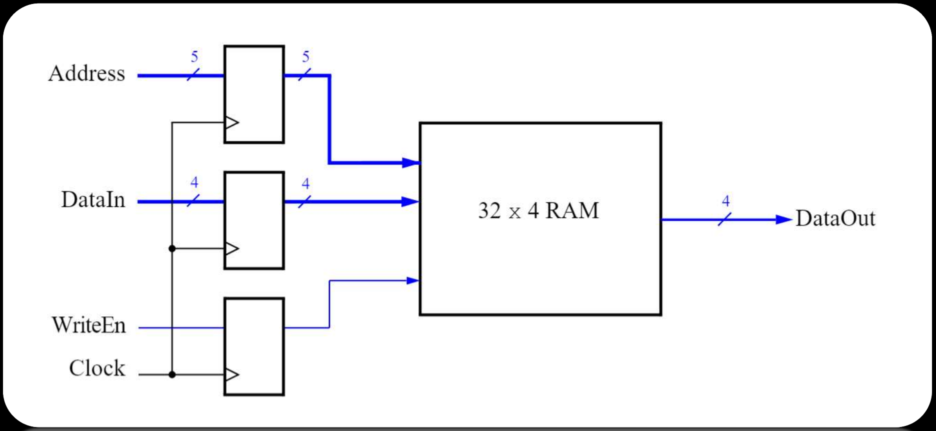
Lab 7 Preparation

Lab 7 Components

- Part I: Create a memory unit
- Part II: Interface with the RGB Video
- Part III: RGB Video animation (bonus)

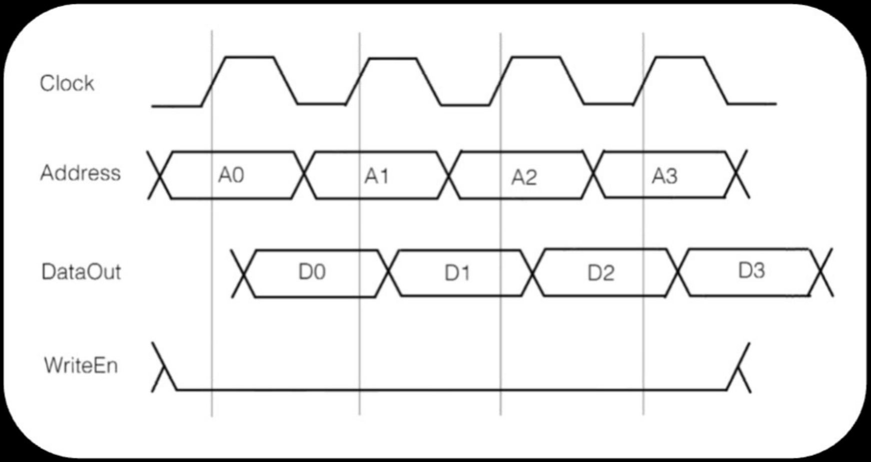
Part I: Memory Unit

- Creating a mini-RAM unit.
- Make use of the built-in RAM
  - Follow lab instructions to create a 4-bit RAM unit with 32 words.
  - Fill the RAM with values `0-F`.
- Once completed, connect this RAM to the HEX display.

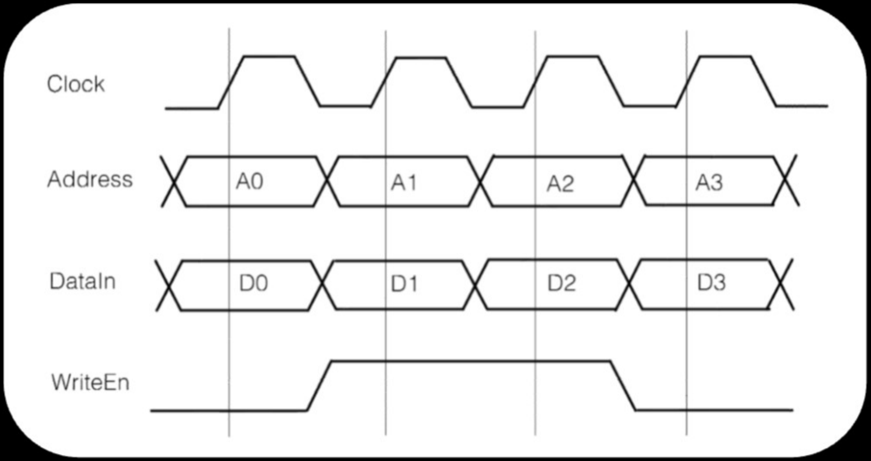


Part I: Read & Write Timing

- Read:
  - Note slight delay after clock signal, before data appears.

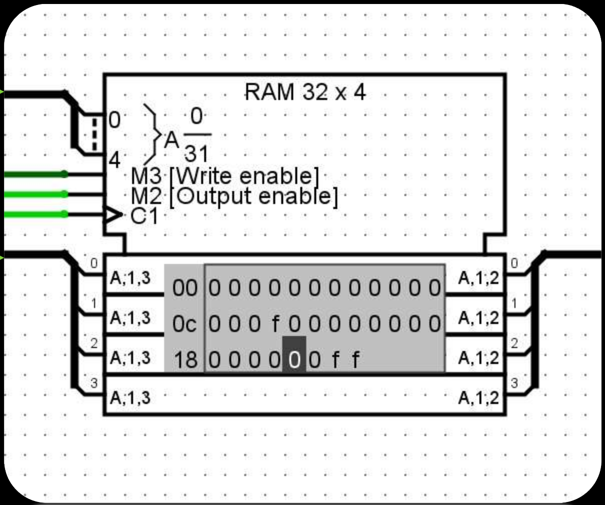


- Write:
  - Note that only `D1` and `D2` are written (because of the `WriteEn` signal).



Part I: Filling memory

- Connect address register and data register to RAM.



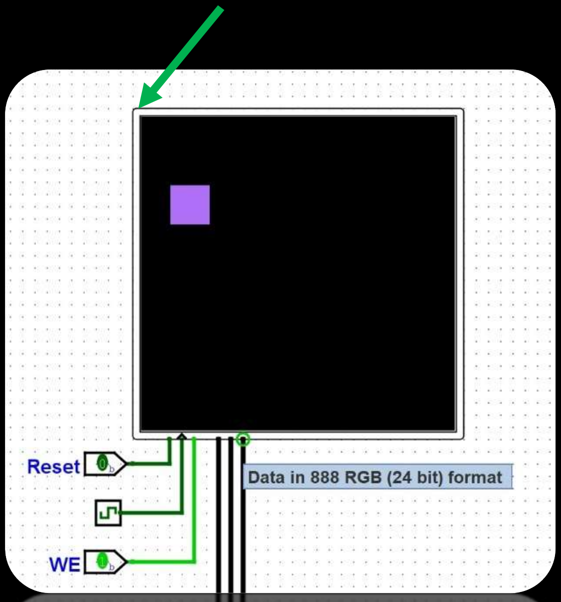
```
00 0 0 0 0 0 0 0 0 0 0 0 0
0c 0 0 0 f 0 0 0 0 0 0 0 0
18 0 0 0 0 0 0 f f
```

- Fill all RAM locations with increasing values, starting at `0` at address `00000`.

- Connect output to 7-segment display.

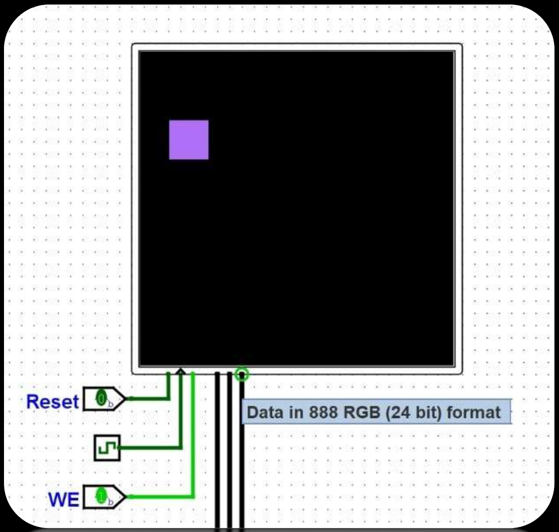
Part II: RGB Video

- The RGB Video component models the VGA display in the lab workstations.
- For this part, given input coordinates X and Y, draw a 16x16 box of coloured pixels, using X and Y as the top left corner of the box.

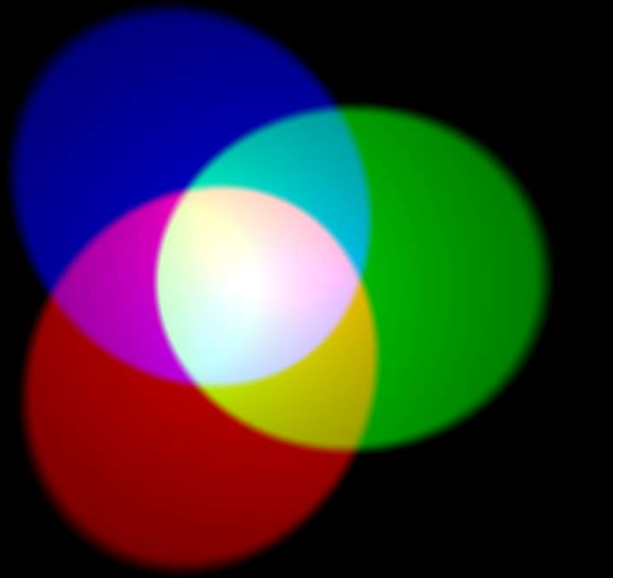


Part II: RGB Video

- The RGB Video component has 6 inputs:
  - Reset
  - Clock
  - Write Enable
    - Like lifting/dropping pen on paper.
  - X Coordinate
  - Y Coordinate
  - Data in 888 RGB (24 bit) format
    - Three sections of 8 bits, representing the RGB values for the pixel (see next slide for more detail)



Part II: RGB Video



- Light colours are additive.
  - As opposed to paint, which is subtractive.
  - Light is made of red, green and blue components.
    - White light is the combination of all three.
- To create a colour, set the 8-bit values for the red, green and blue components and concatenate them into a 24-bit value.
  - Black = `0x000000`, White = `0xFFFFFF`

Part II: RGB Video

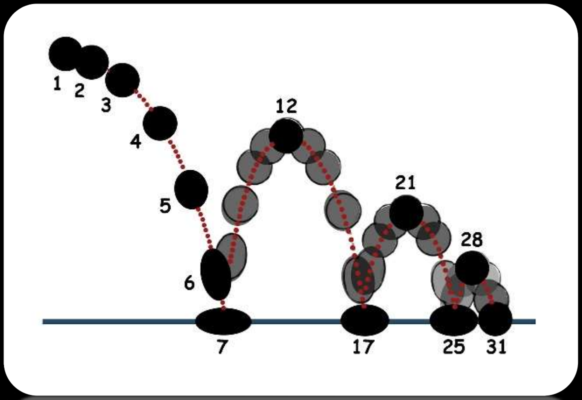
- Circuit components needed:
  - RGB Video
    - built into Logisim, check the handout for input details
  - Datapath
    - Takes in:
      - `X` and `Y` (through switches)
      - control signals (reset, clock, enable etc.)
  - FSM:
    - Controls datapath to load `X` and `Y` values, and iterate through the pixel locations that need to be updated (relative to `X` and `Y`).

Parts I & II: Hints

- Play around with the memory and RGB Video components to verify that they work.
  - E.g. Draw a single pixel on the RGB Video to start.
- The “datapath” doesn’t need to look like Lab 6.
  - Focus on the operations performed on `X` and `Y`.
- Consider using counters to store the offsets from `X` and `Y` that need to be displayed.
  - Maybe a single counter for both?
  - Counters can be part of your FSM as well.

- **It’s Lab 7**. You have full freedom to implement this however you like ☺

Part III: Animation (bonus)



- Note: This part is optional, but can be done for bonus marks in the course.
- Animate a box by drawing it, then waiting, then drawing another at a different location, then waiting…
- Many projects will use animation in some form, so you should try this part out!
  - Also…bonus marks! ☺