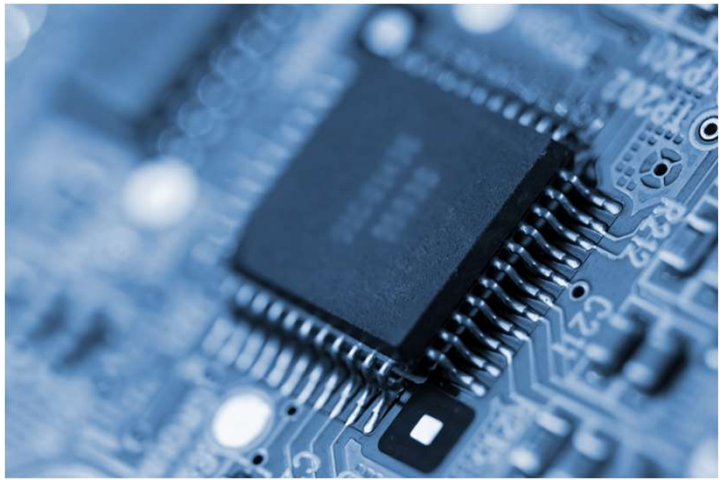
UNIVERSITY OF LEEDS

Computer Processors

Sequential Logic & Memory



Lecture 7/8

Sequential Logic

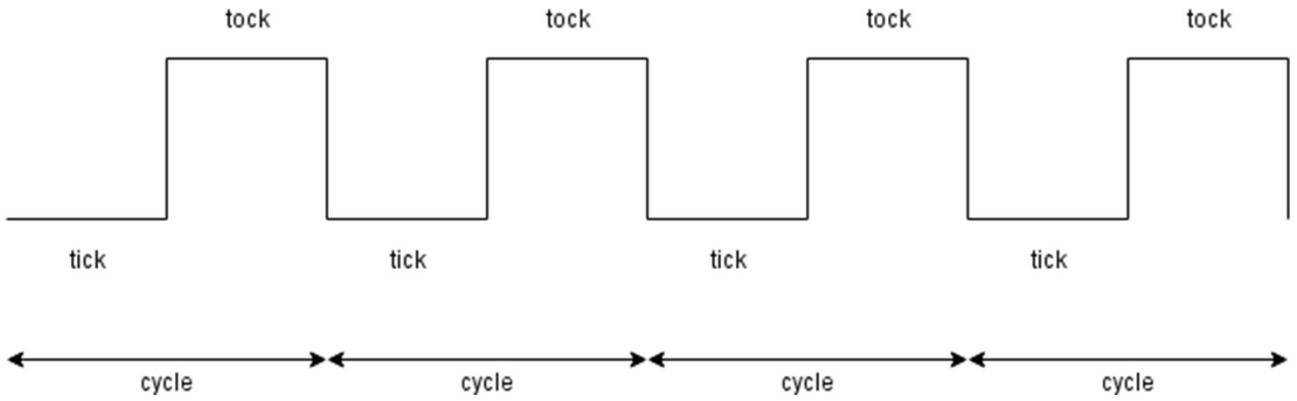
UNIVERSITY OF LEEDS

- Up to now all of the logic design has been combinatorial, that is the functions have no concept of time or ordering
- If a logic circuit computes a particular output it will always output that output
- No way of storing values - No Memory!!!
- Sequential Logic is Boolean Logic that has a concept of time
    - It can store values

Clocks

UNIVERSITY OF LEEDS

- In most computers the passage of time is represented by a master clock
- A clock has two phases tick/tock (or low/high)
- The frequency of the clock, amongst other things, determine the speed of computation
- The waveform need not be regular it is the transition that is important



Memory specification

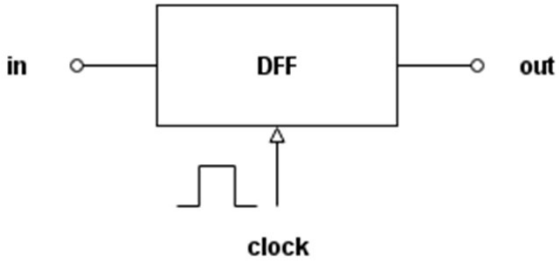
UNIVERSITY OF LEEDS

- We want a logic circuit such that

  out(t) = in(t-1)

  where t is a discrete time interval

- The output of the gate at time t is dependent on the input of the gate at the previous time step (t-1)
- Functionality implemented by a Data Flip Flop (DFF)



t	in	out
0	0	?
1	1	0
2	1	1
3	0	1
4	1	0

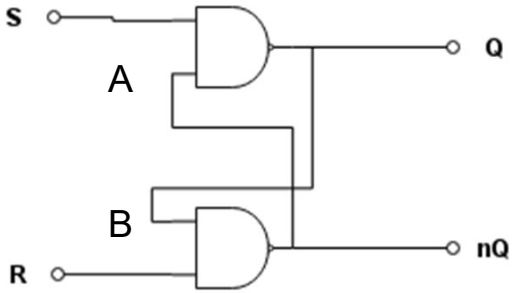
D-type flip flops (DFF)

UNIVERSITY OF LEEDS

- **Q** and **nQ** are a complementary pair
- From the truth table we can see that there is capability to remember a bit

Problems

- S=0, R=0 - Q and nQ are not complementary
- S=1, R=1 - An undetermined state
- No clock

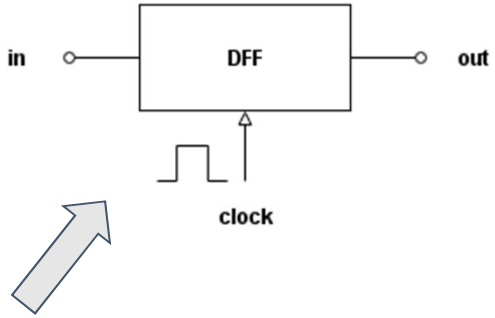


SR Flip Flop

	S	R	Q	nQ	Comment
Reset	0	1	1	0	Q is set to 1 by 0 on S
	1	1	1	0	1 on Q is remembered
Set	1	0	0	1	Q is set to 0 by 0 on R
	1	1	0	1	0 on Q is remembered
	0	0	1	1	Undesirable state

D-type flip flops (DFF)

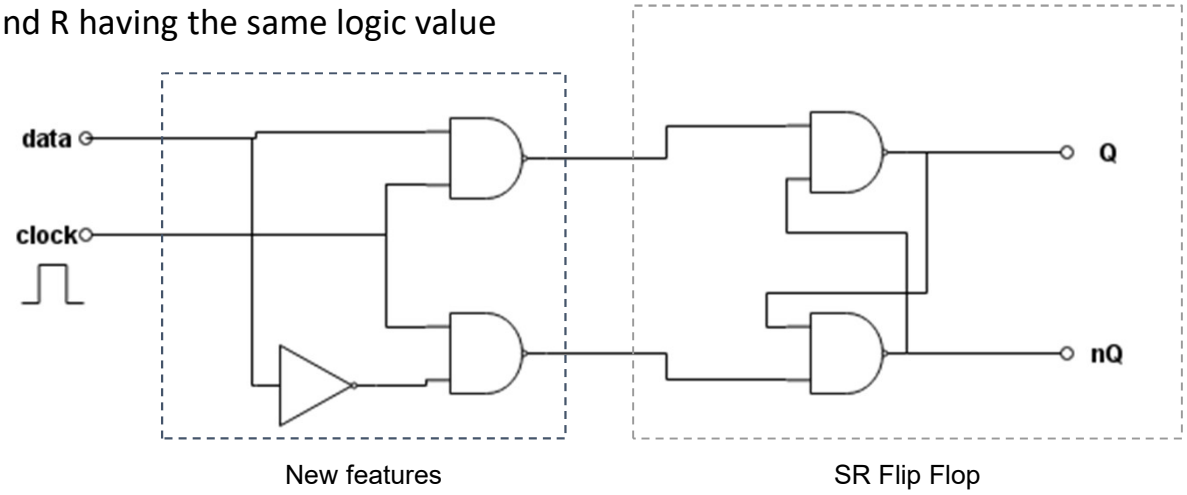
UNIVERSITY OF LEEDS



- Store should only occur when clock cycles

  (low-high)

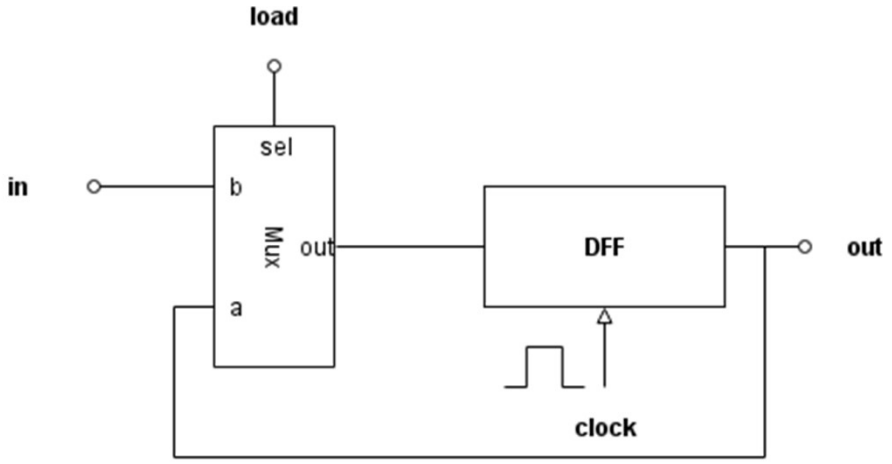
- Prevent S and R having the same logic value



1-bit memory

UNIVERSITY OF LEEDS

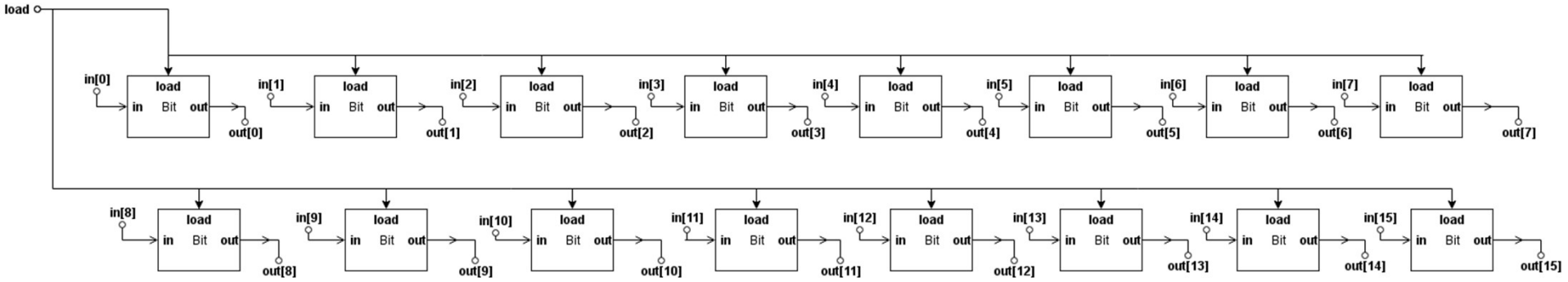
- D-type Flip Flop can store a single bit, but they have no mechanism to read/write
- **Mux** allows for the selection of storing either the currently stored value or the value at **in**
  - Read - value from **out**
  - Write - put bit on **in** and set **load**



Registers

UNIVERSITY OF LEEDS

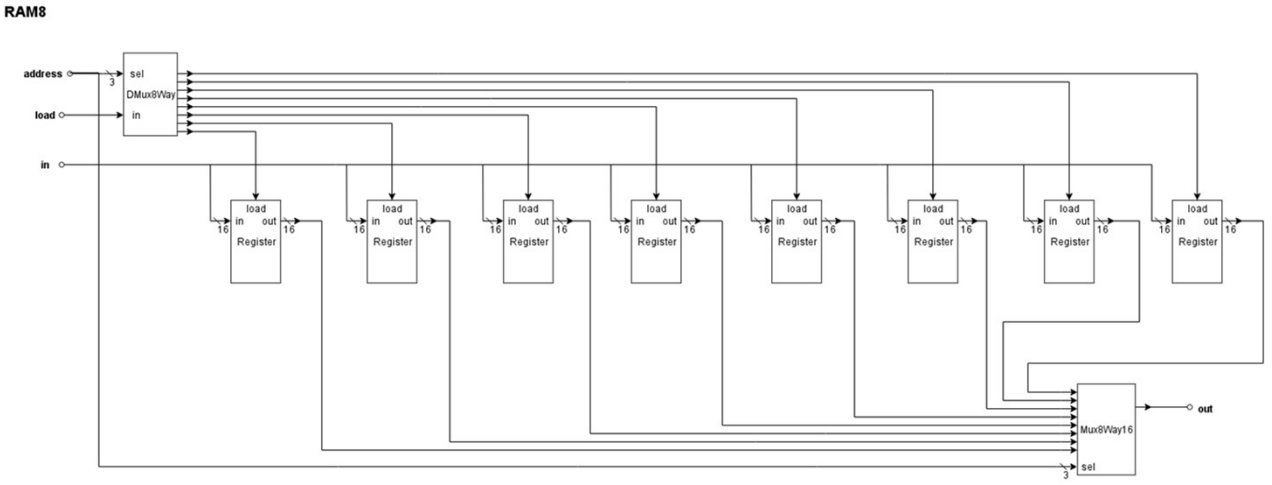
- A register is a collection of bits
- The width of a register is dependent on the architecture
  - Our architecture is 16-bit
- Share a common **load** control bit



Random Access Memory

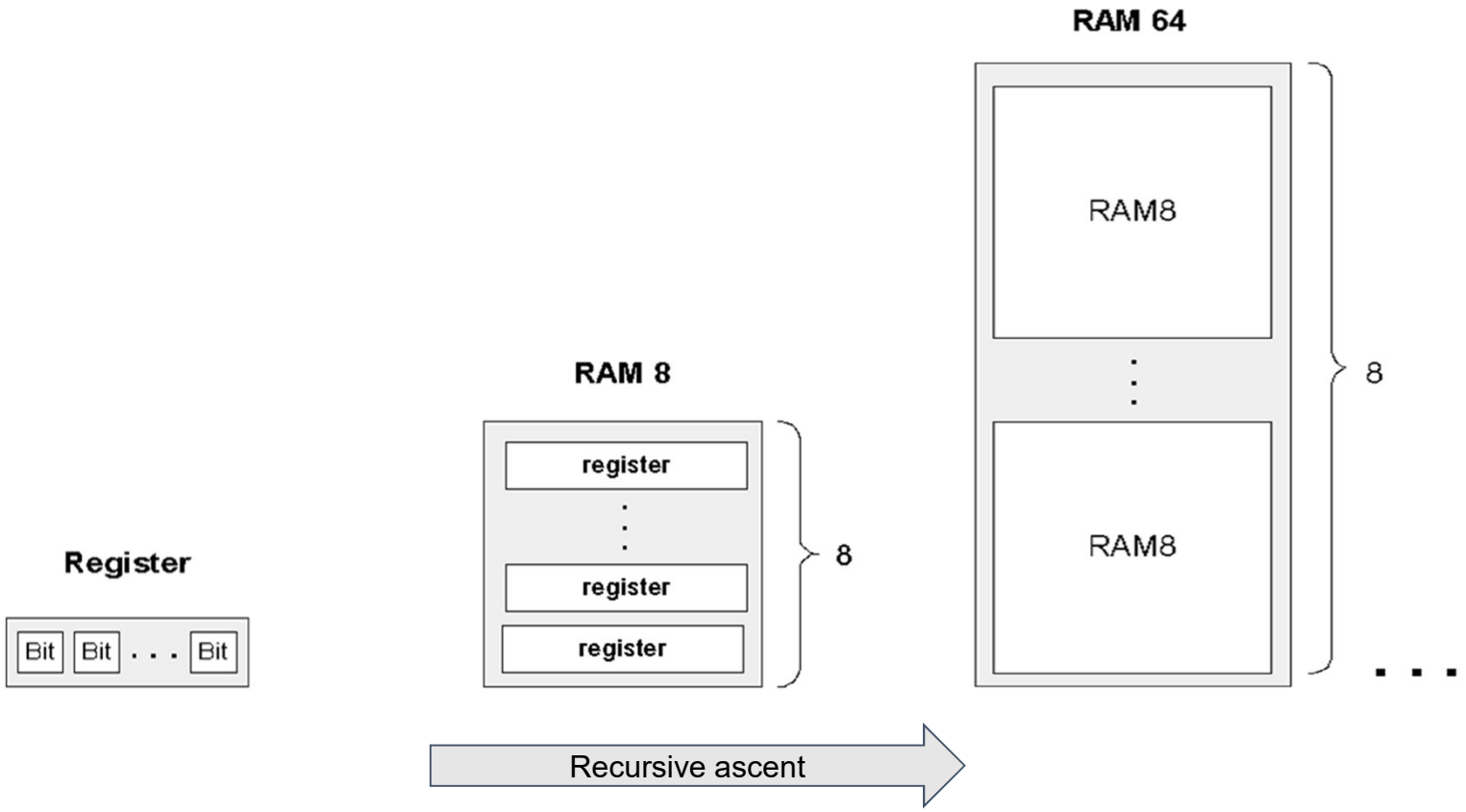
UNIVERSITY OF LEEDS

- Random Access Memory (RAM) is a type of memory where access to arbitrary positions is possible at uniform cost
- Memory locations are numbered from 0 to n often where n is a power of 2



Random Access Memory

UNIVERSITY OF LEEDS



Random Access Memory

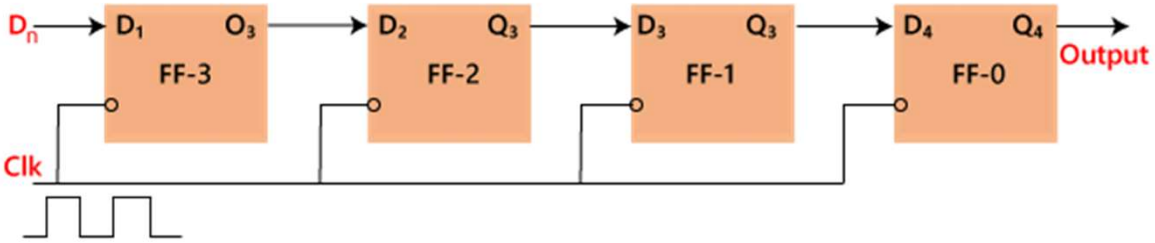
UNIVERSITY OF LEEDS

- With an n-bit address 2^n locations can be addressed
- RAM8 - constructed from 8 registers
- RAM64 - constructed from 8 RAM8's
- RAM512 - constructed from 8 RAM64's
- RAM4k - constructed from 8 RAM512's
- So on...

Shift Registers (Serial In Serial Out)

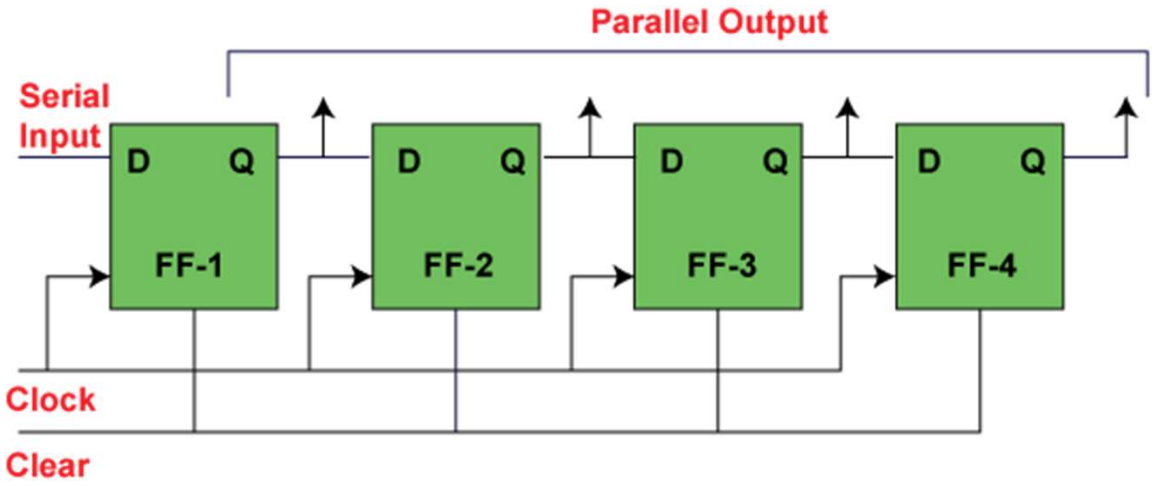
UNIVERSITY OF LEEDS

Time	In	w	x	y	z
0	0	0	0	0	0
1	1	0	0	0	0
2	0	1	0	0	0
3	1	0	1	0	0
4	1	1	0	1	0
5	0	1	1	0	1
6	0	0	1	1	0
7	0	0	0	1	1
8	0	0	0	0	1
9	0	0	0	0	0



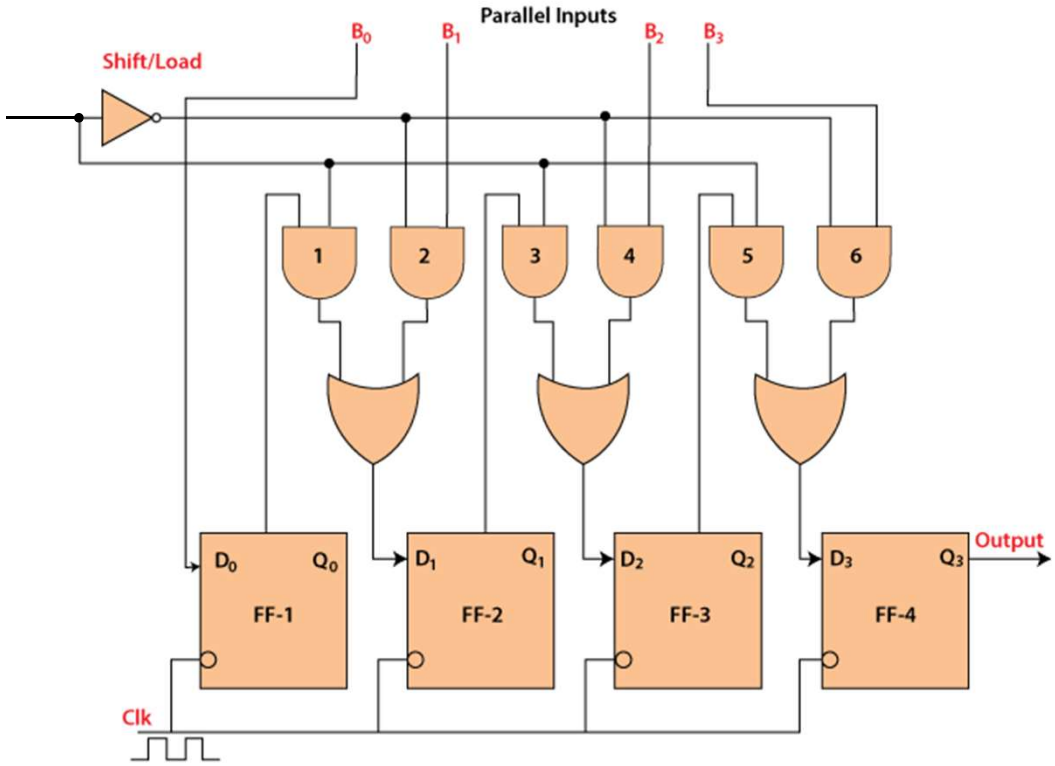
Serial IN Parallel OUT (SIPO)

UNIVERSITY OF LEEDS



Parallel IN Serial OUT

UNIVERSITY OF LEEDS



Counters

UNIVERSITY OF LEEDS

- Counters are a combination of

  combinatorial and sequential logic
- Counters implement
  - Increment by 1
  - Set to some specified value
  - Reset

Counters are used to keep track of where we are

in a program.

Present State Q3Q2Q1Q0	Next State Q3Q2Q1Q0	D3	D2	D1	D0
0000	0001	0	0	0	1
0001	0010	0	0	1	0
0010	0011	0	0	1	1
0011	0100	0	1	0	0
0100	0101	0	1	0	1
0101	0110	0	1	1	0
0110	0111	0	1	1	1
0111	1000	1	0	0	0
1000	1001	1	0	0	1
1001	1010	1	0	1	0
1010	1011	1	0	1	1
1011	1100	1	1	0	0
1100	1101	1	1	0	1
1101	1110	1	1	1	0
1110	1111	1	1	1	1
1111	0000	0	0	0	0

D0	Q1'Q0' 00	Q1'Q0 01	Q1Q0 11	Q1Q0' 10
Q3'Q2' 00				
Q3'Q2 01				
Q3Q2 11				
Q3Q2' 10				

D1	Q1'Q0' 00	Q1'Q0 01	Q1Q0 00	Q1Q0' 10
Q3'Q2'	0	1	0	1
Q3'Q2	0	1	0	1
Q3Q2	0	1	0	1
Q3Q2'	0	1	0	1

Present State Q3Q2Q1Q0	Next State Q3Q2Q1Q0	D3	D2	D1	D0
0000	0001	0	0	0	1
0001	0010	0	0	1	0
0010	0011	0	0	1	1
0011	0100	0	1	0	0
0100	0101	0	1	0	1
0101	0110	0	1	1	0
0110	0111	0	1	1	1
0111	1000	1	0	0	0
1000	1001	1	0	0	1
1001	1010	1	0	1	0
1010	1011	1	0	1	1
1011	1100	1	1	0	0
1100	1101	1	1	0	1
1101	1110	1	1	1	0
1110	1111	1	1	1	1
1111	0000	0	0	0	0

D2	Q1'Q0' 00	Q1'Q0 01	Q1Q0 11	Q1Q0' 10
Q3'Q2' 00	1	0	0	1
Q3'Q2 01	1	0	0	1
Q3Q2 11	1	0	0	1
Q3Q2' 10	1	0	0	1

D2	Q1'Q0' 00	Q1'Q0 01	Q1Q0 11	Q1Q0' 10
Q3'Q2' 00				
Q3'Q2 01				
Q3Q2 11				
Q3Q2' 10				

Present State Q3Q2Q1Q0	Next State Q3Q2Q1Q0	D3	D2	D1	D0
0000	0001	0	0	0	1
0001	0010	0	0	1	0
0010	0011	0	0	1	1
0011	0100	0	1	0	0
0100	0101	0	1	0	1
0101	0110	0	1	1	0
0110	0111	0	1	1	1
0111	1000	1	0	0	0
1000	1001	1	0	0	1
1001	1010	1	0	1	0
1010	1011	1	0	1	1
1011	1100	1	1	0	0
1100	1101	1	1	0	1
1101	1110	1	1	1	0
1110	1111	1	1	1	1
1111	0000	0	0	0	0

D0	Q1'Q0' 00	Q1'Q0 01	Q1Q0 11	Q1Q0' 10
Q3'Q2' 00				
Q3'Q2 01				
Q3Q2 11				
Q3Q2' 10				

D1	Q1'Q0' 00	Q1'Q0 01	Q1Q0 00	Q1Q0' 10
Q3'Q2'	0	1	0	1
Q3'Q2	0	1	0	1
Q3Q2	0	1	0	1
Q3Q2'	0	1	0	1

Present State Q3Q2Q1Q0	Next State Q3Q2Q1Q0	D3	D2	D1	D0
0000	0001	0	0	0	1
0001	0010	0	0	1	0
0010	0011	0	0	1	1
0011	0100	0	1	0	0
0100	0101	0	1	0	1
0101	0110	0	1	1	0
0110	0111	0	1	1	1
0111	1000	1	0	0	0
1000	1001	1	0	0	1
1001	1010	1	0	1	0
1010	1011	1	0	1	1
1011	1100	1	1	0	0
1100	1101	1	1	0	1
1101	1110	1	1	1	0
1110	1111	1	1	1	1
1111	0000	0	0	0	0

D2	Q1'Q0' 00	Q1'Q0 01	Q1Q0 11	Q1Q0' 10
Q3'Q2' 00	1	0	0	1
Q3'Q2 01	1	0	0	1
Q3Q2 11	1	0	0	1
Q3Q2' 10	1	0	0	1

D2	Q1'Q0' 00	Q1'Q0 01	Q1Q0 11	Q1Q0' 10
Q3'Q2' 00				
Q3'Q2 01				
Q3Q2 11				
Q3Q2' 10				

Summary

UNIVERSITY OF LEEDS

- Introduced sequential logic
- Introduced the concept of clocks
- Demonstrated an implementation of a D-type Flip Flop
- Demonstrated a construction of Memory
- Introduced the concept of a counter

Interested in sequential logic? Digital Logic Design - Chapter 4

Counters

UNIVERSITY OF LEEDS