1. (**20 Points**) Consider a, b and c to be Boolean variables and $S(a,b,c)$ and $C(a,b,c)$ to be Boolean functions, such that $S(a,b,c) = a \oplus b \oplus c$ and $C(a,b,c) = ab + ac + bc$. Solve the following expressions and give the result in SOP form:

   (a) (**5 Points**) $\forall a : S(a,b,c) = 0$

   (b) (**5 Points**) $\exists a : S(a,b,c) = 1$

   (c) (**5 Points**) $\forall a : C(a,b,c) = bc$

   (d) (**5 Points**) $\exists a : C(a,b,c) = b + c$

2. (**25 Points**) A deterministic four-state machine produces the output sequence Z in response to the input sequence X, shown below.

   $X:$ 0 0 1 1 0 0 1 1 0 1 1 0 0 0 1

   $Z:$ 0 1 1 1 0 1 0 1 0 0 1 0 1 0 1

   Sketch the state machine diagram that produces the above sequence Z in response to the above input sequence X.

   Is this description unique?

   Can you sketch a deterministic state machine diagram with 3 states that complies with the above conditions? If yes, sketch the diagram. If no, why?

   We can first assume an FSM with 4 states say A, B, C and D [Refer Fig.1]. If this FSM starts with state A, for all possible combinations, we can have the following decision diagram on states. For example, from state A with input 0, we can go to states A, B, C and D. Since for the next input we should have a state transition (the output of the first input is different than the output of the second input), next state of A is not correct. Since this is our first decision and B, C and D are just symbols, we can follow one of the paths. From state B with input 0, we can go to states A, B, C or D. But again for the next bit we need a state transition. So we can go to C. We can continue in a similar way.

   Following all the possible paths in this decision diagram, we will reach more than one path that satisfies the given input/output sequence. Therefore, the result FSM (shown below in Fig. 1) is not unique.

   A 3-state machine cannot be made from this description as no two states in the 4-state machine are equivalent.

3. (**15 Points**) A deterministic four-state machine is in an unknown initial state. Show that the application of 55 consecutive 0's must leave the machine in the same state as the application of 7 consecutive 0's.

   Since this FSM has 4 states, for a number of consecutive symbols (in this example 0's) after at most 4 transitions we will meet a state that we had already met. So, for 4 or more number of consecutive 0's, we will have a cycle of length 1 (e.g. A-A-A-....) or 2 (e.g. A-B-C-B-C....) or 3 (e.g. A-B-C-A-B-C...) or 4 (e.g. A-B-C-D-A-B-C-...). In all these cases, since $7(mod2) = 55(mod2)$ *and* $7(mod3) = 55(mod3)$ *and* $7(mod4) = 55(mod4)$ *we can say that if this FSM has a loop of any length between 2*

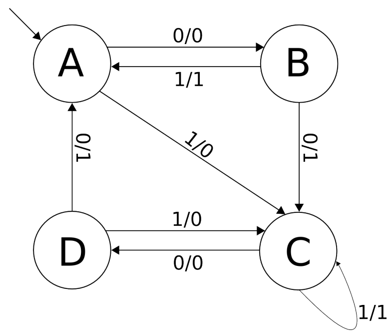

Figure 1: FSM for Q2

and 4, it goes to the same state. If this FSM has a self loop it will go to the same state once it goes to that loop (e.g. A-B-C-C-C-C....). Therefore, if will be in the same state after 7 or after 55 transitions.

4. (**20 Points**) Find a regular expression on the alphabet {0, 1, 2} for the set of strings recognized by the graph shown in Fig. 2.

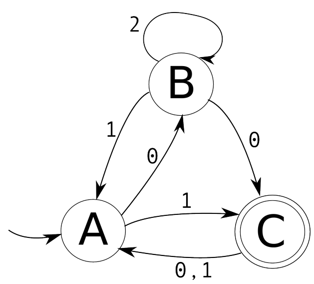

Figure 2: Graph for Problem 4

*(02*1+02*0(0+1)+1(0+1))*(1+02*0)*

5. (**30 Points**) Which of the following sets can be recognized by finite-state machines? Justify your answer. In case that the set can be recognized by finite-state machines, show the state diagram (or part of it in case of large state machines). In each case the alphabet is {0, 1}.

   (a) (**5 Points**) The set consisting of those strings that contain, for all k (k=0, 1, ...), k 1's and k+1 0's in any order.
   No. This would need infinite number of states to recognize the set of strings.

   (b) (**5 Points**) The set of strings in which the magnitude of the difference between the number of 0's and the number of 1's is a multiple of five.
   Yes.
   There are a finite number of states (5 states) : ||No. of 1's - No. of 0's|| mod 5.
   A state machine can be made through transitions between these 5 states.

(c) (**5 Points**) The set of strings in which every 0 is immediately preceded by at least k 1's and is immediately followed by exactly k 1's, where k is a specified positive integer.
Yes.
k is a specified number. Therefore a machine representing $1^* + 1^k1^*(01^k)^*$ will recognize these strings.

(d) (**5 Points**) The set of strings that contain more 1's than 0's.
No. This would need infinite number of states to recognize the set of strings.

(e) (**5 Points**) The set of strings in which the number of groups of consecutive 1's equals the number of groups of consecutive 0's.
Yes.
You can make a state machine by counting the transitions in the input sequence($0 \rightarrow 1$ *and* $1 \rightarrow 0$*). The difference between these two transitions will never be greater than 1, therefore you can make a state machine for this set.*

(f) (**5 Points**) The set of strings in which every possible sub-sequence of length seven appears at least once.
Yes.
There are 128 subsequences of length seven. Let's denote them by $s_1, s_2 ... s_{128}$. *State machine* M_i *can be made to check the appearance of* s_i *as* $(0+1)^*s_i(0+1)^*$. *Taking a product machine of all* M_i *will return a machine which accepts the given set.*

6. (**15 Points**) In which of the following cases do the two expressions describe the same set?

(a) (**5 Points**) $(011^* + 01^*0)^*$ *and* $(00)^*01(0+1)^* + \lambda$
No. Counter example: 0000

(b) (**5 Points**) $(1^*0 + 001)^*01$ *and* $(1^*001 + 00101)^*$
No. Counter example: λ

(c) (**5 Points**) $0^*1(0+10^*1)^*$ *and* $(1+00^*1) + (1+00^*1)(0+10^*1)^*(0+10^*1)$
Yes.

7. (**10 Points**) For each of the following expressions, find a transition graph that recognizes the corresponding set of strings.

   (a) (**5 Points**) $(0+1)(11+0^*)^*(0+1)$

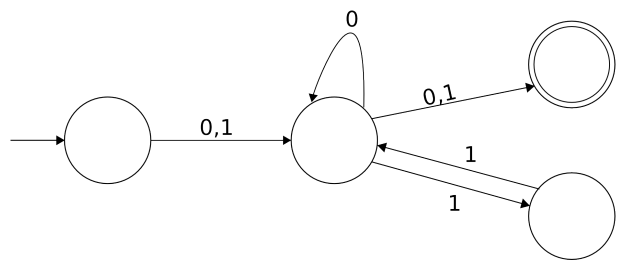

Figure 3: Q6A

   (b) (**5 Points**) $(1010^* + 1(101)^*0)^*1$

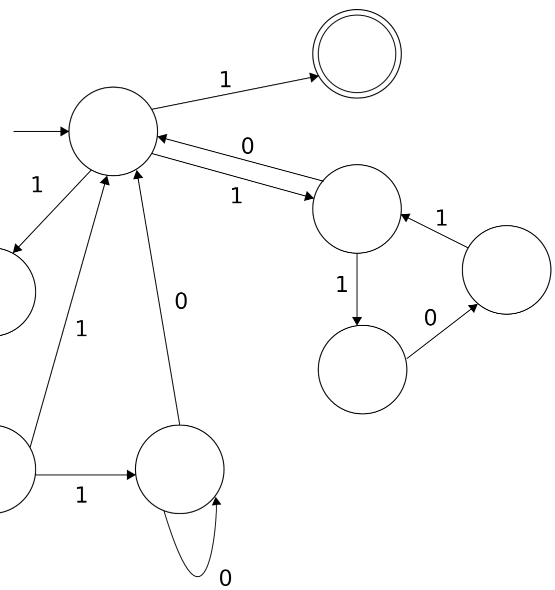

Figure 4: Q6B

8. (**15 Points**) Answer the questions based on the following module definition
   (a) (**5 Points**) The above arbiter design has a display statement to check if the grants are mutually exclusive. Why is this a bad idea?
   This is not a scalable technique if the design of the aribiter is modified. Also, $display statements fall in the active region of the simulation. This means that the update on the grants may or may not get reflected in a deterministic order. (Look up event regions in Verilog LRM)

(b) (**10 Points**) Correct the mistake in the following assertion so that it asserts when the grants are not mutually exclusive.
`assert property (@(posedge clk) disable iff (rst_n)  (grant0 + grant1));`
assert property (@(posedge clk) disable iff (!rst_n) !(grant0 && grant1));

9. (**5 Points**) Holds in all states except the last one.

10. (**5 Points**) Lot of acceptable answers. $posedge returns an event whereas $rose return a boolean. posedge cannot be used in expressions whereas rose can be used in expressions. posedge is checked instantly and returns no value whereas rose is sampled and returns a 1 or 0.

11. (**20 Points**)
(a) (**10 Points**) If a target is selected, then it should assert the signal ẗrdyäfter 2 clock cycles.
assert property(@posedge(clk) disable iff (reset) $rose(sel_bit) |-> ##1 trdy ##1 !trdy);
(b) (**10 Points**) At the end of a transaction, the ësel bitsïgnal is deasserted. One clock cycle after that, the signal ẗrdyšhould be de-asserted.
assert property(@posedge(clk) disable iff (reset) $fell(sel_bit) |-> ##1 trdy);