EECS 150 - Components and Design Techniques for Digital Systems

Lec 13 – Storage: Regs, SRAM, ROM

David Culler
Electrical Engineering and Computer Sciences
University of California, Berkeley

http://www.eecs.berkeley.edu/~culler
http://inst.eecs.berkeley.edu/~cs150

Review: Timing

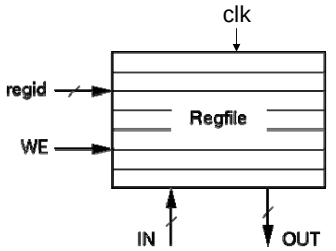
- All gates have delays
  - RC delay in driving the output
- Wires are distributed RCs
  - Delays goes with the square of the length
- Source circuits determines strength
  - Serial vs parallel
- Delays in combinational logic determine by
  - Input delay
  - Path length
  - Delay of each gate along the path
  - Worst case over all possible input-output paths
- Setup and CLK-Q determined by the two latches in flipflop
- Clock cycle : $T_{cycle} \geq T_{CL}+T_{setup}+T_{clk \rightarrow Q}$ + worst case skew
- Delays can introduce glitches in combinational logic

Outline

- Memory concepts
- Register Files
- SRAM
- SRAM Access
- Multiported Memories
  - FIFOS
- ROM, EPROM, FLASH
- Relationship to Comb. Logic

Memory Basics

- Uses:
  Whenever a large collection of state elements is required.
  - data & program storage
  - general purpose registers
  - buffering
  - table lookups
  - CL implementation
- Types:
  - RAM - random access memory
  - ROM - read only memory
  - EPROM, FLASH - electrically programmable read only memory
- Example RAM: Register file from microprocessor



regid = register identifier (address of word in memory)
sizeof(regid) = log2(# of reg)
WE = write enable

Examples in your project

- **Local device configuration**
- **Registry of other devices**
- **Video object store**
- **Video object pixel maps**
- **FIFOs connecting peripherals to the core**
- **Audio storage**

Definitions

Memory Interfaces for Accessing Data

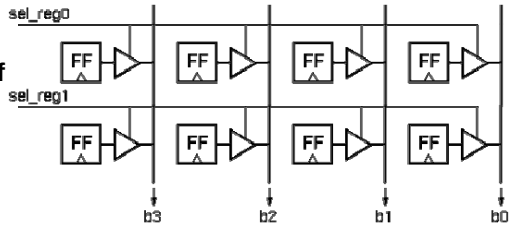
- Asynchronous **(unclocked):**
  A change in the address results in data appearing
- Synchronous **(clocked):**
  A change in address, followed by an edge on CLK results in data appearing or write operation occurring.

  A common arrangement is to have synchronous write operations and asynchronous read operations.

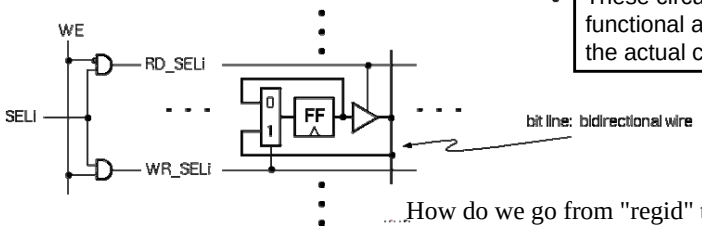
- Volatile:
  Looses its state when the power goes off.
- Nonvolatile:
  Retains it state when power goes off.

Register File Internals

- **For read operations, functionally the regfile is equivalent to a 2-D array of flip-flops with tristate outputs on each**
  - **MUX, but distributed**
  - **Unary control**



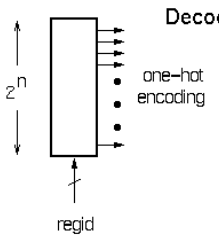
- **Cell with added write logic:**

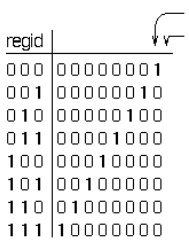


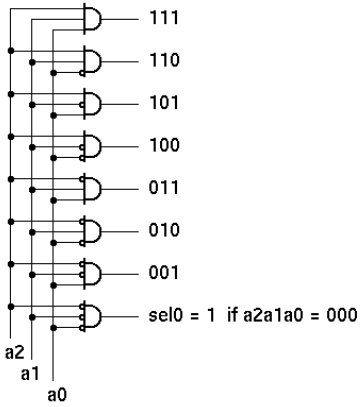
These circuits are just functional abstractions of the actual circuits used.

How do we go from "regid" to "SEL"?

Regid (address) Decoder



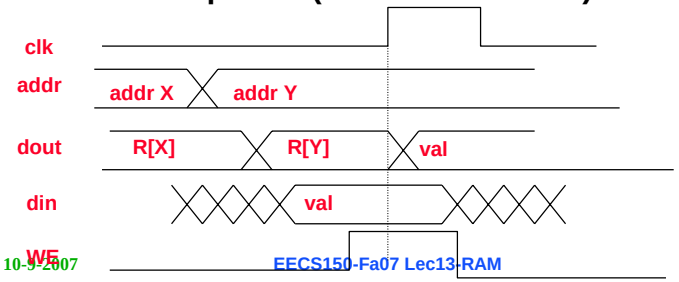




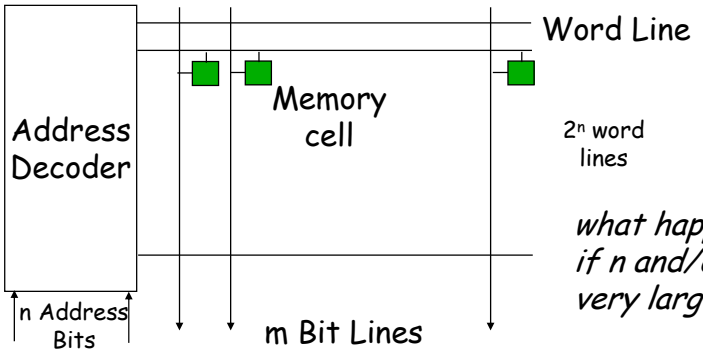
- The function of the address decoder is to generate a one-hot code word from the address.
  - Binary -> unary
  - Simplified DEMUX
- The output is used for row selection.
- Many different circuits exist for this function. A simple one is shown.
- Where have you seen this before?

Accessing Register Files

- Read: output is a combinational function of the address input
  - Change address, see data from a different word on the output
  - Regardless of clock
- Write is synchronous
  - If enabled, input data is written to selected word on the clock edge
- Often multi-ported (more on that later)



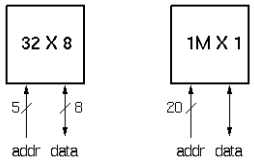
Basic Memory Subsystem Block Diagram



RAM/ROM naming convention:

32 X 8, "32 by 8" => 32 8-bit words

1M X 1, "1 meg by 1" => 1M 1-bit words

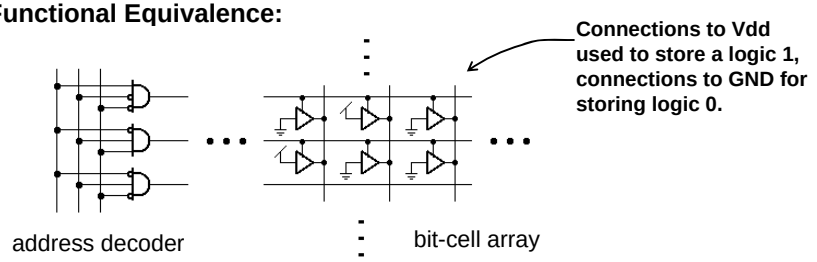


Memory Components Types:

- **Volatile:**
  - Random Access Memory (RAM):
    - » SRAM "static"
    - » DRAM "dynamic"
- **Non-volatile:**
  - Read Only Memory (ROM):
    - » Mask ROM "mask programmable"
    - » EPROM "electrically programmable"
    - » EEPROM "erasable electrically programmable"
    - » FLASH memory - similar to EEPROM with programmer integrated on chip

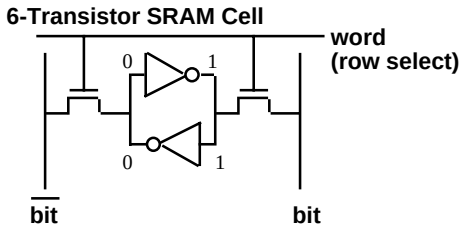
Read Only Memory (ROM)

- Simplified form of memory. No write operation needed.
- Functional Equivalence:



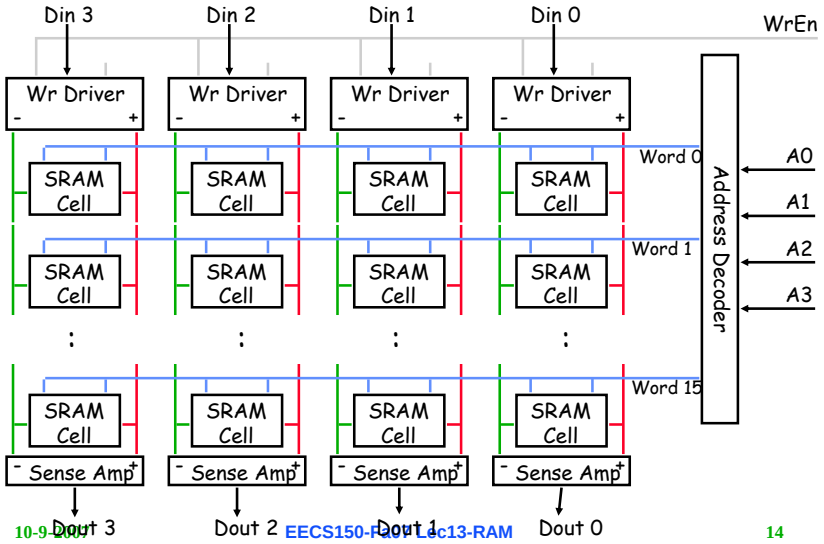
- Full tri-state buffers are not needed at each cell point.
- In practice, single transistors are used to implement zero cells. Logic one's are derived through *precharging* or bit-line *pullup transistor*.

Static RAM Cell

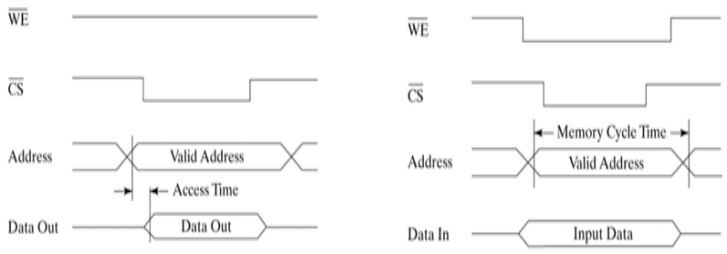


- **Read:**
  - 1. Select row
  - 2. Cell pulls one line low and one high
  - 3. Sense output on bit and $\overline{bit}$
- **Write:**
  - 1. Drive bit lines (e.g, bit=1, $\overline{bit}$=0)
  - 2. Select row
- **Why does this work?**
  - When one bit-line is low, it will force output high; that will set new state

Typical SRAM Organization: 16-word x 4-bit

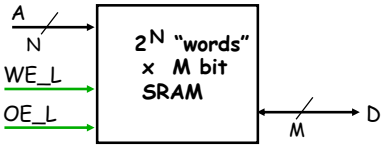


Simplified SRAM timing diagram



- Read: Valid address, then Chip Select
- Access Time: address good to data valid
  - even if not visible on out
- Cycle Time: min between subsequent mem operations
- Write: Valid address and data with WE_l, then CS
  - Address must be stable a setup time before WE and CS go low
  - And hold time after one goes high
- When do you drive, sample, or Z the data bus?

Logic Diagram of a Typical SRAM



- Write Enable is usually active low (WE_L)
- Din and Dout are combined to save pins:
- A new control signal, Output Enable (OE_L)
  - WE_L is asserted (Low), OE_L is unasserted (High)
    - » D serves as the data input pin
  - WE_L is unasserted (High), OE_L is asserted (Low)
    - » D is the data output pin
  - Neither WE_L and OE_L are asserted?
    - » Chip is disconneted
  - Never both asserted!

or chipSelect (CS) + WE

Example: ST microelectronics M68AW256M

FEATURES SUMMARY

- SUPPLY VOLTAGE: 2.7 to 3.6V
- 256K x 16 bits SRAM with OUTPUT ENABLE
- EQUAL CYCLE and ACCESS TIME: 55ns, 70ns
- SINGLE BYTE READ/WRITE
- LOW STANDBY CURRENT
- LOW V_{CC} DATA RETENTION: 1.5V
- TRI-STATE COMMON I/O
- AUTOMATIC POWER DOWN
- TSOP44, and TFBGA48 PACKAGES
  - Compliant with Lead-Free Soldering Processes
  - Standard or Lead-Free Option

TSOP44 Type II (ND)

TFBGA48 (ZH) 6 x 8mm

TFBGA48 (ZB) 7 x 8mm

Figure 2. Logic Diagram

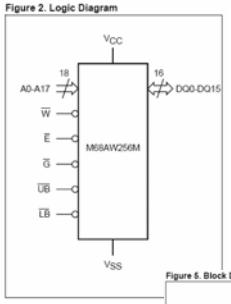
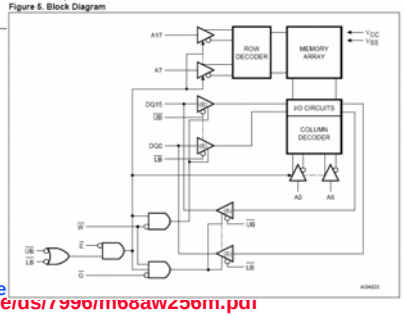


Table 1. Signal Names

A0-A17	Address Inputs
DQ0-DQ15	Data Input/Output
$\overline{E}$	Chip Enable
$\overline{G}$	Output Enable
$\overline{W}$	Write Enable
$\overline{UB}$	Upper Byte Enable Input
$\overline{LB}$	Lower Byte Enable Input
V_{CC}	Supply Voltage
V_{SS}	Ground
NC	Not Connected Internally
DU	Don't Use as Internally Connected

Figure 5. Block Diagram

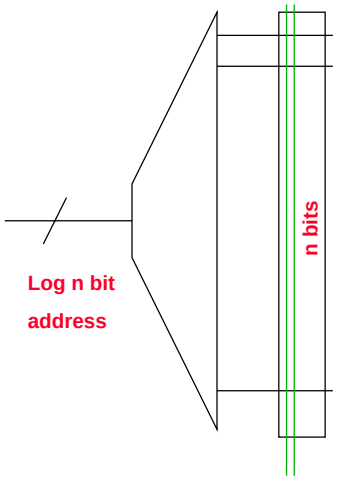


http://www.st.com/stonline/products/literature/ds/7996/m68aw256m.pdf

Administration and Announcements

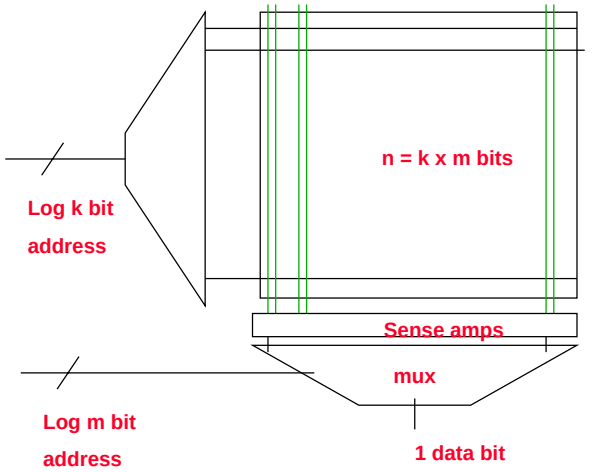
- Thanks for feedback on survey
  - Slow down – course and lecture
  - More examples – lecture and lab lecture
  - Mid term too long
- Lab lecture => Pre Lab => Design review => Execute => Check off
  - You need to pay attention to lab lecture on what to do next while executing current lab.
  - Do the prelab before lab. The beginning of lab is turn in design review document of current and check off of previous.
- Update on lab check offs
  - 4 days of slip that you can use as needed (but cannot extend into weekend)
  - Can accepts black box on lab to catch up
    - » Recoup 50% if implement your own within 2 weeks
- All homeworks graded. On-line grades available for labs, HWs, and Mid
- Solution for current HW will be posted on Friday!

What happens when # bits gets large



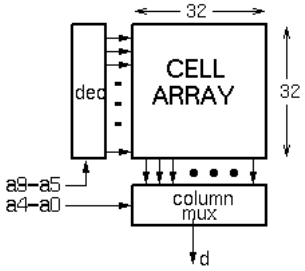
- Big slow decoder
- Bit lines very log
  - Large distributed RC load
- Treat output as differential signal, rather than rail-to-rail logic
  - Sense amps on puts
  - Can 'precharge' both bit lines high, so cell only has to pull one low
- ==> Make it shorter and wider

Inside a Tall-Thin RAM is a short-fat RAM

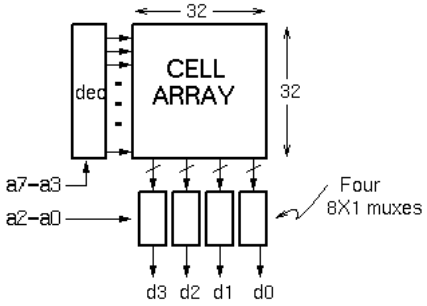


Column MUX

- Controls physical *aspect ratio*
  - Important for physical layout and to control delay on wires.
- In DRAM, allows time-multiplexing of chip address pins (later)



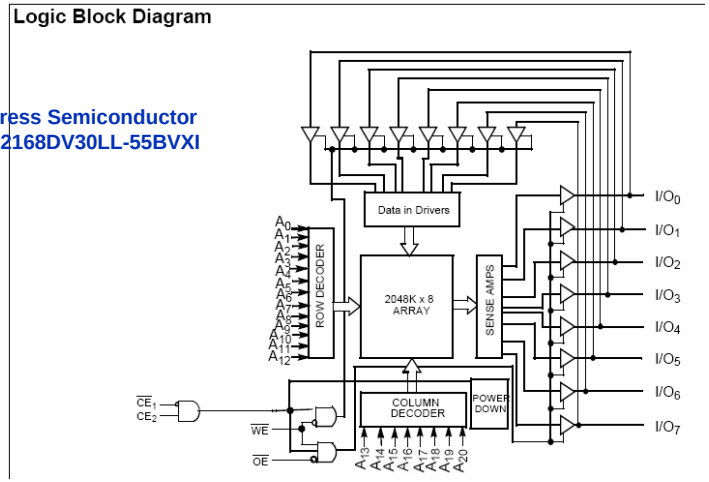
1K X 1 ROM



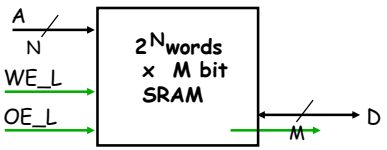
256 X 4 ROM

Example 2: 16M (2M x 8)

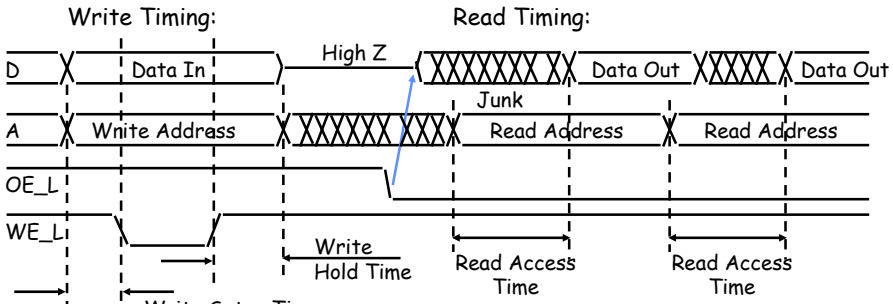
Cypress Semiconductor
CY62168DV30LL-55BVXI



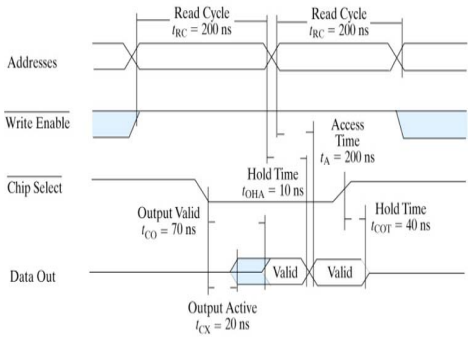
Typical SRAM Timing



OE determines direction
Hi = Write, Lo = Read
Writes are dangerous! Be careful!
Double signaling: OE Hi, WE Lo



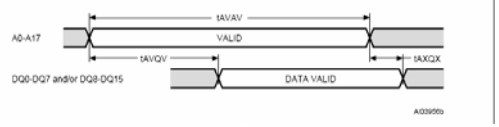
Read Series with CS



- Rate determined by cycle time
- Data valid: max(addr + Access time, CS + Tco)
- Remain valid TOHA after addr changes
- Return to tri-state after read sequence

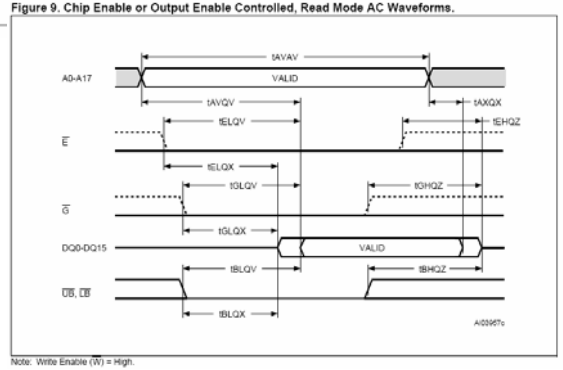
Example 1 continued: read

Figure 8. Address Controlled, Read Mode AC Waveforms



Note: $\overline{E}$ = Low, $\overline{G}$ = Low, $\overline{W}$ = High, $\overline{UB}$ = Low and/or $\overline{LB}$ = Low.

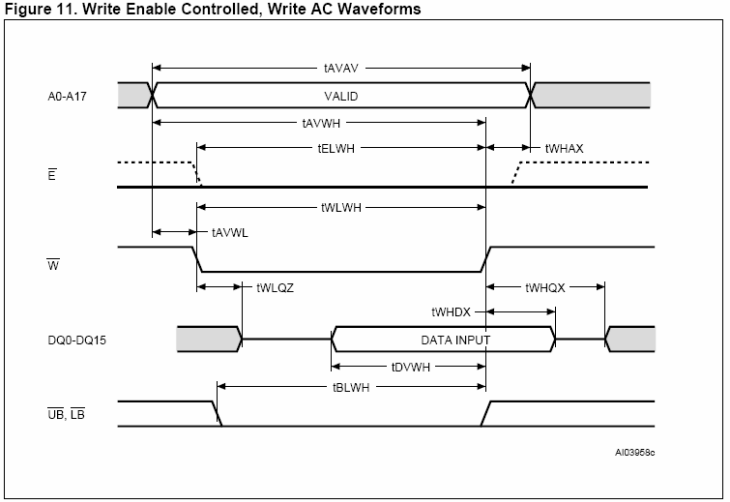
Figure 9. Chip Enable or Output Enable Controlled, Read Mode AC Waveforms.



Note: Write Enable ($\overline{W}$) = High.

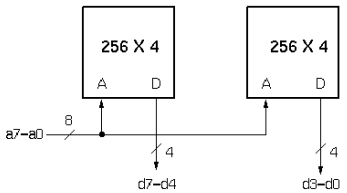
Example 1 continued: write

Figure 11. Write Enable Controlled, Write AC Waveforms

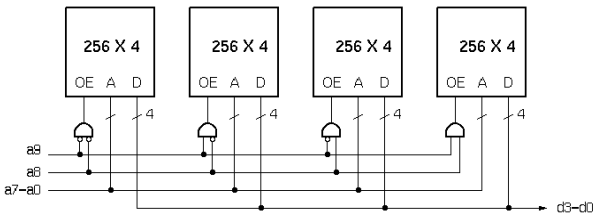


Cascading Memory Modules (or chips)

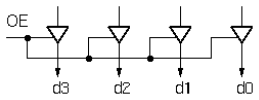
- **Example: assemble of 256 x 8 ROM using 256 x 4 modules:**



- **example: 1K x * ROM using 256 x 4 modules:**

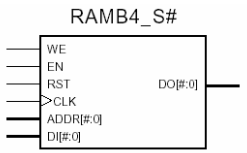


- **each module has tri-state outputs:**



Memory Blocks in FPGAs

- **LUTs can double as small RAM blocks:**
  - **4-LUT is really a 16x1 memory. Normally we think of the contents being written from the configuration bit stream, but Virtex architecture (and others) allow bits of LUT to be written and read from the general interconnect structure.**
  - **achieves 16x density advantage over using CLB flip-flops.**
  - **Furthermore, the two LUTs within a slice can be combined to create a 16 x 2-bit or 32 x 1-bit synchronous RAM, or a 16x1-bit dual-port synchronous RAM.**
  - **The Virtex-E LUT can also provide a 16-bit shift register of adjustable length.**



- **Newer FPGA families include larger on-chip RAM blocks (usually dual ported):**
  - **Called** block selectRAMs **in Xilinx Virtex series**
  - **4k bits each**

Synchronous SRAM

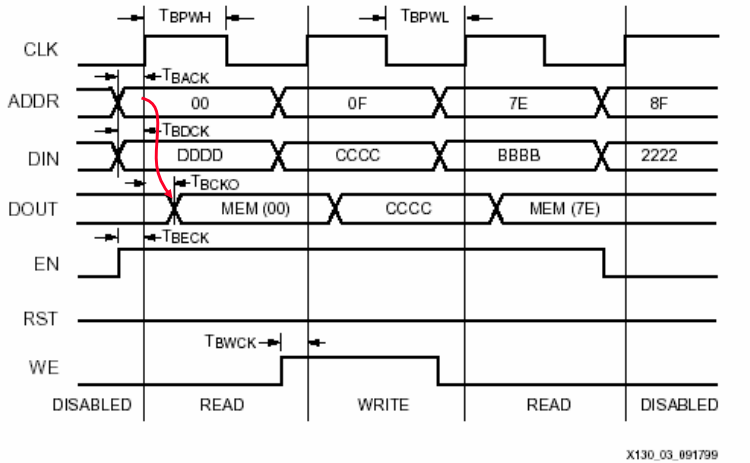


Figure 3: **Timing Diagram for Single-Port Block SelectRAM+ Memory**

Verilog for Virtex LUT RAM

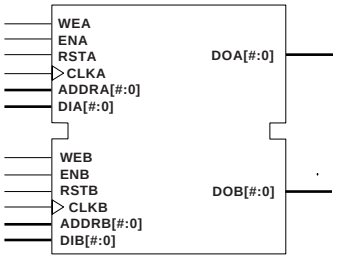
```
module ram16x1(q, a, d, we, clk);
output q;
input d;
input [3:0] a;
input clk, we;
reg mem [15:0];

always @(posedge clk) begin
   if(we)
      mem[a] <= d;
end
assign q = mem[a];
endmodule
```

Note: synchronous write and asynchronous read.

- **Deeper and/or wider RAMs can be specified and the synthesis tool will do the job of wiring together multiple LUTs.**
- **How does the synthesis tool choose to implement your RAM as a collection of LUTs or as block RAMs?**

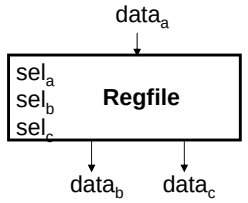
Virtex "Block RAMs"

- **Each block SelectRAM (block RAM) is a fully synchronous (synchronous write *and* read) dual-ported (true dual port) 4096-bit RAM with independent control signals for each port. The data widths of the two ports can be configured independently, providing built-in bus-width conversion.**
- **CLKA and CLKB can be independent, providing an easy way to "cross clock boundaries".**
- **Around 160 of these on the 2000E. Multiples can be combined to implement, wider or deeper memories.**
- ***See chapter 8 of Synplify reference manual on how to write Verilog for implied Block RAMs. Or instead, explicitly instantiate as primitive (project checkpoint will use this method).***



Multi-ported Memory

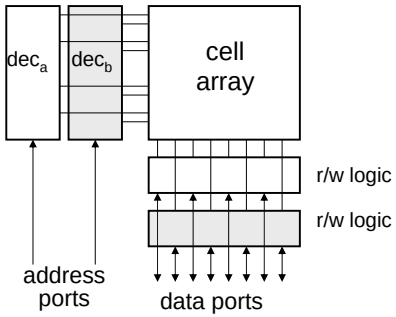
- **Motivation:**
  - **Consider CPU core register file:**
    - » **1 read or write per cycle limits processor performance.**
    - » **Complicates pipelining. Difficult for different instructions to simultaneously read or write regfile.**
    - » **Common arrangement in pipelined CPUs is 2 read ports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



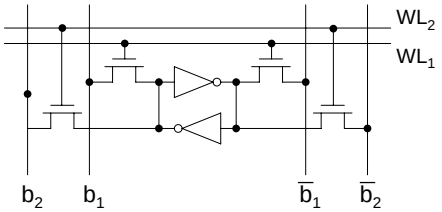
What do we need in the project?

Dual-ported Memory Internals

- Add decoder, another set of read/write logic, bits lines, word lines:



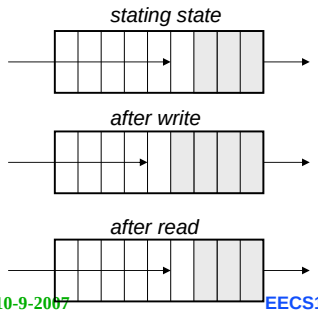
- Example cell: SRAM



- Repeat everything but cross-coupled inverters.
- This scheme extends up to a couple more ports, then need to add additional transistors.

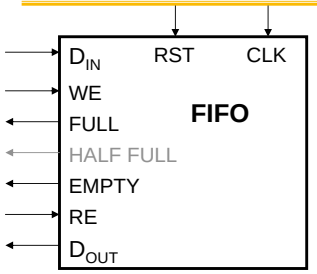
First-in-first-out (FIFO) Memory

- Used to implement *queues*.
- These find common use in computers and communication circuits.
- Generally, used for rate matching data producer and consumer:

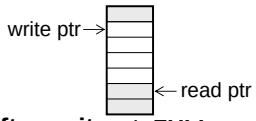


- Producer can perform many writes without consumer performing any reads (or vice versa). However, because of finite buffer size, on average, need equal number of reads and writes.
- Typical uses:
  - interfacing I/O devices. Example network interface. Data bursts from network, then processor bursts to memory buffer (or reads one word at a time from interface). Operations not synchronized.
  - Example: Audio output. Processor produces output samples in bursts (during process swap-in time). Audio DAC clocks it out at constant sample rate.

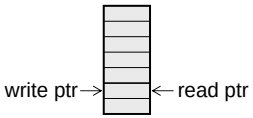
FIFO Interfaces



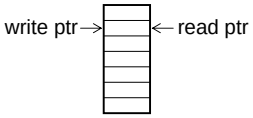
- Address pointers are used internally to keep next write position and next read position into a dual-port memory.



- If pointers equal after write ⇒ FULL:



- If pointers equal after read ⇒ EMPTY:



- After write or read operation, FULL and EMPTY indicate status of buffer.
- Used by external logic to control own reading from or writing to the buffer.
- FIFO resets to EMPTY state.
- HALF FULL (or other indicator of partial fullness) is optional.

FIFO Implementation

- Assume, dual-port memory with asynchronous read, synchronous write.
- Binary counter for each of read and write address. CEs controlled by WE and RE.
- Equal comparator to see when pointers match.
- Flip-flop each for FULL and EMPTY flags:

WE	RE	equal	$EMPTY_i$	$FULL_i$
0	0	0	0	0
0	0	1	$EMPTY_{i-1}$	$FULL_{i-1}$
0	1	0	0	0
0	1	1	1	0
1	0	0	0	0
1	0	1	0	1
1	1	0	0	0
1	1	1	$EMPTY_{i-1}$	$FULL_{i-1}$

Control logic with truth-table "draft"
"equal" determined after clock allows counter to tick
- when is empty and full set?
- could play funny games with clocking

See Xilinx application notes
Can waste a slot to simplify

Xilinx BlockRam Versions

- Our simple version has latency problems.
  - FULL and EMPTY signals are asserted near the end of the clock period.
- Xilinx version solves this by "predicting" when full or empty will be asserted on next write/read operation. Also uses BlockRam with synchronous reads.
- Available on Xilinx website along with "app note" – application note. These are linked to our page.
- Two versions available. Easy to modify to change the width if necessary.
- Will use the "cc" (common clock) version.

Non-volatile Memory

Used to hold fixed code (ex. BIOS), tables of data (ex. FSM next state/output logic), slowly changing values that persist over power off (date/time)

- Mask ROM
  - Used with logic circuits for tables etc.
  - Contents fixed at IC fab time (truly write once!)
- EPROM (erasable programmable) & FLASH
  - requires special IC process (floating gate technology)

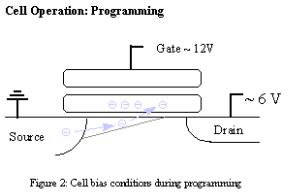


Figure 2: Cell bias conditions during programming

  - writing is slower than RAM. EPROM uses special programming system to provide special voltages and timing.
  - reading can be made fairly fast.
  - rewriting is very slow.
    - » erasure is first required , EPROM - UV light exposure, EEPROM – electrically erasable

FLASH Memory

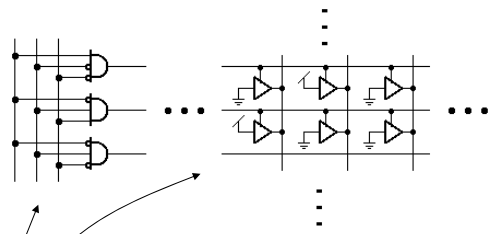
- Electrically erasable
- In system programmability and erasability (no special system or voltages needed)
- On-chip circuitry (FSM) and voltage generators to control erasure and programming (writing)
- Erasure happens in variable sized "sectors" in a flash (16K - 64K Bytes)

See: http://developer.intel.com/design/flash/
for product descriptions, etc.

- Compact flash cards are based on this type of memory.
  - NAND flash
  - Configuration memory, microcontrollers usually NOR flash

Relationship between Memory and CL

- Memory blocks can be (and often are) used to implement combinational logic functions:
- Examples:
  - LUTs in FPGAs
  - 1Mbit x 8 EPROM can implement 8 independent functions each of $\log_2(1M)=20$ inputs.
- The *decoder part* of a memory block can be considered a "minterm generator".
- The *cell array part* of a memory block can be considered an OR function over a subset of rows.



- The combination gives us a way to implement logic functions directly in sum of products form.
- Several variations on this theme exist in a set of devices called Programmable logic devices (PLDs)

A ROM as AND/OR Logic Device

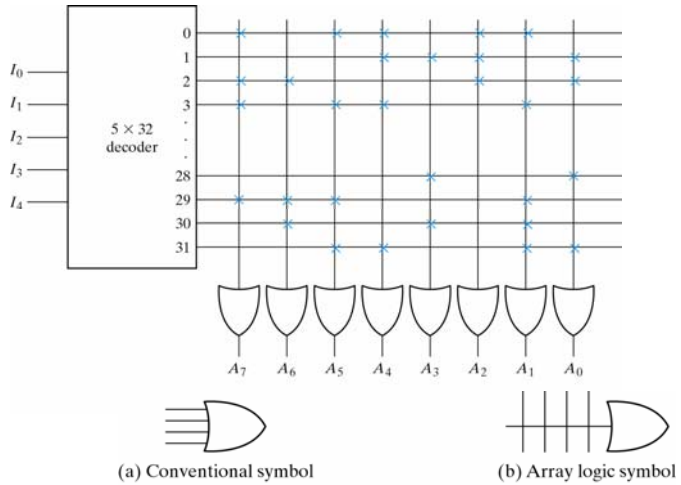


Fig. 7-1 Conventional and Array Logic Diagrams for OR Gate

PLD Summary

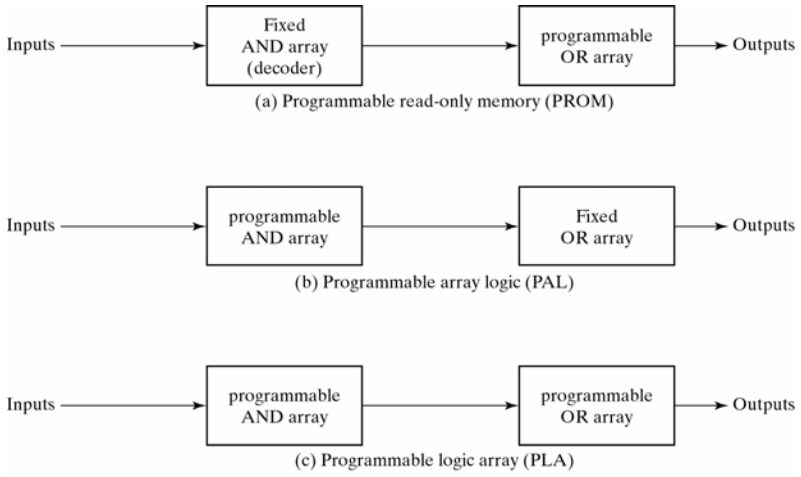


Fig. 7-13 Basic Configuration of Three PLDs

PLA Example

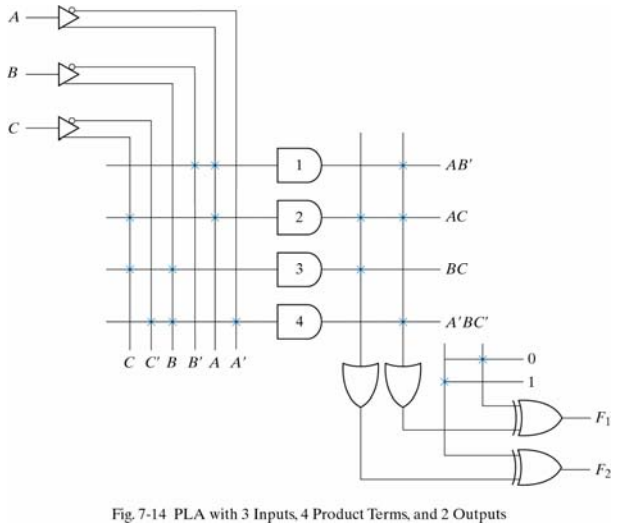


Fig. 7-14 PLA with 3 Inputs, 4 Product Terms, and 2 Outputs

PAL Example

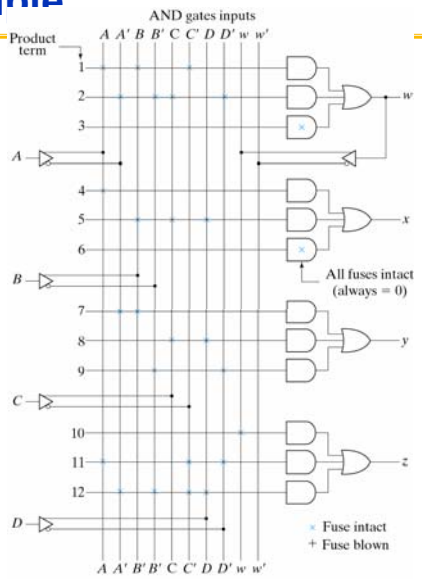


Fig. 7-17 Fuse Map for PAL as Specified in Table 7-6

Summary

- **Basic RAM structure**
  - Address decoder to select row of cell array
  - bit, ~bit lines to read & write
  - Sense difference in each bit
  - Column mux
- **Read/write protocols**
  - Synchronous (reg files, fpga block ram)
  - Asynchronous read, synchronous writes
  - Asynchronous
- **Multiported RAMs**
  - reg files and fifos
- **Non-volatile memory**
  - ROM, EPROM, EEPROM, FLASH
- **Memory as combinational logic**
- **Relationship to programmable logic**