2o1-

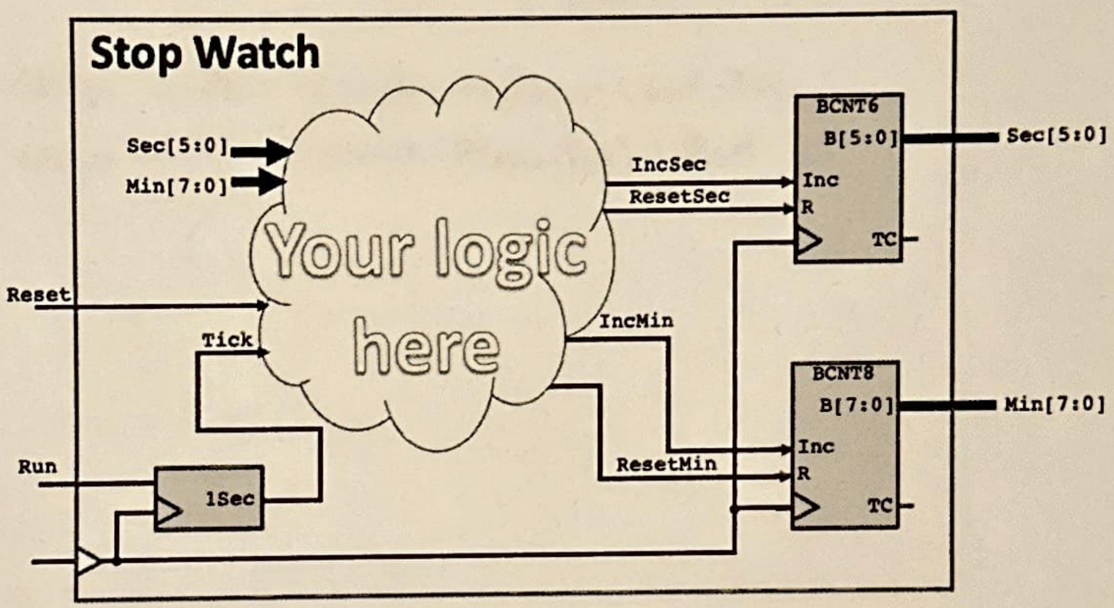
Name: Nichdos Matvianko Last 4 ID: 9670 **CSE 100**
HW 5 **Due: November 17, 2025 at 11:59 PM in Santa Cruz.** **Fall 2025**

Reading: Sections 4.8 and 5.1-5.7.

Instructions: Complete the problems below **in your own handwriting**. Box your answers where necessary. The grader will award 1 point for your name and the last four digits of your student ID, and 1 point for neatness.

Problem 1

(14 points) In this problem, you will **write logic equations for controlling a Stop Watch** that counts up to 199 minutes and 59 seconds. As shown in the figure below, the Stop Watch is constructed from two standard binary counters. A 6-bit counter is used to hold the seconds and an 8-bit counter for the minutes. You are designing the combinational module below that should replace the cloud in the figure below.



The Stop Watch should reset the time to 0 when its **Reset** input is high. Otherwise it will advance the time when its **Run** input is held high. The two standard binary counters, BCNT6 and BCNT8, operate as follows:

- These standard binary counters will reset to 0 if their **R** input is high on the clock edge.
- Otherwise they will increment if their **Inc** input is high on the clock edge and hold their value when their **Inc** input is low.

The Stop Watch should rollover from 199:59 back to 000:00 if **Run** is high. The **Tick** signal will be high for exactly one clock cycle once every second while **Run** is high.

Computer Science and Engineering, University of California, Santa Cruz

Below, write combinational logic equations for **IncSec, ResetSec, IncMin,** and **ResetMin** to control BCNT6 and BCNT8 based on the inputs **Reset** and **Tick**, as well as the current time values **Sec** and **Min.** Use Verilog syntax to access specific bits within a bit vector (e.g. Sec[0] is the LSB of bus Sec).

wire 59seconds = Sec[5] • Sec[4] • Sec[3] • Sec[2] • Sec[1] • Sec[0]

wire 199mins = Min[7] • Min[6] • ~Min[5] • ~Min[4] • ~Min[3]
• Min[2] • Min[1] • Min[0]

assign IncSec = ~59seconds • ~Reset • Tick

assign ResetSec = (59seconds • Tick) | Reset

assign IncMin = 59seconds • ~199mins • ~Reset • Tick

assign ResetMin = (59seconds • 199mins • Tick) | Reset

Computer Science and Engineering, University of California, Santa Cruz

Problem 2

(22 points) Construct state machines that follows the specification below. To recieve full credit:

- Identify whether the state machine will be Mealy or Moore,
- minimize the number of states in your machine,
- draw the state diagram, and
- identify the initial state.

You do not need to implement this machine; **you only need to provide the state diagram** and your work to obtain it. **You will need to minimize it or show that it is already minimal** by applying at least one iteration of the state minimization algorithm to prove that your state machine is minimal.

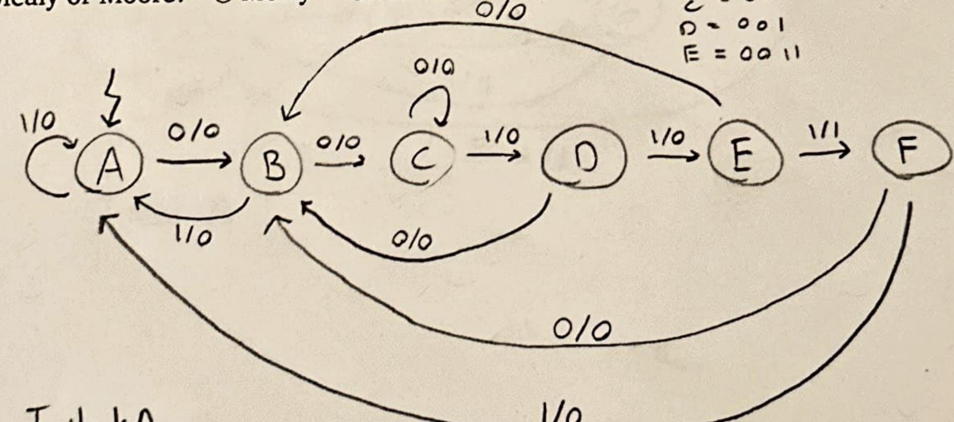
a. (8 points) Construct a state machine for a sequential circuit with one synchronous input x and one output z. The output z should go high when the input sequence is exactly **two consecutive 0's followed by three consecutive 1's**. The output z should go high *after* the cycle on which the sequence was detected and go low the cycle after, until the next time the input sequence is detected. The first five outputs can be either 0 or 1 since the circuit's output will be ignored until five inputs have been given.

The output z is shown below for a sample sequence of inputs **x**:

```
x  0 0 1 1 1 0 1 0 0 1 1 0 0 1 1 1 0 0 1 1 1 1 1 0 . . .
z  ? ? ? ? ? 1 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 1 0 0 . . .
```

Mealy or Moore? ○ Mealy ● Moore

A = N/A
B = 0
C = 00
D = 001
E = 0011
F = 00111



Initial: A

Computer Science and Engineering, University of California, Santa Cruz

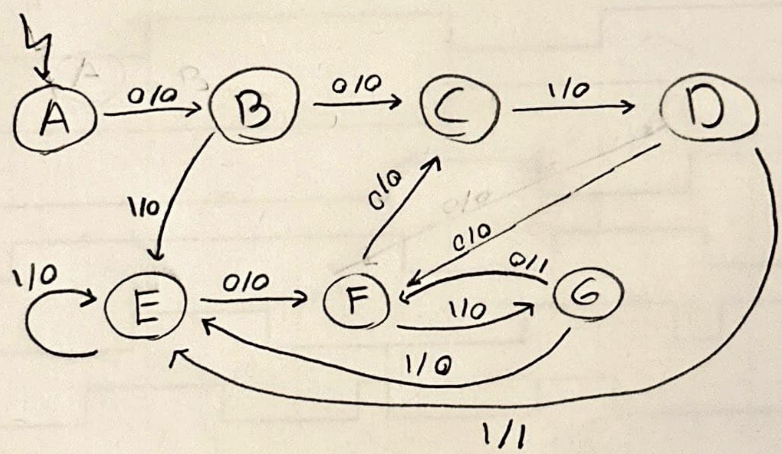
b. (14 points) **Draw the state diagram** for a sequential circuit with one synchronous input **x** and one output **z**. The output **z** is high whenever the last four inputs (**including the current input**) are 0011 or 1010. The output should go high *on the same cycle* in which the current input completes sequence. The first three outputs can be either 0 or 1 since the circuit's output will be ignored until four inputs have been given. Note that the two sequences can overlap with each other and themselves.

The output **z** is shown below for a sample sequence of inputs **x**:

```
x  0 1 0 1 0 0 0 0 1 1 0 0 1 1 0 1 0 0 1 1 0 1 0 0 . . .
z  ? ? ? 0 1 0 0 0 0 1 0 0 0 1 0 0 1 0 0 1 0 0 1 0 . . .
```

Mealy or Moore? ● Mealy ○ Moore

A = N/A F = 10
B = 0 G = 101
C = 00
D = 001
E = 1

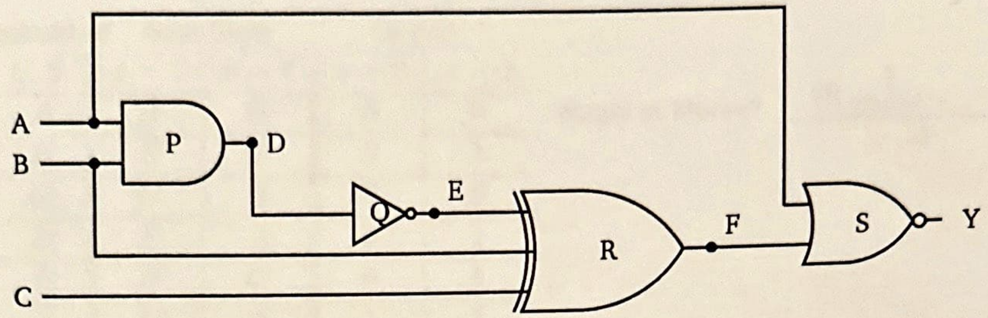


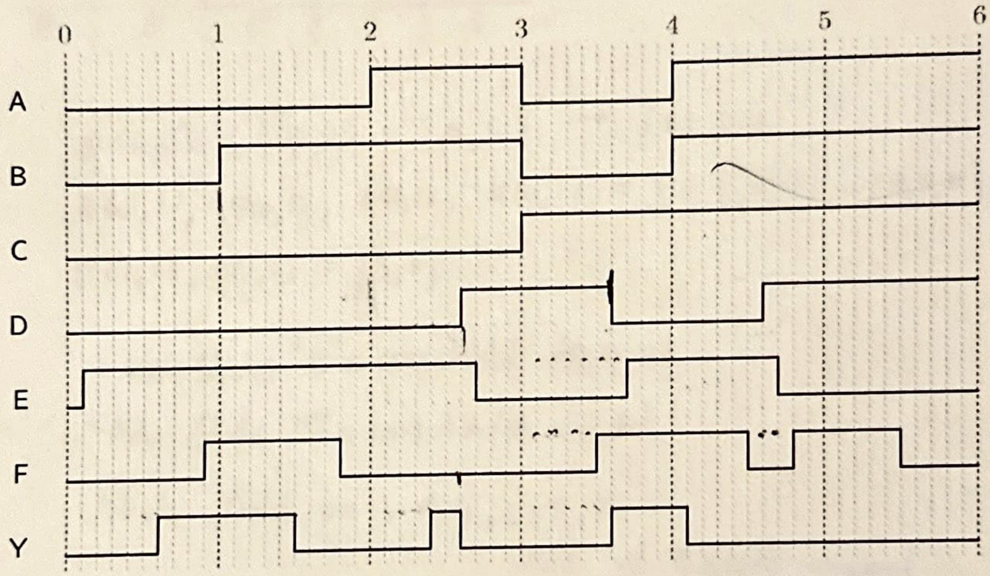
Initial: A

Computer Science and Engineering, University of California, Santa Cruz

Problem 3

(4 points) The waveform for the circuit below is given. Find the gate delay for gates P, Q, R, S. Assume nodes D, E, F and Y have an initial value of 0. Assume the D Flip-Flops have zero propagation delay. Each vertical line in the waveform is 0.1 delay units apart.





$P=$ 0.6 $Q=$ 0.1

$R=$ 0.8 $S=$ 0.6

Computer Science and Engineering, University of California, Santa Cruz

Problem 4

(5 points) A state transition table is given below. State whether the machine is Mealy or Moore, then minimize the number of states in the state machine below by applying the state minimization algorithm as many times as needed. Give the state table of the final minimized machine.

Current State	Next State		Output	
$C.S.$	$x=0$	$x=1$	$x=0$	$x=1$
A	H	B	0	0
B	E	C	0	1
C	C	H	0	0
D	F	A	1	1
E	B	C	0	1
F	D	A	1	1
G	G	A	0	1
H	B	C	0	1

Mealy or Moore? Mealy

A(0,0) , C(0,0) = {A, C} → {A}, {C}

B(0,1), E(0,1), G(0,1), H(0,1) = {B, E, G, H} → {B, E, H}, {G}

D(1,1), F(1,1) = {D, F}

New State "B" = consolidated {B, E, H}

New State "D" = consolidated {D, F}.

Thus, states are A, B, C, D, G

5 states vs 8 states original

C.S.	Next State $x=0$	Next State $x=1$	Output $x=0$	Output $x=1$
A	B	B	0	0
B	B	C	0	1
C	C	B	0	0
D	D	A	1	1
G	G	A	0	1

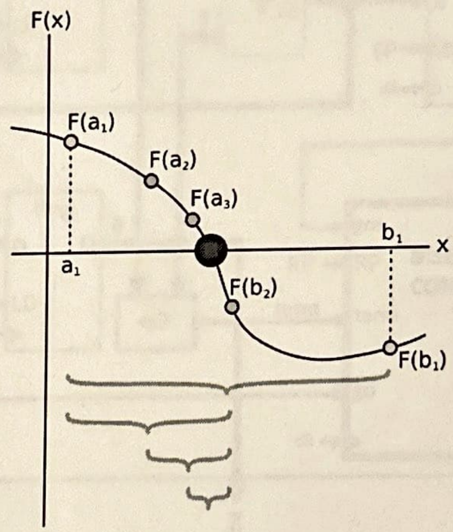
Problem 5

The **bisection method** is a sequential method for finding the root of a polynomial. In this problem you will **design a state machine within a module named** `BISECT CONTROL` **that controls** `POLY ROOT`, which will implement this method.

The bisection method takes as input two values a and b and a polynomial $P(x)$ such that:

1. $a < b$,
2. $P(a) < 0$, and
3. $P(b) > 0$.

Since polynomials are continuous functions, there must be a value z between a and b where $P(z) = 0$ (the polynomial crosses the x-axis at z). This value z is the root of the polynomial. A visualization of this method is provided below.

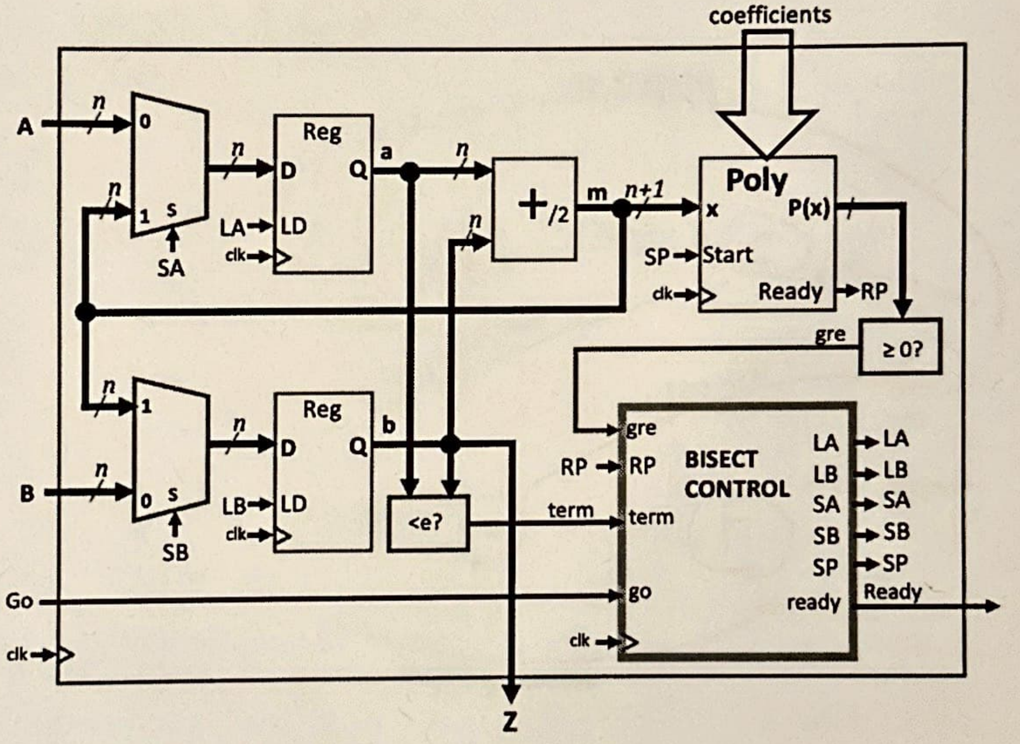


The bisection method is similar to a binary search. It finds the midpoint, m between a and b and then checks whether $P(m) >= 0$. If so, then there is a root between a and m. Otherwise there is a root is between m and b. This is repeated until a and b are close enough, say within "epsilon." (We will use e for "epsilon".) The method can be summarized by the pseudocode below:

```
while (a + e < b) {
  m = (a + b) / 2;
  if (P(m) >= 0), then b = m
  else, a = m
}
return b;
```


Computer Science and Engineering, University of California, Santa Cruz

A datapath for `POLY ROOT` is shown below. There are two registers holding `a` and `b`, and `m` is the output of the module labeled +/2. The registers can either be loaded with the external inputs `A` and `B` or `m` by controlling the two multiplexers. The combinational module +/2 calculates m by adding a and b and shifting right. There are two other combinational modules, one for testing when a and b are within epsilon, and the other for testing whether $P(m)$ is greater than or equal to 0. The output is the value stored in the `b` register. The module calculating $P(m)$ can be configured for each specific polynomial by providing its coefficients.



In this problem you will design the controller, `BISECT CONTROL`. This module controls the registers to load the initial values of `a` and `b` and then update either with `m`. It also controls the `POLY` module which calculates $P(m)$. The calculation of $P(m)$ will begin by asserting `POLY`'s `Start` input. The control has two inputs: `gre`, which signals whether $P(m) >= 0$, and `term`, which signals whether `a` and `b` are within epsilon. The controller will need to wait for the `Ready` output of `POLY` before reacting to `gre`. The controller also has a `Go` input which initiates the root calculation and a `Ready` output that your controller asserts when it is ready to begin a new computation and the result of the previous calculation is available on the output `Z`.

Design BISECT CONTROL by providing the following:

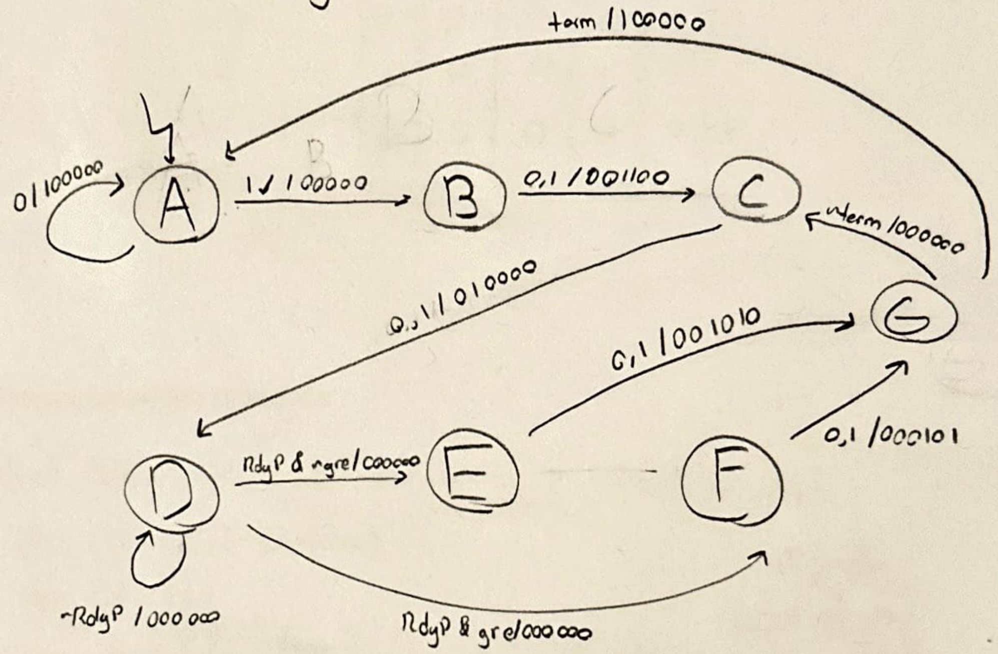
a) (6 points) Draw the state diagram for BISECT CONTROL.

b) (6 points) Make a state transition table.

c) (4 points) Obtain the next-state and output equations for your state machine assuming a one-hot state encoding.

State diagram:

Moore Diagram:



State transition table:

C.S.	Next State	Ready ↓ R	Start ↓ S	LA	LB	SelA ↓ SA	SelB ↓ SB
A	ngo → A, go → B	1	0	0	0	0	0
B	→ C	0	0	1	1	0	0
C	→ D	0	1	0	0	0	0
D	nrdyP → D, rdyP & ngre → E, RdyP & gre → F	0	0	0	0	0	0
E	→ G	0	0	1	0	1	0
F	→ G	0	0	0	1	0	1
G	nterm → C; term → A	0	0	0	0	0	0

Next-state and output equations:

Next State:

$A = (A \cdot nGo) + (G \cdot term)$

$B = (A \cdot Go)$

$C = B + (G \cdot nterm)$

$D = C + (D \cdot nRdyP)$

$E = D \cdot RdyP \cdot ngre$

$F = D \cdot RdyP \cdot gre$

$G = E + F$

Outputs:

$Ready = A$

$Start = C$

$LdA = B + E$

$LdB = B + F$

$SelA = E$

$SelB = F$

Computer Science and Engineering, University of California, Santa Cruz