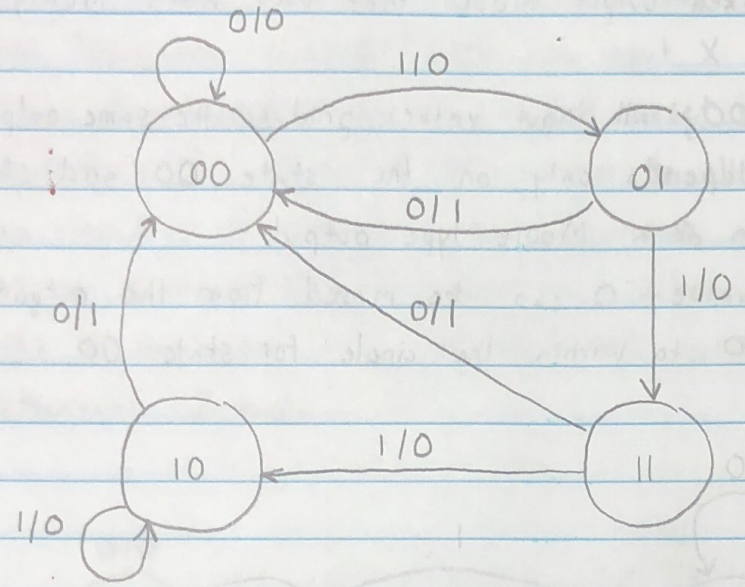
Stewart Dulaney
CS 42
Section 4105
SID: 1545566

Chapter 4 Problems

4-1 Equivalent State Illustration



States 10 and 11 are equivalent and can be merged
- Under input 0, both states produce output 1 and have state 00 as their next state
- Under input 1, both states produce output 0 and have state 10 as their next state

States 01 and 11 are equivalent and can be merged
- Under input 0, both states produce output 1 and have state 00 as their next state
- Under input 1, both states produce output 0 and have next states 11 and 10, which have just been shown to be equivalent

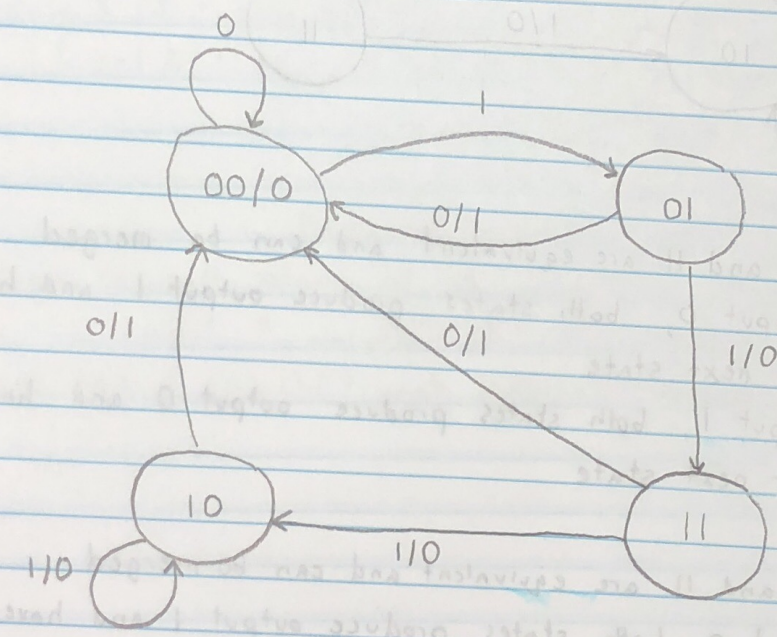
∴ The three states 10, 11, and 01 are equivalent

If the circuit was analyzed for redesign, the new design has two states and one flip-flop instead of four states and two flip-flops

4-2 Mixed Mealy and Moore Outputs

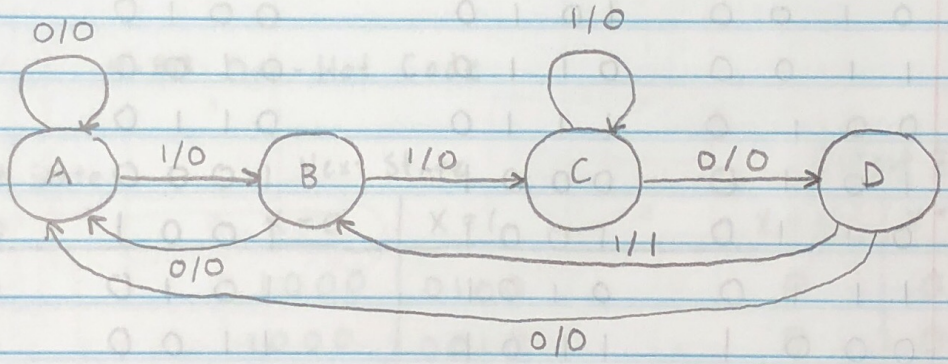
The state diagram for the previous problem can be used to illustrate a mixed output model that uses both Mealy and Moore type outputs

- For state 00, all input values produce the same output value 0 on Z
- The output depends only on the state 00 and satisfies the definition of a Moore type output
- The output value 0 can be moved from the outgoing transitions on state 00 to within the circle for state 00



4-3 Finding a State Diagram for a Sequence Recognizer

- Design a circuit that recognizes the occurrence of a particular sequence of bits, regardless of where it occurs in a longer sequence
- This "sequence recognizer" has one input X and one output Z
- It has Reset applied to the direct reset inputs on its flip-flops to initialize the state of the circuit to all zeros
- The circuit is to recognize the occurrence of the sequence of bits 1101 on X by making Z equal to 1 when the previous three inputs to the circuit were 110 and the current input is a 1
- Otherwise, Z equals 0



Present State	Next State		Output Z	
	X = 0	X = 1	X = 0	X = 1
A	A	B	0	0
B	A	C	0	0
C	D	C	0	0
D	A	B	0	1

4-4 Finding a State Diagram for a BCD-to-Excess-3 Decoder

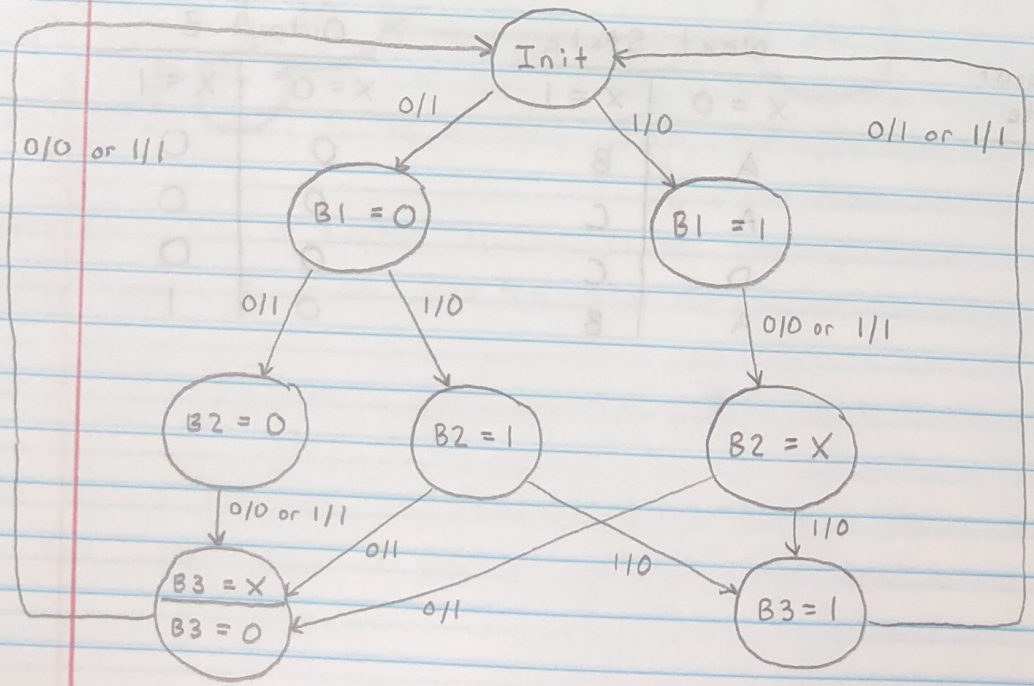
- The excess-3 code for a decimal digit is the binary combination corresponding to the decimal digit digit plus 3

(a) Sequences in Order of Digits Represented

BCD Input				Excess-3 Output			
1	2	3	4	1	2	3	4
0	0	0	0	1	1	0	0
1	0	0	0	0	0	1	0
0	1	0	0	1	0	1	0
1	1	0	0	0	1	1	0
0	0	1	0	1	1	1	0
1	0	1	0	0	0	0	1
0	1	1	0	1	0	0	1
1	1	1	0	0	1	0	1
0	0	0	1	1	1	0	1
1	0	0	1	0	0	1	1

(b) Sequences in Order of Common Sequences

BCD Input				Excess-3 Output			
1	2	3	4	1	2	3	4
0	0	0	0	1	1	0	0
0	0	0	1	1	1	0	1
0	0	1	0	1	1	1	0
0	1	0	0	1	0	1	0
0	1	1	0	1	0	0	1
1	0	0	0	0	0	1	0
1	0	0	1	0	0	1	1
1	0	1	0	0	0	0	1
1	1	0	0	0	1	1	0
1	1	1	0	0	1	0	1



4-5

State Assignments for the Sequence Recognizer

2-Bit Binary Gray Code

Present State	Next State		Output Z	
AB	X = 0	X = 1	X = 0	X = 1
00	00	01	0	0
01	00	11	0	0
11	10	11	0	0
10	00	01	0	1

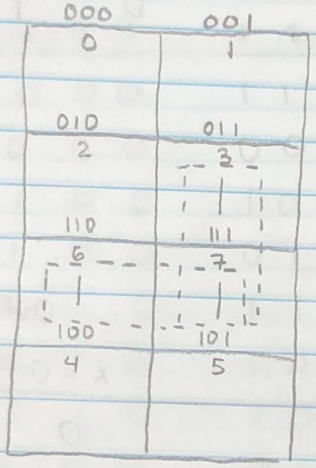
4-Bit One-Hot Code

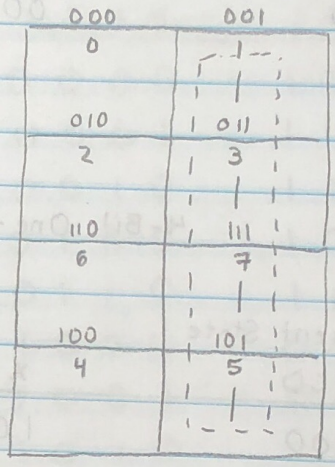
Present State	Next State		Output Z	
ABCD	X = 0	X = 1	X = 0	X = 1
1000	1000	0100	0	0
0100	1000	0010	0	0
0010	0001	0010	0	0
0001	1000	0100	0	1

4-6 Gray Code Design for the Sequence Recognizer

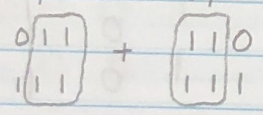
- Two flip flops are needed to represent the four states

$$A(t+1) = D_A(A,B,X) = \Sigma m(3,6,7)$$
$$B(t+1) = D_B(A,B,X) = \Sigma m(1,3,5,7)$$
$$Z(A,B,X) = \Sigma m(5)$$



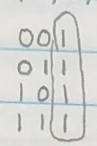


3,7 6,7



$D_A = BX + AB$

1,3,5,7



$D_B = X$

4-7

One-Hot Code Design for the Sequence Recognizer

- Four flip-flops are needed to represent the four states

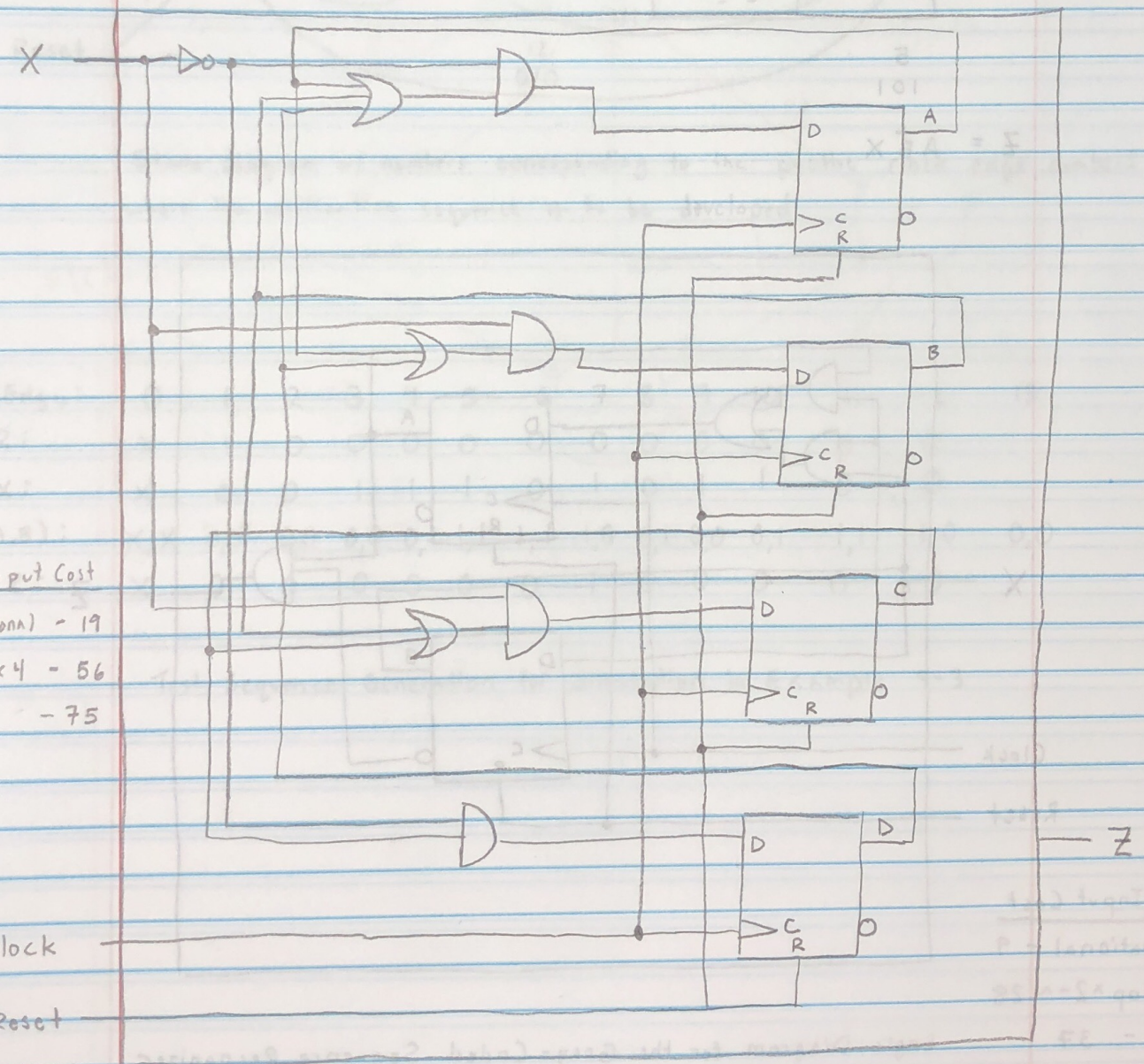
$$A(t+1) = D_A = A\bar{X} + B\bar{X} + D\bar{X} = (A + B + D)\bar{X}$$

$$B(t+1) = D_B = AX + DX = (A + D)X$$

$$C(t+1) = D_C = BX + CX = (B + C)X$$

$$D(t+1) = D_D = C\bar{X}$$

$$Z = DX$$



Logic Diagram for the One-Hot Coded Sequence Recognizer with D Flip-Flops

Gate Input Cost

Combinational - 19

Flip-Flop x 4 - 56

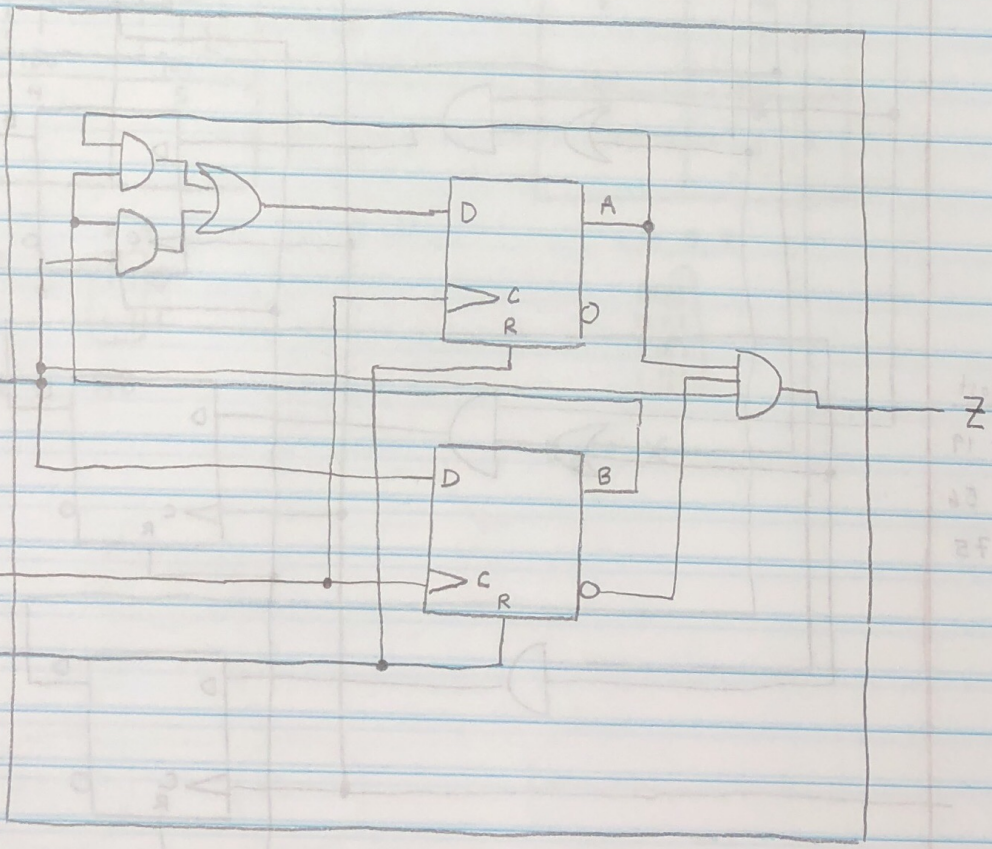
Total - 75

4-6 cont'd

000	001
0	1
010	011
2	3
110	111
6	7
100	101
4	5

5
101

$Z = A\bar{B}X$



Gate Input Cost

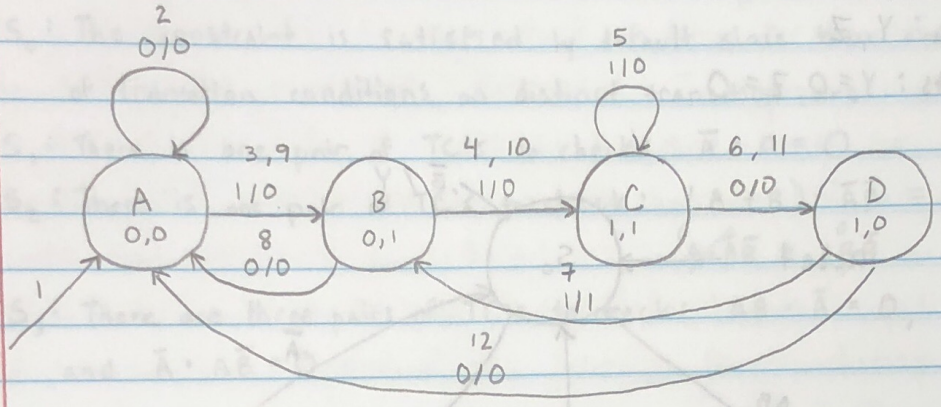
Combinational - 9

Flip-Flop ×2 - ~ 28

Total - 37

Logic Diagram for the Gray-Coded Sequence Recognizer with D Flip-Flops

4-8 Verifying the Sequence Recognizer



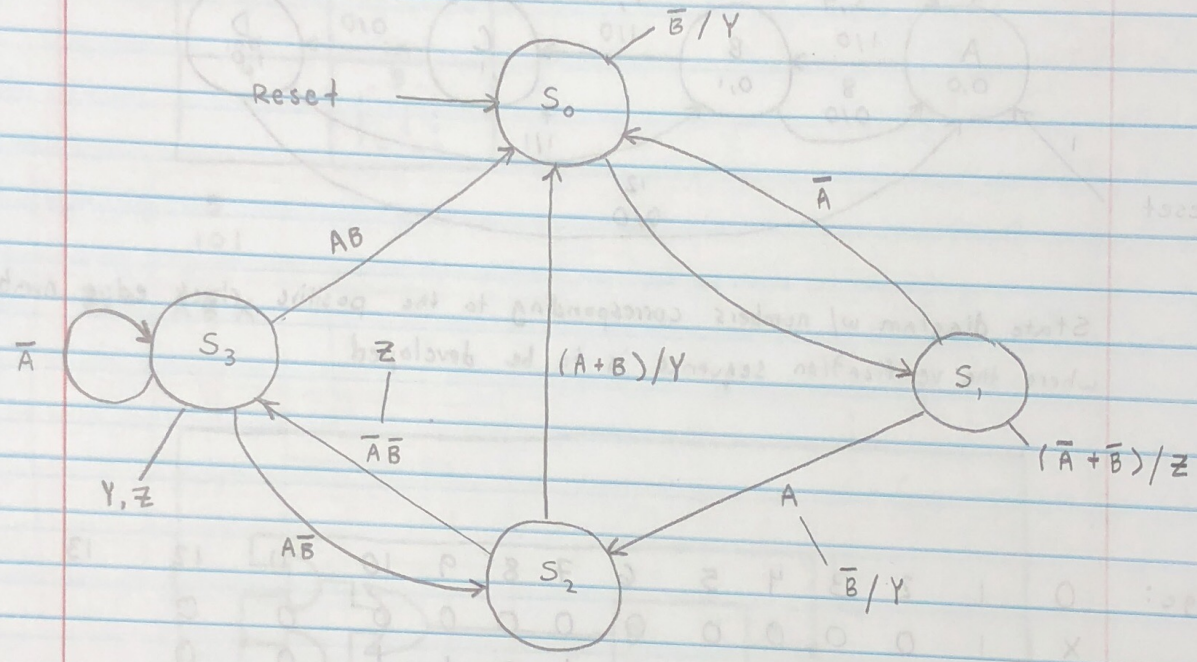
State diagram w/ numbers corresponding to the positive clock edge numbers where the verification sequence is to be developed

Clock Edge:	0	1	2	3	4	5	6	7	8	9	10	11	12	13
Input R:	X	1	0	0	0	0	0	0	0	0	0	0	0	
Input X:	X	0	0	1	1	1	0	1	0	1	1	0	0	
State (A,B):	X,X	0,0*	0,0	0,0	0,1	1,1	1,1	1,0	0,1	0,0	0,1	1,1	1,0	0,0
Output Z:	X	0	0	0	0	0	0	1	0	0	0	0	0	

Test Sequence Generation for Simulation in Example 4-3

4-9 Checking Constraints

Inputs : A, B
Outputs : Y, Z
Defaults : Y = 0, Z = 0



State Machine Diagram

4-9 cont'd

Transition Conditions

Constraint 1: $T_{ij} \cdot T_{ik} = 0$

S_0: The constraint is satisfied by default since there are no pairs of transition conditions on distinct transition arcs.

S_1: There is one pair of TC's to check: $\bar{A} \cdot A = 0$

S_2: There is one pair of TC's to check: $(A + B) \cdot \bar{A}\bar{B} = 0$

$\overset{0}{\cancel{A \cdot \bar{A}\bar{B}}} + \overset{0}{\cancel{AB \cdot \bar{B}}}$

S_3: There are three pairs of TC's to check: $AB \cdot \bar{A} = 0$, $AB \cdot A\bar{B} = 0$, and $\bar{A} \cdot A\bar{B} = 0$

All results are 0 → constraint 1 is satisfied

Constraint 2: $\Sigma T_{ij} = 1$

S_0: The transition is unconditional and has an implicit transition condition of 1.

S_1: $\bar{A} + A = 1$

S_2: $(A + B) + \bar{A}\bar{B} = 1$

S_3: $\bar{A} + AB + A\bar{B} = 1$

All results are 1 → constraint 2 is satisfied

Output Conditions

Constraint 1: $O_{ij} \cdot O_{ik} = 0$

S_0: There is only one output condition, $\bar{B}$ on output action Y, so the constraint is satisfied by default

S_1: The first coincident output variable is Y and its values are 1 where Y appears for TOC $A \cdot \bar{B}$, and 0 by default where Y does not appear for input conditions $\bar{A}$ and AB. Note that if $\bar{B}$ is interpreted without ANDing with transition condition A, then check $\bar{A} \cdot \bar{B} \neq 0$ incorrectly fails! The second coincident output variable is Z, with value 1 for $\bar{A} + \bar{B}$ and 0 by default for input condition AB. In general, it is impossible for an invalid case to occur due to a default output action. So the constraint is satisfied.

S_2: The first coincident output variable is Y and the second is Z. Y has value 1 for output condition $A + B$, and by default value 0 for $\bar{A}\bar{B}$. Z has value 1 for output condition $\bar{A}\bar{B}$ and 0 by default for $A + B$. Due to the use of a default value, the constraint is satisfied.

S_3: There is no coincident output variable with differing output values, so the output constraint is satisfied by default.

4-9 cont'd

Constraint 2: $\Sigma O_{ij} = 1$

S_0: There is a single output condition $\overline{B}$ for which $Y = 1$. By default, $Y = 0$ for the output condition for complement of $\overline{B} = B$. ORing the conditions, $\overline{B} + B = 1$. In general, with a default output specified, this will be the case since the default covers all input combinations not covered by specified output conditions, so the constraint is satisfied.

S_1 through S_3: Because of the default output action for variables Y and Z, as for S_0, the constraint is also satisfied.

All results are 1 → constraint 2 is satisfied

4-10 State-Machine Design for a Batch Mixing System Control

Input and Output Variables for the Batch Mixing System

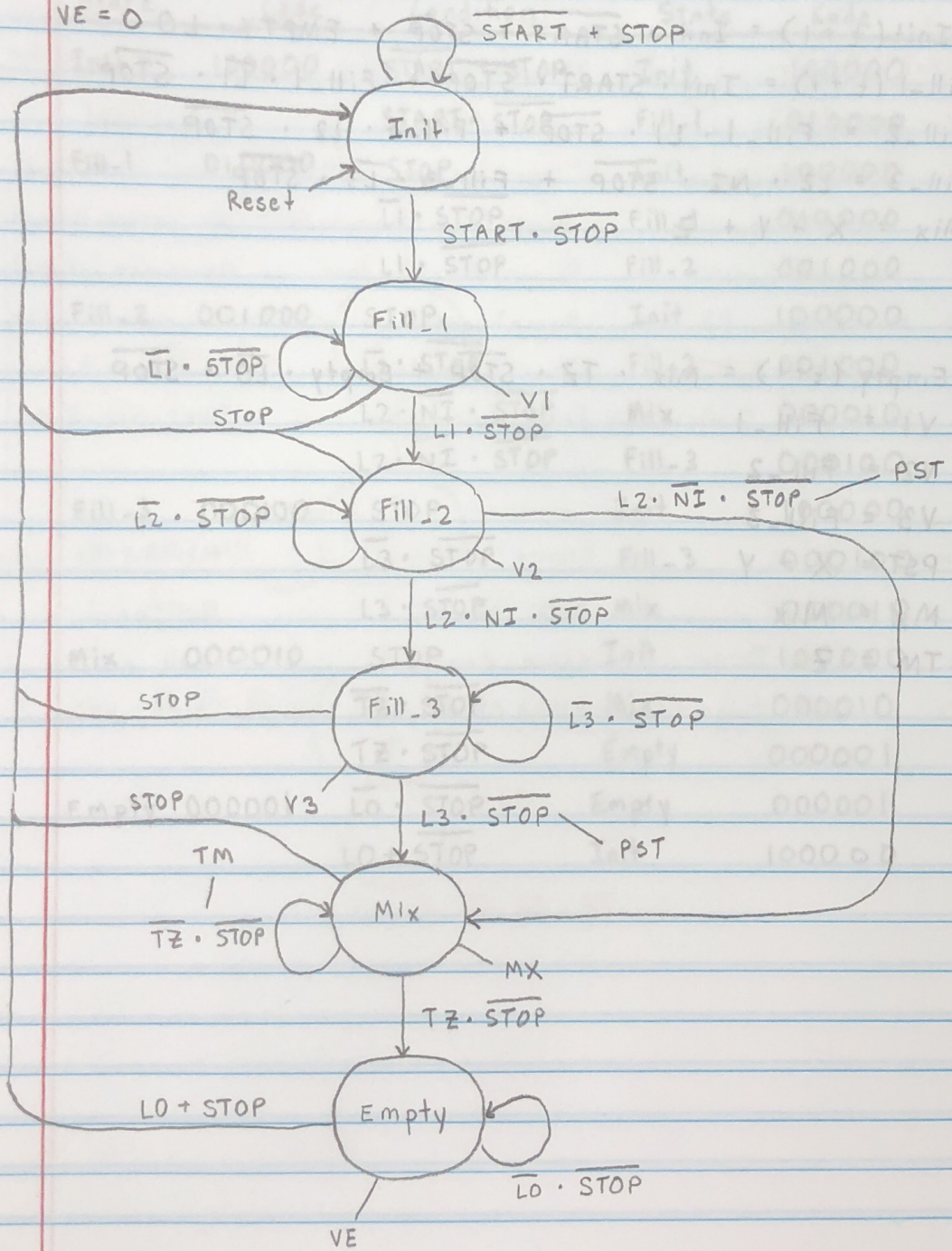
Input	Meaning for Value 1	Meaning for Value 0
NI	Three ingredients	Two ingredients
Start	Start a batch cycle	No action
Stop	Stop an on-going batch cycle	No action
L0	Tank empty	Tank not empty
L1	Tank filled to level 1	Tank not filled to level 1
L2	Tank filled to level 2	Tank not filled to level 2
L3	Tank filled to level 3	Tank not filled to level 3
TZ	Timer at value 0	Timer not at value 0

Output	Meaning for Value 1	Meaning for Value 0
MX	Mixer on	Mixer off
PST	Load timer with value from D	No action
TM	Timer on	Timer off
V1	Valve open for ingredient 1	Valve closed for ingredient 1
V2	Valve open for ingredient 2	Valve closed for ingredient 2
V3	Valve open for ingredient 3	Valve closed for ingredient 3
VE	Output valve open	Output valve closed

4-10 cont'd

State-Machine Diagram for Batch Mixing System

Default: MX = 0, PST = 0
TM = 0, V1 = 0, V2 = 0, V3 = 0,
VE = 0



Excitation and Output Equations

4-10 cont'd

$X = Fill\_2 \cdot L2 \cdot \overline{NI} \cdot \overline{STOP}$

$Y = Fill\_3 \cdot L3 \cdot \overline{STOP}$

$Z = Mix \cdot \overline{TZ} \cdot \overline{STOP}$

$Init(t+1) = Init \cdot \overline{START} + STOP + EMPTY \cdot L0$

$Fill\_1(t+1) = Init \cdot START \cdot \overline{STOP} + Fill\_1 \cdot \overline{L1} \cdot \overline{STOP}$

$Fill\_2 = Fill\_1 \cdot L1 \cdot \overline{STOP} + Fill\_2 \cdot \overline{L2} \cdot \overline{STOP}$

$Fill\_3 = L2 \cdot NI \cdot \overline{STOP} + Fill\_3 \cdot \overline{L3} \cdot \overline{STOP}$

$Mix = X + Y + Z$

$Empty(t+1) = Mix \cdot TZ \cdot \overline{STOP} + Empty \cdot \overline{L0} \cdot \overline{STOP}$

$V1 = Fill\_1$

$V2 = Fill\_2$

$V3 = Fill\_3$

$PST = X + Y$

$MX = Mix$

$TM = Z$

4-10 cont'd

State Table for the Batch Mixing System

State	State Code	Transition Condition	Next State	State Code	Non-Zero Outputs Including Mealy Outputs Using TCs*
Init	100000	$\overline{START} + STOP$	Init	100000	
		$START \cdot \overline{STOP}$	Fill-1	010000	
Fill-1	010000	$STOP$	Init	100000	V1
		$\overline{L1} \cdot \overline{STOP}$	Fill-1	010000	
		$L1 \cdot \overline{STOP}$	Fill-2	001000	
Fill-2	001000	$STOP$	Init	100000	V2
		$\overline{L2} \cdot \overline{STOP}$	Fill-2	001000	
		$L2 \cdot \overline{NI} \cdot \overline{STOP}$	Mix	000010	PST*
		$L2 \cdot NI \cdot \overline{STOP}$	Fill-3	000100	
Fill-3	000100	$STOP$	Init	100000	V3
		$\overline{L3} \cdot \overline{STOP}$	Fill-3	000100	
		$L3 \cdot \overline{STOP}$	Mix	000010	PST*
Mix	000010	$STOP$	Init	100000	MX
		$\overline{TZ} \cdot \overline{STOP}$	Mix	000010	TM*
		$TZ \cdot \overline{STOP}$	Empty	000001	
Empty	000001	$\overline{LO} \cdot \overline{STOP}$	Empty	000001	VE
		$LO + \overline{STOP}$	Init	100000	

4-11

State-Machine Design of a Sliding Door Control

Input and Output Variables for the Sliding Door Control

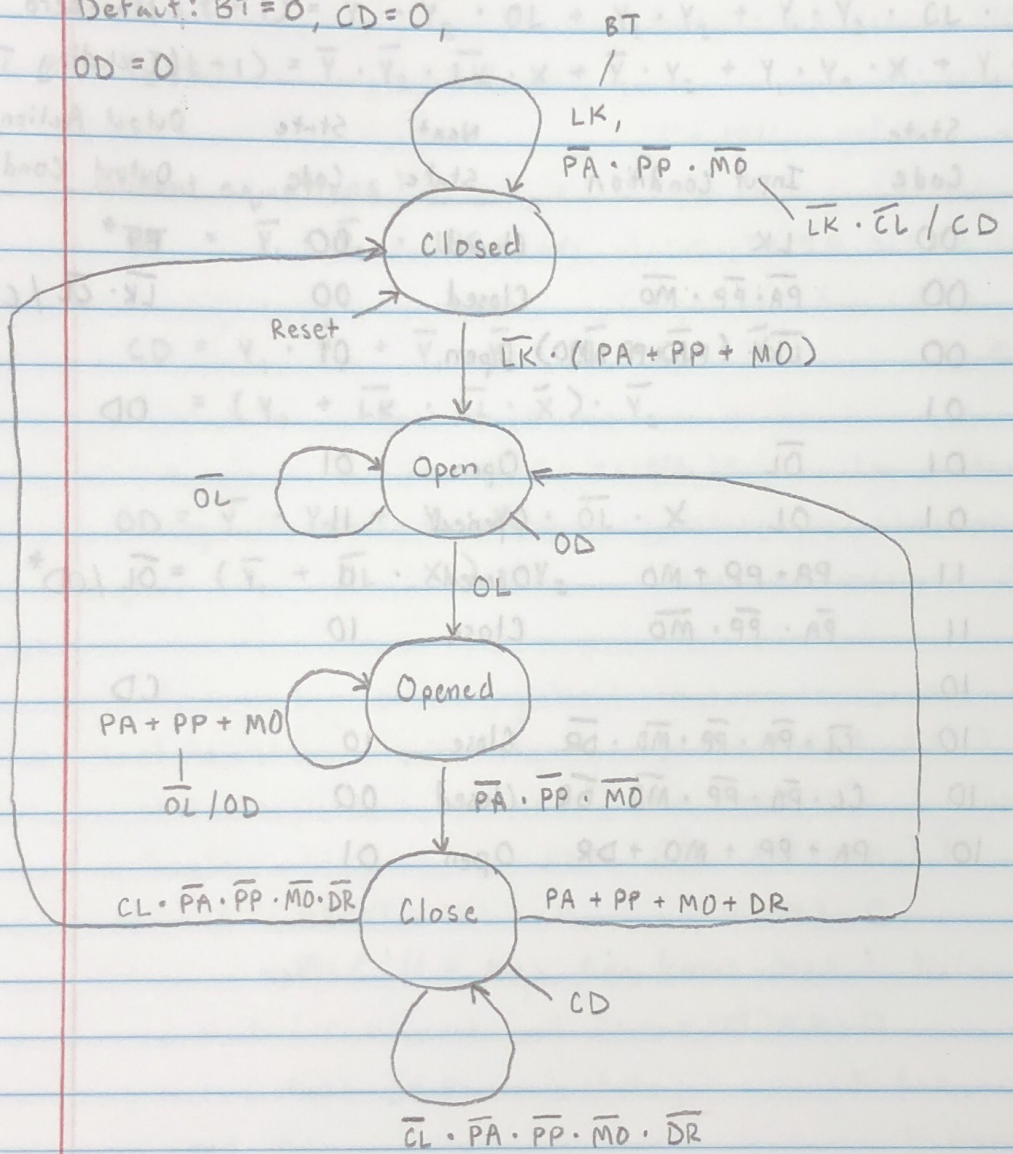
Input Symbol	Name	Meaning For Value 1	Meaning For Value 0
LK	Lock with key	Locked	Unlocked
DR	Door Resistance Sensor	Door resistance ≥ 15 lb	Door Resistance ≤ 15 lb
PA	Approach Sensor	Person/object approach	No person/object approach
PP	Presence Sensor	Person/object in door	No person/object in door
MO	Manual Open PB	Manual open	No manual open
CL	Close Limit Switch	Door fully closed	Door not fully closed
OL	Open Limit Switch	Door fully open	Door not fully open

Output Symbol	Name	Meaning For Value 1	Meaning For Value 0
BT	Bolt	Bolt closed	Bolt open
CD	Close Door	Close door	Null action
OD	Open Door	Close door	Null action

4-11 cont'd

State-Machine Diagram for the Automatic Sliding Door

Default: BT = 0, CD = 0, OD = 0



4-11 cont'd Modified State Table for the Automatic Sliding Door

State	State Code	Input Condition	Next State	State Code	Non-Zero Outputs (Including TCD and TOLD Output Actions and Output Conditions*)
Closed	00	LK	Closed	00	BT*
	00	$\overline{PA} \cdot \overline{PP} \cdot \overline{MO}$	Closed	00	$\overline{LK} \cdot \overline{CL}$ / CD*
	00	$\overline{LK} \cdot (PA + PP + MO)$	Open	01	
Open	01				OD
	01	$\overline{OL}$	Open	01	
	01	OL	Opened	11	
Opened	11	PA + PP + MO	Opened	11	$\overline{OL}$ / OD*
	11	$\overline{PA} \cdot \overline{PP} \cdot \overline{MO}$	Close	10	
Close	10				CD
	10	$\overline{CL} \cdot \overline{PA} \cdot \overline{PP} \cdot \overline{MO} \cdot \overline{DR}$	Close	10	
	10	$CL \cdot \overline{PA} \cdot \overline{PP} \cdot \overline{MO} \cdot \overline{DR}$	Closed	00	
	10	PA + PP + MO + DR	Open	01	

4-11 cont'd

Excitation equations:

$$X = PA + PP + MO$$

$$Y_1(t+1) = \bar{Y}_1 \cdot Y_2 \cdot OL + Y_1 \cdot Y_2 + Y_1 \cdot \bar{Y}_2 \cdot \bar{CL} \cdot \bar{X} \cdot \overline{DR}$$

$$Y_2(t+1) = \bar{Y}_1 \cdot \bar{Y}_2 \cdot \overline{LK} \cdot X + \bar{Y}_1 \cdot Y_2 + Y_1 \cdot Y_2 \cdot X + Y_1 \cdot \bar{Y}_2 \cdot (X + DR)$$

Output equations:

$$BT = \bar{Y}_1 \cdot \bar{Y}_2 \cdot LK$$

$$CD = Y_1 \cdot \bar{Y}_2 + \bar{Y}_1 \cdot \bar{Y}_2 \cdot \overline{LK} \cdot \bar{CL} \cdot \bar{X}$$
$$= (Y_1 + \overline{LK} \cdot \bar{CL} \cdot \bar{X}) \cdot \bar{Y}_2$$

$$OD = \bar{Y}_1 \cdot Y_2 + Y_1 \cdot Y_2 \cdot \bar{OL} \cdot X$$
$$= (\bar{Y}_1 + \bar{OL} \cdot X) \cdot Y_2$$

VHDL for Positive-Edge-Triggered D Flip-Flop with Reset

```
-- Positive-Edge-Triggered D Flip-Flop with Reset:
-- VHDL Process Description
library ieee;
use ieee.std_logic_1164.all;
entity dff is
    port ( CLK, RESET, D: in std_logic;
           Q: out std_logic);
end dff;

architecture pet_pr of dff is
-- Implements positive-edge-triggered bit state storage
-- with asynchronous reset.

begin
process ( CLK, RESET )
    begin
        if ( RESET = '1' ) then
            Q <= '0';
        elsif ( CLK'event and CLK = '1' ) then
            Q <= D;
        end if;
    end if;
  end process;
end;
```

VHDL for the Sequence Recognizer

```
-- Sequence Recognizer : VHDL Process Description
library ieee;
use ieee.std_logic_1164.all;
entity seq_rec is
      port ( CLK, RESET, X : in std_logic;
                  Z : out std_logic );
end seq_rec;

architecture process_3 of seq_rec is
    type state_type is (A,B,C,D);
    signal state, next_state : state_type;
begin

-- Process 1 - state_register : implements positive-edge-triggered
-- state storage with asynchronous reset.
   state_register : process ( CLK, RESET )
    begin
      if ( RESET = '1' ) then
           state <= A;
      elsif ( CLK'event and CLK = '1' ) then
           state <= next_state;
      end if;
end process;
```

4-13 cont'd

```
-- Process 2 - next_state_function: implements next state as
-- a function of input X and state.
    next_state_func : process ( X, state )
    begin
        case state is
            when A =>
        if X = '1' then
            next_state <= B;
        else
            next_state <= A;
                end if;
            when B =>
        if X = '1' then
            next_state <= C;
        else
            next_state <= A;
                end if;
            when C =>
        if X = '1' then
            next_state <= C;
        else
            next_state <= D;
                end if;
            when D =>
        if X = '1' then
            next_state <= B;
        else
            next_state <= A;
                end if;
        end case;
end process;
```

4-13 cont'd

```
-- Process 3 - output_function: implements output as function
-- of input X and state.
    output_func: process ( X, state )
    begin
        case state is
            when A =>
          Z <= '0';
        when B =>
          Z <= '0';
        when C =>
          Z <= '0';
            when D =>
        if X = '1' then
          Z <= '1';
        else
          Z <= '0';
            end if;
        end case;
    end process;
end;
```

4-13 cont'd

```
-- Testbench for sequence recognizer example
library ieee;
use ieee.std_logic_1164.all, ieee.std_logic_unsigned.all;

entity seq_rec_testbench is
end seq_rec_testbench;

architecture testbench of seq_rec_testbench is
    signal clock, X, reset, Z : std_logic;
    signal test_sequence : std_logic_vector (0 to 10)
        := "01110101100";

    constant PERIOD : time := 100 ns;

    component seq_rec is
    port ( CLK, RESET, X : in std_logic;
        Z : out std_logic );
    end component;

begin
    u1 : seq_rec port map (clock, reset, X, Z);

    -- This process applies reset and
    -- then applies the test sequence to input X
    apply_inputs : process
    begin
        reset <= '1';
        X <= '0';
```

4-13 cont'd

```
        -- ensure that inputs are applied
        -- away from the active clock edge
        wait for 5* PERIOD/4;
        reset <= '0';
        for i in 0 to 10 loop
            X <= test_sequence(i);
            wait for PERIOD;
        end loop;
        wait;          -- wait forever
    end process;

    -- This process provides the clock pulses
    generate_clock : process
    begin
        clock <= '1';
        wait for PERIOD/2;
        clock <= '0';
        wait for PERIOD/2;
    end process;
end testbench;
```

Verilog for Positive-Edge-Triggered D Flip-Flop with Reset

```
// Positive-Edge-Triggered D Flip-Flop with Reset:
// Verilog Process Description

module dff_v ( CLK, RESET, D, Q );
    input CLK, RESET, D;
    output Q;
    reg Q;

always @ ( posedge CLK or posedge RESET)
begin
   if ( RESET )
      Q <= 0;
   else
      Q <= D;
end
endmodule
```

4-15

Verilog for the Sequence Recognizer

```
// Sequence Recognizer : Verilog Process Description
module seq_rec_v ( CLK, RESET, X, Z );
    input CLK, RESET, X;
    output Z;
    reg [1:0] state, next_state;
    parameter A = 2'b00, B = 2'b01, C = 2'b10, D = 2'b11;
    reg Z;
// state register : implements positive edge-triggered
// state storage with asynchronous reset.
always @ (posedge CLK or posedge RESET)
begin
    if ( RESET )
        state <= A;
    else
        state <= next_state;
end
// te function : implements next state as function
// of X and state
always @ ( X or state )
begin
    case (state)
        A: next_state = X ? B : A;
        B: next_state = X ? C : A;
        C: next_state = X ? C : D;
        D: next_state = X ? B : A;
    endcase
end
```

4-15 cont'd

```
// output function : implements output as function
// of X and state
always @ ( X or state )
begin
    case ( state )
        A: Z = 1'b0;
        B: Z = 1'b0;
        C: Z = 1'b0;
        D: Z = X ? 1'b1 : 1'b0;
    endcase
end
endmodule
```

4-15 cont'd

```
// Testbench for Verilog sequence recognizer
module seq_rec_v_testbench();
    wire Z;
    reg clock, X, reset;

    reg [0:10] test_sequence = 11'b011_1010_1100;
    integer i;
    parameter PERIOD = 100;

    seq_rec_v DUT (clock, reset, X, Z);

// This initial block initializes the clock, applies reset,
// and then applies the test sequence to input X.
    initial
    begin
      reset = 1'b1;
      X = 1'b0;
      // Ensure that inputs are applied
      // away from the active clock edge
      #(5*PERIOD/4);
      reset = 1'b0;
      for (i=0; i<11; i=i+1)
      begin
        X = test_sequence[i];
        #PERIOD;
      end
      // Stop the simulation after all the inputs
      // in the sequence have been applied
        $stop;
    end
```

4-15 cont'd

```
// This always block provides the clock pulses
always
begin
    clock = 1'b1;
    #(PERIOD/2);
    clock = 1'b0;
    #(PERIOD/2);
end
endmodule
```

4-16

Clock Period and Frequency Calculations

- Suppose that all flip-flops used are the same and have $t_{pd} = 0.2ns$ and $t_s = 0.1ns$
- Then the longest path beginning and ending with a flip-flop will be the path with the largest $t_{pd,COMB}$
- Further, suppose that the largest $t_{pd,COMB}$ is 1.3ns and that t_p (clock period) has been set to 1.5ns

$$t_p = t_{slack} + (t_{pd,FF} + t_{pd,COMB} + t_s)$$
$$1.5ns = t_{slack} + (0.2ns + 1.3ns + 0.1ns)$$
$$t_{slack} = -0.1ns$$

∴ this value of t_p is too small

→ In order for t_{slack} to be greater than or equal to zero for the longest path, $t_p \geq t_{p,min} = 1.6ns$

→ The maximum frequency $f_{max} = 1/1.6ns = 625\ MHz$