Chapter 5
Sequential Systems

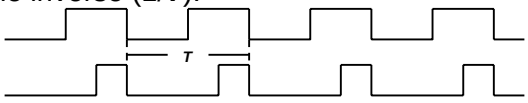
- Latches and Flip-flops
- Synchronous Counter
- Asynchronous Counter

Introduction

- Up to now everything has been combinational – the output at any instant of time depends only on what inputs are at the time.
- Later on of this course: *sequential systems* – systems that have memory. Thus, the output will depend not only on the present input but also on the past history – what has happened earlier.

Clock Signals

- Two versions of a clock signal are as below. In the first, the clock signal is 0 half of the time and 1 half of the time. In the second, it is 1 for a shorter part of the cycle.
- The period of the signal (***T*** on the diagram) is the length of one cycle. The frequency is the inverse (1/***T***).

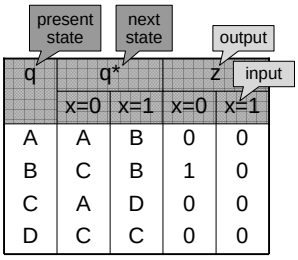


Terminology

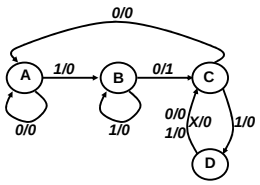
- *State*: what is stored in memory.
- *State table*: shows for each input combination and each state, what the output is and what the next state is – what is to be stored in memory after the next clock.
- *State diagram* (or *state graph*): a graphical representation of the state table.

  ⇒ (Finite) State Machine

State Table and State Diagram



State table



State diagram

- **The next state** is a function of the present state and the input.
- **The output** also depends on the present state (and on the input). It may change on a clock transition, but it may change where the input changes, as well.
- In **state diagram**, there must be one path from each state for each possible input combination.

Moore vs. Mealy Models

- **Moore model circuit (state-based)** → the outputs depend on the present state of the system but not on the inputs.
- **Mealy model circuit (input-based)** → the outputs depend on the inputs as well as the present state of the system.

- **Moore:** A system with no input and three outputs, that represent a number from 0 to 7, such that the outputs cycle through the sequence 0 3 2 4 1 5 7 and repeat on consecutive clock inputs.
- **Mealy:** A system with two inputs, X_1 and X_2, and three outputs, Z_1, Z_2 and Z_3, that represent a number from 0 to 7, such that the output counts up if X_1=0 and down if X_1=1 and recycles if X_2=0 and saturates if X_2=1. Thus, the following output sequences might be seen:

- X_1=0 X_2=0 0 1 2 3 4 5 6 7 0 1 2 3 4 5 6 7..
- X_1=0 X_2=1 0 1 2 3 4 5 6 7 7 7 7 7 7..
- X_1=1 X_2=0 7 6 5 4 3 2 1 0 7 6 5 4 3 2 1 0..
- X_1=1 X_2=1 7 6 5 4 3 2 1 0 0 0 0 0 0..

Design Process of Sequential Systems

- Table 5.1 Page 338
  - State table
  - Timing trace
  - State table with binary states
  - Truth table
  - K-map
  - equations

Design Process of Sequential Systems

State table

q	q*		z	
	x=0	x=1	x=0	x=1
A	A	B	0	0
B	C	B	1	0
C	A	D	0	0
D	C	C	0	0

Timing trace

clk	1	2	3	4	5	6	7	8	9	10	11	12
x	0	1	1	0	1	0	1	0	1	1		
q	A	A	B	B	C	D	C	D	C	D	C	
z	0	0	0	1	0	0	0	0	0	0	0	0

Design Process of Sequential Systems

State assignment

q	q_1	q_2
A	0	0
B	0	1
C	1	0
D	1	1

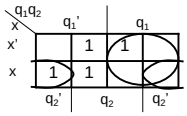
State table with binary states

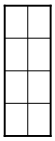
q_1	$q_1^*q_2^*$		z	
	x=0	x=1	x=0	x=1
00	00	01	0	0
01	10	01	1	0
10	00	11	0	0
11	10	10	0	0

Design Process of Sequential Systems

Truth table for system design

x	q_1	q_2	q_1*	q_2*	z
0	0	0	0	0	0
0	0	1	1	0	1
0	1	0	0	0	0
0	1	1	1	0	0
1	0	0	0	1	0
1	0	1	0	1	0
1	1	0	1	1	0
1	1	1	1	0	0





$$q^* = S + \overline{R}q$$

State assignment

Latches and Flip-flops

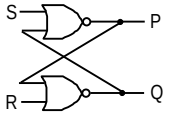
- A latch is a binary storage device, composed of two or more gates, with feedback.

The latch can store either
a 0 (Q = 0 and P = 1) or
a 1 (Q = 1 and P = 0)

The P output is just labelled $\overline{Q}$

Latches (cont.)

- Example: a latch constructed with 2 NORs.



The equations for this system:

$P = \overline{S + Q}$ and $Q = \overline{R + P}$

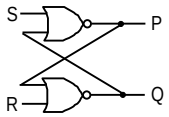
Normal storage stage $\rightarrow$ both inputs inactive (S = R = 0).

$P = \overline{Q}$ and $Q = \overline{P}$

Latches (cont.)

Case 1: If S=1 and R=0

$$P = \overline{(1+Q)} = \bar{1} = 0$$
$$Q = \overline{(0+0)} = \bar{0} = 1$$



Case 2: If S=0 and R=1

$$Q = \overline{(1+P)} = \bar{1} = 0$$
$$P = \overline{(0+0)} = \bar{0} = 1$$

Case 3: Finally, the flip-flop is not operated with both S and R active (=1).

$$P = \overline{(1+Q)} = \bar{1} = 0$$
$$Q = \overline{(1+P)} = \bar{1} = 0$$

Latches (cont.)

- D flip-flops (Delay flip-flops)
- SR flip-flops (Set/Reset flip-flops)
- T flip-flops (Toggle flip-flops)
- JK flip-flops

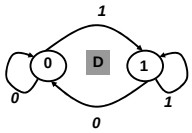
D flip-flops

D	q	q*
0	0	0
0	1	0
1	0	1
1	1	1

$\Rightarrow$

D	q*
0	0
1	1

$\Rightarrow$ q*=D



SR flip-flops

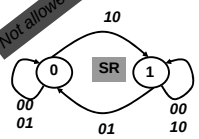
S	R	q	q*
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	-
1	1	1	-

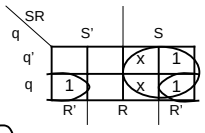
⇒

S	R	q*
0	0	q
0	1	0
1	0	1
1	1	-

⇒ $q^* = S + \overline{R}q$

Not allowed





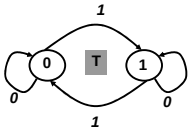
T flip-flops

T	q	q*
0	0	0
0	1	1
1	0	1
1	1	0

⇒

T	q*
0	q
1	$\overline{q}$

⇒ $q^*=T\oplus q$



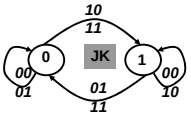
JK flip-flops

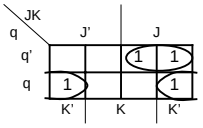
J	K	q	q*
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	0

⇒

J	K	q*
0	0	q
0	1	0
1	0	1
1	1	$\overline{q}$

⇒ $q^* = J\overline{q} + \overline{K}q$





Latches (cont.)

- Edge-triggered
  - rising/leading edge-triggered
  - falling/trailing edge-triggered
- Level-triggered
  - high level-triggered
  - low level-triggered

Analysis of Sequential Systems

- Figure 5.21 Page 342
  - Circuit
  - Equations
  - State table
  - Timing trace
  - Timing diagram
  - Equations A*,B*
  - State diagram

Flip-flop Design Techniques

x	q_1	q_2	q_1*	q_2*	z
0	0	0	0	0	0
0	0	1	1	0	0
0	1	0	0	0	0
0	1	1	1	0	0
1	0	0	0	1	0
1	0	1	0	1	0
1	1	0	1	1	0
1	1	1	1	0	1

Main truth table

From the truth table, it's clear that $z=q_1q_2$

We need to create the appropriate flip-flop design table to obtain a truth table for the flip-flop inputs.

q	q*	Input(s)
0	0	
0	1	
1	0	
1	1	

- **D flip-flops**
- **JK flip-flops**
- **SR flip-flops**
- **T flip-flops**

Design with D Flip-flop

D	q*
0	0
1	1

⇒

q	q*	D
0	0	0
0	1	1
1	0	0
1	1	1

⇒

From the **main truth table** ⇒

x	q_1	q_2	$D_1=q_1^*$	$D_2=q_2^*$
0	0	0	0	0
0	0	1	1	0
0	1	0	0	0
0	1	1	1	0
1	0	0	0	1
1	0	1	0	1
1	1	0	1	1
1	1	1	1	0

Design with D Flip-flop (cont.)

$$D_1 = xq_2 + xq_1 \qquad D_2 = x\overline{q_2} + xq_1$$

Implementation using D Flip-flops

Fig 5.26 page 352

Design with JK Flip-flop

J	K	q*
0	0	q
0	1	0
1	0	1
1	1	$\bar{q}$

⇒

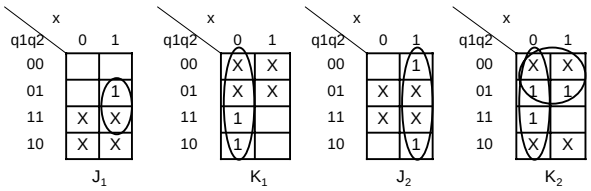
q	q*	J	K
0	0	0	X
0	1	1	X
1	0	X	1
1	1	X	0

⇒

From the main truth table ⇒

x	q_1	q_2	q_1^*	q_2^*	J_1	K_1	J_2	K_2
0	0	0	0	0	0	X	0	X
0	0	1	1	0	0	X	X	1
0	1	0	0	0	X	1	0	X
0	1	1	1	0	X	1	X	1
1	0	0	0	1	0	X	1	X
1	0	1	0	1	1	X	X	1
1	1	0	1	1	X	0	1	X
1	1	1	1	0	X	0	X	0

Design with JK Flip-flop (cont.)



$$J_1 = xq_2 \quad ; \quad K_1 = \bar{x}$$
$$J_2 = x \quad ; \quad K_2 = \bar{x} + \bar{q_1}$$

Design with SR Flip-flop

S	R	q*
0	0	q
0	1	0
1	0	1
1	1	-

⇒

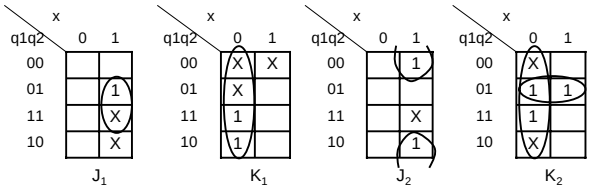
q	q*	S	R
0	0	0	X
0	1	1	0
1	0	0	1
1	1	X	0

⇒

From the main truth table ⇒

x	q_1	q_2	q_1^*	q_2^*	S_1	R_1	S_2	R_2
0	0	0	0	0	0	X	0	X
0	0	1	1	0	0	X	0	1
0	1	0	0	0	0	1	0	X
0	1	1	1	0	0	1	0	1
1	0	0	0	1	0	X	1	0
1	0	1	0	1	1	0	0	1
1	1	0	1	1	X	0	1	0
1	1	1	1	0	X	0	X	0

Design with SR Flip-flop (cont.)



$$S_1 = xq_2 \quad ; \quad R_1 = \bar{x}$$
$$S_2 = x\bar{q}_2 \quad ; \quad R_2 = \bar{x} + q_1q_2$$

Design with T Flip-flop

T	q*
0	q
1	$\bar{q}$

⇒

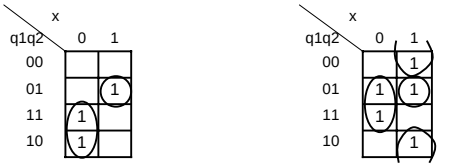
q	q*	T
0	0	0
0	1	1
1	0	1
1	1	0

⇒

x	q_1	q_2	q_1*	q_2*	T_1	T_2
0	0	0	0	0	0	0
0	0	1	1	0	0	1
0	1	0	0	0	1	0
0	1	1	1	0	1	1
1	0	0	0	1	0	1
1	0	1	0	1	1	1
1	1	0	1	1	0	1
1	1	1	1	0	0	0

From the main truth table ⇒

Design with T Flip-flop (cont.)



$$T_1 = \bar{x}q_1 + x\bar{q}_1q_2 \qquad T_2 = \bar{x}q_2 + x\bar{q}_2 + xq_1q_2$$

Because $\boxed{q^* = J\overline{q} + \overline{K}q}$

Notice that when q=0

$$q^* = J \cdot 1 + \overline{K} \cdot 0 = J$$
$$\Rightarrow J = q^*$$

And when q=1

$$q^* = J \cdot 0 + \overline{K} \cdot 1 = \overline{K}$$
$$\Rightarrow K = \overline{q^*}$$

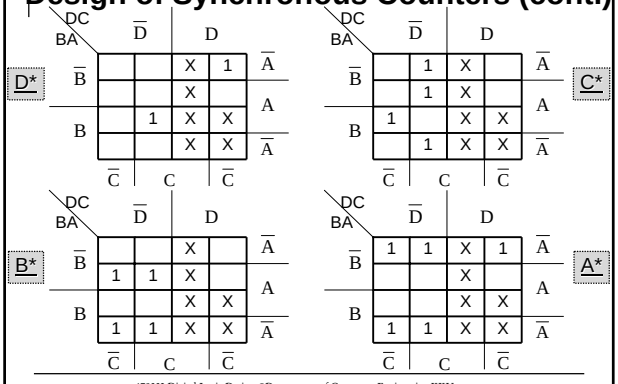
Design of Synchronous Counters

- Design a decimal or decade counter using JK flip-flops:

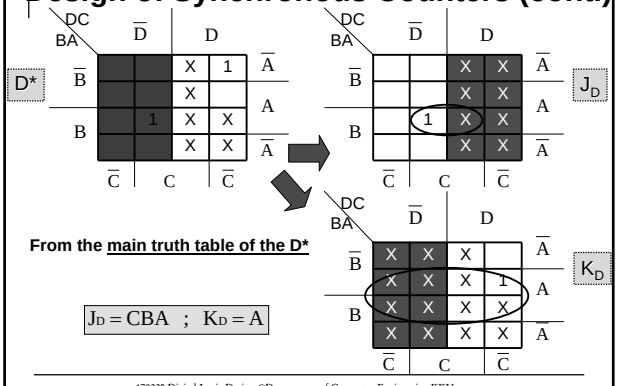
0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 0, 1, …

D	C	B	A	D*	C*	B*	A*
0	0	0	0	0	0	0	1
0	0	0	1	0	0	1	0
0	0	1	0	0	0	1	1
0	0	1	1	0	1	0	0
0	1	0	0	0	1	0	1
0	1	0	1	0	1	1	0
0	1	1	0	0	1	1	1
0	1	1	1	1	0	0	0
1	0	0	0	1	0	0	1
1	0	0	1	0	0	0	0
1	0	1	0	x	x	x	x
…	…	…	…	…	…	…	…
1	1	1	1	X	x	x	x

Design of Synchronous Counters (cont.)



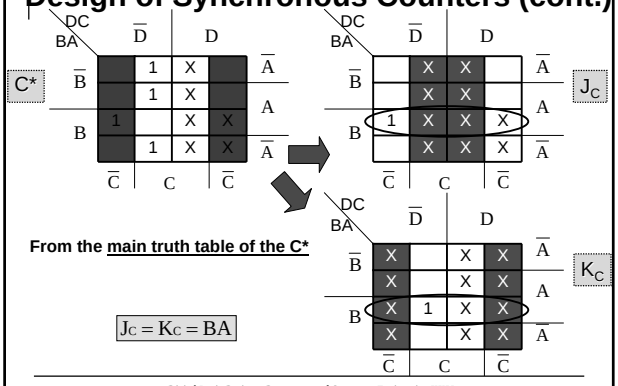
Design of Synchronous Counters (cont.)



From the **main truth table of the D***

$J_D = CBA \;\; ; \;\; K_D = A$

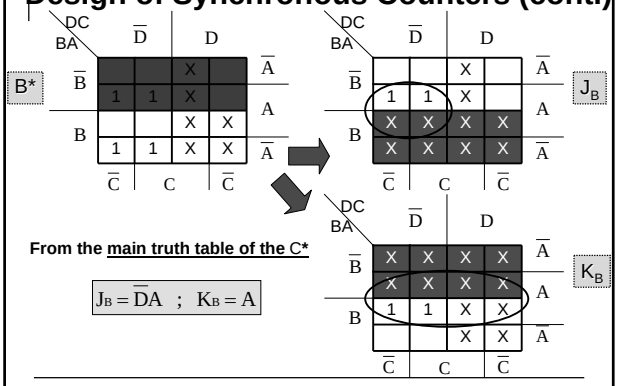
Design of Synchronous Counters (cont.)



From the **main truth table of the C***

$J_C = K_C = BA$

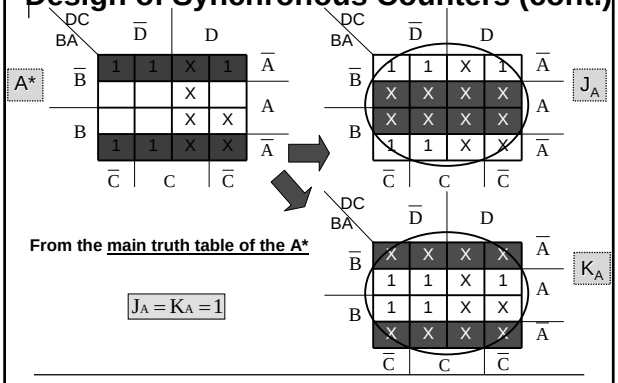
Design of Synchronous Counters (cont.)



From the **main truth table of the C***

$J_B = \overline{D}A \;\; ; \;\; K_B = A$

Design of Synchronous Counters (cont.)



From the main truth table of the A*

$$J_A = K_A = 1$$

q_1	q_2	q_3	q_4	q_1^*	q_2^*	q_3^*	q_4^*
0	0	0	0	0	0	0	1
0	0	0	1	0	0	1	0
0	0	1	0	0	0	1	1
0	0	1	1	0	1	0	0
0	1	0	0	0	1	0	1
0	1	0	1	0	1	1	0
0	1	1	0	0	1	1	1
0	1	1	1	1	0	0	0
1	0	0	0	1	0	0	1
1	0	0	1	0	0	0	0
1	0	1	0	x	x	x	x
…	…	…	…	…	…	…	…
1	1	1	1	x	x	x	x

Design of Asynchronous Counters

- Page 350….
- Figure 5.31(2-bit counter)&5.32(timing delay)
- Advantage:
  - Simplicity of the hardware ⇒ no combinational logic required
- Disadvantage:
  - Speed

Derivation of State tables and State Diagrams

- Consider the problem:

A system with one input ***x*** and one output ***z*** such that ***z*** = 1 at a clock time iff ***x*** is currently 1 and was also 1 at the previous two clock times.

- Another way of wording this same problem is

A Mealy system with one input ***x*** and one output ***z*** such that ***z*** = 1 iff ***x*** has been 1 for three consecutive clock times.

- A sample input/output trace is

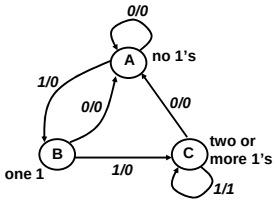
x	0	1	1	1	1	1	1	1	0	1	1	0	1	1	1	0	1
z	0	0	0	1	1	1	1	1	0	0	0	0	0	0	1	0	0	0

- **First approach:** save the previous 2 inputs.
- Knowing them and the present input → output.

		$q_1^*q_2^*$		z	
q_1	q_2	x=0	x=1	x=0	x=1
0	0	0 0	0 1	0	0
0	1	1 0	1 1	0	0
1	0	0 0	0 1	0	0
1	1	1 0	1 1	0	1

- Just discard the older input stored in memory and store the newer one plus the current input.

- **Second approach:** store in memory the number of consecutive 1's as follows:
  - **A** none, that is, the last input was 0
  - **B** one
  - **C** two or more



	q*		z	
q	x=0	x=1	x=0	x=1
A	A	B	0	0
B	A	C	0	0
C	A	C	0	1

- The second approach requires only 3 states, whereas the first requires 4. Both, however, use 2 flip-flops.
- Consider: the system produces a 1 if the input has been 1 for 25 consecutive clock times.
- Now the first approach requires to store the last 24 inputs and a state table of 2^{24} rows.
- The second approach requires 25 states which can be coded with 5 flip-flops.