Digital VLSI System Design

Prof. S. Srinivasan

Department of Electrical Engineering

Indian Institute of Technology, Madras

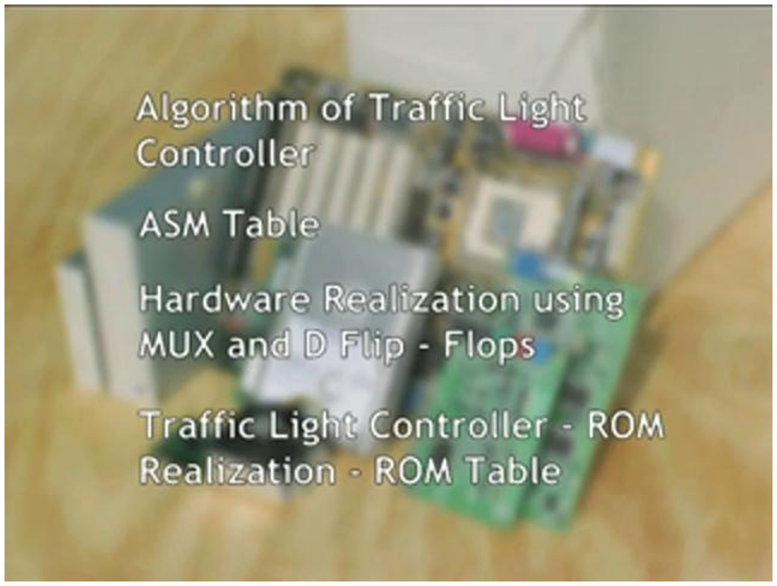
Lecture - 20

Example of System Design using Sequential Circuits

(Continued)

Slide – Summary of contents covered in previous lecture.

(Refer Slide Time: 01:40)



Algorithm of Traffic Light Controller

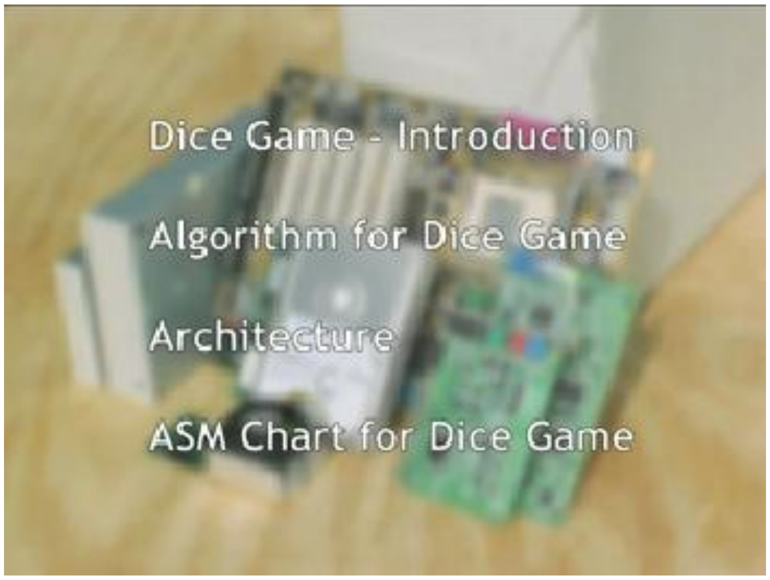
ASM Table

Hardware Realization using MUX and D Flip - Flops

Traffic Light Controller - ROM Realization - ROM Table

Slide – Summary of contents covered in this lecture.

(Refer Slide Time: 01:40)



Dice Game - Introduction

Algorithm for Dice Game

Architecture

ASM Chart for Dice Game

As the third example of the ASM based design, we will take up a game. Dice games are popular both in eastern and western countries. When we go to gambling parlors in western countries like other states we have this casino, where they have this gambling system, by means of some handles and buttons to stimulate a dice throw, based on the dice value some activity takes place and further progress of the game is decided.

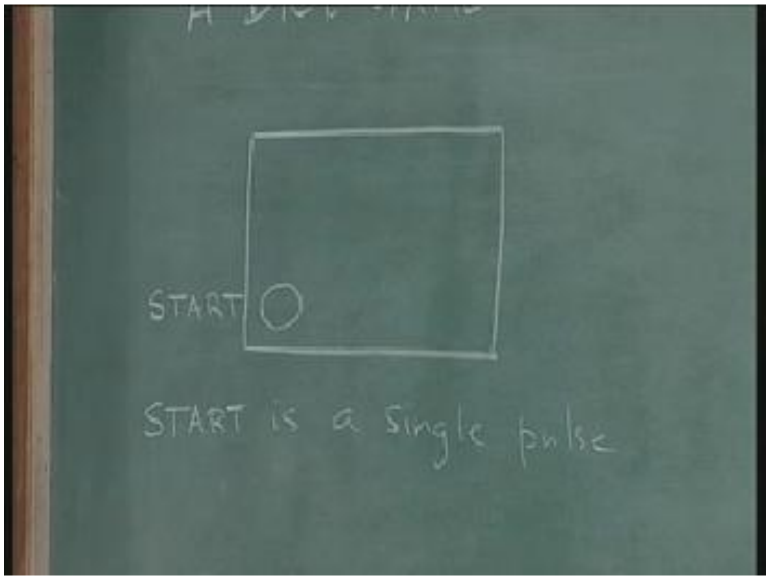
Dice games have been very popular in India. We have lots of variations of dice games popular in rural India, villages. Even our sculptures talk of dice game and consequence of dice games. So, I thought it appropriate to use a dice game example, as the purpose is to tell you almost anything you can conceive of can be implemented in digital system and subsequently as a verilog code and the FPGA implementation. But, I do not want to adapt. When you open a textbook from Britain or the United States of America, Mostly in United States you will find a dice game example invariably which will be dice throw usually based on the card game. But I want to sort of use the same concept and give an example from our tradition of culture were in we have this dice games.

For example, one of the most popular dice games in India has been Snakes and Ladders, in which a player throws a die or a pair of dice and then advances in a chart marked with various symbols or pictures. Whenever he or she comes to a point represented by a ladder, he or she is asked to advance to a certain level and whenever he or she comes to a point represented by a snake, he or she comes down by certain squares or numbers.

We will try to stimulate it in the best possible way. This dice game will have the following features: You 2 two people to play simultaneously in your dice game simulation to find out who wins or who reaches the target first. We will also do it, but then only 1 player can play at a time, so what we do is we have a pair of dice, each with six faces, so that you have a total of six plus six equal to twelve whenever you throw a pair of dice. The score can be any number from 2 to 12 and based on this number we advance the score. Start with a 0, 0 score equal to the first starting square of the snake and ladder game and then advance by as many squares as the value of the dice.

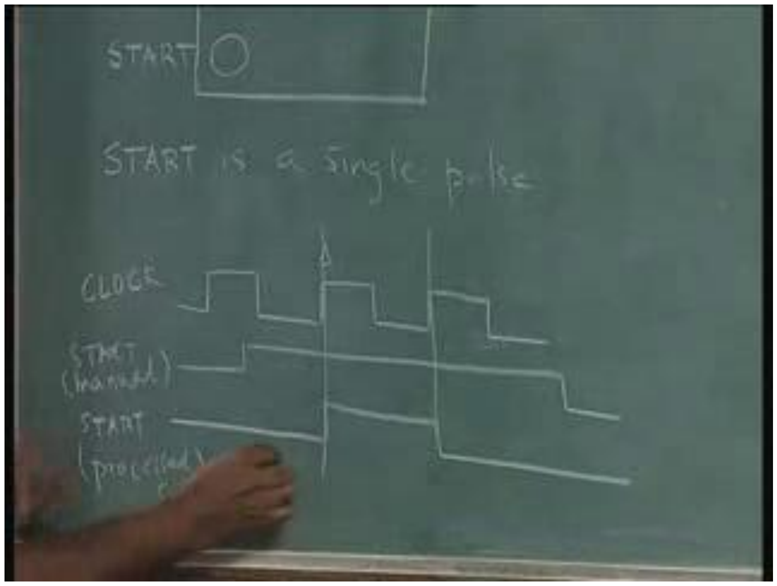
To simulate this Snakes and Ladders game, what I will do is that, if the score is less than a particular value, we will reduce a certain amount from the total score, which means that you slide back. If the dice throw value is more than a certain value, we will add a certain amount to the total score, which means that you are allowed to advance in the chart. In this way, we will simulate our Snakes and Ladders conditions. Again, we will keep count of the number of attempts you make to reach the final destination and then the game is over as far as the first player is concerned. The second player then takes over. At the end, the two players can compare notes and find out which player has reached the target or the goal with fewer steps. That player is declared the winner. This is the game we are playing.

(Refer Slide Time: 06:50)



So the game will have this type of front face, in which there will be a switch or a button called start. Normally, when the game is not being played, nothing happens. If somebody wants to start a game, he or she has to press the button once and create a single pulse. Normally, when you manually press a button in a digital system which work with clocks of, high speed, even if the speed is not very high, we will hold it for a while. Even if you hold it for just a few milliseconds, it will generate many clock cycles. So the start pulse will last for several clock cycles of the system. Hence, we need to have a special circuit, which I am not showing here, by which the pressing of the button, even if held for a long time, will only give a pulse corresponding to the clock period.

(Refer Slide Time: 08:07)

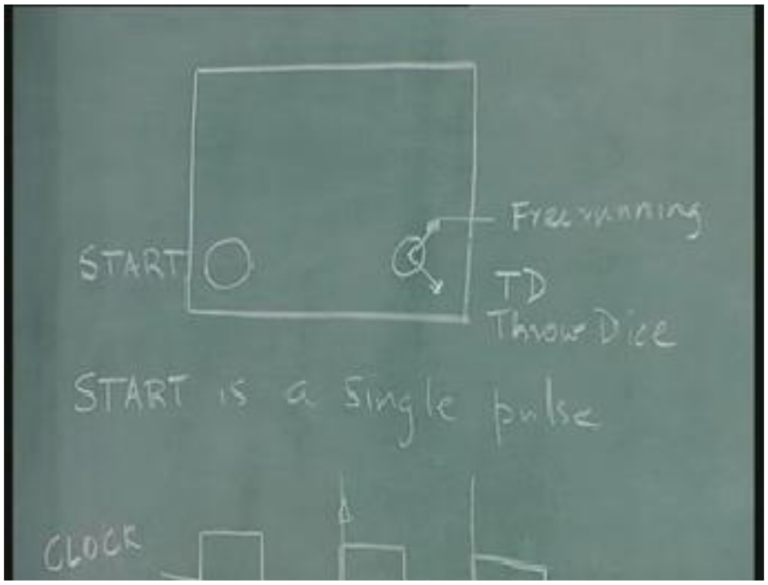


This is the clock on the system. Once you press Start and elevate asynchronously and keep on holding for something, what you get will be from the system even though you hold it like this. I want a start pulse, which will start along a clock edge and last till the next clock edge start at the clock edge (Refer Slide Time: 08:53). This is my start signals so there are 2 start signals. This is the manual start and this is the processed start or a single pulse. I am assuming an electronic circuit that does this job, which is not part of this, design were contemplating so this is what I am getting when I press Start.

Then, how do you simulate to throw a pair of dice? We can have a counter running and when we stop the counter, whatever is the value of the counter can be taken as the value of the dice. You need two dice each going from 1 to 6 so the total can be up to 12. So what we can do is, we can simulate a single counter which repeatedly starts counting from 2 and goes to 12 and at the end of 12, it starts again with 2 and goes back to 12. Any time I stop the counter you will have any value from 2 to 12 and that is taken as the value of the two dice put together. Even though it is a single die and a single clock we will assume this to be the combination sum of the two dice.

So how do you know that? This means that there is a clock, which is running all the time at whatever frequency we want to run it. Whenever I want to stop the clock, the value of the counter at that time is taken as the dice value.

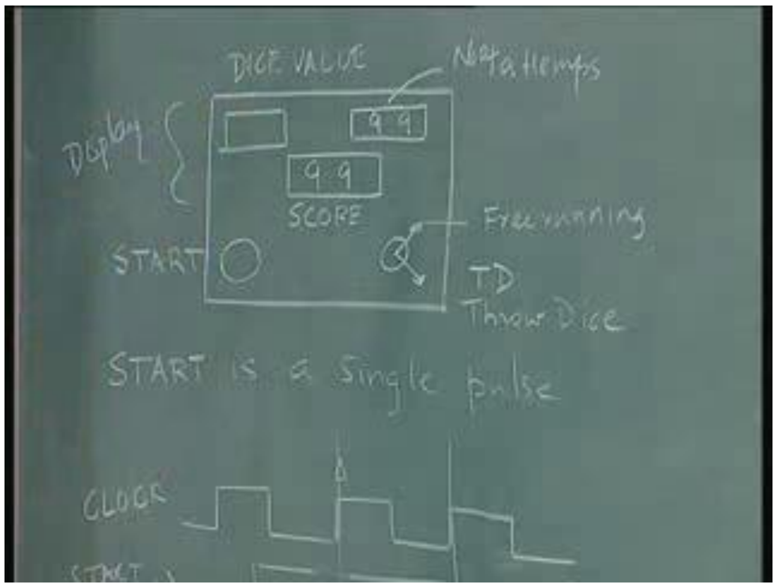
(Refer Slide Time: 10:15)



So there will be a switch here. I will call this TD or Throw Dice, as throwing the dice is simulated by stopping the clock. So, normally, it runs in a free running mode and the counter keeps running. There is a switch (Refer Slide Time: 10:41), which is in this position and once you press it down to this position, the clock stops. As long as you hold it in this position, the value of the dice is given in the counter. The counter holds this value as long as it is. Again, when you are done with your processing, find out the value and add it to the score. Subtract and add all that you have done and you are ready for the next game. When you put this back into the free running mode, the clock starts running and then again it throws in.

So the dice throw is simulated like this. The clock is running normally. The dice throw switch is in this position. We will call this the 0 position of the dice throw and when the dice throw position is 1, the clock starts and freezes, so that it will be held as long as we want it to be held in that position. You take that number, add it or subtract it, process whatever you want. Once you are done with this, put it back in this and the clock starts running. Give it some time so that it can become random, generates again and comes back then. That is all it is. The manual interface between the game and the electronic is only these 2 switches, one switch that starts the game with a single pulse and the other switch when thrown, it will fall from free running position to dice throw position. That is all. Then, what are the 3 things you wanted to display?

(Refer Slide Time: 12:12)



We have 3 displays: one is the total score. I want the total score to be 100. We will think of a game with 100 squares. Instead of starting from 1 to 100, we will start from 0 to 99. In this way, I can use a two digit decimal counter so this score can have a maximum value of 99 and a minimum of 0, of course.

Then, I should know who wins by based on the number of attempts you make to reach the 99. So, you need one more counter called the number of attempts counter. Theoretically, the number of attempts can be 99 because it cannot be 99 theoretically the number of attempts can be because every time you add, you need to have a 2 but then you can also go on sometimes you subtract 10. So, we will also have a 2 digit counter here with a maximum value that assumes that we will not require more than 99 subsidies. The minimum value of the dice is 2, considering which you need only 50 steps to go to 100. However, in between, there may be several minus 10s so you need to increase the number of attempts. I am assuming that you will not always put minus 10, minus 10 that goes on forever as scold as in move. I am not forcing that condition, so we will restrict the number of attempts counted to 99.

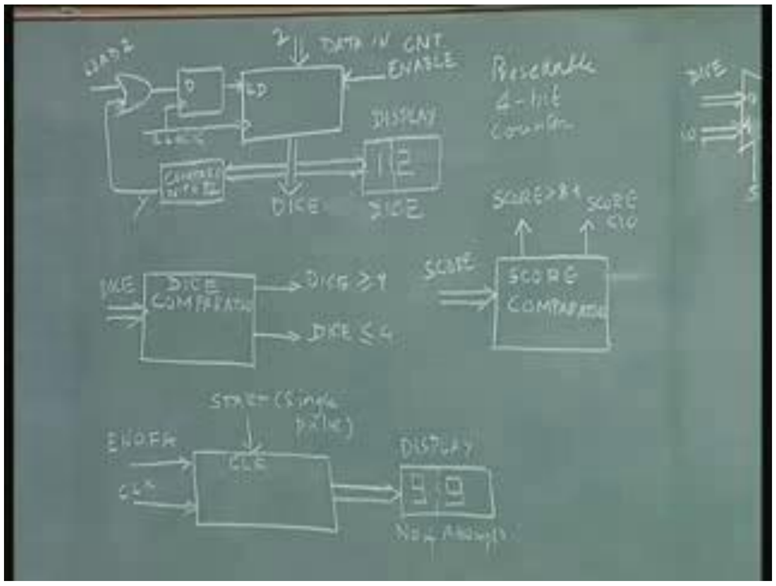
You need one more counter to show you where you stopped the counter, where the counter flows every time you throw the dice, because you do not know that. So I need to know that score every time I throw the dice. I would like to know how much it is. That will give me a lot of happiness, so I need to know that.

So this is my dice value at th3ree displays. These 3 are displays (Refer Slide Time: 14:12). One display will show the value of the dice on every throw, one display will show the total score up to that point and one display will show you how many attempts you have taken to reach up to this.

There are 2 switches: one to start as a single pulse and the second switch to throw from the clock running mode to the throw dice mode. Simple!

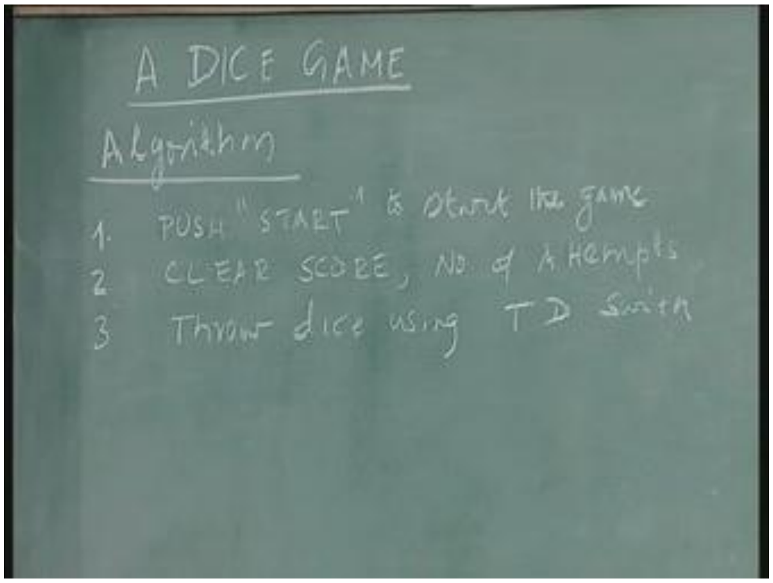
I will explain the game once more in terms of an algorithm. This is the requirement because you have to require what is the procedure. We understand the problem specification completely, then look at the functional blocks required, identify the signals required for the functional blocks, then go for the ASM and then go for the implementation. So, before we go to the functional identification of the system or the functional parts required for the game to be designed and fabricated for implementation, I will have to specify the algorithm. To explain to you qualitatively, well, let us put it in terms of steps.

(Refer Slide Time: 15:26)



So this is called the algorithm of the game.

(Refer Slide Time: 15:30)



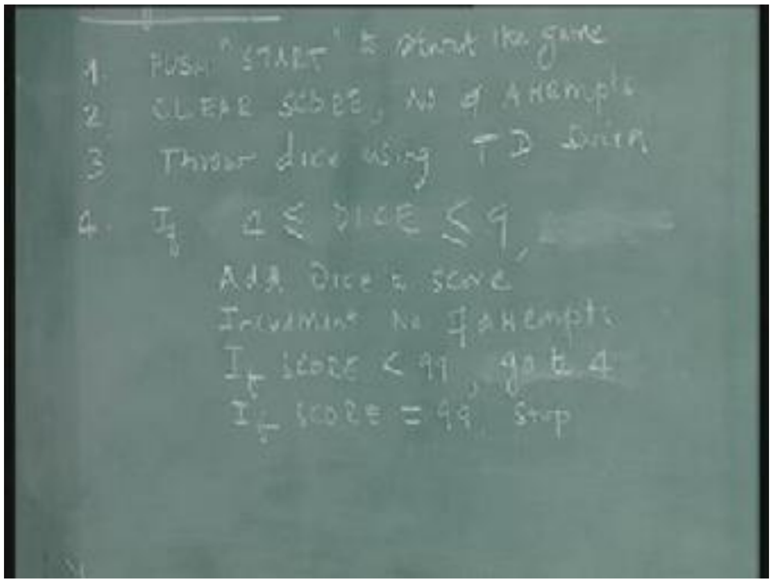
A DICE GAME

Algorithm

1. PUSH "START" to start the game
2. CLEAR SCORE, No. of Attempts
3. Throw dice using TD Switch

What is the algorithm? It is the step by step procedure of the game. Push start to start the game. Once you push start, the counter will start free running. This means that the dice throw can be done any time. Throw dice using the switch I explained earlier. How did I do that? Using the TD switch that is taking the TD switch from the free running position to the dice throw position, you now throw the dice, which means the dice is cast. Okay, now the game is on. Of course, first you are assuming before this after starting clear and then add one step here. Both score and number of attempts have to be cleared because I do not want to start with 0 attempts, 0 score. Then I throw the dice using the TD switch.

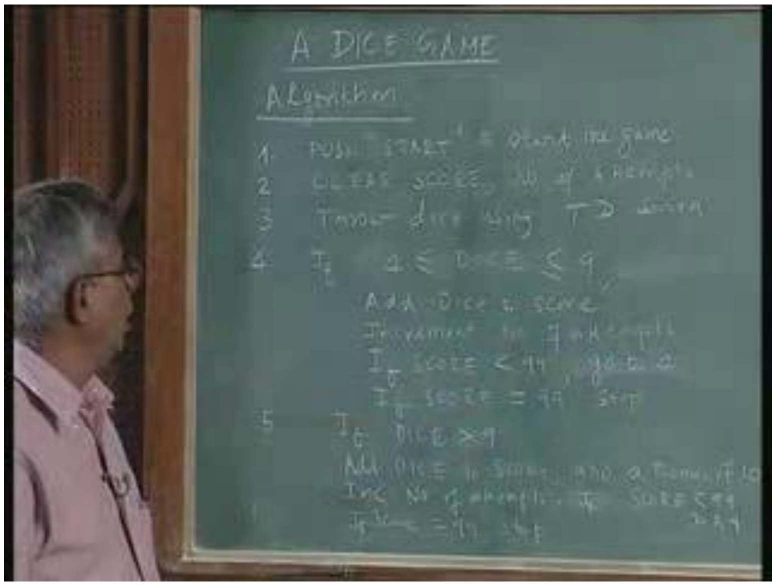
(Refer Slide Time: 17:38)



```
1. PUSH "START" to start the game
2. CLEAR SCORE, No of Attempts
3. Throw dice using TD Driver
4. If 4 ≤ DICE ≤ 9,
       Add Dice to score
       Increment No of attempts
       If score < 99, go to 4
       If score = 99 stop
```

If the value of the dice is, I will have to arbitrarily fix a value, less than 4, I want to give a disincentive or a penalty and if it is more than 9, I want to give an incentive or a bonus and in between I will not do anything else, I will simply add. So I will just put in algorithmically; if the dice is greater than or equal to 4 and less than or equal to 9, add the dice value to the score, increment number of attempts. If the score is less than 99, continue; that is, go to 4 and continue the game. If the score equals 99, stop.

(Refer Slide Time: 19:07)



```
A DICE GAME
Algorithm
1. PUSH "START" to start the game
2. CLEAR SCORE, No of Attempts
3. Throw dice using TD Driver
4. If 4 ≤ DICE ≤ 9,
       Add Dice to score
       Increment No of attempts
       If score < 99, go to 4
       If score = 99 stop
5. If DICE > 9
       Add DICE to Score, and a bonus of 10
       Inc No of attempts, If SCORE < 99
                                    go to 4
       If Score = 99, stop
```

If the dice value is greater than 9, (less than or equal to means this 9 is also taken here) that is. it can be 10, 11 or 12, then I want to give a bonus after adding the score. Add dice score. Add a bonus of 10 and again increment the number of attempts. Check score. If score is less than 99, go to 4. If the score is 99, stop. I am explaining 3 different cases, that is all. This is not necessarily the way in which we implement ASM chart. ASM chart will make sure all these conditions are properly met. Then there is the next possibility where the dice is less than 4. If the dice is less than 4, add its value to the score and subtract 10 from the score as penalty. Again, increment the number of attempts. If the score is less than 99, continue. Or if the score is 99, stop. These are 3 cases that I have explained.

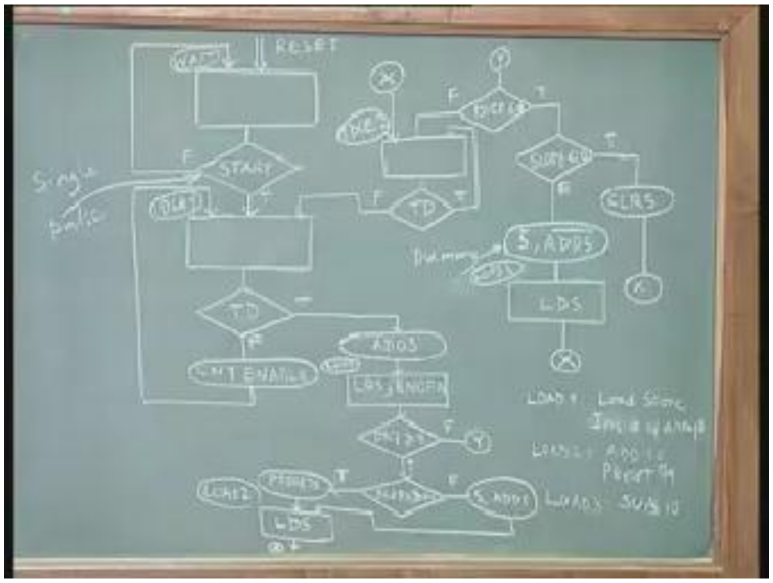
When you first throw the dice, the number of attempts is 0, the dice value is 0, throws or number of attempts is 0, total score is 0. I throw the dice. The dice can be any number from 2 to 12. If the dice is between 4 and 9 that is. 4, 5, 6, 7, 8 or 9, at these 6 values, the dice value is added to the score as it is until we reach 99 and each time we register the number of attempts that is. Increase the number of attempts by 1. At the end, I will say the score is 99; number of attempts is whatever you have made.

One thing I have not clearly explained here is that there may be a case wherein before reaching 99, if you add the dice value to the score, the score may exceed 99. Example, supposing my score is 92 and I got 8 in my dice. 92 plus 8 is equal 100. If I typed add 8 to 92 in a 2 digit counter the result will be 1 0 0, 1 will be discarded and 0 0 will start. I do not want that to happen. Whenever the score exceeds 99, we will retain it as 99 rather than rounding it off. This is something that I have not put in the steps. What I am saying is that if the score is between 4 and 9, the score is added, number of attempts is increased and this goes on, assuming that at no time you get less than 4 or greater than 9 that is. You always get between 4 and 9. There is no bonus or penalty. Keep adding until you reach 99, but in that enthusiasm if the last attempt score reaches beyond 99, you should cap it to 99. That is the normal algorithm.

Now there are 2 conditions, one is a bonus condition and the other is a penalty condition. If at any time, the dice throw is more than 9 that is. 10, 11 or 12, in addition to adding the dice value to the previous score, you want to add 10 to the total. First I add the dice value and then I add 10.

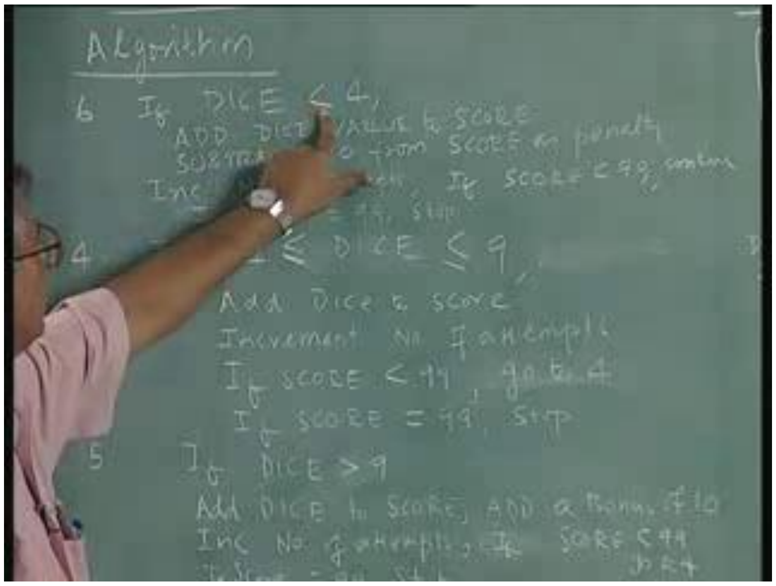
Again, I will use the same system. If the score reaches 99, I will stop but if the score exceeds 99, I will limit to 99.

(Refer Slide Time: 25:07)



Likewise in the penalty case if the dice value is less than 4 that is. 1, 2, 3 and 4 are in the penalty class.

(Refer Slide Time: 25:12)



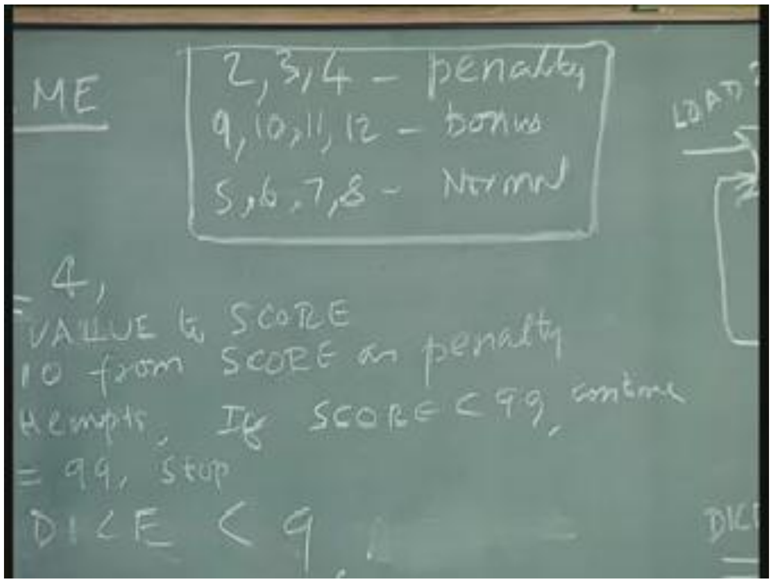
Algorithm

```
6   If DICE ≤ 4,
      ADD DICE VALUE to SCORE
      SUBTRA[illegible] 0 from SCORE as penalty
    Inc [illegible] If SCORE < 99, continue
      [illegible] 99, Stop
4   [illegible] ≤ DICE ≤ 9,
      Add Dice to score
      Increment No of attempts
      If SCORE < 99, go to 4
       If SCORE = 99, Stop
5   If DICE > 9
      Add DICE to SCORE, ADD a Bonus of 10
      Inc No of attempts, If SCORE < 99
                                   go to 4
```

When the dice value is less than 4, it means 2 or 3, I first add the dice value and then subtract 10 and then if the score becomes negative due to the subtraction, I will make it 0 0. Just as I added 10 and topped it off to 99 in the earlier case, here, if the score becomes less than 0 when I subtract 10, I will make it 0. So these two limits of 0, 0 and 99 will not be exceeded, this is one of the conditions of the game I had given but had not mentioned it explicitly.

Now, my students here say that the original specification in the problem was 4 less than dice, less than and that means dice value 2, 3, 4 is penalty dice value, 9, 10, 11 and 12 is the bonus; that means in between 5, 6, 7 and 8, that is four values, you can do it anyway you like. Since we have drawn the ASM and the circuit diagram using this, I am going to change this condition without any confusion with the loss of generality as the design. Without any loss of generality I am now making a difference if dice value falls between 5 and 8, greater than 4 and less than 4 and greater than 9 less than 4 and greater than 9 normal conditions. If dice value is equal to or greater than 9, that is. 9, 10, 11 or 12, in these 4 cases it will be a bonus case.

(Refer Slide Time: 26:46)



ME

2,3,4 - penalty
9,10,11,12 - bonus
5,6,7,8 - Normal

LOAD

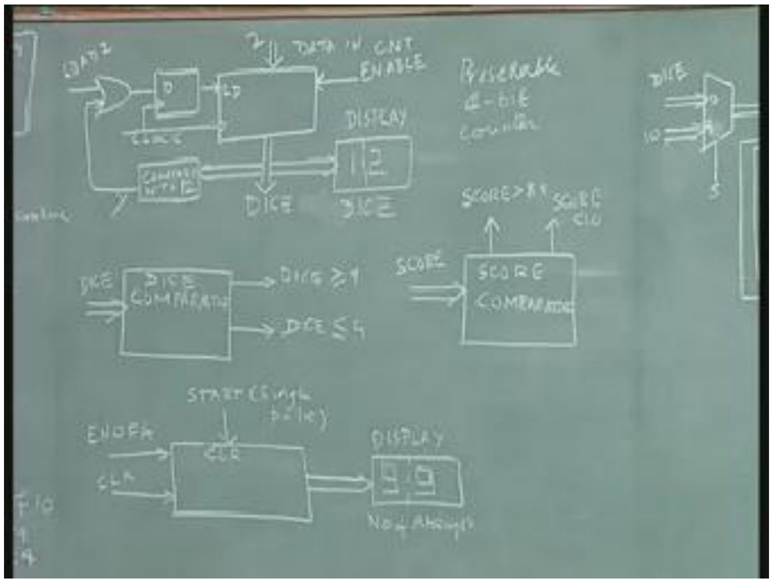
4,
VALUE to SCORE
10 from SCORE as penalty
ttempts, If SCORE < 99, continue
= 99, stop
DICE < 9

DIC

If the dice value is less than 4, that is.2, 3, 4, these cases are penalty; 9, 10, 11, 12 – bonus; 5, 6, 7, 8 – normal. This is a revision I am making in order to satisfy the design we have already planned for this lecture.

It has not changed the rule; it has not changed the approach, or changed the algorithm. It has only changed the values. To reiterate, add 10 as the bonus to the value in the range given, subtract 10 in the range given and otherwise add normally. Clearly remember that you should never allow this score to go beyond 99 and round it off or wrap it up. When addition of the existing score to the new value of the dice or addition of 10 to the existing score results in a number greater than 99, you make the score 99 and stop the game. Otherwise, the person will be at a disadvantage. Likewise whenever you subtract 10 as a penalty, if the result happens to be negative, make it 0, 0. Otherwise, your hardware design will be difficult. Nobody understands a negative value in the game. These are all computational concepts. In a game, you should not say you have negative value; hence we have put this limit that the score never falls below 0, 0 even if the penalty is 10. Similarly, the score never increases beyond 99 with the addition of dice value to the score or addition of 10 as a bonus. This is one case which has to be clearly understood, which is not mentioned in the original diagram. We will now go to the hardware elements.

(Refer Slide Time: 29:12)

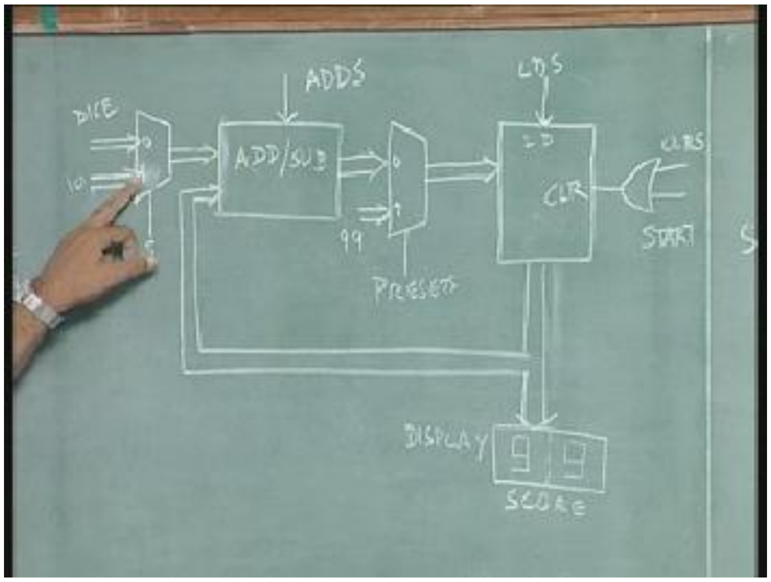


We have this counter, which is the dice simulation. Increment the number of attempts, that counter we are incrementing. Would not it be nice if, no sooner we throw the dice, it increments the number of attempts counter. That is what we are doing anyway. This is one set of specifications. There is a scope of refinement for any desired problem. He wants to know whether we can do incrementing in some other way. First of all, we have not seen how we are doing incrementing. Even assuming we have a different way of doing it, try it. Right from the beginning, I have said that for ASM design, understand the specifications completely and design the way it works. I will be only too happy if your design is more efficient than my design. Specifications should be met unambiguously, which means that you have to ask for specifications unambiguously, specifications should be given unambiguously and should be met totally. Within that constraint, a more efficient design will naturally deserve a prize.

Now, this is the dice simulation. We start from 2, that is, why it is a preset able counter, which means we can preset any value you want. The clock keeps loading this, incrementing it normally, its running free-running mode and the dice value display comes here and when the value of the dice comes, this is enabled, which makes the dice, that is in free-running mode. If you do not put this dice throw mode, it is frozen dice throw mode, the enable is removed so, the dice value will be here as well as here; at that time the value will be compared with 12 and if it is 12, load the value 2 so that, next time again I can start with 2 or more than 2. That is, if compared with two 12 you do not want to reach that when the counter has to reach 12 you compare it and make it 2 again so that instead of going to 13 after 12 it goes to 2. One more load here, this load is for starting when ever beginning, when you are not playing the game, as the power on reset bring it to this load 2 that is the value of load 2.

We have to load value of 2 to the counter in the beginning. This is the dice comparator. Dice value is compared to see whether it is less than or equal to 4 or greater than or equal to 9. These two signals will be generated. This is the increment counter, the counter which increments the number of attempts. Enable number of attempts here (Refer Slide Time: 32:13). So here you start this counter they clear, when you start the signal, this is clear and this is specifically given every time you want to enable the counter by one increment the counter by one, enable it for that one clock period only it will be enabled so that one will be added you know the value of this. This is the score comparator. Whenever there is a score, you want to know whether the score is greater than 89 or less than y. If the score is greater than 89, I cannot add 10 to it. When I add 10 it should become 99. This is why you should know when the score is greater than 89. You should also know if the score is less than 10 and treat it as a special case, since if the score is less than 10, I cannot subtract 10, I can only make it 0 0.

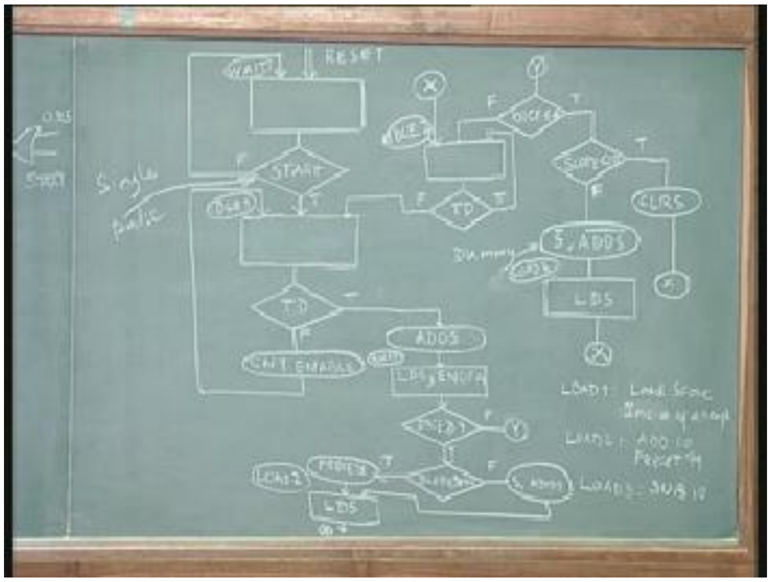
(Refer Slide Time: 33:02)



This is add sub circuit. The dice value or 10, normally dice value, is added and it is stored in this register. This is the score register score display you clear to load this value of this from here value of this. Now whenever you want to add 10 to it or subtract 10 from it, you select this. There is a selector here. Normally, the selector is 0 so the dice value gets added, but whenever you want to select a 10 to be added or subtracted, you select this and this gets 10 and that 10 can be added or subtracted. If you add s as 1, it will add 10 values. If you add s as 0, it will subtract 10 values and whether this value or 99 is on the score counter depends on where we are. If the score is greater than 99, as we see here the score is greater than 89, adding 10 should not lead to more than 99. In that case, this will disable this. This is preset to 99 values. And this is the score display starting; also this is clear to the starting signal.

There is one specific place where you have to clear this when the result becomes less than 0. I want to start with 0, 0. Score has to be cleared to 0, 0. This is the clearing on subtraction. On subtraction, if the result is less than 0, we make this 0. The clearing will happen in the beginning of the game as well as whenever the score becomes less than 0 on subtraction. These are the various units. Now the ASM is waiting for the start signal to go and start signal to come.

(Refer Slide Time: 35:08)



When the start is not there it is waiting normally, in the power on we will have to make sure the power on condition will leave you to this state that is why preset state is called and once you start pulse you have to make sure it is a single pulse. I already told you that is an extra hardware you have to put outside. The start pulse should not be directly used from the push button. The push button has to be denounced and given a single pulse; that is true. Again, you wait here because it is not enough if you start the game, you also have to throw the dice. So when you start and when you are ready to start waiting for the throw dice, you have to load the first value of 2 and after that every clock cycle it will go in throw dice. Actually load 2 should not be done here because every time you throw dice it will remain in 2 which should be above this. Here I can put this.

When waiting, I will load 2 and wait because when the counter starts it will start from two and then idle throw dice is not true. We are going to freeze it into this design. There may be some flaws. We will correct it in the next version. What I mean is that on reset condition, it gets loaded into 2. So the game starts again, there is nothing wrong in that, it will always start with 2 that is what we want. That is idle throw dice is not there count keeps enabled this is what it is.

The counter is enabled. This means it is still counting from 2, 3, 4, 5, 6 and 7 up to 12 and back to 2. This is the loop. But the moment the dice throw is frozen dice throw is available throw dice is available count is frozen enable is not there anymore it is frozen to add the score add score is adding the score that is here. This value will be selected dice value because I am assuming that to be zero and then we also load score after adding in adder; adder only adds it does not load it. So I need to give a load signal specifically and also enable so this ENOFA enables increment the number of attempts.

So I load the score and increment the number and then after adding the score, I verify whether it is 9, less than 9, more than 9, less than 4 and all that. So I verify this as greater than or equal to 9. If it is not greater than or equal to 9, I will have to see whether it is less than or equal to 4, which we will see later. If it is greater than or equal to 9, I have to add 10, but before adding 10, if the score is already 89, I do not want to add 10. I have to just add enough to make it 99. If the score is greater than 89, then I give this preset input, which will select 99 by counter. If it is not so, I will give this as 1 and that will add another 10 to this and this will give the addition also so 10 will be selected and it will be added and whatever is selected it has to be again added. This is called load 1. There is a load 1 where I load this score and increment the attempts and load 2 where I load add 10 or preset to 99, depending on the load being less than or equal to 89. On the other hand, if the dice throw is less than 9 go to this y link and then I see this dice throw is less than 4.

If the dice throw is not less than 4, you are in the safe range. You have already added the score. We have to go. Dice throw is still on. This is where we have to be careful. I keep it down when I throw the dice. If I kept it down, it is equal to only 1 dice throw. We have to return it to free running mode before I can throw it again. So if I have forgotten to do it, I will continue to wait for it so dice throw is still on. This means that you have not put it back to free running mode. It will keep waiting idle 2. On the other hand, if you put it back in the free running mode, it will go and do this, again till you do this dice throw again.

The score is less than 4. I get 4 conditions. I have to subtract 10 but I can subtract 10 only if this score is less than 10. If the score is less than 10, we will have to clear it to make it 0. This is called clear s that is. the second clear. We talked about two clearings; starting clear in the score and clearing on subtraction. On the other hand, if the score is not less than 10 that is. it is equal to 10 or more than 10, I will simply have to load the value because the subtraction of 10 comes automatically. I put this as dummy because to make it clear I have to put this here, s-bar and add s-bar that means this s will be selected as s-bar, which means ten will be selected, this is not s-bar this will be s. We have to issue s and add or subtract is automatically done, because if it is not add it is subtract. This is the one signal add subtract.

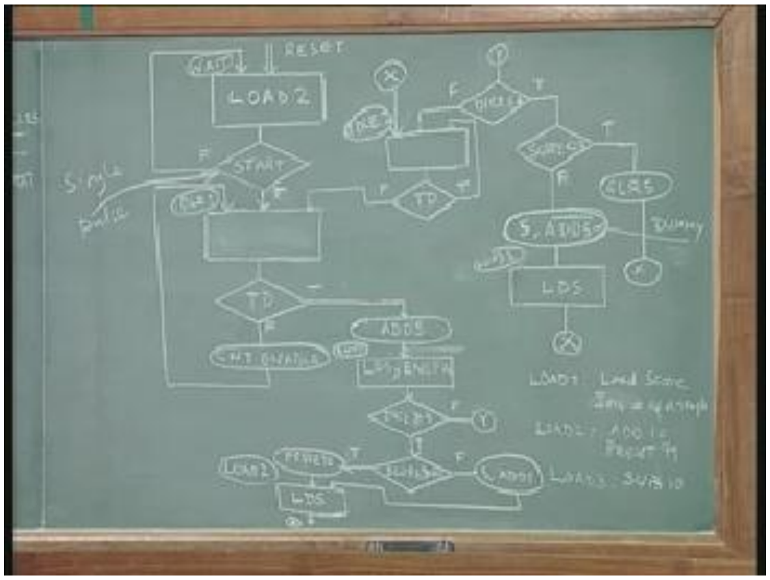
In the previous mode, we added this. In this mode, we subtracted it. This conditional output s has been chosen but add signal is not required. I put it in color and call it a dummy so that you should know it has to be subtracted. There is no need to give a signal whenever add is not 0, add s is not one it is zero so it is called add s-bar, add s-bar need not be generated when add s bar, add s is not there, it will be 0.

But in order to explain that I put it as an extra signal, signal is not really required, s is to be generated and then you load the counter again, the score and then again the new value comes after that. All these conditions you go back and wait here in the idle mode till the throw dice. So you do the processing either direct adding, adding and again adding ten, adding and capping of ninety nine, subtracting, adding and subtracting, or adding and clearing it to 0, all these five possibilities. Finally, you have to go back and wait for the load s to be removed.

So now we have the conditions. For the values between 5 and 8, that is. 5, 6, 7 and 8 the score is added. Here, the condition is not necessary. Now, the question will arise whether the score can become more than 89. This problem of 99 comes only if the score is more than 99. If the score is anywhere in between 5 to 8 that is 5, 6, 7 or 8, we simply add the value. If the score is between 9 to 12 that is 9, 10, 11 or 12, we add the score value and add 10 as well. In this process, you may exceed 99 when you are trying to add 10, so we keep it up to 99. Similarly, when the dice value is less than 4 that is 4, 3 or 2, we add the dice value and then you subtract 10. When you subtract 10, it may come to 0, 0 or it may become negative, in which case we cap it to 0, 0.

There is one case we have not taken into account. Actually there is a flaw in the design that just occurred to me as I am teaching this. Is it possible that you are in the 90 range and without adding 10 you may still reach 99? Suppose I have got 97 score and I got a 6, the score will exceed 99. Hence, I need to put one more condition here. Whether the adding score dice value is 9 or not, if the value becomes 99, actually make it 99. So I have to make a condition here. I will leave this condition as an exercise for you; I do not even want to show where it is. This suffers from one design flaw, namely, we check if the score is 89 only before adding 10. This means that if the score is 9, 10, 11 or 12 before adding 10 you are carefree. If the score is more than 89, there is no point in adding 10, we only round it off to 99. This will happen when the dice throw is 9, 10, 11 or 12.

(Refer Slide Time: 45:56)



Suppose the dice throw is not 9, 10, 11 or 12. I simply keep on adding 2s, 3s, I mean 4s and 5s, not even 4, 4 is negative. I always throw 5, 6, 7 or 8, nothing else. Can I reach 99? Yes, I can reach 99 in any number of attempts, let us say 15 attempts or 20 attempts, it does not matter. If I am somewhere around, say, 96 and I get a 7, I will have to cap it to 99. So which point in this ASM will you modify to take care of the design flaw? Of course, hardware is very good. The hardware is very simple. The algorithm is very clearly understood. I have told you so many times. I will not repeat the algorithm. Hardware is simple. All you need is add or subtract in which you can add 10 or add the dice value, the preset value can be 99 and the current value and then this score.

Clearing can be on the condition of subtraction on the starting. Here also we talked about starting with 2 going to 12, in the 12 it will automatically reach to 2. Initially we do this loading and after every 12 we do loading so everything is clear. I compare the dice with this value nine or four enable, I mean the number of attempts I count and the score I compare it. Everything is fine, only one flaw that appears in the ASM. The flaw is in the ASM, namely, I am not giving you the choice of verifying that the score is 99 in cases when you do not have to add 10, only in cases when you have to add 10 because the dice value is 9, 10, 11 or 12. You check whether the score is more than 89 and add only if it is not, else you cap it off to 99. This also should be introduced for values, any value in the dice before adding to the dice, the score value after adding the score here the first time you add the score here before verifying the dice is less than 4 equal to 4 greater than or equal to 9. We have to verify whether this score is going to be 99 by this addition more than ninety nine. If the score will be more than 99, we will go to 99 and stop the game.

This has to be modified here and this is an exercise that I am leaving to you. The exercise comes out of my realization that there is a design flaw, which just occurred to me as I was explaining it to you. That is how you should improve. Once you have a design problem and you work out the specifications, you may think you have taken care of everything but then it suddenly dawns on you that something is wrong. Its flaw there will be design flaw, suddenly you will find this more than 89 or something likely to happen. You are already at 97, you are expecting to win the game, suddenly the score will show 4 and you will be frustrated. So please correct that design flaw as homework for this lecture.