EECS 151/251A Homework 5

Due Monday, Feb 26th, 2024

Introduction

This homework is meant to test your understanding of the basic principles used to construct finite state machines. If asked to submit a Verilog module, please show the full module. If asked to simulate a Verilog module, create a testbench to run your modules in a simulator. We either recommend the following free, online Verilog simulator: `https://www.edaplayground.com`.

Important: Use the register library in `EECS151.v` when sequential logic is needed for all of the HW problems.

Problem 1: Circuit to State Transition Diagram

Given the following circuit:

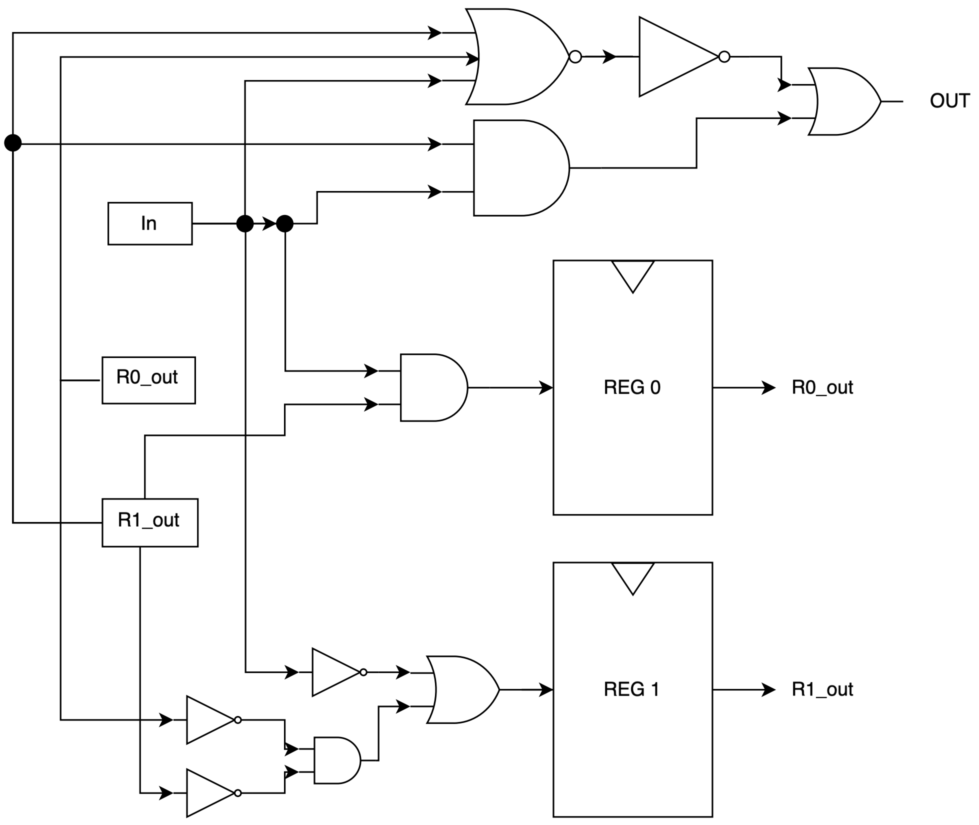
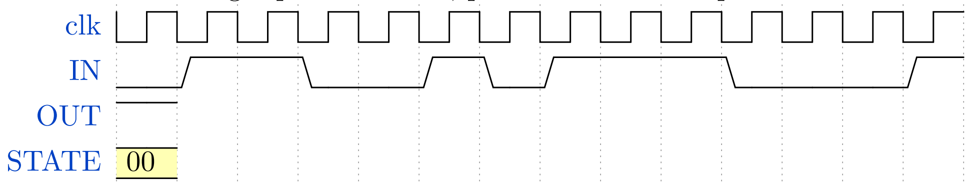


Figure 1: FSM Circuit

1. Describe if the circuit above describes a Mealy or Moore state machine? Give justification for your response.

2. Please draw the state transition diagram that corresponds to this circuit.

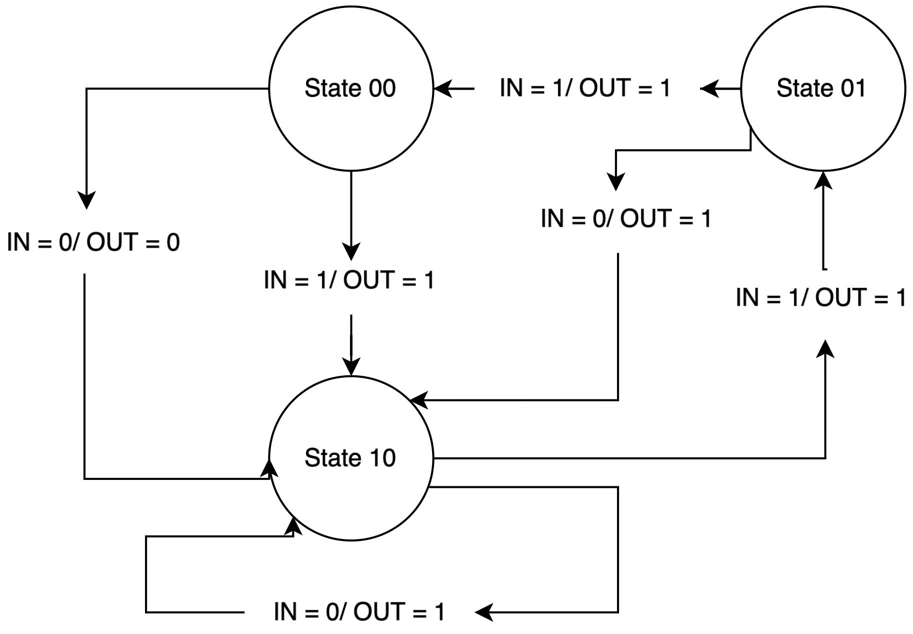
3. Given the following input waveform, please draw the output waveform:



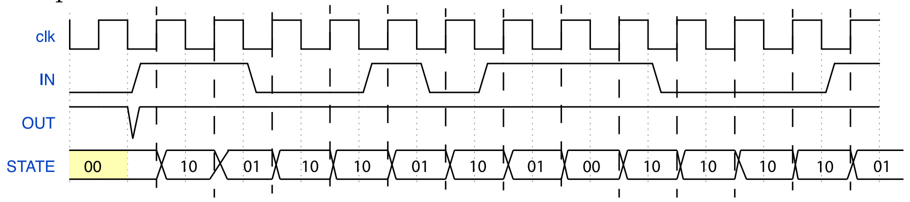
Solution:

1. The machine is a Mealy Machine. This is because the output depends on the input as well as the state, and not just the state.

2. The STD is given as the following:



3. The output waveform is:



Problem 2: FSM Verilog

You are given a sequence of 2-bit numbers, one number at a time. Your task is to design a state machine that outputs a 1 when the current sum of all of the numbers is divisible by 3.

For example for the sequence 3,3,0,2,1,1 (receiving from the left first) we will have:

Input	**Total Sum**	**Output of Machine**
3	3	1
3	6	1
0	6	1
2	8	0
1	9	1
1	10	0

Hint: You SHOULD NOT use an adder or some similar combinational logic in your solution. This is able to be solved using a standard FSM with combinational logic.

1. How many states will you need to implement this algorithm with a Moore Machine? How many registers will that require?
2. Using Karnaugh Maps, please give the simplified Boolean expression for the next state of each register. Be sure to include each Karnaugh Map as well as the boolean expressions in your answer.
3. Write a Verilog Module to implement the machine. The module should accept a 2-bit input and output a single bit. You should use the REGISTER class from the EECS 151 library in order to keep track of state.

Solution:

1. You require 3 states, one each possible sum mod 3. This can be done with 2 registers. The output is only high on state 00.
2. We can write the Karnaugh maps and expressions as given below:

   (a) $R0_{\text{next}} = R0(In1'In0' + In1In0) + R1In1In0' + R1'R0'In1'In0$

		$In1In0$			
		00	01	11	10
	00	0	1	0	0
	01	1	0	1	0
$R1R0$	11	X	X	X	X
	10	0	0	0	1

(b) $R1_{\text{next}} = R1(In1'In0' + In1In0) + R0In1'In0 + R1'R0'In1In0'$

		$In1In0$			
		00	01	11	10
	00	0	0	0	1
	01	0	1	0	0
$R1R0$	11	X	X	X	X
	10	1	0	1	0

3. The Verilog will be:

```
module Div3Calculator(in, out, clk, rst);

  input[1:0] in;
  input clk, rst;
  output out;

  reg r0, r1, r0_next, r1_next;

  assign r0_next = (r0 & (~^in)) |
  (r1 & in[1] & ~in[0]) |
  (~r1 & ~r0 & ~in[1] & in[0]);
  assign r1_next = (r1 & (~^in)) |
  (r0 & ~in[1] & in[0]) |
  (~r1 & ~r0 & in[1] & ~in[0]);

  assign out = ~r0 & ~r1;
  REGISTER_R #(.N(1), .INIT(1'b0)) reg0 (.q(r0), .d(r0_next),
  .clk(clk));
  REGISTER_R #(.N(1), .INIT(1'b0)) reg1 (.q(r1), .d(r1_next),
  .clk(clk));


endmodule
```

Problem 3: Encoding FSMs

Given the following State Transition Diagram:

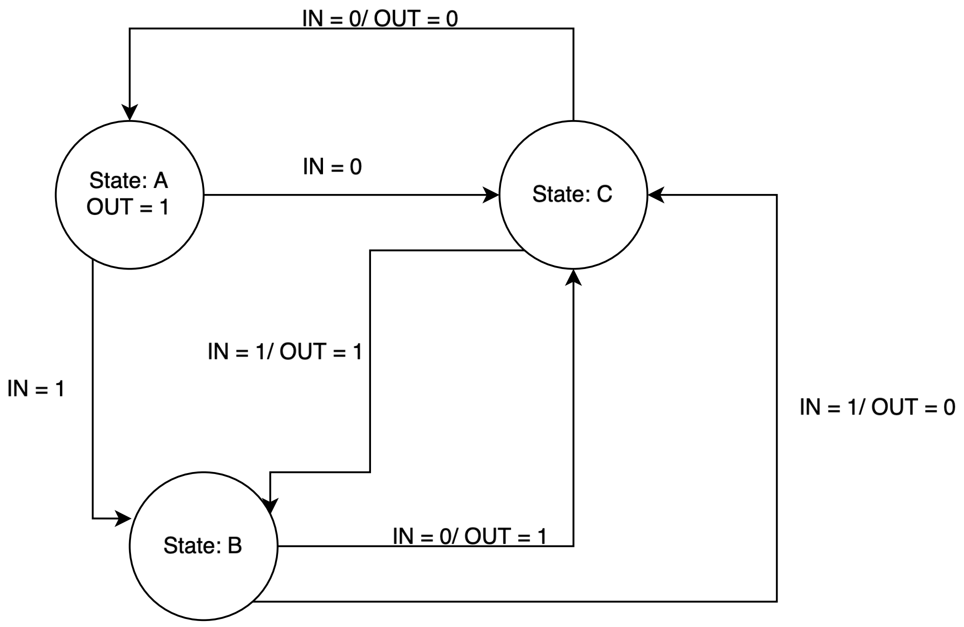
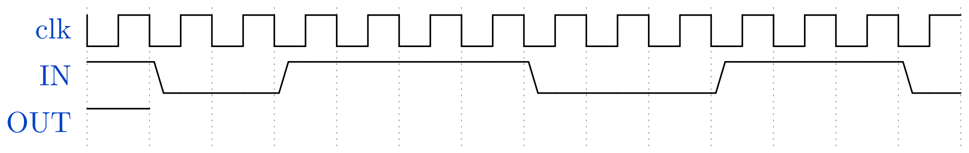


Figure 2: Serial to Parallel Waveform

1. Please complete the waveform diagram. Assume that the state is A when we stop drawing the OUT signal:



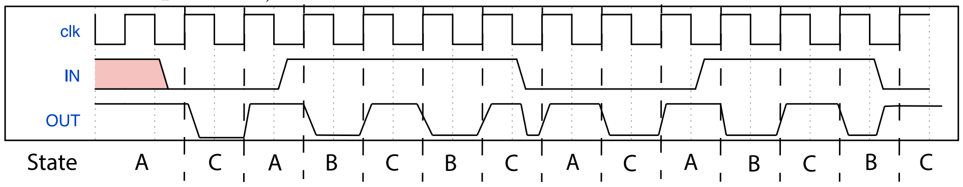
2. What is the least amount of registers needed to encode these states? Using this amount of states, please give the simplified boolean expressions for each of the next register states, as well as the output. You are encouraged to use Karnaugh maps to derive these expressions, but only the expressions need to be submitted.

3. In class we discussed a One-Hot finite state machine design, where you use a single register for each state. In this case, how many registers would you need? For this design, please derive the simplified boolean expressions for each of the next register states as well as the output.

4. Assume that these circuits are implemented using 2-input NANDs and 1-input Inverters only (in addition to registers). For each of the circuits, please calculate the number of transistors used in each design. You may assume a 2-input NAND uses 4 transistors, an inverter uses 2 transistors, and a register uses 12 transistors.

Solution:

1. Waveform will look like the following (you don't need to include the state, but it helps to solve the problem):



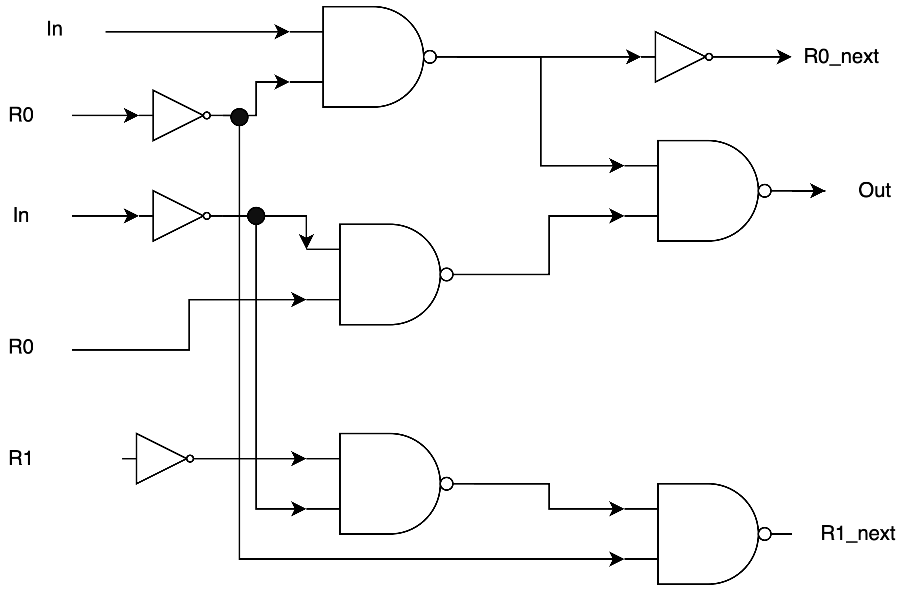
2. There are 3 states, so you can implement the FSM with 2 registers using binary encoded states. If we use A = 00, B = 01, and C = 10, the expressions become:

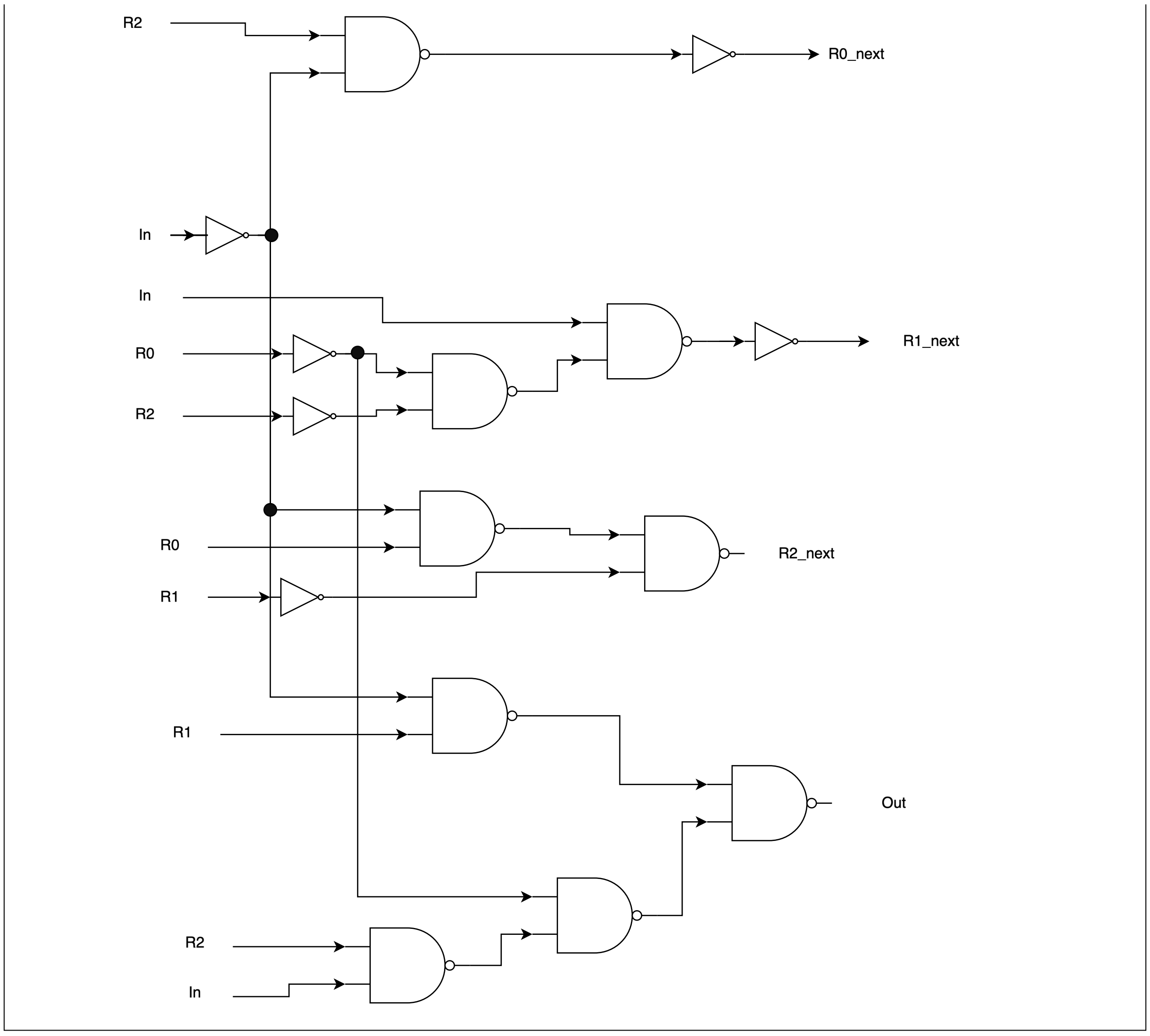
   (a) $R0_{\text{next}} = InR0'$
   (b) $R1_{\text{next}} = R0 + In'R1'$
   (c) OUT $= In'R1' + InR0'$

3. For a One-Hot FSM, you use 1 register per state, which requires 3 registers. Let A = 001, B = 010, and C = 100. The boolean expressions are written below:

   (a) $R0_{\text{next}} = R2In'$
   (b) $R1_{\text{next}} = In(R0 + R2)$
   (c) $R2_{\text{next}} = R1 + R0In'$
   (d) OUT $= R0 + R2In + R1In'$

4. The circuits are drawn below. The Binary encoded circuit requires 52 transistors, while the One-Hot requires 84 transistors.




R2
R0_next
In
In
R0
R1_next
R2
R0
R2_next
R1
R1
Out
R2
In

Problem 4: State Assignment

Given the following state diagram:

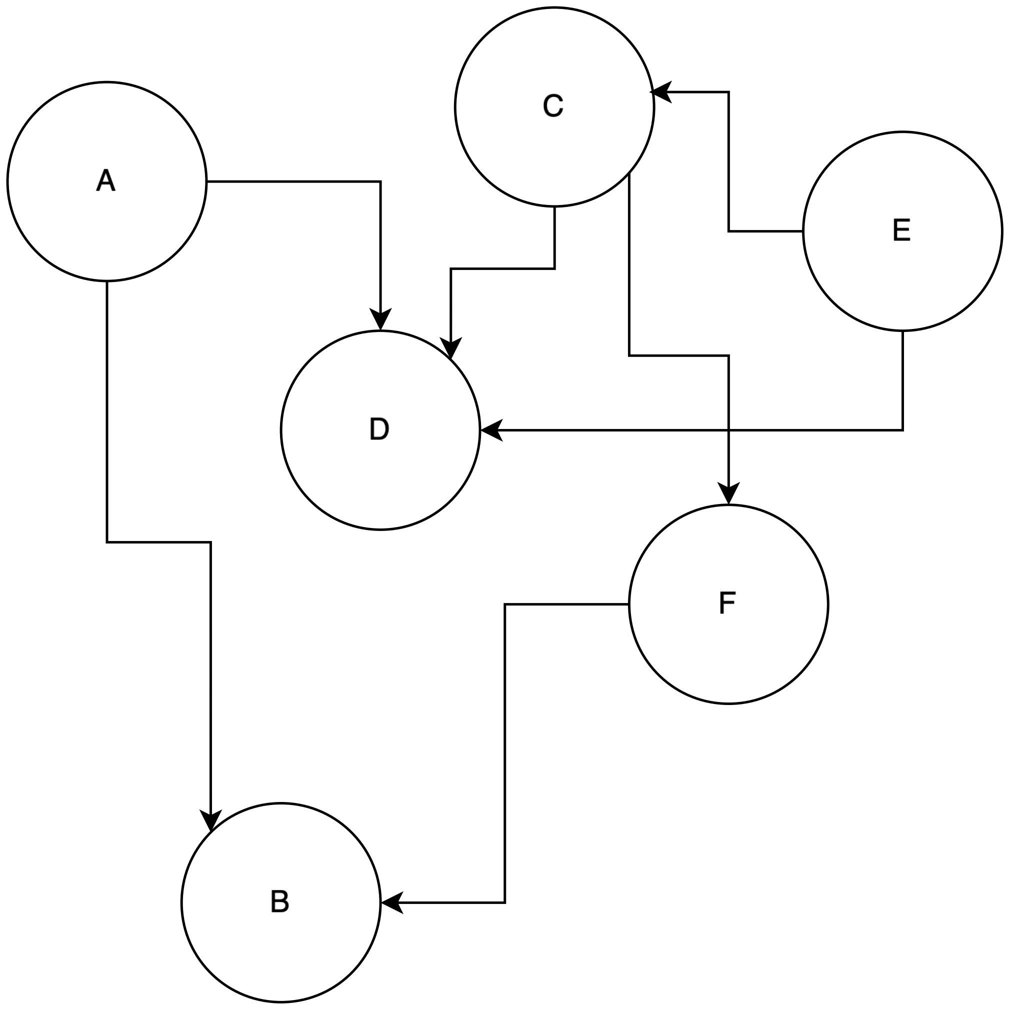


Figure 3: 6 State FSM

1. Please assign the states to values in order to minimize the total distance between states that feed into each other as we showed in lecture. You must encode A as state 0 (with n-bits depending on the number of registers). We use distance here as a heuristic to help minimize the amount of combinational logic that is required to transition between states. In this case, distance here is defined by the square of the Hamming Distance between two states (Hamming Distance is a very popular metric across a number of fields including information theory, coding theory, communication, etc. It is defined as the number of bits that change between two strings). For example if I encode A as 00 and B as 11, then the distance between them is $2^2 = 4$. Total distance is the sum of all distances between states that feed into each other. Minimizing this distance tends to minimize the combinational logic needed to solve for register states. **In your answer, please specify the state assignment as well as the total distance you calculated.**

Solution:

1. You can use a Karnaugh Map to help place the states. The following encoding is minimal, but there are a few options for where F or C can be placed.

   (a) A = 000

   (b) B = 001

   (c) C = 110

   (d) D = 010

   (e) E = 011

   (f) F = 101

   In this case, all of the distances are 1, except C to F which has a distance of 4 (2^2). So the total distance is 10.

Problem 5: PMOS Fabrication

1. What dopant(s) is(are) used to dope the source/drain for a PMOS transistor?
2. Similar to the NMOS we discussed in lecture, please describe the steps needed to fabricate a PMOS transistor.

Solution:

1. Boron
2. (a) Dope Wafer with p-.
   (b) EXTRA STEP: Create a n-well inside the p- wafer. Use diffusion mask w/ Arsenic.
   (c) Grow Gate Oxide.
   (d) Deposit Polysilicon.
   (e) Spin on Photoresist.
   (f) Place negative poly mask and expose with UV.
   (g) Wet etch to remove unmasked area.
   (h) Add diffusion mask to implant p type with Boron.
   (i) Grow a much thicker oxide.
   (j) Mask and etch contact point.
   (k) Add metal on top of oxide that fills in the contact points.