

Quiz 3 for CpE358/CS381 – Switching Theory and Logical Design Stevens Institute of Technology Summer 1, 2004 June 16, 2004	Name
-------------------------------------------------------------------------------------------------------------------------------------	------

Pledge:

This quiz is open book/open notes. PCs are permitted to lookup information in your notes for the course, but electronic communications with others in the class or outside is prohibited.

Total value is 100 points (10% of course grade). All questions are equally weighted. Do any 5 of the 7 questions. Do more than 5 for extra credit. Some question can be answered in more than one way. Only one answer is required, but extra credit will be given for identifying and explaining alternate answers. Some questions ask for N answers. Extra credit will be given for more than N answers.

- (1) The simplest way to avoid transmission errors in a communications system is to transmit the same message $2N+1$ times and decide the result based on the majority of the results. Design a combinatorial circuit that has 5 inputs and one output. The output is 1 if most of the inputs are 1 and 0 if most of the inputs are 0.

You could recognize that the output must be 1 when three or more of the inputs are 1 and generate the terms directly. This might cause you to include terms that include 4 and 5 inputs, which are actually included in the terms made up of three inputs. On the other hand, you could minimize a 5 variable Karnaugh map as shown below. The result will be the same.

AB\CD	00	01	11	10
00	0	0	0	0
01	0	0	1	0
11	0	1	1	1
10	0	0	1	0

E=0

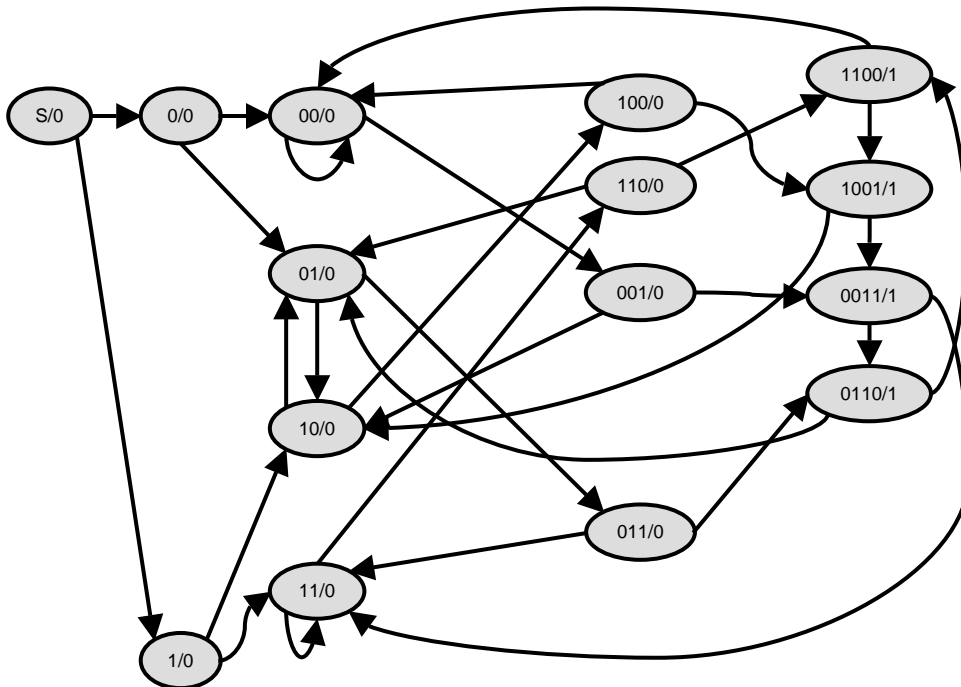
AB\CD	00	01	11	10
00	0	0	1	0
01	0	1	1	1
11	1	1	1	1
10	0	1	1	1

E=1

$$F(A,B,C,D,E) = ABC + ABD + BCD + ACD + ECD + ABE + ADE + ACE + BCE + BDE$$

- (2) Design a sequential circuit that recognizes particular sequences of 1's and 0's: The output should be 1 if the last 4 bits of the input match any part of the sequence 00110011.... Provide a state table or state diagram (both for extra credit) and (extra credit) provide a logic diagram.

Here are the state table and state diagram. I left off the input values for the state transitions since it was too messy – the values are in the state table:

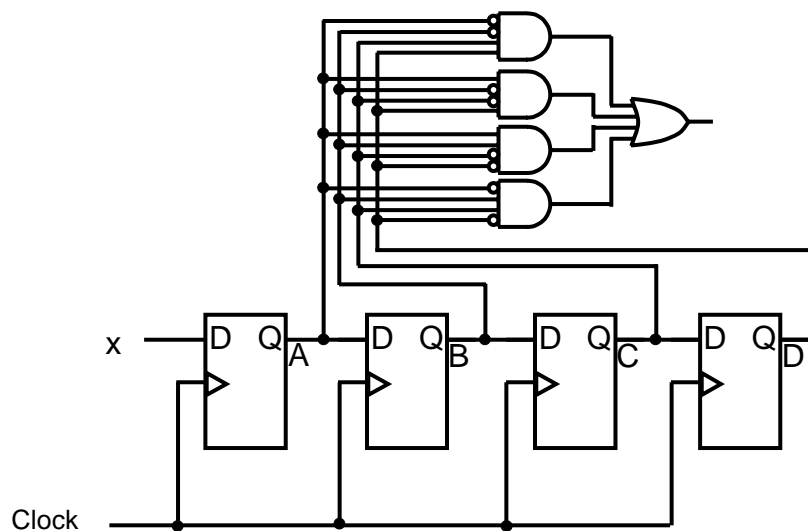


Here is the state table:

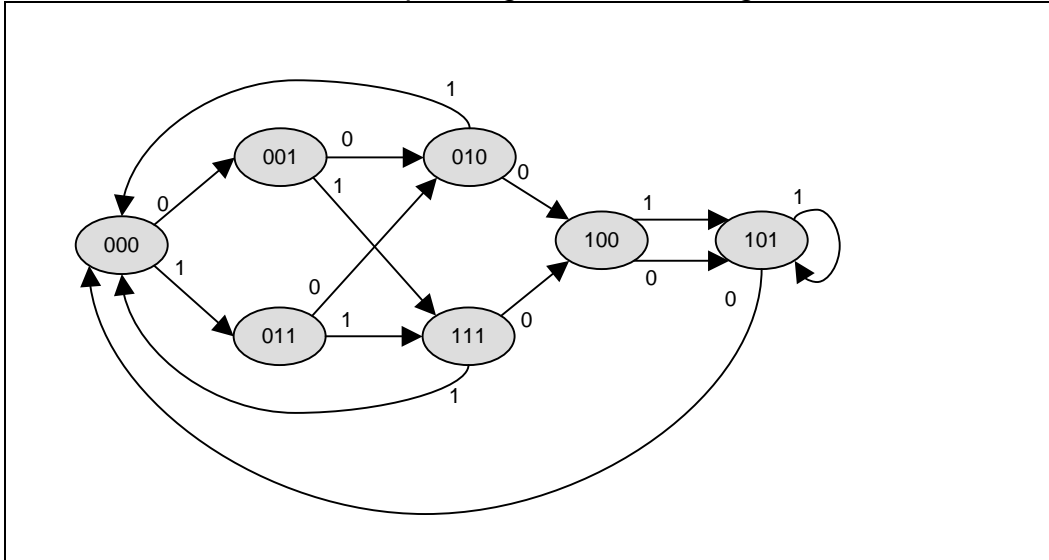
State	Next state with input 0	Next state with input 1	output
S	0	1	0
0	00	01	0
1	10	11	0
00	00	001	0
01	10	011	0
10	100	01	0
11	110	11	0
001	10	0011	0
011	0110	11	0
100	00	1001	0
110	1100	01	0
0011	0110	11	1
0110	1100	01	1
1001	10	0011	1
1100	00	1001	1

We could rename the states to have binary values and assign them to flip-flops, then determine the J-K inputs to the flip-flops, but here is an easier design:

Recognize that what is really needed is a 4-bit shift register and a state decoder to output a 1 when the pattern is matched. Here is a circuit to perform that function:



(3) Derive the state table corresponding to the state diagram shown below.



There are two possible ways of expressing the state table:

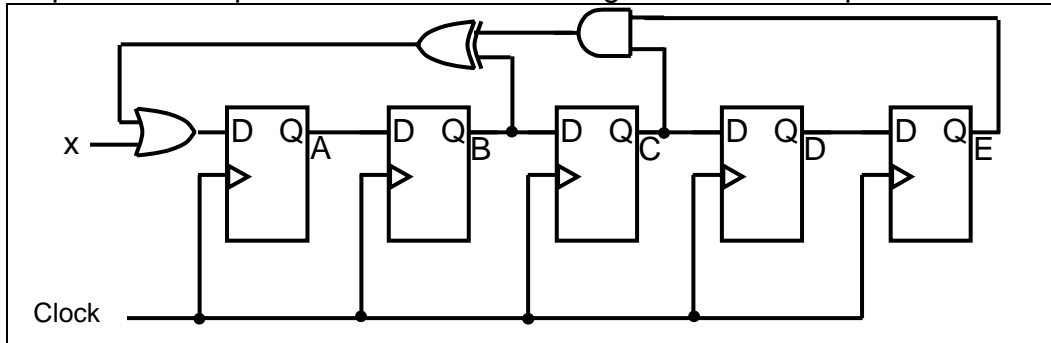
(1):

State	Input	Next State
000	0	001
000	1	011
001	0	010
001	1	111
010	0	100
010	1	000
011	0	010
011	1	111
100	0	101
100	1	101
101	0	000
101	1	101
110	X	Undefined
111	0	100
111	1	000

(2)

State	Next State With input 0	Next State With input 1
000	001	011
001	010	111
010	100	000
011	010	111
100	101	101
101	000	101
110	Undefined	Undefined
111	100	000

(4) The shift register circuit below starts in the state 01010. What is the sequence of output values that this circuit generates with input 101101?



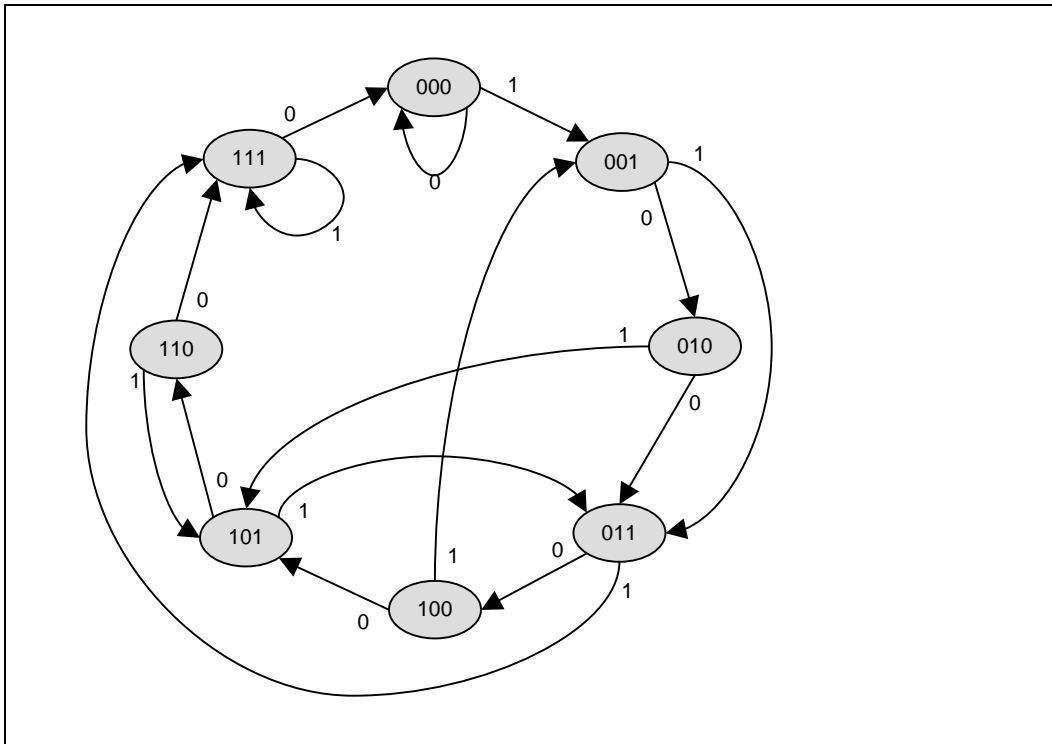
It is easiest to generate an abbreviated state table for this:

Present state	x	CE	CE xor B	CE xor B or x	E (output)	Next state
01010	1	X	X	X	0	1-0101
10101	0	1	1	1	1	1-1010
11010	1	X	X	1	0	1-1101
11101	1	X	X	1	1	1-1110
11110	0	0	1	1	0	1-1111
11111	1	X	X	1	1	1-1111

By the way, the first 5 outputs are simply the shifted state of the register.

(5) Draw the state diagram for the following state table

State	Input	Next State
000	0	000
000	1	001
001	0	010
001	1	011
010	0	011
010	1	101
011	0	100
011	1	111
100	0	101
100	1	001
101	0	110
101	1	011
110	0	111
110	1	101
111	0	000
111	1	111



(6) What is the minimum number of flip-flops needed to implement this state table?

State	Next State With input 0	Next State With input 1	Output
00001	01010	10001	1
00010	10111	10010	0
00011	01100	01111	1
00100	01100	10000	1
01001	10010	00001	0
01010	01100	01100	0
01100	01001	01001	1
01111	01100	11000	1
10000	01010	01010	0
10001	10111	01010	1
10010	10010	01010	0
10111	10001	01100	1
11000	10111	01010	0
11100	01010	01100	0
11110	01100	01001	1

There are 5 bits used to represent each state, but this doesn't mean we need 5 FFs – the states can be renamed with 4 bit values, since there are 15 states. We might be able to reduce the number of states to less than 8, but that would be extra credit, since it will take a bit of work to verify. The 5 bit state names is probably too cumbersome to work with, so let's start with single letter values for the state names.

State	Next State With input 0	Next State With input 1	Output
A	F	K	1
B	M	L	0
C	G	H	1
D	G	J	1
E	L	A	0
F	G	G	0
G	E	E	1
H	G	N	1
J	F	F	0
K	M	F	1
L	L	F	0
M	K	G	1
N	M	F	0
P	F	G	0
Q	G	E	1

The next thing to do is to find the equivalence classes:

{ACDGHKMQ} {BEFJLNP} based on output

{ACDGHKMQ} {BEFJLNP}

with 0

{ACDGHKMQ} {BEFJLNP}

{AG}{CDHKMQ} {BFN} {EJLP}

With 1

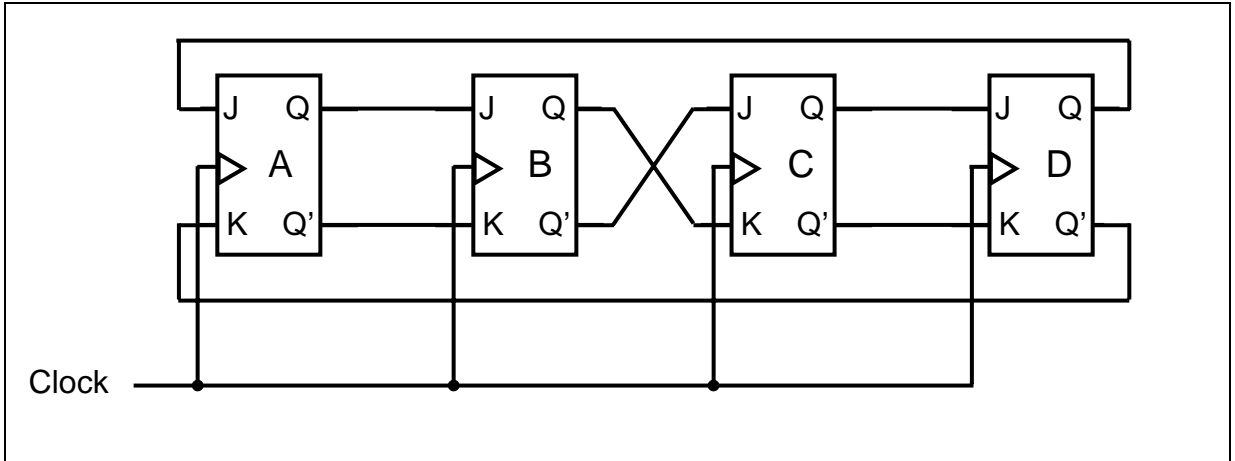
{A}{G}{C}{D}{H}{K}{M}{Q}{B}{F}{N}{E}{J}{L}{P}

{A}{G}{C}{D}{Q}{H}{K}{M}{B}{F}{N}{E}{P}{J}{L}

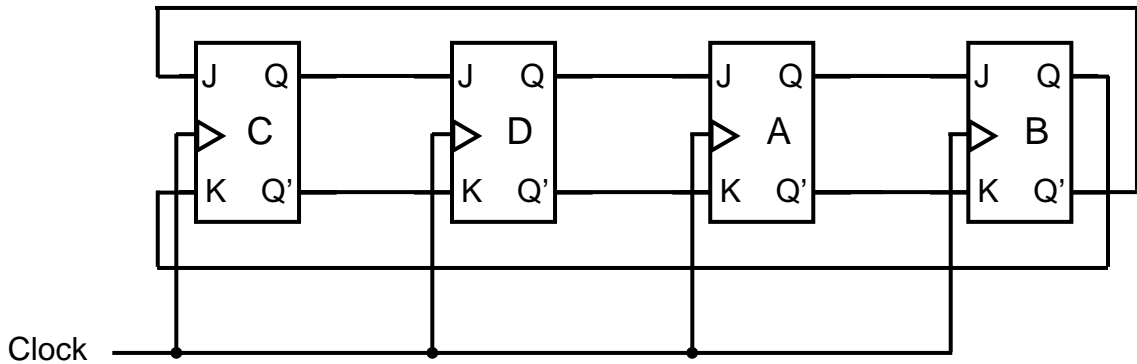
This is 11 states, so at least 4 FFs are needed.

If we had found that there were 8 or fewer equivalence classes, we could have realized the system with 3 FFs, but since the number is more than 8, 4 FFs are needed.

(7) Analyze the following sequential circuit and derive either a state table or a state diagram to describe its operation. Assume that the circuit starts in the 0000 state



This is a variation of a switch tail ring counter. Redraw the circuit:



Now, determine the state table, first with states labeled CDAB, then reordered:

State CDAB	NextState CDAB
0000	1000
1000	1100
1100	1110
1110	1111
1111	0111
0111	0011
0011	0001
0001	0000

State ABCD	NextState ABCD
00	00 10
00	00 11
00	10 11
10	11 11
11	11 01
11	11 00
11	01 00
01	00 00

And here is the state diagram:

