EECS150 Homework #7 Solutions:

7-1)

a) 12, 16
b) 31, 8
c) 24, 32
d) 18, 64

7-2)

a) 8K bytes
b) 2G bytes
c) 64M bytes
d) 4M bytes

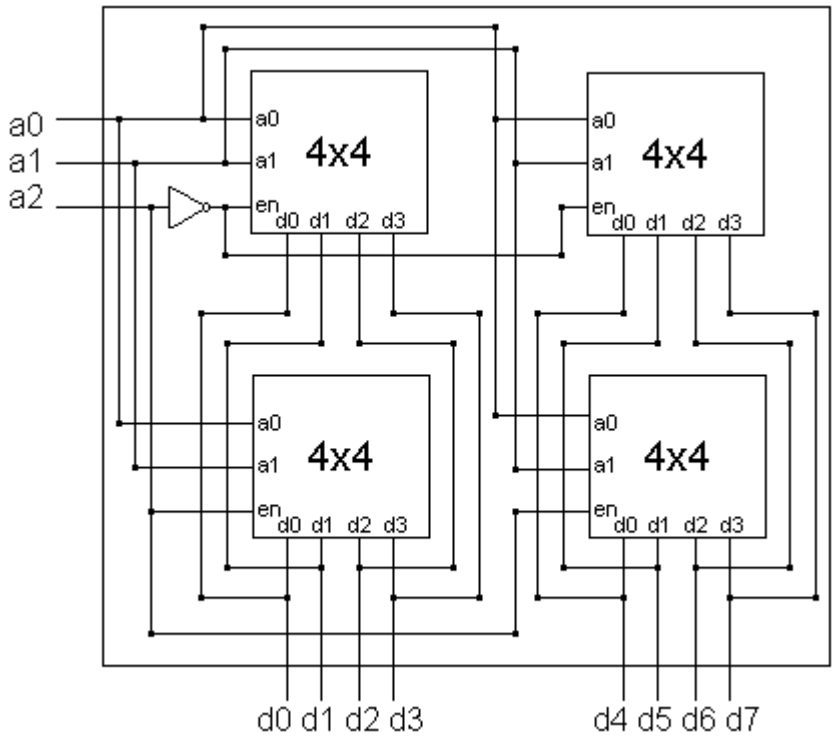
7-5)

```
module prob75
  reg en, rw;
  reg [3:0] in;
  reg [5:0] addr;
  wire [3:0] out;

  memory mem(en, rw, addr, in, out);

  initial
    begin
      addr = 6'b000000; in = 4'b0111;
      en = 1; rw = 0;
      #5 en = 0; addr = 6'b111100; in = 4'b1110;
      #5 en = 1;
      #5 en = 0; rw = 1;
      #5 en = 1;
      #5 addr = 6'b000000;
      #5 en = 1;
    end
endmodule
```

7-6)



7-7)

a) 7x128 decoders, 256 AND gates
b) X = 0101111; Y = 0100000

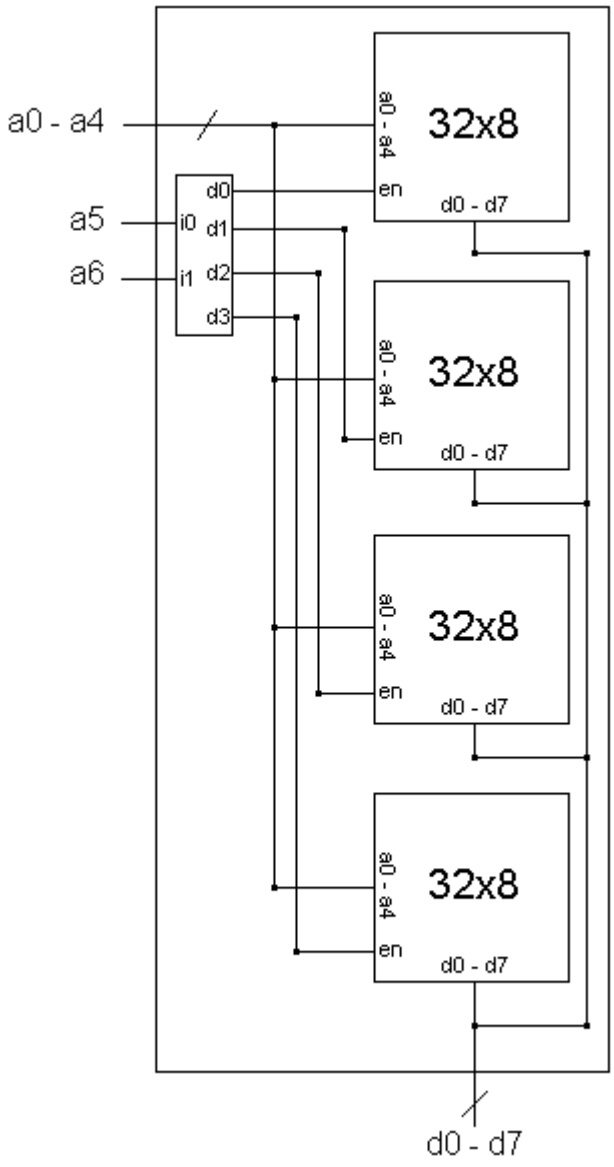
7-8)

a) 8 chips
b) 18; 15
c) 3; 3x8 decoder

7-9)

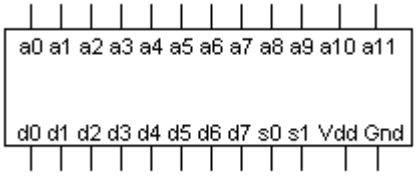
128M bits

7-15)



7-16)

24 pins



7-17)

I	D	I	D
00000	000 000	10000	011 001
00001	000 001	10001	011 010
00010	000 010	10010	011 011
00011	000 011	10011	011 100
00100	000 100	10100	100 000
00101	001 000	10101	100 001
00110	001 001	10110	100 010
00111	001 010	10111	100 011
01000	001 011	11000	100 100
01001	001 100	11001	101 000
01010	010 000	11010	101 001
01011	010 001	11011	101 010
01100	010 010	11100	101 011
01101	010 011	11101	101 100
01110	010 100	11110	110 000
01111	011 000	11111	110 001

7-18)
a) 256x8
b) 512x5
c) 1024x4
d) 32x7

7-19)

I	D (ABCD)	I	D (ABCD)
000	0100	100	1000
001	1101	101	0001
010	1011	110	1110
011	0001	111	0101

7-20)
Product Terms: yz', xz', x'y'z, xy', x'y, z

7-24)
A = yz' + xz' + x'y'z
B = x'y' + xy + yz
C = A + xyz

$D = z + x'y$

2)
1216512 bits

3)
Storing the entire video frame information, not just the visible pixels, has the advantage that the control logic for processing the frame buffer contents and sending it to the video encoder is simplified. All it needs to do is to constantly scan the contents of the frame buffer and send them to the encoder. However, such a scheme will increase the amount of memory consumed by the frame buffer and complicate the control logic for writing into the frame buffer from the network side.

4)
In this case, bandwidth refers to the number of bits/second or Bytes/second needed to represent the video steam. The 4:2:0 format uses four Y values and two C values for every 4 pixels, or 6 Bytes/4 pixels. 4:2:2 format uses four Y values and four C values for every 4 pixels, or 8 Bytes/4 pixels. The ratio is 6/8 or a savings of 25%.

5)
Information comes over the network with the even lines holding their Y values and half the C values for it (the C_R values only) and the *next* line; the next odd line comes with its own Y values and half the C values (C_B values only) for it and the *previous* line. The conversion process must combine the C values from a pair of lines, sending the combined C values (a C_B C_R pair) with a pair of Y values. One way to do this is to store all the information from two lines as it comes from the network. Then after two complete lines are received, send the information in the proper order and grouping to the frame buffer. A simple way is to sort the values into four separate buffers – one for C_B values, one for even-pixel-number Y values, one for C_R values, and one for odd-pixel-number Y values. The Y buffers will need to be deeper than the C value buffers because twice as many Y values as C values are sent per line and to permit the storage of two lines. If a dual-port memory is used for each buffer, the write side from the network can be 4-bits wide allowing nibbles to be written as they are received, and the read side going to the frame buffer, can be 8-bit wide, with all four buffers accessed at once providing 32-bits per cycle to the frame buffer.

6)
To understand the effect, assume that there is a write pointer into the frame buffer from the network side and a read pointer for the video display side. If the two rates are the same on average, the write pointer would stay ahead of the read pointer and video would be displayed in the order that it is sent. However, if the video server sends frames faster than the video is displayed, the write pointer will eventually advance past the read pointer resulting in some frames being displayed with a portion of the lines from one frame and the remainder from the next. Therefore, parts of two frames will be displayed simultaneously. Assuming that adjacent video frames are very similar, the result will be very difficult to detect. The actual explanation is somewhat more complicated because the video frames come from the network in progressive scan format and are displayed in interface format (but the basic idea stays the same).

If the result were unacceptable, control logic could be included to "slow down" the video server by occasionally discarding lines or frames. This technique, however, is still likely to result in video artifacts. The best way to solve the problem is to regenerate display frames at the proper rate based by interpolation of input frames.