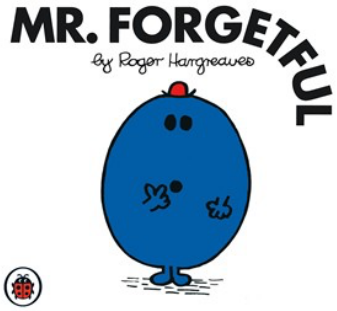
Logic Design (Part 4)
Sequential Logic - Latches (Chapter 3)

1

So Far: Combinational Logic



Combinational Logic:

- Always gives the same output for a given set of inputs
- Aka “state-less” (i.e., no “state” or “memory”)

Sequential Logic:

- Its output depends on its inputs & its last output!
- Forms the basis for “state” or “memory” for a computer

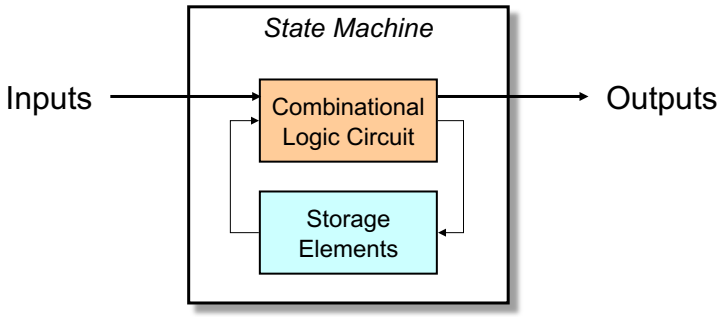
2

2

(Finite) State Machine

type of sequential circuit

- Combines combinational logic with storage
- "Remembers" state, and changes output (and state) based on inputs and current state



3

3

Sequential Logic: Starting point

- Where do we start: Build a device, using combinational logic devices, to store a value
  - RS Latch
    - Use this to build a more useful/easier to use latch – the D Latch
  - concept of memory
- Build it using the devices we have thus far
  - How ? Use "feedback" circuit
- What is the methodology behind design of sequential logic circuits
  - Finite State Machines
  - Example of Vending machine
- Combine sequential and combinational logic devices to "assemble" a simple processor!

4

4

Feedback Circuits

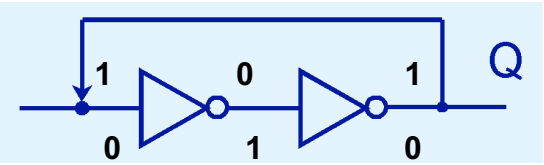
- What happens if we feed the output of a combinational logic circuit to an input in the circuit ?
  - This is the key to circuits that can store values!
- Stable circuit
  - Output point of circuit retains value indefinitely
- Unstable circuit
  - State that remains constant only for a duration of a few gate delays
- key takeaway: build stable circuits that can “hold” their value

5

5

Feedback Circuits

- To retain their state values, sequential circuits rely on *feedback.*
- Feedback in digital circuits occurs when an output is looped back to the input.
- A simple example of this concept is shown below.
  - If Q is 0 it will always be 0, if it is 1, it will always be 1.



6

6

Latches and Flip-Flops

- Latch: basic circuit for storage
  - Operate on changes in Level (i.e., 1 or 0)
- Flip-flop:
  - Sequential circuits take input from output of storage
  - Latches that work on change of level can lead to unstable sequential circuits
    - ➢ As level changes the outputs change --- inputs change!
  - Flip-Flop circuits designed to operate properly when they are part of a sequential circuit
  - Designed to work in a synchronized circuit with a Clock

7

7

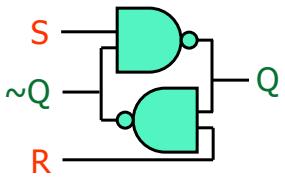
Most Basic Sequential Logic Circuit: *R-S Latch*

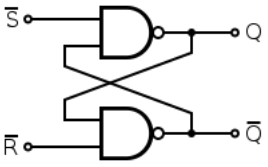
Most fundamental unit for static memory

- Has the ability to "store" its last output

R-S Latch – Cross-Coupled NAND gates

- Output of each NAND gate serves as input to the other
- Two inputs: S (*SET*) & R (*RESET*)
- Two outputs: Q and NOT(Q) Recall: NOT(Q)= ~Q = Q' = $\bar{Q}$
- Called a "Latch" because it can "Latch" onto data coming in





Another common way of drawing the same circuit

8

8

R-S Latch

- The R-S latch is a bi-stable circuit which means that it can happily exist in either of two stable states. Just like a see-saw.
- You can push the latch from one state to another by setting or resetting it with the S-R signals
- The logic levels are maintained because of the feedback paths from outputs to inputs.

9

9

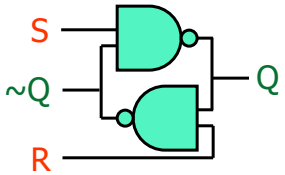
Most Basic Sequential Logic Circuit: *R-S Latch*

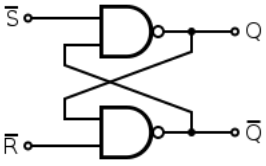
Most fundamental unit for static memory

- Has the ability to "store" its last output

R-S Latch – Cross-Coupled NAND gates

- Output of each NAND gate serves as input to the other
- Two inputs: S (*SET*) & R (*RESET*)
- Two outputs: Q and NOT(Q) Recall: NOT(Q)= ~Q = Q' = $\overline{Q}$
- Called a "Latch" because it can "Latch" onto data coming in





Another common way of drawing the same circuit

10

10

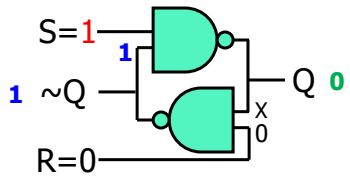
R-S Latch – Behavior/Truth Table

First, recall truth table for a NAND gate:

A	B	C
0	0	1
0	1	1
1	0	1
1	1	0

R-S Latch Operation:

- Best place to start is S=1, R=0



Next, look at top NAND gate
→Its inputs are: 1 and 1
Blue 1, comes from lower NAND
→Produces a 0 at its output
Therefore, when S=1, R=0
The output of latch is: Q=0, ~Q=1

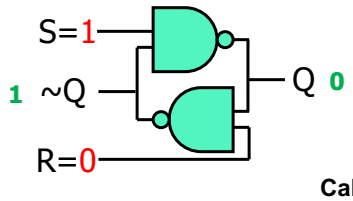
11

Most Basic Sequential Logic Circuit: *R-S Latch*

First, recall truth table for a NAND gate:

A	B	C
0	0	1
0	1	1
1	0	1
1	1	0

R-S Latch Operation:



Truth Table for R-S Latch:

ACTION	S	R	Q	~Q
	0	0		
	0	1		
RESET	1	0	0	1
	1	1		

Called the "RESET" action, as Q is set to 0
Also, notice: Q and ~Q opposite

12

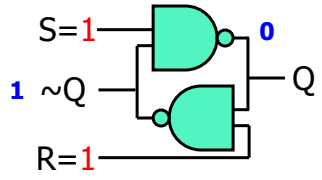
Most Basic Sequential Logic Circuit: *R-S Latch*

Truth table for a NAND gate:

A	B	C
0	0	1
0	1	1
1	0	1
1	1	0

Next, set S=1 R=1
Check value of Q

R-S Latch Operation:



Truth Table for R-S Latch:

ACTION	S	R	Q	~Q
	0	0		
SET	0	1	?	?
RESET	1	0	0	1
HOLD	1	1	-	-
HOLD	1	1	0	1

HOLD's last value on its outputs!
OUTPUT depends on input and last output

13

Most Basic Sequential Logic Circuit: *R-S Latch*

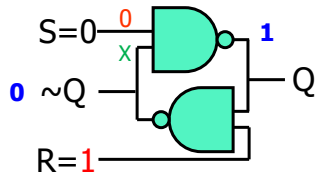
Truth table for a NAND gate:

A	B	C
0	0	1
0	1	1
1	0	1
1	1	0

Next, set S=0 R=1
Check value of Q

R-S Latch Operation:

- **Next input case is called the "SET", when inputs are: S=0, R=1**



1st look at upper NAND gate
→Its inputs are: 0 and X (anything)
→Produces a 1 at its output
Lower NAND gate
→Inputs are: 1 and 1
→Produces a 0 at its output

14

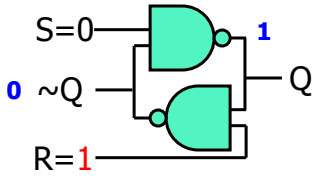
14

Most Basic Sequential Logic Circuit: *R-S Latch*

Truth table for a NAND gate:

A	B	C
0	0	1
0	1	1
1	0	1
1	1	0

R-S Latch Operation:



Truth Table for R-S Latch:

ACTION	S	R	Q	~Q
	0	0		
SET	0	1	1	0
RESET	1	0	0	1
	1	1		

***SET*s LATCH to have a "1" at the output**

15

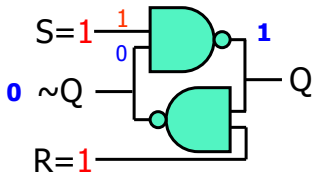
Most Basic Sequential Logic Circuit: *R-S Latch*

Truth table for a NAND gate:

A	B	C
0	0	1
0	1	1
1	0	1
1	1	0

R-S Latch Operation:

- **Last valid input case is the "HOLD" S=1, R=1**
  - **If we have just "SET" Latch, we will have Q=1, ~Q=0, already on outputs**



Upper NAND gate
- →Has S=1 & former value of ~Q=0
- →Produces a 1 at its output
  (same ~Q as when it started)

Lower NAND gate
- →Inputs are: 1 and 1
- →Produces a 0 at its output *(same Q)*

16

16

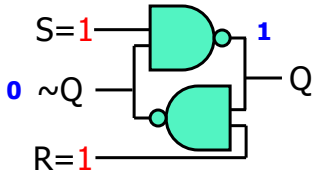
Most Basic Sequential Logic Circuit: *R-S Latch*

Truth table for a NAND gate:

A	B	C
0	0	1
0	1	1
1	0	1
1	1	0

Next, set S=1 R=1
Check value of Q

R-S Latch Operation:



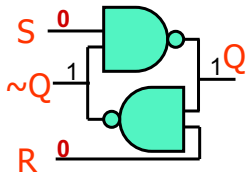
Truth Table for R-S Latch:

ACTION	S	R	Q	~Q
	0	0		
SET	0	1	1	0
RESET	1	0	0	1
HOLD	1	1	1	0

HOLD's value we "SET" last

17

Storage - Cross-Coupled NANDs (R-S Latch)



Set value of Q by R=0 or S=0
Store value of Q with R=1, S=1
What happens with S=0 and R=0?

- **Short answer: confusion**
- **Real circuits depend on both Q and ~Q**
- **Strange things may happen if both are 1**

ACTION	S	R	Q	~Q
ILLEGAL	0	0	1	1
SET	0	1	1	0
RESET	1	0	0	1
HOLD	1	1	1	0
HOLD	1	1	0	1

D-Latches shows a way to prevent the RS Latch from ever getting S=R=0 as its input

18

18

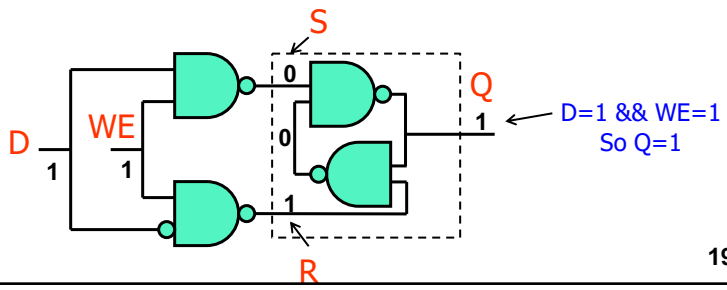
Gated D-Latch: Preventing "Illegal State" of RS Latch

Add logic to an R-S latch

- **Create a more convenient interface, prevent S=0 && R=0**

Two inputs: D (data) and WE (write enable)

- **When WE = 1, latch is set to value of D**
  - ➢ **S = NOT(D), R = D**



19

19

Circuits with Memory

- We now have a device (RS Latch) that is capable of storing a bit
  - It hold the previous value of Q
- but need to make sure the inputs are never R=S=0
- Modify the circuit to get a D latch

20

20

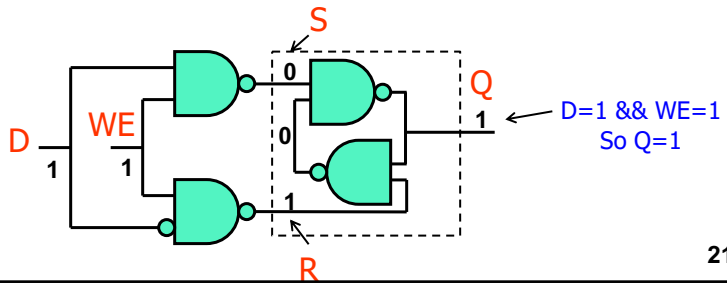
Gated D-Latch: Preventing "Illegal State" of RS Latch

Add logic to an R-S latch

- **Create a more convenient interface, prevent S=0 && R=0**

Two inputs: D (data) and WE (write enable)

- **When WE = 1, latch is set to value of D**
  - **S = NOT(D), R = D**



21

21

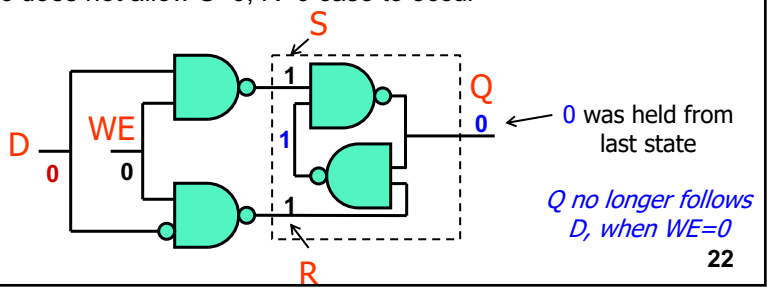
Gated D-Latch: Preventing "Illegal State" of RS Latch

Add logic to an R-S latch

- Create a more convenient interface, prevent S=0 && R=0

Two inputs: D (data) and WE (write enable)

- When WE = 1, latch is set to value of D
  - S = NOT(D), R = D
- When WE = 0, latch continues to hold previous value
  - S = R = 1 (hold condition for SR latch)
- Extra logic does not allow S=0, R=0 case to occur

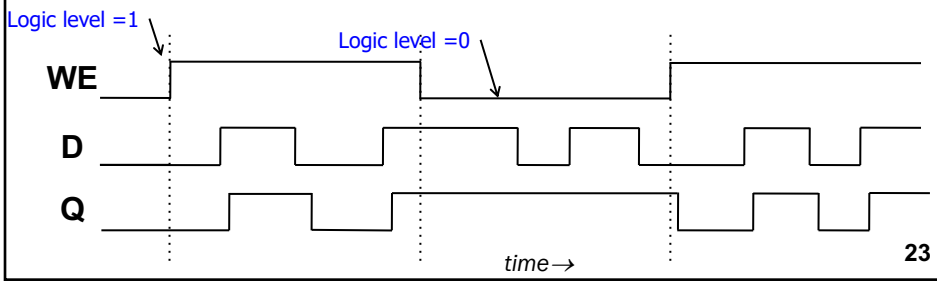


22

22

D-Latch Timing Diagram

- The diagram below is called a "Timing" Diagram
  - ➢ Our D-Latch is previous-state dependent
    - – We can think of this as a time dependency
    - – Moving to the right on diagram, represents forward moving time
  - ➢ The inputs & outputs to our D-Latch are on left
    - – Inputs/Outputs can be either "HIGH" (logic 1) or "LOW" (logic)
  - ➢ Think of this as a time-dependent truth table



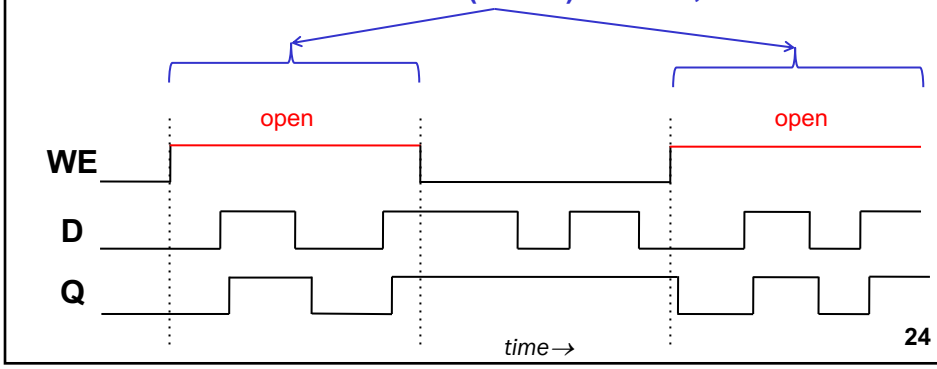
23

23

D-Latch Timing Diagram

- When the WE signal is high the latch is said to be open and the output signal, Q, follows the input signal, D.
  - ➢ As in any combinational circuit there will be a small delay between the time that the input changes and the time that the output follows suit.

When latch is *OPEN* (WE=1): Notice, Q follows D



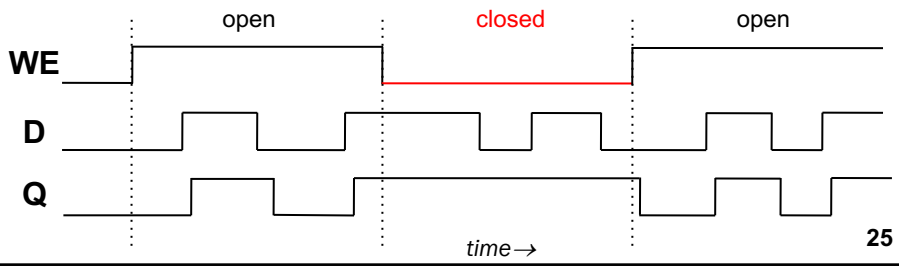
24

24

D-Latch Timing Diagram

- When the WE signal is low the latch is closed and the output signal, Q retains its value.

When latch is CLOSED (WE=0): Notice, Q doesn't follow D
(Instead, Q has previous value)

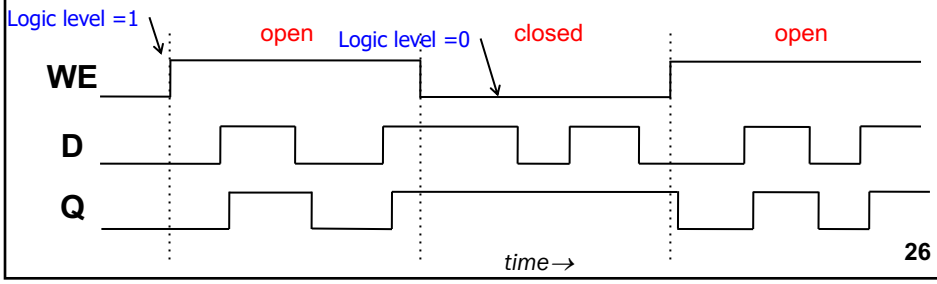


25

25

D-Latch Timing Diagram

- Setup / Hold Times
  - ➢ The input signal should (D) be stable a certain amount of time before the WE signal is set to CLOSED (WE=0)
    - – This is referred to as the SETUP time
  - ➢ In addition, the input signal (D) must be stable for a time after the WE is set to CLOSED (WE=0)
    - – This is referred to as HOLD time
  - ➢ Why? Time must be given for inputs to propagate through NAND gates! Gates are not instantaneous!



26

26

Next… Storage Devices

- Ok…we now have a device (D-Latch) that can store a bit

- Use this to build 'real' storage devices….
- Temporary storage in a computer ?
  - Where are variables stored before being sent to the arithmetic unit for operations on them?
- Register
  - Can we build an n-bit register using latches?
- What about "main" memory
- Disk
  - Later…

27

27