DIGITAL TECHNICS II

Dr. Bálint Pődör

Óbuda University,
Microelectronics and Technology Institute

5. LECTURE: ANALYSIS AND SYNTHESIS OF SYNCHRONOUS SEQUENTIAL CIRCUITS

2nd (Spring) term 2017/2018

1

5. LECTURE

Analysis and synthesis of

synchronous sequential circuits:

Design examples and case studies

2

SYNTHESIS: GENERAL CONCEPTS

Synchronous sequential circuits synthesis procedure

- Word description of problem (hardest; art, not science)
- Derive state diagram and state table
- Minimize (moderately hard)
- Assign states (very hard)
- Produce state and output transition tables
- Determine what FFs to use and find their excitation maps
- Derive output equations/K-maps
- Obtain the logic diagram

This is the so called ”next state method” (c.f. Zsom Vol II)

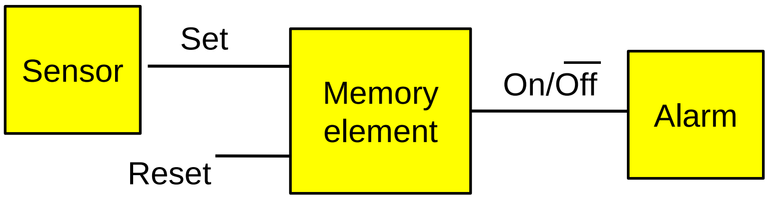
INTRODUCTORY EXAMPLES

Control of an alarm system – role of memory

Two-flip-flop circuit - designing with next-state

4

EXAMPLE 1: CONTROL OF AN ALARM SYSTEM

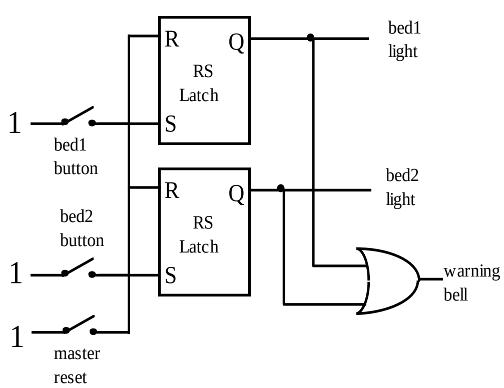


Control of an alarm system is one of the simplest case of sequential logic.
Alarm is ON when the sensor generates a positive voltage, SET, in response to an undesirable event.
Once alarm is on, it can only be turned off manually through a RESET button.
Memory is needed to remember the alarm has to be active until the reset signal arrives.

5

EXAMPLE 1: APPLICATION OF THE SR LATCH

- An important application of SR latches is for recording short lived events
  - e.g. pressing an alarm bell in a hospital



SYNTHESIS OF SYNCHRONOUS SEQUENTIAL CIRCUIT: EXAMPLE 2

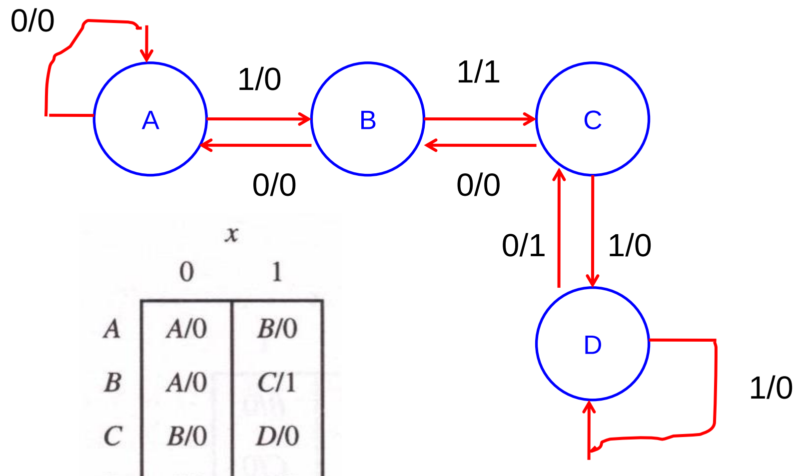
Synthesize the synchronous circuit which operates according to the given state table.

E.g. if the system in state C and the X input variable is 1, then in the next clock period the system goes to state D and the output variable will take the value 0.

	x = 0	x = 1
A	$A/0$	$B/0$
B	$A/0$	$C/1$
C	$B/0$	$D/0$
D	$C/1$	$D/0$

7

STATE TRANSITION DIAGRAM



	x = 0	x = 1
A	$A/0$	$B/0$
B	$A/0$	$C/1$
C	$B/0$	$D/0$
D	$C/1$	$D/0$

8

SYNTHESIS: EXAMPLE 2

- Example:
  - Find D FF realization of circuit defined in table (a)
  - (b): state assignment
  - (c): transition table
  - (d): output K-map
  - (e): excitation K-map

	x = 0	1
A	*A*/0	*B*/0
B	*A*/0	*C*/1
C	*B*/0	*D*/0
D	*C*/1	*D*/0

(a)

State	y_1	y_2
A	0	0
B	0	1
C	1	1
D	1	0

(b)

$y_1 y_2$ \ x	0	1
00	**00/0**	**01/0**
01	**00/0**	**11/1**
11	**01/0**	**10/0**
10	**11/1**	**10/0**

$Y_1 Y_2/z$

(c)

$y_1 y_2$ \ x	0	1
00	**0**	**0**
01	**0**	**1**
11	**0**	**0**
10	**1**	**0**

z

(d)

$y_1 y_2$ \ x	0	1
00	**0**	**0**
01	**0**	**1**
11	**0**	**1**
10	**1**	**1**

D_1 (= Y_1)

$y_1 y_2$ \ x	0	1
00	**0**	**1**
01	**0**	**1**
11	**1**	**0**
10	**1**	**0**

D_2 (= Y_2)

(e)

STATE ENCODING

	x = 0	1
A	*A*/0	*B*/0
B	*A*/0	*C*/1
C	*B*/0	*D*/0
D	*C*/1	*D*/0

State	y_1	y_2
A	0	0
B	0	1
C	1	1
D	1	0

$y_1 y_2$ \ x	0	1
00	**00/0**	**01/0**
01	**00/0**	**11/1**
11	**01/0**	**10/0**
10	**11/1**	**10/0**

$Y_1 Y_2/z$

FLIP-FOP CONTROL: D-FF

$$D_1 = y_1\bar{y}_2 + xy_2$$
$$D_2 = \bar{x}y_1 + x\bar{y}_1 = x \oplus y_1$$
$$z = x\bar{y}_1y_2 + \bar{x}y_1\bar{y}_2$$

(a)

State	y_1	y_2
A	0	0
B	0	1
C	1	1
D	1	0

(b)

$y_1 y_2$ \ x	0	1
00	**00/0**	**01/0**
01	**00/0**	**11/1**
11	**01/0**	**10/0**
10	**11/1**	**10/0**

$Y_1\,Y_2/z$

(c)

$y_1 y_2$ \ x	0	1
00	**0**	**0**
01	**0**	**1**
11	**0**	**0**
10	**1**	**0**

z

$y_1 y_2$ \ x	0	1
00	**0**	**0**
01	**0**	**1**
11	**0**	**1**
10	**1**	**1**

D_1 $(= Y_1)$

$y_1 y_2$ \ x	0	1
00	**0**	**1**
01	**0**	**1**
11	**1**	**0**
10	**1**	**0**

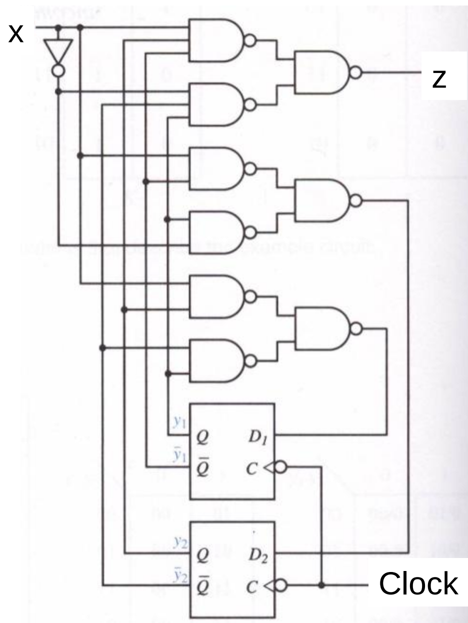
D_2 $(= Y_2)$

11

IMPLEMENTATION

- Example solution:
  - Logic diagram

$$D_1 = y_1\bar{y}_2 + xy_2$$
$$D_2 = \bar{x}y_1 + x\bar{y}_1 = x \oplus y_1$$
$$z = x\bar{y}_1y_2 + \bar{x}y_1\bar{y}_2$$



SYNTHESIS WITH JK FLIP-FLOPS

Example is same as before, but use JK FFs

y_1y_2 \ x	0	1
00	00/0	01/0
01	00/0	11/1
11	01/0	10/0
10	11/1	10/0

$Y_1\,Y_2/z$

(a)

y_1y_2 \ x	0	1
00	0d	0d
01	0d	1d
11	d1	d0
10	d0	d0

J_1K_1

y_1y_2 \ x	0	1
00	0d	1d
01	d1	d0
11	d0	d1
10	1d	0d

J_2K_2

(b)

(a): transition table; (b): Excitation tables; (c): Excitation maps

y_1y_2 \ x	0	1
00	0	0
01	0	1
11	d	d
10	d	d

J_1

y_1y_2 \ x	0	1
00	d	d
01	d	d
11	1	0
10	0	0

K_1

y_1y_2 \ x	0	1
00	0	1
01	d	d
11	d	d
10	1	0

J_2

y_1y_2 \ x	0	1
00	d	d
01	1	0
11	0	1
10	d	d

K_2

(c)

SYNTHESIS: JK FLIP-FLOP

y_1y_2 \ x	0	1
00	00/0	01/0
01	00/0	11/1
11	01/0	10/0
10	11/1	10/0

$Y_1\,Y_2/z$

y_1y_2 \ x	0	1
00	0d	0d
01	0d	1d
11	d1	d0
10	d0	d0

J_1K_1

y_1y_2 \ x	0	1
00	0d	1d
01	d1	d0
11	d0	d1
10	1d	0d

J_2K_2

FLIP-FLOP-1 CONTROL

$y_1 y_2$ \ x	0	1
00	**00/0**	**01/0**
01	**00/0**	**11/1**
11	**01/0**	**10/0**
10	**11/1**	**10/0**

$Y_1\,Y_2/z$

$y_1 y_2$ \ x	0	1
00	**0**	**0**
01	**0**	**1**
11	*d*	*d*
10	*d*	*d*

J_1

$y_1 y_2$ \ x	0	1
00	*d*	*d*
01	*d*	*d*
11	**1**	**0**
10	**0**	**0**

K_1

15

FLIP-FLOP-2 CONTROL

$y_1 y_2$ \ x	0	1
00	**00/0**	**01/0**
01	**00/0**	**11/1**
11	**01/0**	**10/0**
10	**11/1**	**10/0**

$Y_1\,Y_2/z$

$y_1 y_2$ \ x	0	1
00	**0**	**1**
01	*d*	*d*
11	*d*	*d*
10	**1**	**0**

J_2

$y_1 y_2$ \ x	0	1
00	*d*	*d*
01	**1**	**0**
11	**0**	**1**
10	*d*	*d*

K_2

16

IMPLEMENTATION

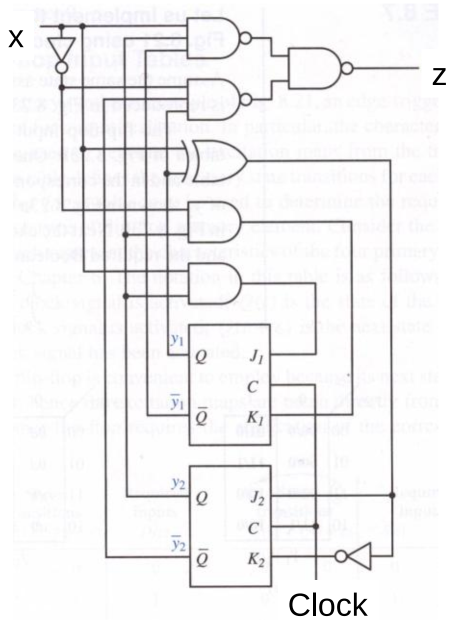
- Example JK FF solution:
  - Logic diagram

$J1 = X\,Y2$

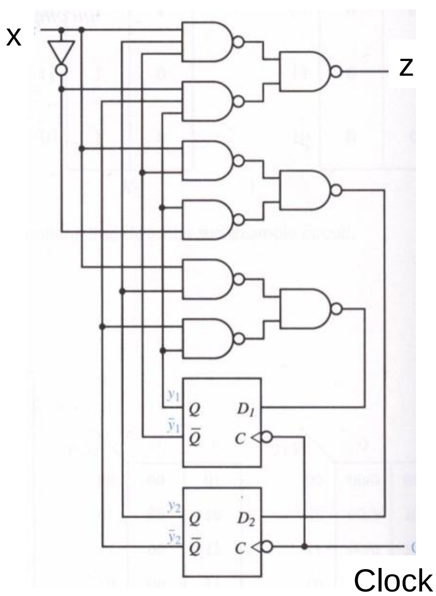
$K1 = \overline{X}\,Y2$

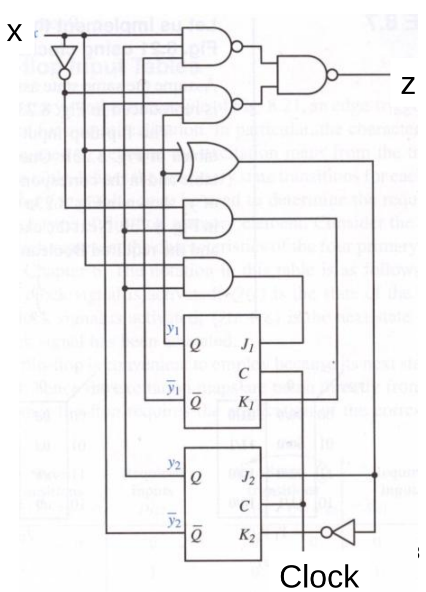
$J2 = \overline{X}\,Y1 + X\,\overline{Y1}$

$K2 = \overline{X}\,\overline{Y1} + X\,Y1$



COMPARISON OF TWO DESIGNS





COMPARISON OF DIFFERENT DESIGNS

Flip-flop:	D	D	JK	JK
Logic:	AND-OR	XOR	AND-OR	XOR
Pin count:	20	16	28	15
Gate count:	9	7	11	7

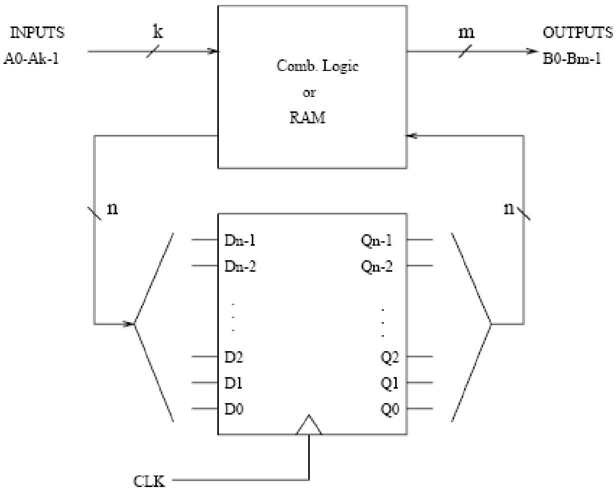
19

SYNTHESIS OF SYNCHRONOUS CIRCUITS: GENERAL PROCEDURE (EMPHASIS)

1. Constructing the state transition diagram.
2. Selection or specifying the encoding of the states.
3. Constructing the state transition tables. It gives for each cycle the next-state of each flip-flop in the function of the previous states of all flip-flops and in the function of the control conditions (up/down).
4. Selection or specifying the type of flip-flop used in the implementation. Excitation table of the flip-flop type.
5. Determination of the logic functions of the control input(s) of each flip-flop. Performing the necessary or appropriate minimization.
6. Selection of the types of logic gates to be used and implementation of the feedback/control network.

20

STATE MACHINE



General scheme of a state machine.

21

STATE MACHINE SYNTHESIS

The strategy for applying this scheme to a given problem consists of the following:

1. Identify the number of required states, m. The number of bits of memory (e.g. number of flip-flops) required to specify the m states is at minimum n = log2(m).

2. Make a state diagram which shows all states, inputs, and outputs.

3. Make a truth table for the logic section. The table will have n + k inputs and n + m outputs.

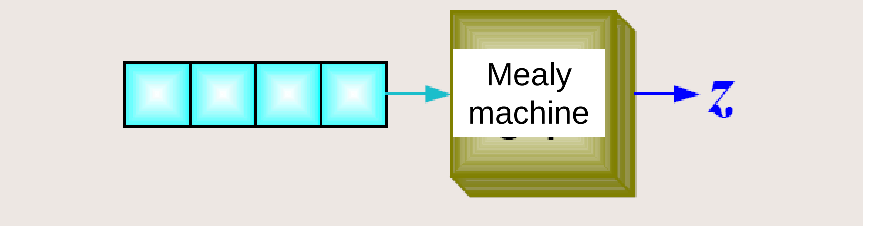
4. Implement the truth table using combinational logic techniques.

22

SYNTHESIS OF SEQUENTIAL CIRCUIT: A CASE STUDY

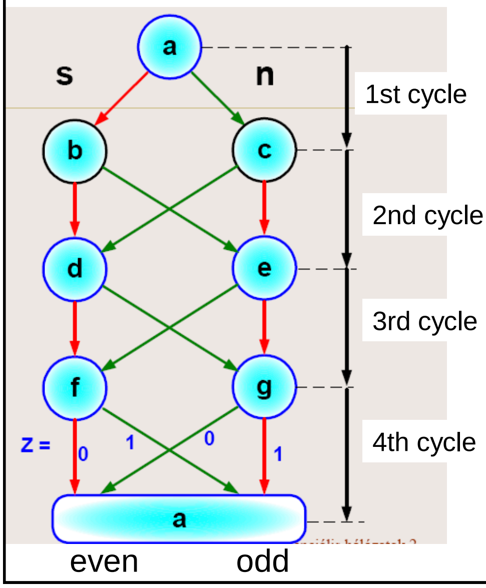
- Synthetize a network which determines the parity of a four bit serial code word.
- Should indicate the parity of the incoming code word after receiving the 4-th bit as
  - 1 if the parity is odd,
  - 0 if the parity is even.
- The output is irrelevant (don't care) during the first three cycle of the period.

4-BIT PARITY INDICATOR



- When checking the parity the order of the bits is irrelevant.
- Construct the state transition diagram of the Mealy-machine.

4-BIT PARITY INDICATOR: STATE TRANSITION DIAGRAM



left (s)- even
right (n) - odd

red - incoming bit 0
green - incoming bit 1

The output Z is defined only in the fourth cycle, otherwise it is ”don’t care”.

For the code word 1011
a → c → d → f → a

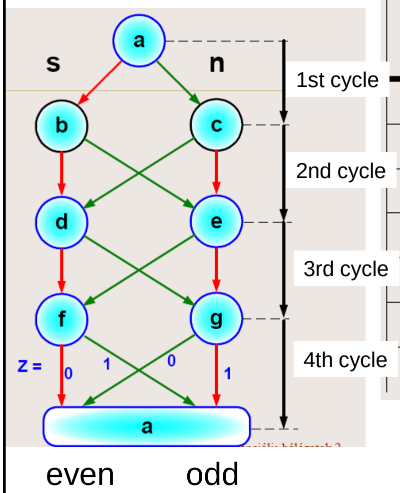
25

CHARACTERISTICS

- Because there are two input conditions, two connecting lines emanate from each node.
- The network returns to its initial state after the fourth cycle.
- The operation of the network is cyclic, the length of the period is four cycles.

26

STATE TRANSITION TABLE AND DIAGRAM



q^n	q^{n+1} X=0	q^{n+1} X=1	Z X=0	Z X=1
a	b	c	x	x
b	d	e	x	x
c	e	d	x	x
d	f	g	x	x
e	g	f	x	x
f	a	a	0	1
g	a	a	1	0

27

THE NUMBER OF INTERNAL STATES AND THEIR ENCODING

- Total number of internal states: seven
- Three flip-flops (Q_1, Q_2, Q_3) are necessary and enough for the encoding.
- The actual state encoding greatly influences the complexity and structure of the network.
- Here we use the final (optimal) state encoding.

28

STATE ENCODING

q^n	Q_1	Q_2	Q_3
a	0	0	x
b	0	1	0
c	0	1	1
d	1	1	0
e	1	1	1
f	1	0	0
g	1	0	1

- In the firs row, we make use of the redundancy.
- To the states in the same level of the state transition diagram, the same Q1 and Q2 codes are ascribed.
- Q1, Q2: cycle counters.
- Q3: indicates whether the system is in the even or on the odd branch of the state transition diagram.

29

STATE FUNCTIONS AND THE OUTPUT FUNCTION

i	n-edik ütem					(n+1)-edik ütem				Z
	X	Q_1	Q_2	Q_3	q^n	q^{n+1}	Q_1	Q_2	Q_3	
0	0	0	0	0	a	b	0	1	0	x
1		0	0	1	a	b	0	1	0	x
2		0	1	0	b	d	1	1	0	x
3		0	1	1	c	e	1	1	1	x
4		1	0	0	f	a	0	0	x	0
5		1	0	1	g	a	0	0	x	1
6		1	1	0	d	f	1	0	0	x
7		1	1	1	e	g	1	0	1	x

STATE FUNCTIONS AND THE OUTPUT FUNCTION

i	n-edik ütem					(n+1)-edik ütem				Z
	X	Q_1	Q_2	Q_3	q^n	q^{n+1}	Q_1	Q_2	Q_3	
8	1	0	0	0	a	c	0	1	1	x
9	1	0	0	1	a	c	0	1	1	x
10	1	0	1	0	b	e	1	1	1	x
11	1	0	1	1	c	d	1	1	0	x
12	1	1	0	0	f	a	0	0	x	1
13	1	1	0	1	g	a	0	0	x	0
14	1	1	1	0	d	g	1	0	1	x
15	1	1	1	1	e	f	1	0	0	x

STATE FUNCTIONS AND THE OUTPUT FUNCTION

$Q_1^{n+1} = \Sigma^4(2,3,6,7,10,11,14,15);$

$Q_2^{n+1} = \Sigma^4(0\text{-}3,8\text{-}12);$

$Q_3^{n+1} = \Sigma^4(3,7,8,9,10);\ x:(4,5,12,13);$

$Z^n = \Sigma^4(5,12);\ x:(0\text{-}3,6\text{-}11,14,15);$

The weighing of the variables:

X^n	8
Q_1^n	4
Q_2^n	2
Q_3^n	1

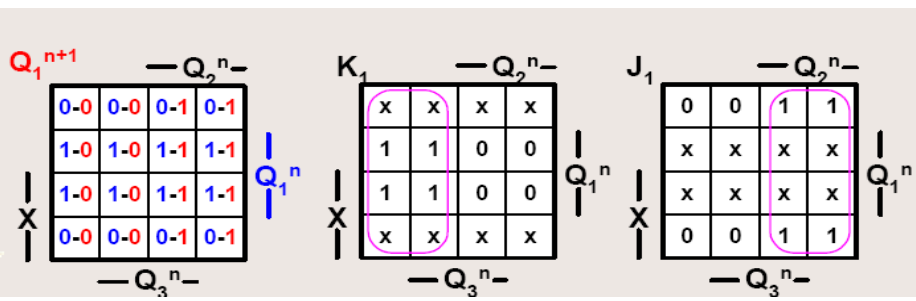
EXCITATION TABLE OF THE JK FLIP-FLOP

The logic synthesis is based on the excitation table of the flip-flop chosen for the implementation.

Q^n	$\rightarrow$	Q^{n+1}	J	K
0	$\rightarrow$	0	0	X
0	$\rightarrow$	1	1	X
1	$\rightarrow$	0	X	1
1	$\rightarrow$	1	X	0

33

CONTROL OF FLIP-FLOP Q_1

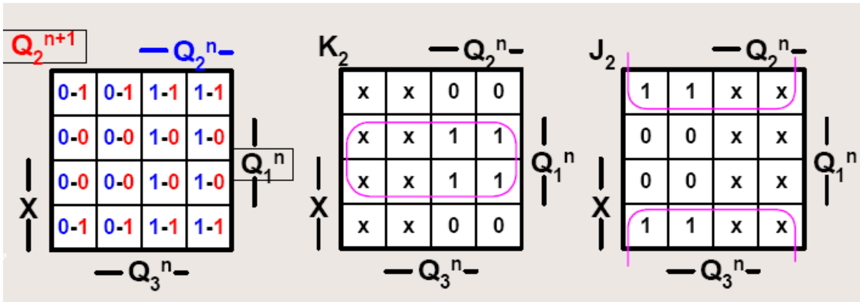


$K_1 = \overline{Q_2}$ $\qquad$ $J_1 = Q_2$

Note the role of the don't care terms in the minimization.

34

CONTROL OF FLIP-FLOP Q_2

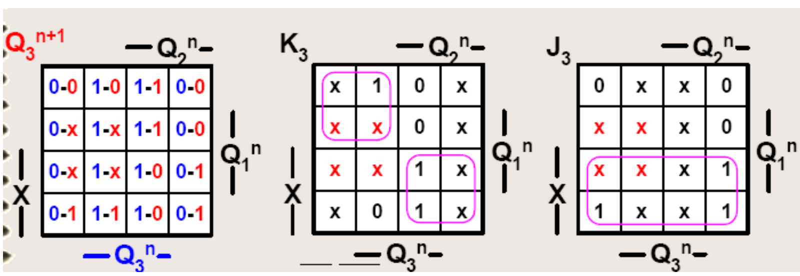


$K_2 = Q_1$ $\quad\quad$ $J_2 = \overline{Q_1}$

Due to the proper state-encoding, the X input variable is not present in the control equations of Q_1 és Q_2 .These two flip-flops act as cycle counter.

35

CONTROL OF FLIP-FLOP Q_3

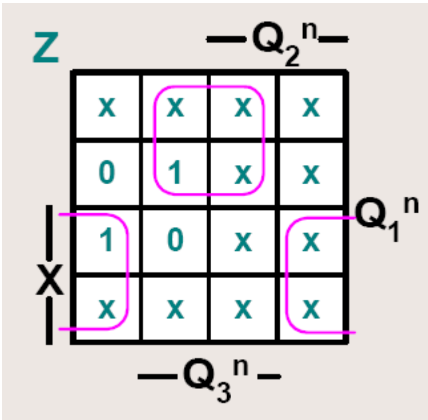


$K_3 = \overline{X}\,\overline{Q_2} + X\,Q_2 = X \oplus \overline{Q_2}$ $\quad\quad$ $J_3 = X$

The X input is among the variables controlling the flip-flop. The state of Q_3 will represent the actual parity. Q_3 will “remember” then parity of the input sequence.

36

THE OUTPUT FUNCTION Z



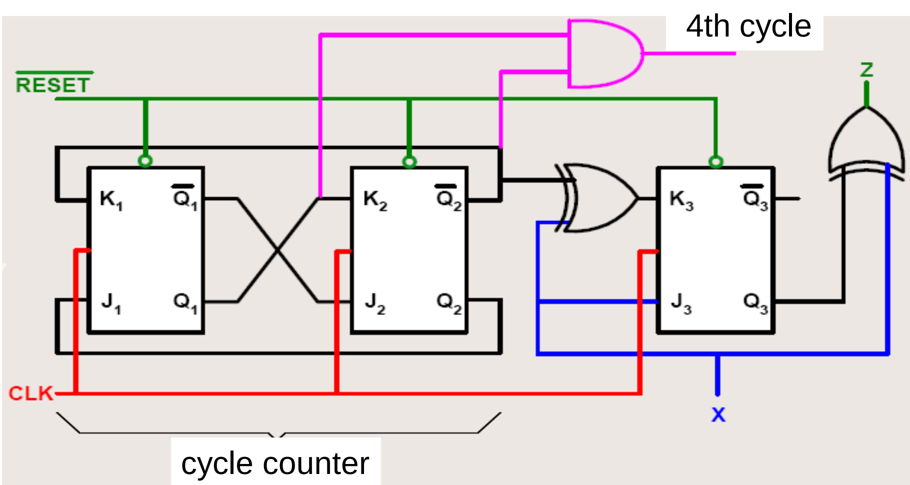
Note the chessboard pattern!
This implies XOR function:

$$Z = \overline{X}\,Q_3 + X\,\overline{Q_3} =$$

$$= X \oplus Q_3$$

37

THE LOGIC DIAGRAM OF THE PARITY CHECK CIRCUIT



38

IMPLEMENTATION ALTERNATIVE USING D FLIP-FLOPS

$D_1 = Q_2$ $D_2 = \overline{Q}_1$

$D_3 = X\,\overline{Q}_1 + X\,\overline{Q}_3 + \overline{X}\,Q_1\,Q_2$

Due to the ”clever” sate encoding, the control of the two flip-flops acting as the cycle counter corresponds to the usual one. However the control network of the third flip-flop is somewhat more complex than in the former implementation.

39

IMPLEMENTATION USING T FLIP-FLOPS

The feedback network is somewhat more complicated than in the case of D flip-flops.
Main reason: Counting in Gray code with T flip-flops needs more gates for the feedback.

Perhaps somebody might check a design with T flip-flops, the cycle counter operating in the simple binary code…

40

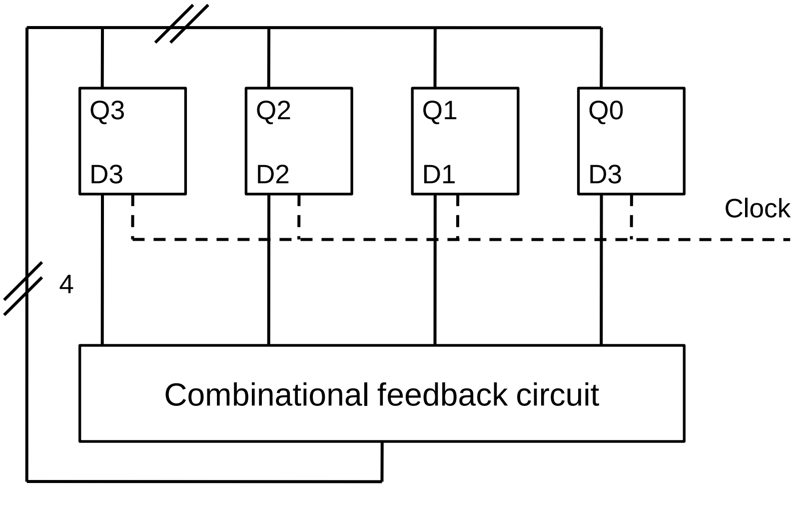
SYNCHRONOUS COUNTER DESIGN EXAMPLE AND CASE STUDY

Consider the synthesis of a 4-bit up-counter in Gray-code using D flip-flops.

A Gray-code counter using D flip-flops can be designed by finding the appropriate function of each D terminal. Given a present state of the counter, the D terminal of each flip-flop should be made equal to the value of the same bit position of the next-number in the Gray code.

41

4-BIT GRAY CODE COUNTER: CONCEPTUAL DIAGRAM



42

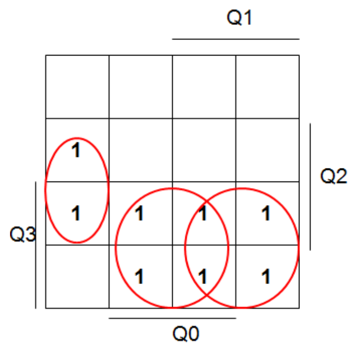
STATE TRANSITION TABLE

Minterm index	$Q3^n$	$Q2^n$	$Q1^n$	$Q0^n$	$Q3^{n+1}$ D3	$Q2^{n+1}$ D2	$Q1^{n+1}$ D1	$Q0^{n+1}$ D0
0	0	0	0	0	0	0	0	1
1	0	0	0	1	0	0	1	1
3	0	0	1	1	0	0	1	0
2	0	0	1	0	0	1	1	0
6	0	1	1	0	0	1	1	1
7	0	1	1	1	0	1	0	1
5	0	1	0	1	0	1	0	0
4	0	1	0	0	1	1	0	0
12	1	1	0	0	1	1	0	1
13	1	1	0	1	1	1	1	1
15	1	1	1	1	1	1	1	0
14	1	1	1	0	1	0	1	0
10	1	1	1	0	1	0	1	1
11	1	1	1	1	1	0	0	1
9	1	1	0	1	1	0	0	0
8	1	0	0	0	0	0	0	0

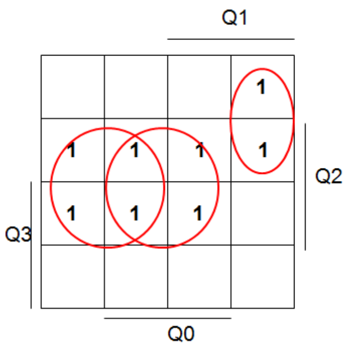
43

KARNAUGH MAPPING

D3



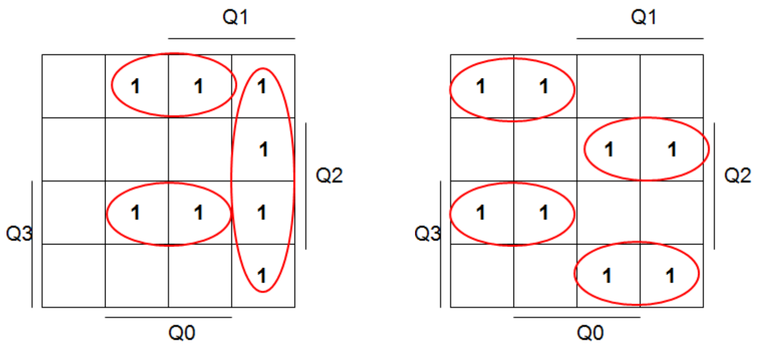
D2



44

KARNAUGH MAPPING

D1 D0



45

FLIP-FLOP CONTROL EQUATIONS

$Q3^{n+1} = D3 = Q3Q0 + Q3Q1 + Q2\overline{Q1}\,\overline{Q0}$

$Q2^{n+1} = D2 = Q2\overline{Q1} + Q2Q0 + \overline{Q3}Q1\overline{Q0}$

$Q1^{n+1} = D1 = Q1\overline{Q0} + \overline{Q3}\,\overline{Q2}Q0 + Q3Q2Q0$

$Q0^{n+1} = D0 = \overline{Q3}\,\overline{Q2}\,\overline{Q1} + Q3Q2\overline{Q1} + \overline{Q3}Q2Q1 + Q3\overline{Q2}Q1$

Implementation options: two-level AND-OR (13 AND, 4 OR) in modular logic or PLA, or two-level NAND-NAND in modular logic, or PROM.

46

FLIP-FLOP CONTROL EQUATIONS

Design alternative: D1 and D0 controls can be implemented in AND-OR-XOR LOGIC too.

$$Q1^{n+1} = D1 = Q1\overline{Q0} + \overline{Q3}\,\overline{Q2}Q0 + Q3Q2Q0 =$$

$$Q1\overline{Q0} + (Q3 \oplus \overline{Q2})Q0$$

$$Q0^{n+1} = D0 = \overline{Q3}\,\overline{Q2}\,\overline{Q1} + Q3Q2\overline{Q1} + \overline{Q3}Q2Q1 + Q3\overline{Q2}Q1 =$$

$$Q3 \oplus \overline{Q2} \oplus Q1$$

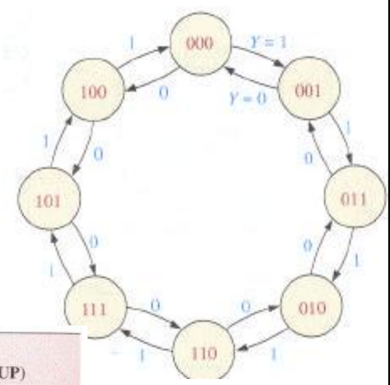
Give a three-level combinational network (7 AND, 3 OR, 2 XOR, and 1 INV).

UP/DOWN 3-BIT GRAY CODE COUNTER

State transition diagram

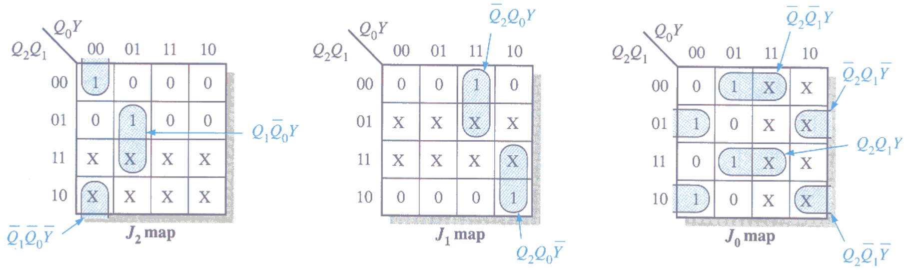
Next-state table

UP/$\overline{\text{DOWN}}$ control input: Y

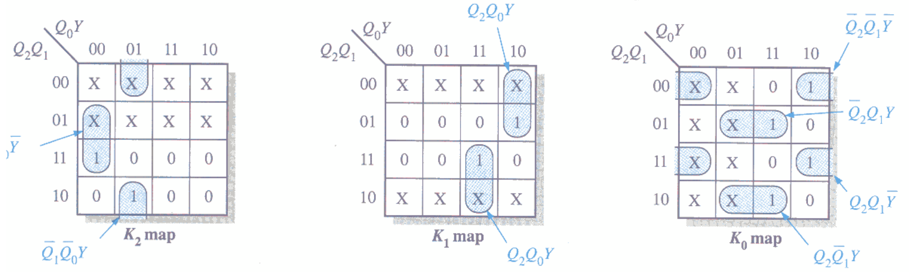


Present State			Next State Y = 0 (DOWN)			Y = 1 (UP)		
Q_2	Q_1	Q_0	Q_2	Q_1	Q_0	Q_2	Q_1	Q_0
0	0	0	1	0	0	0	0	1
0	0	1	0	0	0	0	1	1
0	1	1	0	0	1	0	1	0
0	1	0	0	1	1	1	1	0
1	1	0	0	1	0	1	1	1
1	1	1	1	1	0	1	0	1
1	0	1	1	1	1	1	0	0
1	0	0	1	0	1	0	0	0

UP/DOWN 3-BIT GRAY CODE COUNTER



Variables: Q2, Q1, Q0, and Y



UP/DOWN 3-BIT GRAY CODE COUNTER

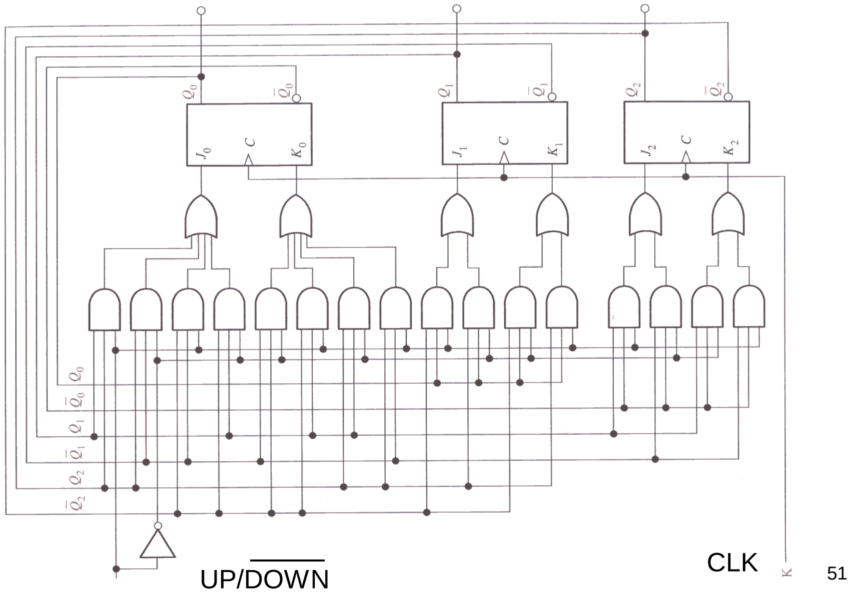
$$J_0 = Q_2Q_1Y + Q_2\overline{Q}_1\overline{Y} + \overline{Q}_2Q_1\overline{Y} + \overline{Q}_2\overline{Q}_1Y$$
$$J_1 = \overline{Q}_2Q_0Y + Q_2Q_0\overline{Y}$$
$$J_2 = Q_1\overline{Q}_0Y + \overline{Q}_1\overline{Q}_0\overline{Y}$$

$$K_0 = \overline{Q}_2\overline{Q}_1\overline{Y} + \overline{Q}_2Q_1Y + Q_2Q_1\overline{Y} + Q_2\overline{Q}_1Y$$
$$K_1 = \overline{Q}_2Q_0\overline{Y} + Q_2Q_0Y$$
$$K_2 = Q_1\overline{Q}_0\overline{Y} + \overline{Q}_1\overline{Q}_0Y$$

Logic expressions for flip-flop control

50

UP/DOWN 3-BIT GRAY CODE COUNTER



51

4-BIT BI-DIRECTIONAL GRAY CODE COUNTER

Features of design provided by one of the students of my previous course.

Compared designs using D or T flip-flops.

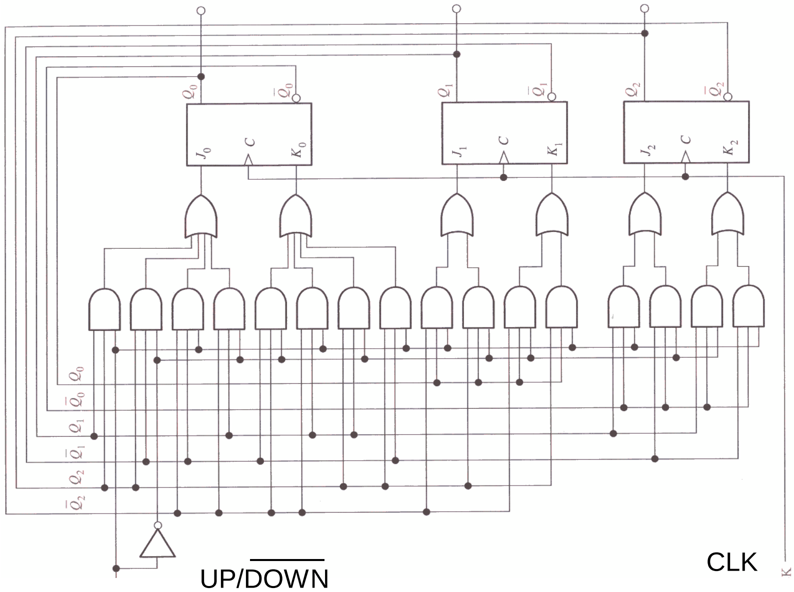
Using T flip-flops, some several common terms could be realized by XOR gate or XOR gate and inverter, leading to further simplification of the feedback circuit.

Complexity: 16 NAND gates (2,3 or 4 inputs), 2 XOR gates and 2 inverters.

Estimated the maximum clock frequency of the counter when using high speed CMOS logic components.

52

UP/DOWN 3-BIT GRAY CODE COUNTER



53

SUGGESTED PROBLEM

Design a 3-bit counter that will count in the sequence 000, 010, 011, 101, 110,111, and repeat the sequence. The counter has two unused states. These are 001 and 100. Implement the counter as a self-correcting such that if the counter happens to be one of the unused states (001 or 100) upon power-up or due to error, the next clock pulse puts it in one of the valid states and the counter provides the correct output. Use T flip-flops. Note that the initial states of the flip-flops are unpredictable when power is turned ON. Therefore, all the unused (don't care) states of the counter should be checked to ensure that the counter eventually goes into the desirable counting sequence. This is called a self-correcting counter.

54

REVISION QUESTIONS

1. Describe the main steps involved in the synthesis of a synchronous sequential circuit/finite state machine.

2. Describe and illustrate with a simple example the operation of a state machine with memory.

PROBLEMS AND EXERCISES

1. Design a finite state machine which determines whether the two 4-bit binary numbers arriving simultaneously on the two inputs are equal or not. If not, it should also indicate which is the greater. The code-words in pairs arrive cyclically to the **X** and Y inputs of the circuit. The MSBs arrive at first to the input.

2. Four-bit codewords representing normal BCD coded decimal digits arrive cyclically to the **X** input of a synchronous sequential circuit. The MSB arrives first. Design a synchronous sequential circuit which indicates with 1 on its **Z** output if the arriving 4-bit codeword presumably representing a normal BCD digit is invalid.

PROBLEMS AND EXERCISES

3. Design a synchronous sequential circuit to control a bottled drink vending machine. A bottle of drink costs 200 HUF. The machine accepts 50, 100, and 200 HUF coins. When the amount of money inserted equals or exceeds the price of the merchandize, the machine vends a bottle and returns change if any, then waits for the next transaction.

4. Design a synchronous counter according to the specifications given below:
   - Encoding: Excess-3 (Stibitz) code
   - Counting direction: up or down, externally controllable
   - Mode of operation: self-correcting (returns to the counting cycle from the invalid states).

PROBLEMS AND EXERCISES

5. A sequential circuit has one input and one output. The state diagram is shown below. Design the circuit with (a) JK flip-flops, (b) D flip-flops, (c) SR flip-flops, and (d) T flip-flops.