1- Hi!

2- In this first presentation we will consider the basic concepts associated with finite state machines.

3- After watching this presentation, I expect you to be able to explain what is the distinctive feature of this class of circuits, and how they differ from regular sequential circuits. You should also be able to explain the difference between Mealy and Moore machines, to create a state transition diagram, the corresponding VHDL description, and a simulation source file for simple finite state machines.

4- We will follow this sequence, starting with the basic question, then moving on to the difference between the two types of machines, and finally we'll close with an example addressing a sequence detector application.

5- What is a finite state machine?

6- Or in other words, what is the distinctive feature of this type of circuits? Like regular sequential circuits, finite state machines are a subclass of synchronous sequential circuits. The distinctive feature here is that contrary to regular sequential circuits, finite state machines don't have predefined state transition diagrams and output behaviors. In the case of finite state machines the state diagram defines the function of the circuit, while in the case of regular sequential circuits it is the other way around.

7- You may recall this diagram, which is also useful to show that we may consider the existence of an overlap between regular sequential circuits and finite state machines. Actually the counter-based e-dice that was recently considered in this course fits into the area where the boundaries of these two classes overlap – that solution was implemented using a standard regular sequential circuit (the 3-bit counter), but the predefined state diagram was customised with additional logic to enable the cheating mode. We had some degree of freedom that enabled this customisation, although limited, since we wouldn't for example be able to customise it to the point of converting the counter into a shift-register…

8- There are two types of finite state machines, according to how the outputs are updated.

9- The distinctive feature of a Moore machine is that the outputs are solely a function of the present state of the circuit, and therefore they can only change at the rising edge of the clock. On the contrary, in a Mealy machine, the outputs can change at any time, since they are also a direct function of the external inputs, which change at unpredictable times.

10- This explains why the direct connection between the external inputs and the combinational output logic was represented in a different color in the general block diagram for synchronous sequential circuits. It's just that it marks the difference between the two types of machines – we will have Mealy when the connection is present, and Moore when it is absent.

11- It is not always clear that choosing one type or the other will make a difference. However the counter-based e-dice is an interesting example, because the outputs generated in the two additional cheating states depended on the external result selection inputs, and this qualifies the implementation as a Mealy machine. If we wanted to make it as a Moore implementation, the number of states would be much higher…

12- Let's now consider a specific example to illustrate the design process of a finite state machine.

13- In this case we want to detect when the 5-bit sequence "10110" is read at a serial data input. When that happens the finite state machine output shall go high for one clock period as shown in this slide.

14- Notice that if overlapping is allowed this sequence may be detected more often, as shown in this slide. We will ignore overlapping sequences to start.

15- Our first task – and actually the more difficult in state machine design – is to draw a state transition diagram that provides a formal representation of the circuit behavior. A state diagram comprises circles to represent states, and transitional arcs to represent state transitions. In the case of a single external input, there are only two possible conditions to define the next state, so the representation is fairly simple as shown here. We have chosen a Moore machine for this solution, so the output value can be represented inside the state (and is shown in red here). We describe the state diagram as follows: if we are in state S0, then the output will be '0'; and at the next rising edge of the clock the circuit will remain in the same state if the external input is '0', or move on to state S1 if it is '1'. You may wish to pause the video at this point, and make sure that you understand what happens in each of the remaining states. The tricky part, as I said a while ago, is to convert the initial functional requirement into this formal behavioral representation of the circuit. You may try to do it yourself to realise the challenge, and you may also try to modify this diagram to accept overlapping sequences. It will be an interesting and fruitful exercise.

16- This slide shows the VHDL design file for this finite state machine. You should notice in particular that the architecture body is subdivided into three parts that replicate the general block diagram: the state register section, the next state combinational logic block, and the output combinational logic block. The state register section always takes the same form, and the next-state and output sections are of course different from one application to another. Notice also how the VHDL description replicates the state diagram using "case" and "if" statements. There's a lot to learn from this slide, and you should definitely download the file and work on it yourself.

17- Vivado synthesis created the generic technology model shown in this slide. Try to compare it with the general block diagram presented earlier, and notice that the connection represented in different color is not present in this case, since we designed our solution as a Moore machine.

18- For design verification purposes I created the simulation source file shown in this slide…

19- ...which led to the waveforms shown here. Notice that the input sequence driven leads to the state transition highlighted at the right, and includes the expected "10110" sequence.

20- And that's it, and that was a lot. Thanks for your attention!