Digital Circuit And Logic Design I

Lecture 9

Outline

- Sequential Logic Design Principles (2)

  1. Clocked Synchronous State-Machine Design

Sequential Logic Design Principles (2)

1. Clocked Synchronous State-Machine Design

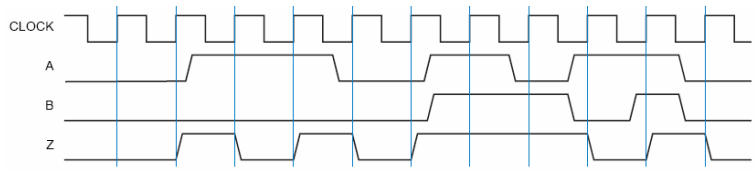
- The steps for designing a clocked synchronous state machine, starting from a word description or specification, are just about the reverse of the analysis steps that we used in the previous lecture:
  1. Construct a state/output table corresponding to the word description or specification, using mnemonic names for the states. (It's also possible to start wit a state diagram)
  2. (Optional) Minimize the number of states in the state/output table.
  3. Choose a set of state variables and assign state-variable communications to the named states.

1. Clocked Synchronous State-Machine Design (cont.)

4. Substitute the state-variable combinations into the state/output table to create a transition/output table that shows the desired next state-variable combination and output for each state/input combination.
5. Choose a flip-flop type (e.g., D or J-K) for the state memory.
6. Construct an excitation table that shows the excitation values required to obtain the desired next state for each state/input combination.
7. Derive excitation equations from the excitation table.
8. Derive output equations from the transition/output table.
9. Draw a logic diagram that shows the state-variable storage elements and realizes the excitation and output equations.

1. Clocked Synchronous State-Machine Design (cont.)

- State-Table Design Example
  - Design a clocked synchronous state machine with two inputs, A and B, and a single output Z that is 1 if:
    - A had the same value at each of the two previous clock ticks, or
    - B has been 1 since the last time that the first condition was true.
  - Otherwise, the output should be 0.



Pictures from text book DDPP

1. Clocked Synchronous State-Machine Design (cont.)

- State-Table Design Example (cont.)
  - From the word description, we know that our example is Moore machine – its output depends only on the current state.

		A B				
Meaning	S	00	01	11	10	Z
Initial state	INIT					0
	. . .					
	. . .					
	. . .					
	S*					

		A B				
Meaning	S	00	01	11	10	Z
Initial state	INIT	A0	A0	A1	A1	0
Got a 0 on A	A0					0
Got a 1 on A	A1					0
	S*					

Pictures from text book DDPP

1. Clocked Synchronous State-Machine Design (cont.)

- State-Table Design Example (cont.)

		A B				
Meaning	S	00	01	11	10	Z
Initial state	INIT	A0	A0	A1	A1	0
Got a 0 on A	A0	OK	OK	A1	A1	0
Got a 1 on A	A1					0
Got two equal A inputs	OK					1
	S*					

		A B				
Meaning	S	00	01	11	10	Z
Initial state	INIT	A0	A0	A1	A1	0
Got a 0 on A	A0	OK	OK	A1	A1	0
Got a 1 on A	A1	A0	A0	OK	OK	0
Got two equal A inputs	OK					1
	S*					

		A B				
Meaning	S	00	01	11	10	Z
Initial state	INIT	A0	A0	A1	A1	0
Got a 0 on A	A0	OK	OK	A1	A1	0
Got a 1 on A	A1	A0	A0	OK	OK	0
Got two equal A inputs	OK	?	OK	OK	?	1
	S*					

		A B				
Meaning	S	00	01	11	10	Z
Initial state	INIT	A0	A0	A1	A1	0
Got a 0 on A	A0	OK0	OK0	A1	A1	0
Got a 1 on A	A1	A0	A0	OK1	OK1	0
Two equal, A=0 last	OK0					1
Two equal, A=1 last	OK1					1
	S*					

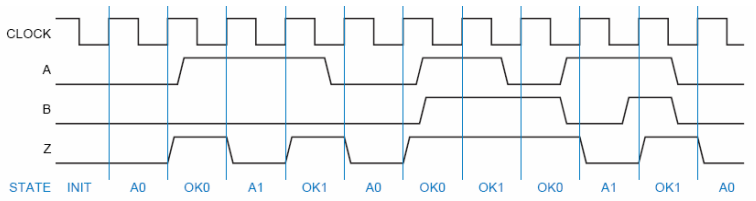
Pictures from text book DDPP

1. Clocked Synchronous State-Machine Design (cont.)

- State-Table Design Example (cont.)

Meaning	S	A B 00	01	11	10	Z
Initial state	INIT	A0	A0	A1	A1	0
Got a 0 on A	A0	OK0	OK0	A1	A1	0
Got a 1 on A	A1	A0	A0	OK1	OK1	0
Two equal, A=0 last	OK0	OK0	OK0	OK1	A1	1
Two equal, A=1 last	OK1					1
		S*				

Meaning	S	A B 00	01	11	10	Z
Initial state	INIT	A0	A0	A1	A1	0
Got a 0 on A	A0	OK0	OK0	A1	A1	0
Got a 1 on A	A1	A0	A0	OK1	OK1	0
Two equal, A=0 last	OK0	OK0	OK0	OK1	A1	1
Two equal, A=1 last	OK1	A0	OK0	OK1	OK1	1
		S*				



Pictures from text book DDPP

1. Clocked Synchronous State-Machine Design (cont.)

- State Minimization
  - The basic idea of formal minimization procedures is to identify **equivalent states**, where two states are equivalent if it is impossible to distinguish the state by observing only the current and future outputs of the machine (and not the internal state variables).
  - A pair of equivalent state can be replaced by a single state.
  - Two states S1 and S2 are equivalent if two conditions are true.
    - First, S1 and S2 must produces the same value at the state-machine output(s); in Mealy machine, this must be true for all input combination.
    - Second, for each input combination, S1 and S2 must have either the same next state or equivalent next state.

1. Clocked Synchronous State-Machine Design (cont.)

- State Minimization (cont.)

Nonminimal state/output table

(a)

Meaning	S	A B 00	01	11	10	Z
Initial state	INIT	A0	A0	A1	A1	0
Got a 0 on A	A0	OK00	OK00	A1	A1	0
Got a 1 on A	A1	A0	A0	OK11	OK11	0
Got 00 on A	OK00	OK00	OK00	OKA1	A1	1
Got 11 on A	OK11	A0	OKA0	OK11	OK11	1
OK, got a 0 on A	OKA0	OK00	OK00	OKA1	A1	1
OK, got a 1 on A	OKA1	A0	OKA0	OK11	OK11	1
		S*				

(b)

Meaning	S	A B 00	01	11	10	Z
Initial state	INIT	A0	A0	A1	A1	0
Got a 0 on A	A0	OK00	OK00	A1	A1	0
Got a 1 on A	A1	A0	A0	OK11	OK11	0
Got 00 on A	OK00	OK00	OK00	A001	A1	1
Got 11 on A	OK11	A0	A110	OK11	OK11	1
Got 001 on A, B=1	A001	A0	AE10	OK11	OK11	1
Got 110 on A, B=1	A110	OK00	OK00	AE01	A1	1
Got bb...10 on A, B=1	AE10	OK00	OK00	AE01	A1	1
Got bb...01 on A, B=1	AE01	A0	AE10	OK11	OK11	1
		S*				

minimal state/output table equivalent to (a) and (b)

Meaning	S	A B 00	01	11	10	Z
Initial state	INIT	A0	A0	A1	A1	0
Got a 0 on A	A0	OK0	OK0	A1	A1	0
Got a 1 on A	A1	A0	A0	OK1	OK1	0
Two equal, A=0 last	OK0	OK0	OK0	OK1	A1	1
Two equal, A=1 last	OK1	A0	OK0	OK1	OK1	1
		S*				

Pictures from text book DDPP

1. Clocked Synchronous State-Machine Design (cont.)

- State Assignment
  - The next step in the design process is to determine how many binary variables are required to represent the states in the state table
  - And to assign a specific combination to each named state.
  - We'll call the binary combination assigned to a particular state a **coded state**.
  - The total number of states in a machine with n flip-flops is 2^n, so the number of flip-flops needed to code s states is $\lceil \log_2 s \rceil$, the smallest integer greater than or equal to $\log_2 s$.
  - The our example machine has five states, so it requires at least three flip-flops.

1. Clocked Synchronous State-Machine Design (cont.)

- State Assignment (cont.)
  - The simplest assignment of s coded states to 2^n possible states is to use the first s binary integers in binary counting order
  - However, the simplest state assignment does not always lead to the simplest excitation equations, output equations, and resulting logic circuit.
  - In fact, the state assignment often has a major effect on circuit cost, and it may interact with other factors.
  - So, how do we choose the best assignment for a given problem?
  - The only formal way to find the best assignment is to try all the assignments. That's too much work.

1. Clocked Synchronous State-Machine Design (cont.)

Possible state assignments for the example machine

	Assignment			
State name	Simplest Q1–Q3	Decomposed Q1–Q3	One-hot Q1–Q5	Almost one-hot Q1–Q4
INIT	000	000	00001	0000
A0	001	100	00010	0001
A1	010	101	00100	0010
OK0	011	110	01000	0100
OK1	100	111	10000	1000

Pictures from text book DDPP

1. Clocked Synchronous State-Machine Design (cont.)

- State Assignment (cont.)
  - Guidelines for making reasonable state assignments:
    - Choose an initial coded state into which the machine can easily be forced at reset (00…00 or 11…11 in typical circuits).
    - Minimize the number of state variables that change on each transition
    - Maximize the number of state variables that don't change in a group of related states (i.e., a group of states in which most of transitions stay in the group).
    - Exploit symmetries in the problem specification and the corresponding symmetries in the state table. That is, suppose that one state or group of states means almost the same thing as another. Once an assignment has been established for the first, a similar assignment, differing only in one bit, should be used for the second.

1. Clocked Synchronous State-Machine Design (cont.)

- State Assignment (cont.)
  - If there are unused states, then choose the "best" of the available state-variable combinations to achieve the foregoing goals That is, don't limit the choice of coded states to the first s n-bit integers.
  - Decompose the set of state variables into individual bits or fields, where each bit or field has a well-defined meaning with respect to the input effects or output behavior of the machine.
  - Consider using more than minimum number of state variables to make a decomposed assignment possible.

1. Clocked Synchronous State-Machine Design (cont.)

- State Assignment (cont.)
  - There are two approaches to handle with **unused states**:
    - **Minimal risk**. This approach assumes that it is possible for the state machine somehow to get into one of the unused (or "illegal") states. Therefore, all of the unused state-variable combinations are identified and explicit next-state entries are made so that, for any input combination, the unused states go to the "initial" state, the "idle" state, or other "safe" state.
    - **Minimal cost**. This approach assumes that the machine will never enter an unused state. Therefore, in the transition and excitation tables, the next-state entries of the unused states can be marked as "don't-cares."

1. Clocked Synchronous State-Machine Design (cont.)

- Synthesis Using D Flip-Flops
  - Coded states are substituted for named states in the (possibly minimized) state table to obtain a **transition table**.
  - The transition table shows the next coded state for each combination of current coded state and input.
  - The next step is to write an **excitation table** that show, for each combination of coded state and input, the flip-flop excitation input values needed to make the machine to to the desired next coded state. This structure and content of this table depend on the type of flip-flops that are used.
  - A D flip-flop has the simplest characteristic equation, Q* = D. So, the excitation table is identical to the transition table, except for labeling of its entries.

1. Clocked Synchronous State-Machine Design (cont.)

- Synthesis Using D Flip-Flops (cont.)

Transition and output table for example problem.

	AB				
Q1 Q2 Q3	00	01	11	10	Z
000	100	100	101	101	0
100	110	110	101	101	0
101	100	100	111	111	0
110	110	110	111	101	1
111	100	110	111	111	1
	Q1* Q2* Q3*				

Excitation and output table for example problem using D flip-flops.

	AB				
Q1 Q2 Q3	00	01	11	10	Z
000	100	100	101	101	0
100	110	110	101	101	0
101	100	100	111	111	0
110	110	110	111	101	1
111	100	110	111	111	1
	D1 D2 D3				

Pictures from text book DDPP

1. Clocked Synchronous State-Machine Design (cont.)

- Synthesis Using D Flip-Flops (cont.)
  - The excitation table of example machine is like a truth table for three combinational logic functions (D1, D2, D3) of five variables (A, B, Q1, Q2, Q3).
  - Accordingly, we can design circuits to realize these functions using any of the combinational design methods.
  - In particular, we can transfer the information in the excitation table to Karnaugh maps, which we may call **excitation maps**, and find a minimal sum-of-products or product-of-sums expression for each function.

$$D1 = Q1 + Q2' \cdot Q3'$$
$$D2 = Q1 \cdot Q3' \cdot A' + Q1 \cdot Q3 \cdot A + Q1 \cdot Q2 \cdot B$$
$$D3 = Q1 \cdot A + Q2' \cdot Q3' \cdot A$$
$$Z = Q1 \cdot Q2 \cdot Q3' + Q1 \cdot Q2 \cdot Q3 = Q1 \cdot Q2$$

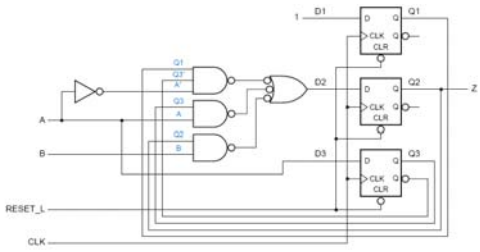
Pictures from text book DDPP

1. Clocked Synchronous State-Machine Design (cont.)

- Synthesis Using D Flip-Flops (cont.)
  - If we choose in our example to derive minimal-cost excitation equations, we write “don’t cares” in the next-state entries for the unused states.
  - The excitation equations obtained from this map are some what simpler than before:

$$D1 = 1$$
$$D2 = Q1 \cdot Q3' \cdot A' + Q3 \cdot A + Q2 \cdot B$$
$$D3 = A$$

$$Z = Q2$$



Pictures from text book DDPP

1. Clocked Synchronous State-Machine Design (cont.)

- Synthesis Using J-K Flip-Flops
  - Up through the state-assignment step, the design procedure with J-K flip-flops is basically the same as with D flip-flops. The only difference is that a designer might select a slightly different state assignment, knowing the sort of behavior that can easily be obtained from J-k flip-flops
  - The big difference occurs in the derivation of an excitation table from the transition table. The required values for J and K are expressed as functions of Q and Q* in a J-K **application table**

Q	*Q**	*J*	*K*
0	0	0	d
0	1	1	d
1	0	d	1
1	1	d	0

Pictures from text book DDPP

1. Clocked Synchronous State-Machine Design (cont.)

- Synthesis Using J-K Flip-Flops
  - Assuming that unused states go to state 000.

	AB				
Q1 Q2 Q3	00	01	11	10	Z
000	1d, 0d, 0d	1d, 0d, 0d	1d, 0d, 1d	1d, 0d, 1d	0
100	d0, 1d, 0d	d0, 1d, 0d	d0, 0d, 1d	d0, 0d, 1d	0
101	d0, 0d, d1	d0, 0d, d1	d0, 1d, d0	d0, 1d, d0	0
110	d0, d0, 0d	d0, d0, 0d	d0, d0, 1d	d0, d1, 1d	1
111	d0, d1, d1	d0, d0, d1	d0, d0, d0	d0, d0, d0	1
	J1 K1, J2 K2, J3 K3				

$J1 = Q2' \cdot Q3'$ $\quad K1 = 0$

$J2 = Q1 \cdot Q3' \cdot A' + Q1 \cdot Q3 \cdot A$ $\quad K2 = Q1' + Q3' \cdot A \cdot B' + Q3 \cdot A' \cdot B'$

$J3 = Q2' \cdot A + Q1 \cdot A$ $\quad K3 = Q1' + A'$

Pictures from text book DDPP

1. Clocked Synchronous State-Machine Design (cont.)

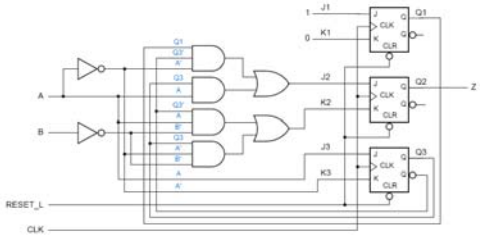
- Synthesis Using J-K Flip-Flops
  - Minimal-cost approach

$J1 = 1$ $\quad K1 = 0$

$J2 = Q1 \cdot Q3' \cdot A' + Q3 \cdot A$ $\quad K2 = Q3' \cdot A \cdot B' + Q3 \cdot A' \cdot B'$

$J3 = A$ $\quad K3 = A'$

  - The state encoding for the J-K circuit is the same as in the D circuit, so the output equation is the same, Z = Q1·Q2 for minimal risk, Z = Q2 for minimal cost.



Pictures from text book DDPP

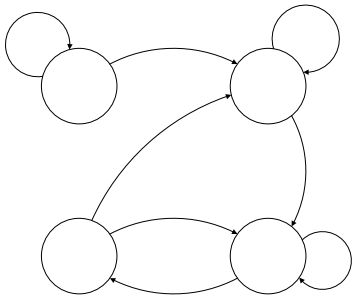
1. Clocked Synchronous State-Machine Design (cont.)

- Designing State Machines Using State Diagram
  - State diagrams are often used to design small- to medium-sized state machines
  - Designing a state diagram is much like designing a state table which is much like writing a program.
  - However, there is one fundamental difference between a state diagram and a state table, a difference that makes state-diagram design simpler but also more error prone:
    - A state table is an exhaustive listing of the next states for each state/input combination. No ambiguity is possible.
    - A state diagram contains a set of arcs labeled with transition expressions. Even when there are many inputs, only one transition expression is required per arc. However, when a state diagram is constructed, there is no guarantee that the transition expressions written on the arcs leaving a particular state cover all input combinations exactly once.

1. Clocked Synchronous State-Machine Design (cont.)

- State-Diagram Design Example
  - Problem Statement
    - Design a sequential circuit having one input, X, and one output, Z. Z in to be 1 whenever the four most recent inputs are 1011, where the rightmost input is the rightmost in the string. Overlapping of sequence is allowed so that the input sequence 1011011 will produce an output 0001001.

State diagram (Mealy machine) of 1011 sequence detector.



1. Clocked Synchronous State-Machine Design (cont.)

- State-Diagram Design Example (cont.)
  - Write a state assignment table, we use simplest assignment in the example.
  - There are four states, so we use 2 flip-flops.
  - Therefore, we can derive transition/output table from state assignment table and state diagram

State assignment (simplest assignment)

State	Q1	Q0
INIT	0	0
Seen 1	0	1
Seen 10	1	0
Seen 101	1	1

Transition/output table

Q1Q0	X=0	X=1
00	00/0	01/0
01	10/0	01/0
10	00/0	11/0
11	01/1	10/0
	Q1*Q0*/Z	

1. Clocked Synchronous State-Machine Design (cont.)

- State-Diagram Design Example (cont.)
  - From transition/output table, we can derive the excitation expression and output expression
    - D1 = X′·Q1′·Q0 + X·Q1
    - D0 = X′·Q1·Q0 + X·Q1′ + X·Q0′
    - Z = X·Q1·Q0