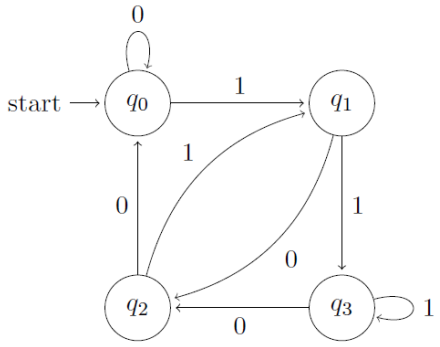
Digital Circuits and Systems

NOC, Spring 2015

Quiz 6 Solutions

For questions, refer to the Quiz page. Only the solutions are given below.

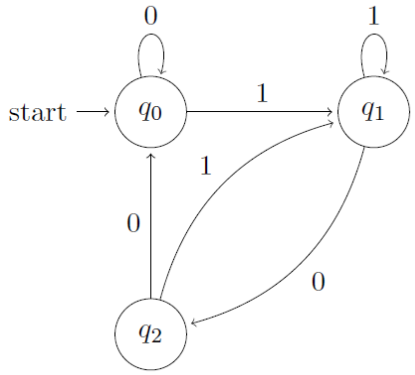
Q1: When a number is divided by 4, then we will get a remainder either 0,1,2, or 3.
So, FSM needs 4 states(q0, q1,q2,q3).



State Transition Table

State	State on input symbol ‘0’	State on input symbol ‘1’
q0(Final)(Initial)	q0	q1
q1	q2	q3
q2	q0	q1
q3	q2	q3

Minimization of the above finite automata is



State	State on input symbol '0'	State on input symbol '1'
q0(Initial)	q0	q1
q1	q2	q1
q2	q0	q1

Answer : 3

Q2: Given condition is number of 0's divisible by 3 and number of 1's divisible by 4:

Number of states required for Mod3 is 3 states.
Number of states required for Mod4 is 4 states.
Based on compound automata , number of states required is :
States in Mod3 * States in Mod4 = 3*4 =12
Number of final states is 1.
so, 12 states are required.
To realize 12 states, 4bits are required,so,four D-flipflops is enough.

Alternatively, go ahead and construct a state diagram to do this. You will realize that it needs 12 states.

Answer : 4

Q3: If a state machine has 3 states and is implemented using 2-D flip-flops, the number of possible assignments are $(2^2)! / (2^2-3)! = 4!/1! = 24$.

Answer : 24

Q4: Figure M1

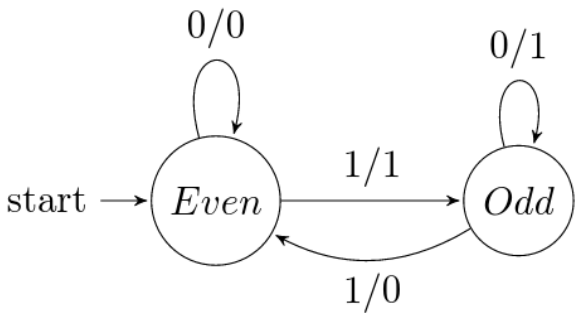
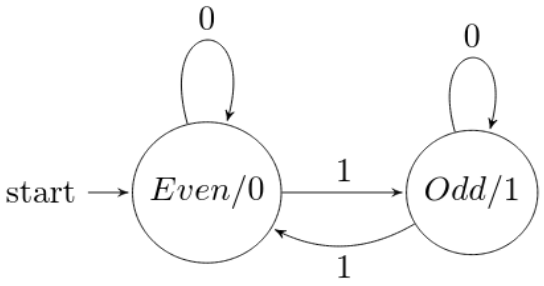


Figure: M2



Observations

1. When we observe Figure M1, the output is associated with transition. So, it is Mealy machine.
2. When we observe Figure M2, the output is associated with state. So, it is Moore machine.
3. Starting state of M1 and M2 is same.
4. on all input symbols both M1 and M2 goes to the same state. Where the output of a transition in M1 is equal to state of M2

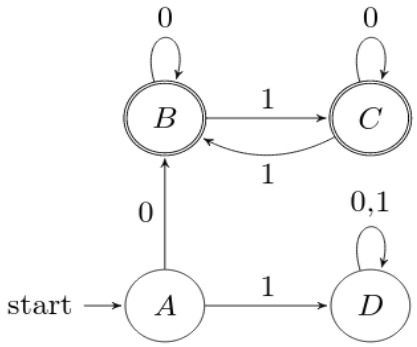
so, from observations 3 and 4, we can conclude M1 is equivalent to M2.

Answer : A,D

Q5: The values of X , Y are D and A respectively.

Answer : B

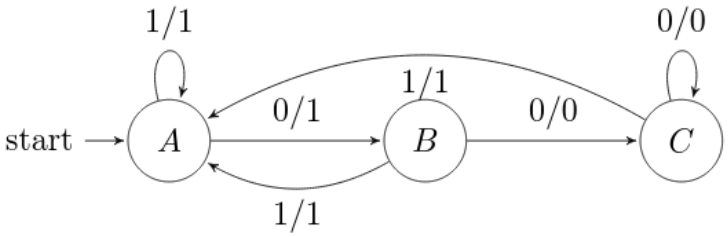
Q6: In the FSM below once we reach the state ‘B’, then the state machine will stayin state ‘B’ or in state ‘C’ for all input symbols. Both the states are final states. So we can merge the two states.



So, number of states in the reduced FSM is 3.

Answer : B

Q7: Equivalent minimized Moore machine of the following Mealy machine



Transition table of Mealy machine.

Current State	on input ‘0’		on input ‘1’	
	next state	output	next state	output
A	B	1	A	1
B	C	0	A	1
C	C	0	A	1

when we combine state and output in the above table we get the following states.
A1, B1, and C0.
These are the states of moore machine. We need to introduce initial state, assume that it is A0.
Therefore states of moore machine are A0, A1, B1, and C0.

Transition Table of equivalent moore machine is

state	on input symbol '0'	on input symbol '1'	output
A0(initial state)	B1	A1	0
A1	B1	A1	1
B1	C0	A1	1
C0	C0	A1	0

So, number of states in equivalent moore machine is 4.

Answer : 4

Q8:
From the state chart group the states based on output values.
G1 : For Output 0: {A,B,D,G}
G2 : For Output 1: {C,E,F,H}
For each state , check whether on every input value whether it goes to the state in same group or not . If on both the inputs, the resultant state is in same group then leave the current state in the group otherwise create a new group and place in the new group.
This classification is known as Equivalence classes.

G1: state A on 0 goes to C and on 1 goes to D where C , D are in different groups.
so place A in a new group.
G1 :{B,D,G}
G2 : {C,E,F,H}
G3 : {A}
G1: state B on 0 goes to H and on 1 goes to F Keep B as it is.
G1: state D on o goes to E and on 1 goes to A so as , D is similar to A because on same input the both states go to same group.
So, place D in group G3.
G1 :{B,G}
G2 : {C,E,F,H}
G3 : {A,D}

so, if you continue like this , only A and D will be in same equivalence class. so , A and D are equivalent.

A and D are equivalent.

Answer : B

Verilog Assignment 1: Half Adder

```
module halfadder(output carry, output sum, input a, input b);

assign sum=a^b;
assign carry=a&b;

endmodule
```

Notes: Notice that half-adder is a very simple design, there are two inputs and the equations to get sum and carry are easy to derive. The operator "^" stands for XOR and "&" stands for AND.

Verilog Assignment 2: Full Adder

```
module fulladder(output cout, output sum,  input a, input b, input cin);

assign sum=a^b^cin;
assign cout=(a&b)|(b&cin)|(cin&a);

endmodule
```

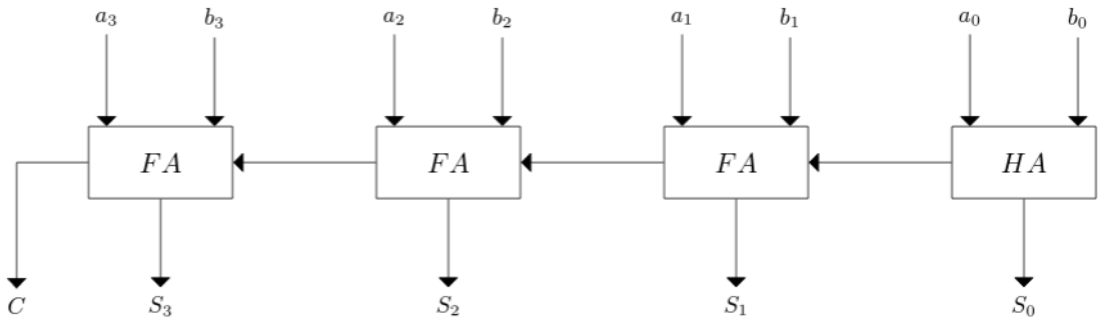
Notes: Notice that full-adder is also very easy to design.

Verilog Assignment 3: Four-bit Adder

```
module halfadder(output carry, output sum, input a, input b);
//fill in code for half adder here
   assign sum=a^b;
   assign carry=a&b;
endmodule


module fulladder(output cout, output sum,  input a, input b, input cin);
//fill in code for half adder here
   assign sum=a^b^cin;
   assign cout=(a&b)|(b&cin)|(cin&a);

endmodule


module four_bit_adder(output carry, output [3:0]sum, input [3:0]a, input [3:0]b);
  //fill in your code for 4-bit adder here
   wire [3:0]c;
   halfadder ha3(c[0], sum[0], a[0], b[0]);
   fulladder fa1(c[1], sum[1], a[1], b[1], c[0]);
   fulladder fa2(c[2], sum[2], a[2], b[2], c[1]);
   fulladder fa3(c[3], sum[3], a[3], b[3], c[2]);
   assign carry=c[3];
endmodule
```

Notes: In this particular circuit we give the LSB of the inputs to the Half Adder which computes the sum and carry. The sum is passed as the LSB of output and the carry is forwarded to the next Full Adder. This process is repeated for the remaining 3 pairs of input.

Some of you tried to use a full adder for adding a0 and b0 but got stuck in figuring out what the carry in bit must be. You could have added a 0 as carryin or alternatively just use the half adder like the picture below.


a_3
b_3
a_2
b_2
a_1
b_1
a_0
b_0
FA
FA
FA
HA
C
S_3
S_2
S_1
S_0