EECS151/251A

Spring 2018

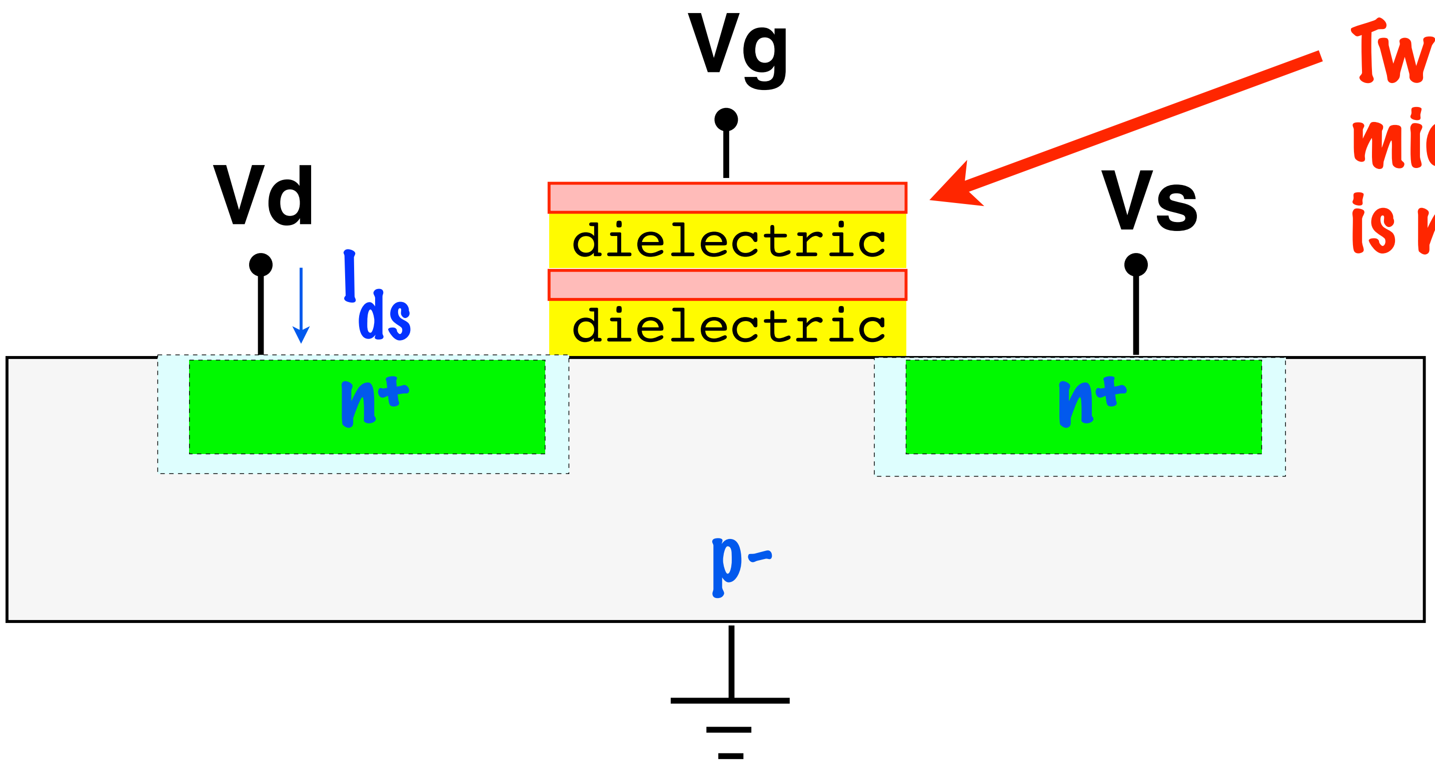
Digital Design and Integrated Circuits

Instructors:

John Wawrzynek and Nick Weaver

Lecture 18: State and How To Use It

The physics of FLASH memory

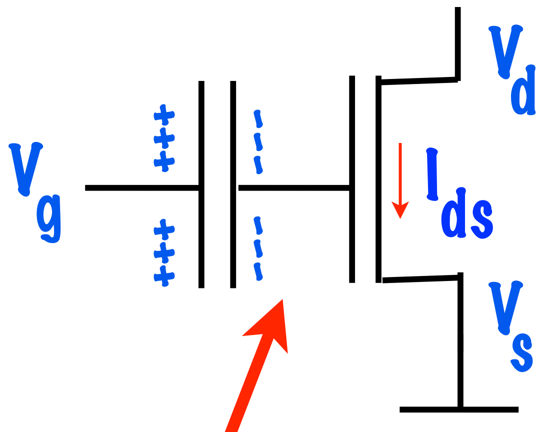


Two gates. But the middle one is not connected.

1. Electrons "placed" on floating gate stay there for many years (ideally).

2. 10,000 electrons on floating gate shift transistor threshold by 2V.

3. In a memory array, shifted transistors hold "0", unshifted hold "1".

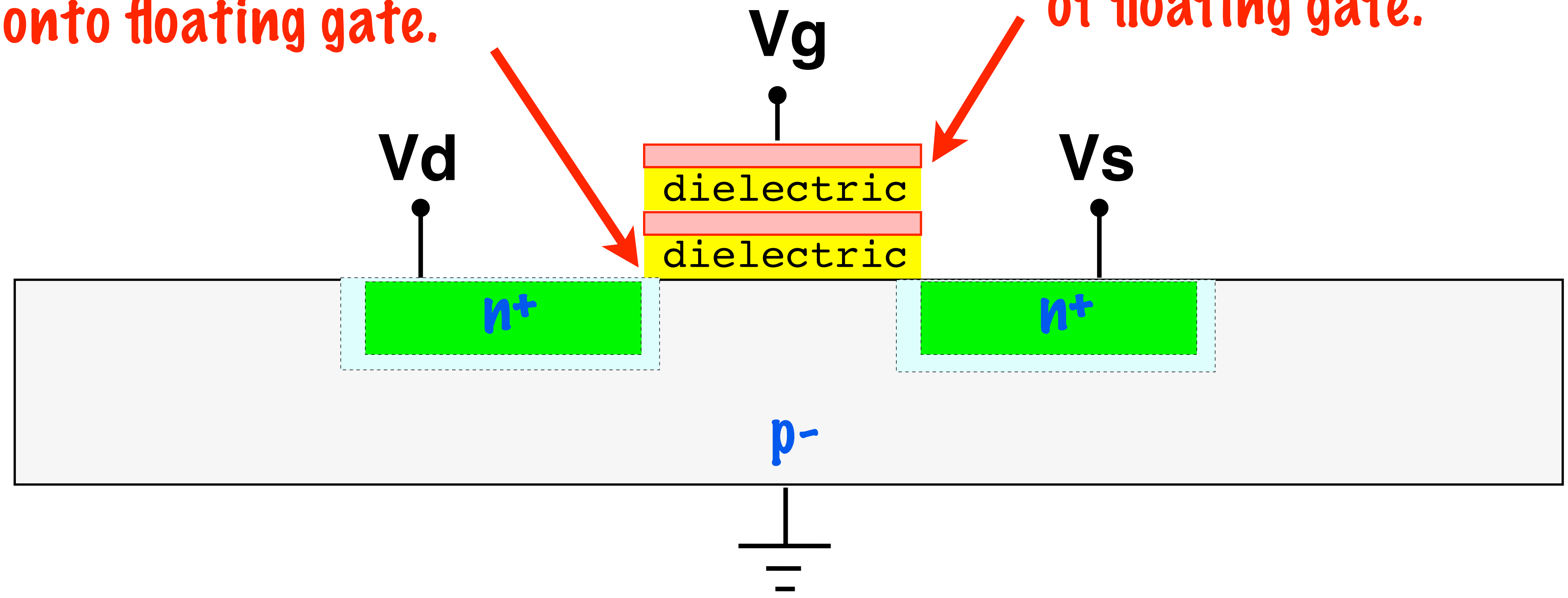


"Floating gate".

Moving electrons on/off floating gate

A high drain voltage injects "hot electrons" onto floating gate.

A high gate voltage "tunnels" electrons off of floating gate.



1. Hot electron injection and tunneling produce tiny currents, thus writes are slow.

2. High voltages damage the floating gate. Too many writes and a bit goes "bad".

Architecture ...

NAND Flash Memory

Flash: Disk Replacement

Chip "remembers" for 10 years.

Presents memory to the CPU as a set of pages.

Page format:

2048 Bytes	+	64 Bytes
(user data)		(meta data)

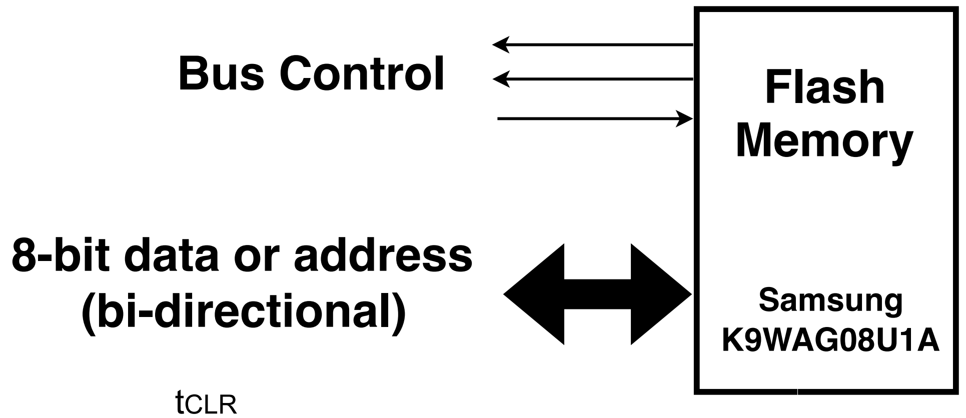
1GB Flash: 512K pages

2GB Flash: 1M pages

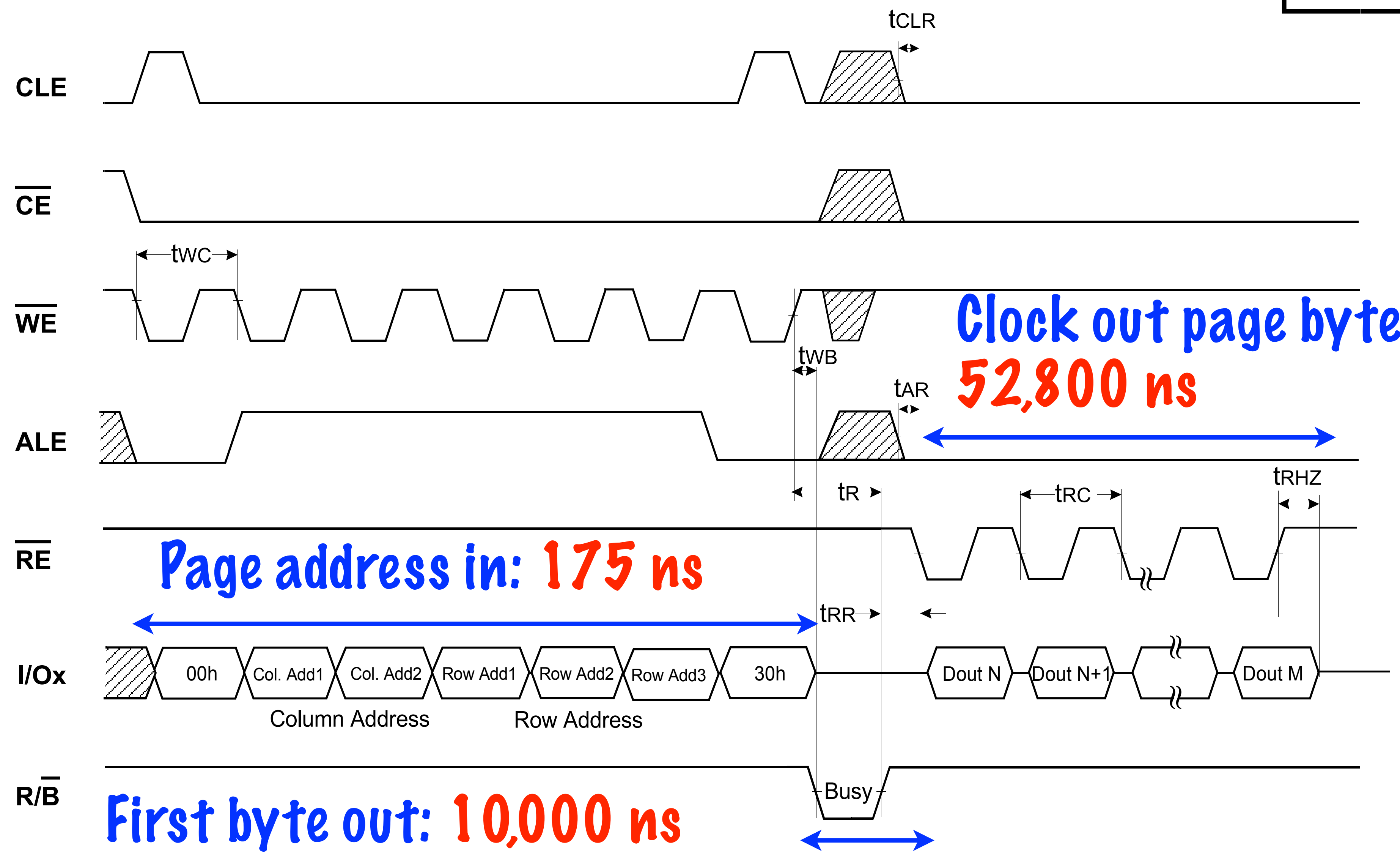
4GB Flash: 2M pages

Reading a Page ...

33 MB/s Read Bandwidth



Read Operation



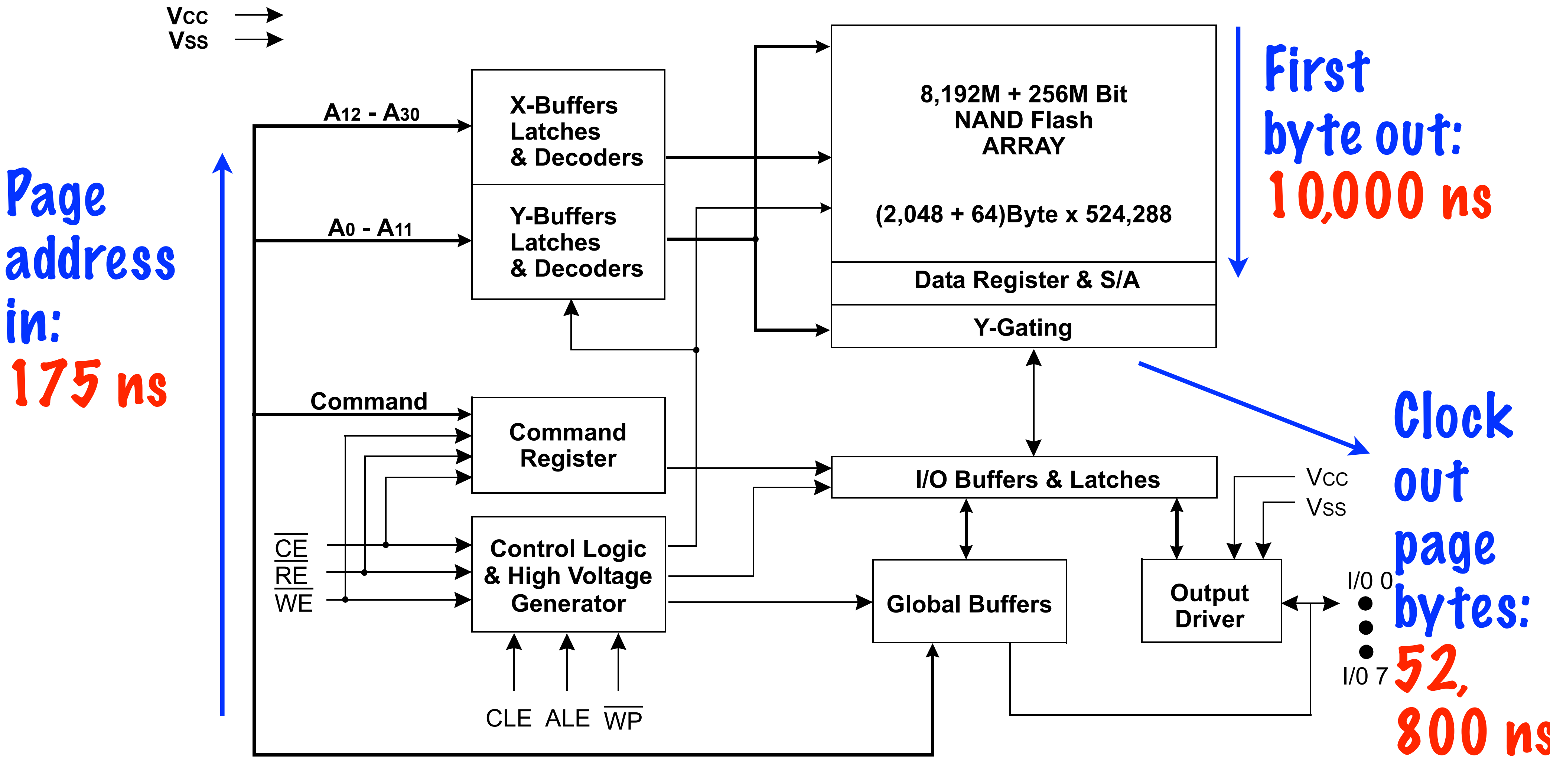
Clock out page bytes: 52,800 ns

Page address in: 175 ns

First byte out: 10,000 ns

Where Time Goes

Figure 1. K9K8G08U0A Functional Block Diagram



Vcc
Vss
A12 - A30
A0 - A11
Command
X-Buffers Latches & Decoders
Y-Buffers Latches & Decoders
8,192M + 256M Bit NAND Flash ARRAY
(2,048 + 64)Byte x 524,288
Data Register & S/A
Y-Gating
Command Register
I/O Buffers & Latches
Control Logic & High Voltage Generator
Global Buffers
Output Driver
CE
RE
WE
CLE ALE WP
Vcc
Vss
I/0 0
I/0 7
Page address in: 175 ns
First byte out: 10,000 ns
Clock out page bytes: 52,800 ns

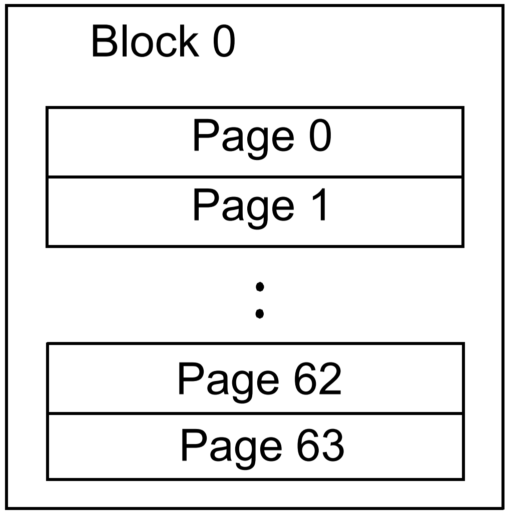
Writing a Page ...

A page lives in a block of 64 pages:

1GB Flash: 8K blocks

2GB Flash: 16K blocks

4GB Flash: 32K blocks



To write a page:

1. Erase all pages in the block (cannot erase just one page).

Time: 1,500,000 ns

2. May program each page individually, exactly once.

Time: 200,000 ns per page.

Block lifetime: 100,000 erase/program cycles.

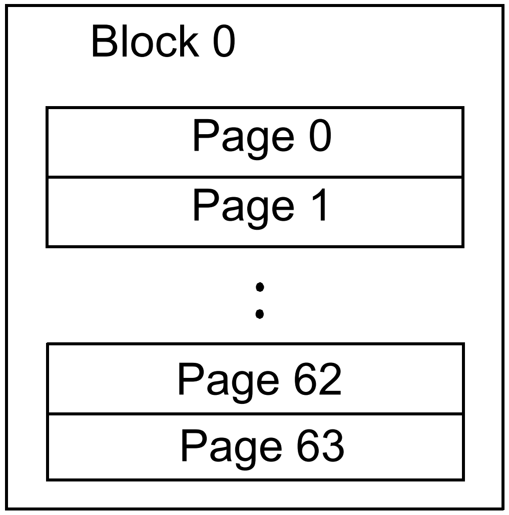
Block Failure

Even when new, not all blocks work!

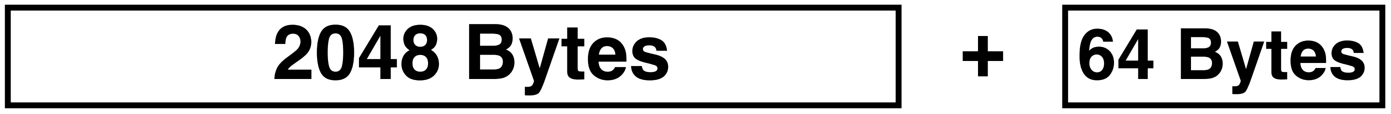
1GB: 8K blocks, 160 may be bad.

2GB: 16K blocks, 220 may be bad.

4GB: 32K blocks, 640 may be bad.



During factory testing, Samsung writes good/bad info for each block in the meta data bytes.

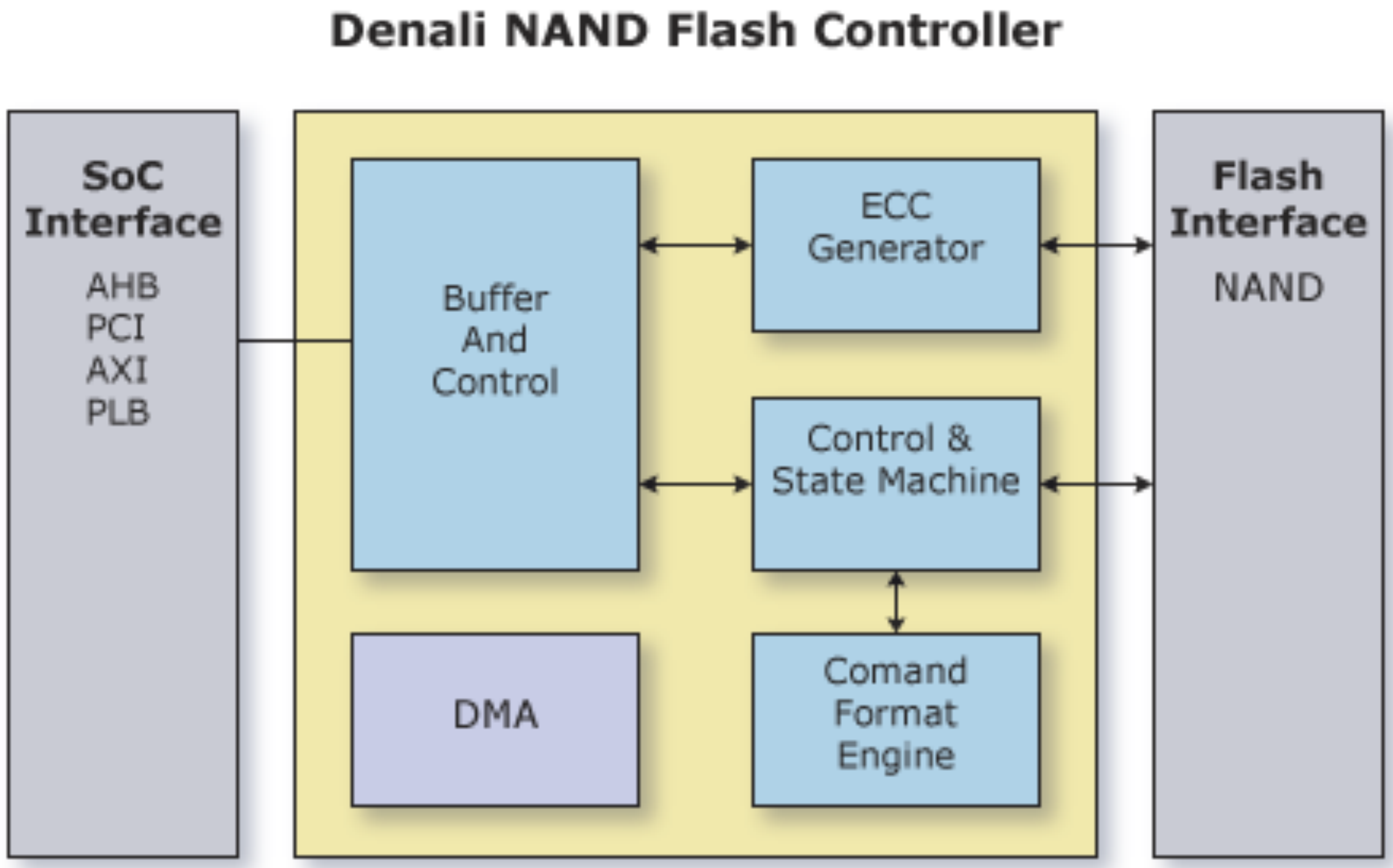


(user data) **(meta data)**

After an erase/program, chip can say "write failed", and block is now "bad". OS must recover (migrate bad block data to a new block). Bits can also go bad "silently" (!!!).

Flash controllers: Chips or Verilog IP ...

Flash memory controller manages write lifetime management, block failures, silent bit errors ...



Software sees a "perfect" disk-like storage device.

So Why Use Flash?

- It is ***persistent***: Data exists when power is off
  - We need memory to bootstrap our systems
- It is ***dense***:
  - Even more dense than DRAM
- It is ***low latency...*** Compared to spinning disk!
  - Can hide the much higher write latency with a combination of buffering and ensuring that there are always blank pages available

Memory Generation Options

Some Types of Memory...

- Single port:
  - One address port, one data in port, one data out port
  - Can read or write
- Simple dual port:
  - Two address ports, one for reading, one for writing
    - Very good for implementing FIFOs!
- True dual port:
  - Two address ports, both can be used for reading or writing
- Suggestion for Xilinx version of the project:
  - Best way to do the processor reg-file is instantiate simple-dual-port memories

Using State: FIFO

- FIFO (First-In-First-Out) is incredibly common
  - Either a fixed delay or a variable delay
- Complication: Most FIFOs need to both read and write potentially every clock cycle
  - Otherwise the FIFO can become the bottleneck
  - Such FIFOs ***require*** either a separate read port or need to be clocked at 2x the external interface clock

One Way of Specifying Memory: Library Blocks

- Just directly instantiate memories from the right size
  - Every synthesis flow will have this
- EG, on the Xilinx we can directly instantiate both BlockRAMs and SLICEM Rams
  - Vivado has language templates for the common ones
  - With core generators also able to create IP variants

Other Way: Verilog RAM Specification

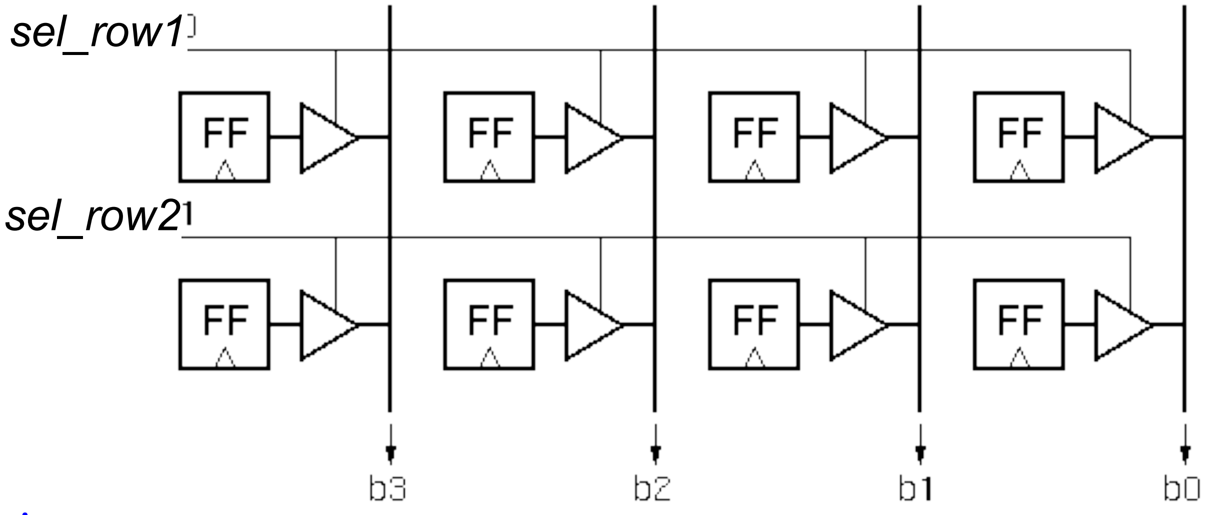
```
//
// Single-Port RAM with Asynchronous Read
//
module ramBlock (clk, we, a, di, do);
    input  clk;
    input  we;            // write enable
    input  [19:0] a;       // address
    input  [7:0] di;     // data in
    output [7:0] do;     // data out
    reg    [7:0] ram [1048575:0];  // 8x1Meg
    always @(posedge clk) begin    // Synch write
        if (we)
            ram[a] <= di;
    assign do = ram[a];            // Asynch read
endmodule
```

What do the synthesis tools do with this?

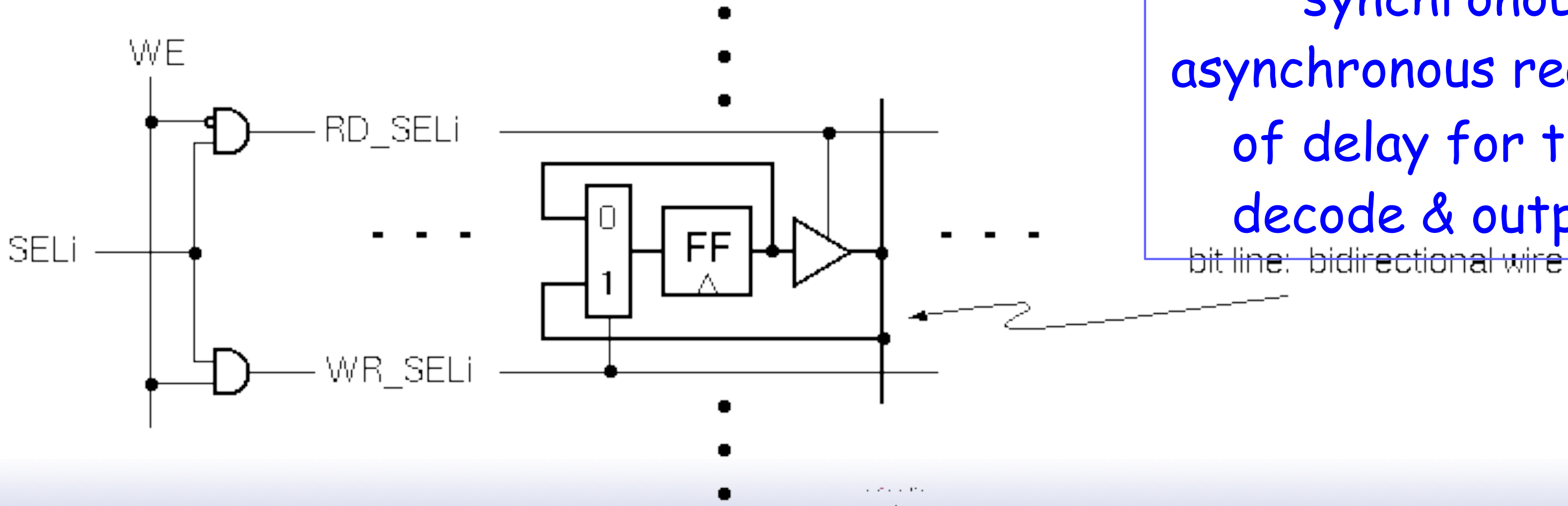
Flip-flop based memory block

For read operation, a 2-D array off flip-flops with tristate outputs on each.

A decoder supplies row select signals (one hot).



For write operation, includes a means to change state value:



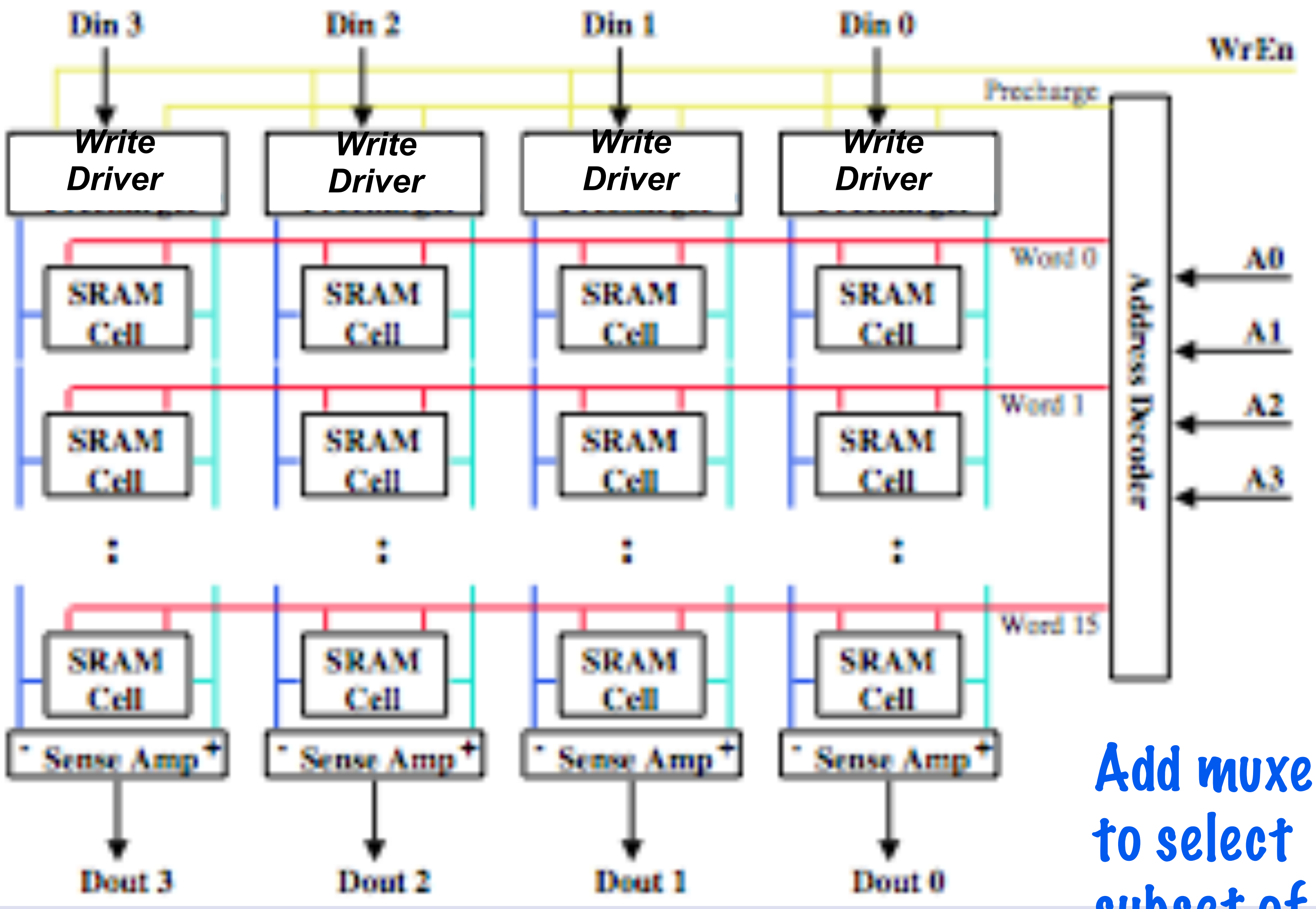
- Timing just like a register: synchronous write, asynchronous read (with a bit of delay for the address decode & output muxing)

☺ *Asynchronous Read,* ☹ *Not dense!*

Dense RAM Array

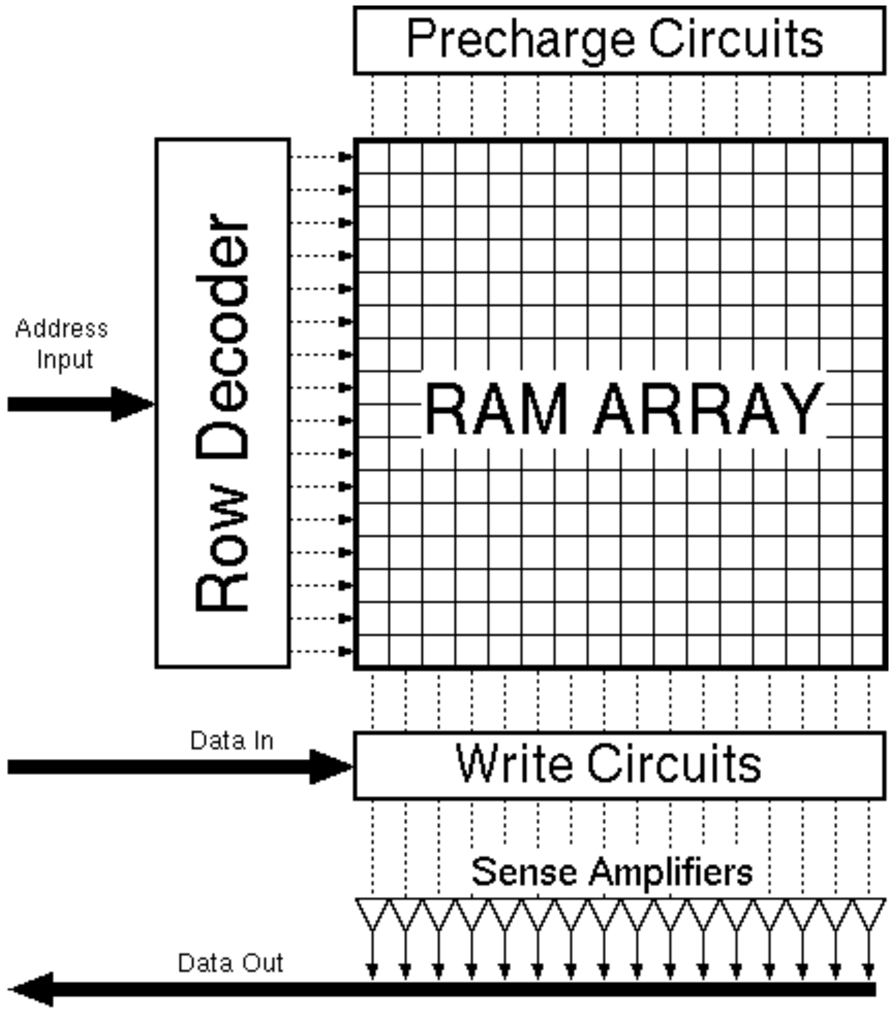
Dedicated memory circuits provide more density (bits/unit-area)

Parallel Data I/O Lines



Add muxes to select subset of bits

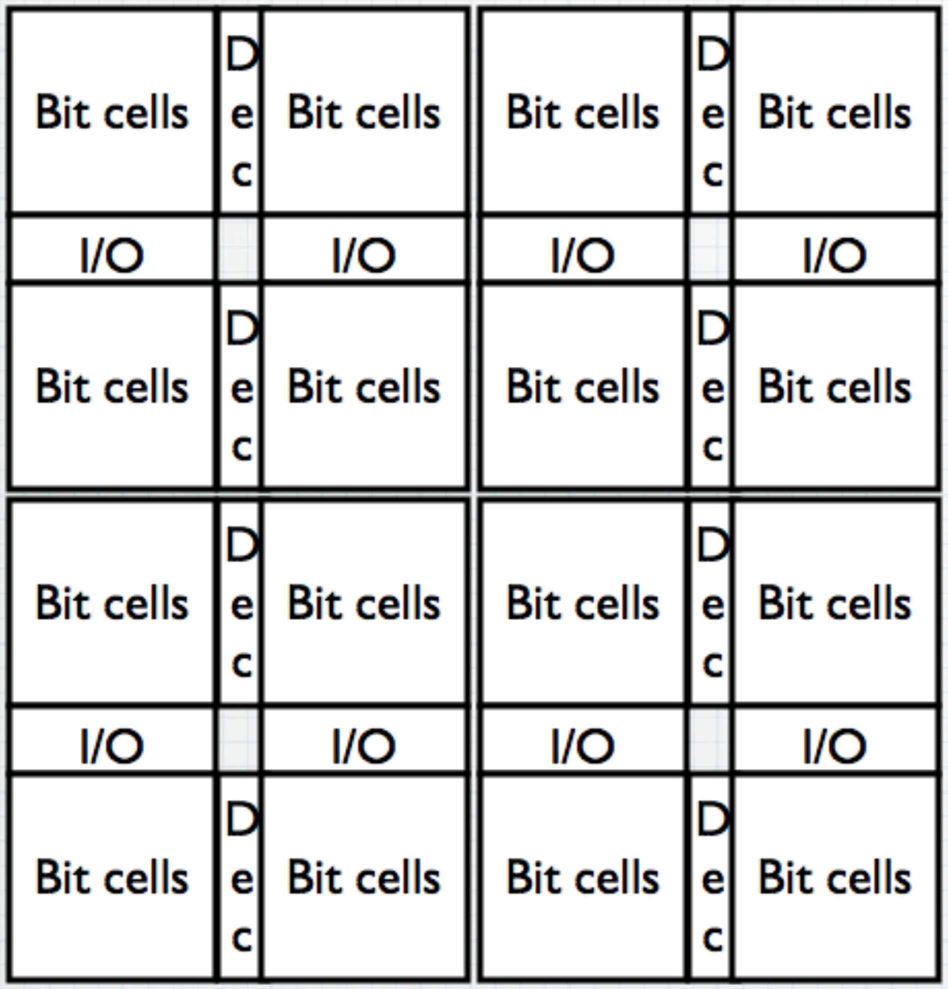
SRAM Block



- Extra circuitry and timed control signals needed
  - Periphery circuits add a “fixed” area overhead
  - Row select, sensing, precharge must be sequenced, based on input clock signal
  - Read operation needs a clock: “synchronous read”

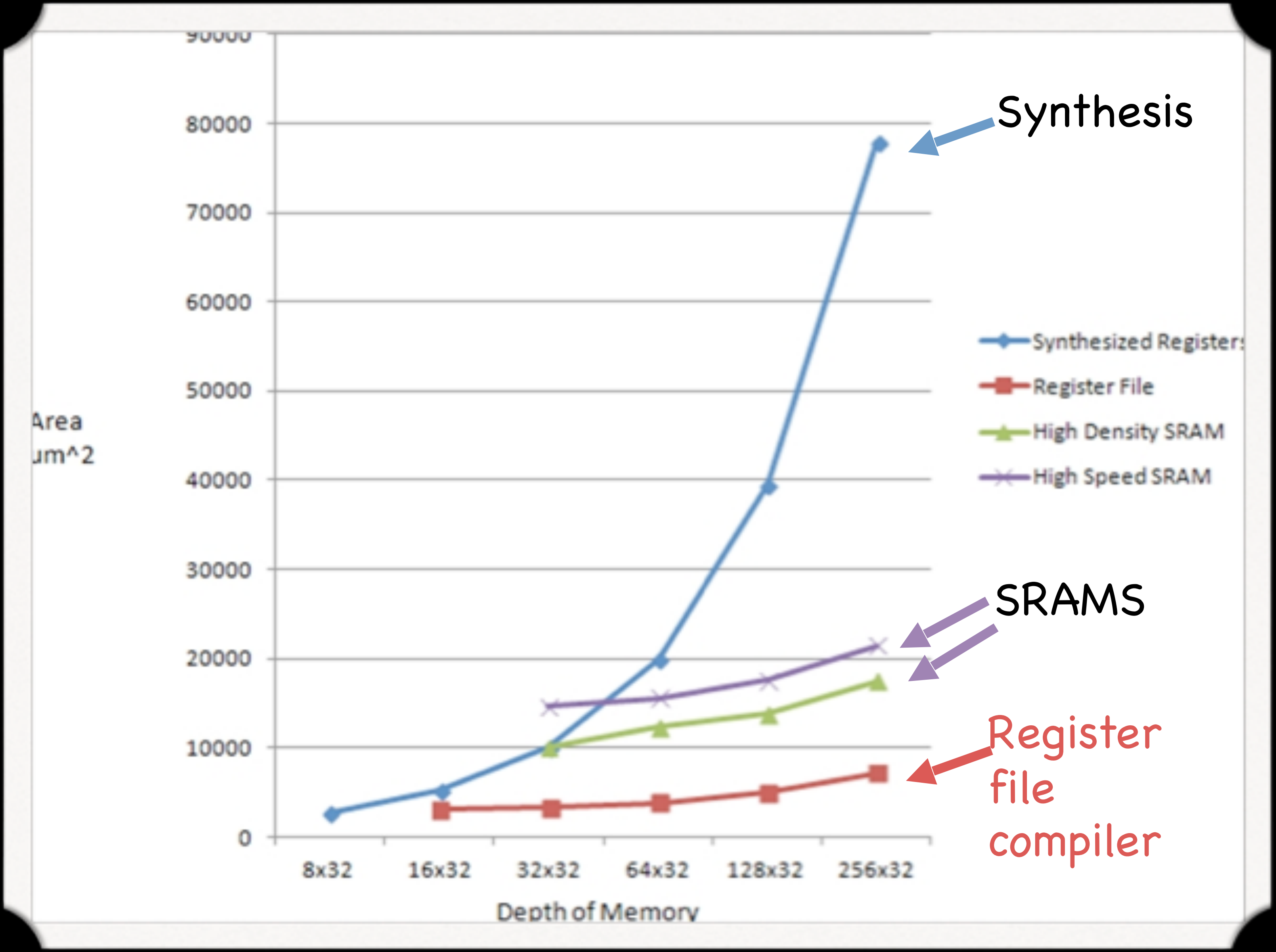
☺ *Dense, lower power, faster* ☹ *Synchronous read (who cares)*

Building Larger Memories



- Large arrays constructed by tiling multiple leaf arrays, sharing decoders and I/O circuitry
  - e.g., sense amp attached to arrays above and below
- Leaf array limited in size to 128-256 bits in row/column due to RC delay of wordlines and bitlines
- Also to reduce power by only activating selected sub-bank
- In larger memories, delay and energy dominated by I/O wiring

For small register files, logic synthesis is competitive.

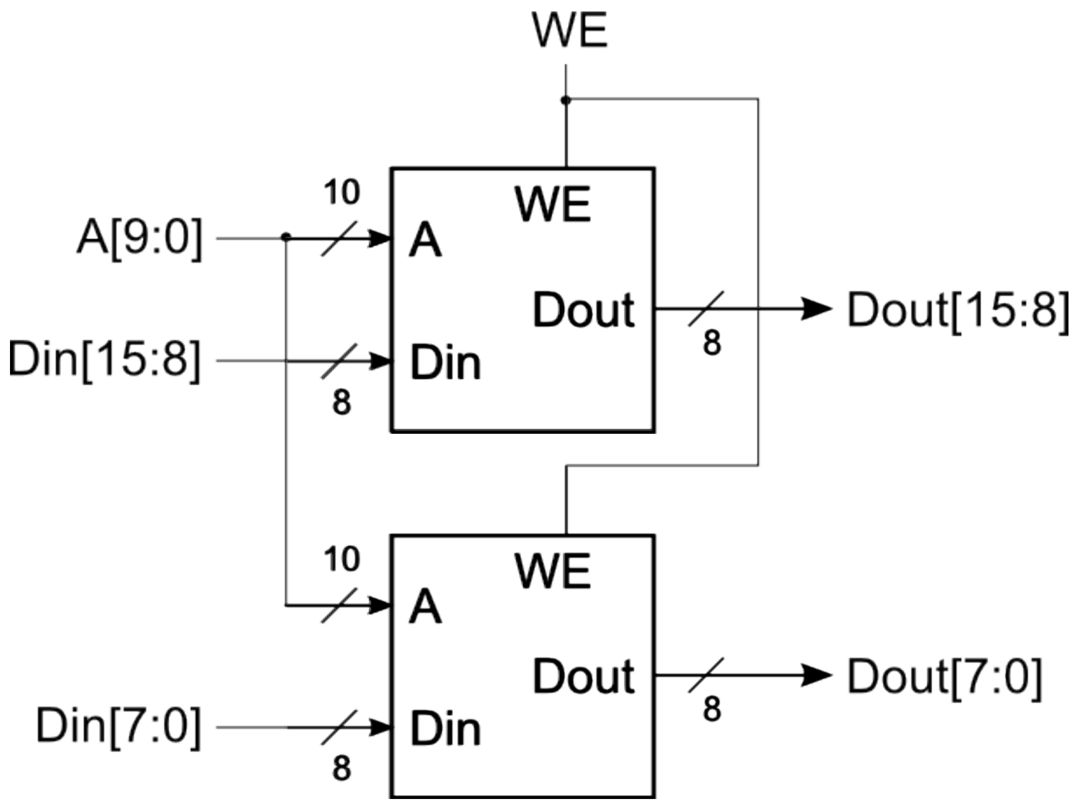


Cascading Memory Blocks

Cascading Memory-Blocks

How to make larger memory blocks out of smaller ones.

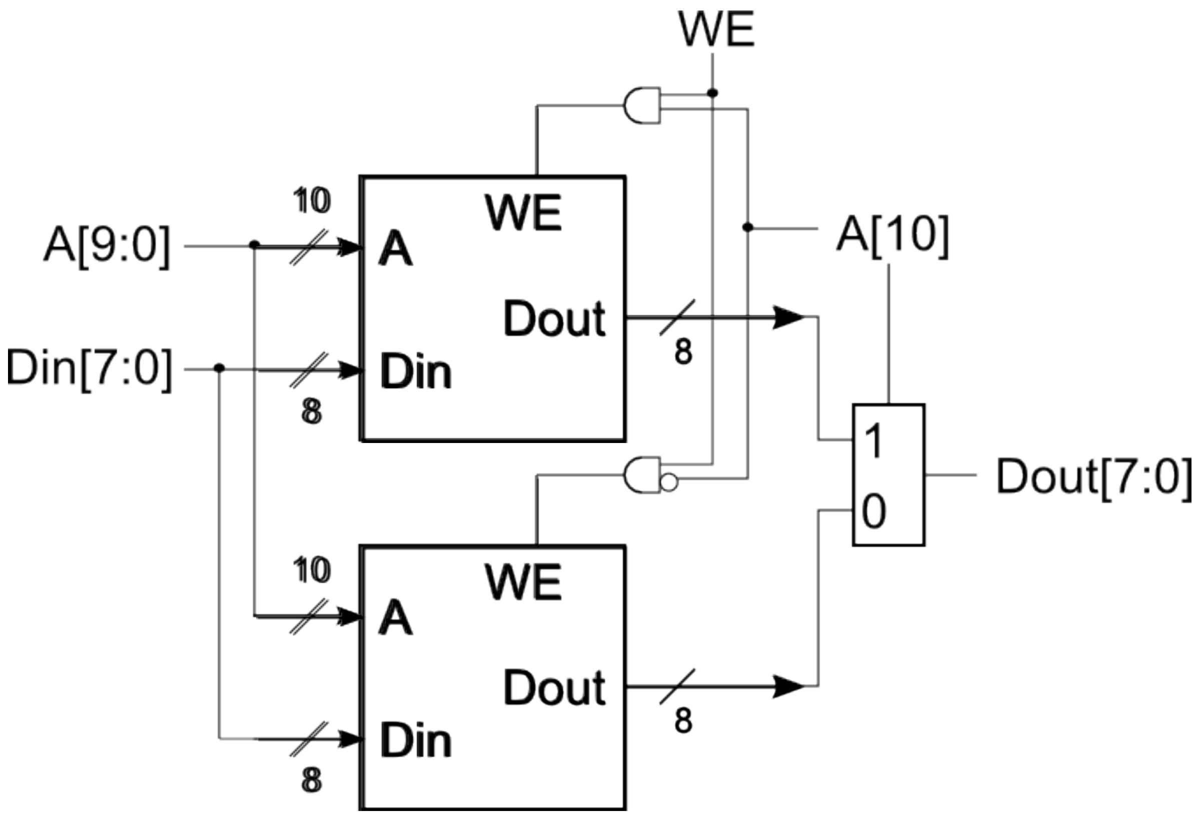
Increasing the width. Example: given 1Kx8, want 1Kx16



Cascading Memory-Blocks

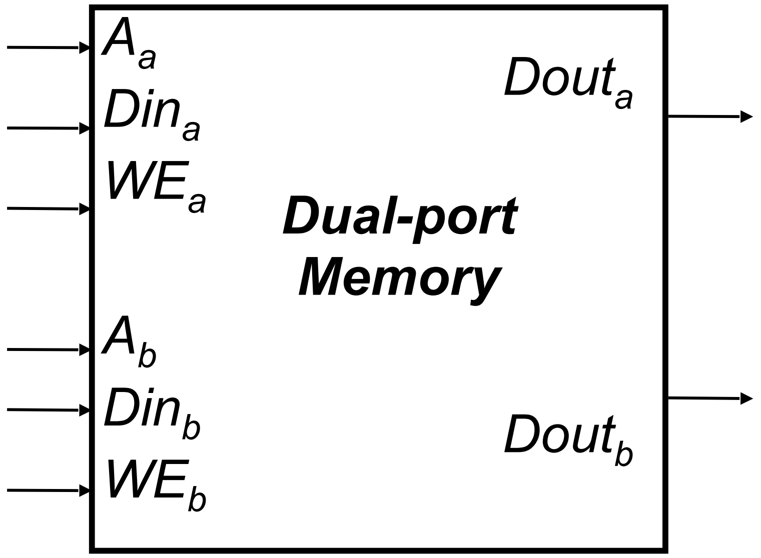
How to make larger memory blocks out of smaller ones.

Increasing the depth. Example: given 1Kx8, want 2Kx8

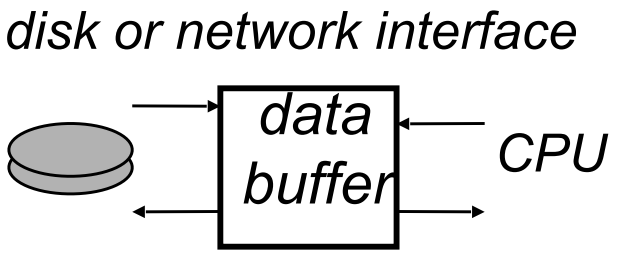


Multi-ported Memory

- Motivation:
  - Consider CPU core register file:
    - 1 read or write per cycle limits processor performance.
    - Complicates pipelining. Difficult for different instructions to simultaneously read or write regfile.
    - Common arrangement in pipelined CPUs is 2 read ports and 1 write port.



– I/O data buffering:

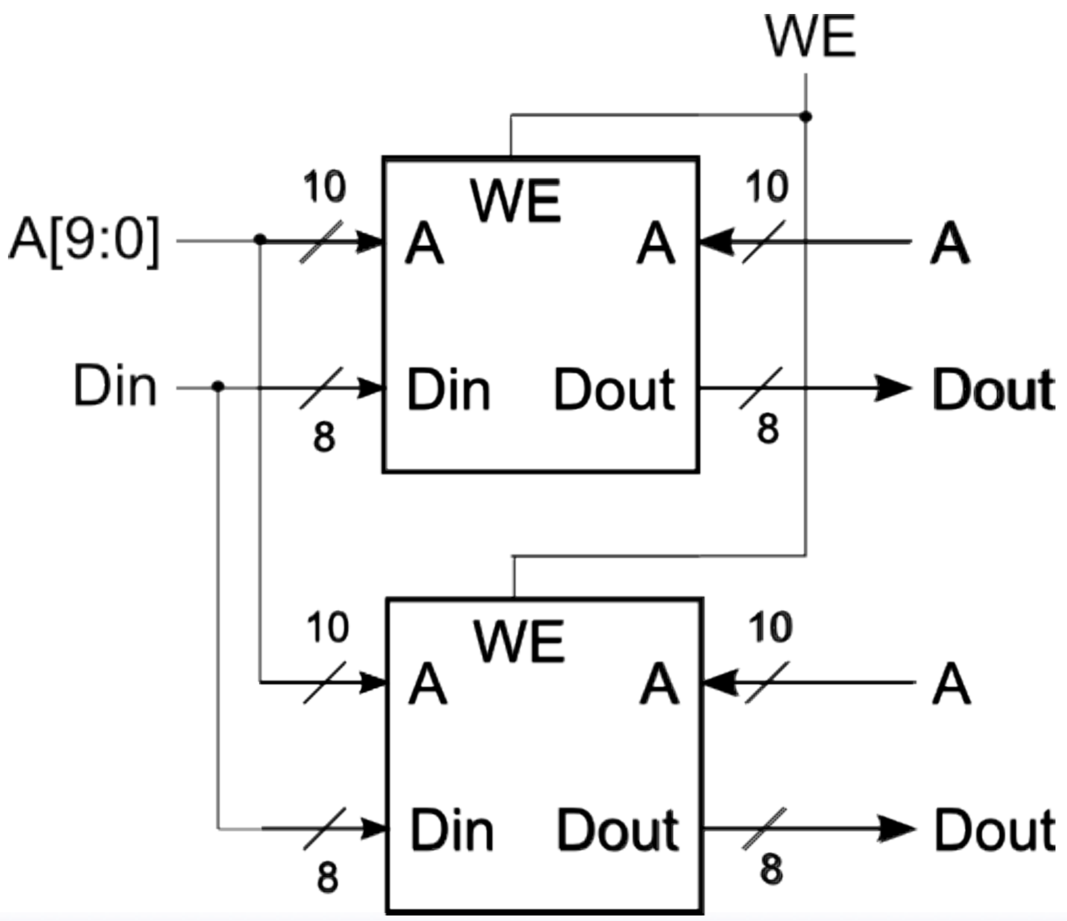


- dual-porting allows both sides to simultaneously access memory at full bandwidth.

Adding Ports to Primitive Memory Blocks

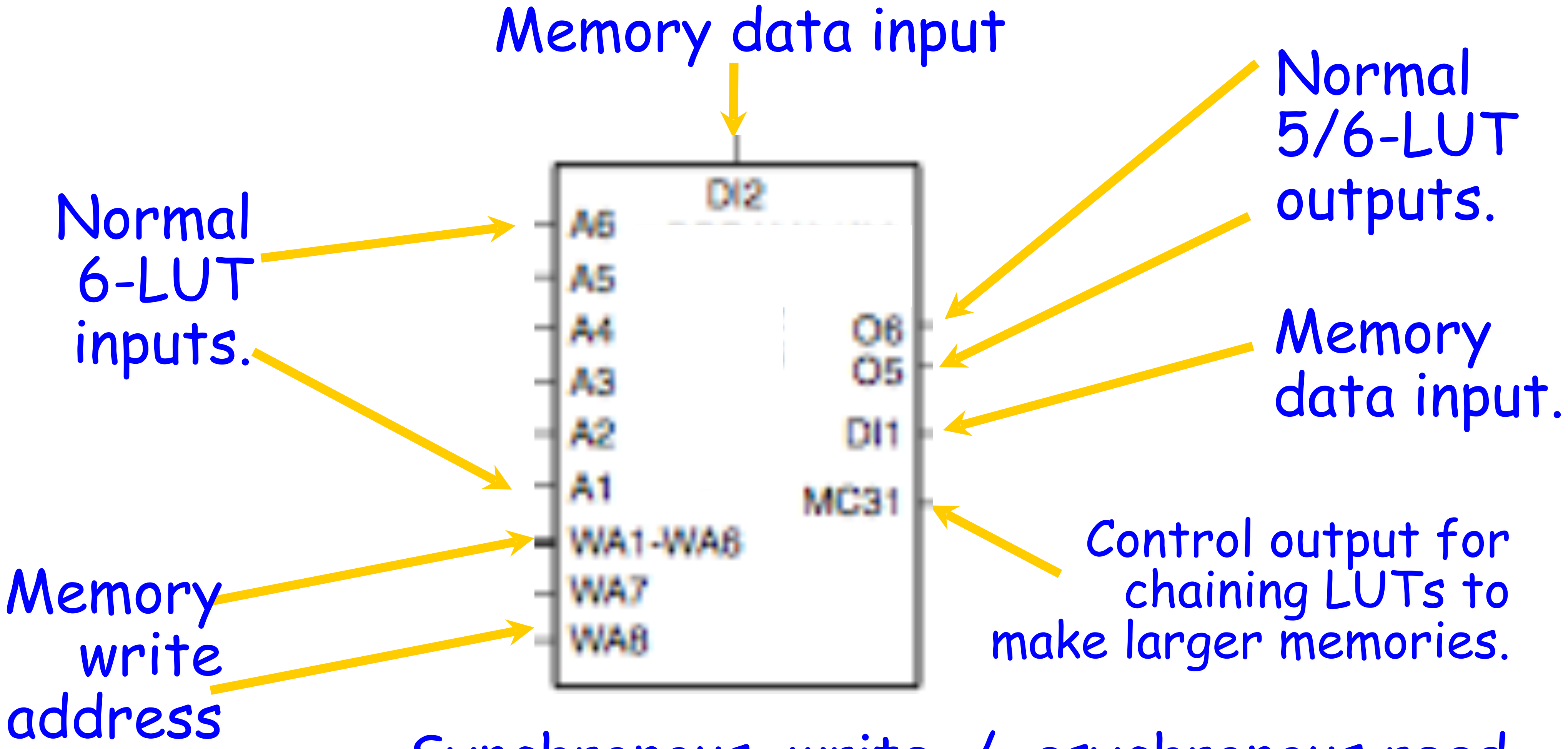
Adding a read port to a simple dual port (SDP) memory.

Example: given 1Kx8 SDP, want 1 write & 2 read ports.



Memory on Xilinx 7-series FPGAs

A SLICEM 6-LUT ...



Synchronous write / asychronous read

SLICEL vs SLICEM ...

SLICEL

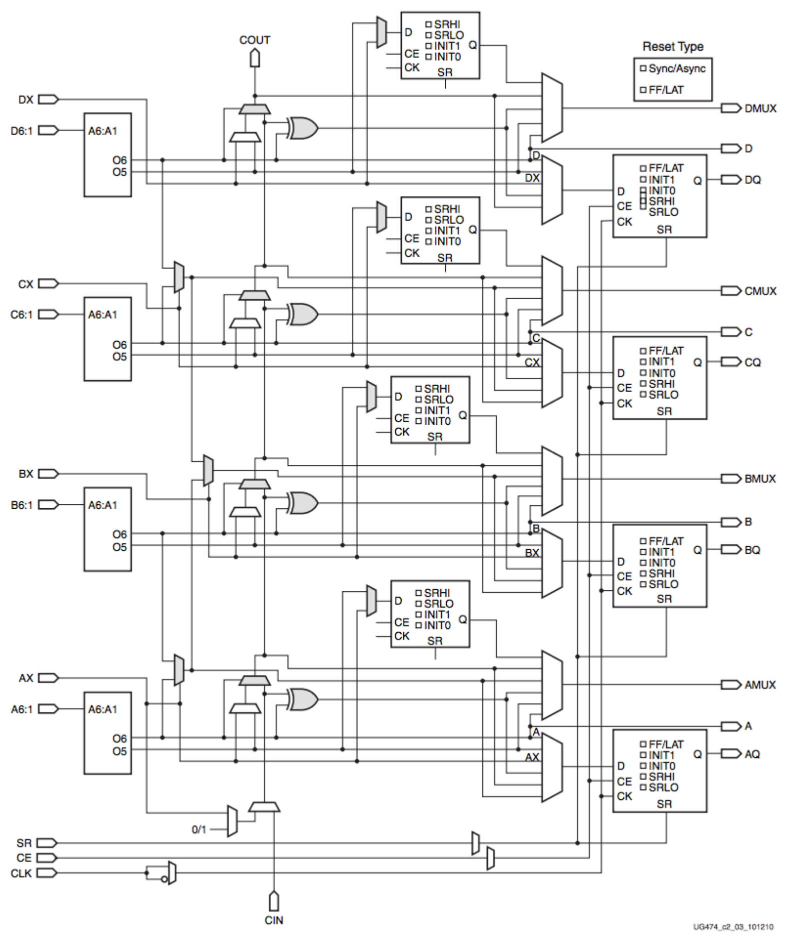


Figure 2-4: **Diagram of SLICEL**

SLICEM

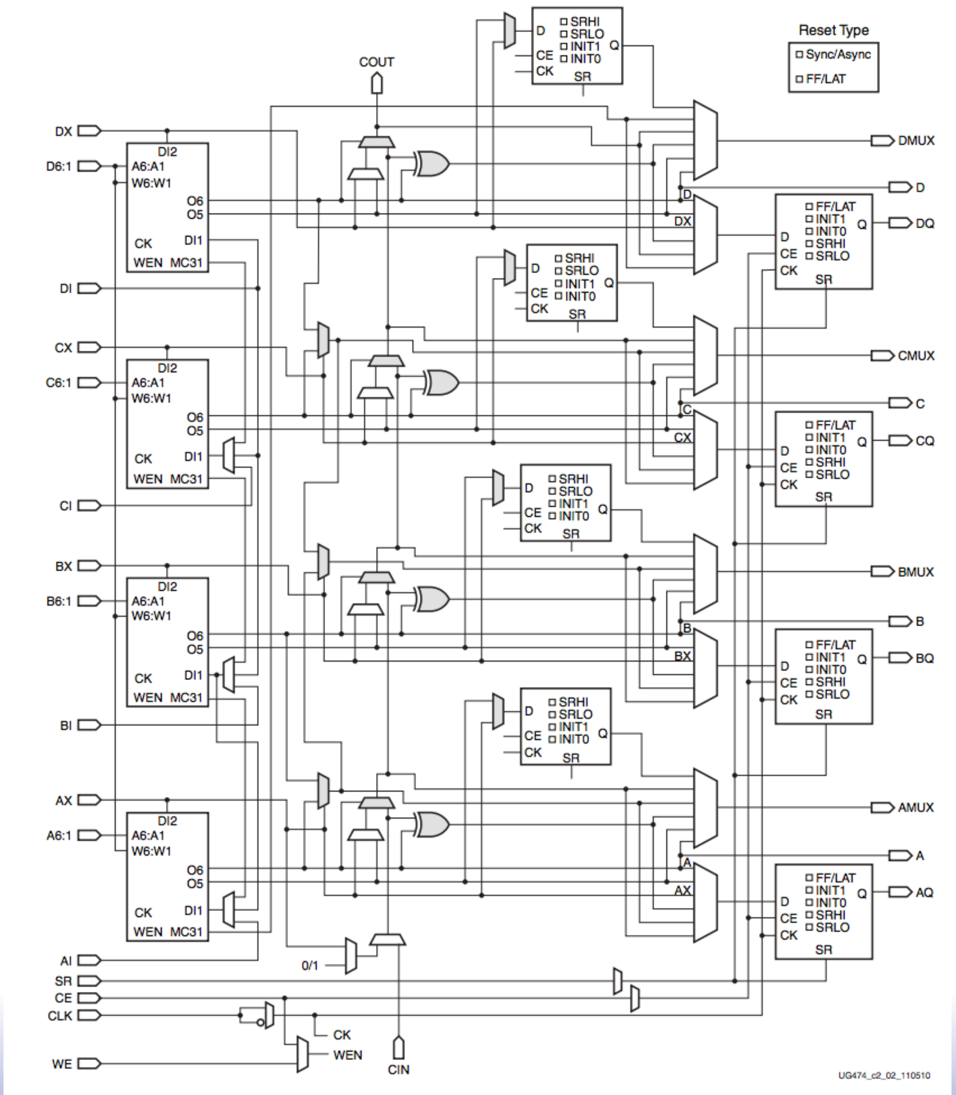


Figure 2-3: **Diagram of SLICEM**

SLICEM adds memory features to LUTs, + muxes.

Example Distributed RAM (LUT RAM)

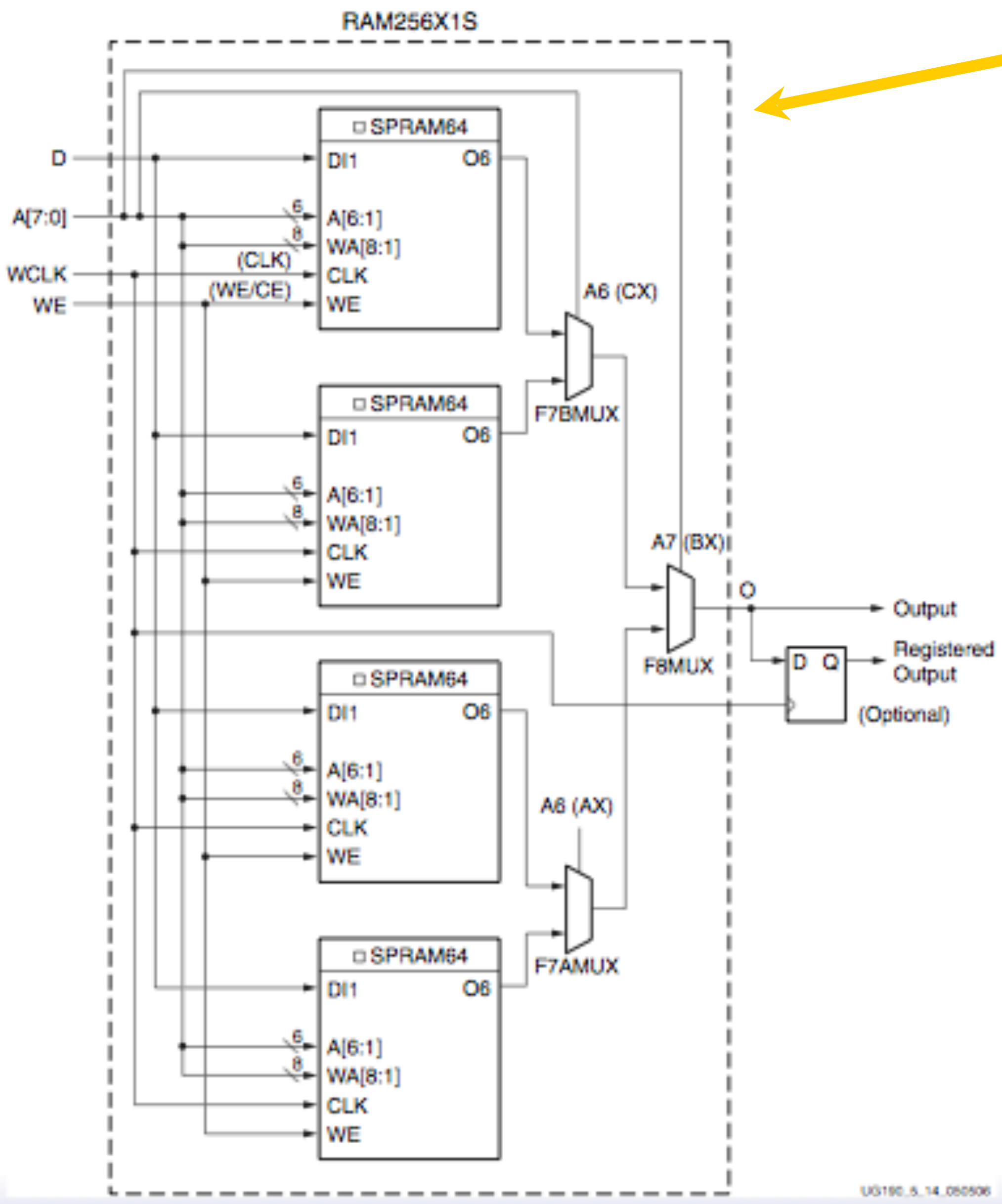


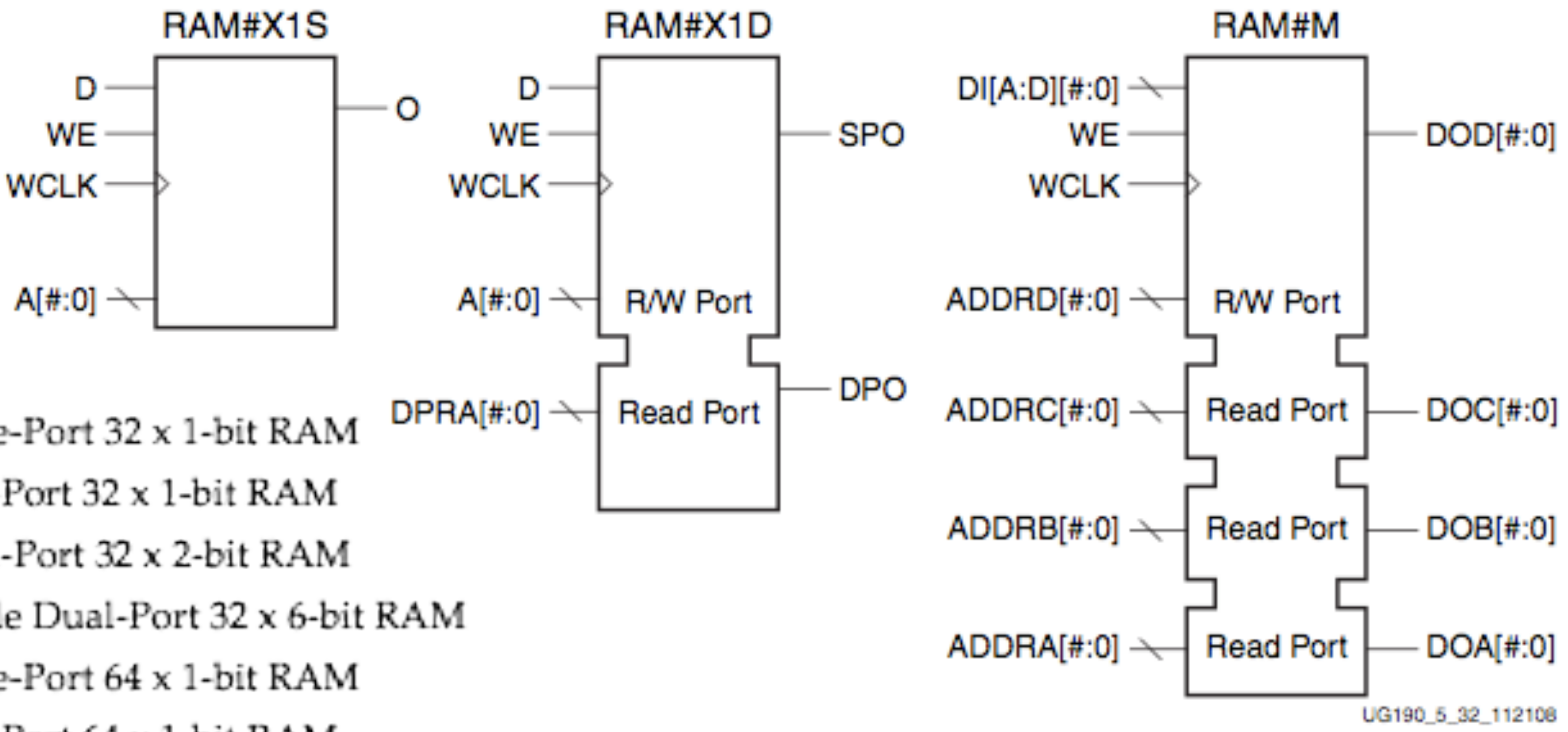
Figure 5-14: **Distributed RAM (RAM256X1S)**

Example configuration:
Single-port 256b x 1,
registered output.

These things are fast:
CLK can be as fast as
2.5ns on -1 speed grade
parts (400 MHz).

1.4ns from CLK edge to
updated data out

Distributed RAM Primitives



- Single-Port 32 x 1-bit RAM
- Dual-Port 32 x 1-bit RAM
- Quad-Port 32 x 2-bit RAM
- Simple Dual-Port 32 x 6-bit RAM
- Single-Port 64 x 1-bit RAM
- Dual-Port 64 x 1-bit RAM
- Quad-Port 64 x 1-bit RAM
- Simple Dual-Port 64 x 3-bit RAM
- Single-Port 128 x 1-bit RAM
- Dual-Port 128 x 1-bit RAM
- Single-Port 256 x 1-bit RAM

All are built from a single slice or less.

Remember, though, that the SLICEM LUT is naturally only 1 read and 1 write port.

Distributed RAM Timing

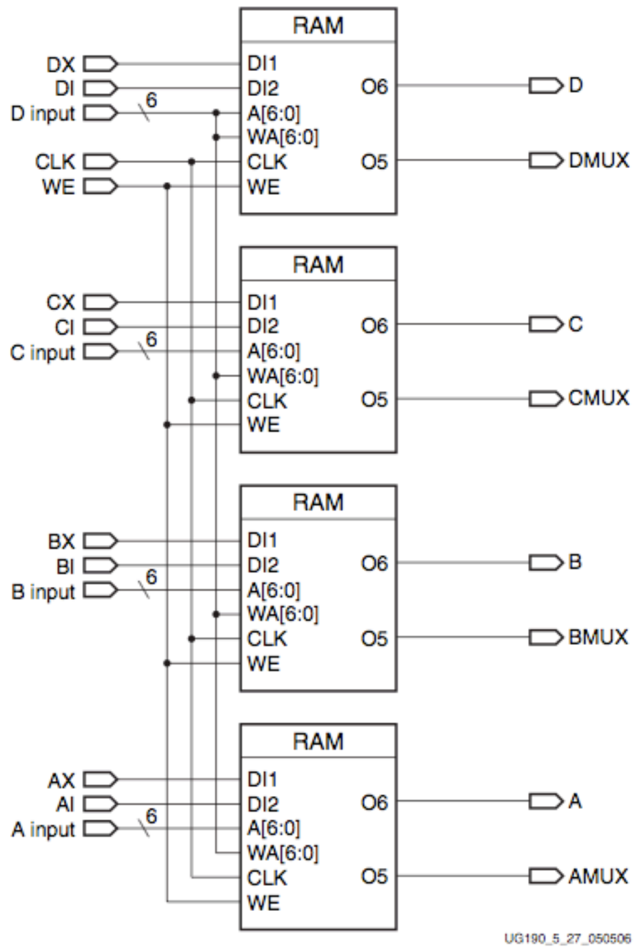
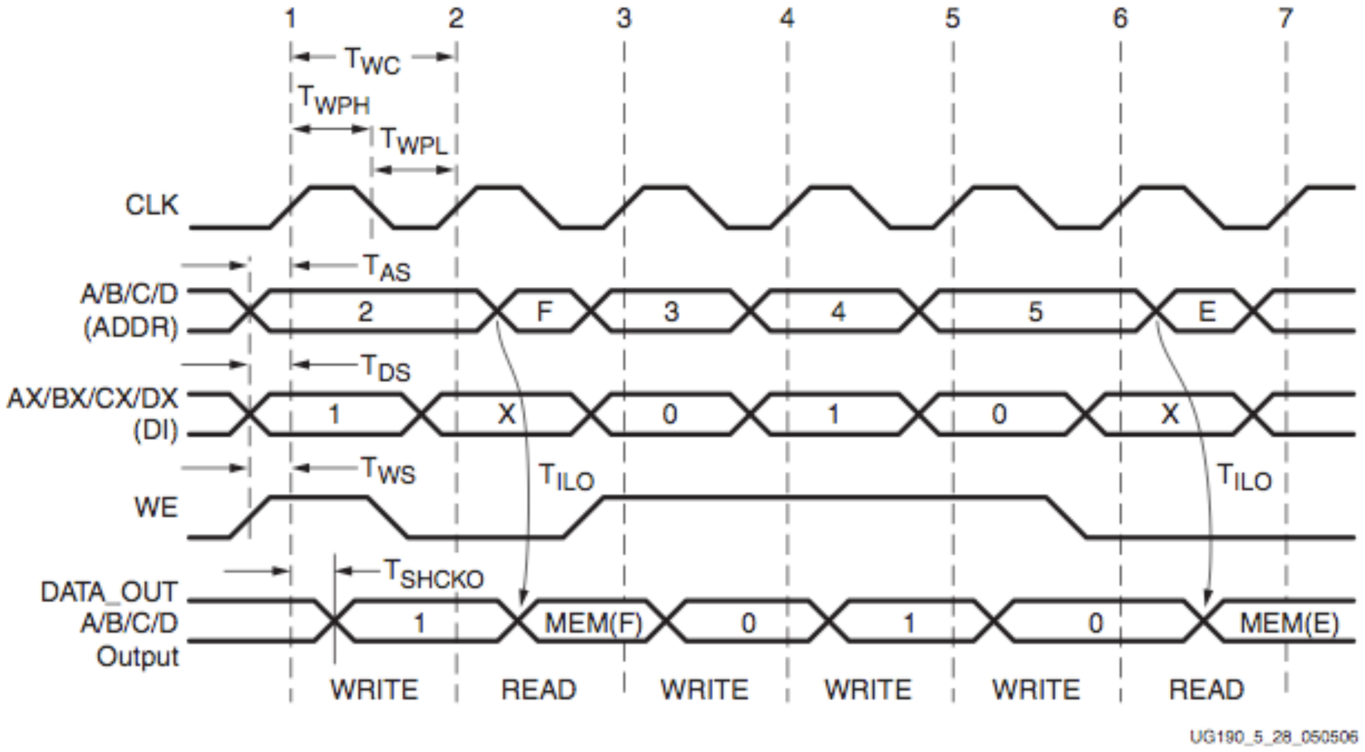


Figure 5-27: **Simplified Virtex-5 FPGA SLICEM Distributed RAM**



Distributed RAM and Regfiles

- The distributed RAM is literally ***designed*** to do RISC microprocessor register files
  - A 32 register, 32b, 2 asynchronous read ports, 1 synchronous write port register file takes just 12 slices
    - Using the 32x6 simple dual port mode
  - Bonus: Asynchronous read, synchronous write...
    - So if you write-back on positive edge, you don't need forwarding from the WB stage...
- Only special case control logic for a typical RISC is write suppression for register 0
  - Which takes a single 6-LUT to implement

But Also Shift Registers...

- Each LUT can act as a 32b Shift register...
  - Taking advantage of internal latch-based structure of the LUT
- Amount you actually read can be variable:
  - Since you have the address lines on the LUT to select the bit in question
- Can cascade from the previous LUT
  - To make much longer shift registers
- Can also do a non-cascadeable 16x2 shift register in one LUT

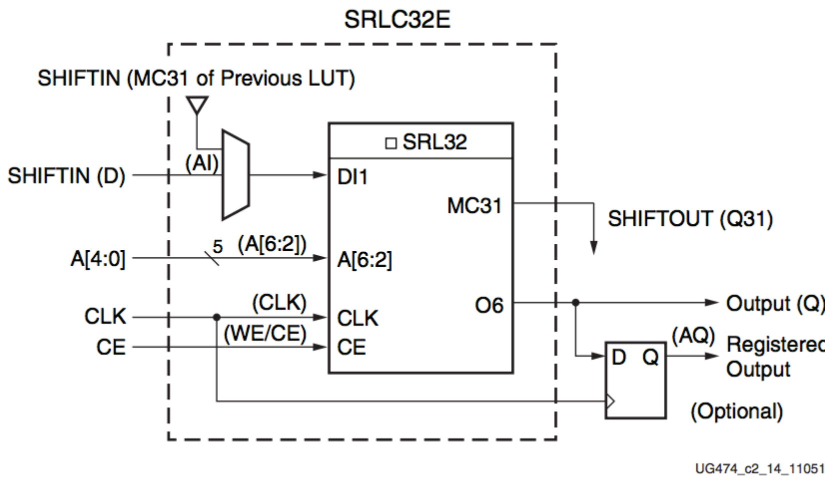


Figure 2-15: **32-Bit Shift Register Configuration**

Cascaded Shift Register

- Max is 128b
  - All 4 LUTs in a slice
  - Fully addressable
- If you need larger...
  - You can stitch together multiple slices but you no longer have the muxes and dedicated logic

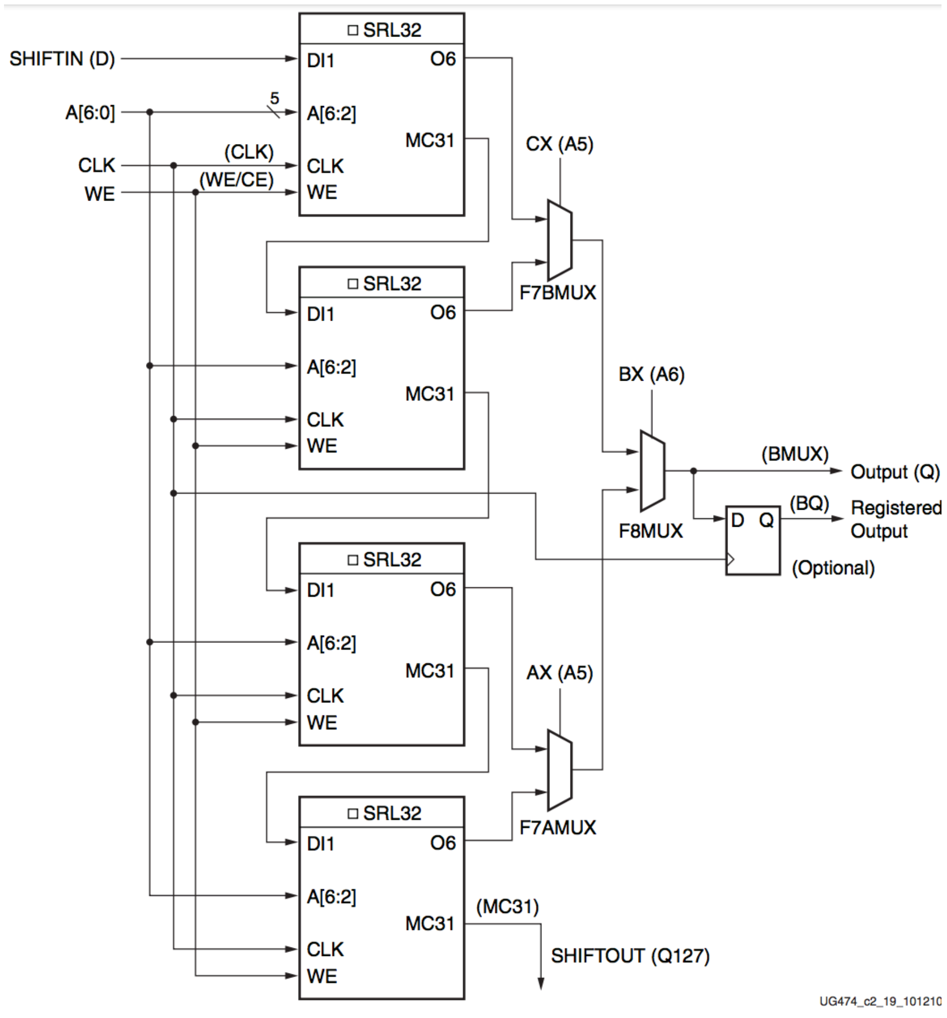
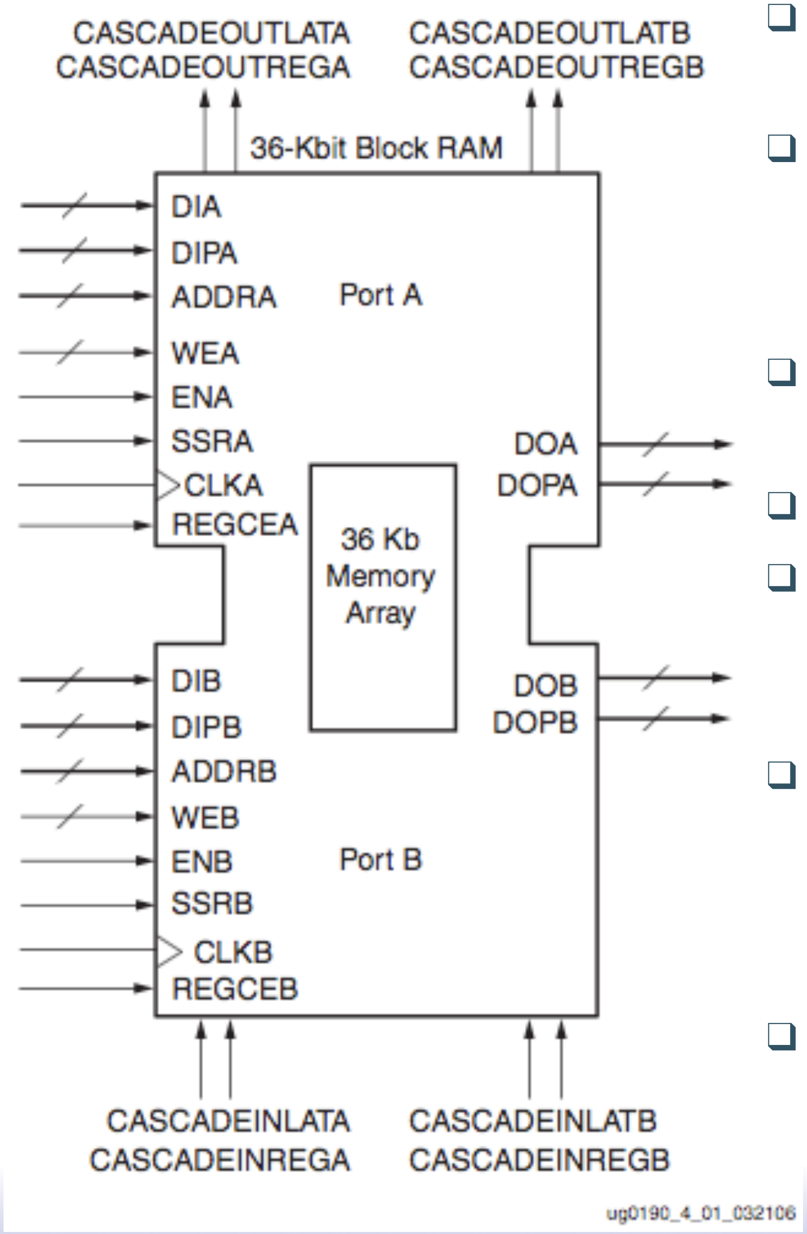


Figure 2-20: **128-Bit Shift Register Configuration**

BlockRAM

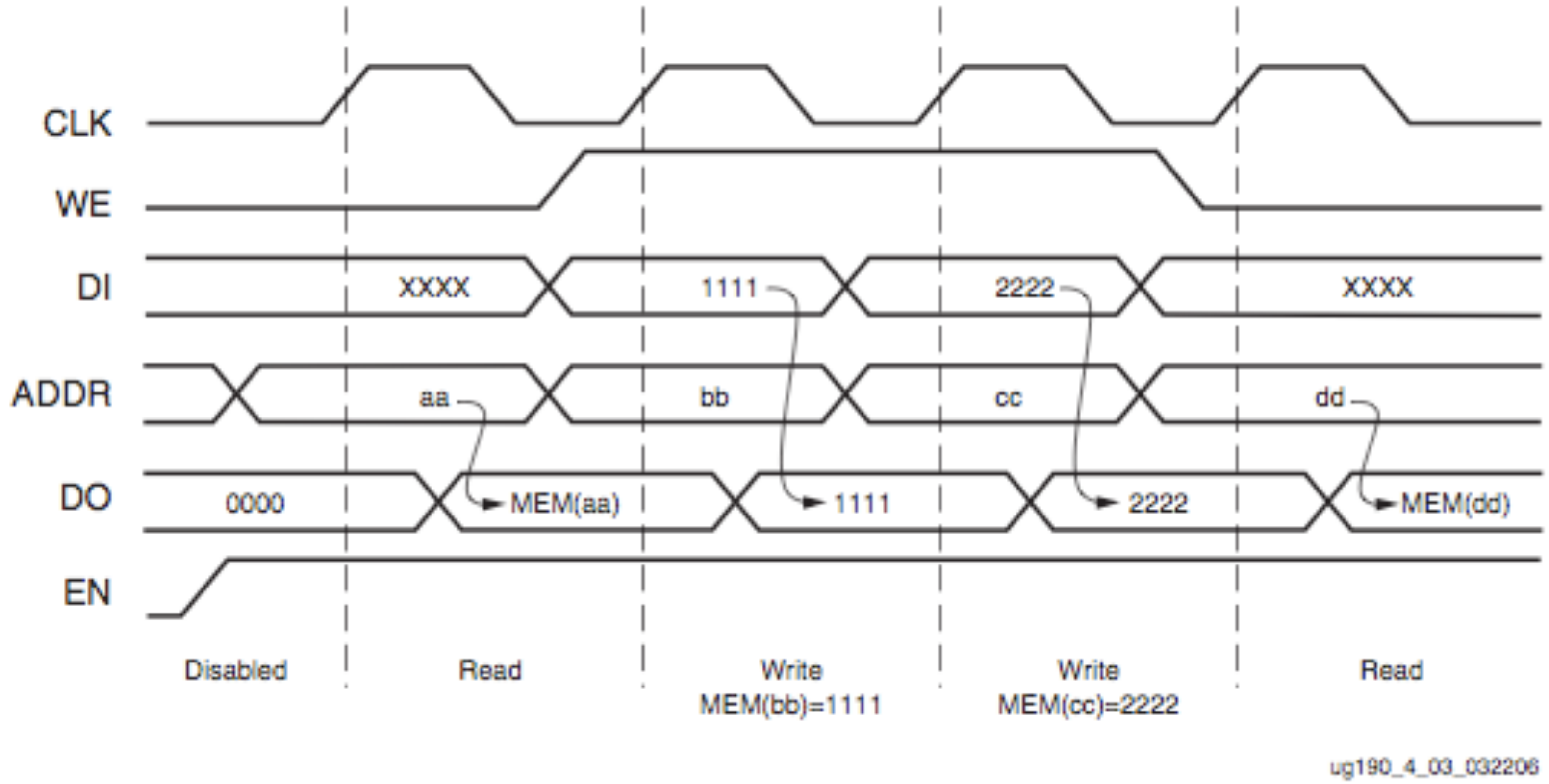
- There are also vertical columns of memory
  - 36 Kb memories
  - Split into 18 Kb halves
- Each half is independent

Block RAM Overview



- 36K bits of data total, can be configured as:
  - 2 independent 18Kb RAMs, or one 36Kb RAM.
- Each 36Kb block RAM can be configured as:
  - 64Kx1 (when cascaded with an adjacent 36Kb block RAM), 32Kx1, 16Kx2, 8Kx4, 4Kx9, 2Kx18, or 1Kx36 memory.
- Each 18Kb block RAM can be configured as:
  - 16Kx1, 8Kx2, 4Kx4, 2Kx9, or 1Kx18 memory.
- Write and Read are synchronous operations.
- The two ports are symmetrical and totally independent (can have different clocks), sharing only the stored data.
- Each port can be configured in one of the available widths, independent of the other port. The read port width can be different from the write port width for each port.
- The memory content can be initialized or cleared by the configuration bitstream.

Block RAM Timing



- Optional output register, would delay appearance of output data by one cycle.
- Maximum clock rate: roughly 400MHz depending on mode

More on BlockRAMs

- The two 18Kb halves can be fully independent dual-ported memories
  - Artifact of treating 36Kb as the BlockRAM is as much a historical accident: Virtex 5 was a bit more limited when addressing independent 18Kb blocks
- You also have simple-dual-port mode
  - Which ***doubles*** the available output width to 36b for 18Kb mode and 72 for 36Kb mode
- The extra bits are for parity/CRC
  - Only available on wide (9b outputs or wider)

BlockRAM performance

- On clk, about 2.5ns to data out for read
  - But can pipeline to reduce the delay to 0.9ns with embedded output registers
- Max clock frequency >300 MHz
  - Depending on mode of operation and speed grade, slowest is things like a FIFO with CRC checking. It can be up to 500 MHz for simple modes on fastest speed grade
  - Slows down somewhat if using built-in ECC

Zynq XCZ7020 Size & Memory

- 26,000 slices
  - Only some of which are SLICE-Ms
    - I can't find the ratio... :(
- 140 BlockRAMs
  - Each one is a 36Kb memory...

BlockRAMs as FIFOs...

- What is one of the most common memory use? Why FIFOs
  - So why should you have to build control logic every time you want a FIFO?
- Have built-in FIFO modes
  - Single reader, single writer, same width
    - Can't play width-shifting games in FIFO mode
- Two modes:
  - Synchronous: Same clock for read & write -> better latency on request
  - Asynchronous: Two independent clocks -> great for crossing clock domains

Simpler Memory Interfaces: AXI-4

- Xilinx provides 4 AXI slave-interfaces on the Zynq
  - These connect from the FPGA to the memory bus, enabling the programmable logic to access the DRAM
  - 32b or 64b wide interfaces
  - Support bursts for efficiency
- Abstracts away the DRAM... In theory...
  - You still want to read in ways that make the DRAM happy

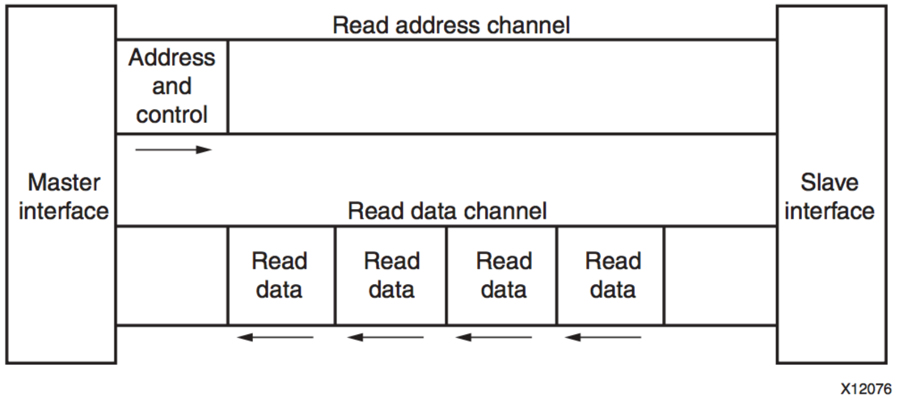
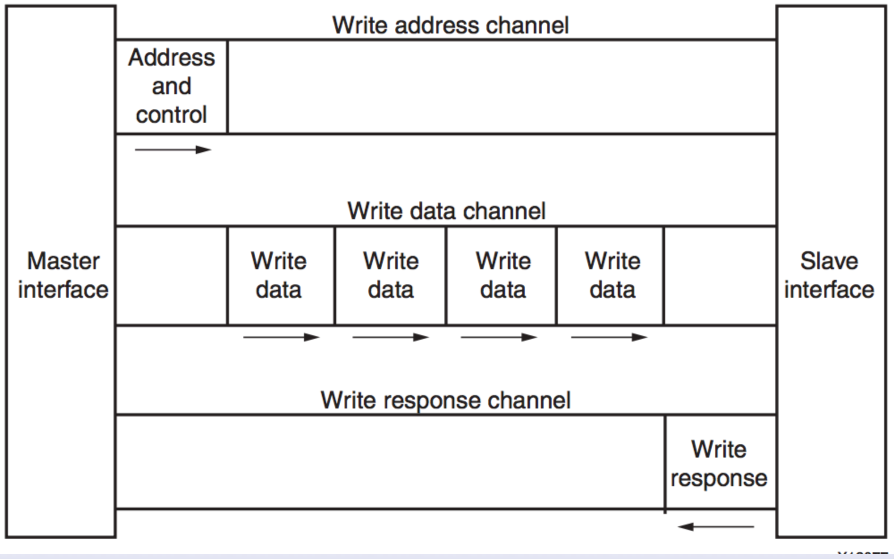


Figure 1-1: **Channel Architecture of Reads**

Figure 1-2 shows how a Write transaction uses the Write address, Write data, and Write response channels.



Write address channel
Address and control
Master interface
Slave interface
Write data channel
Write data
Write data
Write data
Write data
Write response channel
Write response