

Sequential Logic Circuits – Part I (Memory Element)

CIT 595
Spring 2008

Sequential Logic Circuits

- Output depends on stored information (current state) and may be on current inputs
 - Example:
 - > state = Score board of basketball game (number of points, time remaining, possession)
 - > input = which team scored the point
 - > output = point increase for the team that just scored
- Sequential Circuits are built out of combinational logic and one or more memory/storage elements
 - E.g. Registers, Memories, Counters, Control Unit
- So first lets see how the memory element is implemented?

CIT 595

2

1-Bit Memory Element Characteristics

Need a unit that can:

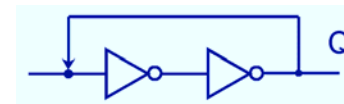
- Retain/Remember a single bit with possible values (state) i.e. 0 or 1
 - This will allow us to read a previous value stored
- Able to change the value (state). Since there's only a single bit, there are only two choices:
 - **Set** the bit to 1
 - **Reset**, or **clear**, the bit to 0

CIT 595

3

How do we make a storage unit out of logic gates?

- To remember/retain their state values, rely on concept of *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



A simple feedback circuit

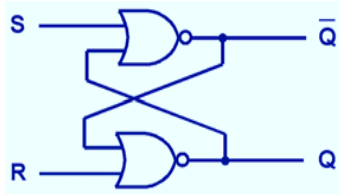
CIT 595

4

SR Latch

SR Latch: basic memory element (storage unit)

- Made out of two cross-coupled NOR gates
- The “SR” stands for set/reset and are inputs that can change the “state” of the circuit i.e. value to be stored
- Here Q and Q' are feed back into the circuit. They're not only outputs, they're also inputs!

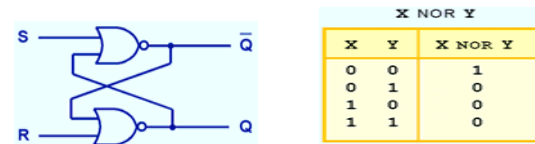


CIT 595

5

SR Latch Outputs - Q and Q'

- The two outputs are *taken* to be *complements* of each other
- This due property of NOR gate
 - 0 input acts as like inverter with respect to the other input
 - If one of the inputs is 1, the output is always 0
- So one of the NOR gates acts like inverter and while the other injects a 0 back into the feedback loop, depending on the value of R and S



Caveat: Assumption works as long as SR != 11

CIT 595

6

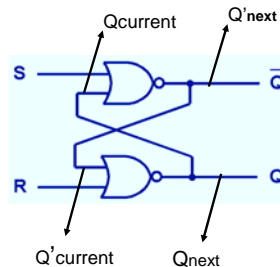
Output Equations for SR Latch

- To figure out how Q and Q' change, we have to look at not only the inputs S and R, but also the *current* values of Q and Q':

$$Q_{\text{next}} = (R + Q'_{\text{current}})'$$

$$Q'_{\text{next}} = (S + Q_{\text{current}})'$$

State/Characteristic Equation



CIT 595

7

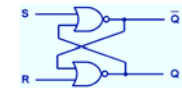
Input combination: SR = 00

- What if S = 0 and R = 0?
- The equations on the right reduce to:

$$Q_{\text{next}} = (0 + Q'_{\text{current}})' = Q_{\text{current}}$$

$$Q'_{\text{next}} = (0 + Q_{\text{current}})' = Q'_{\text{current}}$$

- So when SR = 00, then $Q_{\text{next}} = Q_{\text{current}}$
- Whatever value Q has, it keeps
- So we have unit that can retain values



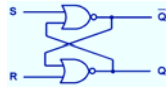
CIT 595

8

Input combination: SR = 10

- What if S = 1 and R = 0?

- Since S = 1, Q'_{next} is 0, *regardless of* $Q_{current}$
 $Q'_{next} = (1 + Q_{current})' = 0$



- This new value of Q' goes into the bottom NOR gate, along with R = 0
 $Q_{next} = (0 + 0)' = 1$

$$Q_{next} = (0 + 0)' = 1$$

- So when SR = 10, then $Q'_{next} = 0$ and $Q_{next} = 1$
- This is how you **set** the flip-flop to 1. The S input stands for "set"
- In a physical circuit, there will usually be a delay from the time S becomes 1 to the time Q_{next} becomes 1
- But once Q_{next} becomes 1, the outputs will stop changing
 - This is a "stable state"

CIT 595

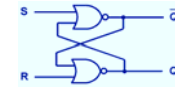
9

Input Combination: SR = 01

- What if S = 0 and R = 1?

- Since R = 1, Q_{next} is 0, *regardless of* $Q_{current}$:

$$Q_{next} = (1 + Q'_{current})' = 0$$



- This new value of Q goes into the top NOR gate, where S = 0
 $Q'_{next} = (0 + 0)' = 1$

$$Q'_{next} = (0 + 0)' = 1$$

- So when SR = 01, then $Q_{next} = 0$ and $Q'_{next} = 1$
- This is how you **reset**, or **clear**, the flip-flop to 0. The R input stands for "reset"
- Again, there will be delays before a change in R propagates to the output Q'_{next} , but eventually it will become stable

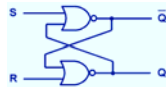
CIT 595

10

Input Combination: SR = 11

- Both Q_{next} and Q'_{next} will become 0

- $Q = Q' = 0$, which is *logically inconsistent*
- Violates an assumption



- What happens if we then make S = 0 and R = 0 together

$$Q_{next} = (0 + 0)' = 1$$

$$Q'_{next} = (0 + 0)' = 1$$

- But these new values go back into the NOR gates, and in the next step we get:

$$Q_{next} = (0 + 1)' = 0$$

$$Q'_{next} = (0 + 1)' = 0$$

- The circuit enters an infinite loop, where Q and Q' cycle between 0 and 1 forever
- This is actually the worst case, but the moral is don't ever set SR=11!

CIT 595

11

SR Latch Summarized in Tabular Form

Present State			Next State
S	R	Q(t)	Q(t+1)
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	undefined
1	1	1	undefined

Truth Table

S	R	Q(t+1)
0	0	Q(t) (no change)
0	1	0 (reset to 0)
1	0	1 (set to 1)
1	1	undefined

Characteristic Table

$Q(t)$ = current/present state = $Q_{current}$

$Q(t + 1)$ = next state = Q_{next}

CIT 595

12

Asynchronous vs. Synchronous Sequential Circuits

- **Asynchronous**
 - State changes are abrupt
 - Become active the moment any input changes
 - E.g. SR Latch
- **Synchronous**
 - State changes are controlled by a "Clock"
 - Clock allows ordering of events
 - Parts of the computer are made of synchronous sequential circuit components

CIT 595

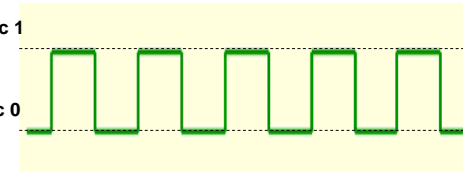
13

Clock

- Clocks produce electrical waveforms such as the one shown below
 - Each pulse has a precise width
 - There is a precise interval between pulses – known as clock cycle time

High = Logic 1

Low = Logic 0

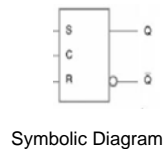


CIT 595

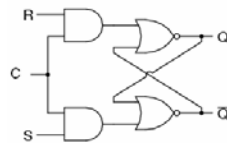
14

Gated SR Latch

- To avoid state change abruptly, the clock is used as control variable
- When clock (C) = 0, the state of the SR Latch is unaffected (since inputs are 00)
 - Values of SR will not affect as it is gated by the AND gate
- When clock = 1, the circuit's function as described on SR Latch
 - The inputs and current state will accordingly change the output



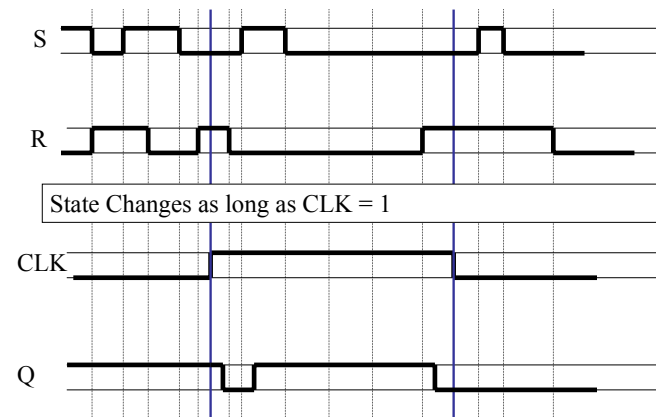
Symbolic Diagram



CIT 595

15

Gated SR Latch

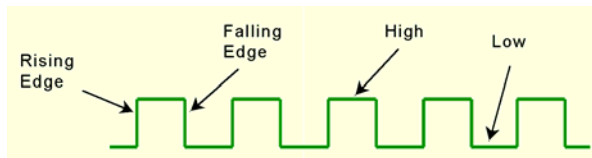


CIT 595

16

Level-Triggered vs. Edge Triggered

- *Level-triggered circuits* change state when the clock voltage reaches its highest or lowest level
 - E.g. Gated SR Latch
- Circuits that change state on the rising edge, or falling edge of the clock pulse are called *edge-triggered*
 - The storage elements are then known as Flip-Flops



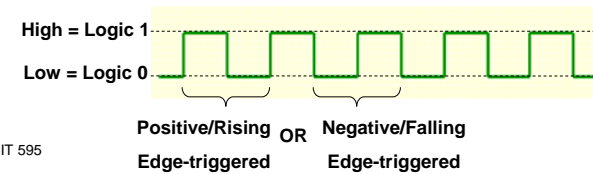
Note: Rising a.k.a Positive and Falling a.k.a Negative

CIT 595

17

Advantage of Flip-Flop

- The problem with *Latch* is that state keeps changing according to the values of the input during the period when the clock is active (i.e. clock = 1)
- But to allow complete synchronization in the system, we can restrict the change in state only at the edges (rising or falling) of the clock
 - This allows changes within defined intervals i.e. within a clock cycle

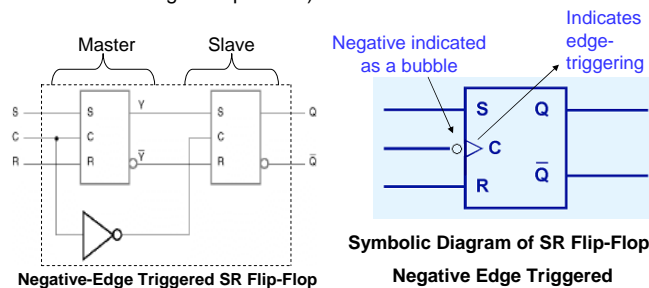


CIT 595

18

SR Flip-Flop: Master-Slave SR Latch Negative Edge-Triggered

- When C (clock) = 1, master is enabled and stores new data, slave restores old data (as slave gets C' i.e. 0)
- When C = 0, master's state passes to enabled slave (master not sensitive to change in inputs SR) and slave stores the new value

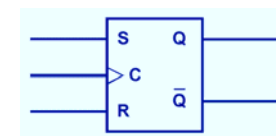


CIT 595

19

SR Flip-Flop Positive Edge-Triggered

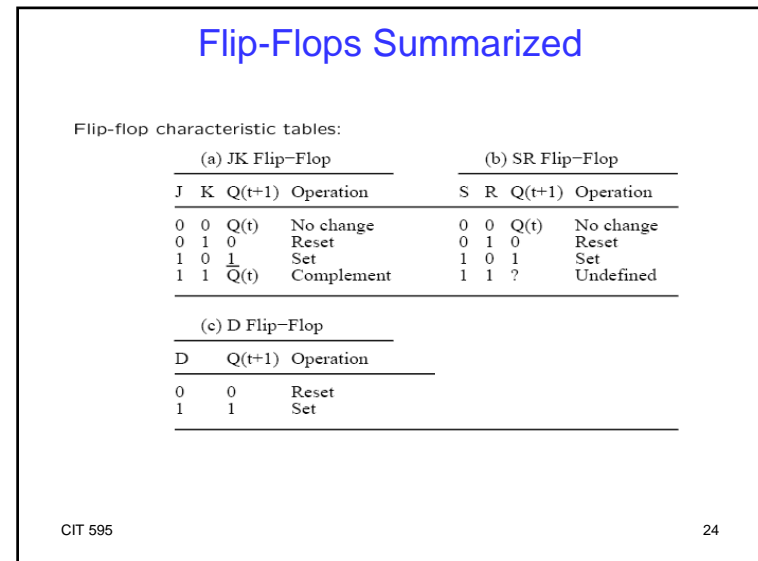
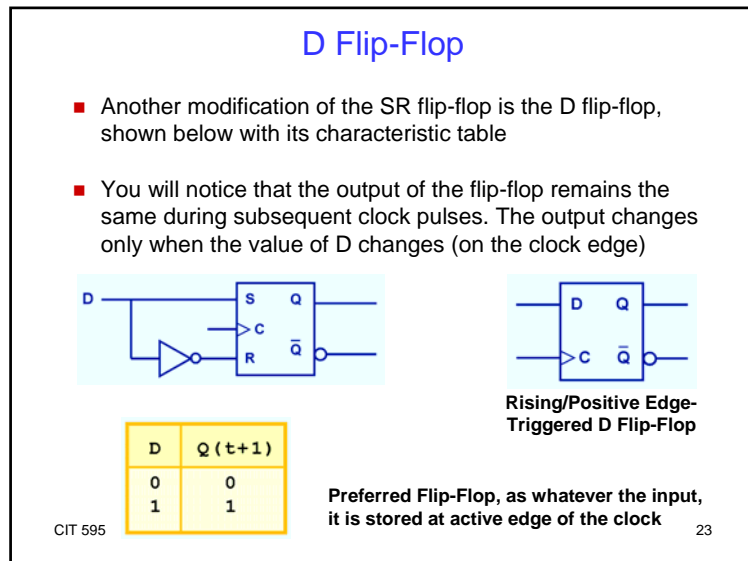
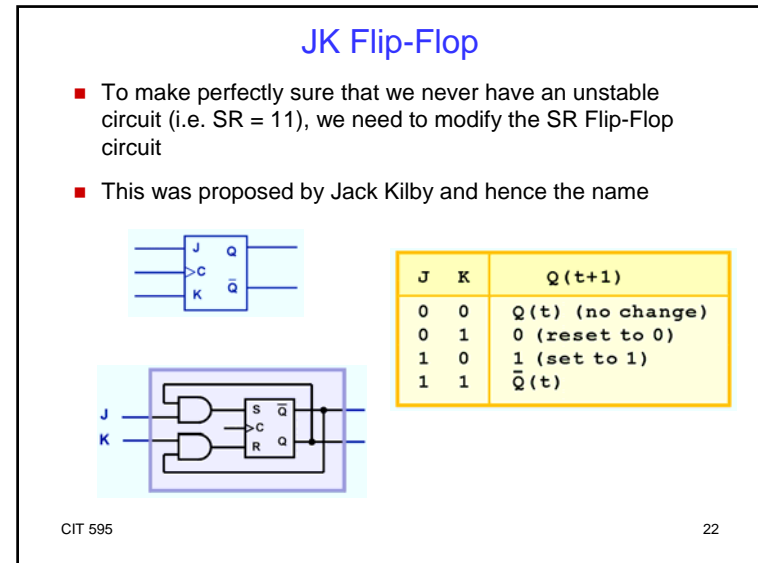
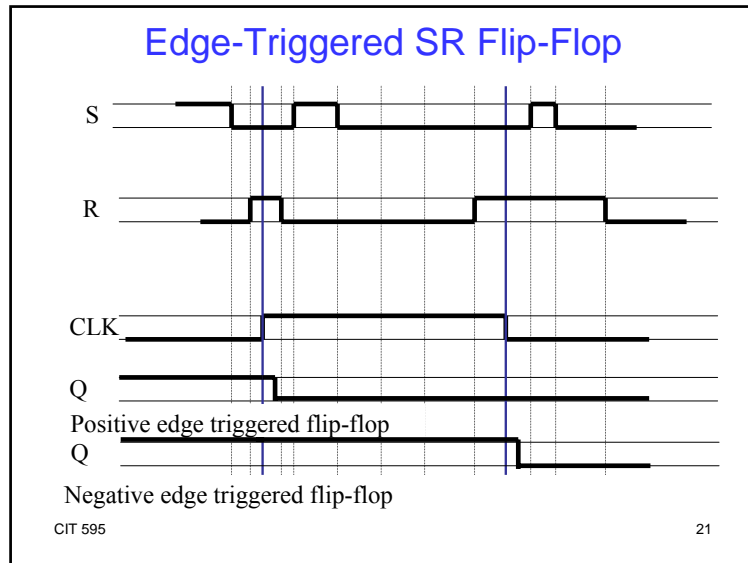
- Again Master Slave Configuration
- But Master gets complement of Clock i.e. C'
- Whereas Slave is directly connected Clock i.e. C



Symbolic Diagram of SR Flip-Flop Positive Edge-Triggered

CIT 595

20



Building blocks of Sequential Circuit

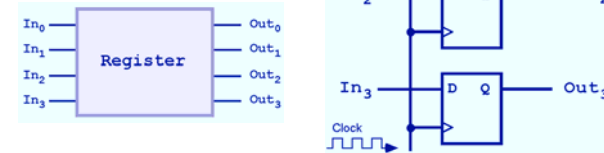
- Flip-flops are used to implement complex sequential circuits along with combinational logic
- Examples: Register, Memory, and up to an entire Control Unit

CIT 595

25

Example: 4-Bit Register

- The inputs are updated only on active clock edge, this allows synchronous update of all bits



Is this positive or negative edge-triggered??

CIT 595

26

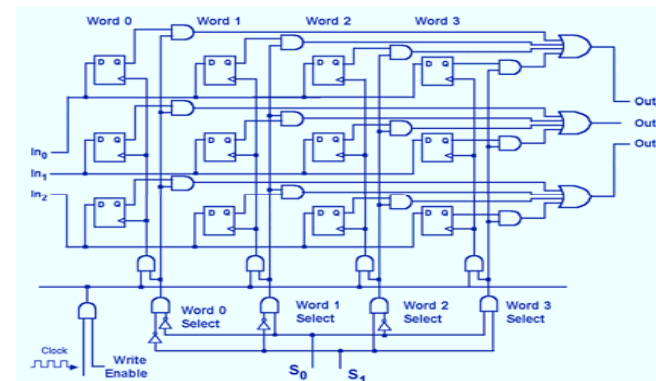
Example 2: 4 x 3 Memory

- The memory has 4 locations and can store 3 bits of information
- To represent 4 locations we need only 2-bits for address: address decoder performs the address decoding i.e. selects a particular *row or word* in memory
- To store information we use D Flip-Flop for each bit (total 12, as each location as 3 bits and we have 4 total locations)
- We also need select variable to chose if we want to read on write data and that is combined with clock signal (using some combinational logic)
- We also need some other combinational logic...

CIT 595

27

Example 2: 4 x 3 Memory



Write Enable = 1 allows memory write

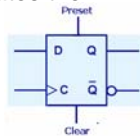
Write Enable = 0 allows memory read

CIT 595

28

Bringing the Flip-Flop to a known state

- Flip-flops discussed so far are called *synchronous* because the transfer of the data from the input to the output lines are synchronized with the triggering edge of the clock pulse
- Most integrated circuit flip-flops also have *asynchronous* inputs that affect state of the flip-flop *independent of the clock*
- E.g. D Flip-Flop with “Preset” and “Clear” inputs, If the D Flip-Flop is made out NOR cross coupled gates then
 - Preset = 1, and Clear = 0, then flop is set to 1
 - Preset = 0 and Clear = 1, then flop is cleared/reset to 0



CIT 595