Lecture 16, March 14, 2007.

Design of Sequential Circuits

While analysis starts with a circuit diagram and ends up with a functional description, e.g., in the form of a state diagram, design or synthesis starts with a functional description and the end result should be a circuit diagram.

State reduction

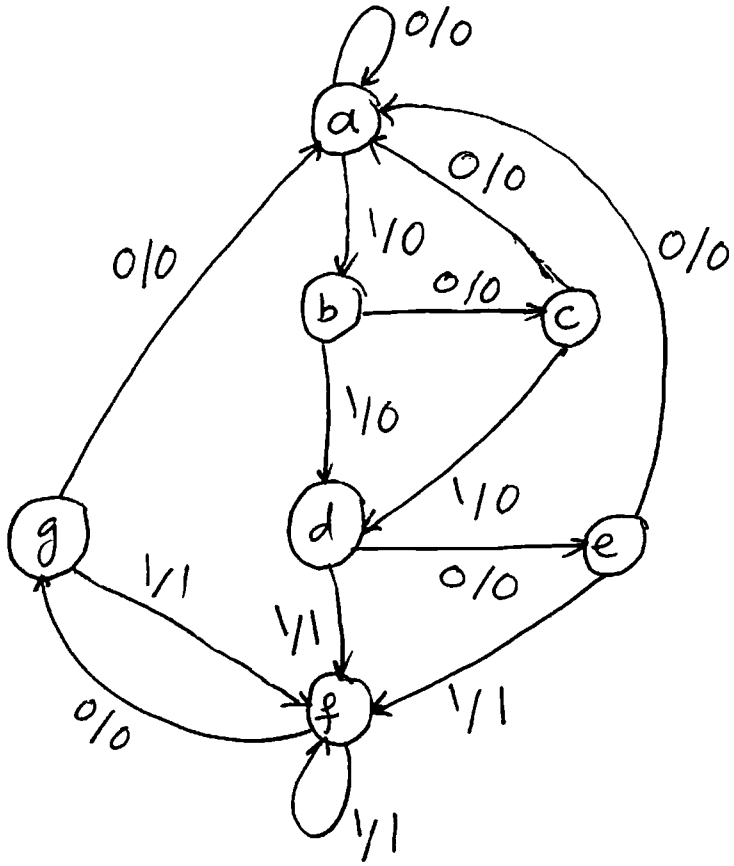
When we start with a functional description, we may have defined redundant (equivalent) states. That is, there may be states that act the same (they respond the same way to an input and go to the same next state and generate the same output). These states may be combined. This procedure is called state reduction. The interest in reducing the number of states is that by reducing the number of states, we, possibly,

reduce the number of flip-flops. Remember that the number of states is 2^m where m is the number of flip-flops. This means that when the number of states is reduced often the number of flip-flops is also reduced. For example, if we can reduce the number of states from 6 to 4, the number of flip-flops will be reduced from 3 to 2. However, if we reduce the number of states from, say 8 to 5, we still need 3 flip-flops.

In reducing the number of states, usually, the following fact is used:

Two states are <u>equivalent</u> if, for every input, they give the same output a result into transition to the same next state or to an equivalent state.

Example: Consider a circuit whose state diagram is given as:



The state table for this state diagram is:

Present State	Next State		output	
	x=0	x=1	x=0	x=1
a	a	b	0	0
b	c	d	0	0
c	a	d	0	0
d	e	f	0	1
e	a	f	0	1
f	g	f	0	1
g	a	f	0	1

Using the above equivalency condition, we observe that e and g are equivalent states since for $x=0$ both transition to state a and both generate the output 0. For $x=1$, the next state for both is f and the output is 1.

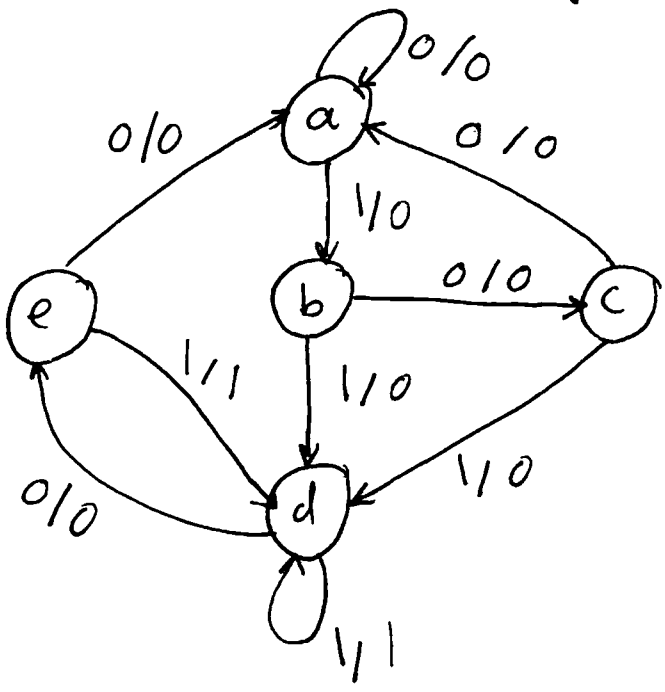
Deleting state g from the table (in fact combining e and g), we have:

Present State	Next State		Output	
	$x=0$	$x=1$	$x=0$	$x=1$
a	a	b	0	0
b	c	d	0	0
c	a	d	0	0
d	e	f	0	1
e	a	f	0	1
f	e	f	0	1

Note that, we have also replaced g with e in the table (in the line for state f). Now, states d and f are equivalent. If we combine states d and f in one state, say, state d, we get:

Present State	Next State		Output	
	x=0	x=1	x=0	x=1
a	a	b	0	0
b	c	d	0	0
c	a	d	0	0
d	e	d	0	1
e	a	d	0	1

The reduced state diagram is:



State Assignment

In order to be able to design the sequential circuit, we need to assign binary values to states. When the number of states is a power of 2, the choice is clear, we assign binary values from 0 to 2^{m-1} to states.

When the number of states is not a power of 2, there are several choices; we may use the first S numbers counting from 0 to S-1, where S is the number of states. For example, for the above 5-state circuit, we may use 000, 001, 010, 011, and 100.

Another choice is Gray Code. Since in a Gray code any two consecutive numbers differ in only one bit, we may have some simplification in logic design.

Third option is to use <u>one-hot</u> assignment. Here we use S bits to represent each state and only one bit is equal to 1 in each state index. This way there is one flip-flop per state. This makes the design of combinational circuit trivial, but, results in waste of flip-flops and is only wise if there are lots of flip-flops on the chip. These three state assignment schemes are shown in the following table.

State	Binary	Gray Code	One-hot
a	000	000	00001
b	001	001	00010
c	010	011	00100
d	011	010	01000
e	100	110	10000

Using the binary assignment scheme, the reduced state table of the previous example can be written as:

Present State	Next State		Output	
	x=0	x=1	x=0	x=1
000	000	001	0	0
001	010	011	0	0
010	000	011	0	0
011	100	011	0	1
100	000	011	0	1

Design Procedure

A design (synthesis) starts with a description of how the circuit should work and ends up with a circuit diagram consisting of gates and flip-flops.

A design process consists of the following tasks:

1- Translating the word description of the circuit into a state diagram.

2- Reducing the number of states if possible.

3- Assigning binary values to the states.

4- Obtaining the binary-coded state table.

5- Choosing the type of flip-flops to be used.

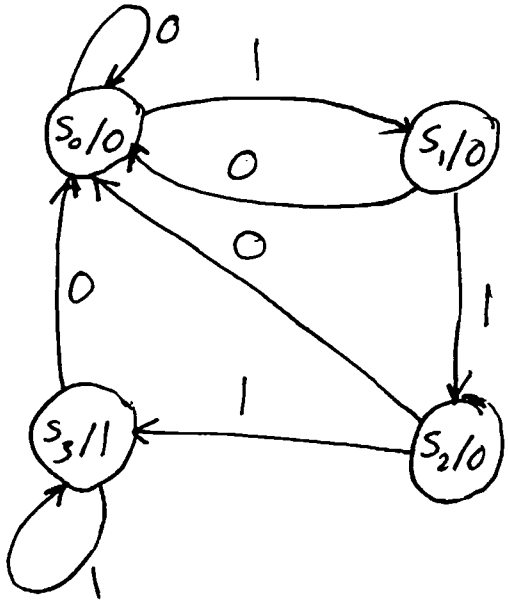
6- Writing down the flip-flop input equations and output equations.

7- Drawing the circuit diagram.

Example: We start with the following description:

Design a circuit that detects the occurence of three (or more) consecutive ones.

The circuit can be represented by a state diagram with 4 states. Each state represents the number of consecutive 1's observed, i.e., 0, 1, 2, 3. We name these states S_0, S_1, S_2, S_3. Each time a zero is observed system goes to S_0. After the 1st one, the system transitions from S_0 to S_1. The system jumps back to S_0 if while in S_1, a zero occurs, else, it goes to S_2. From S_2, the system goes to S_0 or S_3 depending on the value of the input. In all states, except S_3, the output is zero.



Drawing the above state diagram completes the first step in the design. It is clear that the number of states cannot be reduced.

Next, we index the four states with two bits: 00, 01, 10, 11. Since, we have four states, we need two flip-flops.

Next, we draw the transition table:

Present state		Input	Next state		Output
A	B	x	A	B	y
0	0	0	0	0	0
0	0	1	0	1	0
0	1	0	0	0	0
0	1	1	1	0	0
1	0	0	0	0	0
1	0	1	1	1	0
1	1	0	0	0	1
1	1	1	1	1	1

Synthesis with D flip-flops

The next step is to choose the type of the flip-flops. Assume that we choose

D flip-flops. Then we have to write the following flip-flop input equations:

$$D_A(A, B, x) = A(t+1) = \Sigma(3, 5, 7)$$

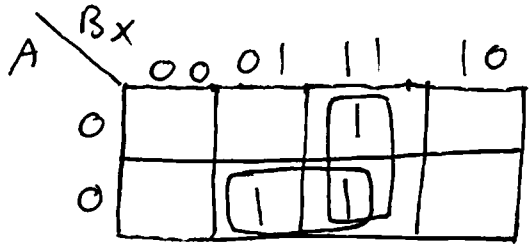
and

$$D_B(A, B, x) = B(t+1) = \Sigma(1, 5, 7)$$

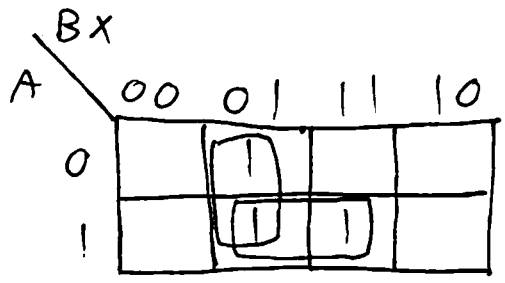
and the output equation:

$$y(A, B, x) = \Sigma(6, 7)$$

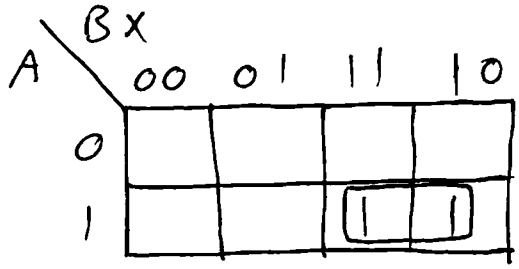
Using the K-maps we find D_A, D_B and y:



$$D_A = Ax + Bx$$

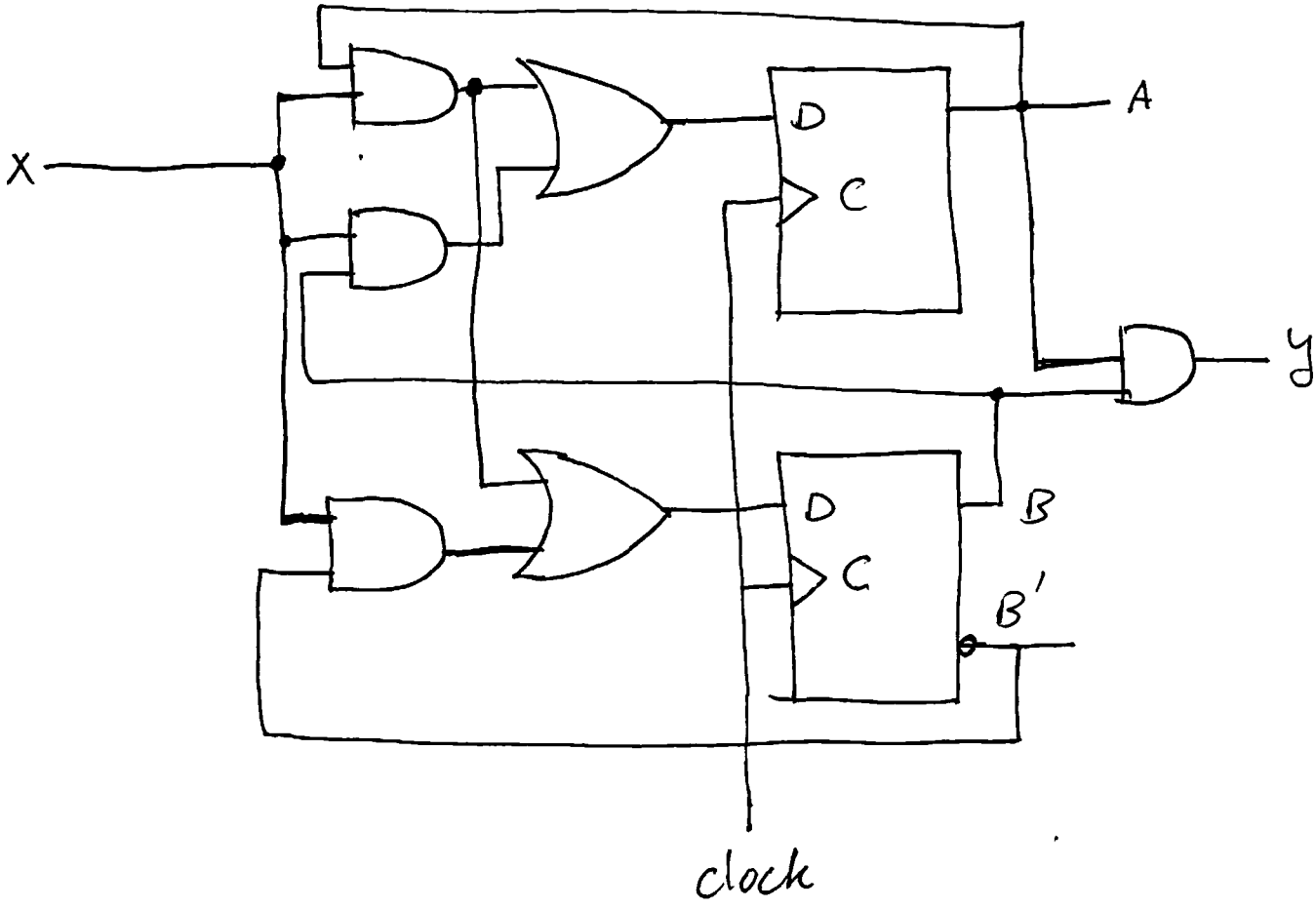


$$D_B = Ax + Bx'$$



$$y = AB$$

Now, it is easy to draw the logic diagram



Excitation tables :

The good thing about D flip-flops is that we can find their input equation directly from the transition table. With JK or T flip-flops, on the other hand, we need to derive a functional relationship between the state transition table and the input equations. To do this, we need a table that determines the value of the input equations

for each state transition. Such a table is called an excitation table.

The excitation table for the JK flip-flop is:

Q(t)	Q(t+1)	J	K
0	0	0	X
0	1	1	X
1	0	X	1
1	1	X	0

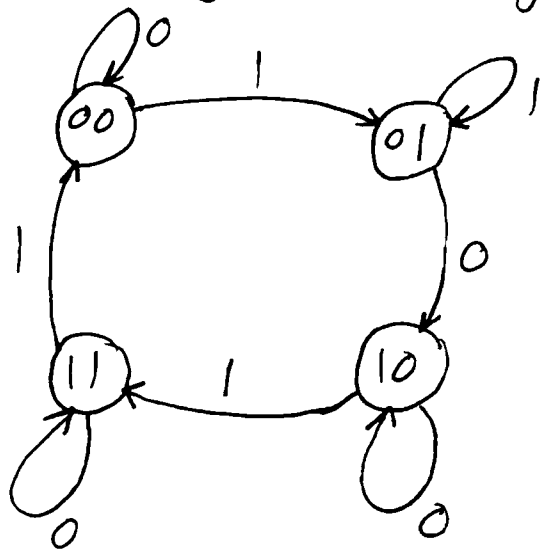
Note that when a JK FF is in state zero and we want it to stay in state zero, we need to have $J=0$ and whether K is 0 or 1 is irrelevant. And a transition from state 0 to 1 happens for $J=1$, $K=0$ and $J=1, K=1$. So, $J=1$ and $K=X$ (don't care). Other table entries are clear.

For a T flip-flop the excitation table is

Q(t)	Q(t+1)	T
0	0	0
0	1	1
1	0	1
1	1	0

Synthesis with JK flip-flops:

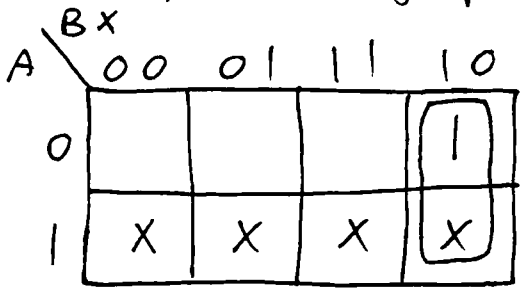
Assume that we want to design the circuit with the following state diagram using JK flip-flop

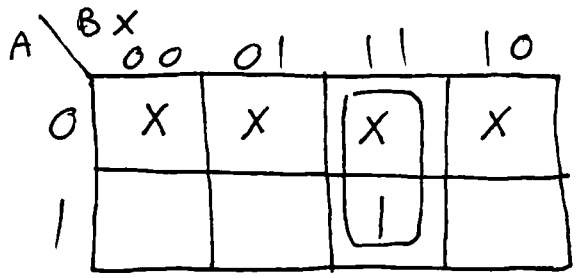


The state table is:

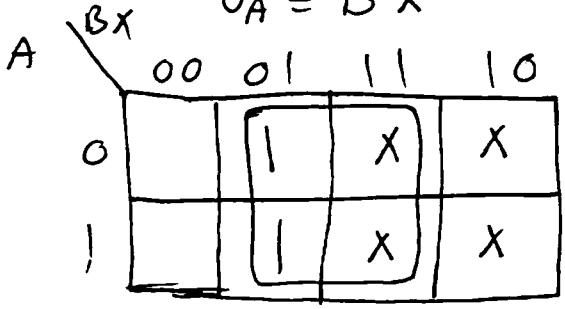
Present state A	Present state B	input X	Next State A	Next State B	FF Inputs J_A	K_A	J_B	K_B
0	0	0	0	0	0	X	0	X
0	0	1	0	1	0	X	1	X
0	1	0	1	0	1	X	X	1
0	1	1	0	1	0	X	X	0
1	0	0	1	0	X	0	0	X
1	0	1	1	1	X	0	1	X
1	1	0	1	1	X	0	X	0
1	1	1	0	0	X	1	X	1

Using K-maps, we can find the logic for the inputs to the flip-flops:

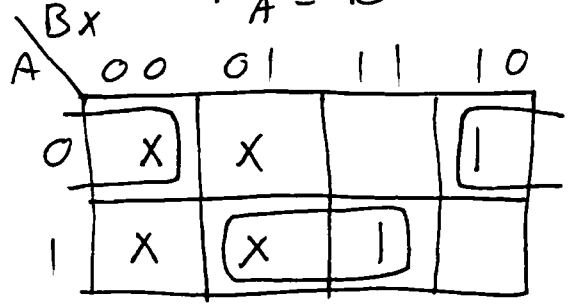




$J_A = Bx'$



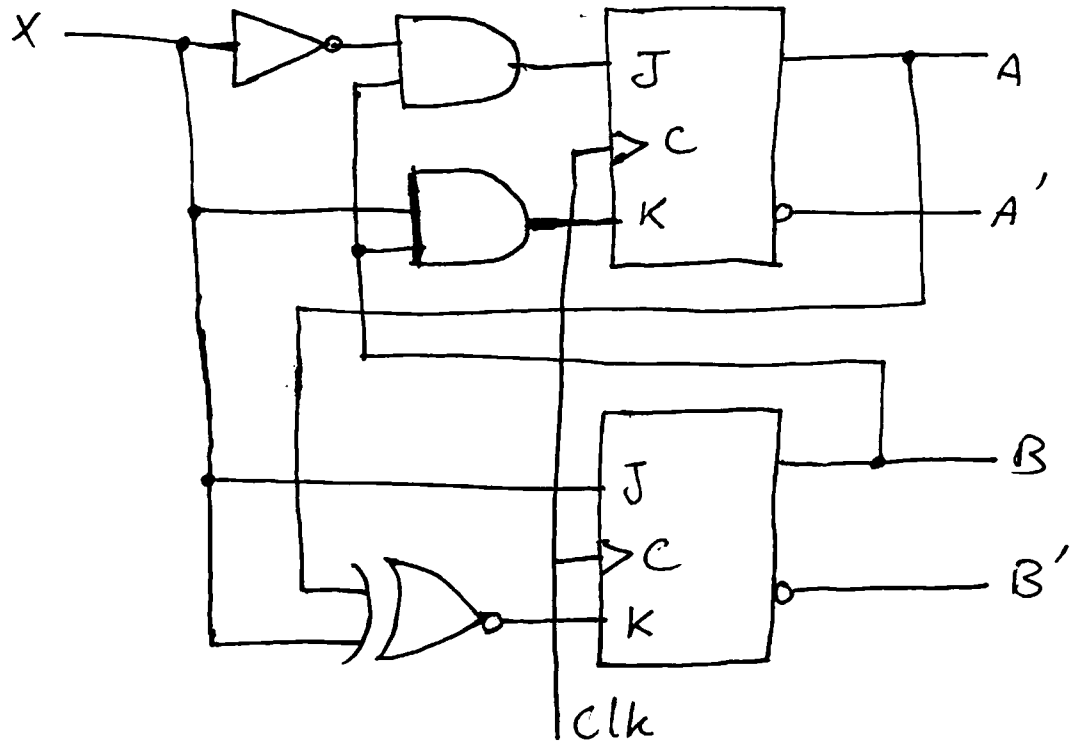
$K_A = Bx$



$J_B = x$

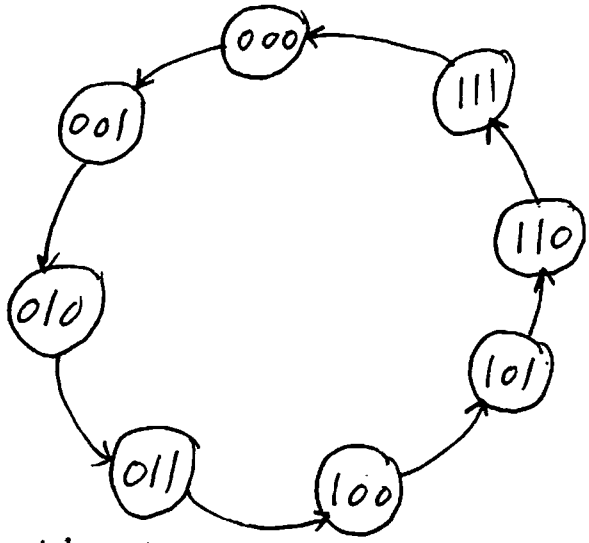
$K_B = Ax + A'x' = (A \oplus x)'$

The logic diagram will be:



Synthesis Using T flip-flops

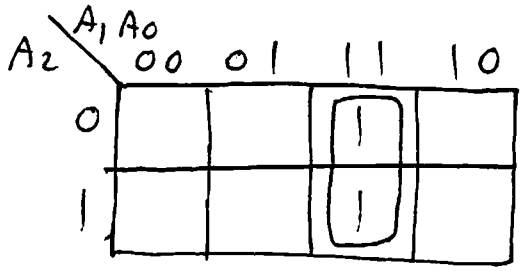
Take as an example the design of a binary counter. An n-bit binary counter counts from 0 to $2^n - 1$. For example a 3-bit binary counter counts from 0 to 7. The state diagram of a 3-bit counter is:



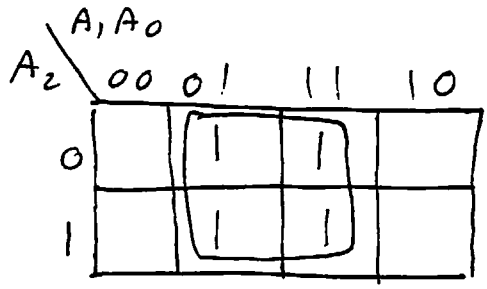
The state table is:

Present state			next state			Flip-flop inputs		
A_2	A_1	A_0	A_2	A_1	A_0	T_{A_2}	T_{A_1}	T_{A_0}
0	0	0	0	0	1	0	0	1
0	0	1	0	1	0	0	1	1
0	1	0	0	1	1	0	0	1
0	1	1	1	0	0	1	1	1
1	0	0	1	0	1	0	0	1
1	0	1	1	1	0	0	1	1
1	1	0	1	1	1	0	0	1
1	1	1	0	0	0	1	1	1

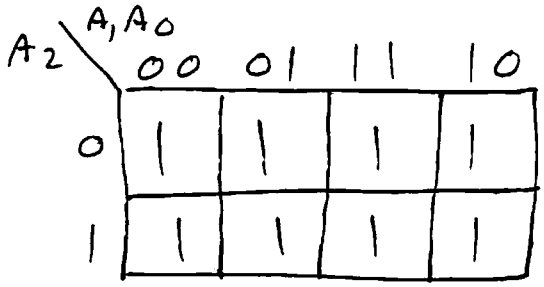
The flip-flop input equations are found using the following K-maps



$T_{A_2} = A_1 A_0$



$T_{A_1} = A_0$



$T_{A_0} = 1$

The logic diagram is: