CS1026 – Digital Logic Design

Sequential Logic Analysis

Shane Sheehan [1]

[1]ADAPT
Trinity College Dublin

February 20, 2017

Today's Overview

How to make good async logic I

A couple of rules to avoid logic hazards and critical races!

- Remember the SR-Latch!

How to make good async logic II

Rules of the game:

1. One external input signal changes at a time
2. Before the next external signal is allowed to change, the circuit must be given time to reach a new stable state

How to make good async logic III

Applying this to an SR Latch

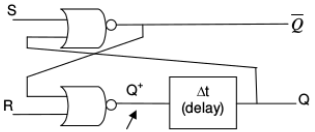
Step 1. Write the compressed characteristic table ⇨ Step 2. Draw the compressed K-map ⇨ Step 3. Draw the schematic (since Q is used for Q+, we need a delay element)

S	R	Q+
0	0	Q
0	1	0
1	0	1
1	1	0

SR	00	01	11	10
	$\boxed{Q}$	0	0	1

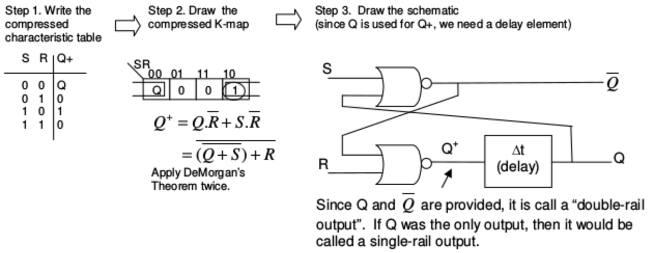
$$Q^{+} = Q.\overline{R} + S.\overline{R}$$
$$= \overline{\overline{(Q+S)} + R}$$

Apply DeMorgan's Theorem twice.



Since Q and $\overline{Q}$ are provided, it is call a "double-rail output". If Q was the only output, then it would be called a single-rail output.

How to make good async logic IV



Step 1. Write the compressed characteristic table

S	R	Q+
0	0	Q
0	1	0
1	0	1
1	1	0

Step 2. Draw the compressed K-map

SR	00	01	11	10
	Q	0	0	1

$$Q^{+} = Q.\overline{R} + S.\overline{R}$$

$$= \overline{(\overline{Q}+\overline{S}) + R}$$

Apply DeMorgan's Theorem twice.

Step 3. Draw the schematic (since Q is used for Q+, we need a delay element)

Since Q and $\overline{Q}$ are provided, it is call a "double-rail output". If Q was the only output, then it would be called a single-rail output.

Note: This circuit still encounters a critical race condition:

- when SR transitions from 11 to 00.

Designing a Sync Circuit I

We design a clock circuit!

Be careful of Jargon

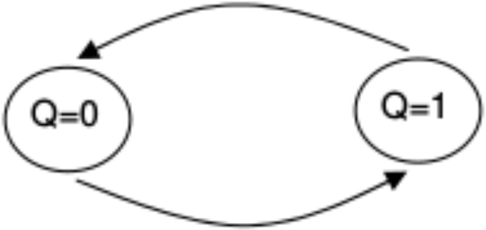
Sometimes we call this simple asynchronous sequential logic

- Which I find unnecessarily confusing!

Designing a Sync Circuit II

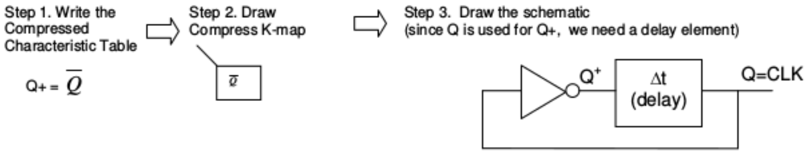
Let's build a NOT "buffer".. First we need:

- A state diagram



Designing a Sync Circuit III

Use the same design process:



Step 1. Write the Compressed Characteristic Table

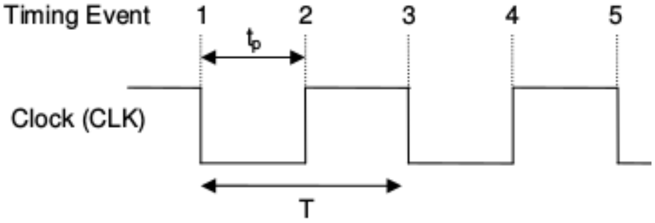
$Q+ = \overline{Q}$

Step 2. Draw Compress K-map

Step 3. Draw the schematic (since Q is used for Q+, we need a delay element)

Designing a Sync Circuit IV

A timing diagram can help you visualise things



Designing a Sync Circuit V

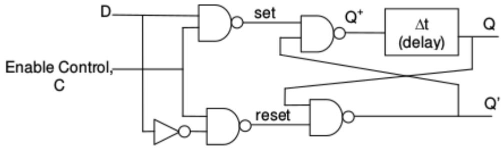
In practice..

- We use crystal oscillators and multipliers for the clock
  - It gives a precise frequency

Note: Clocks can go too fast!

The other flip flops I

D-Flip Flop (basically a modified SR flip flop):

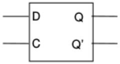


Practical Notes

- The most commonly used flip-flops due to its simplicity
- D flip-flop does not have an inherent critical race

The other flip flops II

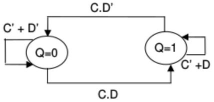
D-Flip Flop



D flip-flop Symbol

C	D	Q+
0	0	Q
0	1	Q
1	0	0
1	1	1

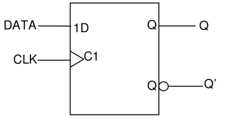
Compressed Characteristic Table



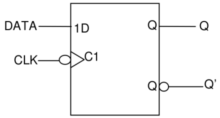
State Diagram

The other flip flops III

Block diagrams really help keep things simple



Positive-Edge-Triggered D Flip-Flop
(triangle called dynamic indicator)

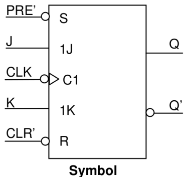


Negative-Edge-Triggered D Flip-Flop
(Bubbled triangle indicator)

Note postively and negatively tiggered cases

The other flip flops IV

JK-Flip Flop



Characteristic Table

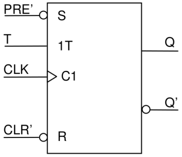
J	K	Q^+	Comment
0	0	Q	no change
0	1	0	reset condition
1	0	1	set condition
1	1	Q'	toggle

Characteristic equation: $Q^+ = J.Q' + K'.Q$

J and K inputs called *excitation inputs*

The other flip flops V

T-Flip Flop



Symbol

Characteristic Table

T	Q^+	Comment
0	Q	no Change
1	Q'	toggle

Characteristic equation: $Q^+ = T'.Q + T.Q'$

How to analyse a Sync Circuit I

Flip-flops used to design circuits with feedback:

- E.g. counters, shift registers, sequence detectors and controllers.

Feedback systems are classified as synchronous when all changes are synchronized with the system clock.

Feedback systems that do not use the system clock and change as the input change are called asynchronous.

How to analyse a Sync Circuit II

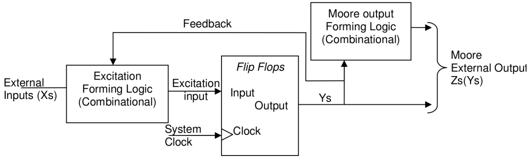
We prefer synchronous systems over asynchronous systems since they will not have synchronization issues.

Synchronous state machines may be implemented in one of three models based on the characteristic of its output:

- Moore or Mealy Sync Finite State Machines

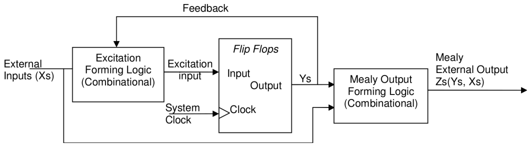
How to analyse a Sync Circuit III

Moore-type Synchronous Finite State Machines represent the function the machine state: $Z1(Y1, Y2, Y3)$
[Coudert and Madre, 2003]



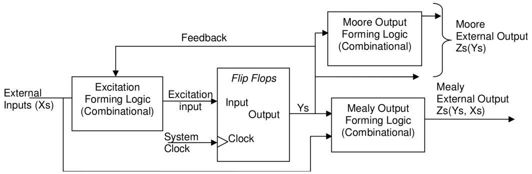
How to analyse a Sync Circuit IV

Mealy-type Synchronous Finite State Machines denote a function of the state of the machine and external inputs:
$Z1(Y1, Y2, \ldots X1, X2, \ldots)$ [Sarray et al., 2015]


Feedback
External Inputs (Xs)
Excitation Forming Logic (Combinational)
Excitation input
Flip Flops
Input
Output
System Clock
Clock
Ys
Mealy Output Forming Logic (Combinational)
Mealy External Output Zs(Ys, Xs)

How to analyse a Sync Circuit V

We can also have mixed types :-X


Feedback
Moore Output Forming Logic (Combinational)
Moore External Output Zs(Ys)
External Inputs (Xs)
Excitation Forming Logic (Combinational)
Excitation input
Flip Flops
Input
Output
Clock
System Clock
Ys
Mealy Output Forming Logic (Combinational)
Mealy External Output Zs(Ys, Xs)

But we don't worry about them too much

References (Homework) I

Coudert, O. and Madre, J. C. (2003).
A unified framework for the formal verification of sequential circuits.
In *The Best of ICCAD*, pages 39–50. Springer.

Sarray, I., Ressouche, A., Gaffé, D., Tigli, J.-Y., and Lavirotte, S. (2015).
Safe composition in middleware for the internet of things.
In *Proceedings of the 2nd Workshop on Middleware for Context-Aware Applications in the IoT*, pages 7–12. ACM.