UNITED STATES
AIR FORCE
ACADEMY

ECE 383 – Embedded Computer Systems II
Lecture 9 – Finite State Machines

UNITED STATES AIR FORCE ACADEMY

Lesson Outline

1. D Flip Flop
2. Finite State Machine
3. FSM Timing
4. The DAISY System
5. HW #6

Reminder: How do we do processes (i.e. Registers) in this class?

Mod 10 Counter – VHDL Cod

```
process(clk)
 begin
     if (rising_edge(clk)) then
         if (reset = '0') then
             processQ <= (others => '0');
             rollSynch <= '0';
     elsif ((processQ < 9) and (ctrl = "01")) then
             processQ <= processQ + 1;
```

Always

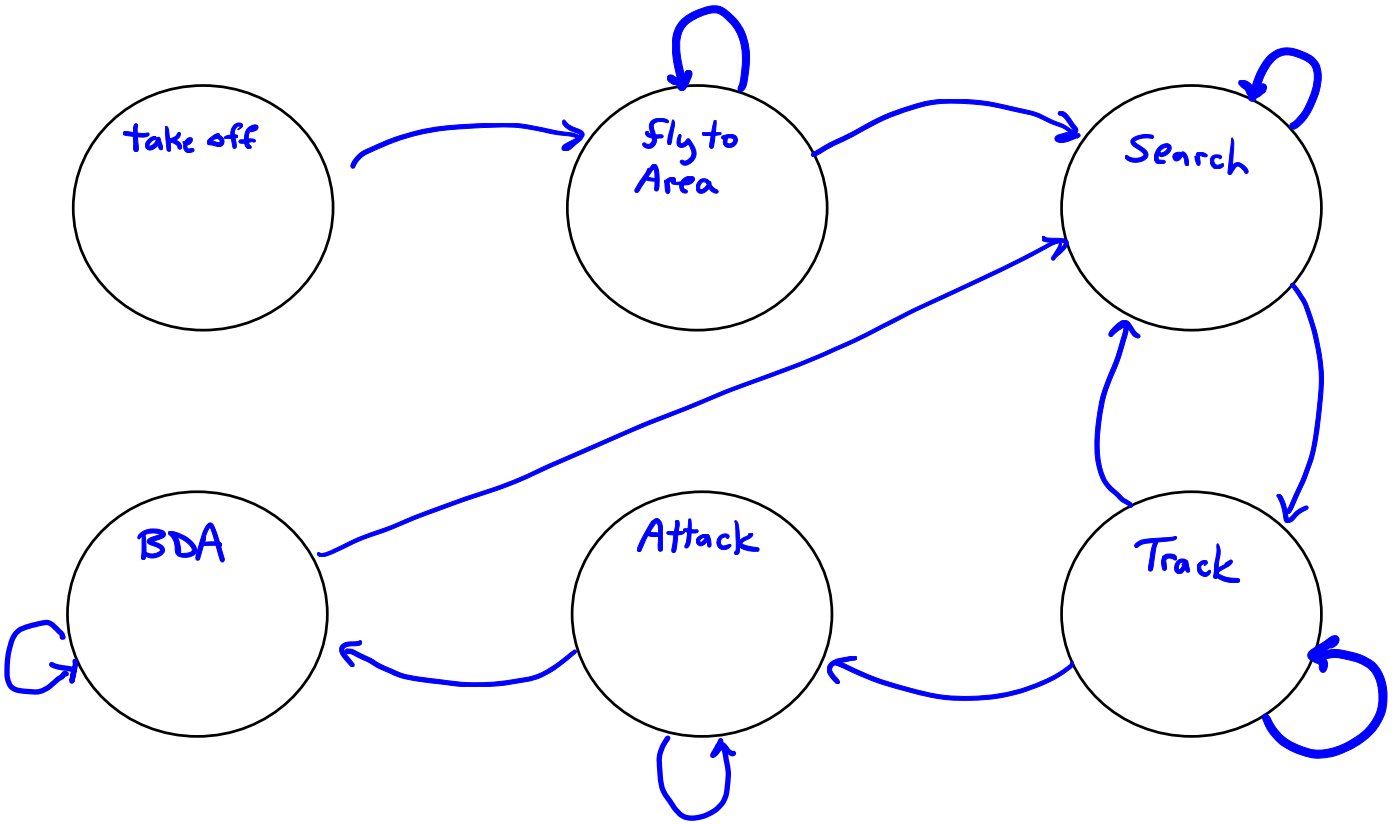
See Lec 4 slide 18 and 20 ⟹

then do your work

UNITED STATES AIR FORCE ACADEMY

State Machine Paradigm

- **Autonomous UAS**



D Flip Flop

Delay

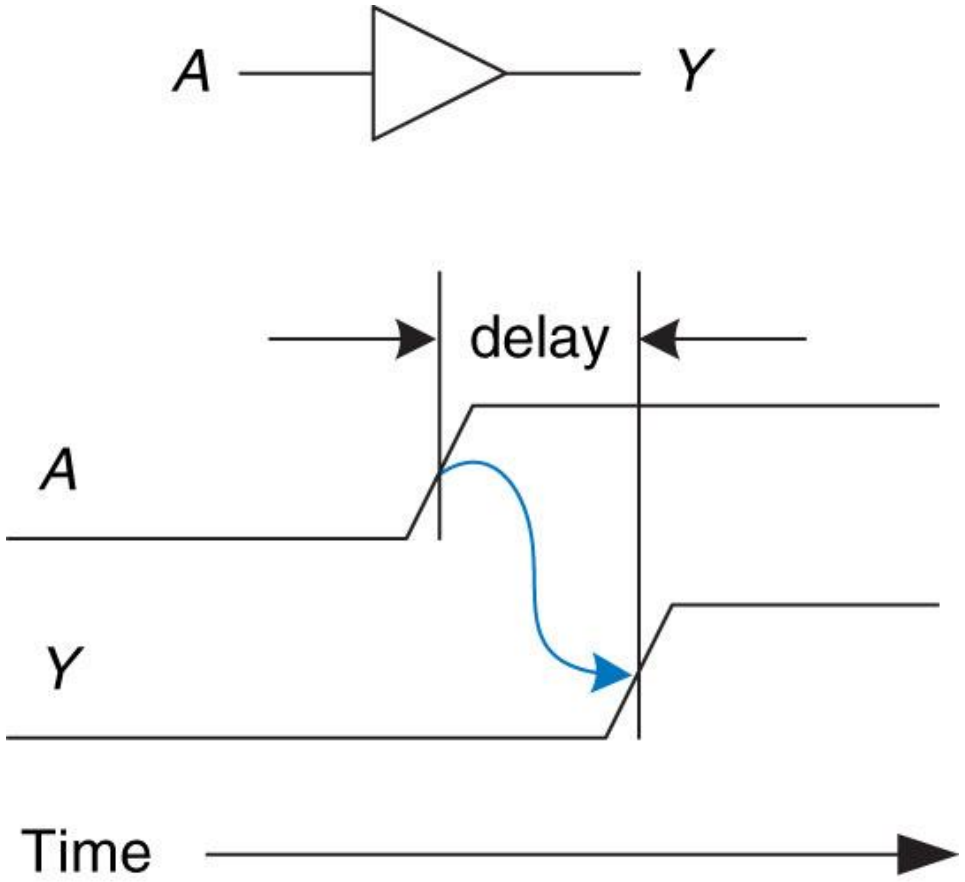


Figure 2.66 Circuit delay

UNITED STATES AIR FORCE ACADEMY

Propagation Delay and Contamination Delay

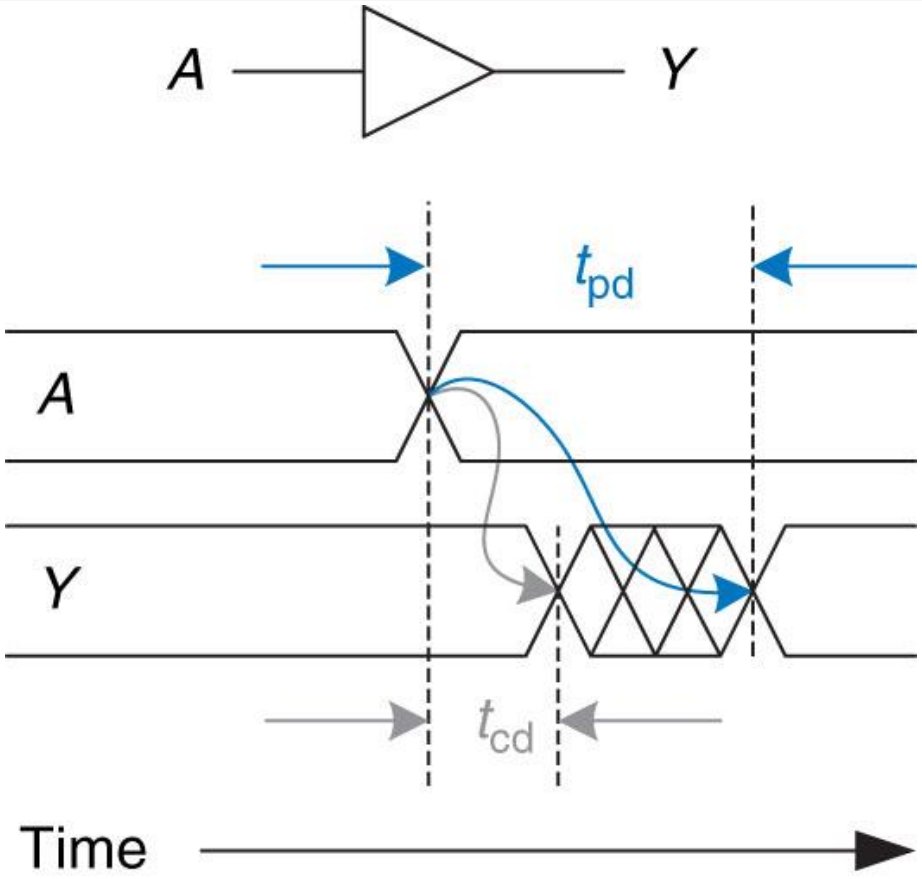


Figure 2.67 Propagation and contamination delay

UNITED STATES AIR FORCE ACADEMY

Critical Path and Shortest Path

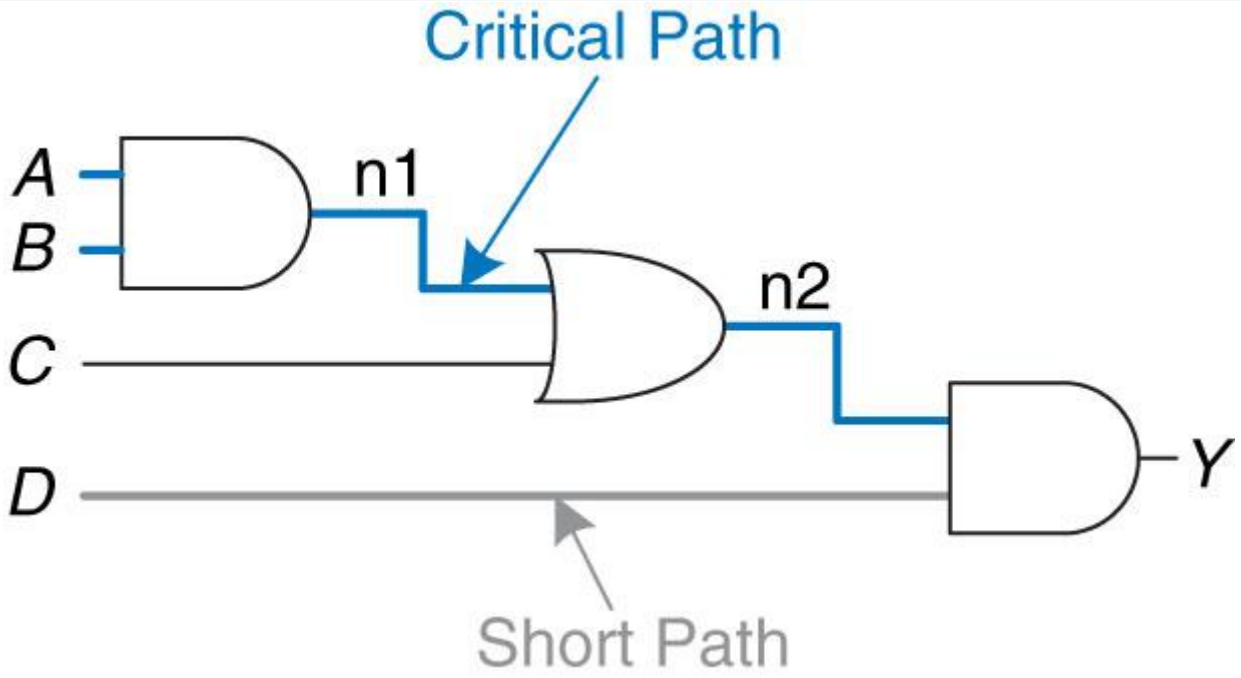


Figure 2.68 Short path and critical path

UNITED STATES AIR FORCE ACADEMY

Critical Path and Shortest Path Waveform

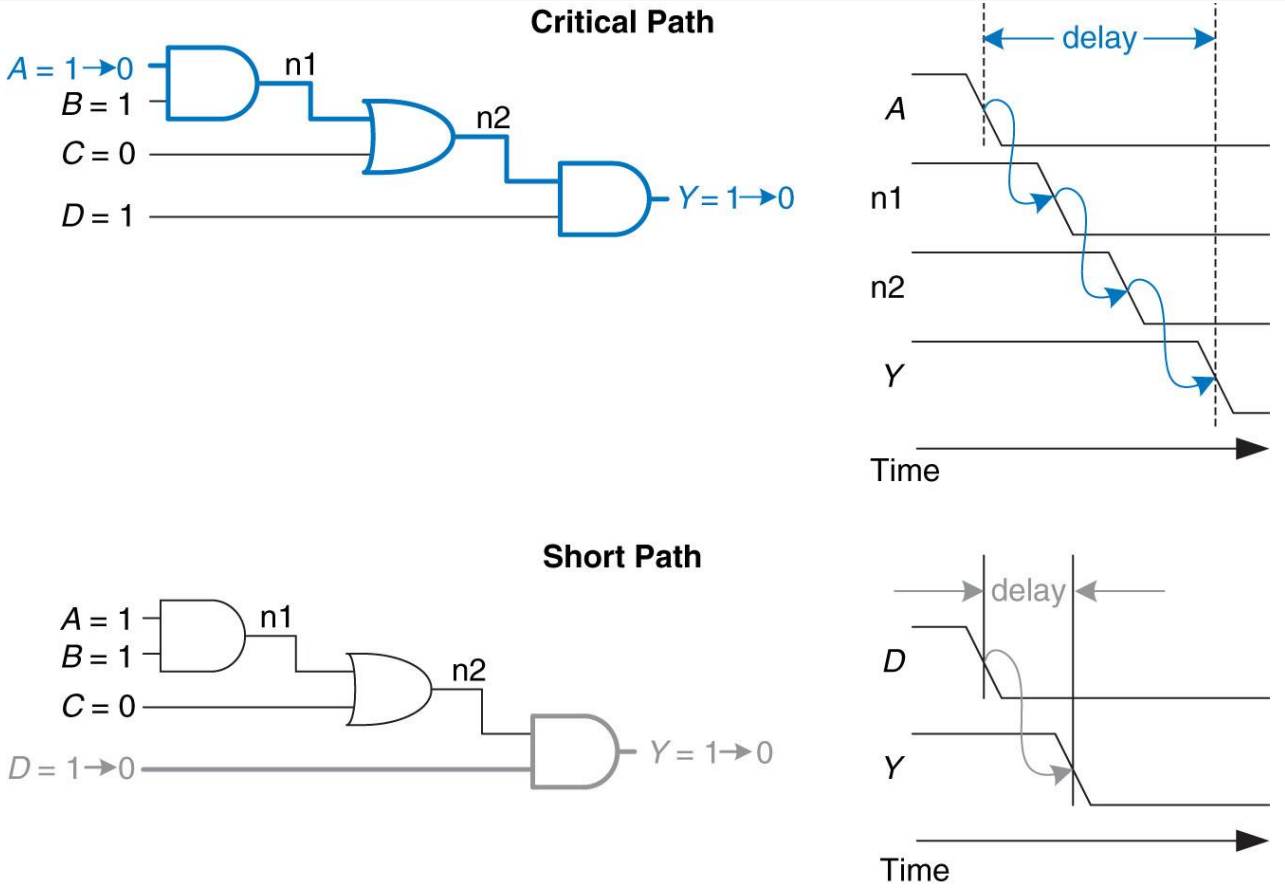
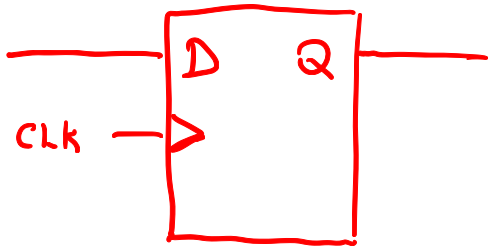


Figure 2.69 Critical and short path waveforms

UNITED STATES AIR FORCE ACADEMY

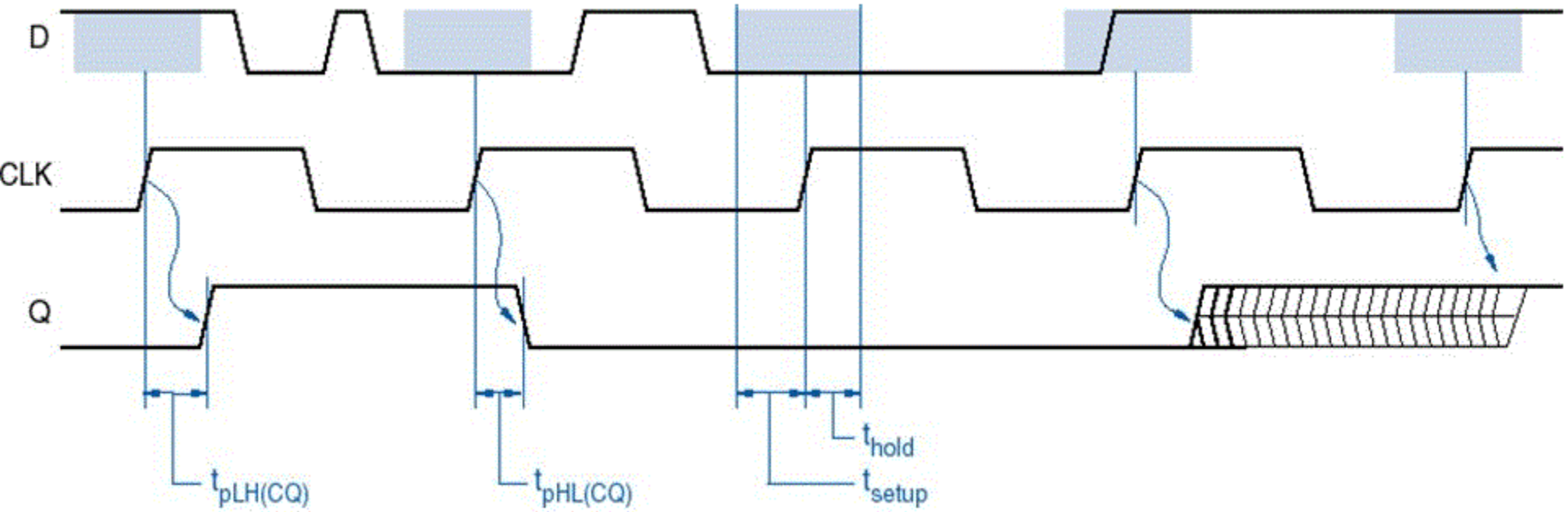


D Flip Flop

- Setup time(t_{su}) - is the amount of time before the rising edge of the clock in which the data inputs must be stable.
- Hold time(t_h) is the amount of time after the rising edge of the clock during which the data input must be stable.
- Propagation Delay(t_p), is the amount of time after the rising edge of the clock required for the new Q value to become valid.
  - It is also known as t_{CQ}, for "time clock to Q", or t_{FF}. These time values are illustrated in the picture below.

UNITED STATES AIR FORCE ACADEMY

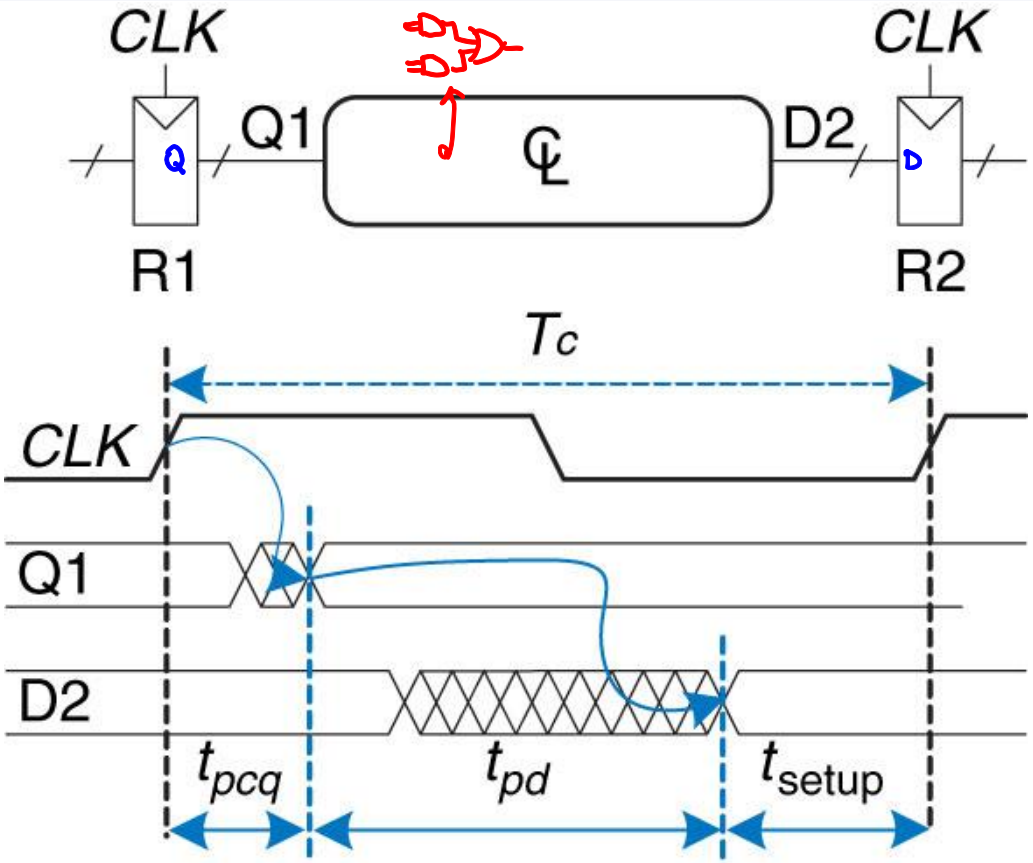
D Flip Flop



UNITED STATES AIR FORCE ACADEMY

RTL

Time Clock to Q Propagation Delay



$f_c =$

where is hold time?

Figure 3.39 Maximum delay for setup time constraint

UNITED STATES AIR FORCE ACADEMY

Time Clock to Q Contamination Delay

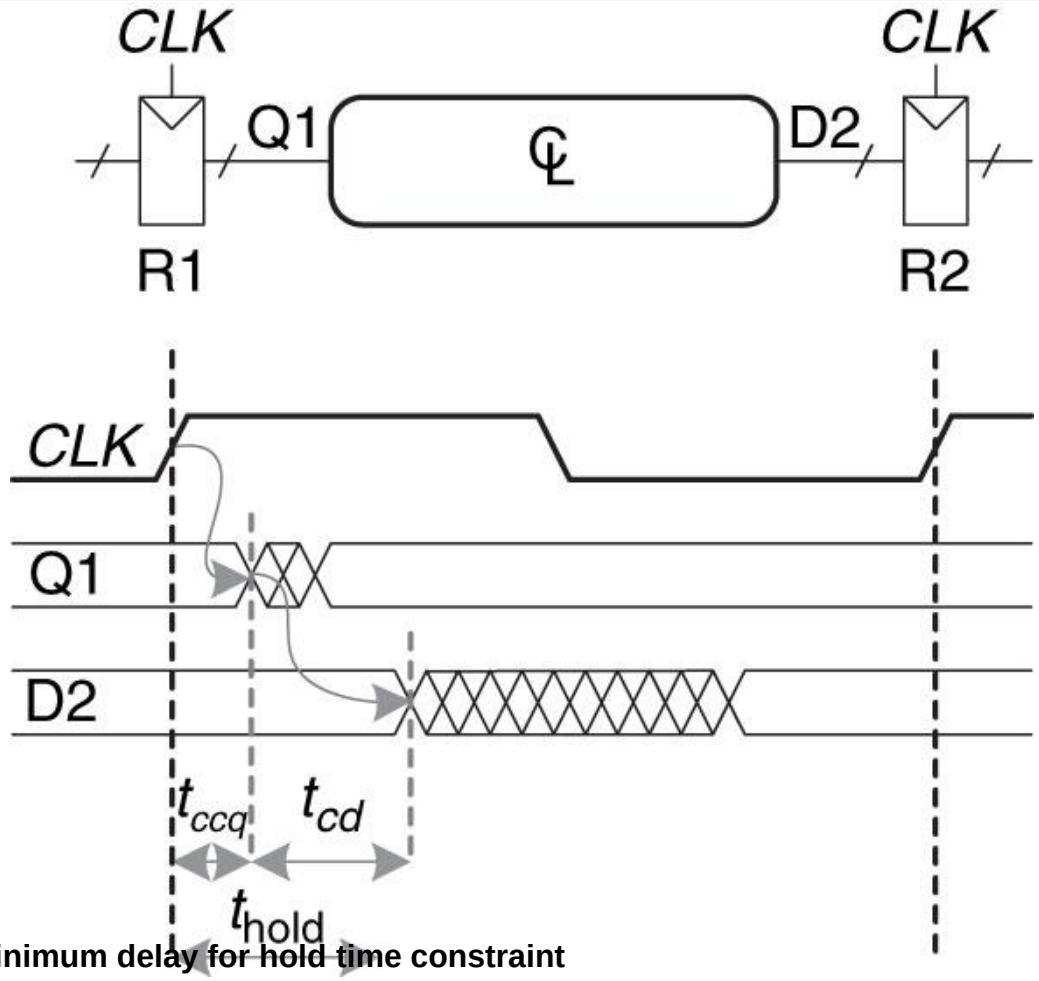
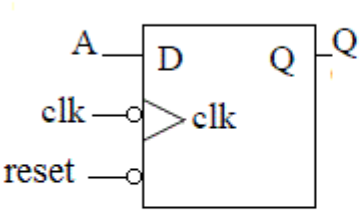
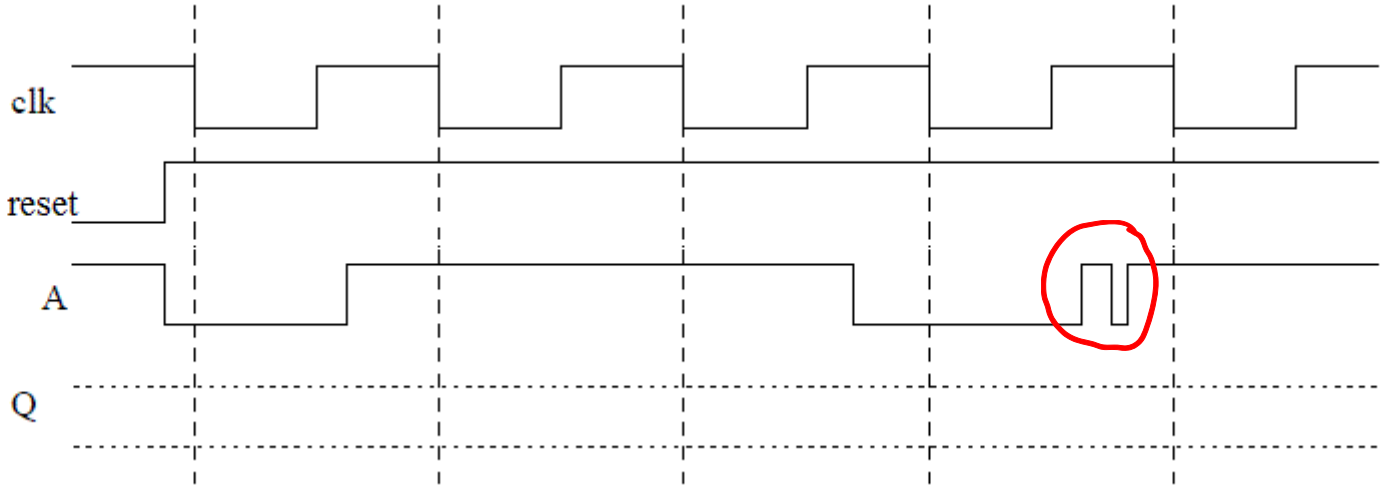


Figure 3.40 Minimum delay for hold time constraint

D Flip Flop

- What type of D Flip Flop is this? _____
- Fill in waveform!





- T_{su}, T_h, T_{pd}

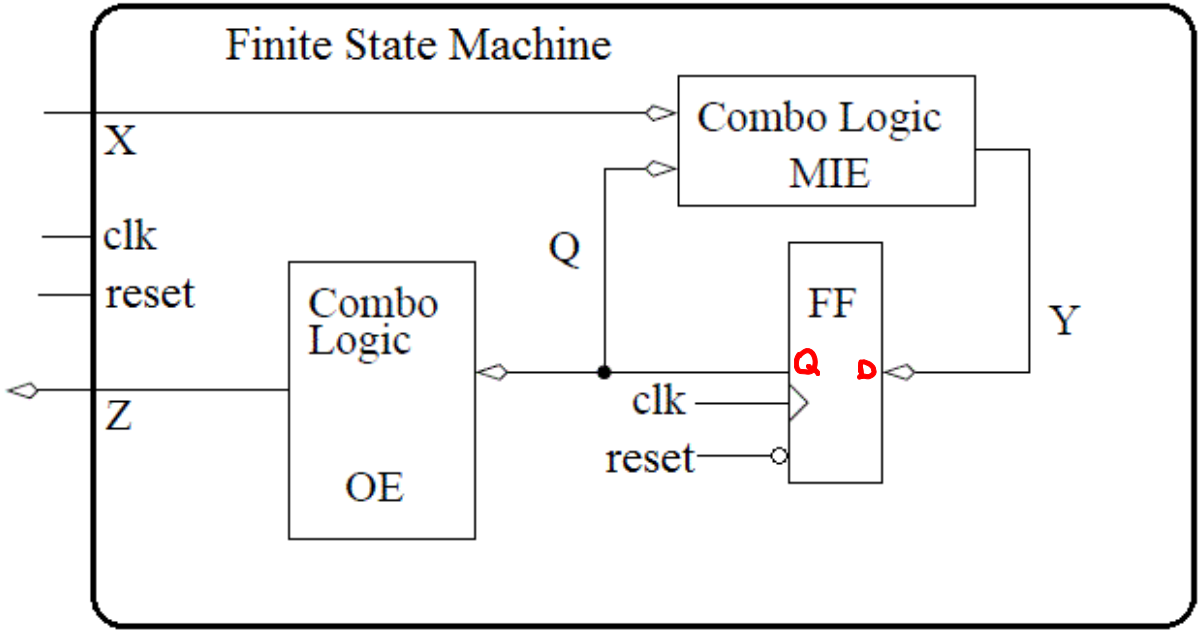
Finite State Machine

Finite State Machine

- Finite State Machine (FSM) – most general form of a sequential circuit, a circuit whose output is a function of input and an internal state

Current State? Next State?
CS NS

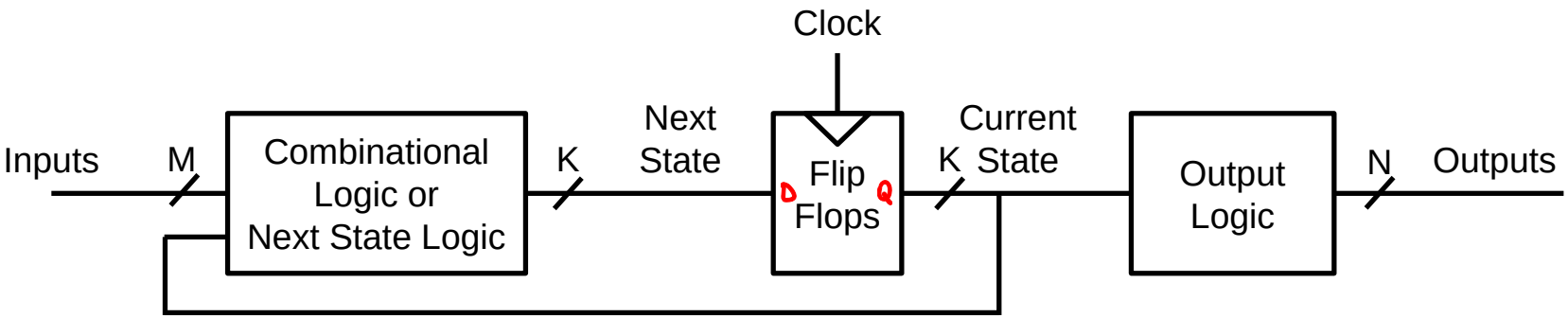
- Moore or Mealy?



UNITED STATES AIR FORCE ACADEMY

Finite State Machines

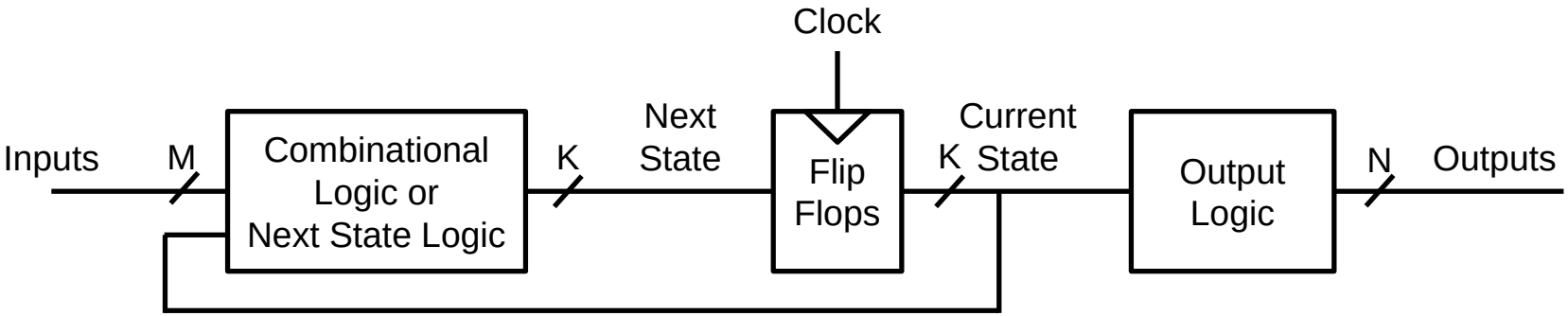
- **FSM – M- inputs, N – Outputs, and K - bits of state**
  - **FSMs have K registers that can be one of a finite number (2^K) unique states**



- **Two types of FSMs?**

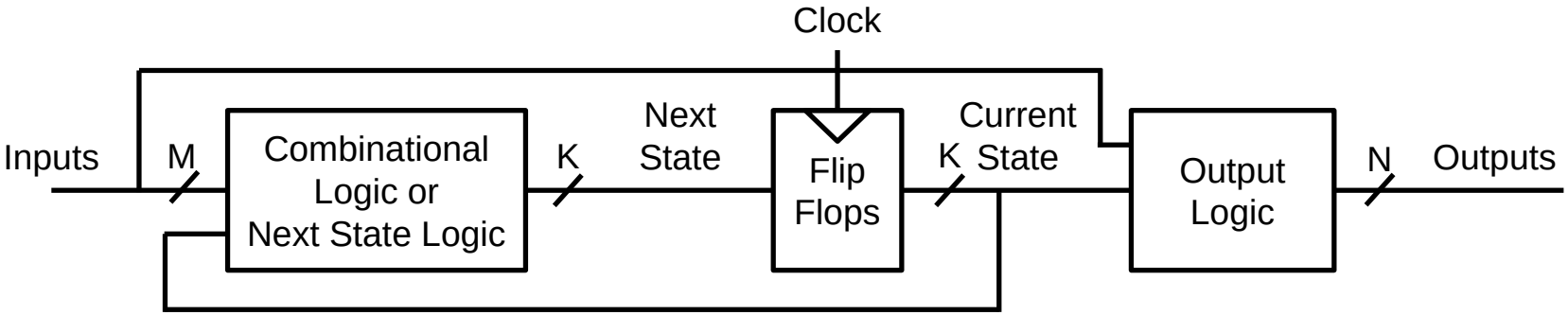
Finite State Machines - Moore Machine

- **Moore Machine – outputs depend only on current state of the machine.**



Finite State Machines - Mealy Machine

- **Mealy Machine – outputs depend on both the current state and current inputs of the machine.**



UNITED STATES AIR FORCE ACADEMY

Finite State Machines - Design

Process

- **How to Design a State Machine:**
  - **0) Description**
  - **1) State Transition Diagram** Step
  - **2) State Transition Table & Output Table** Step
  - **3) Next State and Output State Equations**
  - **4) Schematic** or _____ Step

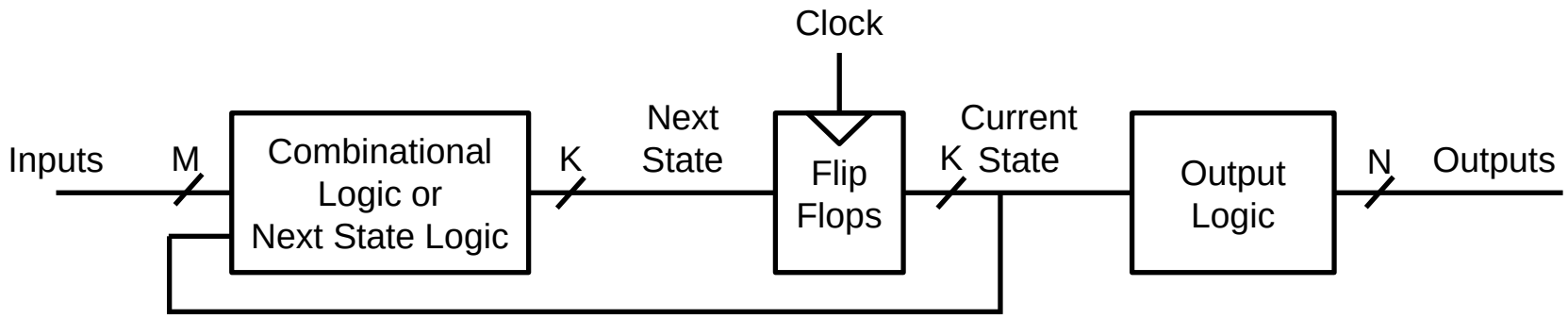
UNITED STATES AIR FORCE ACADEMY

FSM Timing

How fast can I clock the State Machine?

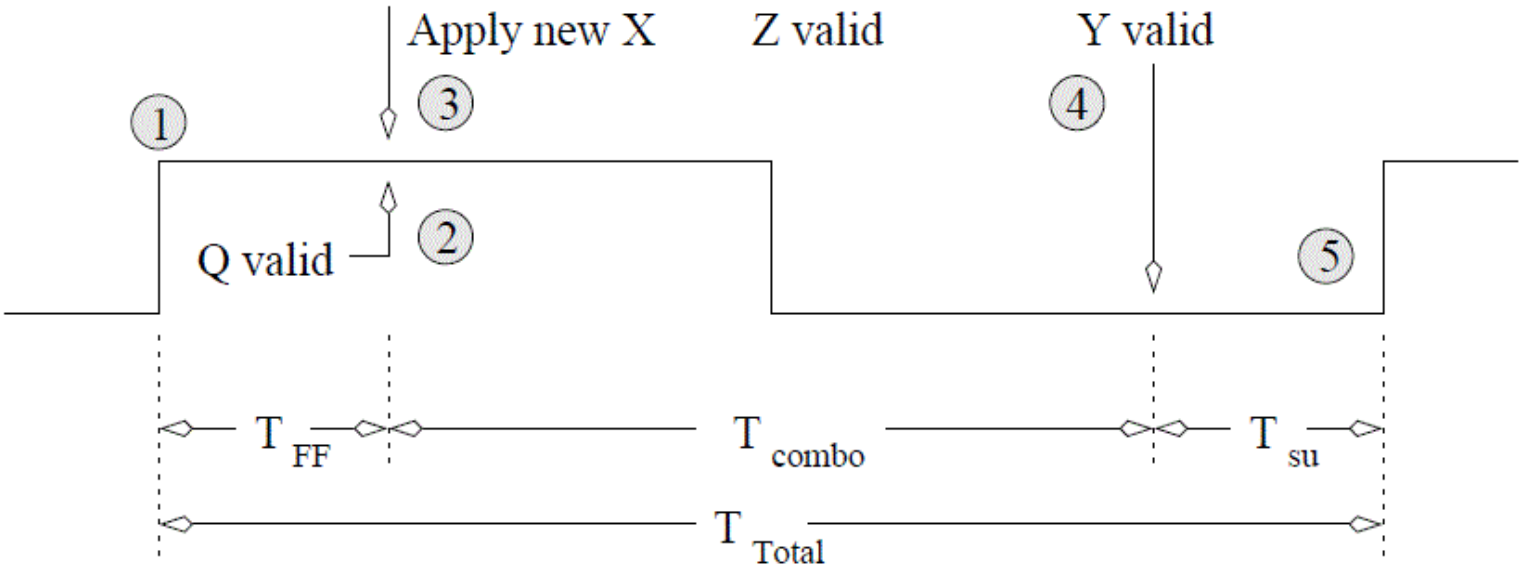
Label 1 to 5 events…

Assume…



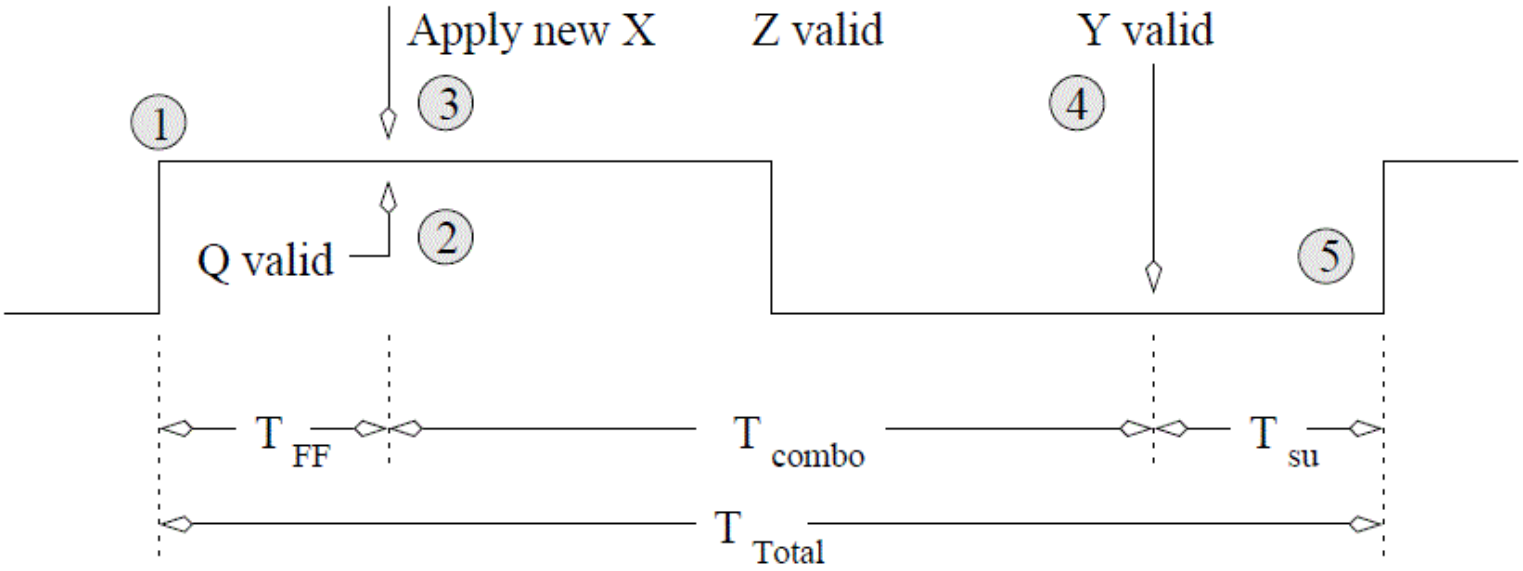
FSM Timing

- **<u>Event 1</u>- Since flip flops sample their inputs on the positive edge of the clock, this point is the beginning of the timing analysis.**



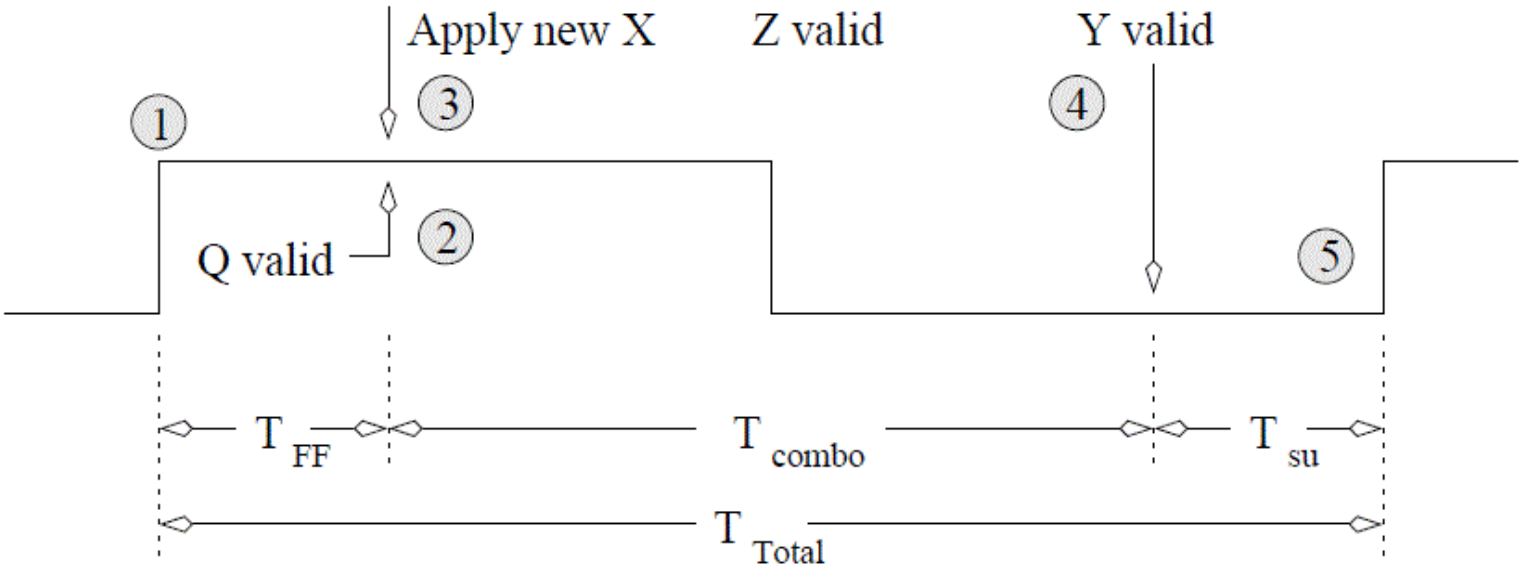
FSM Timing

- **<u>Event 2</u> - The propagation delay of the flip flops means a small delay occurs between the clock edge and the flip flop outputs, Q, becoming valid. This is the called the propagation delay of the flip flop and denoted T_ff in the diagram below.**



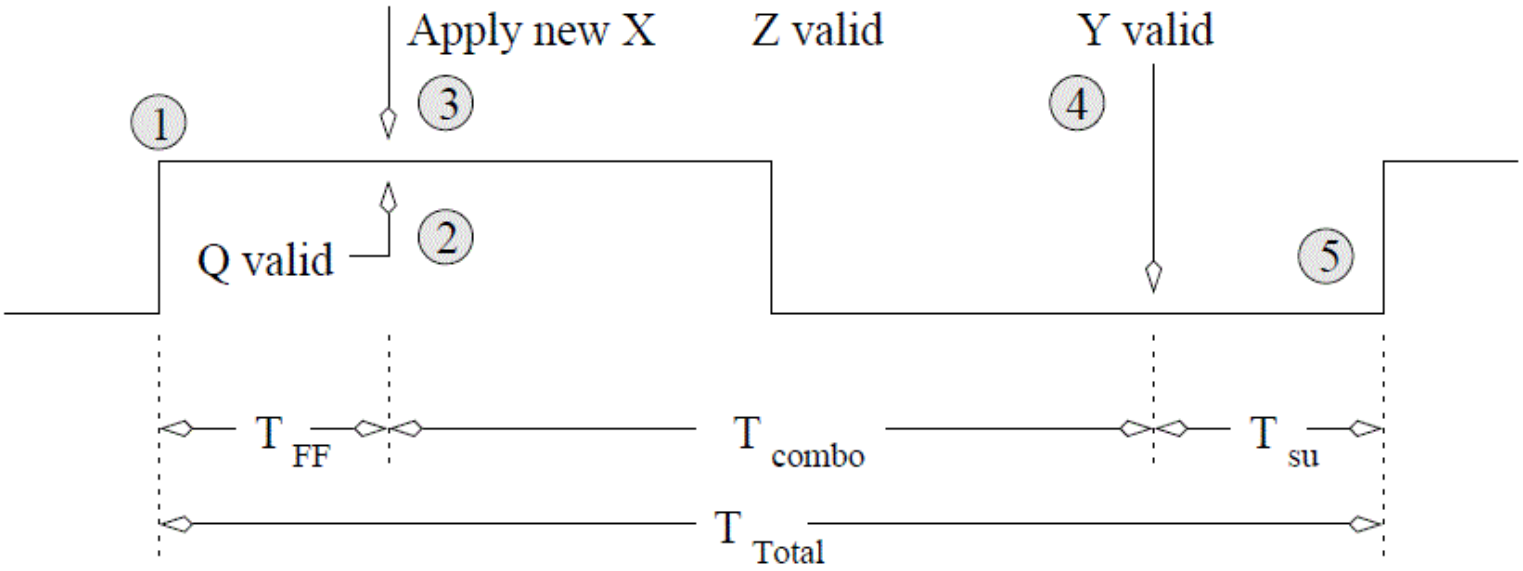
FSM Timing

- **<u>Event 3</u> - In order to maximize the clocking frequency of the FSM, the new inputs, X, to the FSM should be applied at the same moment that the flip flop outputs become valid.**



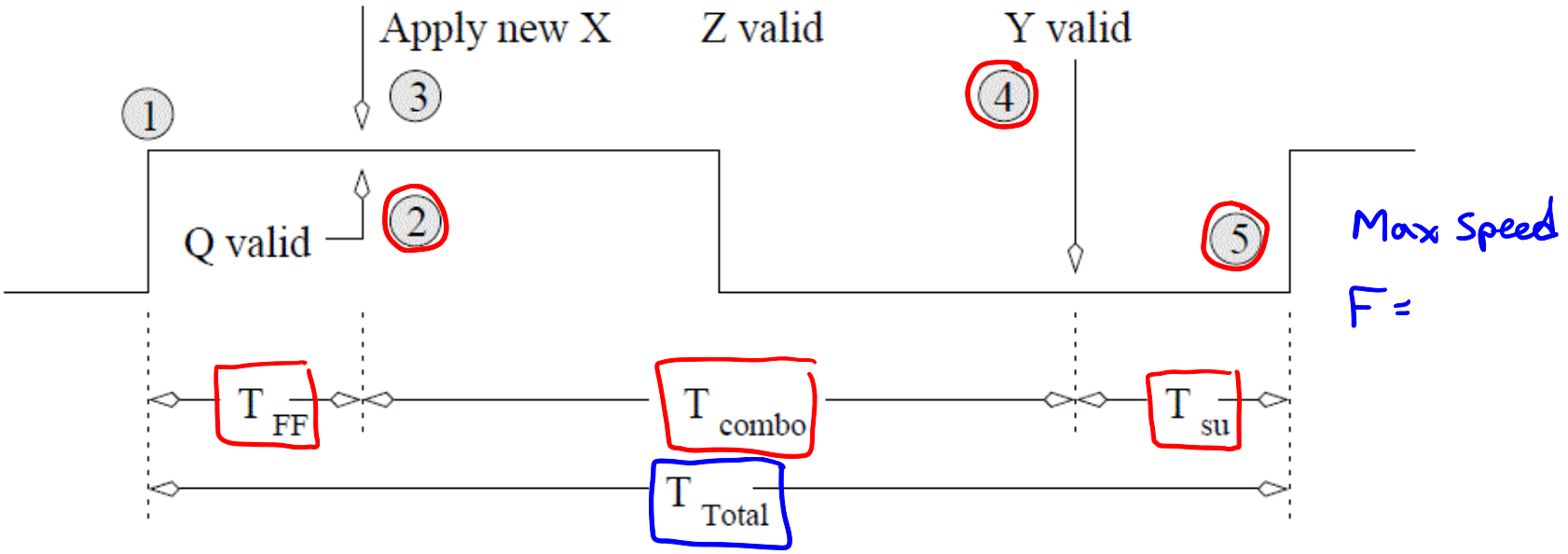
FSM Timing

- **Event 4 - According to Figure above, changing Q and X causes the memory inputs to change (the Y signal above). The delay between the application of the new inputs to the MIE logic and Y becoming valid is the propagation delay of the combination logic, denoted Tcombo.**



FSM Timing

- **Event 5 - When the Y values are valid, a small delay occurs while the flip flops register their new inputs, denoted Tsu. After this setup time, the FSM is ready for another clock edge.**



- **If our FPGA is guarantied to be able to handle state machines clocked at F = 100 MHz, what is T?**

$T =$

Dr Coulston…

Lets bring in the cows.

The DAISY System

UNITED STATES AIR FORCE ACADEMY

The DAISY System

- **D**airy **A**utomated **I**nformation **Sy**stem, or DAISY for short
- **Word Statement** Cows have a RFID tag attached to their collars. When the cow passes through the cattle chute on their way into the barn, a RFID reader reads the unique ID stored on the RFID tag and logs the cow into the barn.
  - The RFID system outputs a single bit: a 1 means the system has read an RFID tag and has successfully checked a cow back into the barn; a 0 means the RFID system is either still processing a tag or is not currently reading a tag.
  - The RFID system outputs a single bit:
    - Logic 1 – Cow Checked In
    - Logic 0 – Cow Not Processed

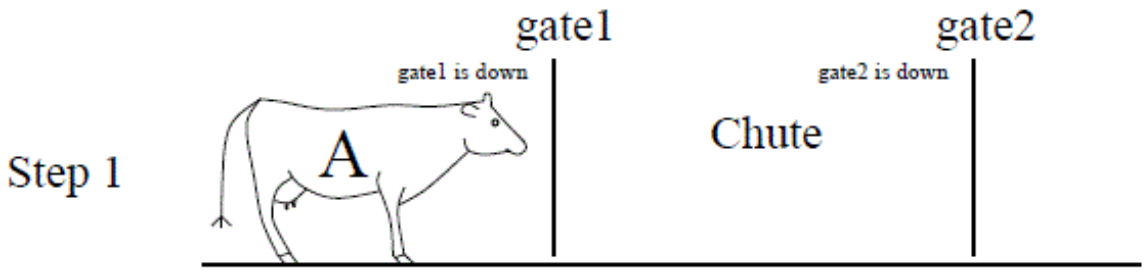
UNITED STATES AIR FORCE ACADEMY

The DAISY System

- In order to ensure each cow is scanned, the flow of cows into the barn is controlled by two gates at either end of the chute. Each gate is controlled by a single bit. To lift a gate, this input must be held at logic 1; to lower a gate, the input must be held at a logic 0. The sequence of raising and lowering the gates in order to control the flow of cows.
  - Flow of cows is controlled by two gates
    - Logic 1 – To lift a gate
    - Logic 0 – To lower a gate

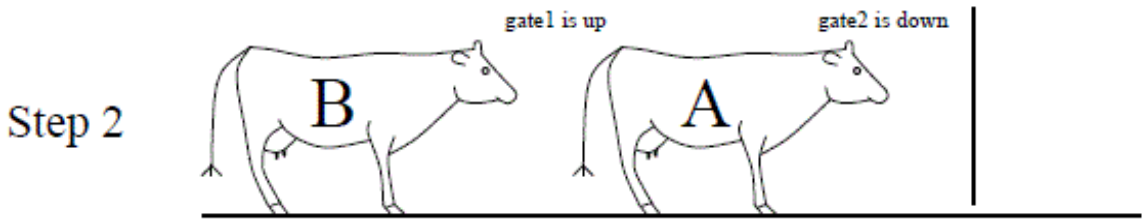
The DAISY System

- Step 1 - Gate1 is lifted allowing cow A to enter the chute.



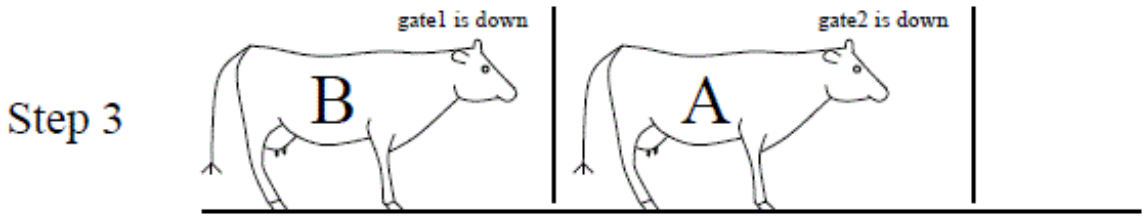
The DAISY System

- Step 2 - The DAISY system has detected cow A is in the chute and closes gate1.



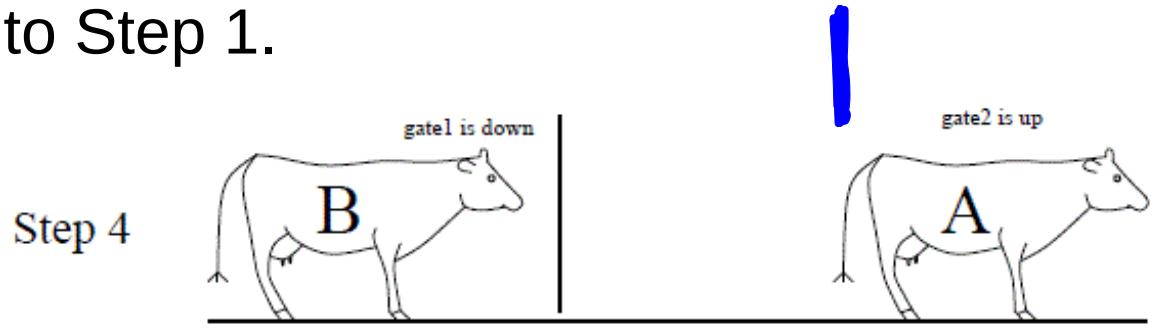
The DAISY System

- Step 3 - The cow waits in the closed off chute until the RFID reader signals that it has read the tag and checked in cow A.



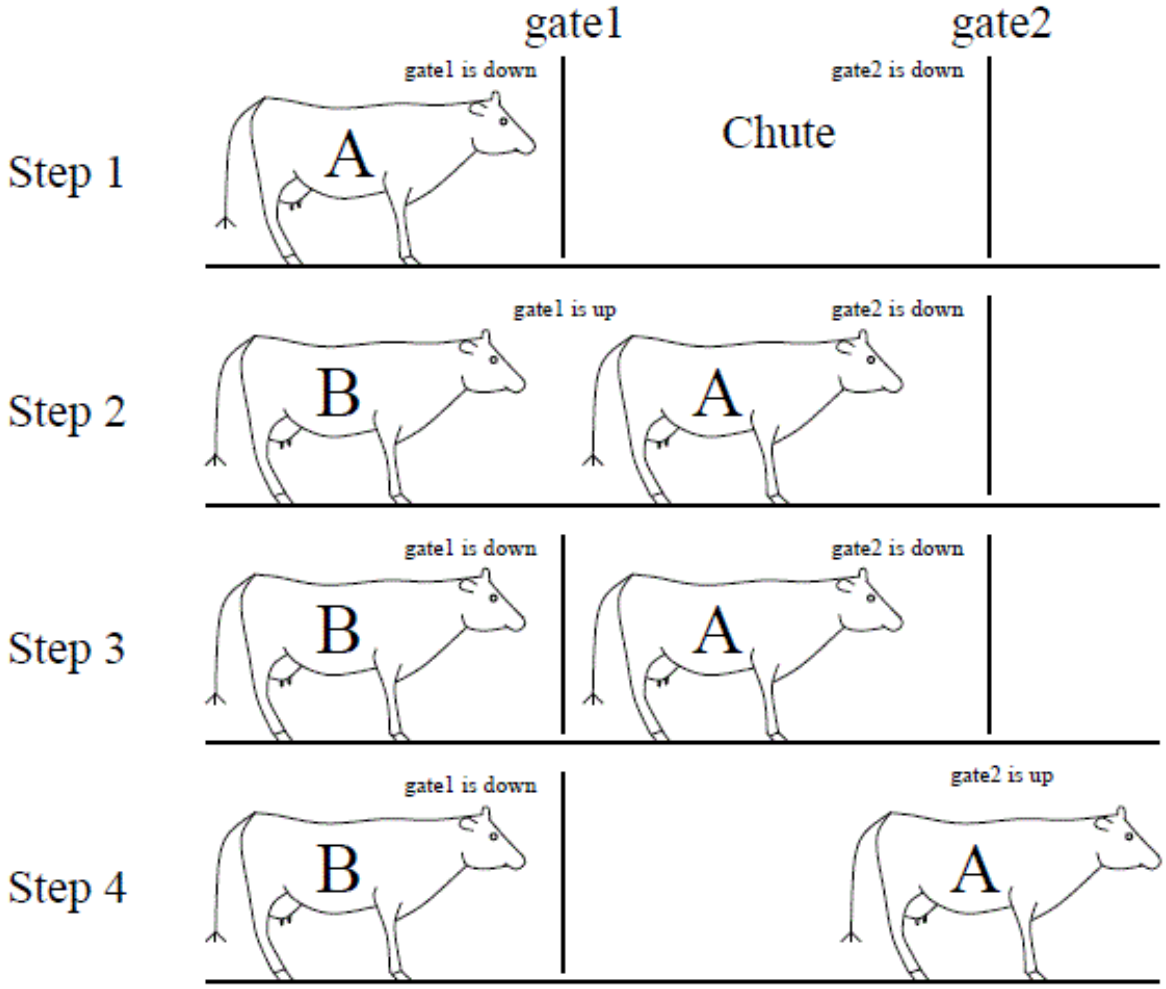
The DAISY System

- Step 4 - Gate2 is raised allowing cow A to leave. If the cow takes more than 30 seconds to leave, then the cow is "goosed" by a three-second burst of compressed air. An air bust is repeated at 30-second intervals until the cow leaves the chute. At any time when the cow leaves the chute, Gate 2 is closed and the system transitions back to Step 1.



UNITED STATES AIR FORCE ACADEMY

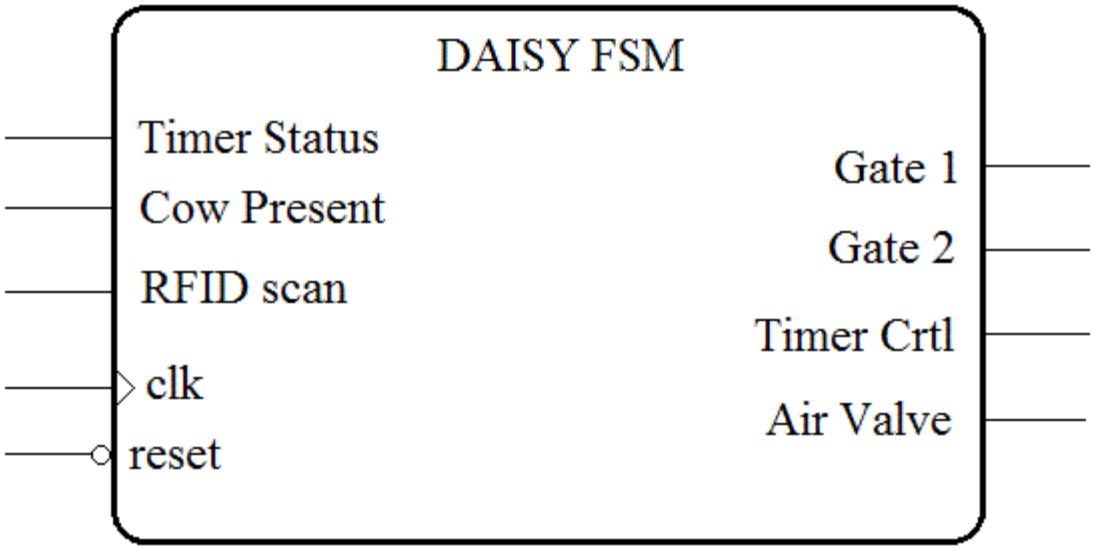
The DAISY System



States?

The DAISY System

- DAISY FSM Entity



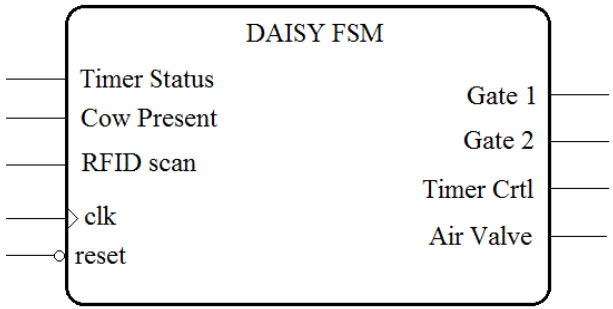
The DAISY System

- Inputs to Daisy

RFID Scanner = r	Cow Present = c	Timer Status = t
1 - Cow checked in	1 - cow present	1 - timer up
0 - Cow not processed	0 - no cow	0 - timer running

- Outputs to Daisy

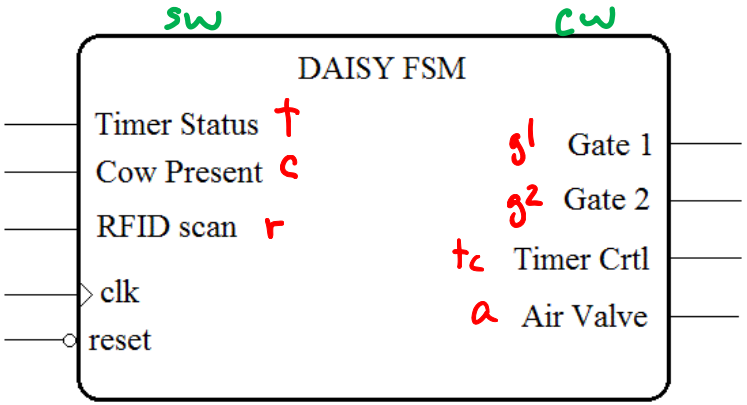
Gate1	Gate2	Timer Control	Air Valve
1-gate up	1-gate up	00 Stop timer	0 closed
0-gate down	0-gate down	01 Set to 30 secs	1 open
		10 Set to 3 secs	
		11 Run timer	



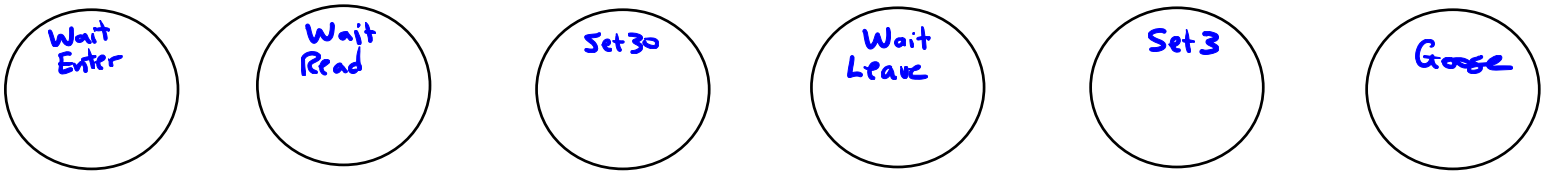
Step 1

The DAISY System

- State Diagram

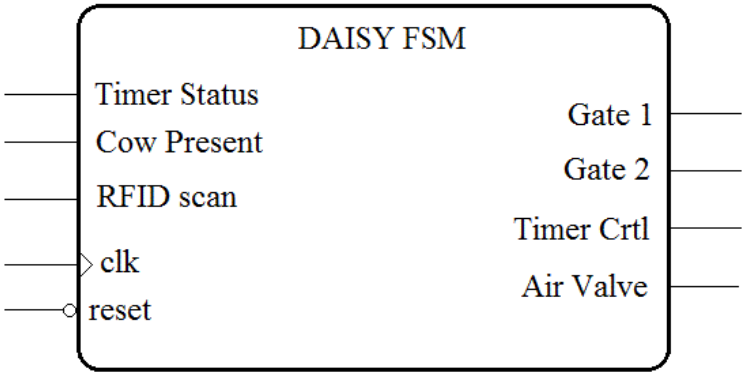


1. Open gate1
2. Wait for cow to enter chute
3. Close gate1
4. Wait for RFID to read cow
5. Open gate2
6. Wait for cow to leave
7. If 30 seconds has transpired, then "goose" cow; goto Step 6
8. Else if the cow has left, then close gate2; goto Step 1

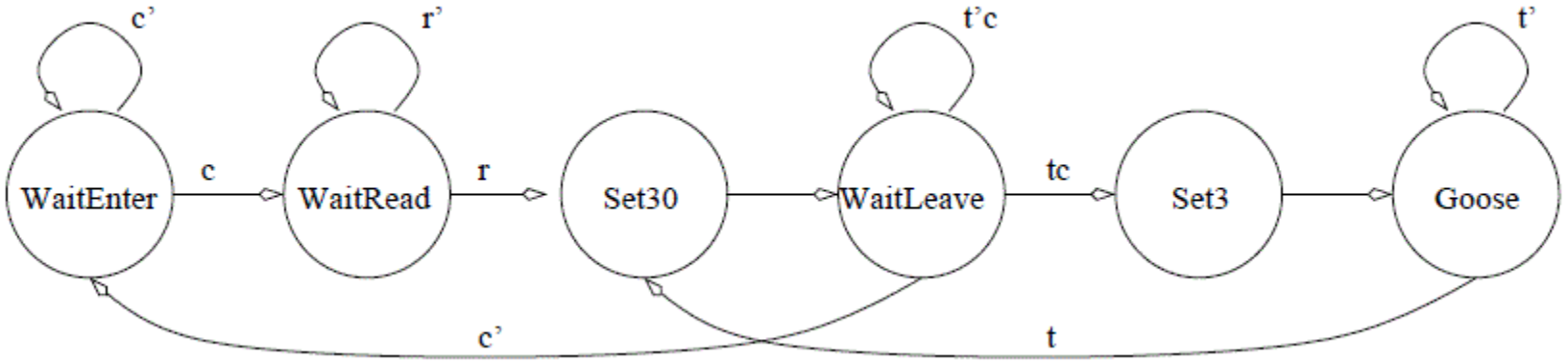


The DAISY System

- State Diagram



1. Open gate1
2. Wait for cow to enter chute
3. Close gate1
4. Wait for RFID to read cow
5. Open gate2
6. Wait for cow to leave
7. If 30 seconds has transpired, then "goose" cow; goto Step 6
8. Else if the cow has left, then close gate2; goto Step 1



The DAISY System

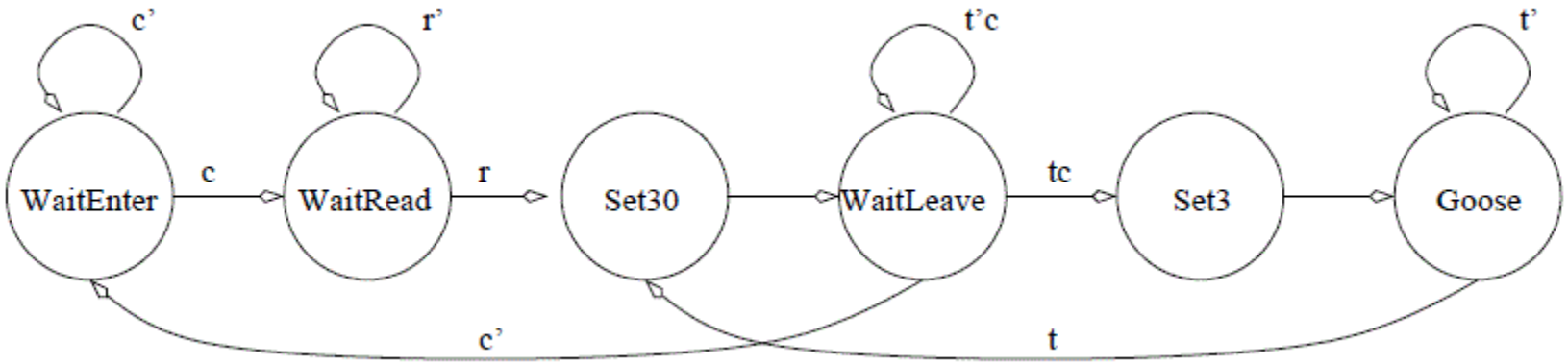
- Memory Input Equations

6 states

State	Code
WaitEnter	000
WaitRead	001
Set30	010
WaitLeave	011
Set3	100
Goose	101

Ghost States?

- How many Flip Flops do we need?



The DAISY System

Next State Equations?

- Solve for three (one for each flip flop) 6-input Boolean Expressions
  - Using **espresso**

Nope: Don't need to simplify equations!

6 Input K-MAP?

Current State | Next State

$Q_2 Q_1 Q_0$ r ct | $D_2 D_1 D_0$

YIKES!

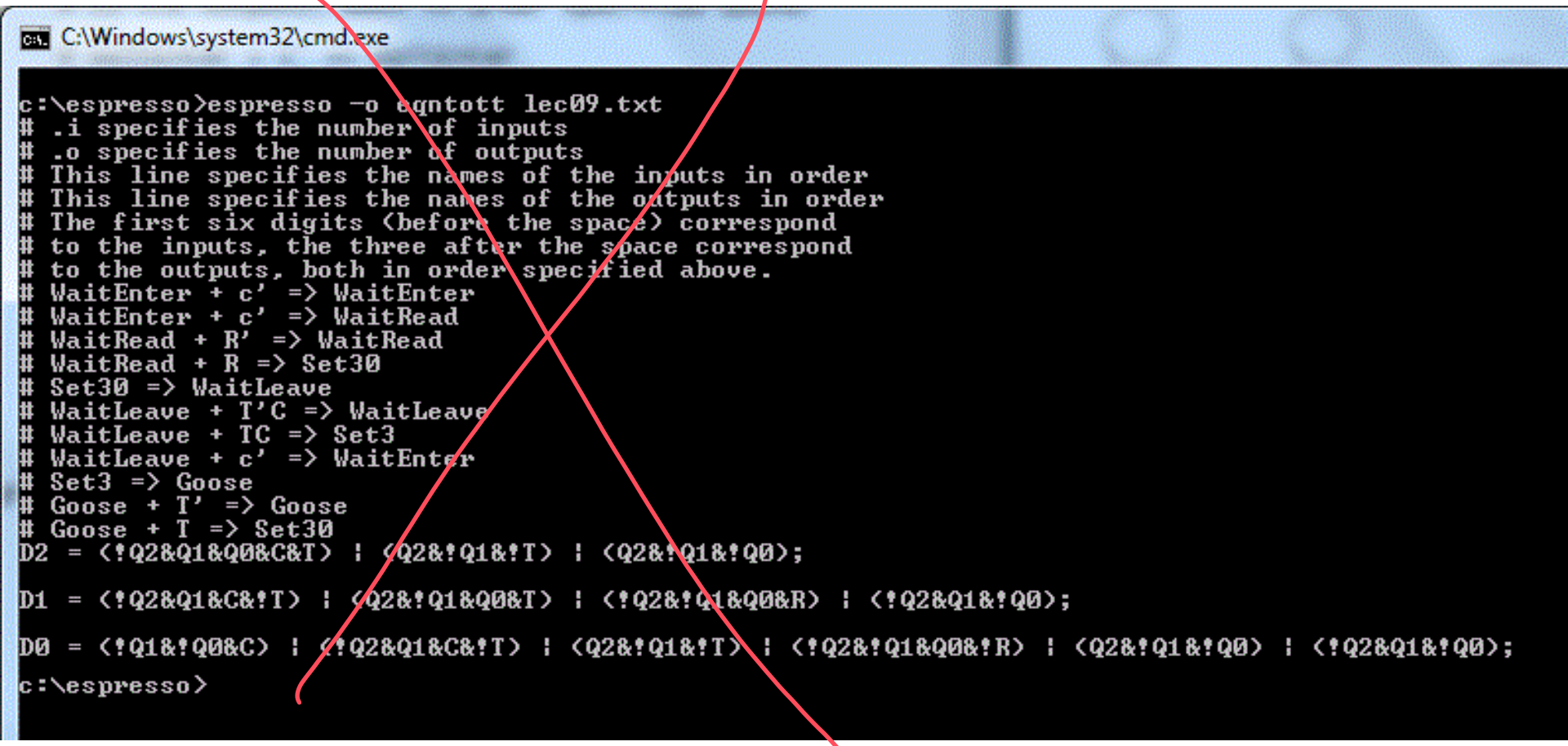
The DAISY System – Expresso input file

```
.i 6 # .i specifies the number of inputs
.o 3 # .o specifies the number of outputs
.ilb Q2 Q1 Q0 R C T # This line specifies the names of the inputs in order
.ob D2 D1 D0        # This line specifies the names of the outputs in order
                    # The first six digits (before the space) correspond
                    # to the inputs, the three after the space correspond
                    # to the outputs, both in order specified above.
000 -0- 000 # WaitEnter + c' => WaitEnter
000 -1- 001 # WaitEnter + c => WaitRead
001 0-- 001 # WaitRead + R' => WaitRead
001 1-- 010 # WaitRead + R => Set30
010 --- 011 # Set30 => WaitLeave
011 -10 011 # WaitLeave + T'C => WaitLeave
011 -11 100 # WaitLeave + TC => Set3
011 -0- 000 # WaitLeave + c' => WaitEnter
100 --- 101 # Set3 => Goose
101 --0 101 # Goose + T' => Goose
101 --1 010 # Goose + T => Set30 .e # Signifies the end of the file.
```

Not Needed Anymore

UNITED STATES AIR FORCE ACADEMY

The DAISY System – Expresso Output



UNITED STATES AIR FORCE ACADEMY

VHDL states are Enumerated Type

The DAISY System – One's Hot Encoding

- **Using a One's Hot Encoding MIEs:**

State	Code
WaitEnter	000001
WaitRead	000010
Set30	000100
WaitLeave	001000
Set3	010000
Goose	100000

- **D_WaitEnter = Q_WaitEnter * c' + Q_WaitLeave * c'**
- **D_WaitRead = Q_WaitRead * r' + Q_WaitEnter * c**
- **D_Set30 = Q_WaitRead * r + Q_Goose * t**
- **D_WaitLeave = Q_Set30 + Q_WaitLeave * t' * c**
- **D_Set3 = Q_WaitLeave * t * c**
- **D_Goose = Q_Goose * t' + Q_Set3**

Not Needed anymore! Just code transitions directly from State Diagram

UNITED STATES AIR FORCE ACADEMY

Step 2

The DAISY System – Output Equations

- **The first step in generating the output equations is to build a control word table - a table listing, for each state, its output.**

State	Gate1	Gate2	Timer	Control Air
	1-gate up	1-gate up	00 Stop timer	0 - closed
	0-gate down	0-gate down	01 Set to 30 secs	1 - open
			10 Set to 3 secs	
			11 Run timer	
WaitEnter				
WaitRead				
Set30				
WaitLeave				
Set3				
Goose				

UNITED STATES AIR FORCE ACADEMY

The DAISY System – Output Equations

- **The first step in generating the output equations is to build a control word table - a table listing, for each state, its output.**

State	Gate1	Gate2	Timer	Control Air
	1-gate up	1-gate up	00 Stop timer	0 - closed
	0-gate down	0-gate down	01 Set to 30 secs	1 - open
			10 Set to 3 secs	
			11 Run timer	
WaitEnter	1	0	00	0
WaitRead	0	0	00	0
Set30	0	1	01	0
WaitLeave	0	1	00	0
Set3	0	1	10	0
Goose	0	1	11	1

The DAISY System – Output Equations

- Z_Gate1 <= Q_WaitEnter;
- Z_Gate2 <= Q_Set30 + Q_WaitLeave + Q_Set3 + Q_Goose
- Z_Timer_1 <= Q_Set3 + Q+Goose
- Z_Timer_0 <= Q_Set30 + Q_Goose
- Z_Air = Q_Goose

Not Needed anymore

Just code output table in a CSA.

STEP 3: Code VHDL

```
1   ----------------------------------------------------------------------
2   -- Name:     Chris Coulston
3   -- Date:     Jan 28, 2015
4   -- File:     lec09.vhdl
5   -- Event:    Lecture 9
6   -- Crs:      ECE 383
7   ----------------------------------------------------------------------
8   library IEEE;
9   use IEEE.STD_LOGIC_1164.ALL;
10  use IEEE.NUMERIC_STD.ALL;
11
12  entity lec09 is
13        Port(  clk: in  STD_LOGIC;
14               reset : in  STD_LOGIC;
15               sw: in STD_LOGIC_VECTOR(2 downto 0);
16               cw: out STD_LOGIC_VECTOR(4 downto 0));
17  end lec09;
18
19  architecture behavior of lec09 is
20
21      type state_type is (WaitEnter, WaitRead, Set30, WaitLeave, Set3, Goose);
22      signal state: state_type;
23
24      constant rfid: integer := 2;        -- helps keep status bits straight
25      constant cow: integer := 1;
26      constant timer: integer := 0;
27
28  begin
29    1 state_process: process(clk,reset)
30        begin
31      2 if (rising_edge(clk)) then
32          3 if (reset = '0') then
33              state <= WaitEnter;
34            else
35              case state is
36                  when WaitEnter =>
37                     if (sw(cow) = '1') then state <= WaitRead; end if;
38                  when WaitRead =>
39                     if (sw(rfid) = '1') then state <= Set30; end if;
40                  when Set30 =>
41                     state <= WaitLeave;
42                  when WaitLeave =>
43                     if (sw(cow) = '0') then state <= WaitEnter;
44                     elsif (sw(timer) = '1' and sw(cow) = '1') then state <= Set3; end if;
45                  when Set3 =>
46                     state <= Goose;
47                  when Goose =>
48                     if (sw(timer) = '1') then state <= Set30; end if;
49              end case;
50          end if;
51      end if;
52    end process;
53
54    cw <=     "10000" when state = WaitEnter else
55              "00000" when state = WaitRead else
56              "01010" when state = Set30 else
57              "01110" when state = WaitLeave else
58              "01100" when state = Set3 else
59              "01111"; -- when state = Goose;
60
61  end behavior;
62
63
```

383 Statemachine Style different from 281!

what if stay in a state?

STATE TRANSITIONS

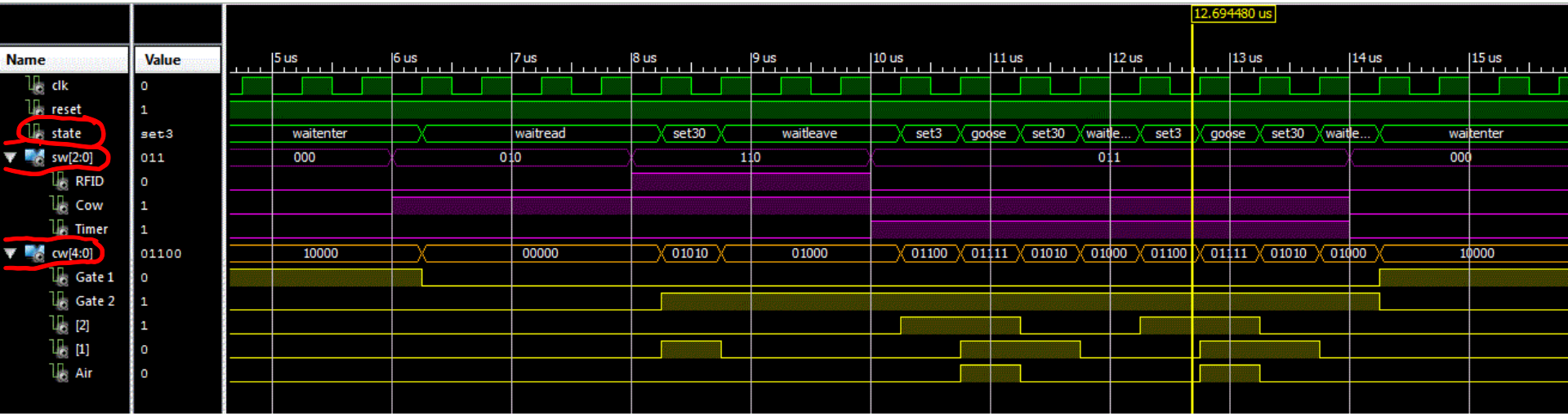
Moore Machine

OUTPUTS in CSA

← Need Doc to define CWs

UNITED STATES AIR FORCE ACADEMY

The DAISY System – VHDL



UNITED STATES AIR FORCE ACADEMY

Homework #6 ???