Lecture 12: Finite State Machines

Professor Peter Cheung
Department of EEE, Imperial College London

Points Addressed in this Lecture

- General model of finite state machines
- FSM design procedure

Design of Synchronous Binary Counter

- From the last lecture we have seen:
  - synchronous counters use a register to hold the outputs
  - the inputs of synchronous counters are derived as logical combinations of outputs
  - We will now design the counter as a FSM

Procedure

- Step 1: Start with the transition table of the flip-flop to be used
  - E.g. D-type

Output Transition Required	Input D
0 to 0	0
0 to 1	1
1 to 0	0
1 to 1	1

- Step 2: Construct Karnaugh maps for each input from the state transition table and the flip-flop transition table
  - State Transition Table

Current State			Next State		
Q2	Q1	Q0	Q2+	Q1+	Q0+
0	0	0	0	0	1
0	0	1	0	1	0
0	1	0	0	1	1
0	1	1	1	0	0
1	0	0	1	0	1
1	0	1	1	1	0
1	1	0	1	1	1
1	1	1	0	0	0

– D2

Q2\Q1Q0	00	01	11	10
0	0	0	1	0
1	1	1	0	1

– D1

Q2\Q1Q0	00	01	11	10
0	0	1	0	1
1	0	1	0	1

– D0

Q2\Q1Q0	00	01	11	10
0	1	0	0	1
1	1	0	0	1

- Step 3: Extract Boolean expressions from the Karnaugh maps

Q2\Q1Q0	00	01	11	10
0	0	0	1	0
1	1	1	0	1

$$D2 = Q2.\overline{Q1} + \overline{Q2}.Q1.Q0 + Q2.Q1.\overline{Q0}$$

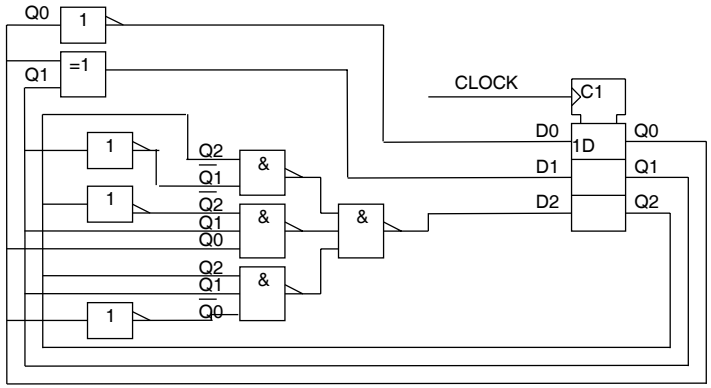
Q2\Q1Q0	00	01	11	10
0	0	1	0	1
1	0	1	0	1

$$D1 = \overline{Q1}.Q0 + Q1.\overline{Q0} = Q1 \oplus Q0$$

Q2\Q1Q0	00	01	11	10
0	1	0	0	1
1	1	0	0	1

$$D0 = \overline{Q0}$$

- Step 4: Implement using combinational logic



Design of Arbitrary Code Counters

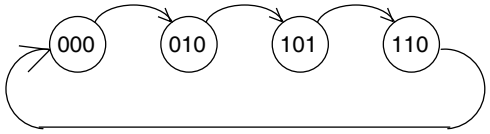
- It may be necessary to design counters which count sequences other than a binary count
- Examples:
  - Counters which never reach their maximum value
    - e.g. a zero-to-nine counter (needs 4 bits)
  - Counters which follow a specific code sequence
    - e.g. 7 segment display sequence counter
- The design procedure presented for binary counters can be extended to arbitrary code counters

Example

- A 3-bit counter to count the decimal sequence 0, 2, 5, 6.
- Counting sequence:

Decimal	Q2	Q1	Q0
0	0	0	0
2	0	1	0
5	1	0	1
6	1	1	0

- State Diagram:

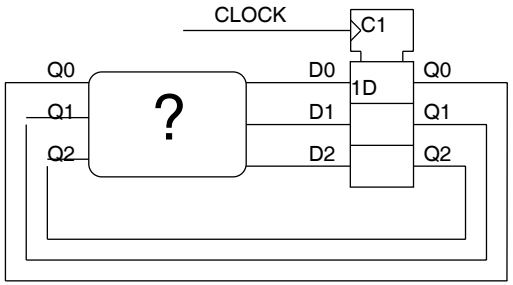


- State Transition Table

Current State			Next State		
Q2	Q1	Q0	Q2+	Q1+	Q0+
0	0	0	0	1	0
0	1	0	1	0	1
1	0	1	1	1	0
1	1	0	0	0	0

- Undefined States
  - In this example, some of the possible 8 states are undefined
  - What happens if the circuit gets into one of the undefined states?
    - e.g. at power-on
  - We will consider two approaches here:
    - 1. Ignore this possibility
      treat the undefined states using "don't care"
    - 2. Take account of this possibility
      map the undefined states explicitly to the "zero" state
  - In both cases we follow the design procedure already presented

- Circuit Diagram



Undefined states as "don't care"

- D2

Q2\Q1Q0	00	01	11	10
0	0	X	X	1
1	X	1	X	0

- D1

Q2\Q1Q0	00	01	11	10
0	1	X	X	0
1	X	1	X	0

- D0

Q2\Q1Q0	00	01	11	10
0	0	X	X	1
1	X	0	X	0

Current State			Next State		
Q2	Q1	Q0	Q2+	Q1+	Q0+
0	0	0	0	1	0
0	1	0	1	0	1
1	0	1	1	1	0
1	1	0	0	0	0

$$D2 = Q0 + \overline{Q2}.Q1.\overline{Q0}$$

$$D1 = \overline{Q1}$$

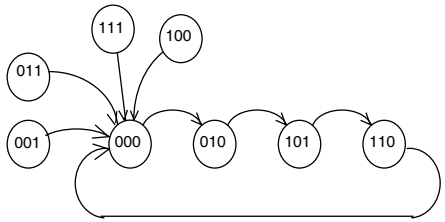
$$D0 = \overline{Q2}.Q1$$

Undefined states map to "zero"

- Modified State Transition Table

Current State			Next State		
Q2	Q1	Q0	Q2+	Q1+	Q0+
0	0	0	0	1	0
0	0	1	0	0	0
0	1	0	1	0	1
0	1	1	0	0	0
1	0	0	0	0	0
1	0	1	1	1	0
1	1	0	0	0	0
1	1	1	0	0	0

- Modified State Diagram



– D2

Q2\Q1Q0	00	01	11	10
0	0	0	0	1
1	0	1	0	0

$$D2 = Q2.\overline{Q1}.Q0 + \overline{Q2}.Q1.\overline{Q0}$$

– D1

Q2\Q1Q0	00	01	11	10
0	1	0	0	0
1	0	1	0	0

$$D1 = \overline{Q2}.\overline{Q1}.\overline{Q0} + Q2.\overline{Q1}.Q0$$

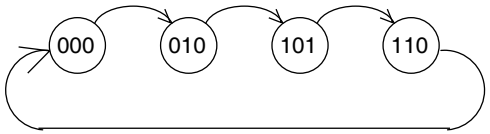
– D0

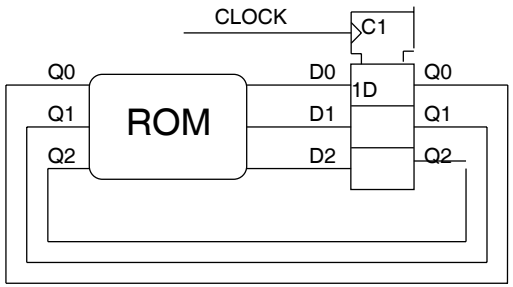
Q2\Q1Q0	00	01	11	10
0	0	0	0	1
1	0	0	0	0

$$D0 = \overline{Q2}.Q1.\overline{Q0}$$

– The term $\overline{Q2}.Q1.\overline{Q0}$ can be re-used to simplify the circuitry
– This implementation is a bit more complicated but will always work (unlike the "don't care" implementation)

Implementing FSM with ROM





Implementing FSM with ROM

Address A2:0			Data D2:0		
A2	A1	A0	D2	D1	D0
0	0	0	0	1	0
0	1	0	1	0	1
1	0	1	1	1	0
1	1	0	0	0	0

All other locations contain 000