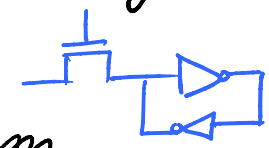
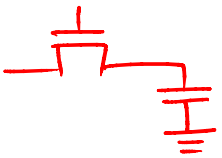
SRAM = Static Random Access Memory

↳ memory cell is 

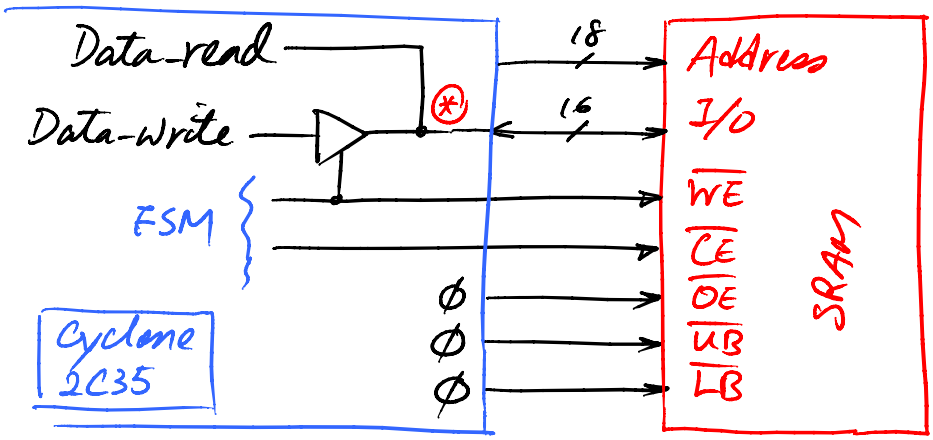
DRAM = Dynamic Random Access Memory

↳ memory cell looks like 

Connection of SRAM chip to the FPGA (on DE2 board)

→ each pin on our SRAM chip is connected to the FPGA chip (CycloneII 2C35).

→ each pin has a name like SRAM_DQ (*)
(in DE2_pin_assignment.gsf) SRAM_Address[17:0]
SRAM_WE_N, ...



Describing Tri-state ▷ in verilog

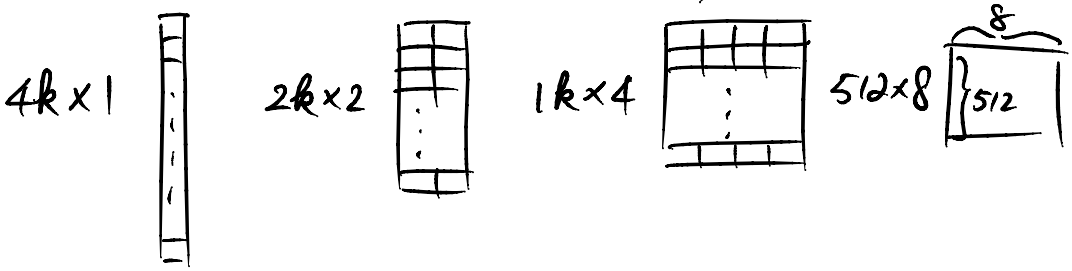
```
    :
inout [15:0] SRAM_DQ;      ← both ways
    :
assign Data_read = SRAM_DQ;
assign SRAM_DQ = (SRAM_WE_N == 0 ?) Data_write : 16'bz;
```

means tri-state (*)

Using memory inside the FPGA chip (Cyclone II 2C35)

memory blocks called m4k (4096 SRAM cells)

aspect ratio



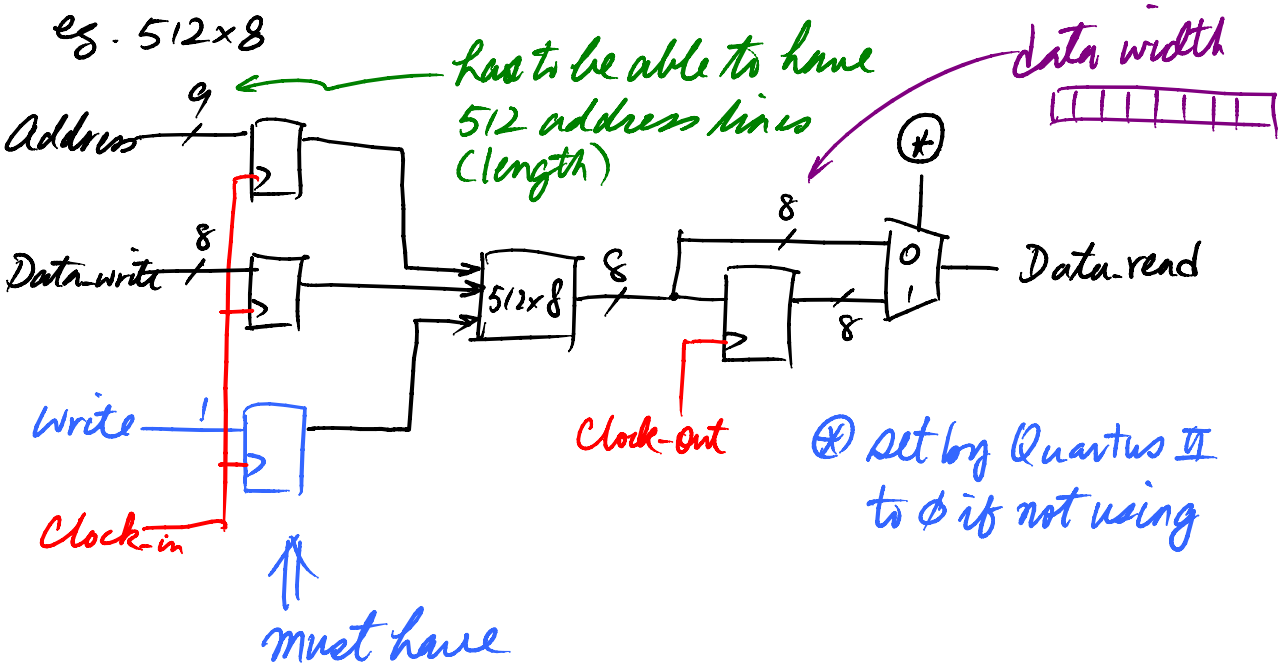
from Quartus II, use Megawizard plug-in manager.
select memory compiler

SRAM memory we have talked about IS61LV25616 ...

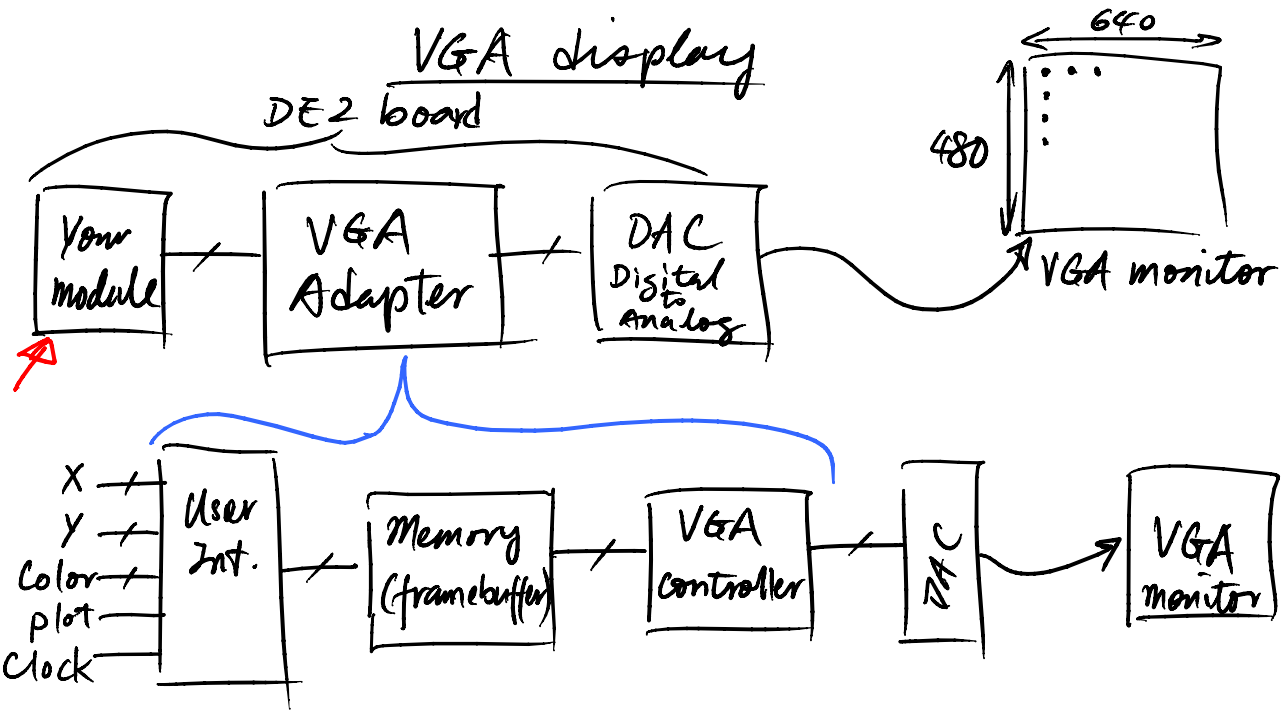
SRAM on DE2 board (2 major differences)

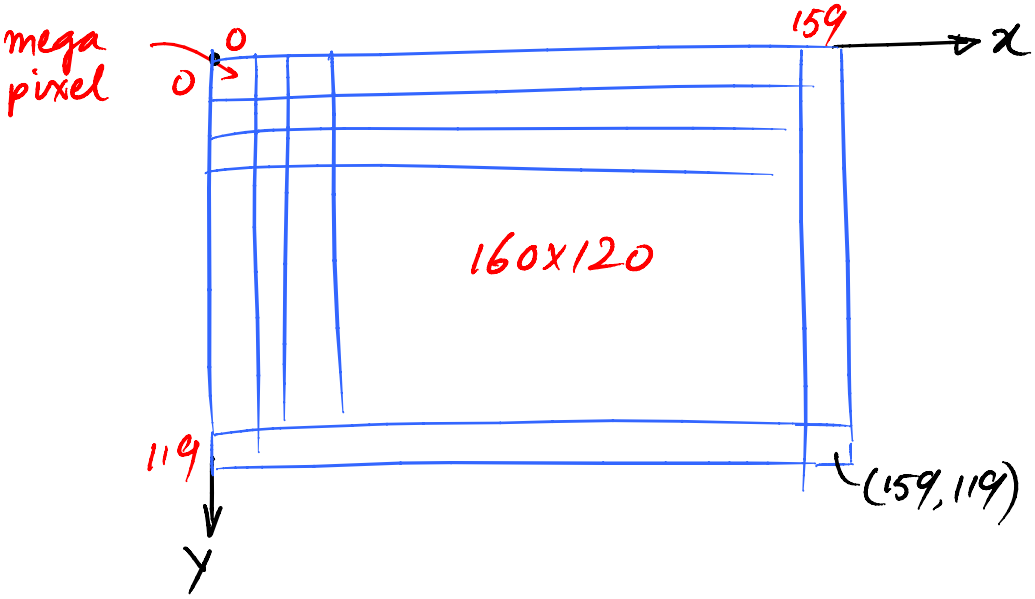
1. DataIn and DataOut are separate wires (no tri-state needed)
2. address, DataIn, and write are all stored in registers (FFs) inside the m4k block.

eg. 512x8



VGA display

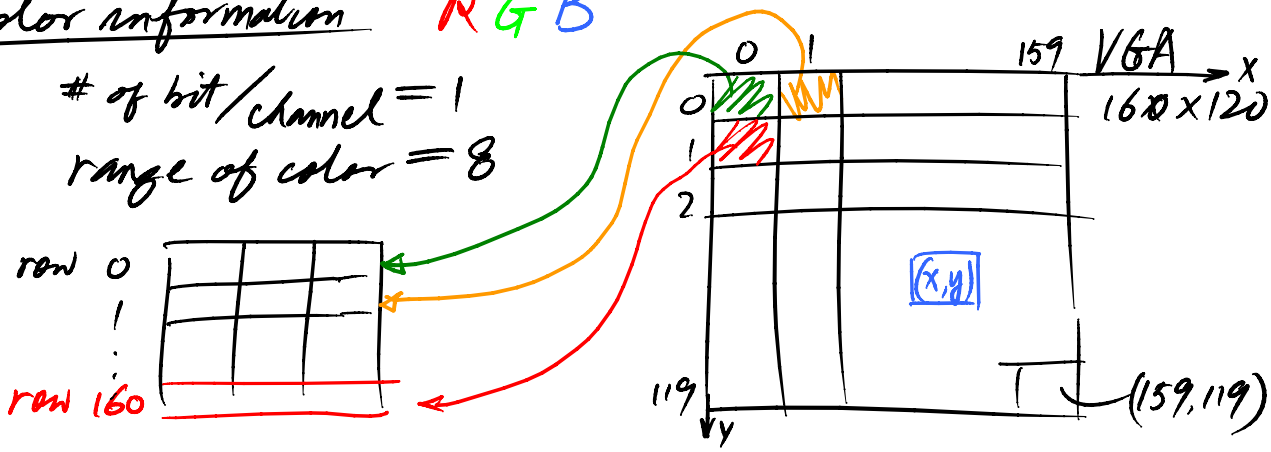




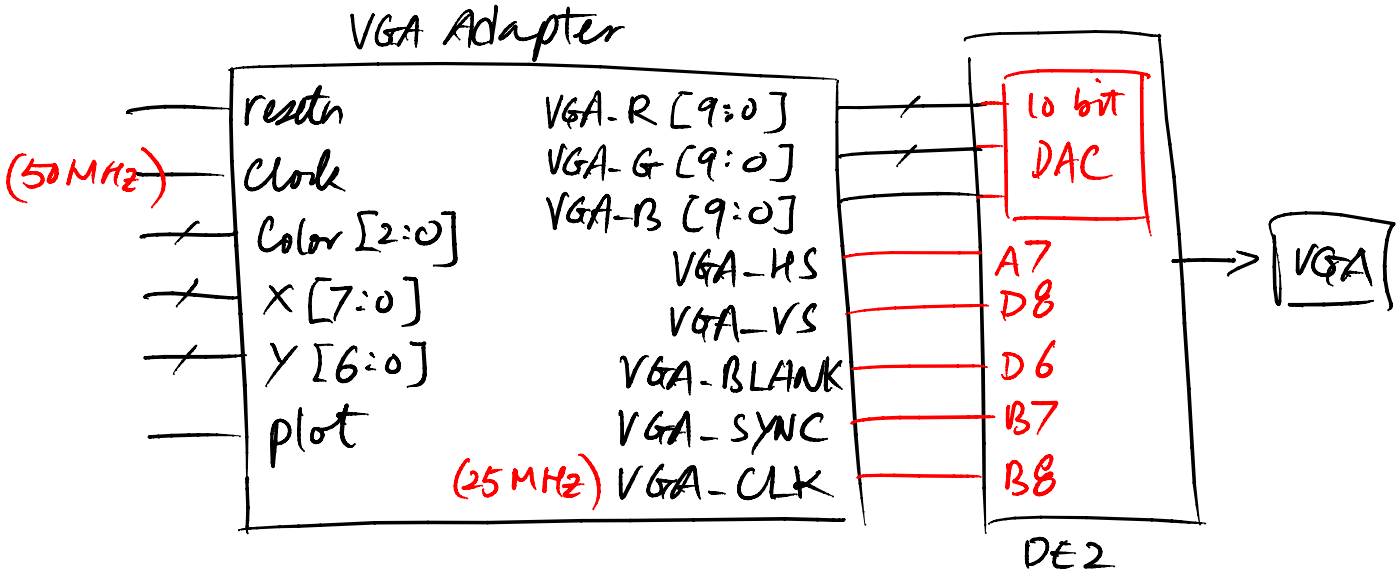
Color information R G B

of bit/channel = 1

range of color = 8



Color information at coordinate (x, y) will be stored in memory row = 160Y + X



Changing VGA Adapter Parameters

Option One

by instantiate

```
vga_adapter VGA( . . .
                 ↑ your name
:
                                        default "320x240"
defparam VGA.RESOLUTION = "160x120";
         VGA.MONOCHROME = "FALSE";
         VGA.COLOUR_CHANNEL_DEPTH = 1;
         VGA.BACKGROUND_IMAGE = "image.colour.mif";
                                default "background.mif"
```

Option two

schematic diagram

vga_adapter.bsf

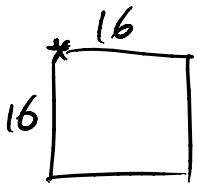
Parameter	Value
BITS_PER_COLOUR_CHANNEL	1
MONOCHROME	"FALSE"
RESOLUTION	"160x120"
BACKGROUND_IMAGE	"image.colour.mif"

Resources = bmp2mif.exe

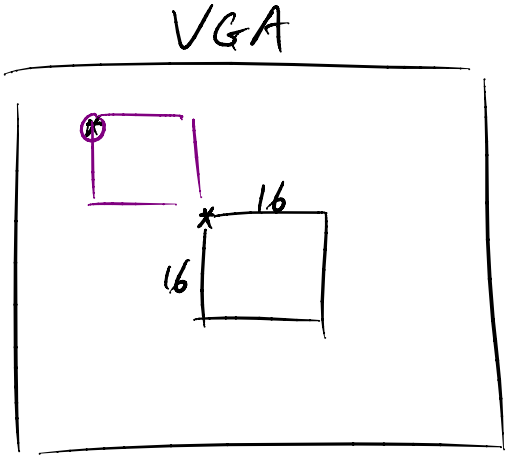
bmp2mif your bmpfile → make it 160x120

⇒ image_colour.mif
and
image_mono.mif

Lab 7. VGA



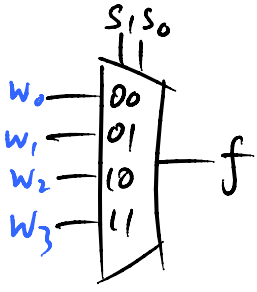
→ save in memory



Shannon's Expansion Theorem

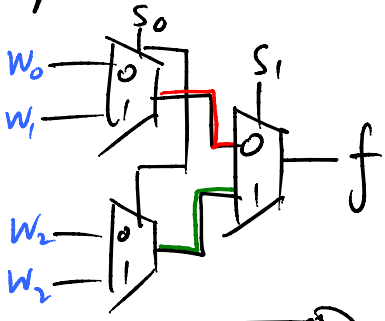
⇒ Implementing logic functions in multiplexers

4-to-1 mux



S_1	S_0	f
0	0	W_0
0	1	W_1
1	0	W_2
1	1	W_3

→ to implement this using 2-to-1 mux



→ to implement this using logic gates

$$f = \bar{S}_1 \bar{S}_0 W_0 + \bar{S}_1 S_0 W_1 + S_1 \bar{S}_0 W_2 + S_1 S_0 W_3$$