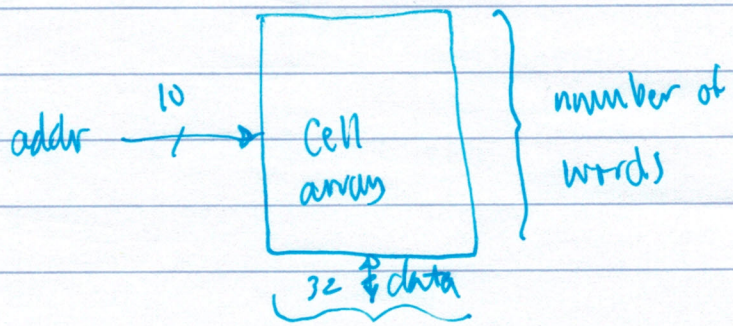
EEC 180

May 27, 2021

Memories III

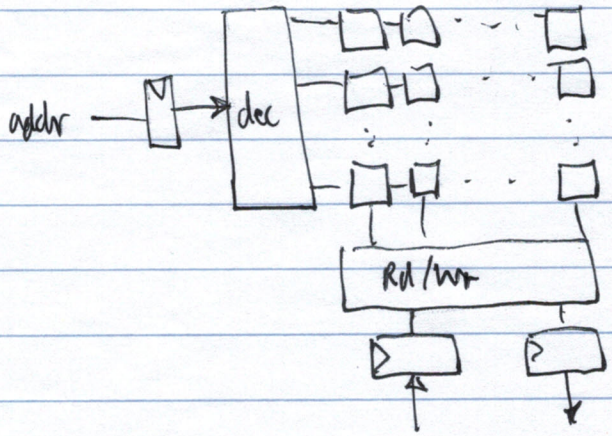
1024 word x 32 bit
= 32Kbit



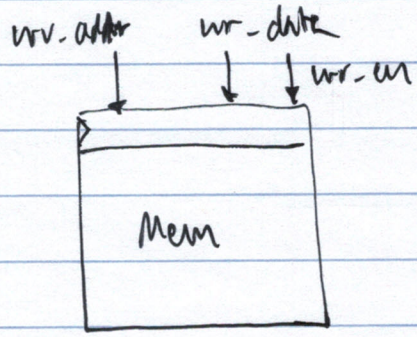
Total memory size = #words x # bits/word
$= 2^{addr\_width} \times$ #bits/word

word width

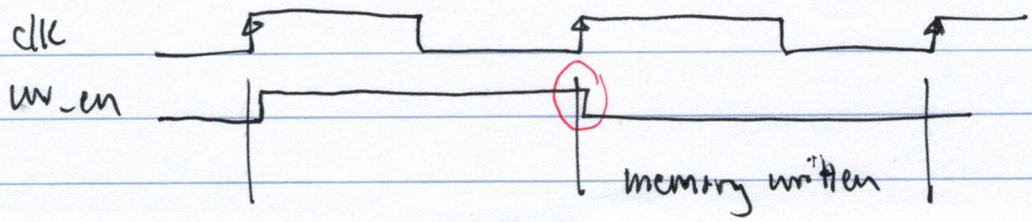
Timing



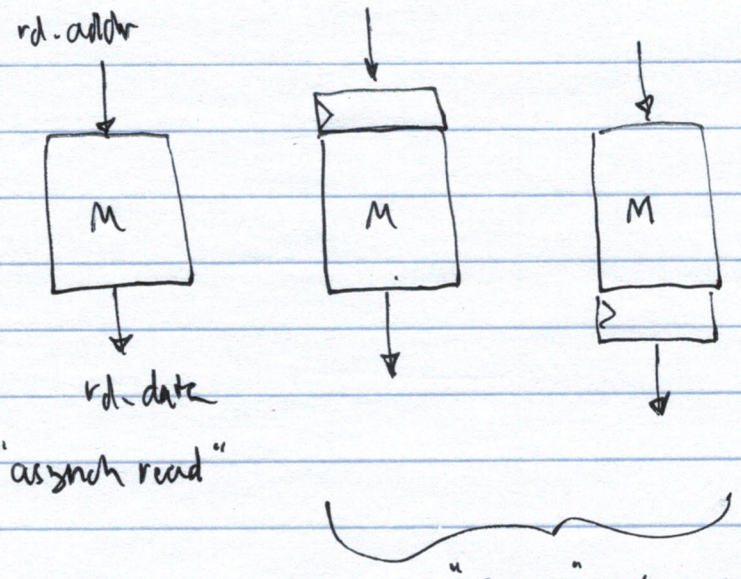
Memory write



Read After Write hazard
RAW



Memory Reads

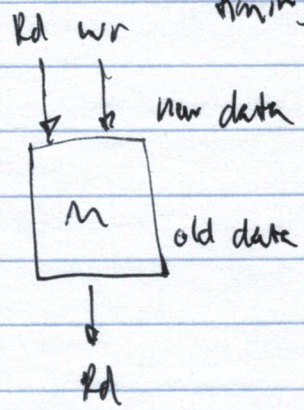


Simultaneous Rd + Wr

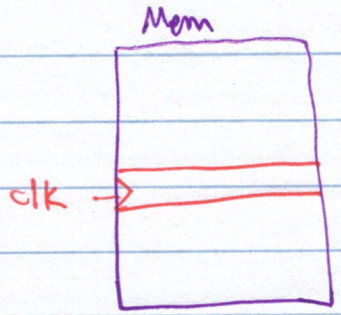
1) Rd returns old data

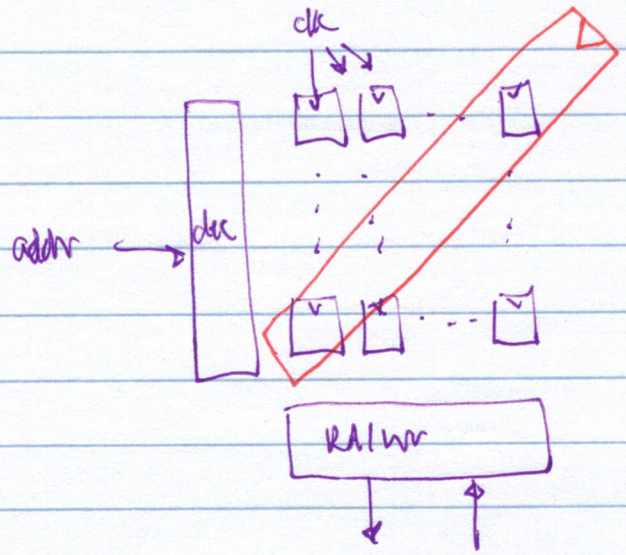
2) Rd returns new data

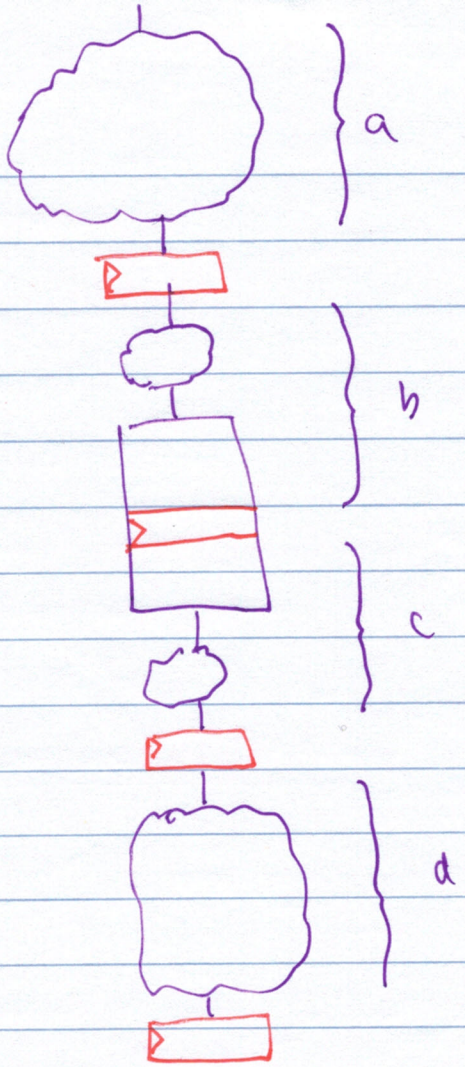
3) Don't care / don't know
   - old
   - new
   - some of each



Style #2 - built from FFs



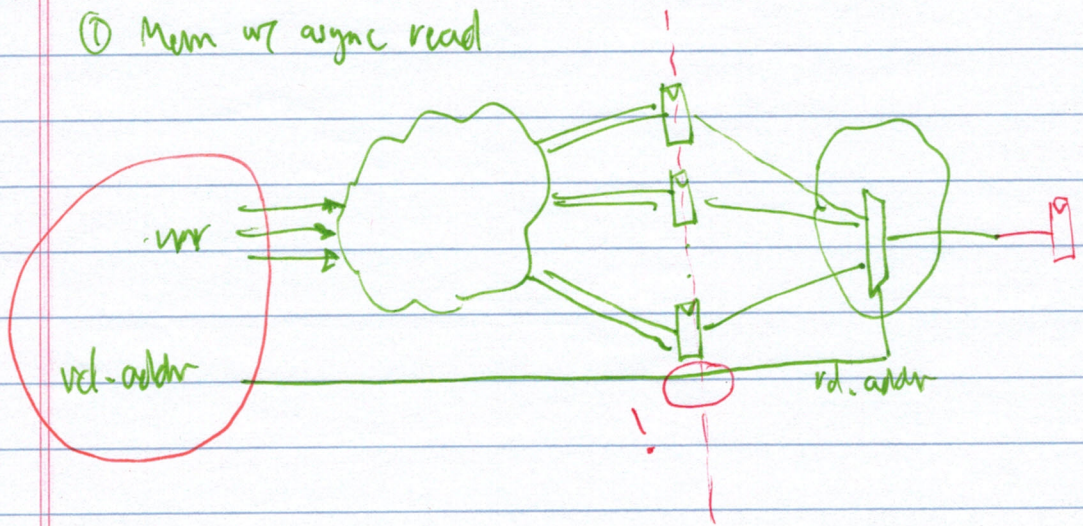




For max clk freq, a, b, c, d

should have ~ balanced delays

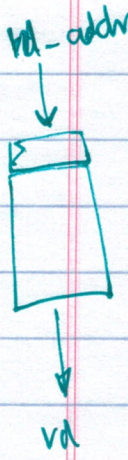
① Mem w/ async read

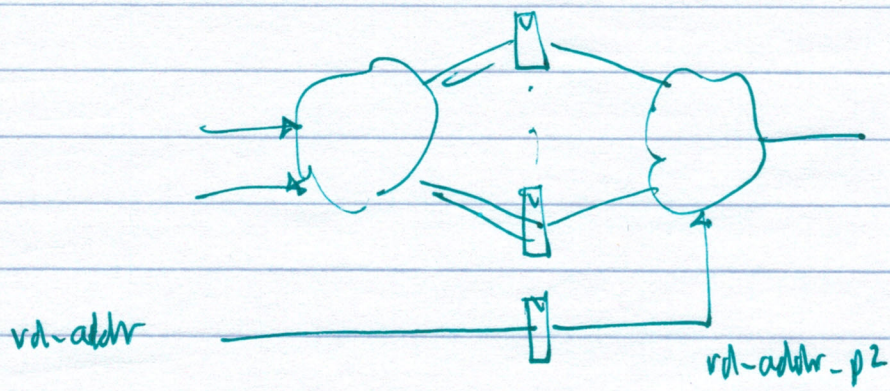


Not a valid pipeline block diagram

```
always @(posedge clk) begin
    if (wr_en ...)
        mem[wr_addr] <= #1 wr_data;
end
assign rd_data = mem[rd_addr];
```

*

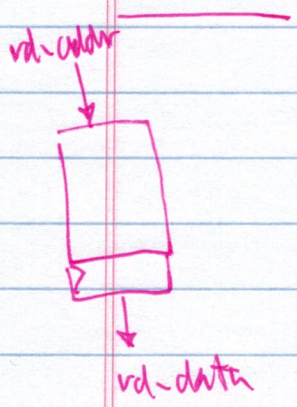


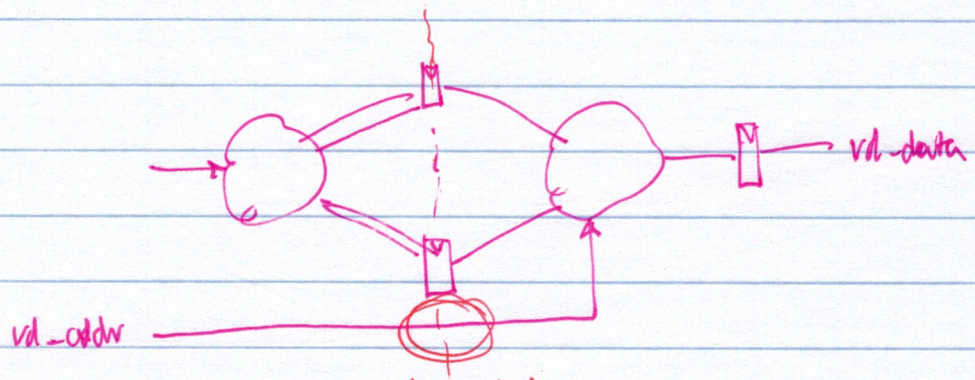



```
*
if (posedge clk) begin
    rd_addr_p2 <= #1 rd_addr;
end

assign rd_data = mem[rd_addr_p2];
```

Simultaneous rd + wr to same address → rd gets new data



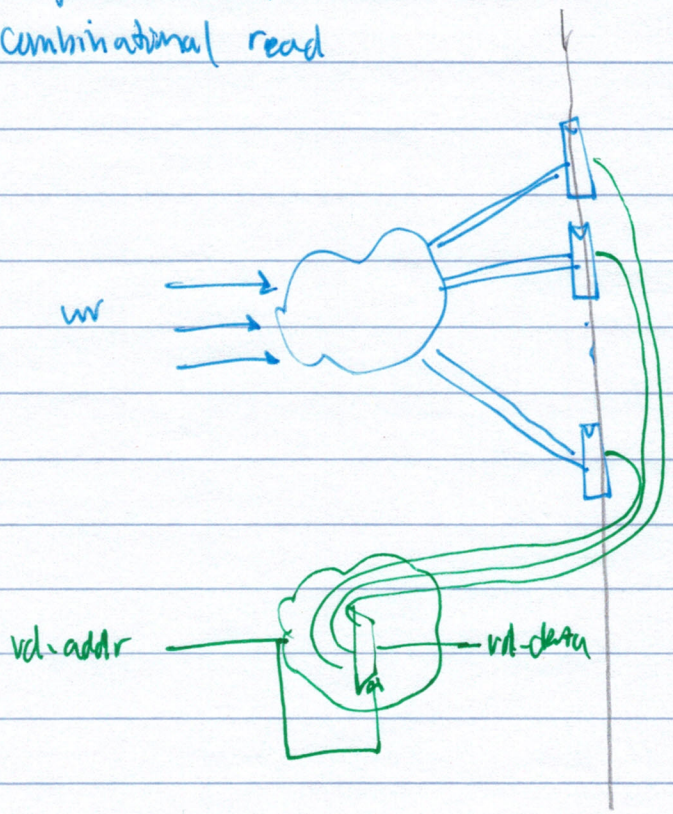


not a valid pipelined block diagram

Simult. rd/wr to same addr → rd gets old data

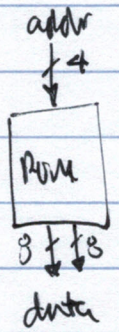
```
*
if (posedge clk) begin
    rd_data <= #1 mem[rd_addr];
end
```

Ex: "asynchronous" read
combinational read



Simult. rd/wr to same addr → rd get old data

Read-Only Memories (ROMs)



Common solution: large case statement

```
always @(addr) begin
    case (addr)
        4'b0000: begin  r = 8'b....;  i = 8'b....;
        ===
        ===
        ===
    endcase
end
```

1) Synth. from std. cells

Efficiency of solution depends on randomness of data

- Generate case statements with a program
  - matlab

2) Block RAM (RAM)
   - M9K - Altera FPGA

M9K Block RAMs on Altera FPGA chips