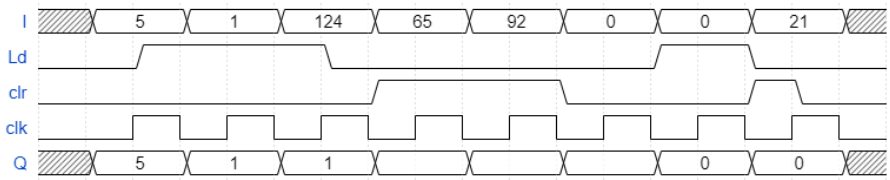
Chapter 4

Digital Design, 2nd Ed, by Frank Vahid, Wiley publication, 2010

4.2

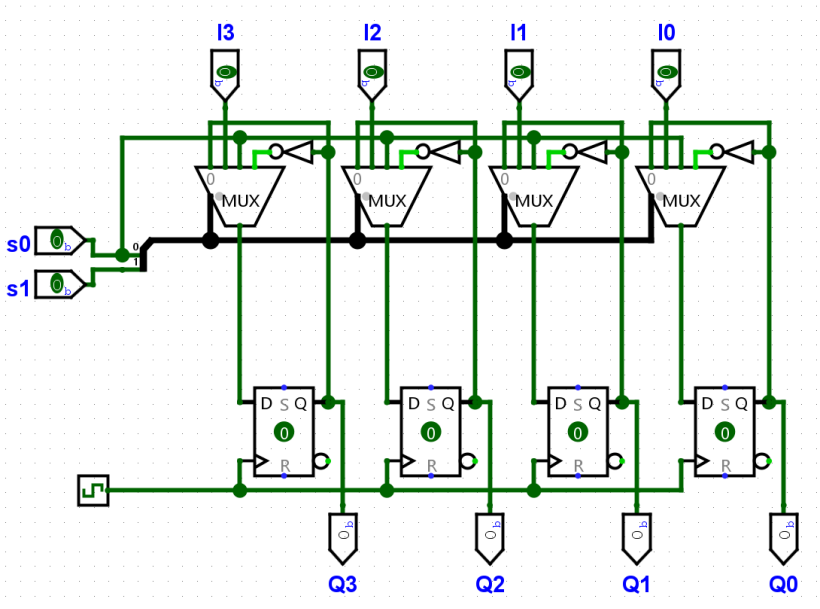


Timing diagram parallel load register

4.3

Operation table:

S_1	S_0	Operation
0	0	Maintain
0	1	Parallel load
1	0	Clear
1	1	Complement



Circuit of register

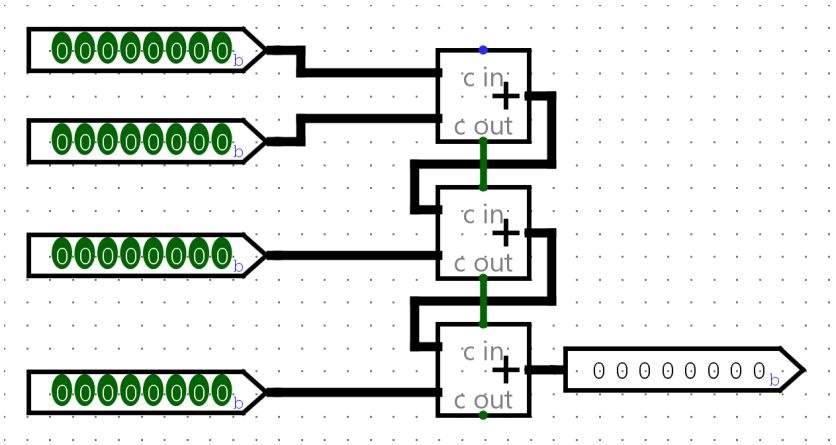
4.10

Assuming the following:

- AND gates have a propagation delay of 2ns
- OR gates have a propagation delay of 1ns
- XOR gates have a propagation delay of 3ns

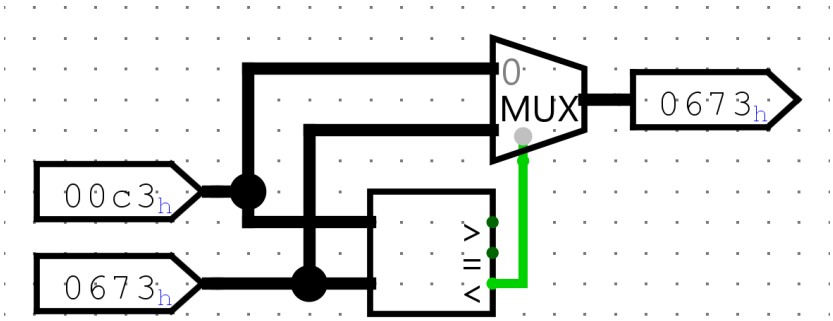
We want to find the longest propagation delay of an 8-bit ripple-carry adder. An typical 8-bit ripple-carry adder would use eight full adders. The total propagation delay would be the sum of the propagation delays of the full adders, since each successive full adder must wait for the carry-out of the previous full adder. In a typical full adder, the carry-out bit is generated by an AND gate fed into an OR gate. Thus, the longest propagation delay would be $8 \times (2 + 1) = 24ns$.

4.13



Circuit that adds 4 8-bit numbers

4.21



Circuit that outputs the greater of two 16-bit numbers

4.30

We can convert two's complement binary to decimal by looking at the most significant bit. If the most significant bit is 1, then the number is negative, so we first find the magnitude by taking the two's complement. This is done by inverting all the bits and adding 1. If the most significant bit is 0, then the number is positive, so we can convert it to decimal directly.

(a) 11100000 is negative: $00011111 + 1 = 00100000 = 32$

(b) 01111111 is positive: $2^0 + 2^1 + 2^2 + 2^3 + 2^4 + 2^5 + 2^6 = 127$

(c) 11110000 is negative: $00001111 + 1 = 00010000 = 16$

(d) 11000000 is negative: $00111111 + 1 = 01000000 = 64$

(e) 11100000 is the same as part (a)?

4.33

To convert from decimal to two's complement binary, we first convert to binary, then take the two's complement if the number is negative.

(a) $29 = 16 + 8 + 4 + 1 = 11101$

(b) $100 = 64 + 32 + 4 = 1100100$

(c) $125 = 64 + 32 + 16 + 8 + 4 + 1 = 1111101$

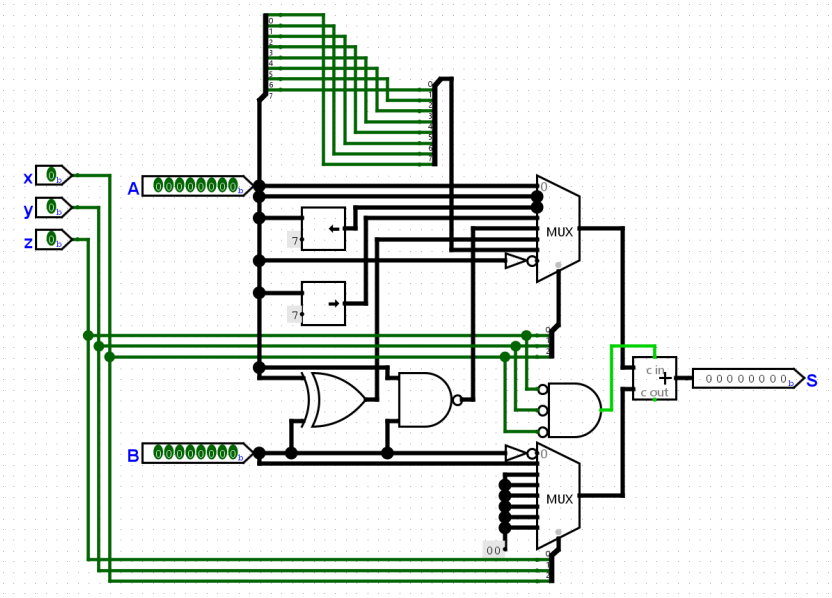
(d) $-29 = 11101 \rightarrow 00010 + 1 = 00011$

(e) $-100 = 1100100 \rightarrow 0011011 + 1 = 0011100$

(f) $-125 = 1111101 \rightarrow 0000010 + 1 = 0000011$

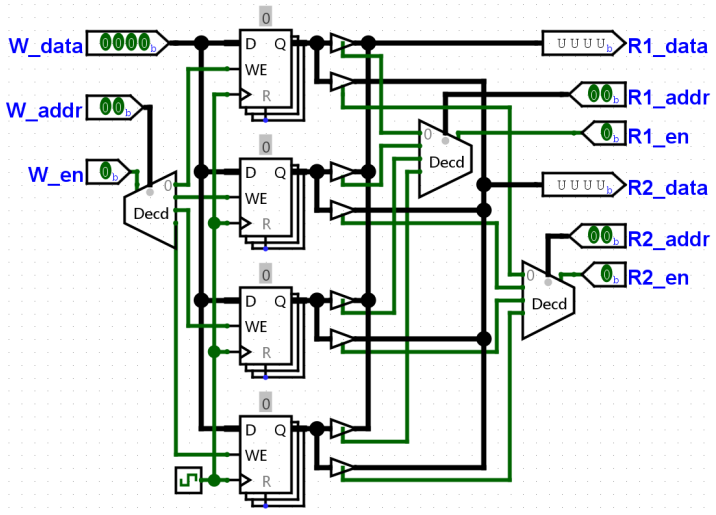
(g) $-2 = 10 \rightarrow 01 + 1 = 10$

4.40



8-bit ALU with arithmetic logic extender

4.62



Three port 2 read and 1 write register file

4.64

Given that each of a 4x4 register file's registers initially holds `0101`:

(a) If we want to simultaneously read from register 3 and write `1110` to it, we would need set the read and write enable signals to 1, set the read and write addresses to `10`, and set the write data to `1110`.

(b) Before the rising clock edge, all register should hold `0101`, and it outputs nothing. After the rising clock edge, register 3 should hold `1110`, while the other registers still hold `0101`. The output should be `0101`.