CEG 3320 - Digital System Design

Instructor: Travis Doom, Ph.D.
331 Russ Engineering Center
775-5105
travis.doom@wright.edu
http://www.wright.edu/~travis.doom

Lecture slides created by T. Doom for Wright State University's course in Digital System Design. Some slides contain fair use images or material used with permission from textbooks and slides by R. Haggard, F. Vahid, Y. Patt, J. Wakerly, M. Mano, and other sources.

Review from previous modules:
#1s greater than 2 device

Module III:
Sequential Analysis and Design

State Devices

Parameterized Boolean Algebra

Sequential Function Analysis

State Diagrams

Sequential Timing Analysis

Design with State Diagrams

Sequential Synthesis

Introduction to Sequential Devices

State Memory

Bi-stable elements

Latches

Flip-flops

Parameterized Boolean Algebra

Timing Characteristics

Logic Devices

- Logic devices divide into two major types:

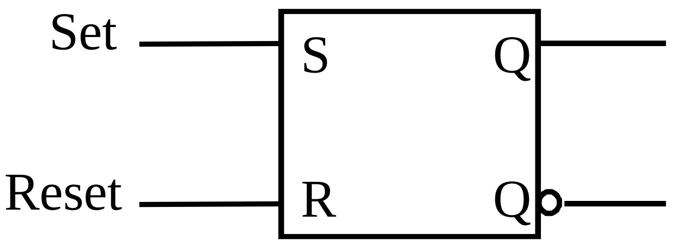
- Combinational Logic
  - Current output depends on current input only
  - Examples: gates, decoders, multiplexors (MUXs), ALUs
  - Familiarity with combinational logic is a course prerequisite

- Sequential Logic
  - Current output depends on past inputs as well as current input
  - Thus has a memory (usually called the state)
  - Examples: latches, flip-flops, state machines, counters, shift registers

How can we ‘store’ a bit over time?

S-R Latch

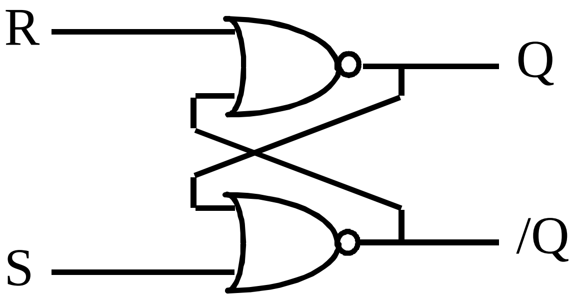
Symbol



Function Table

	S	R	Q	/Q
Hold	0	0	Last Q	Last /Q
Reset	0	1	0	1
Set	1	0	1	0
ILLEGAL	1	1	0	0

Schematic



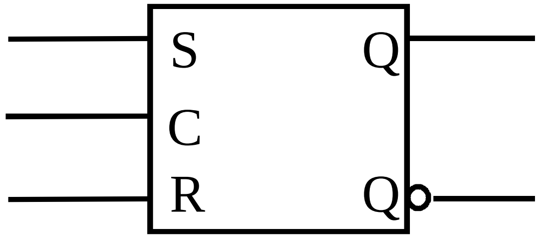
Characteristic Equation:

$Q(t+1) = S + R'Q(t)$

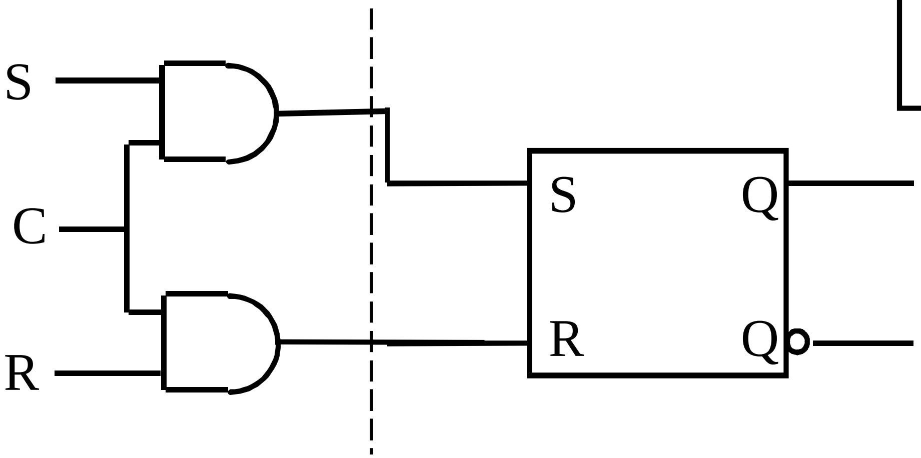
Consider:

- Timing Diagram
- Propagation delay
- Minimum pulse width
- Oscillation

S-R Latch with Enable



S	R	C	Q	/Q
0	0	1	Last Q	Last /Q
0	1	1	0	1
1	0	1	1	0
1	1	1	NA	NA
X	X	0	Last Q	Last /Q

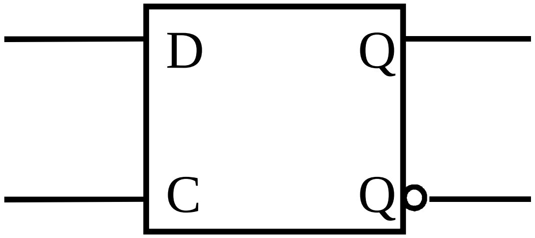


Only sensitive to S and R when enabled (C=1)

Same oscillation problem

How does C effect the minimum pulse width?

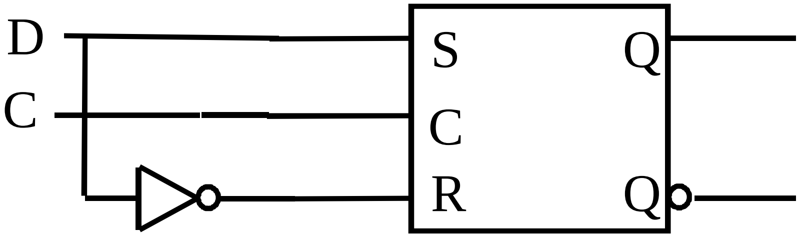
D Latch



C	D	Q	/Q
1	0	0	1
1	1	1	0
0	X	Last Q	Last /Q

Characteristic Equation:

$$Q(t+1) = D$$



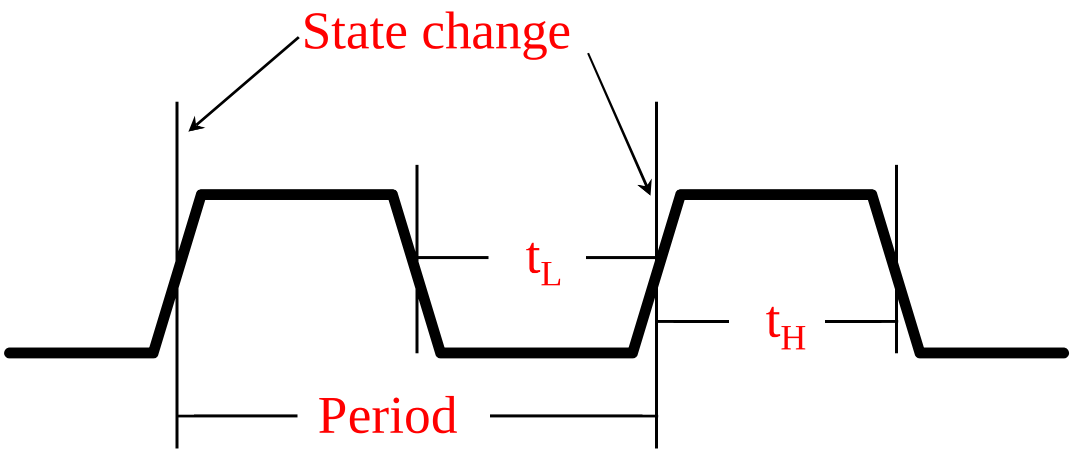
Store a data bit, not set/reset
The “Transparent” latch
No illegal operation problem

Types of Sequential Logic

- An **Asynchronous Sequential Circuit** uses ordinary gates and feedback loops to implement “memory” in a logic circuit.
  - Meeting minimum pulse width requirements may be ‘tricky’
- A **Synchronous Sequential Circuit** uses flip-flops (internally, an asynchronous sequential device) to form useful sequential logic functions or applications.
  - The state variables and outputs of a synchronous system change with respect to a controlling clock signal
  - Meeting minimum pulse width requirements is simplified by **restating all timing constraints in terms of the clock signal**

Clock Characteristics

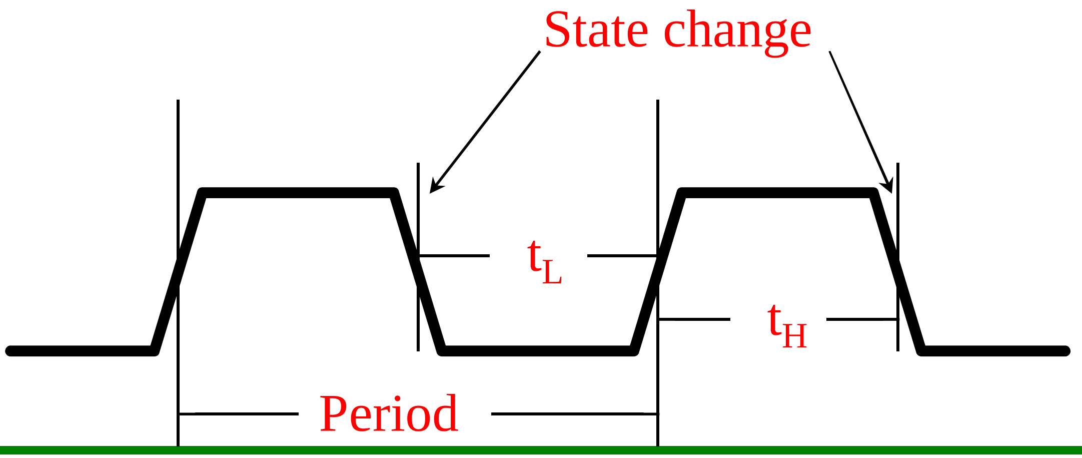
Active High



Frequency
= 1/Period

Duty Cycle
= t_H/Period

Active Low

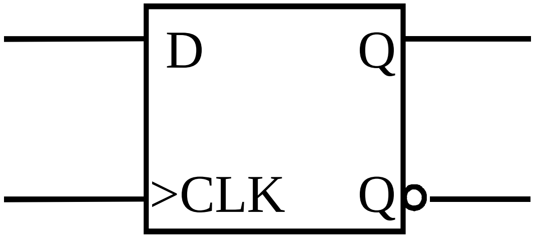


Duty Cycle
= t_L/Period

Sequential Logic Definitions

- **Clock** - the master timing element behind the state changes of most sequential circuits.
  - a clock signal is active high if the state changes occur at the rising edge (for edge triggered devices) or in the logic 1 state (for pulse-triggered devices)
  - active low if state changes occur at the falling edge or in the logic 0 state.
- **Clock Period** - time between successive transitions in the same direction
- **Clock Frequency** - reciprocal of the clock period
- **Clock Tick** - the first edge or pulse in a clock period, or the period itself
- **Duty Cycle** - the percentage of time that a clock is at its assertion level

Positive-Edge-Triggered D Flip-Flop



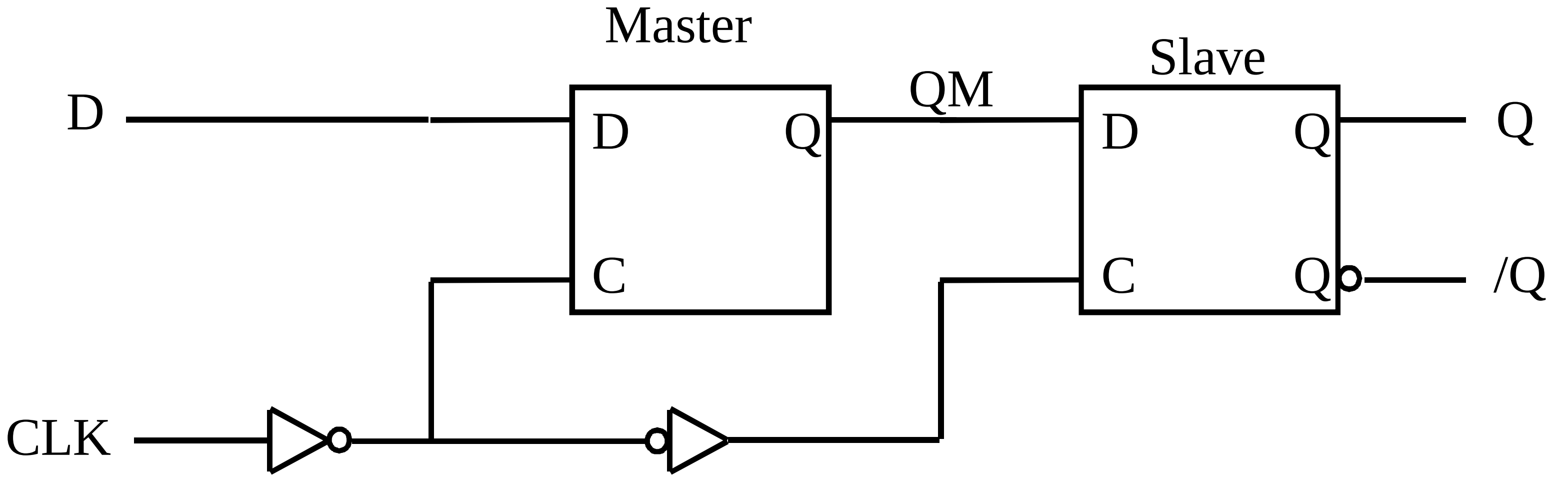
D	CLK	Q	/Q
0	↑	0	1
1	↑	1	0
X	0	Last Q	Last /Q
X	1	Last Q	Last /Q

Device samples inputs and changes state only on a clock edge.

Simplifies the meeting of setup/hold times.

Master FF - transparent on CLK’ (entire pulse)

Slave FF - transparent on CLK (master fixed)



What Are Flip-flops?

- Common asynchronous (feedback) sequential circuits
- Latch
  - Single-bit storage (memory)
  - Changes state at any time due to input change
  - Must guarantee a minimum pulse width to avoid metastability
  - Fast and cheap (small # of transistors)
  - Often used in high speed microprocessor design
- Flip-flop
  - Also single-bit storage
  - Changes state ONLY when a clock edge or pulse is applied
  - Uses setup and hold times before and after the clock pulse to avoid metastability
  - Clocking simplifies the design process

Characteristic Equations

- Describe the next state of a flip-flop as function of current state and inputs: $Q(t+1) = f\ (Q(t), \text{inputs})$
  - $t+1$ represents the next clock tick
  - t represents the current clock tick
  - $t-1$ represents the previous clock tick
  - and so on...

- Standard practice is to use Q* to represent $Q(t+1)$
- Derived from basic function table for a given flip-flop type
- Very useful in state machine analysis and design

- EXERCISE: What is the equation of a D-type FlipFlop?

Characteristic Equations

D flip-flop

D	Q(t)	Q(t+1)
0	0	0
0	1	0
1	0	1
1	1	1

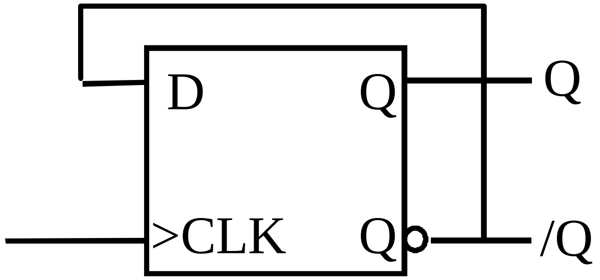
Characteristic Equation:

$$Q(t+1) = D$$

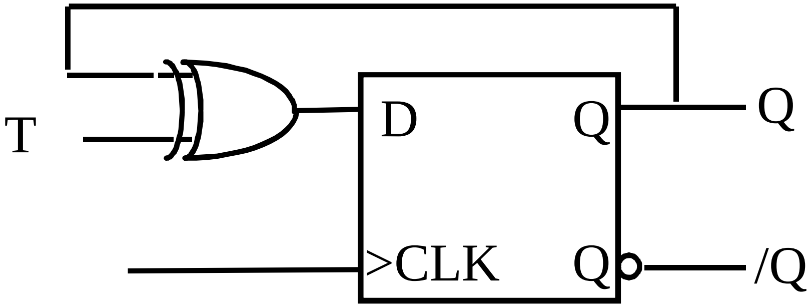
$$Q^* = D$$

T (toggle) Flip-Flop

- A T flip-flop changes state on every clock tick (if enabled)
- Possible circuit designs
  - T without enable

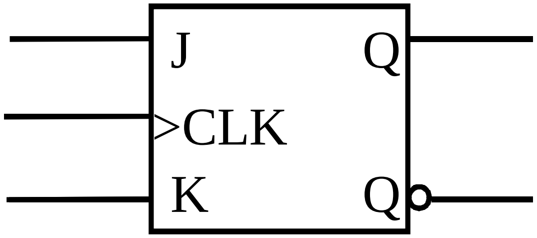


  - T with enable

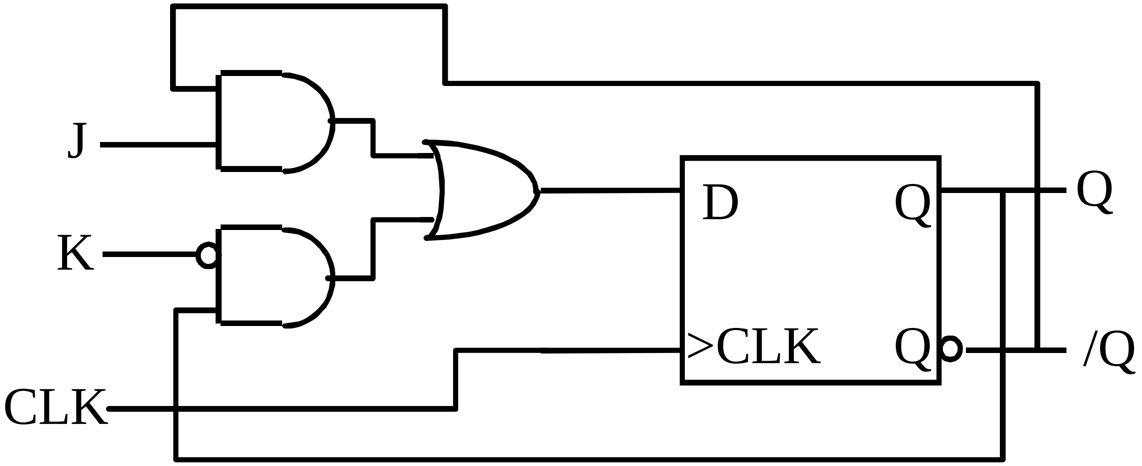


Equation?

Edge-Triggered J-K Flip-Flop



J	K	Q	/Q
0	0	Last Q	Last /Q
0	1	0	1
1	0	1	0
1	1	Last /Q	Last Q



Equation?

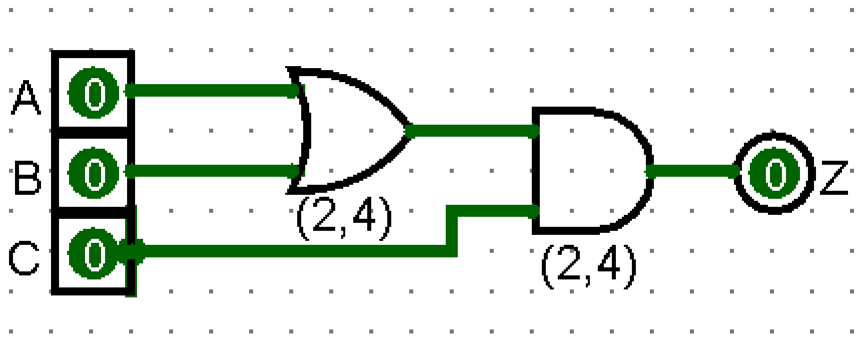
Functional Analysis of Clocked Synchronous State Machines

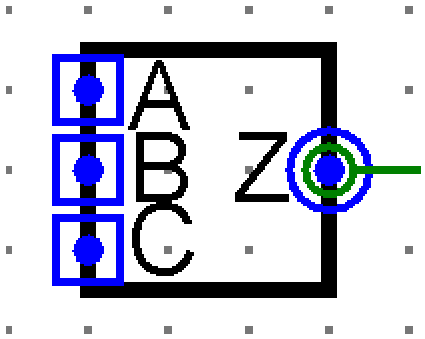
Analysis procedure

Moore Vs. Mealy

Methods of representation

Review: What does this circuit do?

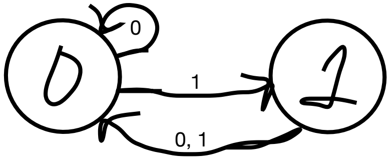




Q X	Q* Z
0 0	0 0
0 1	1 0
1 0	0 1
1 1	0 1

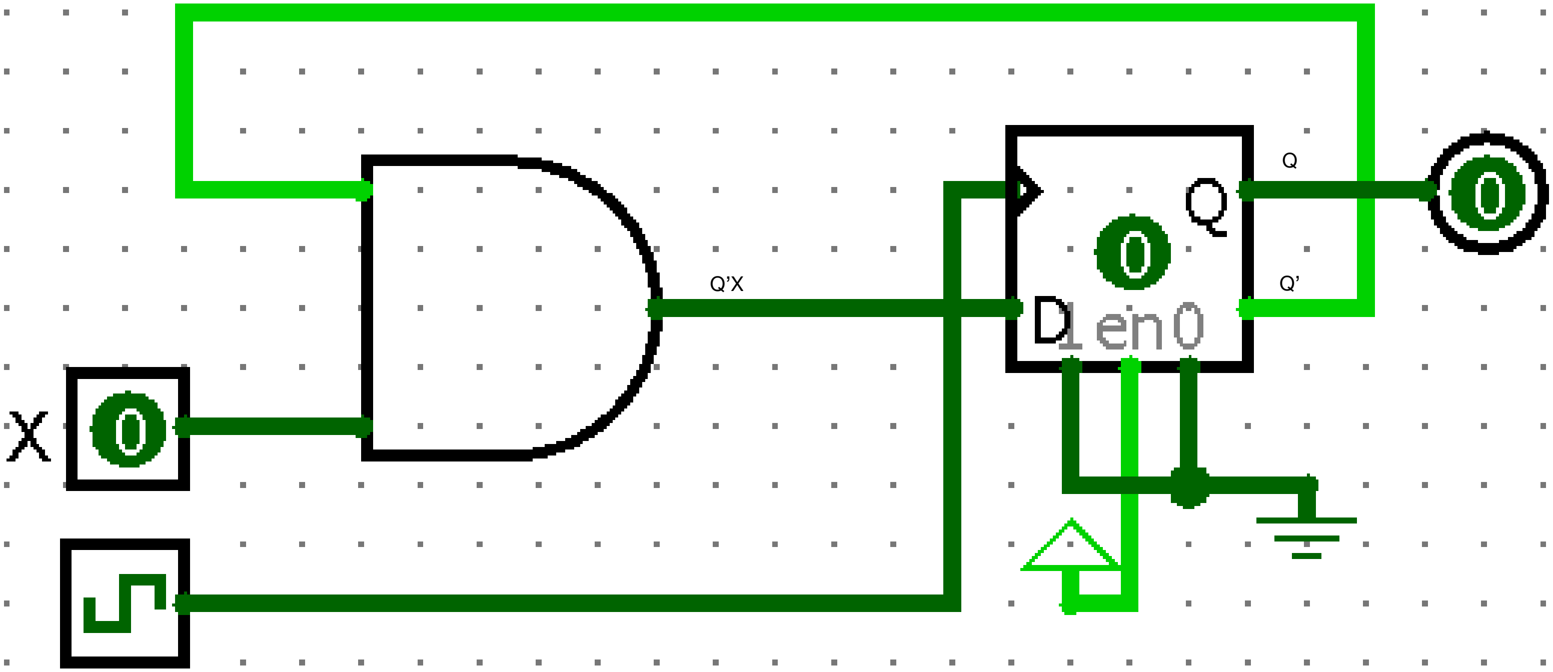
Q* = Q’X

ARCS are X, circles are Q*

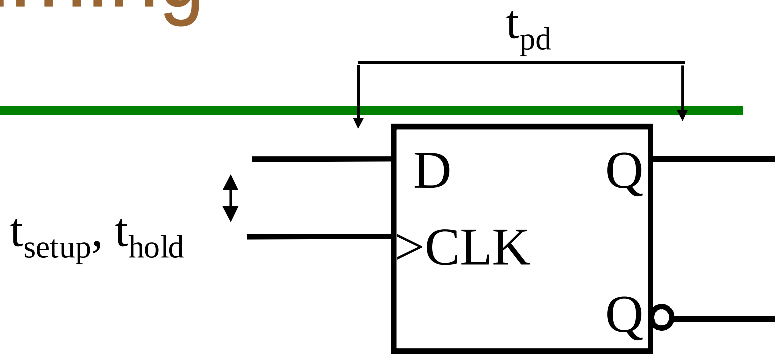


This is a state diagram

What does this circuit do?



Flip-flop Timing



Combinational

$t_{pd,min}$	- minimum propagation delay, input to output
$t_{pd,max}$	- maximum propagation delay, input to output

Latch

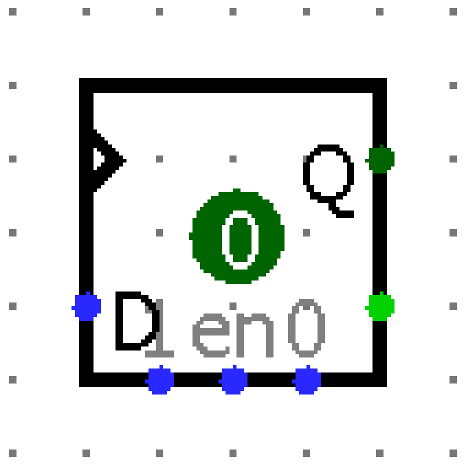
$t_{pd,min}$	- minimum propagation delay, input to output
$t_{pd,max}$	- maximum propagation delay, input to output
t_w	- minimum pulse width, input to input

Flip-flop

$t_{pd,\ min}$	- minimum propagation delay, CLK to output
$t_{pd,\ max}$	- maximum propagation delay, CLK to output
t_{setup}	- required time of stable input before CLK, input before CLK
t_{hold}	- required time of stable input after CLK, input after CLK

Synchronous & Asynchronous Inputs

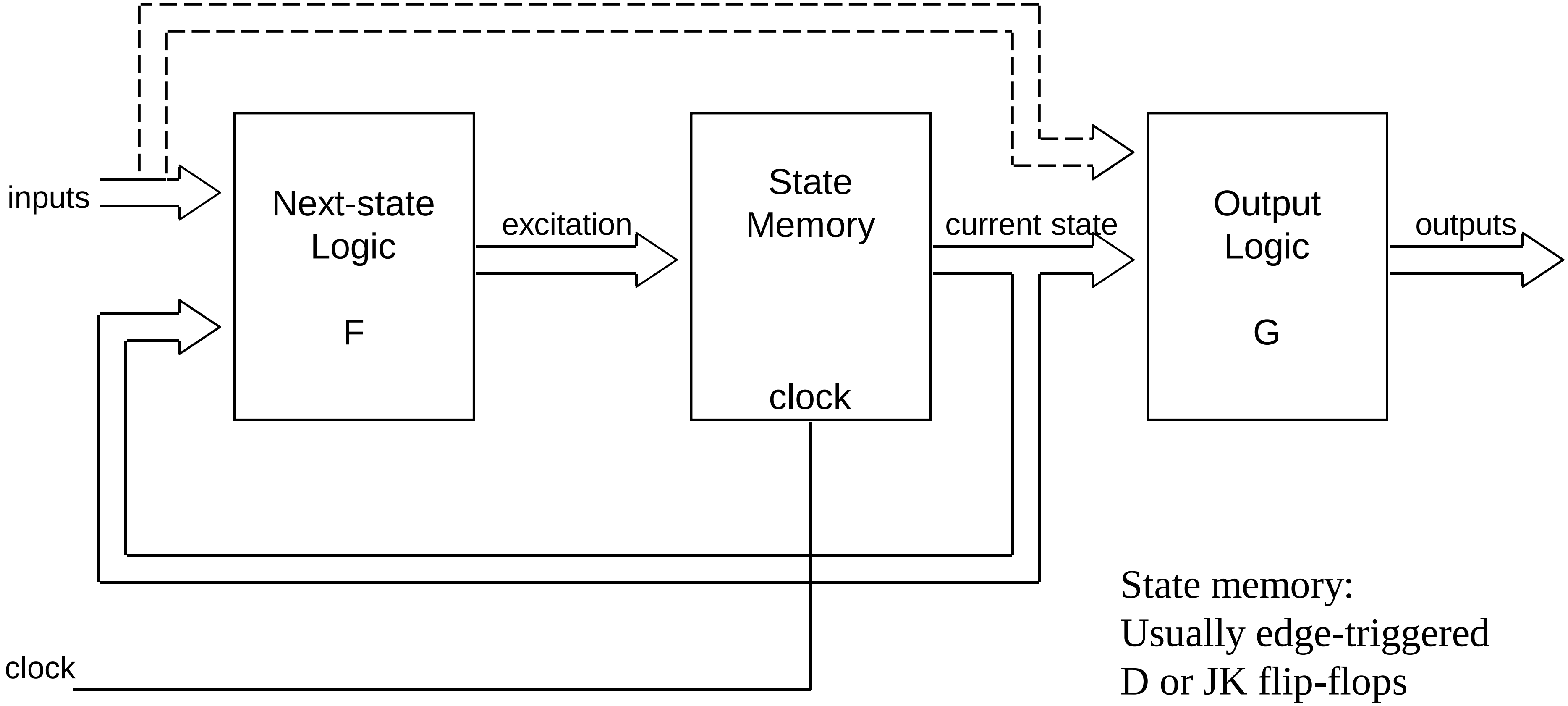
- Synchronous inputs
  - are aware of and respect the timing restrictions of the clocked device
  - Behavior which follows from a Synchronous Input is itself Synchronous and deterministic



- Asynchronous inputs
  - Are not aware of the clock
  - Have effects that occur immediately,
  - State depends on order of events
- Example: Preset and Reset direct inputs
- Good design practice dictates:
  - NEVER use asynchronous inputs for logic functions, only for system initialization to a known state
  - Why?

Clocked Synchronous State-machine Model

(Mealy machine)



State memory:
Usually edge-triggered
D or JK flip-flops

Clocked Synchronous State Machine Analysis

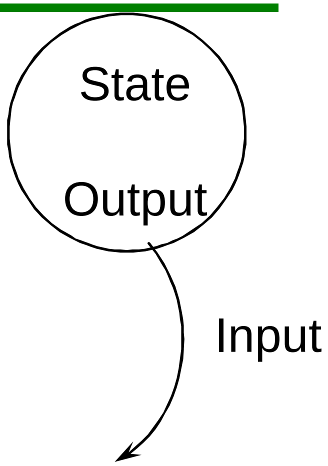
- Analysis
  - How does a given circuit work? What does it do?
  - How do input sequences map to output sequences?
- Clocked synchronous state-machine (CSSM)
  - **Clocked**: storage elements (flip-flops) use a clock input
  - **Synchronous**: all flip-flops use the same clock signal
- State-machine types
  - **Mealy Machine** (most general type):
    - Next state = F (current state, inputs)
    - Output = G (current state, inputs)
  - **Moore Machine**:
    - Next state = F (current state, inputs)
    - Output = G (current state)

Basic Analysis of State Machines

- Determine **next-state** and **output** functions F and G
- Use F and G to construct a **state/output table**
- Draw a graphical representation of the state/output table
  - State Diagram
    - Common for small designs
    - Similar to a finite automata
  - Timing Diagram
    - Common for all designs

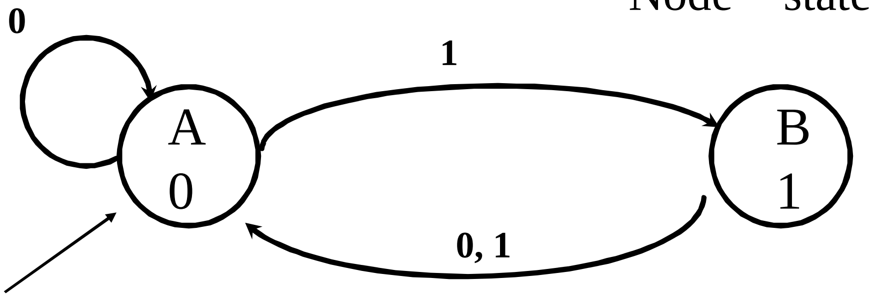
State Diagram

Basic Format:

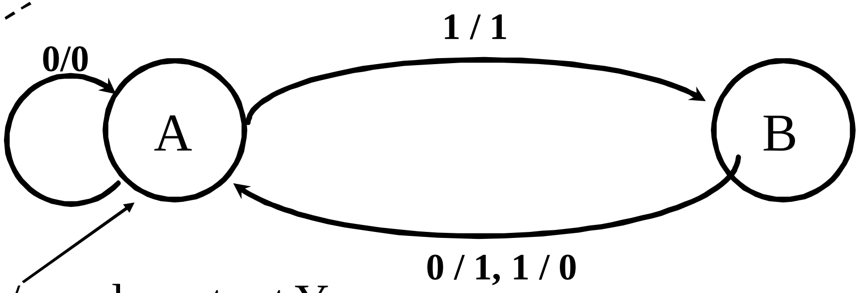


Moore

Format:
Arc = input X
Node = state/output Q

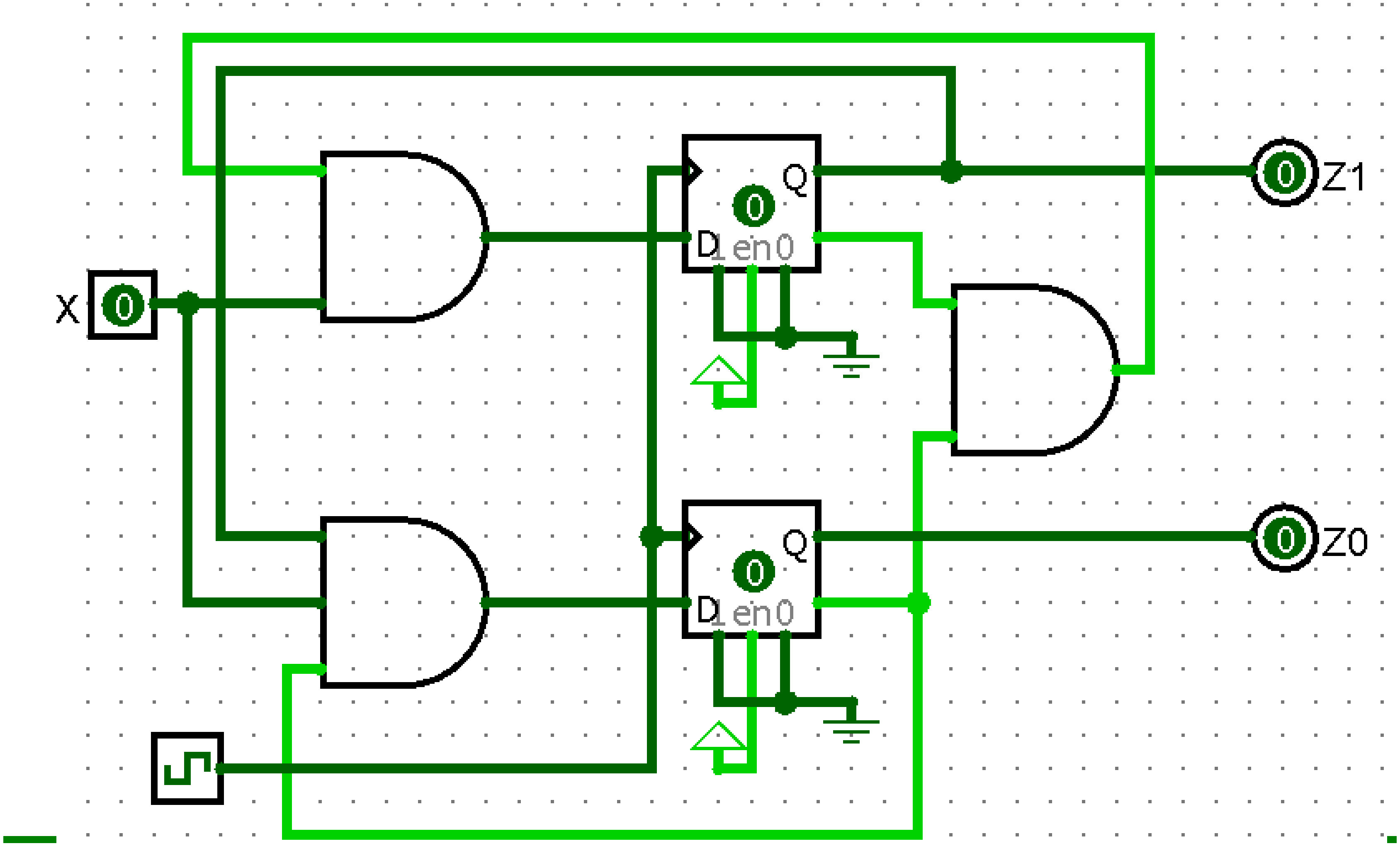


Mealy



Format:
Arc = input X / mealy output Y
Node = state

What does this device do?



Timing Analysis of Clocked Synchronous State Machines

Analysis procedure

Moore Vs. Mealy

Methods of representation

Timing Analysis

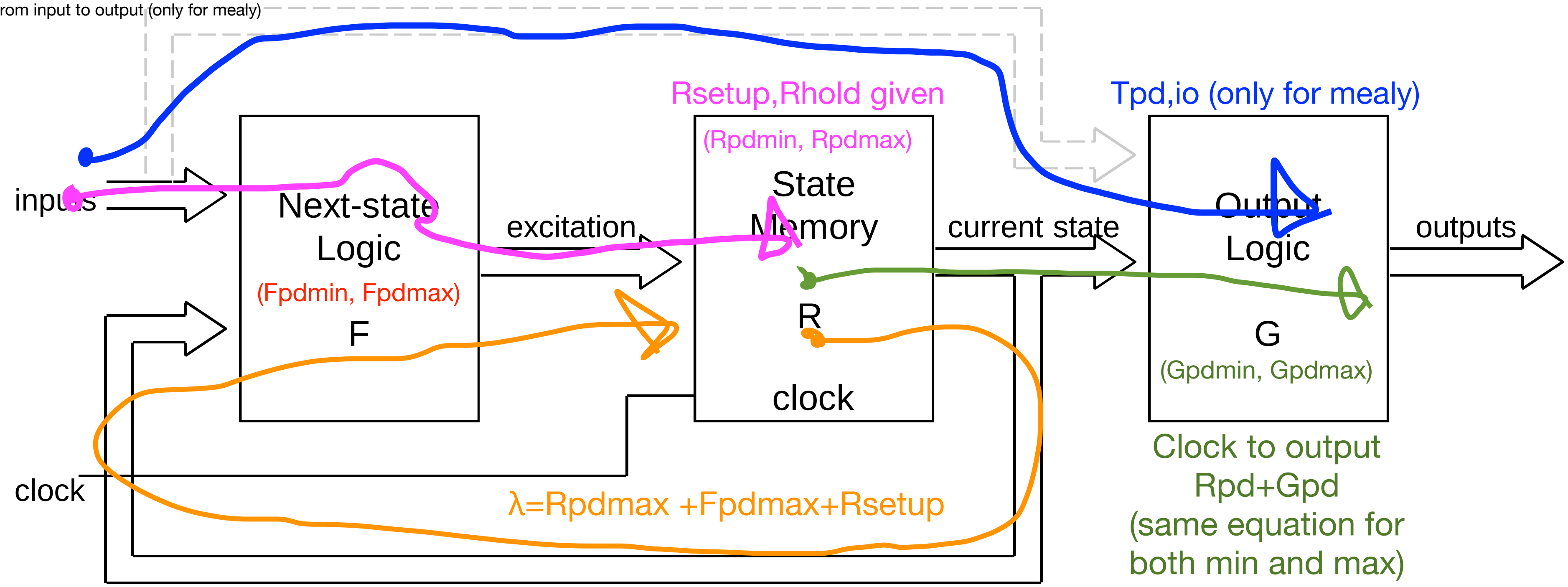
- All digital devices have associated propagation delays (min,max)
- Sequential devices have setup and hold times that must be satisfied to avoid metastable behavior
- Providing a synchronous clock simplifies timing analysis
  - All devices produce effects within a well-defined range
- At what speeds will a device function?
  - One transition per clock
  - What is the maximum clock rate?

- Sequential devices require the following timing documentation:
  - Maximum propagation delay (clock to output)
  - Minimum propagation delay (clock to output)
  - Setup time (input before clock)
  - Hold time (input after clock)
  - Maximum Clock Frequency (minimum clock period)

Clocked Synchronous State-machine Structure

Notes
R_setup(max) is from input to end of setup
period λ is max from one flip flop to any other flip flop
tpd,clock-to-output is path from flip flop to output
tpd,in-out is from input to output (only for mealy)

(Mealy machine)



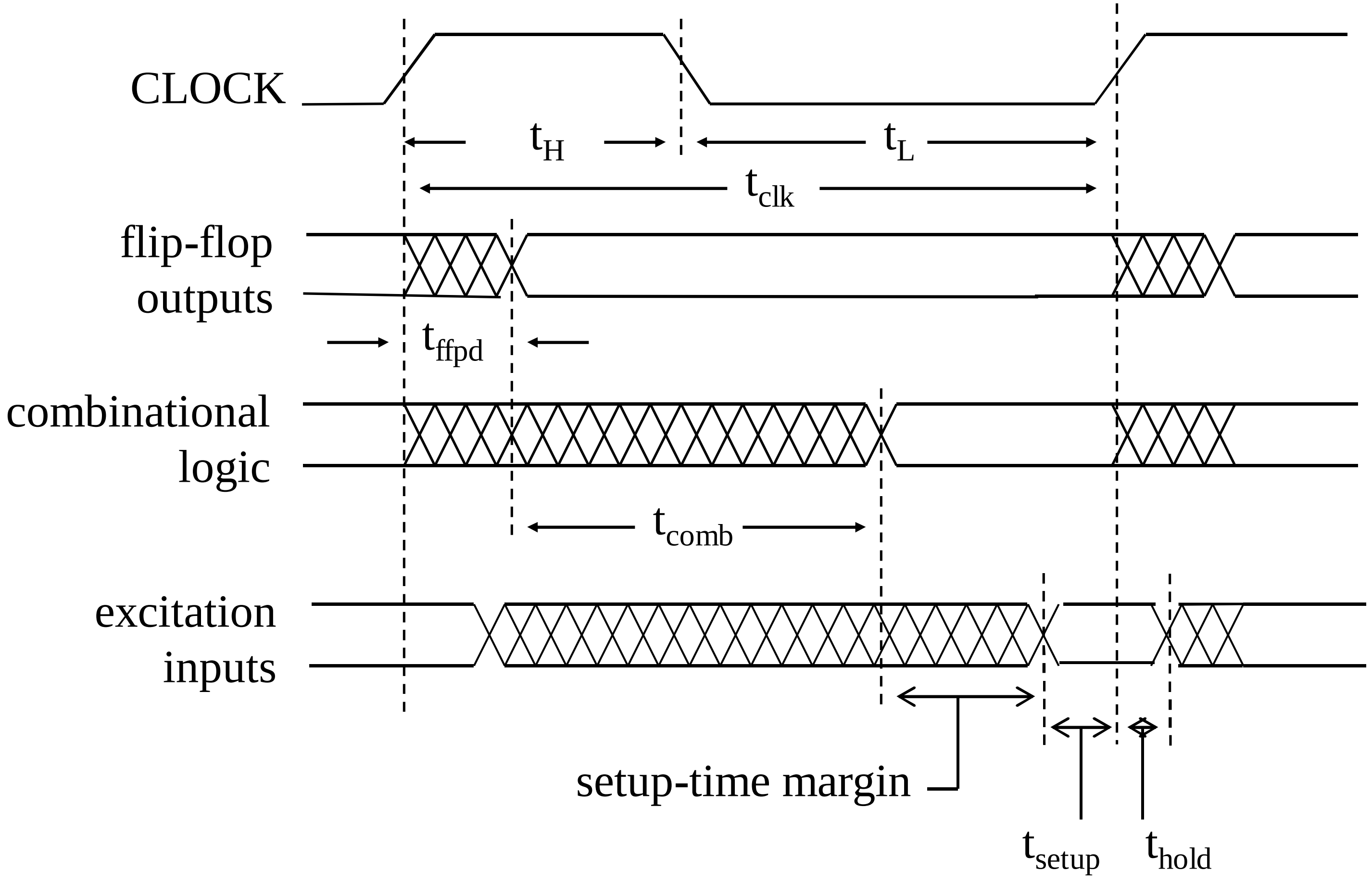
Calculate:

t, F_pd(max)
t, F_pd(min)

t, R_pd(max)
t, R_pd(min)
Rsetup+Fpdmax t, R_setup(max)
Rhold-Fpdmin t, R_hold(max)

t, G_pd(max)
t, G_pd(min)

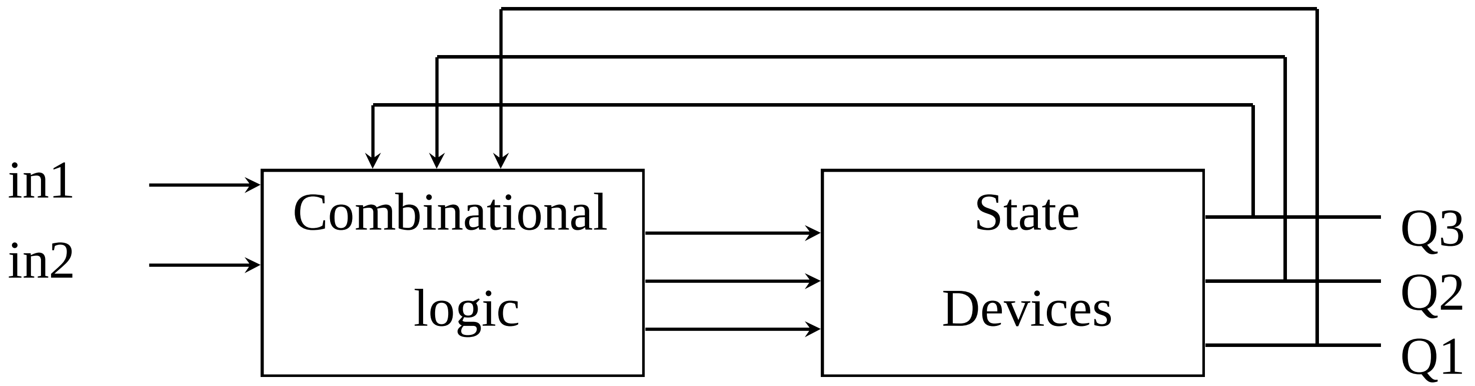
Timing Diagram



Calculating Sequential Device Timing Specs.

- Calculate the delay from *clock edge* to worst-case primary output:
  - t_pd,clock-to-output (min) = t,R_pd(min) + t,G_pd(min)
  - t_pd,clock-to-output (max) = t,R_pd(max) + t,G_pd(max)
- Calculate the delay from *input* to worst-case (Mealy) primary output:
  - t_pd,input-to-output (min) = t,G_pd(min)
  - t_pd,input-to-output (min) = t,G_pd(min)
- Calculate the worst-case setup time for any input:
  - t_setup = t,F_pd(max) + t,R_setup(max)
- Calculate the worst-case hold time for any input:
  - t_hold = t,R_hold(max) - t,F_pd(min)
- Calculate the maximum clock rate by finding the minimum period:
  - min. period = t,R_pd(max) + t,F_pd(max) + t,R_setup(max)
- Make certain that the device works!
  - t,R_pd(min) + t,F_pd(min) > t,R_hold(max)

Synchronous System Example



$t_{pd,comb}$ = 2 ns (min) to 20 ns (max) $t_{pd,ff}$ = 3 ns (min) to 15 ns (max)

t_{setup}= 5 ns; t_{hold} = 4 ns

time,prop.delay (clock->output) $t_{pd(co),min}$ = 3 ; $t_{pd(co),max}$= 15

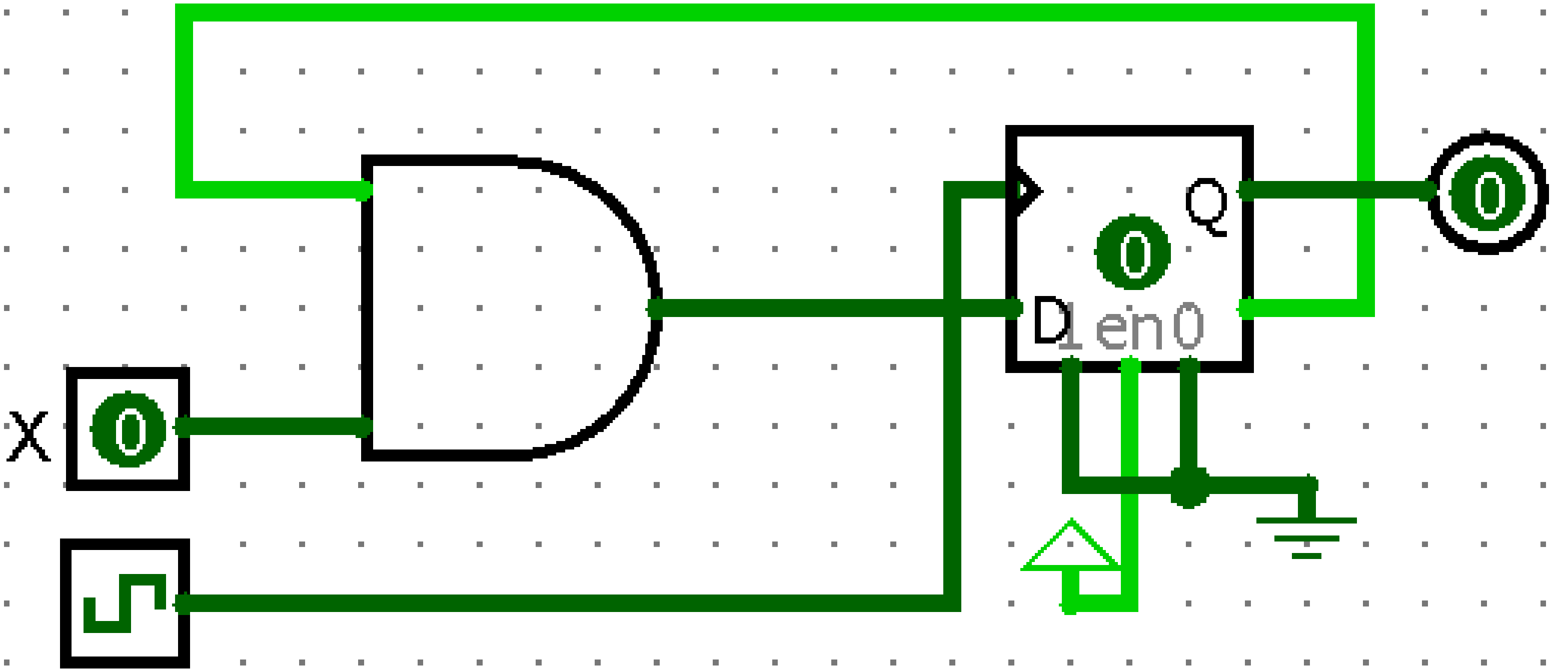
time,prop.delay(input->output) $t_{pd(io),min}$ = — ; $t_{pd(io),max}$= —

Setup/Hold Time: t_{setup} = 5+20 = 25 ; t_{hold} = 4-2 = 2

Max Frequency? t_{clk} >= 15 + setup time = 40

f_{max} <= 1/40

What does this circuit do?



What does this device do?

get

tsetup: 5+6 = 11 (setup + fpdmax from 3input and)

thold: 1-2 = 0 (hold - fpdmin from 2input and)

pd,clk->out (min): 3+0 = 3 (rpdmin + gpdmin, there is no g because nothing block path to output)

pd,clk->out (max): 6+0 = 3 (rpdmax+gpdmax, there is no g because nothing block path to output)

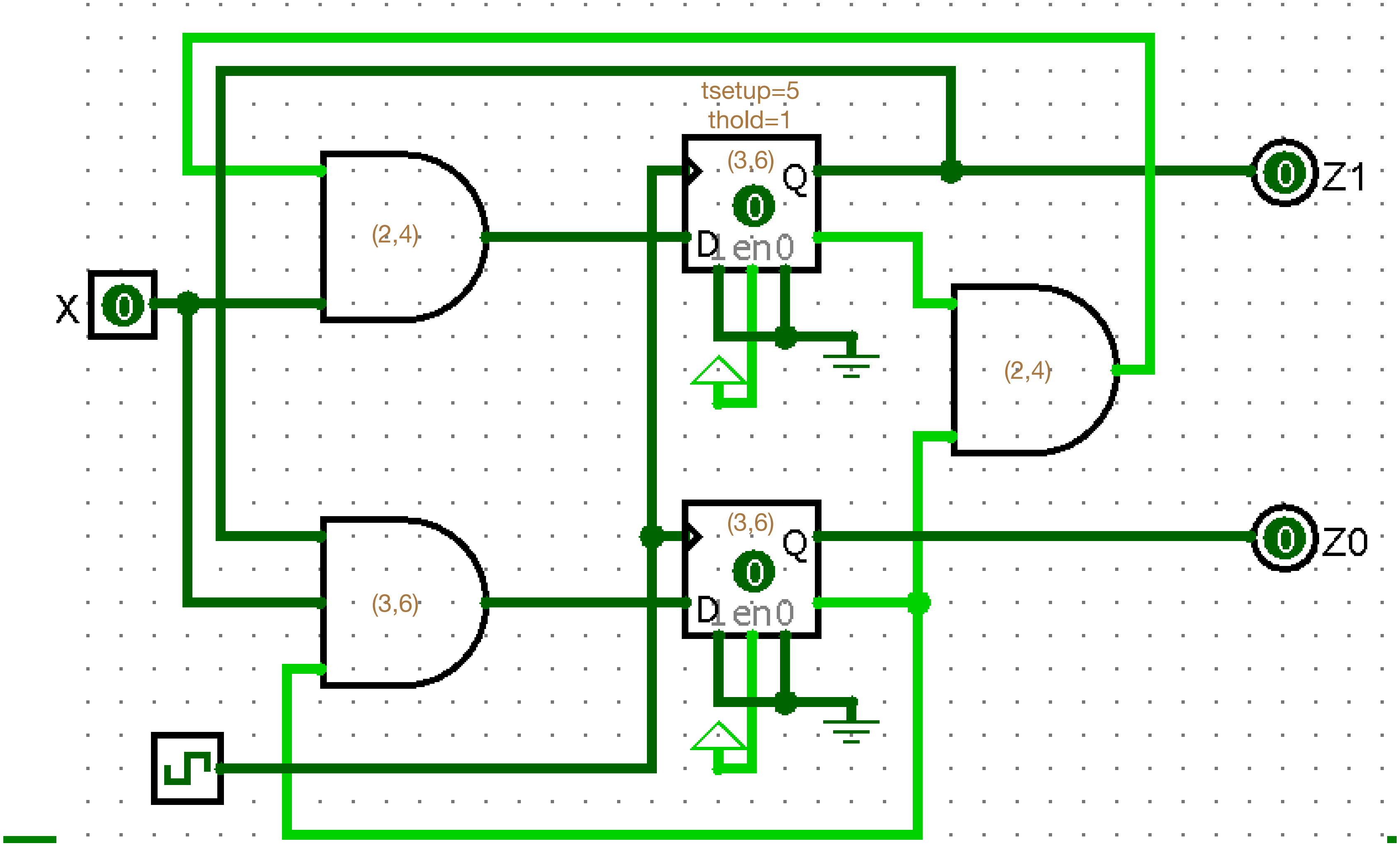
pd,in>out (min): -

pd,in->out (max): -

period: 6+8+5 = 19 (Rpdmax + fpdmax through 2 2input and + tsetup)

Frequency = 1/period (can be in the form where period is exactly as calculated)

example: 1/(19ns)



Introduction to the design of Clocked Synchronous State Machines

Basic Design procedure

State diagrams

Synthesis process

State Machine Design Procedure

Most difficult and creative

1. Build state/output table (or state diagram) from word description
2. Minimize number of states
3. Choose state variables and assign bit combinations to named states

Well-defined procedure - can be automated

4. Build transition/output table from state/output table/diagram
5. Choose flip-flop type (D, J-K, etc.)
6. Build excitation table for flip-flop inputs from transition table
7. Derive excitation equations from excitation table
8. Derive output equations from transition/output table
9. Draw logic diagram with excitation logic, output logic, and state memory elements

Synthesis is generally followed by simulation and verification

Step 1: State Table Design from Word Description

- Much like writing a computer program:
  - Start with a vague description
  - Make decisions, sometimes using common sense, and sometimes arbitrarily
  - Handle all special cases (even those not covered in the word description)
  - Design may not work, so debug and iterate

Design

Synthesis

Unused States: Minimum Cost/Risk

- If extra unused state codes exist (number of available states $2^n > s$), then two choices are common:

- **Minimal Risk** = most reliable but most expensive
  - Assume it is possible to enter unused state by noise, weird inputs, etc.
  - So include all unused states as present states, but all corresponding next states go to “initial” or “idle”

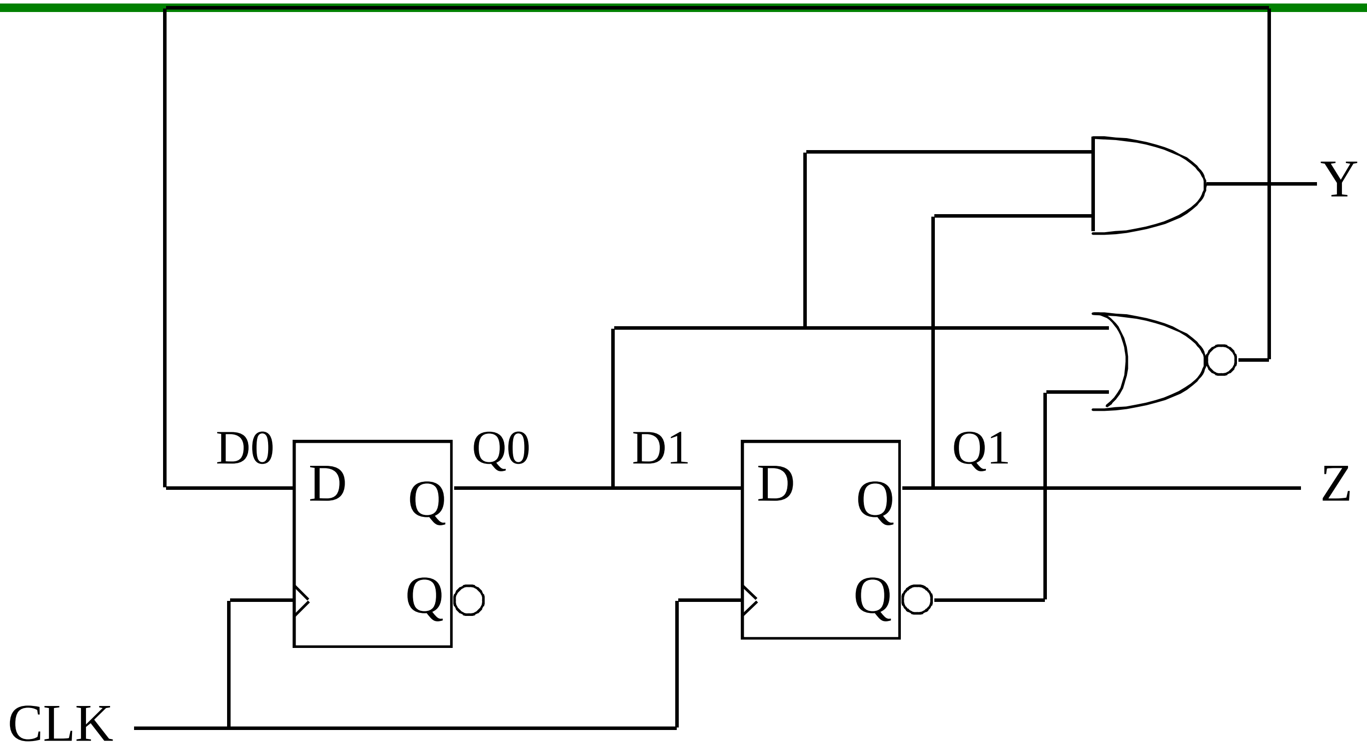
- **Minimal Cost** = somewhat risky but least expensive
  - Assume unused states NEVER entered accidentally.
  - So the next state and outputs of all unused states = “don’t care” to reduce next state and output logic

- Achieving both minimal risk and minimal cost is sometimes possible!

Examples of analysis, design, and implementation

These examples are ‘extra’ problems for those who missed class or just want more problems to review

Example 1 - Circuit w/o Primary Inputs



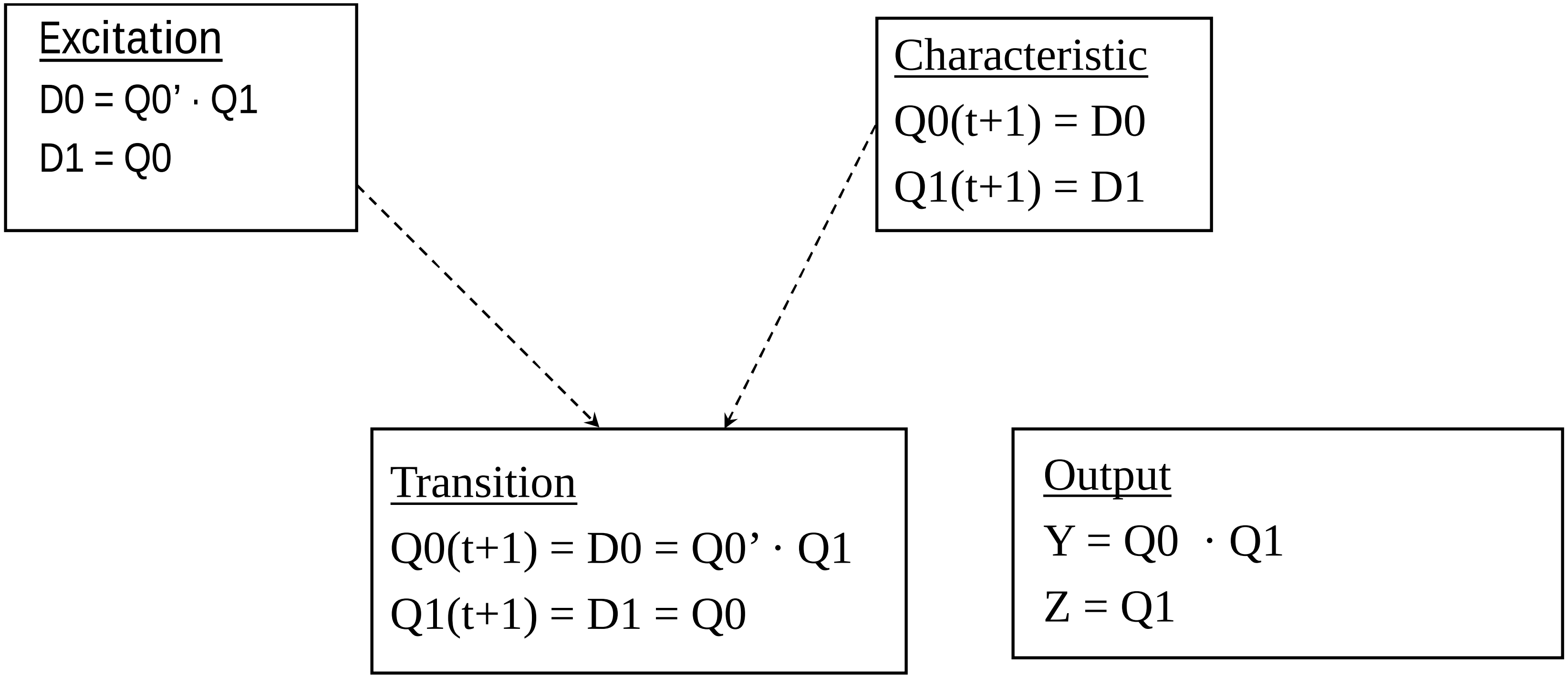
Excitation: $D0 = (Q0 + Q1')' = Q0' \cdot Q1$

$D1 = Q0$

Output: $Y = Q0 \cdot Q1$

$Z = Q1$ **Thus, Moore machine**

Example 1 - Equations



Excitation

$D0 = Q0' \cdot Q1$

$D1 = Q0$

Characteristic

$Q0(t+1) = D0$

$Q1(t+1) = D1$

Transition

$Q0(t+1) = D0 = Q0' \cdot Q1$

$Q1(t+1) = D1 = Q0$

Output

$Y = Q0 \ \cdot Q1$

$Z = Q1$

Example 1 - Tables

No inputs!

State Table:

Q1	Q0		Y	Z
0	0	00	0	0
0	1	10	0	1
1	0	01	0	0
1	1	01	1	1

Q1(t+1)Q0(t+1)

Transition

$Q1(t+1) = D1 = Q1' \cdot Q0$

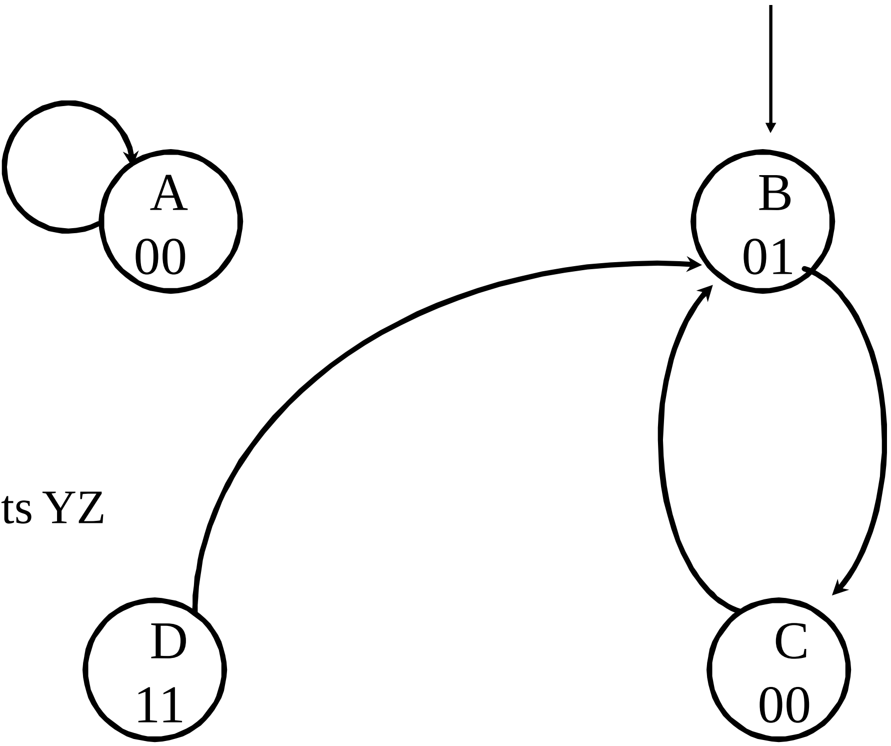
$Q0(t+1) = D0 = Q1$

State Table w/named states:

S		Y	Z
A	A	0	0
B	C	0	1
C	B	0	0
D	B	1	1

S(t+1)

Example 1 - State Diagram

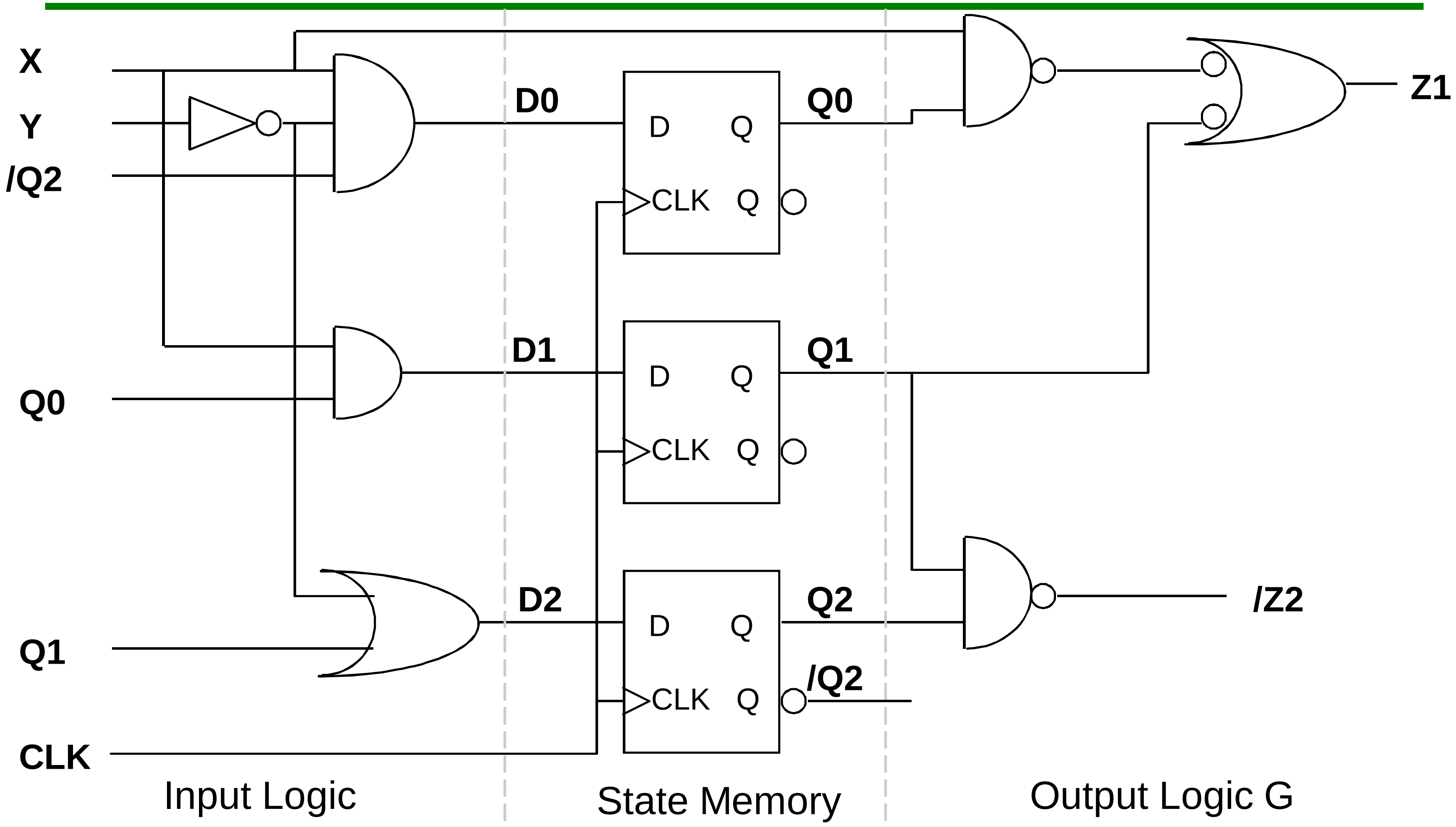


Format:
Arc = no input
Node = state/outputs YZ

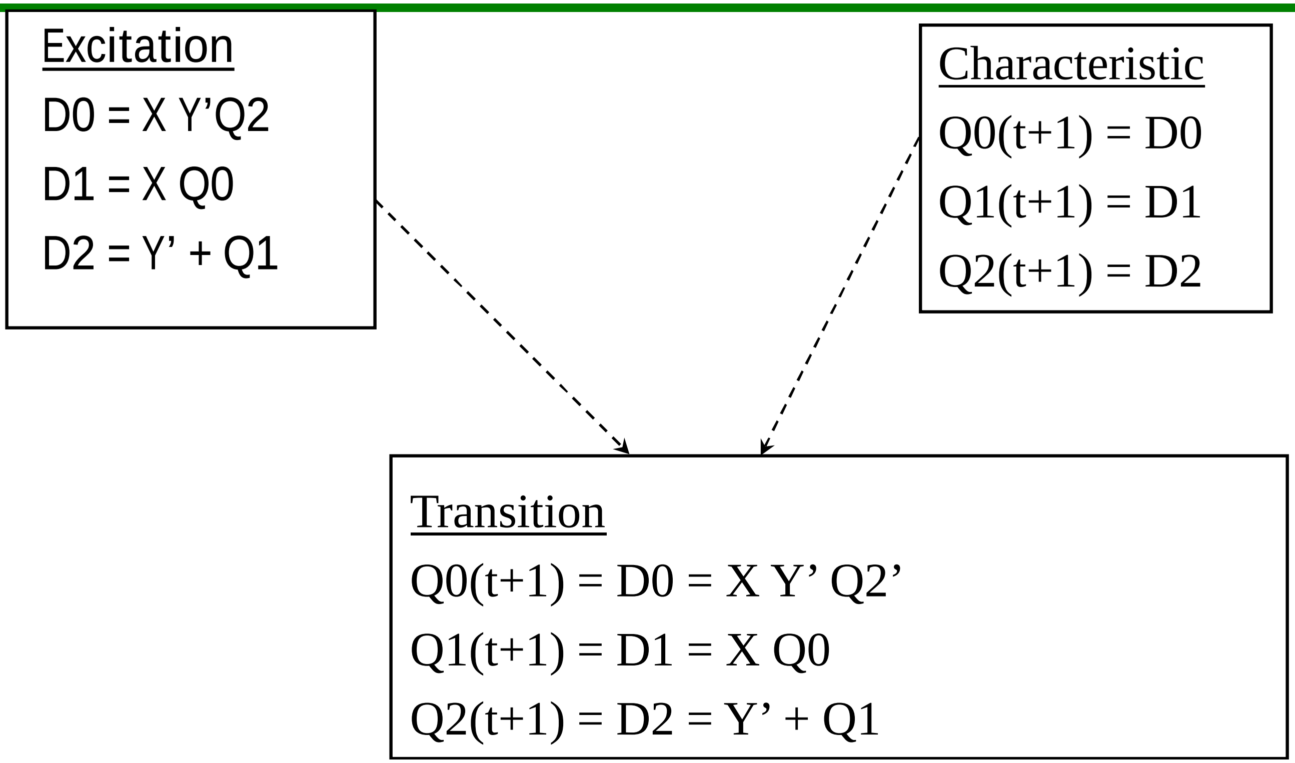
A,D Unreachable, Only B,C are useful.

Therefore, only 1 flip-flop is needed.

Example 2 - State Machine with D Flip-flops



Example 2 - Equations



Excitation

$D0 = X Y'Q2$

$D1 = X Q0$

$D2 = Y' + Q1$

Characteristic

$Q0(t+1) = D0$

$Q1(t+1) = D1$

$Q2(t+1) = D2$

Transition

$Q0(t+1) = D0 = X Y' Q2'$

$Q1(t+1) = D1 = X Q0$

$Q2(t+1) = D2 = Y' + Q1$

Output

$Z1 = X Q0 + Q1'$

$/Z2= (Q1 Q2)'$

Example 2 - Two-Dimensional State table

state				XY			
name	Q2	Q1	Q0	00	01	11	10
A=	0	0	0	100, 11	000, 11	000, 11	101, 11
B=	0	0	1	100, 11	000, 11	010, 11	111, 11
C=	0	1	0	100, 01	100, 01	100, 01	101, 01
D=	0	1	1	100, 01	100, 01	110, 11	111, 11
E=	1	0	0	100, 11	000, 11	000, 11	100, 11
F=	1	0	1	100, 11	000, 11	010, 11	110, 11
G=	1	1	0	100, 00	100, 00	100, 00	100, 00
H=	1	1	1	100, 00	100, 00	110, 10	110,10

Q2(t+1) Q1(t+1) Q0(t+1), Z1 /Z2
(Next State, Outputs)

Transition Equations

Q0(t+1) = D0 = X Y’ Q2’

Q1(t+1) = D1 = X Q0

Q2(t+1) = D2 = Y’ + Q1

X	Y	Q2	Q1	Q0	
1	0	0	-	-	
1	-	-	-	1	
-	0	-	-	-	or
-	-	-	1	-	

Output Equations

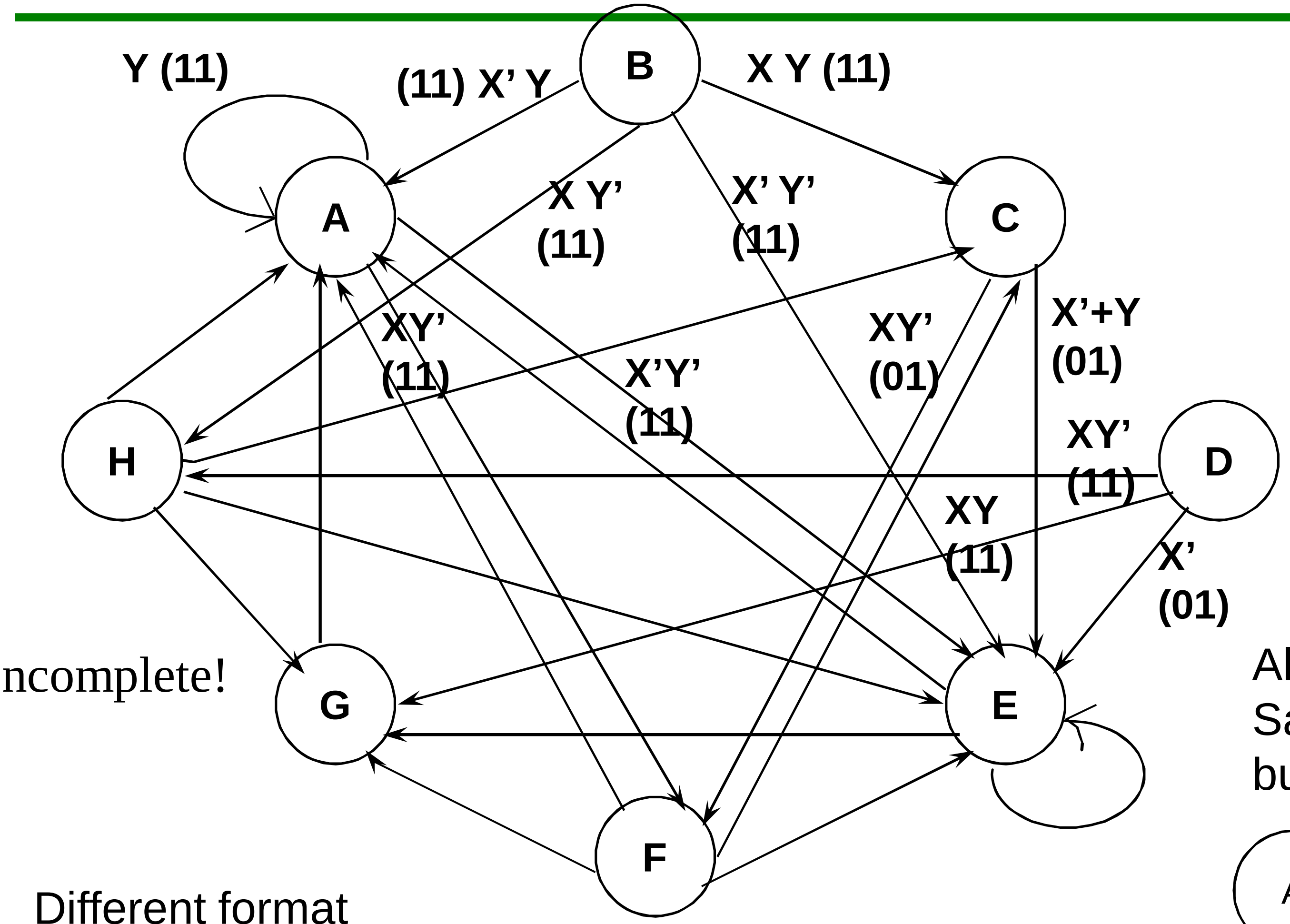
Z1 = X Q0 + Q1’

/Z2= (Q1 Q2)’

Example 2 - Named State / Output table

	XY			
S	00	01	11	10
A	E, 11	A, 11	A, 11	F, 11
B	E, 11	A, 11	C, 11	H, 11
C	E, 01	E, 01	E, 01	F, 01
D	E, 01	E, 01	G, 11	H, 11
E	E, 11	A, 11	A, 11	E, 11
F	E, 11	A, 11	C, 11	G, 11
G	E, 00	E, 00	E, 00	E, 00
H	E, 00	E, 00	G, 10	G,10
	S(t+1), Z1 /Z2			

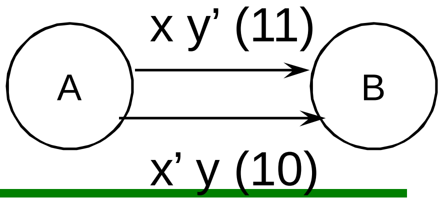
Example 2 - State Diagram



Incomplete!

Different format
Arc: input expression (outputs) = expression (Z1 /Z2)

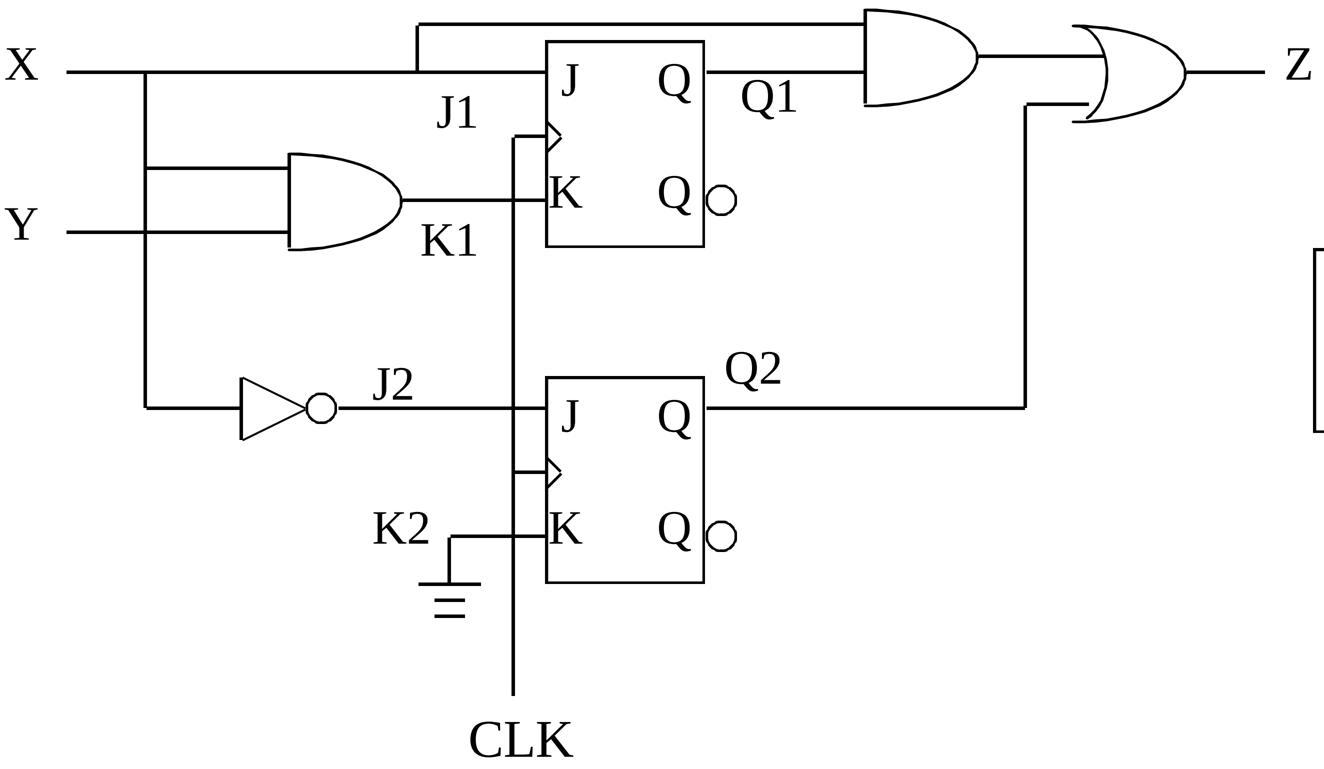
Also possible:
Same transition,
but different outputs



Analysis of J-K Flip-Flop State Machines

- There are two excitation equations per flip-flop **(J , K)**
- The characteristic equation : **$Q(t+1) = J \cdot Q(t)' + K' \cdot Q(t)$**
- Use the same analysis procedure shown previously

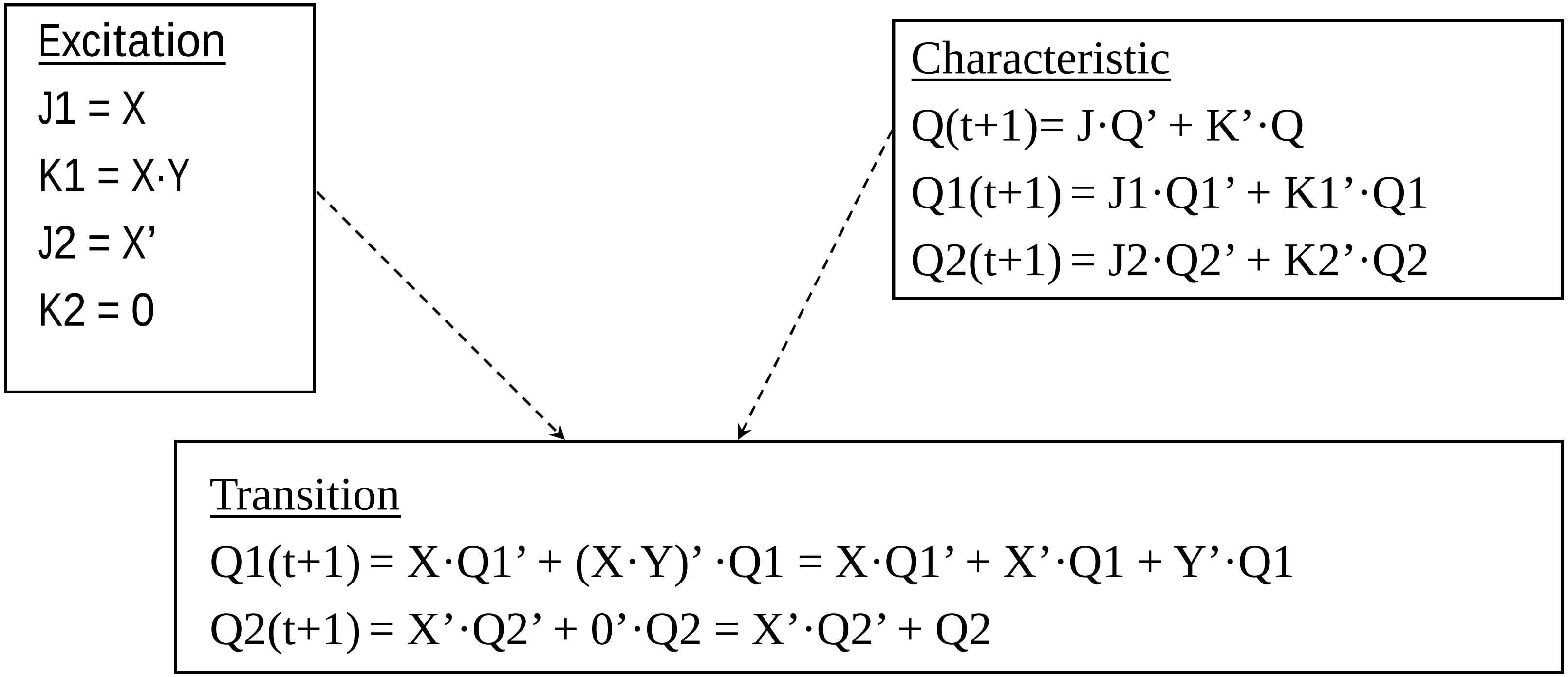
Example 3 - State Machine with J-K Flip-flops



Mealy Output:

$Z = X{\cdot}Q1 + Q2$

Example 3 - Equations



Excitation

$J1 = X$

$K1 = X \cdot Y$

$J2 = X'$

$K2 = 0$

Characteristic

$Q(t+1) = J \cdot Q' + K' \cdot Q$

$Q1(t+1) = J1 \cdot Q1' + K1' \cdot Q1$

$Q2(t+1) = J2 \cdot Q2' + K2' \cdot Q2$

Transition

$Q1(t+1) = X \cdot Q1' + (X \cdot Y)' \cdot Q1 = X \cdot Q1' + X' \cdot Q1 + Y' \cdot Q1$

$Q2(t+1) = X' \cdot Q2' + 0' \cdot Q2 = X' \cdot Q2' + Q2$

Mealy Output

$Z = X \cdot Q1 + Q2$

Example 3 - State Table

			XY			
S	Q1	Q2	00	01	11	10
A	0	0	01,0	01,0	10,0	10,0
B	0	1	01,1	01,1	11,1	11,1
C	1	0	11,0	11,0	00,1	10,1
D	1	1	11,1	11,1	01,1	11,1

Q1(t+1) Q2(t+1), Z

Transition

$Q1(t+1) = X\cdot Q1' + (X\cdot Y)'\cdot Q1 = X\cdot Q1' + X'\cdot Q1 + Y'\cdot Q1$

$Q2(t+1) = X'\cdot Q2' + 0'\cdot Q2 = X'\cdot Q2' + Q2$

Mealy Output

$Z = X\cdot Q1 + Q2$

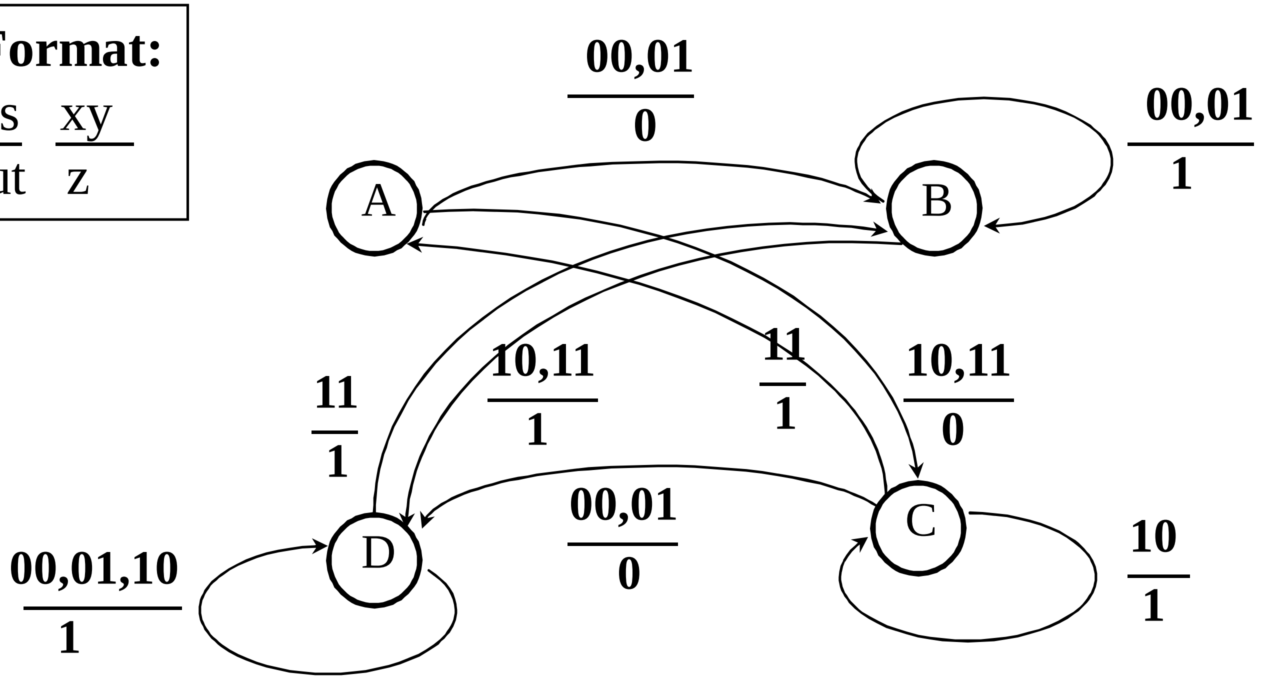
Example 3 - Named State/Output Table

	XY			
S	00	01	11	10
A	B,0	B,0	C,0	C,0
B	B,1	B,1	D,1	D,1
C	D,0	D,0	A,1	C,1
D	D,1	D,1	B,1	D,1
	S(t+1), Z			

Example 3 - State Diagram

Arc Format:

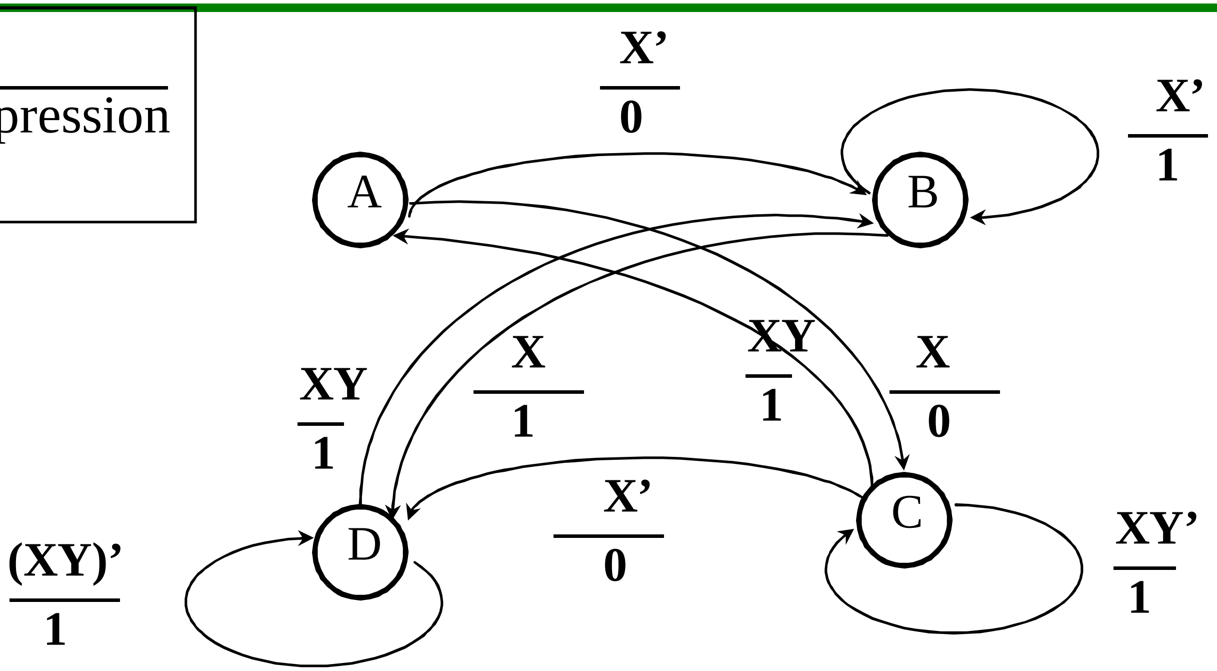
$$\frac{\text{inputs}}{\text{output}} \quad \frac{xy}{z}$$



Example 3 - State Diagram

Arc Format:

$$\frac{\text{Transition Expression}}{\text{output}}$$



For each state/input combination there must be exactly one next-state (and output).

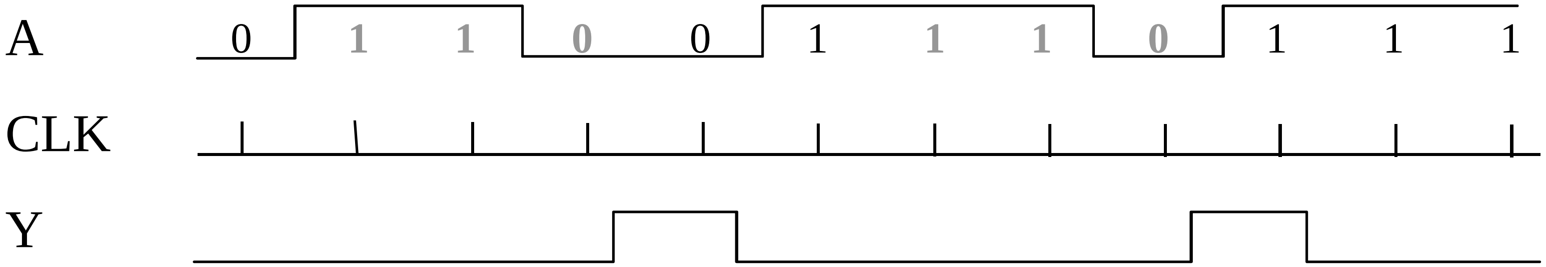
Mutual Exclusion: No more than one transition arc from any state can be satisfied by any input assignment

All Inclusion: At least one transition arc must exist from any state for any input assignment

Design Example 1: 110 Detector

- Word description (input sequence detector)
  - Design a state machine with input A and output Y.
  - Y should be 1 whenever the sequence 1 1 0 has been detected on A on the last 3 consecutive clock ticks.
  - Otherwise, Y = 0
  - Note: this is a Moore machine, that is the output, Y, depends only on inputs at previous clocks, not on the current input.

- Interpretation of word description (only rising clock edges, or **ticks**, are shown)

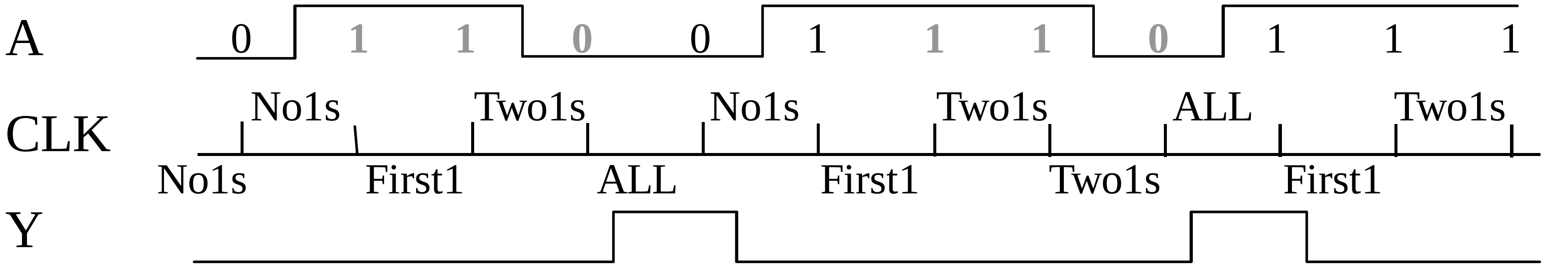


Design Example 1: Choosing States

- Possible states (What do you need to remember?)

– Initial	:	power up, no clocks yet	Y = 0
– No1s	:	first 1 not found	Y = 0
– First1	:	first 1 found	Y = 0
– Two1s	:	at least 2 consecutive 1s found	Y = 0
– ALL	:	found 1 1 0	Y = 1

- Are all the states needed?
  - Notice: Initial is equivalent to NO1s
  - We can drop the state Initial and replace it with state No1s

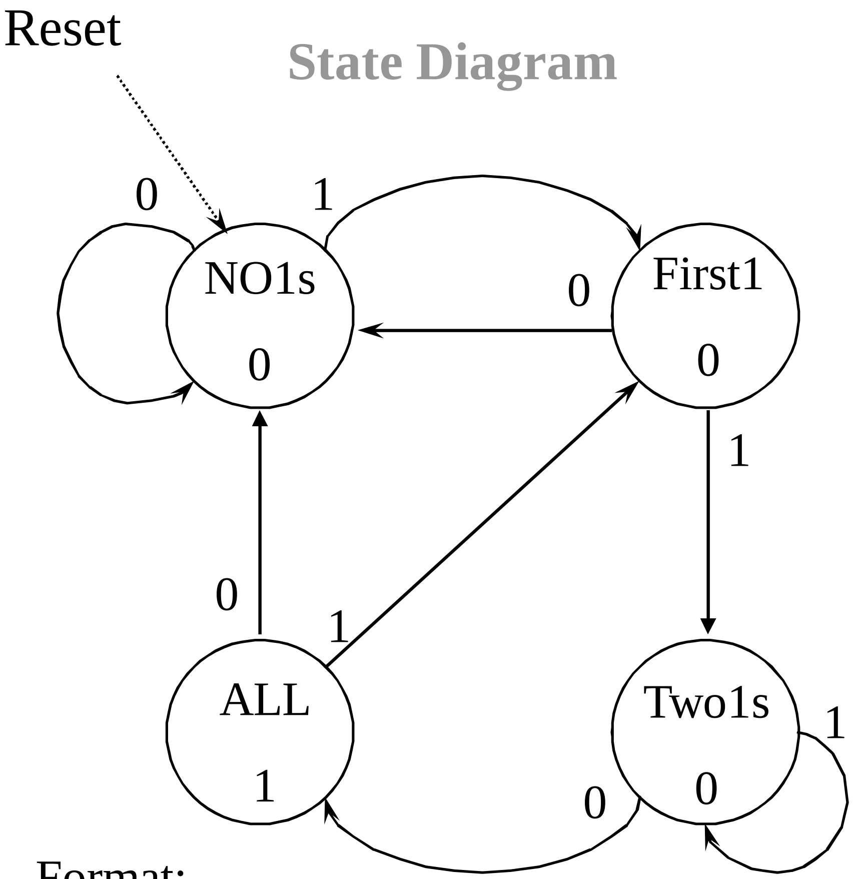


Design Example 1: State Table and Diagram

State Table

	A		
S	0	1	Y
NO1s	NO1s	First1	0
First1	NO1s	Two1s	0
Two1s	ALL	Two1s	0
ALL	NO1s	First1	1
	S(t+1)		

State Diagram

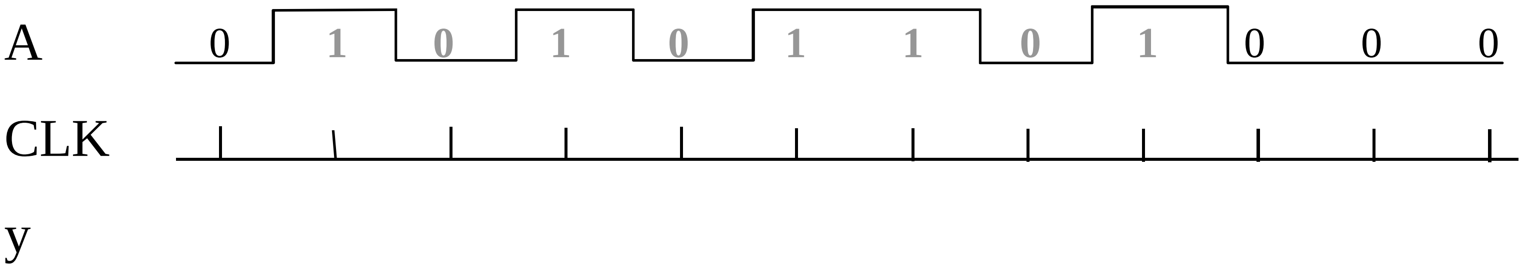


Format:
Arc: input A
Node: state/output Y

Design Example 2: 110/101 Detector

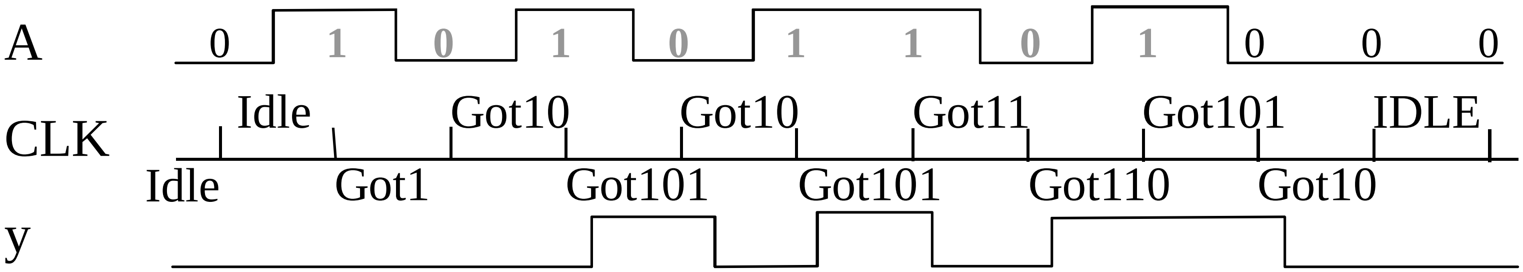
- Word description (input sequence detector)
  - Design a state machine with input A and output Y.
  - Y = 1 when either sequence 1 1 0 **or** 1 0 1 has been detected on A on the last 3 consecutive clock ticks.
  - Otherwise Y = 0
  - Note: Correct sequences may overlap and still be accepted

- Interpretation of word description (only rising clock ticks are shown)



Design Example 2: Choosing States

- Possible states (What do you want to remember?)
  - Idle : Initial, no starting 1 yet Y = 0
  - Got1 : A = 1 on last tick Y = 0
  - Got10 : Sequence A = 10 on last two ticks Y = 0
  - Got101 : Sequence A = 101 on last three ticks Y = 1
  - Got11 : Sequence A = 11 on last two ticks Y = 0
  - Got110 : Sequence A = 110 on last three ticks Y = 1

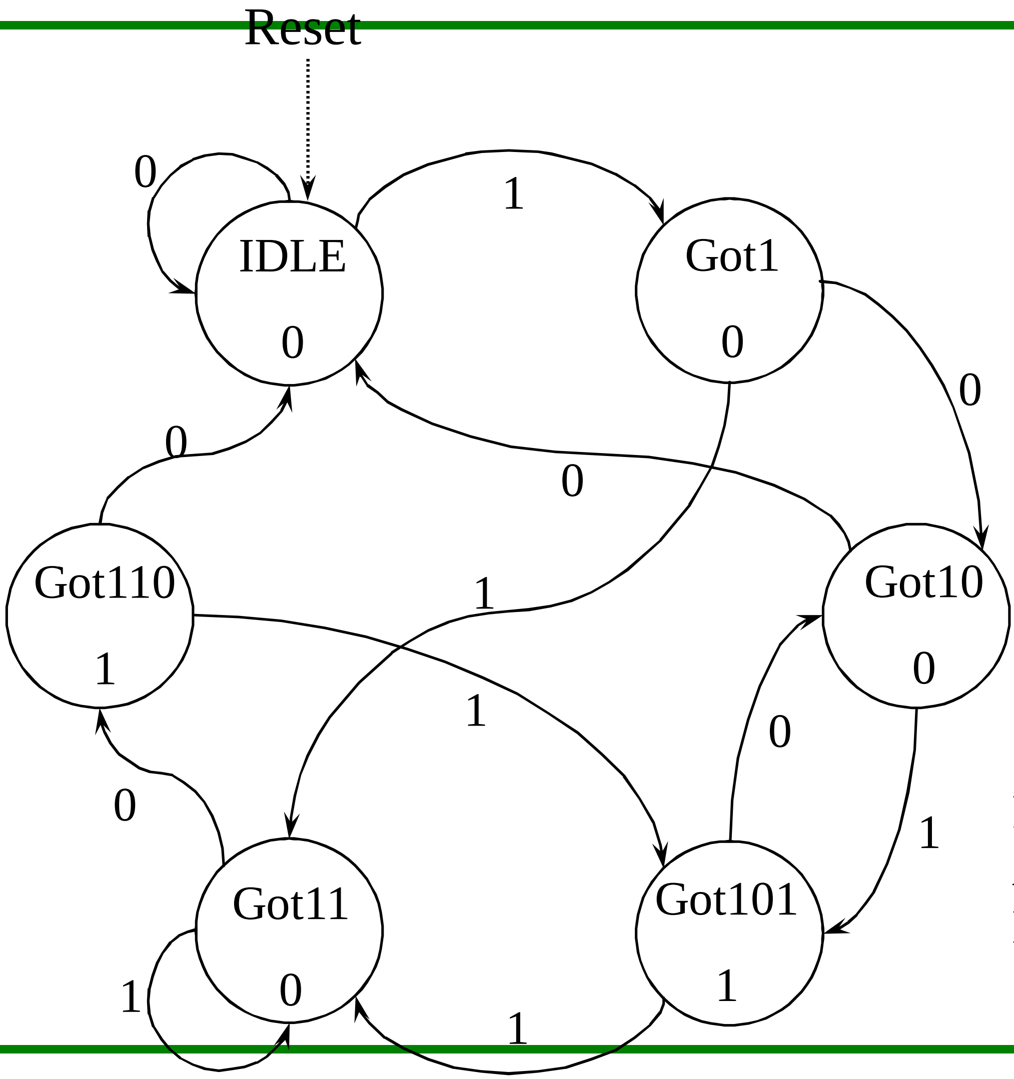


Design Example 2: State Table

	A		
S	0	1	Y
IDLE	IDLE	Got1	0
Got1	Got10	Got11	0
Got10	IDLE	Got101	0
Got101	Got10	Got11	1
Got11	Got110	Got11	0
Got110	IDLE	Got101	1

S(t+1)

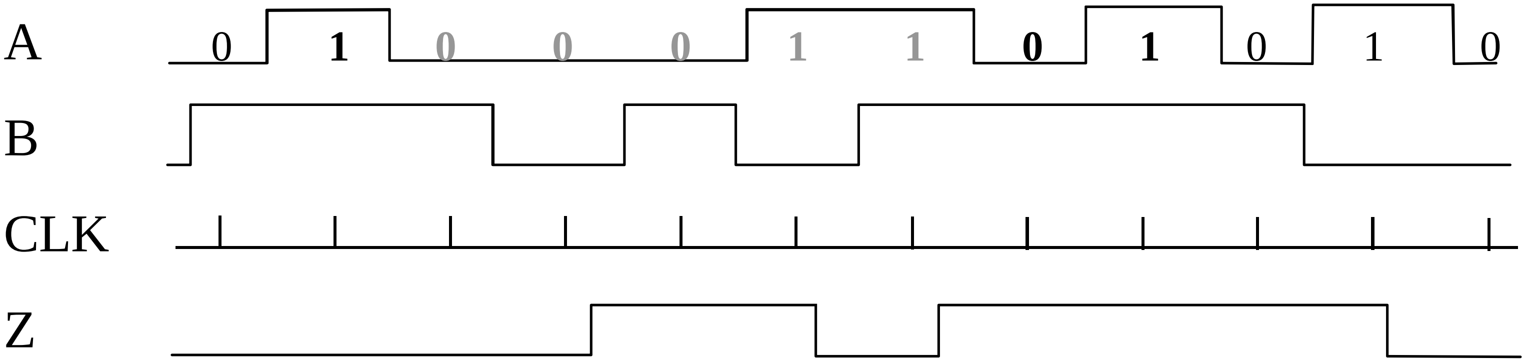
Design Example 2: State Diagram



Format:
Arc: input A
Node: state/output Y

Design Example 3: Interpretation

- Word description (input sequence detector)
  - Design a state machine with inputs A and B, and output Z.
  - Z = 1 if either:
    - A had the same value for both previous clock ticks or
    - B has been 1 ever since the first condition occurred
  - Else Z = 0
- Interpretation of word description (only rising clock ticks are shown)



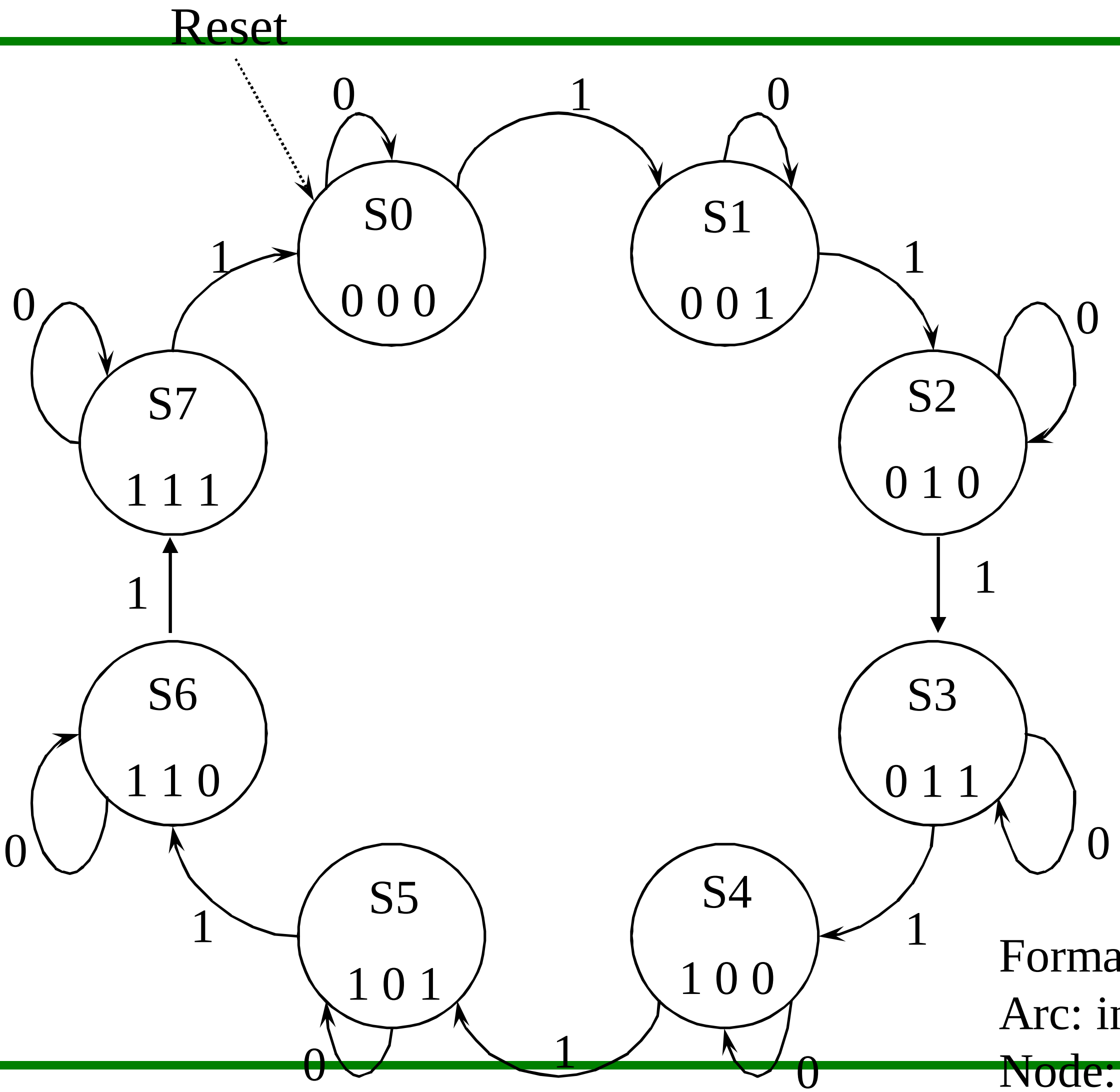
Design Example 4: State Table

- Word description (output sequence generator)
  - 3-bit counter with enable (0, 1, 2, 3, 4, 5, 6, 7, 0, 1 ...)

	EN		OUT		
S	0	1	C_2	C_1	C_0
S0	S0	S1	0	0	0
S1	S1	S2	0	0	1
S2	S2	S3	0	1	0
S3	S3	S4	0	1	1
S4	S4	S5	1	0	0
S5	S5	S6	1	0	1
S6	S6	S7	1	1	0
S7	S7	S0	1	1	1

S(t+1)

Design Example 4: State Diagram



Format:
Arc: input EN
Node: state/output C_2 C_1 C_0

Design Example 5: State Table

- Word description (output sequence generator)
  - Design state machine with input GO and BCD output code $B_3B_2B_1B_0$
  - Anytime GO = 1 at a clock tick and machine is IDLE, output the BCD sequence 1, 2, 5, 9 during the next 4 clock ticks (regardless of GO)
  - Return to IDLE with BCD = 0 until GO = 1 is next detected

State Table

	Go					
S	0	1	B_3	B_2	B_1	B_0
IDLE	IDLE	S1	0	0	0	0
S1	S2	S2	0	0	0	1
S2	S5	S5	0	0	1	0
S5	S9	S9	0	1	0	1
S9	IDLE	IDLE	1	0	0	1

S(t+1)