Design of Digital Circuits and Systems, HW2

Memories

Solution Outlines

Problem 1:

a) **256×8** – The largest possible result is 0b1111*0b1111 = 225 = 0b11100001, which requires 8 bits. There are 2^4*2^4 = 256 possible 4-bit multiplications.

b) **128×4** – The largest possible result is 0b111+0b111 = 0b1110, which requires 4 bits. There are 7 input bits in total (two 3-bit inputs plus an add/sub selector bit), so 2^7 = 128 input combinations.

Problem 2:

a) **8 chips** – 64 MiB = 2^6*2^3 Mi-bits (*i.e.*, 8 bits per byte). Each RAM chip holds $2^2\times2^4 = 2^6$ Mi-bits so we need 2^{9-6} = 8 chips.

b) **25 bits** – 64 MiB = 2^{29} bits of memory are split across 16-bit words, so there are 2^{29-4} addresses.

c) **22 bits** – Each chip has 4 Mi-addresses = 2^{22} addresses.

d) **3:8 decoder** – We need 3 bits to specify one of eight RAM chips (from part (a)).

Problem 3:

a) Sample testbench not given.

b) Sample solution code not provided, but basic notes given below:

- The `$readmem` system task call needs to match the data in *`truthtable4.txt`* (hex).
- The widths of `addr`, `data`, and `rom` need to match the data in *`truthtable4.txt`*.

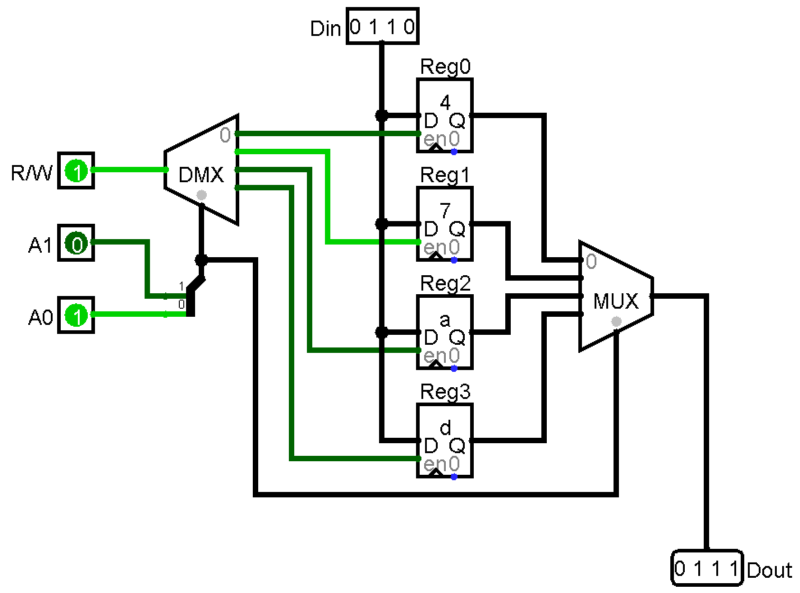
c) The two inputs to be added (`a` and `b` from `sign_mag_add`) need to be concatenated together to form the address at which you would find the corresponding sum stored in `sync_rom`.

d) The ROM-based implementation of the adder uses significantly more resources than the SystemVerilog-based implementation (module). The numbers should have looked something like: 0 ALMs and 1024 block memory bits (ROM) vs. 5 ALMs and 0 block memory bits (module).

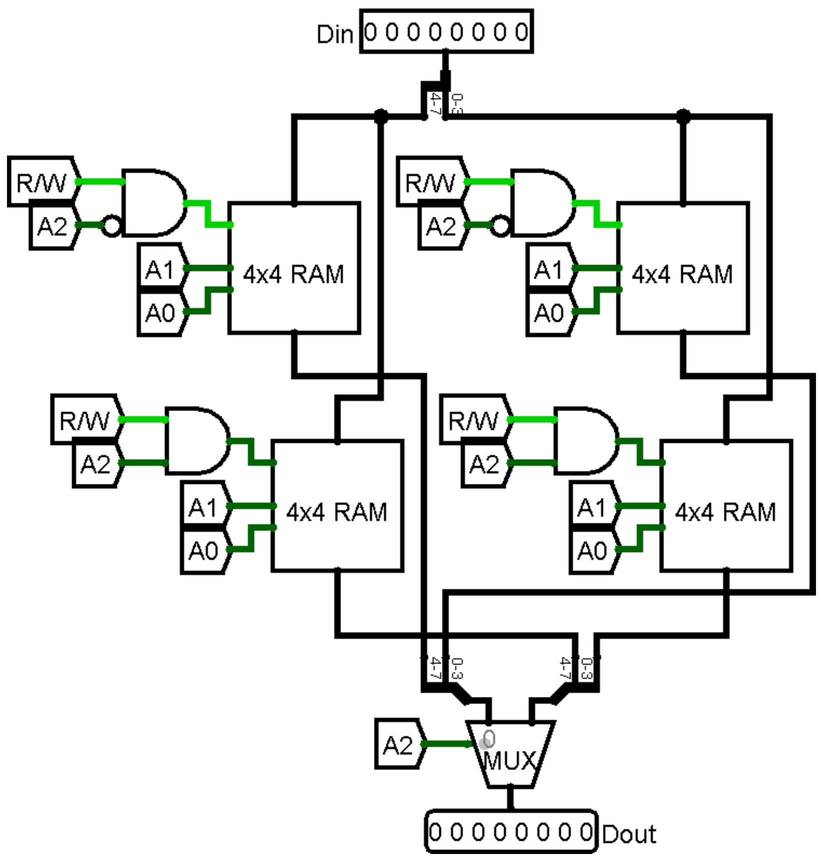
This difference gets more pronounced for larger functions. For example, for an 8-bit sign-and-magnitude adder: 25 ALMs and 524,288 block memory bits (ROM) vs. 11 ALMs and 0 block memory bits (module).

Problem 4:

a)


Din
0 1 1 0
Reg0
4
D Q
en0
Reg1
7
Reg2
a
Reg3
d
R/W
1
A1
0
A0
1
DMX
MUX
0 1 1 1
Dout

b)


Din
0 0 0 0 0 0 0 0
4-7
0-3
R/W
A2
A1
A0
4x4 RAM
MUX
0 0 0 0 0 0 0 0
Dout