Synchronous Sequential Logic II

CS207 Lecture 10

James YU

Apr. 22, 2020

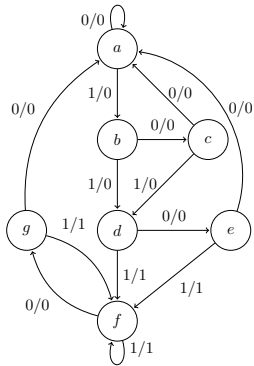
南方科技大学
SOUTHERN UNIVERSITY OF SCIENCE AND TECHNOLOGY

Design of clocked sequential circuits

- The *analysis* of sequential circuits starts from a circuit diagram and culminates in a state table or diagram.
- The *design* (synthesis) of a sequential circuit starts from a set of specifications and culminates in a logic diagram.
- Two sequential circuits may exhibit the same input–output behavior, but have a different number of internal states in their state diagram.
  - In general, reducing the number of flipflops reduces the cost of a circuit.
  - *State reduction* and *state assignment*.

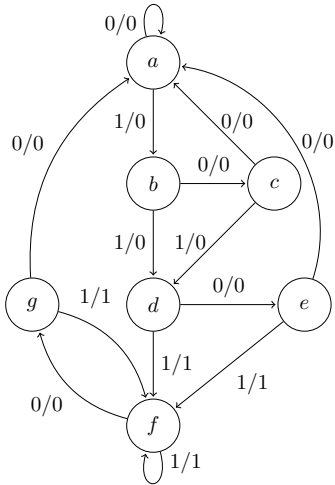
State reduction

- The reduction in the number of flip-flops in a sequential circuit is referred to as the *state-reduction* problem.
  - Reducing the number of states in a state table, while keeping the external input–output requirements unchanged.
- We illustrate the procedure with an example.



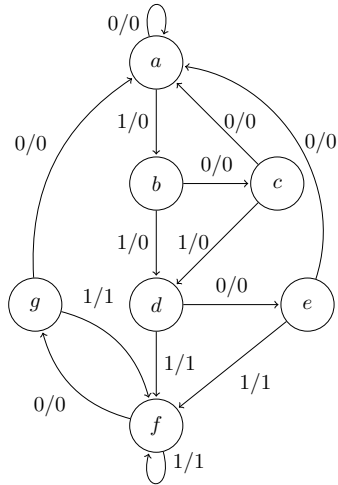
State reduction

- In our example, only the input–output sequences are important.
  - The internal states are used merely to provide the required sequences.
- There are an infinite number of input sequences that may be applied to the circuit.
  - Each results in a unique output sequence.
  - Example, consider the input sequence `01010110100` starting from state a.



State reduction

State	Input	Output
a	0	0
a	1	0
b	0	0
c	1	0
d	0	0
e	1	1
f	1	1
f	0	0
g	1	1
f	0	0
g	0	0
a		



State reduction

- Let us assume that we have found a sequential circuit whose state diagram has fewer than seven states.
  - If identical input sequences are applied to the two circuits and identical outputs occur for all input sequences, then the two circuits are said to be equivalent.
  - One may be replaced by the other.
- The problem of state reduction is to find ways of reducing the number of states in a sequential circuit without altering the input–output relationships.

State reduction

- First, we need the state table:

Present	Next		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

State reduction

- Two states are said to be equivalent if, for each member of the set of inputs, they give exactly the same output and send the circuit either to the same state or to an equivalent state.
  - States g and e are equivalent, and one of these states can be removed.

Present	Next		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

State reduction

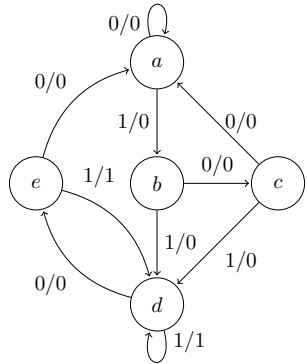
- Now states f and d are equivalent, and state f can be removed and replaced by d.

Present	Next		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

State reduction

- Finally we have

Present	Next		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 sequential circuit of this example was reduced from seven to five states.

State assignment

- In order to design a sequential circuit with physical components, it is necessary to assign unique coded binary values to the states.
- For a circuit with m states, the codes must contain n bits, where $2^n \geq m$.
  - Before the state reduction, we must assign binary values to seven states; the remaining state is unused.
  - If the state table after reduction is used, only five states need binary assignment, and we are left with three unused states.
  - Unused states are treated as don't-care conditions during the design.
- Since don't-care conditions usually help in obtaining a simpler circuit, it is more likely but not certain that the circuit with five states will require fewer combinational gates than the one with seven states.

State assignment

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

- One-hot encoding usually leads to simpler decoding logic for the next state and output.

Design procedure

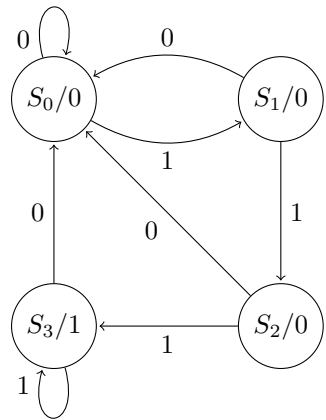
- The design of a clocked sequential circuit starts from a set of specifications.
  - Different from combinational circuits, a sequential circuit requires a state table.
- It consists of choosing the flip-flops and combinational gates.
  - Number of flip-flops determined from the number of states.
  - Combinational circuits derived from state table.
- The procedure for designing synchronous sequential circuits can be summarized by a list of recommended steps:
  1. From the word description and specifications of the desired operation, derive a state diagram for the circuit.
  2. Reduce the number of states if necessary.
  3. Assign binary values to the states.
  4. Obtain the binary-coded state table.
  5. Choose the type of flip-flops to be used.
  6. Derive the simplified flip-flop input equations and output equations.
  7. Draw the logic diagram.

Design procedure

- The first step is a critical part of the process, because succeeding steps depend on it.
  - We will give one simple example to demonstrate how a state diagram is obtained from a word specification.

Design procedure

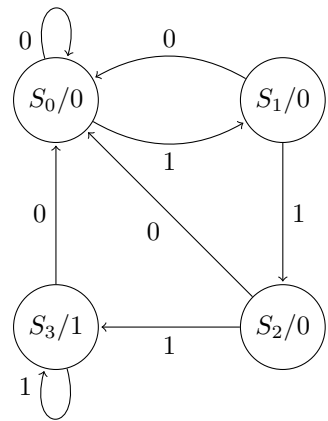
- Suppose we wish to design a circuit that detects a sequence of three or more consecutive 1's in a string of bits coming through an input line.
- Starting with state $S0$, the reset state.
  - If the input is `0`, the circuit stays in $S0$.
  - If the input is `1`, it goes to state $S1$ to indicate that a `1` was detected.
- If the next input is `1`, the change is to state $S2$ to indicate the arrival of two consecutive `1`'s, but if the input is `0`, the state goes back to $S0$.
- The third consecutive `1` sends the circuit to state $S3$. If more `1`'s are detected, the circuit stays in $S3$. Any `0` input sends the circuit back to $S0$.



Design procedure

- Once the state diagram has been derived, the rest of the design follows a straightforward synthesis procedure.

Present		Input	Next		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



Design procedure

- Once the state diagram has been derived, the rest of the design follows a straightforward synthesis procedure.

Present		Input	Next		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

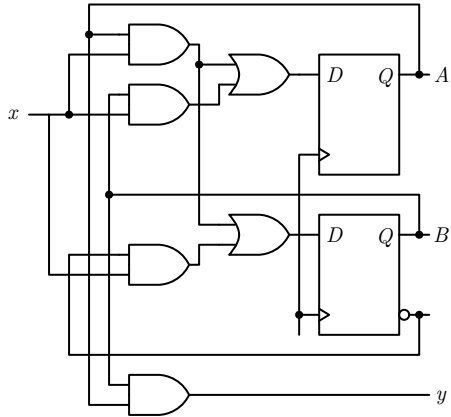
- We use D flip-flop.
- The flip-flop input equations can be obtained:

$$
\begin{aligned}
A(t+1) &= D_A(A,B,x) \\
&= \sum(3,5,7), \\
B(t+1) &= D_B(A,B,x) \\
&= \sum(1,5,7), \\
y(A,B,x) &= \sum(6,7).
\end{aligned}
$$

Design procedure

- The equations are simplified by means of K-map:

$$D_A = Ax + Bx,$$
$$D_B = Ax + B'x,$$
$$y = AB.$$



Excitation tables

- The design of a sequential circuit with flip-flops other than the D type is complicated by the fact that the input equations for the circuit must be derived indirectly from the state table.
  - When D-type flip-flops are employed, the input equations are obtained directly from the next state.
  - This is not the case for the JK and T types of flip-flops.
- In order to determine the input equations for these flip-flops, it is necessary to derive a functional relationship between the state table and the input equations.
- During the design process, we usually know the transition from the present state to the next state and wish to find the flip-flop input conditions that will cause the required transition.
- For this reason, we need a table that lists the required inputs for a given change of state: *excitation table*.

Excitation tables

- JK flip-flop:

$Q(t)$	$Q(t=1)$	J	K
0	0	0	X
0	1	1	X
1	0	X	1
1	1	X	0

- T flip-flop:

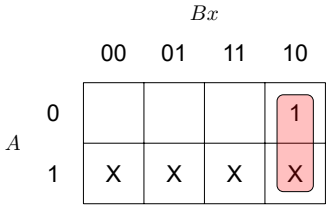
$Q(t)$	$Q(t=1)$	T
0	0	0
0	1	1
1	0	1
1	1	0

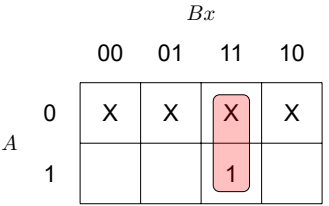
Synthesis using JK flip-flops

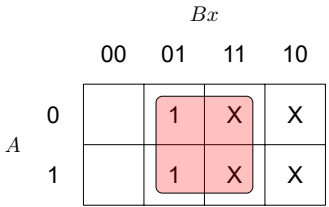
- The manual synthesis procedure for sequential circuits with JK flip-flops is the same as with D flip-flops.
  - Except that the input equations must be evaluated from the present state to the next-state transition derived from the excitation table.

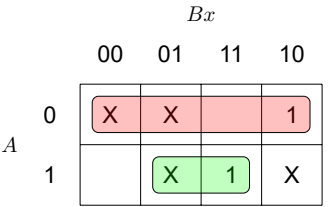
Present		Input	Next		Flip-flip Inputs			
A	B	x	A	B	J_A	K_A	J_B	K_B
0	0	0	0	0	0	X	0	X
0	0	1	0	1	0	X	1	X
0	1	0	1	0	1	X	X	1
0	1	1	0	1	0	X	X	0
1	0	0	1	0	X	0	0	X
1	0	1	1	1	X	0	1	X
1	1	0	1	1	X	0	X	0
1	1	1	0	0	X	1	X	1

Synthesis using JK flip-flops



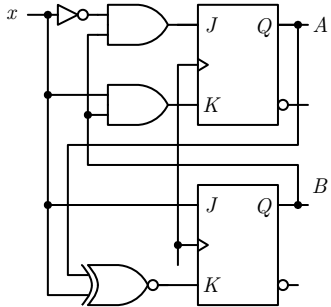






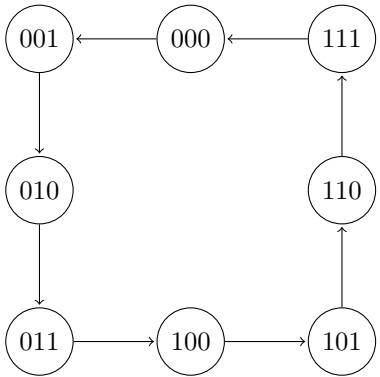
Synthesis using JK flip-flops

$$
\begin{aligned}
J_A &= Bx', \\
J_B &= x, \\
K_A &= Bx, \\
K_B &= (A \oplus x)'.
\end{aligned}
$$



Synthesis using T flip-flops

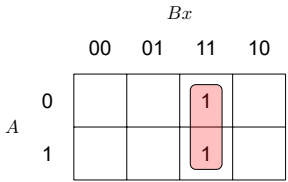
- The procedure for synthesizing circuits using T flip-flops will be demonstrated by designing a binary counter.
- An n-bit binary counter consists of n flip-flops that can count in binary from 0 to $2^n - 1$.

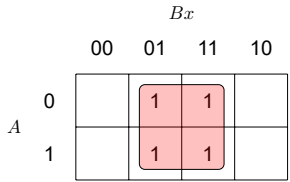


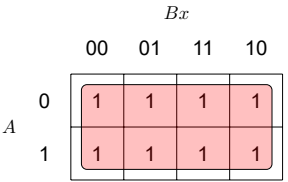
Synthesis using T flip-flops

Present			Next			Flip-flip Inputs		
A_2	A_1	A_0	A_2	A_1	A_0	T_{A2}	T_{A1}	T_{A0}
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	1	1
1	1	1	0	0	0	1	1	1

Synthesis using T flip-flops







$$T_{A2} = A_1 A_0,$$
$$T_{A1} = A_0,$$
$$T_{A0} = 1.$$