Digital Electronics: Sequential Logic

Applications of Flip-Flops

Counters

- A clocked sequential circuit that goes through a predetermined sequence of states
- A commonly used counter is an n-bit binary counter. This has n FFs and 2^n states which are passed through in the order 0, 1, 2, ….2^n-1, 0, 1, .
- Uses include:
  - Counting
  - Producing delays of a particular duration
  - Sequencers for control logic in a processor
  - Divide by m counter (a divider), as used in a digital watch

Memories

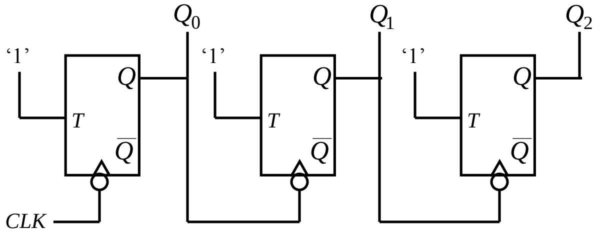
- For example,
  - Shift register
    - Parallel loading shift register : can be used for parallel to serial conversion in serial data communication
    - Serial in, parallel out shift register: can be used for serial to parallel conversion in a serial data communication system.

Counters

- In most books you will see 2 basic types of counters, namely *ripple* counters and *synchronous* counters
- In this course we are concerned with synchronous design principles. Ripple counters do not follow these principles and should generally be avoided if at all possible. We will now look at the problems with ripple counters

Ripple Counters

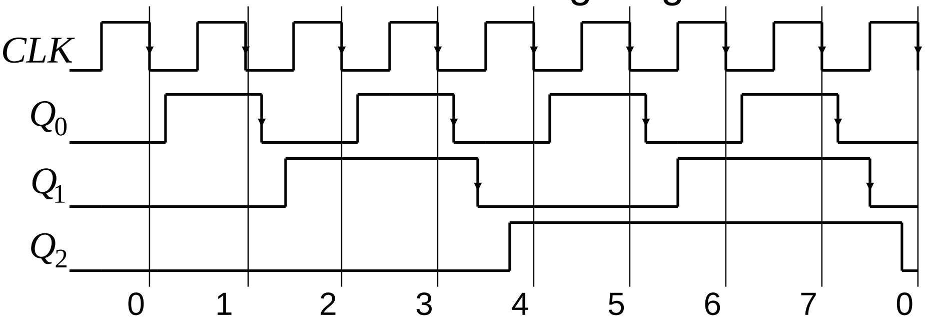
- A ripple counter can be made be cascading together negative edge triggered T-type FFs operating in ‘toggle’ mode, i.e., T =1



- See that the FFs are not clocked using the same clock, i.e., this is **not** a synchronous design. This gives some problems….

Ripple Counters

- We will now draw a timing diagram



- Problems:

See outputs do not change at the same time, i.e., synchronously. So hard to know when count output is actually valid.

Propagation delay builds up from stage to stage, limiting maximum clock speed before miscounting occurs.

Ripple Counters

- If you observe the frequency of the counter output signals you will note that each has half the frequency, i.e., double the repetition period of the previous one. This is why counters are often known as dividers
- Often we wish to have a count which is not a power of 2, e.g., for a BCD counter (0 to 9).To do this:
  - use FFs having a Reset/Clear input
  - Use an AND gate to detect the count of 10 and use its output to Reset the FFs

Synchronous Counters

- Owing to the problems identified with ripple counters, they should not usually be used to implement counter functions
- It is recommended that *synchronous* counter designs be used
- In a synchronous design
  - all the FF clock inputs are directly connected to the clock signal and so all FF outputs change at the same time, i.e., *synchronously*
  - more complex combinational logic is now needed to generate the appropriate FF input signals (which will be different depending upon the type of FF chosen)

Synchronous Counters

- We will now investigate the design of synchronous counters
- We will consider the use of D-type FFs only, although the technique can be extended to cover other FF types.
- As an example, we will consider a 0 to 7 up-counter

Synchronous Counters

- To assist in the design of the counter we will make use of a modified *state transition table.* This table has additional columns that define the required FF inputs (or e*xcitation* as it is known)
  – Note we have used a state transition table previously when determining the state diagram for an RS latch
- We will also make use of the so called '*excitation table*' for a D-type FF
- First however, we will investigate the so called *characteristic table* and *characteristic equation* for a D-type FF

Characteristic Table

- In general, a characteristic table for a FF gives the next state of the output, i.e.,Q' in terms of its current state Q and current inputs

Q	D	Q'
0	0	0
0	1	1
1	0	0
1	1	1

Which gives the characteristic equation,

$$Q' = D$$

i.e., the next output state is equal to the current input value

Since Q' is independent of Q the characteristic table can be rewritten as

D	Q'
0	0
1	1

Excitation Table

- The characteristic table can be modified to give the excitation table. This table tells us the required FF input value required to achieve a particular next state from a given current state

Q	Q'	D
0	0	0
0	1	1
1	0	0
1	1	1

As with the characteristic table it can be seen that Q', does not depend upon, Q , however this is not generally true for other FF types, in which case, the excitation table is more useful. Clearly for a D-FF,

$$D = Q'$$

Characteristic and Excitation Tables

- Characteristic and excitation tables can be determined for other FF types.
- These should be used in the design process if D-type FFs are not used
- For example, for a J-K FF the following tables are appropriate:

Characteristic and Excitation Tables

J	K	Q'
0	0	Q
0	1	0
1	0	1
1	1	$\overline{Q}$

Truth table

Q	Q'	J	K
0	0	0	x
0	1	1	x
1	0	x	1
1	1	x	0

Excitation table

- We will now determine the modified state transition table for the example 0 to 7 up-counter

Modified State Transition Table

- In addition to columns representing the current and desired next states (as in a conventional state transition table), the modified table has additional columns representing the required FF inputs to achieve the next desired FF states

Modified State Transition Table

- For a 0 to 7 counter, 3 D-type FFs are needed

Current state	Next state	FF inputs
$Q_2Q_1Q_0$	$Q_2'Q_1'Q_0'$	$D_2D_1D_0$
0 0 0	0 0 1	0 0 1
0 0 1	0 1 0	0 1 0
0 1 0	0 1 1	0 1 1
0 1 1	1 0 0	1 0 0
1 0 0	1 0 1	1 0 1
1 0 1	1 1 0	1 1 0
1 1 0	1 1 1	1 1 1
1 1 1	0 0 0	0 0 0

The procedure is to:

Write down the desired count sequence in the current state columns

Write down the required next states in the next state columns

Fill in the FF inputs required to give the defined next state

Note: Since $Q' = D$ (or $D = Q'$) for a D-FF, the required FF inputs are identical to the Next state

Synchronous Counter Example

- If using J-K FFs for example, we need J and K input columns for each FF
- Also note that if we are using D-type FFs, it is not necessary to explicitly write out the FF input columns, since we know they are identical to those for the next state
- To complete the design we now have to determine appropriate combinational logic circuits which will generate the required FF inputs from the current states
- We can do this from inspection, using Boolean algebra or using K-maps.

Synchronous Counter Example

Current state	Next state	FF inputs
$Q_2Q_1Q_0$	$Q_2^{'}Q_1^{'}Q_0^{'}$	$D_2D_1D_0$
0 0 0	0 0 1	0 0 1
0 0 1	0 1 0	0 1 0
0 1 0	0 1 1	0 1 1
0 1 1	1 0 0	1 0 0
1 0 0	1 0 1	1 0 1
1 0 1	1 1 0	1 1 0
1 1 0	1 1 1	1 1 1
1 1 1	0 0 0	0 0 0

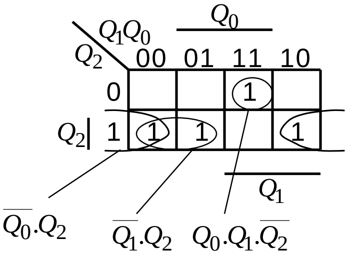
By inspection,

$$D_0 = \overline{Q_0}$$

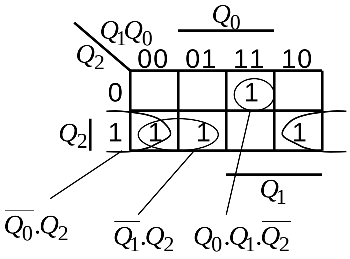
Note: FF_0 is toggling

Also, $D_1 = Q_0 \oplus Q_1$

Use a K-map for D_2,



Synchronous Counter Example

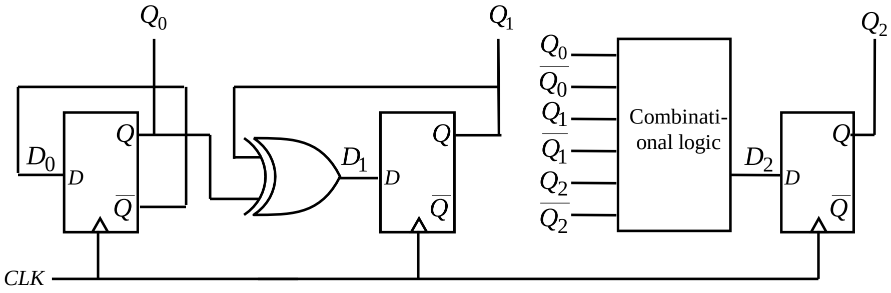


	00	01	11	10
0			1	
1	1	1		1

So,

$$D_2 = \overline{Q_0}.Q_2 + \overline{Q_1}.Q_2 + Q_0.Q_1.\overline{Q_2}$$

$$D_2 = Q_2.(\overline{Q_0}. + \overline{Q_1}) + Q_0.Q_1.\overline{Q_2}$$

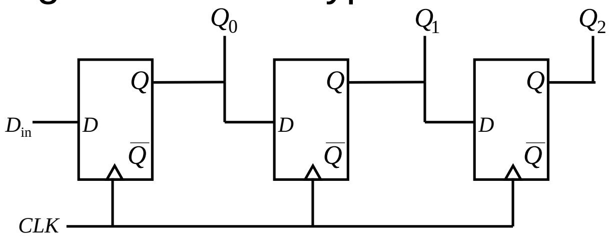


Synchronous Counter

- A similar procedure can be used to design counters having an arbitrary count sequence
  - – Write down the state transition table
  - – Determine the FF excitation (easy for D-types)
  - – Determine the combinational logic necessary to generate the required FF excitation from the current states – **Note:** remember to take into account any unused counts since these can be used as don’t care states when determining the combinational logic circuits

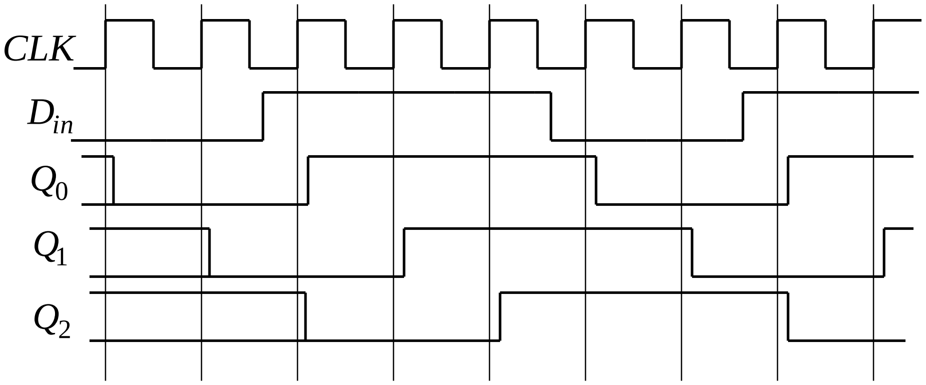
Shift Register

- A shift register can be implemented using a chain of D-type FFs



- Has a serial input, D_{in} and parallel output Q_0, Q_1 and Q_2.

Shift Register

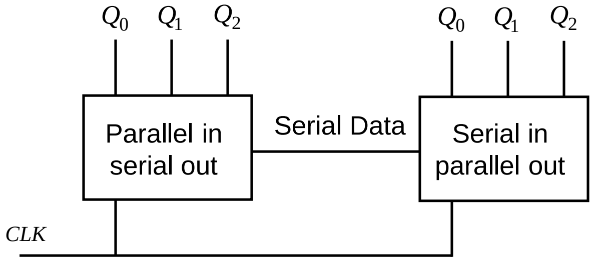


- See data moves one position to the right on application of each clock edge

Shift Register

- Preset and Clear inputs on the FFs can be utilised to provide a parallel data input feature
- Data can then be clocked out through Q_2 in a serial fashion, i.e., we now have a parallel in, serial out arrangement
- This along with the previous serial in, parallel out shift register arrangement can be used as the basis for a serial data link

Serial Data Link



- One data bit at a time is sent across the serial data link
- See less wires are required than for a parallel data link