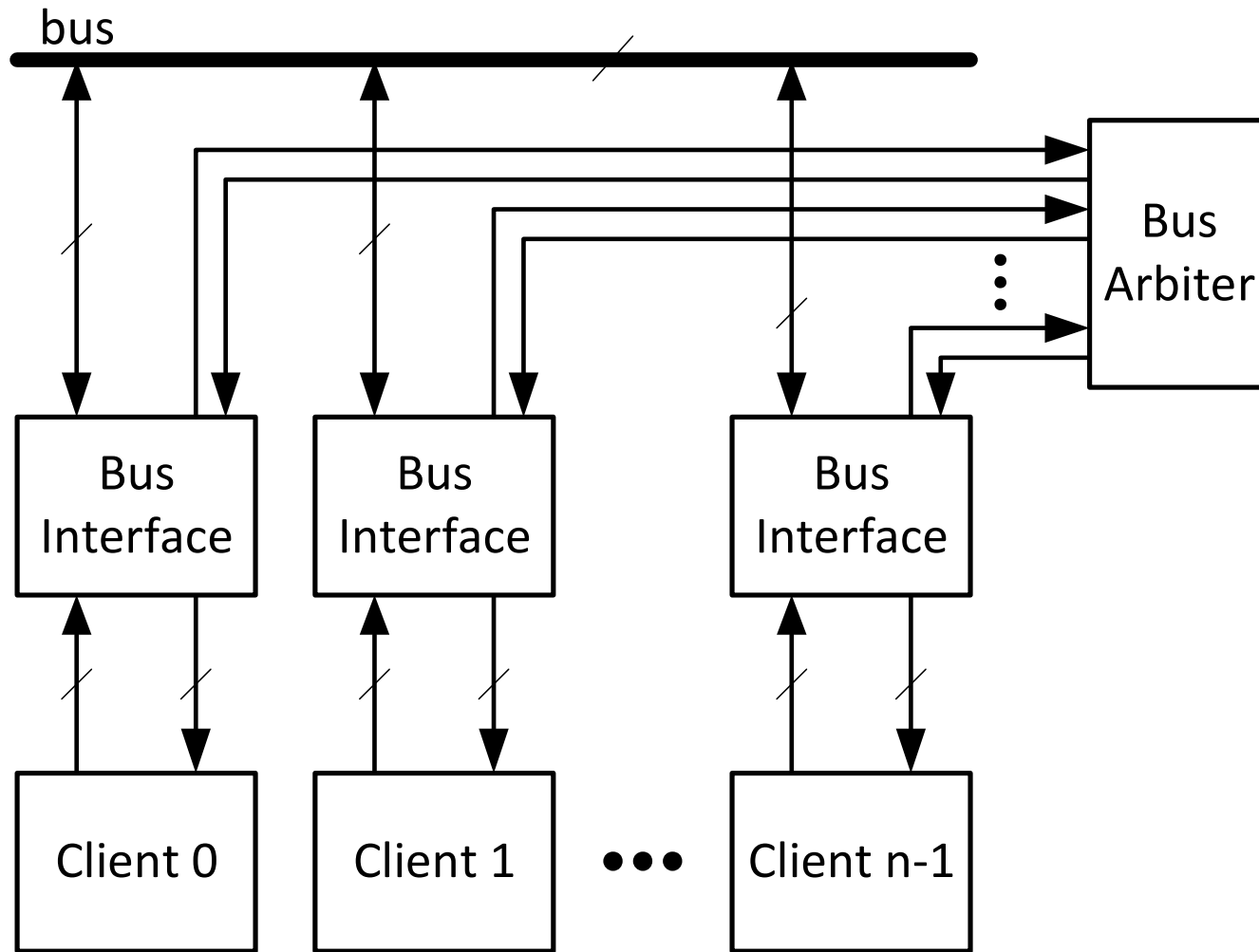


Interconnect



- Many clients need to communicate
- Ad-hoc point-to-point wiring or shared interconnect
- Like a telephone exchange

Bus



Verilog for a simple bus interface

```
// Combinational Bus Interface
// t (transmit) and r (receive) in signal names are from the perspective of the bus
module BusInt(cr_valid, cr_ready, cr_addr, cr_data, // bus rx - to the bus
             ct_valid, ct_data,                  // bus tx - from the bus
             br_addr, br_data, br_valid,         // to the bus
             bt_addr, bt_data, bt_valid,         // from the bus
             arb_req, arb_grant,                 // the arbiter
             my_addr) ;                          // address of this interface

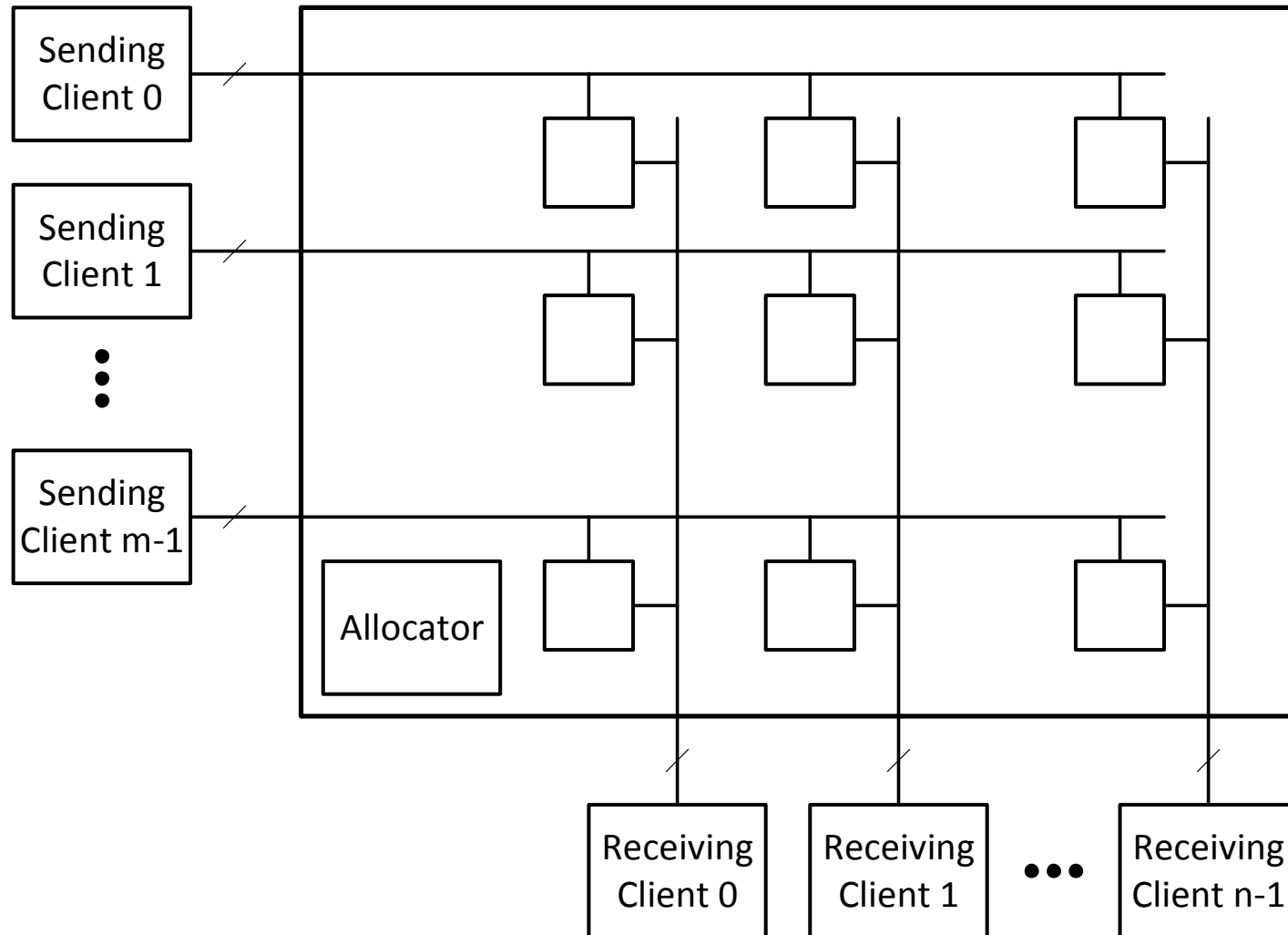
parameter aw = 2 ; // address width
parameter dw = 4 ; // data width
input cr_valid, arb_grant, bt_valid ;
output cr_ready, ct_valid, arb_req, br_valid ;
input [aw-1:0] cr_addr, bt_addr, my_addr ;
output [aw-1:0] br_addr ;
input [dw-1:0] cr_data , bt_data ;
output [dw-1:0] br_data, ct_data ;

// arbitration
wire arb_req = cr_valid ;
wire cr_ready = arb_grant ;

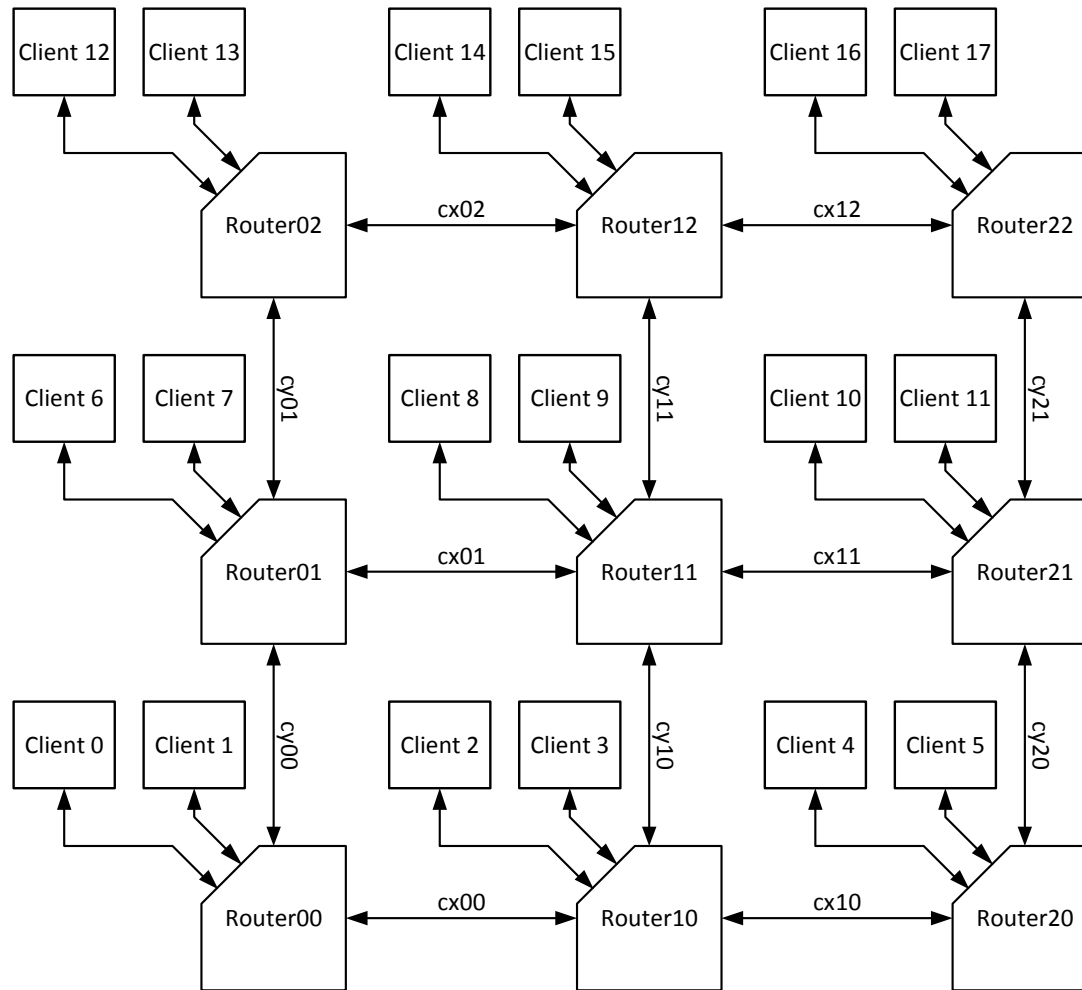
// bus drive
// assumes bus ORs these signals with those from other interfaces
wire br_valid = arb_grant ;
wire [aw-1:0] br_addr = arb_grant ? cr_addr : 0 ;
wire [dw-1:0] br_data = arb_grant ? cr_data : 0 ;

// bus receive
wire ct_valid = bt_valid & (bt_addr == my_addr) ;
wire [dw-1:0] ct_data = bt_data ;
endmodule
```

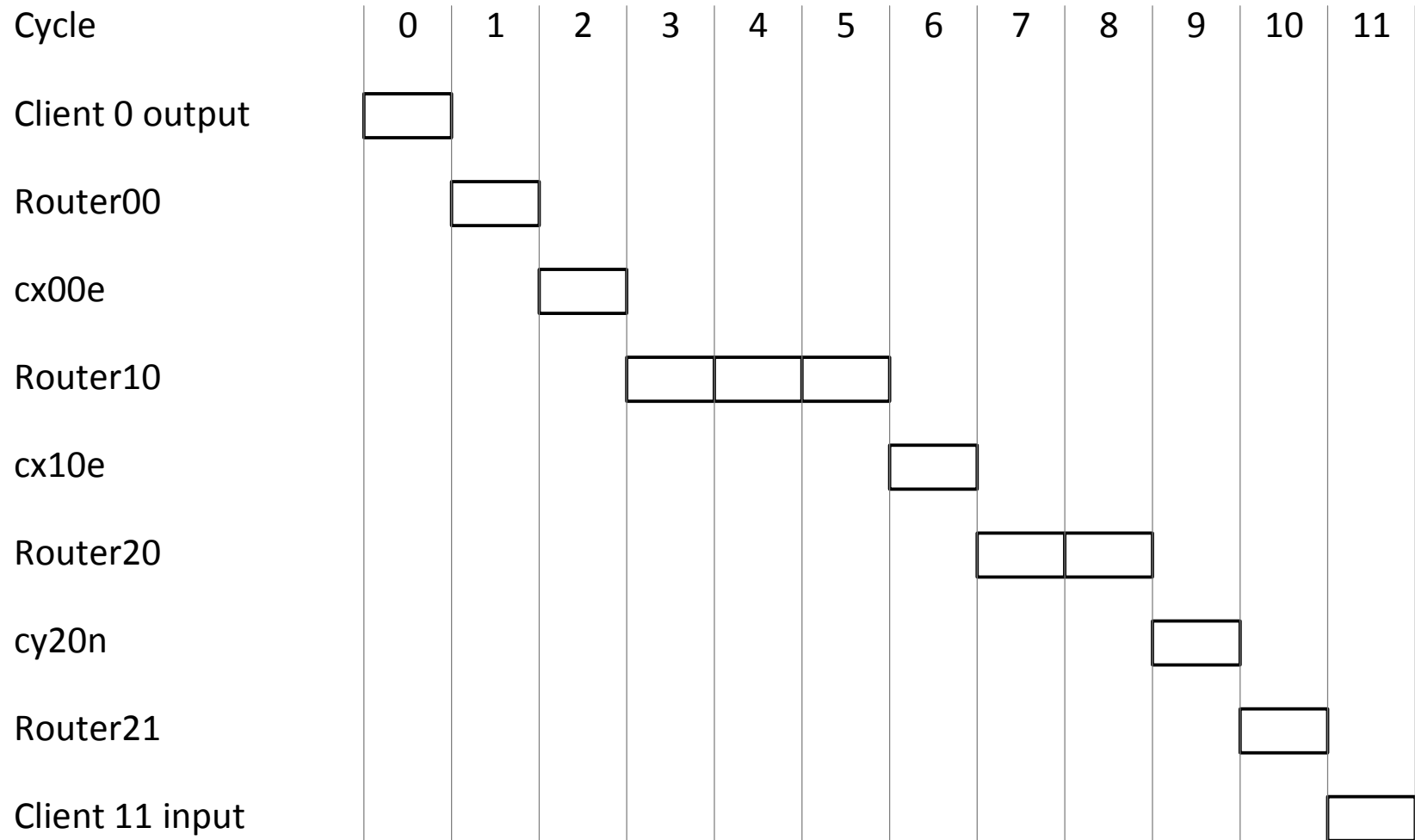
Crossbar Switch



Interconnection Networks

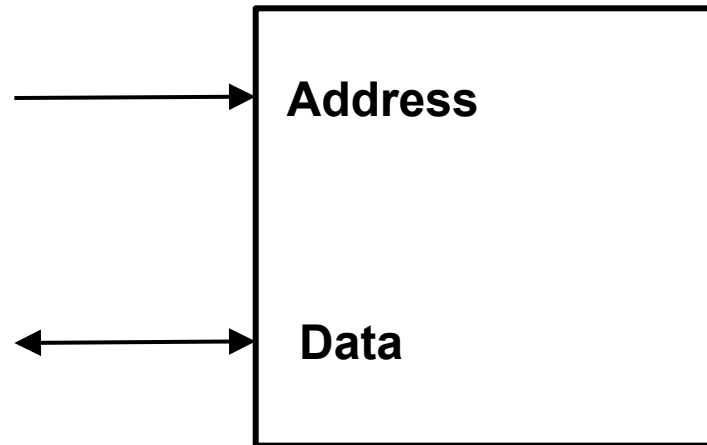


Interconnection Networks



What factors determine which interconnect solution you pick?

Memory



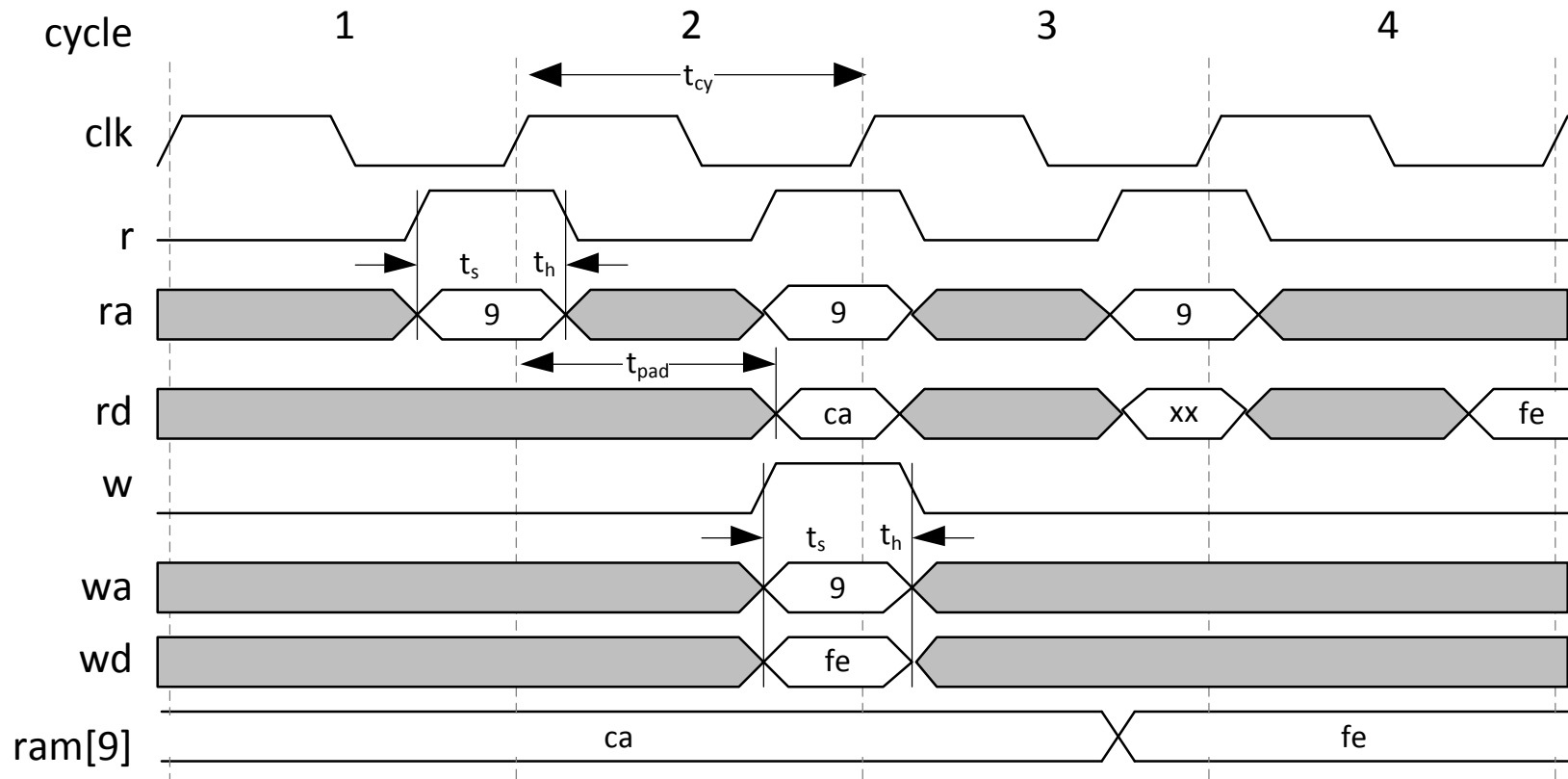
Capacity

Bandwidth

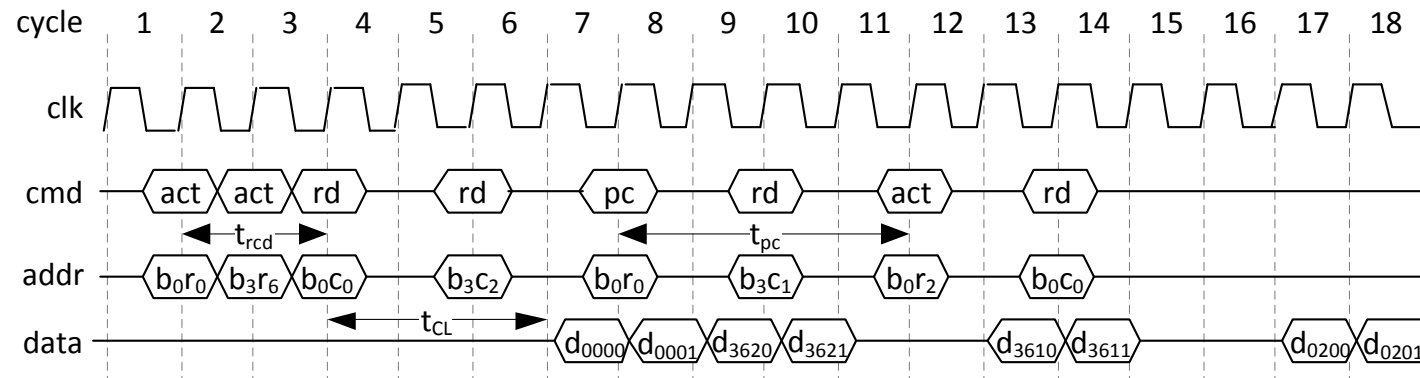
Latency

Granularity

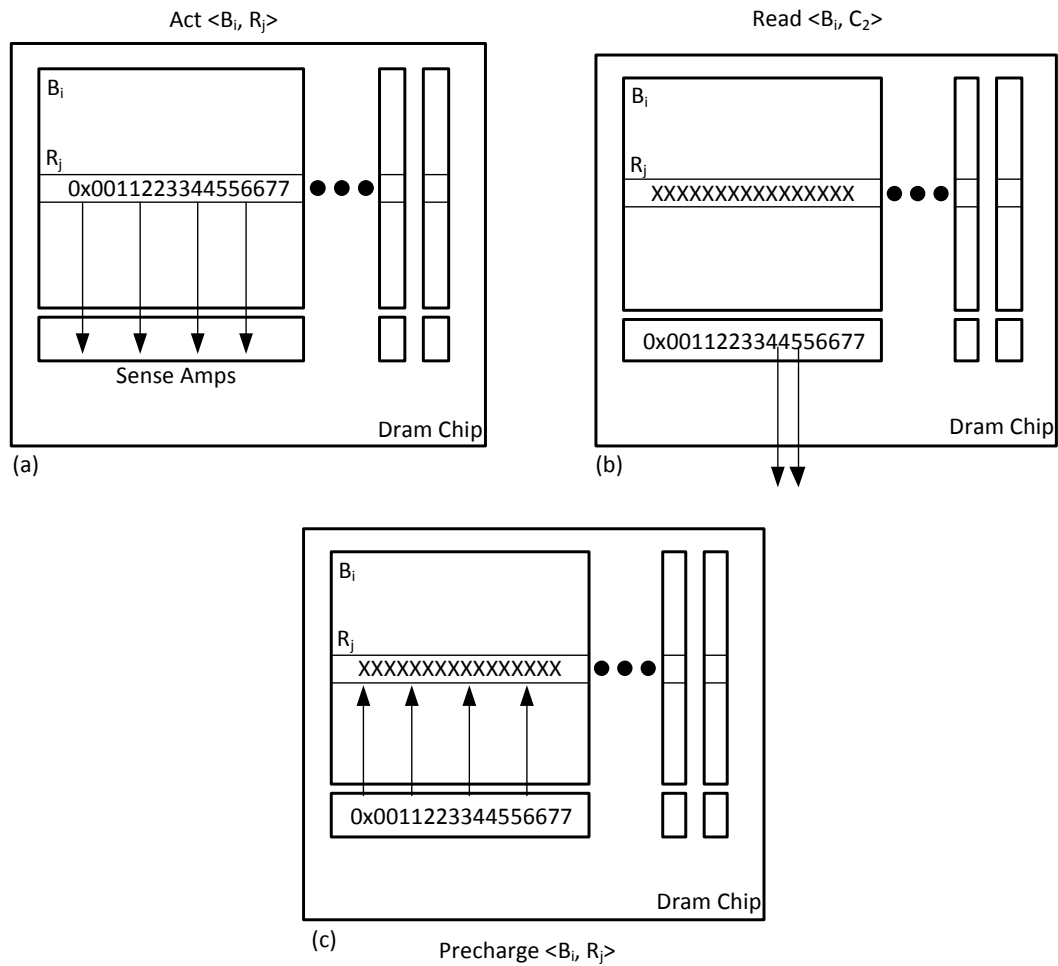
SRAM Primitive



DRAM Primitive

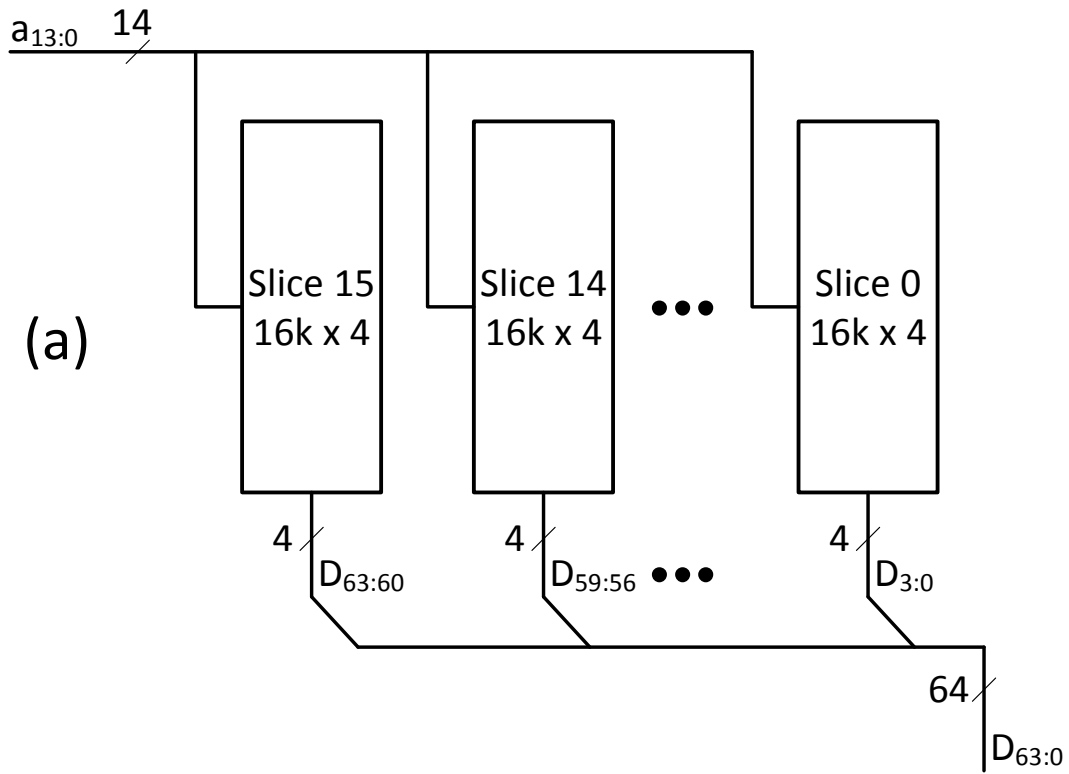


DRAM Operation

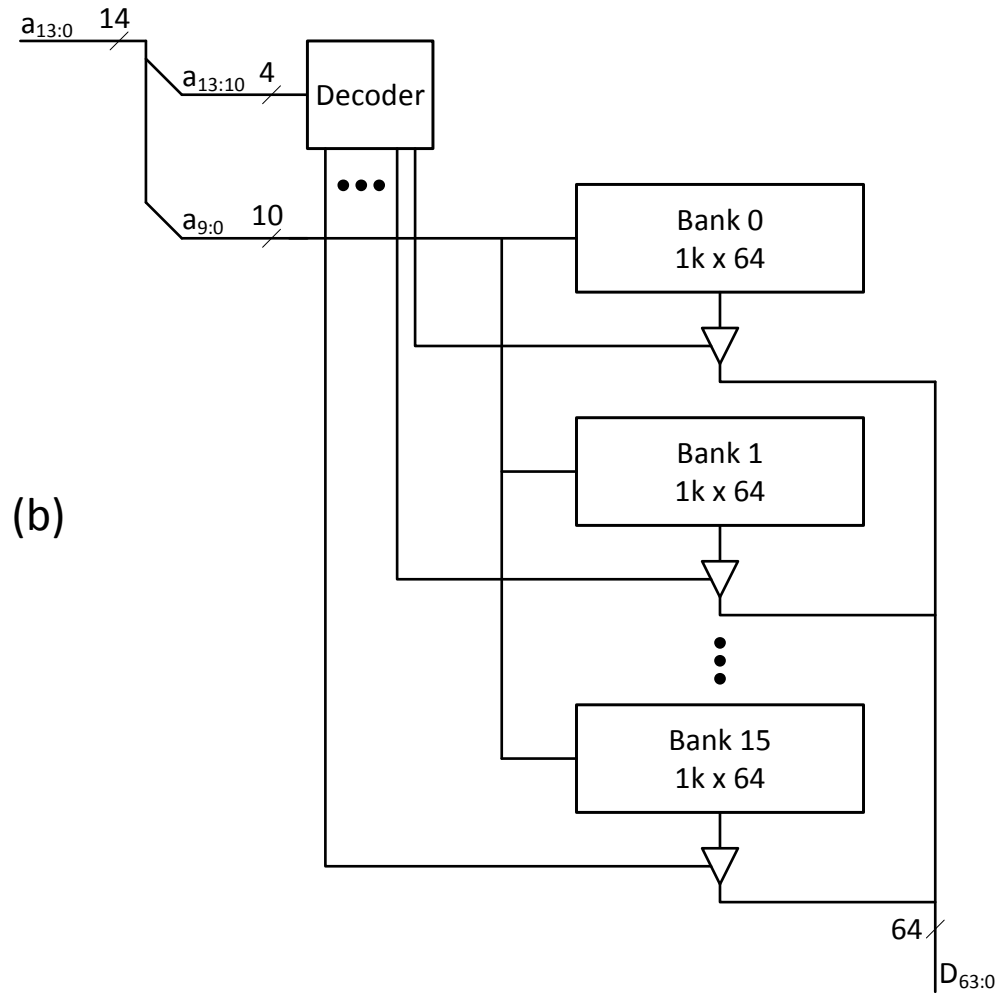


What if you need more memory or more bandwidth than one primitive?

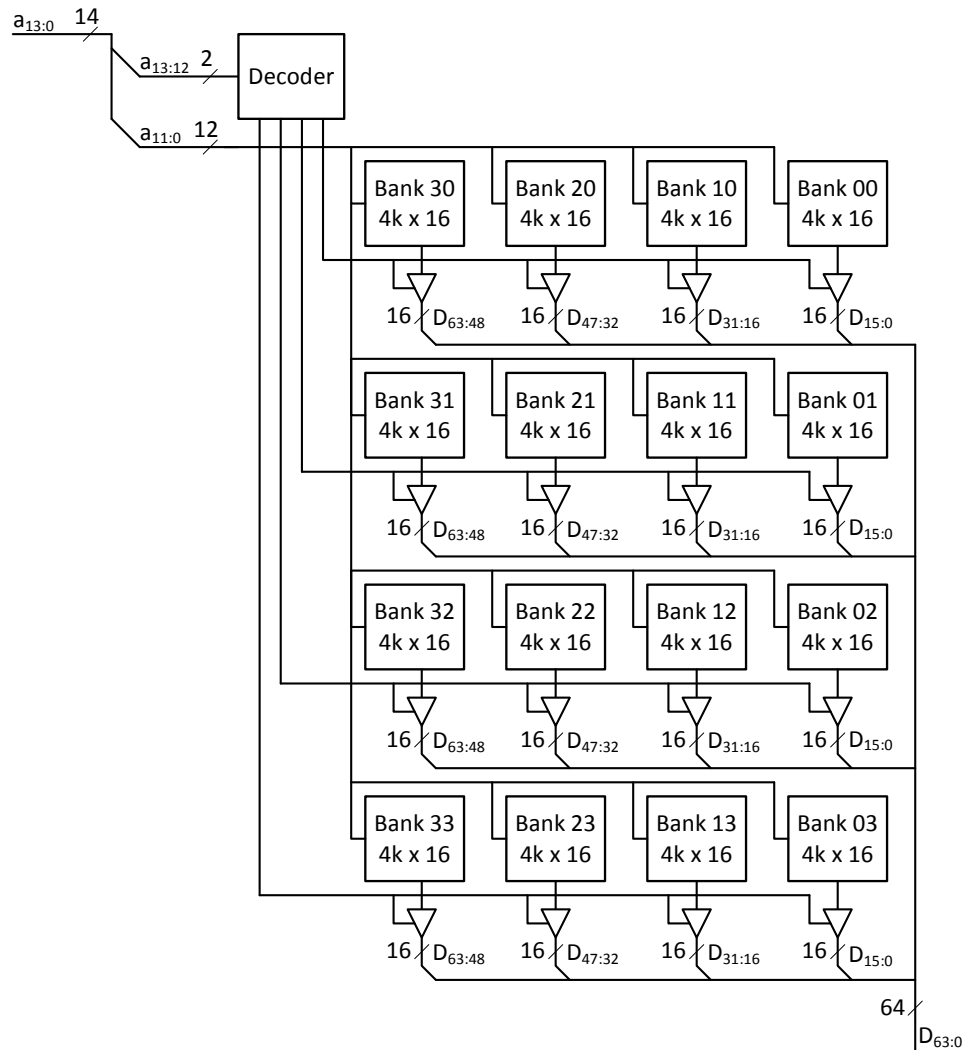
Bit-Slicing



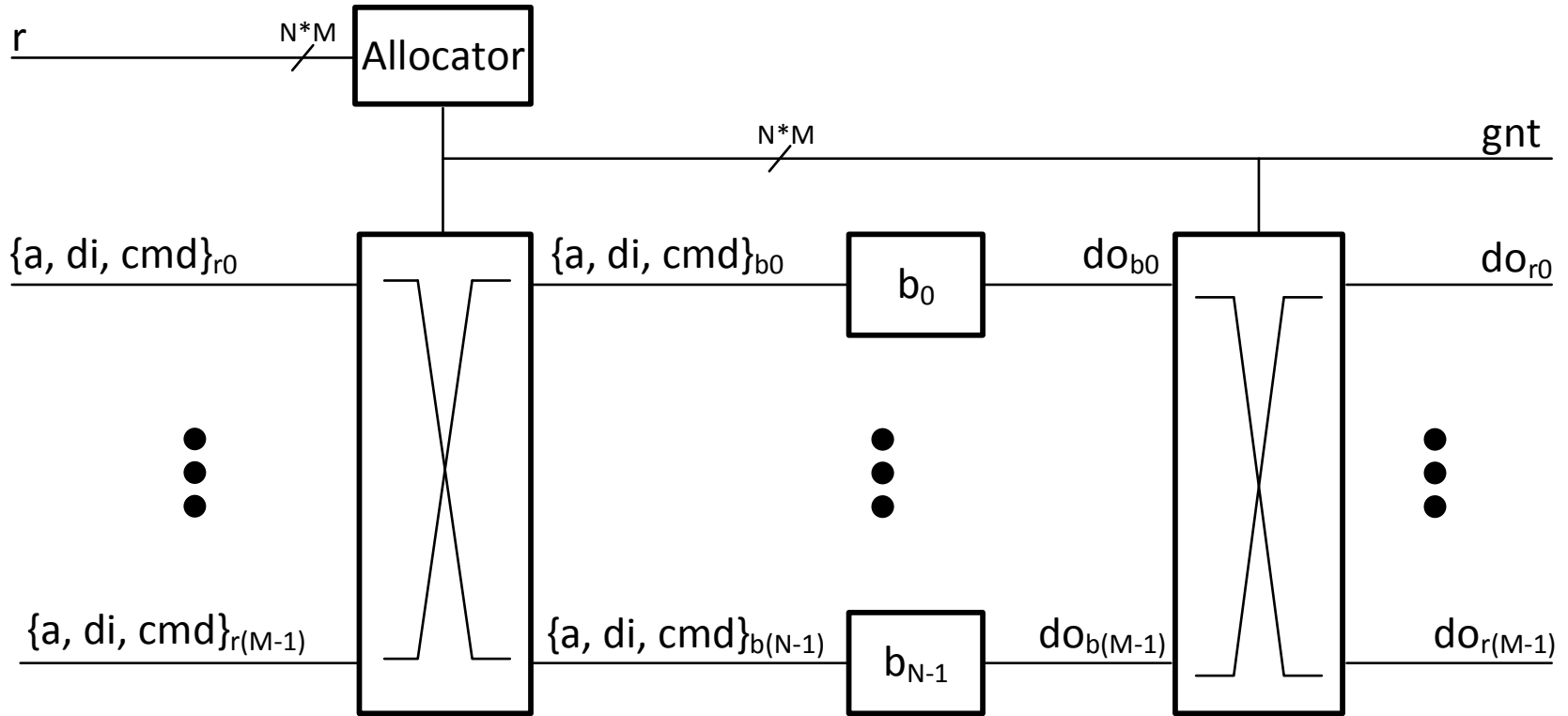
Banking



Bit slicing & banking



Interleaving



Avoid head-of-line blocking

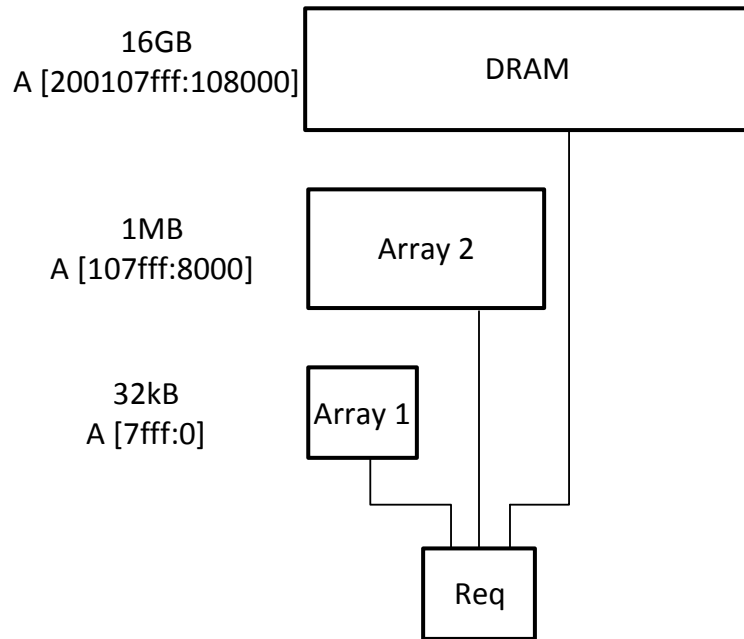
Time	Q0	Q1	Q2	Q3	G0	G1	G2	G3
1	0,1,2,3	0,1,2,3	0,1,2,3	0,1,2,3	Q0	–	–	–
2	1,2,3	0,1,2,3	0,1,2,3	0,1,2,3	Q1	Q0	–	–
3	2,3	1,2,3	0,1,2,3	0,1,2,3	Q2	Q1	Q0	–
4	3	2,3	1,2,3	0,1,2,3	Q3	Q2	Q1	Q0
5	–	3	2,3	1,2,3	–	Q3	Q2	Q1
6	–	–	3	2,3	–	–	Q3	Q2
7	–	–	–	3	–	–	–	Q3

Table 24.2: With head of line blocking, the memory system does not have full throughput. Requests that can be granted are stuck behind requests waiting for a over-subscribed resource.

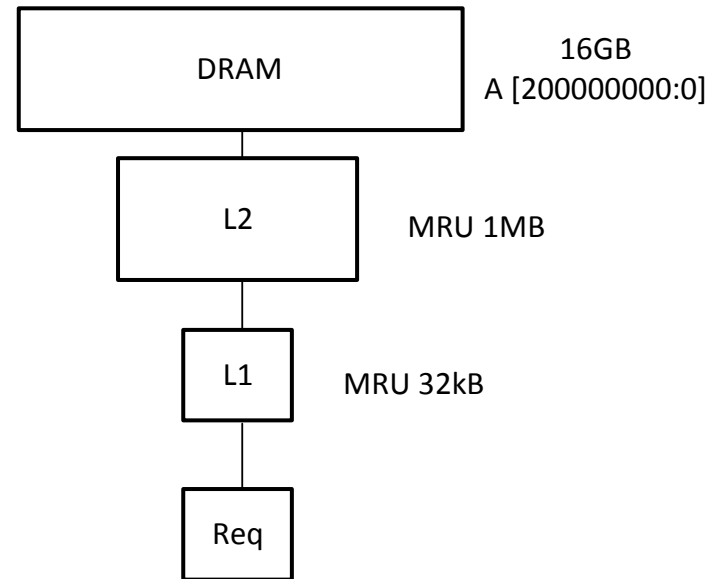
Time	Q0	Q1	Q2	Q3	G0	G1	G2	G3
1	0,1,2,3	0,1,2,3	0,1,2,3	0,1,2,3	Q0	Q1	Q2	Q3
2	1,2,3	0,2,3	0,1,3	0,1,2	Q3	Q0	Q1	Q2
3	2,3	0,3	0,1	1,2	Q2	Q3	Q0	Q1
4	3	0	1	2	Q1	Q2	Q3	Q0

Table 24.3: When the arbiter is able to see requests beyond the head of the full, the memory system has no head of line blocking. It achieves full throughput.

Hierarchy



(a)



(b)

Summary

- Interface timing
 - Convention about when data is transferred
 - When valid, when accepted
 - Open loop – always valid or periodic
 - Ready-valid flow-control, both ways, push, or pull
 - Serialization
- Interconnect
 - Allow any pair of clients to communicate
 - Bus, Crossbar, Network
- Memory
 - Capacity, bandwidth, latency, and granularity
 - SRAM and DRAM primitives
 - Bit-slice bank, or interleave to combine primitives
 - Hierarchy