University of California at Berkeley
College of Engineering
Department of Electrical Engineering and Computer Sciences

EECS151/251A J. Wawrzynek
Spring 2023 3/9/23

Exam 1

Name: _____

Student ID number: _____

Class (EECS151 or EECS251A): _____

Before solving the problems, write your student ID number on all pages.

You have three hours to take the exam. This exam comprises a set of questions with 1 point per approximate expected minute of completion—with a total of 100 points.

For each problem if you find yourself taking excessive time to work out a solution consider skipping the problem or a fresh approach. Also, start by answering the easier questions and then move on to the more difficult ones.

No calculators, phones, or other devices allowed.

Neatness counts. We will deduct points if we need to work hard to understand your answer.

1 Dennard Scaling [3 pts]

With Dennard scaling we scale dimensions, and Vdd by a factor of $1/k$ ($k > 1$), and doping concentrations by k. For a given design and layout (say a microprocessor, for instance), what would be the effect of Dennard scaling on area, maximum clock frequency, and power, in terms of k?

Solution:

Cost: $1/k^2$, Maximum clock frequency: k, Power: $1/k^2$.

2 Pareto Optimality [2 pts]

Your project partner gives you two designs for a 32-bit adder circuit and claims that they are Pareto Optimal with respect to cost/performance:

- Circuit A comprises 384 logic gates and has delay of 32 ns.
- Circuit B comprises 636 logic gates and has delay of 8 ns.

For this problem assume gate count is equvalent to cost. You suspect that they are not optimal among all possible designs. How would you attempt to prove that?

Solution:

For each of them, find a design that uses a fewer number of gates with the same delay, or a design that has a smaller delay with the same number of gates.
(Trying to find a one that uses less than 384 gates and also has less than 8 ns is not appropriate, since it is unlikely to be possible.)

3 Costs [4 pts]

Suppose we find a source for *square* wafers to use in our factory (impossible, but makes the arithmetic easier). These wafers are 20 cm on a side. Manufacturing costs per wafer are \$6K. Our die layout is 2 cm × 3 cm. The manufacturing die yield is 50% for 6 cm^2. Calculate the per die cost.

Solution:

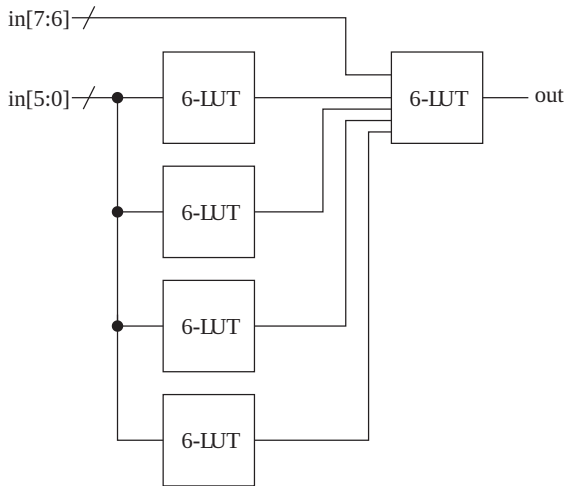
We can place $10 \times 6 = 60$ dies on a wafer, while 20×2 cm^2 is wasted. Among those 60 dies, 30 dies (50%) are defect-free. By dividing \$6K by 30, \$200 is the cost per die. *Update (3/16): you may rotate dies to put more than 60 dies on a wafer.*

4 FPGAs [5 pts]

You are given a set of 6-input LUTs and asked to use them (with no other components or logic gates) to implement a 8-input LUT. Show how this can be done using the least number of 6-input LUTs as possible. How many 6-input LUTs does it require?

Solution:

First of all, an 8-input LUT has $2^8 = 256$ configuration bits. Since each 6-input LUT has $2^6 = 64$ configuration bits, we need four 6-input LUTs to store all of them. Each of those four 6-input LUTs takes the first six inputs and selects one bit to output. Then, we can implement a 4-to-1 multiplexer using a 6-input LUT, which selects one of those four outputs depending on the remaining two inputs. This design uses five 6-input LUTs.



5 Boolean Algebra [5 pts]

Given the Boolean function $f = a'c + abc$, *algebraically* derive f' in a reduced (minimal) sum-of-products form. Show your work.

Solution:

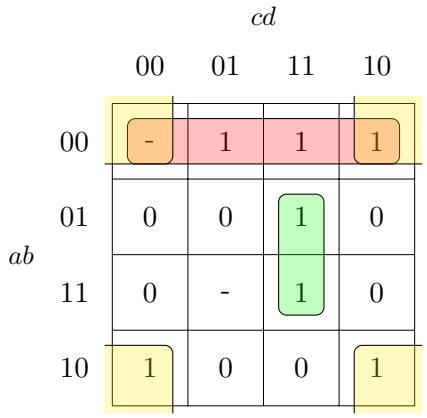
$$\begin{aligned} f' &= (a'c + abc)' \\ &= (a'c)'(abc)' \\ &= (a + c')(a' + b' + c') \\ &= aa' + ab' + ac' + c'a' + c'b' + c'c' \\ &= 0 + ab' + ac' + c'a' + c'b' + c' \\ &= ab' + c'(a + a' + b' + 1) \\ &= ab' + c' \end{aligned}$$

6 Don't-Cares [5 pts]

Given the reduced Boolean function $f = a'b'd + bcd + b'cd' + ab'd'$ representing the output of circuit. Now assume that the following input combinations are invalid—i.e., they will never appear in actual use of the circuit: $\{a, b, c, d\} = 0000$ and $\{a, b, c, d\} = 1101$. Can you further reduce the function for the remaining valid input patterns? Show your work.

Solution:

Karnaugh map:

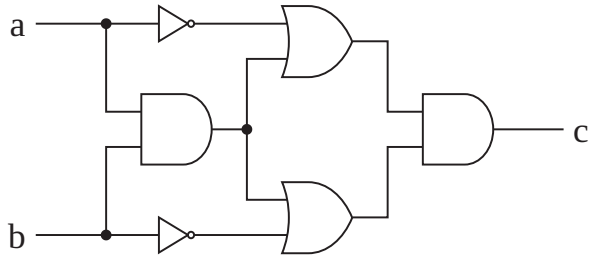


Boolean expression:

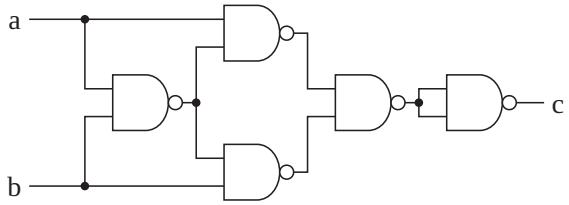
$$f = a'b' + b'd' + bcd$$

7 NAND Networks [4 pts]

Implement the circuit shown below using *only* 2-input NAND gates (no inverters or other gates).



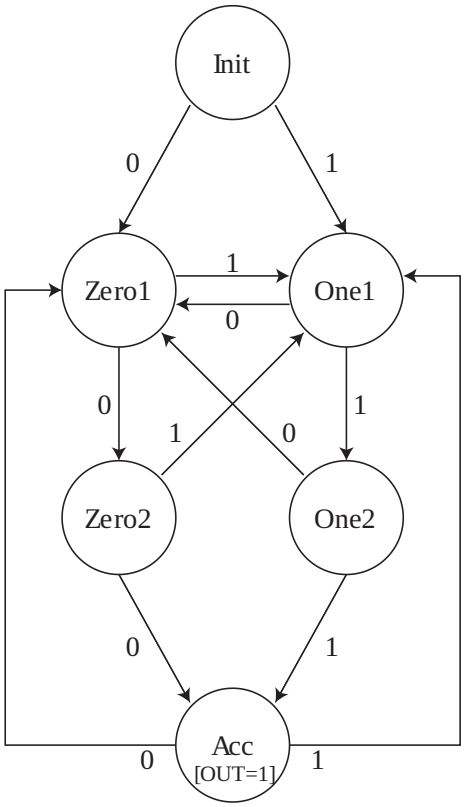
Solution:



8 State Transition Diagrams [5 pts]

In the space below, neatly draw the state transition diagram for a Moore finite state machine with the following specification. The FSM has a single input, IN, and a single output, OUT. After reset it outputs a 0. When it sees a consecutive sequence of three 1's or three 0's at its input, it outputs a 1, then starts looking again. (For example, for the input sequence 0111_1111_0001, it outputs a sequence 0000_1001_0001, assuming that it resets at the beginning and outputs a 0 in the first cycle.)

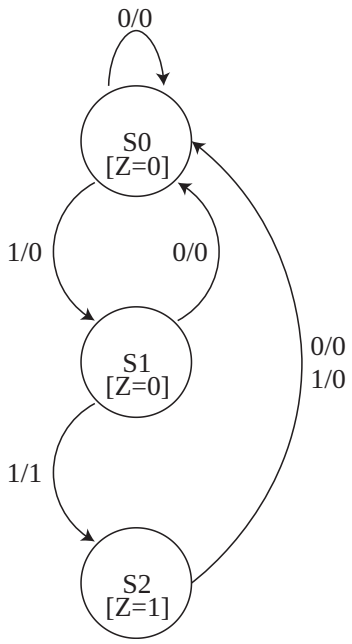
Solution:



Labels on the edges represent the value of IN, and OUT is 0 unless specified.

9 Verilog

Consider the following state transition diagram. It takes an input X, and has a Mealy style output (Y) and a Moore style output (Z). Labels on the edges represent X/Y. Neatly write out the behavioral Verilog specification for this machine with binary encoded states and one-hot encoded states on the following pages. *Do not change the provided templates.*



9.1 Verilog with Binary Encoded States [6 pts]

```
module fsm_binary(input X, output reg Y, output reg Z, input rst, input clk);
  localparam S0 = 2'b00;
  localparam S1 = 2'b01;
  localparam S2 = 2'b10;

  wire [1:0] ps;
  reg  [1:0] ns;
  REGISTER_R #(.N(2), .INIT(S0))
  state_reg (.q(ps), .d(ns), .rst(rst), .clk(clk));

  // write your answer below
```

Solution:

```
always @(*) begin
  ns = ps;
  Y = 0;
  Z = 0;
  case(ps)
  S0: if(X) ns = S1;
  S1: if(X) begin
        ns = S2;
        Y = 1;
      end
      else ns = S0;
  S2: begin
        ns = S0;
        Z = 1;
    end
  endcase
end
```

```
endmodule
```

9.2 Verilog with One-Hot Encoded States [6 pts]

```
module fsm_onehot(input X, output Y, output Z, input rst, input clk);
  localparam S0 = 3'b001;
  localparam S1 = 3'b010;
  localparam S2 = 3'b100;

  wire [2:0] ps;
  wire [2:0] ns;
  REGISTER_R #(.N(3), .INIT(S0))
  state_reg (.q(ps), .d(ns), .rst(rst), .clk(clk));

  // write your answer below
```

Solution:

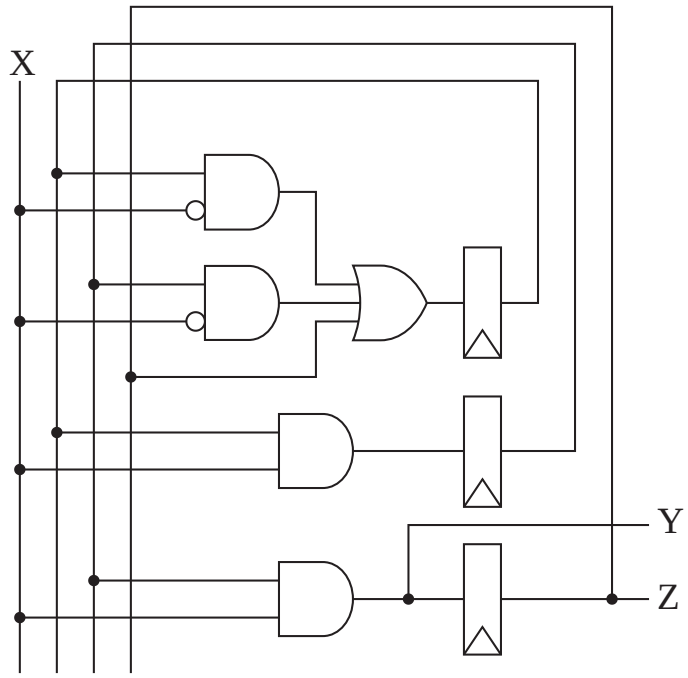
```
assign Y = ns[2];
assign Z = ps[2];
assign next_state[0] = (ps[0] & ~X) |
                       (ps[1] & ~X) |
                        ps[2];
assign ns[1] = ps[0] & X;
assign ns[2] = ps[1] & X;
```

```
endmodule
```

10 Implementation of Finite State Machine [6 pts]

Based on your answer in the previous question, draw a gate-level circuit diagram for a one-hot encoded implementation for this machine, using simple logic gates and flip-flops.

Solution:



11 Counters [8 pts]

Consider a special 4-bit counter with 4 outputs, *a*-*d*, that has the following output sequence (note that 0000 never appears). This counter is an instance of a "Galois counter", known for generating pseudo-random sequences with a very small number of logic gates.

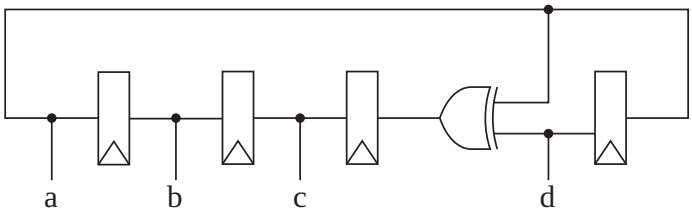
$$\{a, b, c, d\} : 0001 \rightarrow 0010 \rightarrow 0100 \rightarrow 1000 \rightarrow 0011 \rightarrow 0110 \rightarrow 1100 \rightarrow 1011 \rightarrow 0101$$
$$\rightarrow 1010 \rightarrow 0111 \rightarrow 1110 \rightarrow 1111 \rightarrow 1101 \rightarrow 1001 \rightarrow 0001$$

Draw a circuit diagram for this counter with 4 FFs (minimize the number of gates).

Hint: you may use any method you like to solve this, however, we suggest you begin by writing out the truth-table representing the next state *function, then carefully examine it. That might lead to some shortcuts.*

Solution:

This is an LFSR where c_{next} is an XOR of a and d.



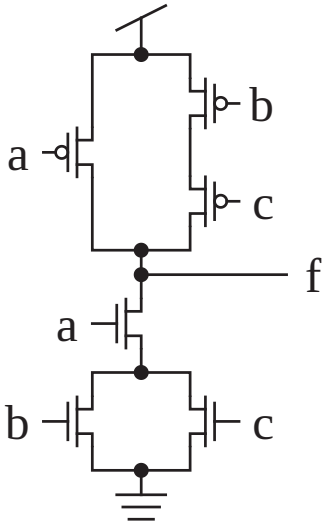
12 CMOS Design [4 pts]

For the function defined in the following truth-table, design a static CMOS gate (always pulls its output to Vdd or GND). Better solutions have fewer transistors.

a	b	c	f
0	0	0	1
0	0	1	1
0	1	0	1
0	1	1	1
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	0

Solution:

$$f' = ab'c + abc' + abc = a(b'c + bc' + bc) = a(b'c + b) = a(b + c)$$

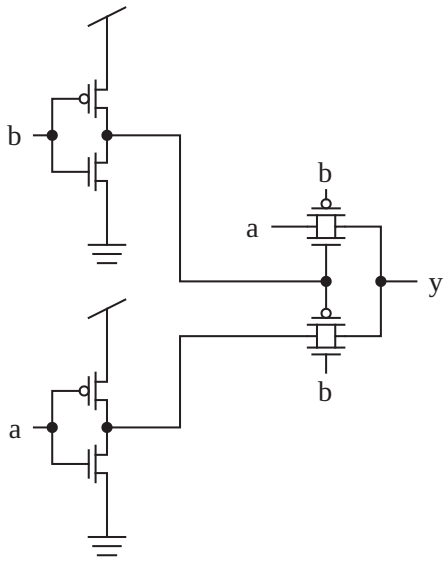


13 XOR Gate Implementation [5 pts]

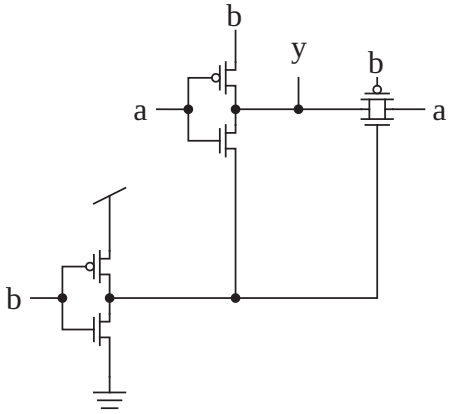
Draw a transistor-level circuit diagram for an $y = xor(a, b)$. You cannot assume you have complemented inputs. Better solutions have fewer transistors. *Hint*: Think in terms of transmission gate circuits. Your solution need not be a *static* gate.

Solution:

Good (full credit):

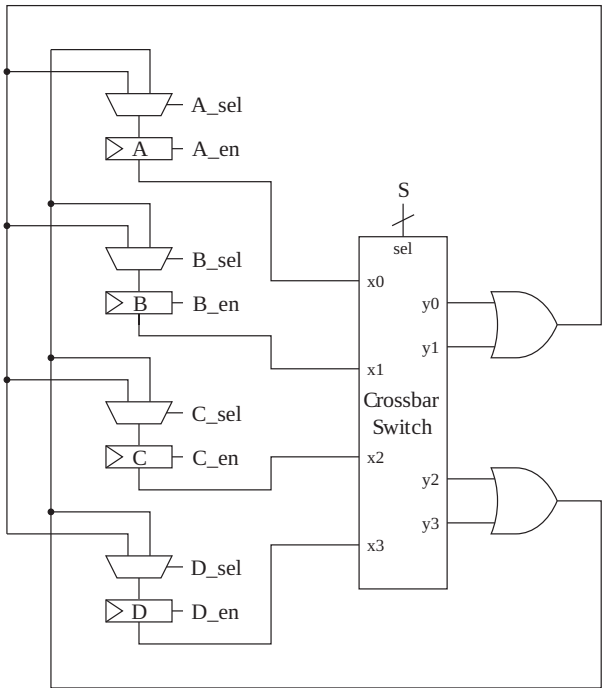


Minimum:



14 Register Transfers [4 pts]

Given the block diagram shown below, where the registers are initialized with the values A, B, C, and D as shown, what is the minimum number of cycles needed to achieve a state where the register values (from top to bottom) are A, $A+B$, $A+B+C$, and $A+B+C+D$. Explain what happens on each clock cycle. Note that a *crossbar switch* is a combinational logic circuit that can be configured with the select signal to perform any permutation of its inputs. You can change the values of the select signals and enables arbitrarily each cycle (but you don't need to tell us their values).



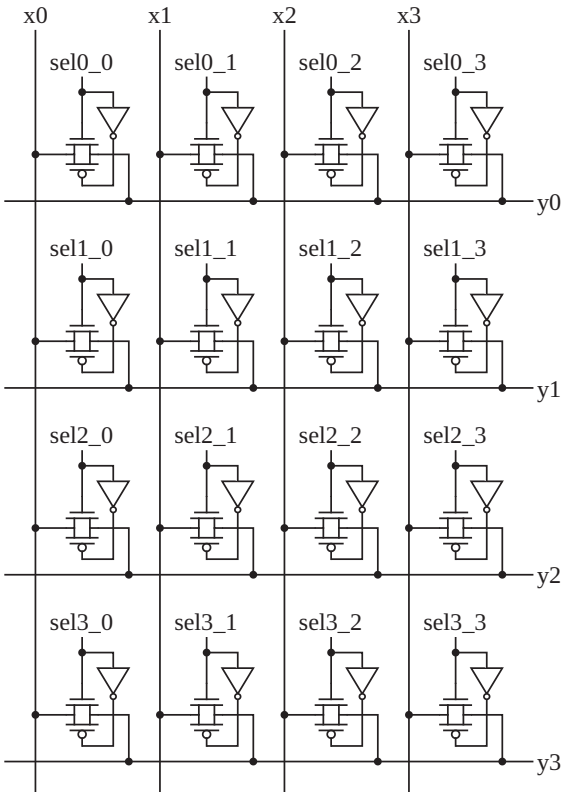
Solution:

2 cycles.
First cycle: Store $A+B$ in the second register overwriting B, and $C+D$ in the last register overwriting D.
Second cycle: Store $(A+B)+C$ in the third register overwriting C, and $(A+B)+(C+D)$ in the last register overwriting $C+D$.

15 Switch Networks [5 pts]

For the 4×4 crossbar switch used in the previous question, describe a scheme to implement it using transmissions gates. It takes a 4-bit one-hot select signal for each output (16 bits in total). The data paths from input to output can only use transmission gates, but, if needed, you can use simple logic gates for the control paths. Try to use the least number of transmission gates as possible. You don't need to draw the entire circuit, particularly for parts that are repeated.

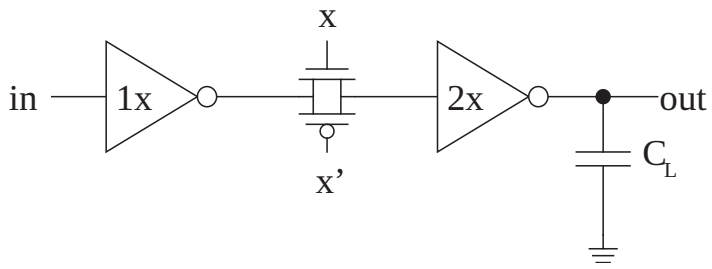
Solution:



16 Circuit Delay [5 pts]

Consider the circuit shown below with a transmission gate in series with a couple inverters. The right most inverter is sized 2X and drives a capacitive load represented by C_L. We represent the drive resistance of a unit sized inverter as R_N and its input capacitance as $3C_N$ (we assume the inverter is sized to have balanced pullup and pulldown strength). The transistors in the transmission gate have the same sizes as the ones in the unit sized inverter. Furthermore, assume that $\gamma = 1$. Ignore wire resistance and capacitance.

Write an expression for the total delay from input to output of this circuit, in terms of R_N, C_N, and C_L.



Solution:

The parasitic capacitance of 1x inverter is $3C_N$ as $\gamma = 1$. There is $3C_N$ capacitance on each side of the transimission gate. The transmission gate has resistance R_N ($R_N/2$ is also okay). The input capacitance of 2x inverter is $6C_N$. So, the delay of the first inverter is

$$\ln 2 \cdot (R_N(3C_N + 3C_N + 3C_N + 6C_N) + R_N(3C_N + 6C_N)) = 24 \ln 2 \cdot R_N C_N \tag{1}$$

The paracitic capacitance of 2x inverter is $6C_N$. The delay of the second inverter is

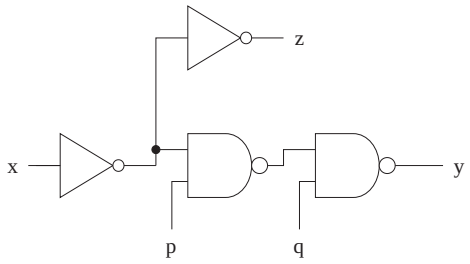
$$\ln 2 \cdot \frac{R_N}{2}(6C_N + C_L) = 3 \ln 2 \cdot R_N C_N + \ln 2 \cdot \frac{R_N C_L}{2} \tag{2}$$

Therefore, the total delay is

$$27 \ln 2 \cdot R_N C_N + \ln 2 \cdot \frac{R_N C_L}{2} \tag{3}$$

17 Propagation Delay [4 pts]

Consider the circuit shown below comprising *unit sized* inverters and NAND gates. Each 2-input NAND gate has the same input capacitance as the inverter. We know that the propagation delay for an inverter can be expressed as $t_p = t_{p0}(1 + f/\gamma)$, and for a 2-input NAND gate as $t_{p0}(2+4f/3\gamma)$, where t_{p0} is the intrinsic delay of an inverter. Derive an expression for the delay from input x to output y in terms of t_{p0} and f, where f is the fanout of the last NAND gate. *Do not resize the gates.* For this problem assume that $\gamma = 1$.



Solution:

The inverter has fanout 2, the first NAND gate has fanout 1, and the last NAND gate has fanout f.

$$t_{p0}(1+2) + t_{p0}(2+\frac{4}{3}) + t_{p0}(2+\frac{4}{3}f) = t_{p0}(\frac{25}{3}+\frac{4}{3}f)$$

18 Layouts [2 pts]

CMOS standard cell libraries are often designed with the layout of the transistors being wider than the minimum allowed by the process design rules. Explain why this is done (non-minimal sized transistors used).

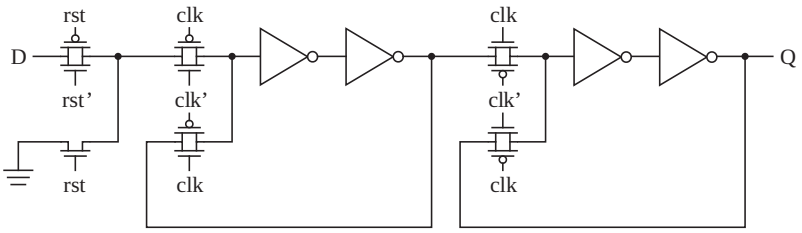
Solution:

The wire delay may be reduced by doing so.

19 Resetting Flip-Flop [5 pts]

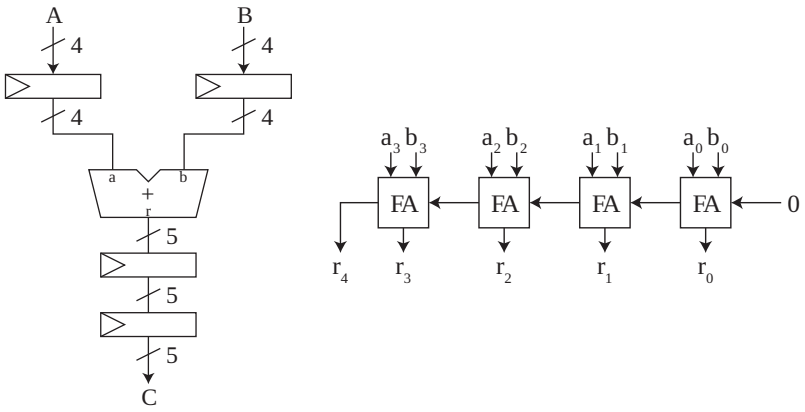
Draw the circuit diagram for a positive-edge-trigger d-flip-flop with a synchronous reset input (it resets to 0). Better solutions have fewer extra transistors.

Solution:



20 Multi-Stage Circuits

Consider the circuit shown below with a 4-bit ripple adder with registered inputs and output. Also shown a detailed circuit of the ripple adder. For FAs, the delay is 20 ps from any input to any output. For FFs, the clock to q delay is 10 ps, the setup time is 10 ps, and the hold time is 10 ps.



20.1 Clock Frequency [2 pts]

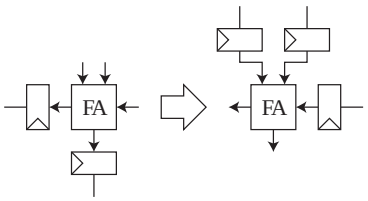
What is the maximum clock frequency for this circuit?

Solution:

Four FAs in a series, causing 80 ps delay. Thus, $f_{max} = \frac{1}{(10+10+80)\ \text{ps}} = 10$ GHz

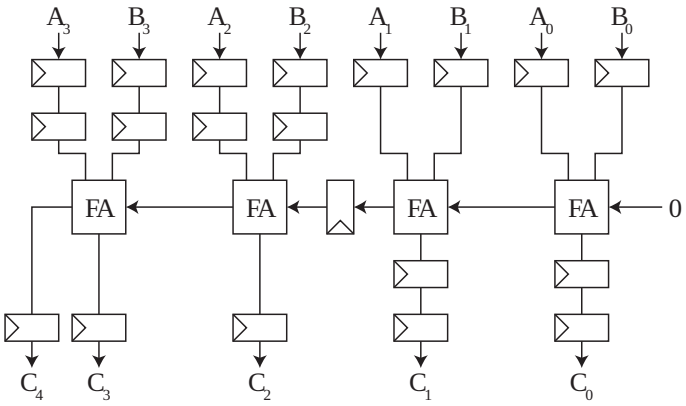
20.2 Retiming [5 pts]

Retime and redraw the circuit to improve the maximum clock frequency. What is the improved frequency? *Hint*: You can separate an N-bit FF into N 1-bit FFs. An example of valid retiming over the FA is shown below.



Solution:

Two FAs in a series, causing 40 ps delay. Thus, $f_{max} = \frac{1}{(10+10+40)\ \text{ps}} \approx 16.6$ GHz



Build timestamp: Friday 17th March, 2023 11:50