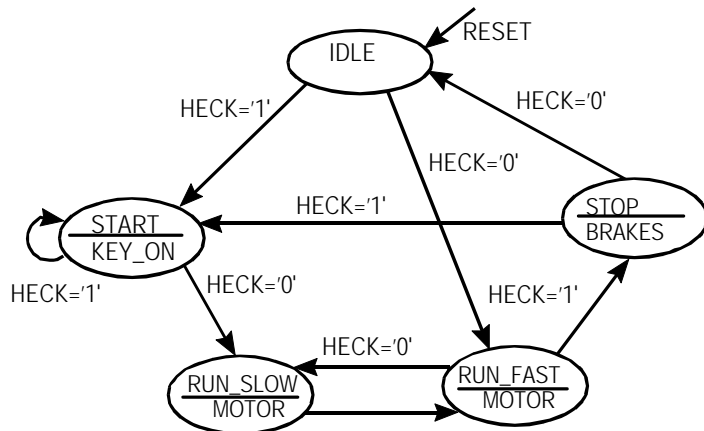


HOW DO I TEST MY FSM?

Before one may answer this, you must ask what kind of test are you referring to: a functional test, and electrical test, etc. These notes refer to functional testing. This tests the functionality of your FSM. It is not concerned with propagation delays, voltage levels, etc.

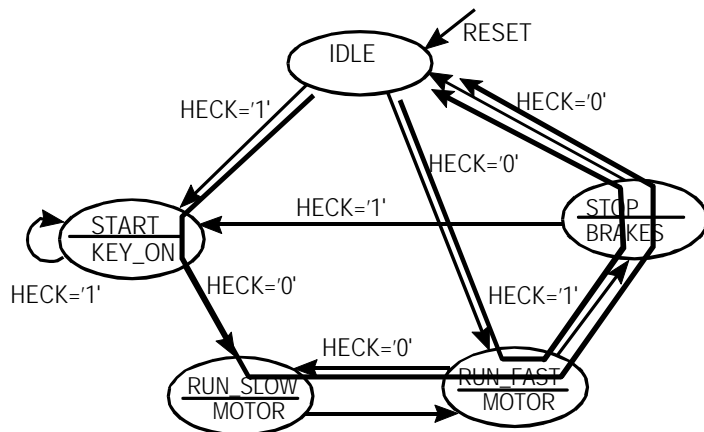
Functionality Testing involves testing two things: (1) The Next State Decoder logic and (2) the Output Decoder logic. To put this another way, it involves checking to see (1) if the correct outputs are obtained and (2) if the machine transitions from state to state as it should.

Consider the machine to the right. It has four outputs to test (KEY_ON, MOTOR, FAST, and BRAKES). It has nine transitions it is supposed to make. To test the transitions from a particular state, the machine must be placed in that state and then clocked from that state with all possible input combinations. Note that the outputs may be tested while the machine is in this state and hence output testing can simply be attached to the transitional testing.



To test all possible transitions in a FSM, $N + A$ tests must be performed where N is the number of possible transitions (arrows on the diagram) and A is the number of Auxiliary tests needed to get the machine to the state to be tested. N is fixed - there is nothing you can do to reduce this. A , on the other hand, may be reduced by careful analysis of the machine. We shall examine the type of analysis to be made.

Each state must be analyzed as follows. We will start with the INIT state, assuming that RESET is used initially to get the machine in this state. We look at all possible transitions from this state (there are two) and check to see which transition requires the fewest number of tests to return the machine to this state. In this case, the transition with $HECK = 0$ is shorter, requiring three tests where as the transition with $HECK = 1$ requires 5.



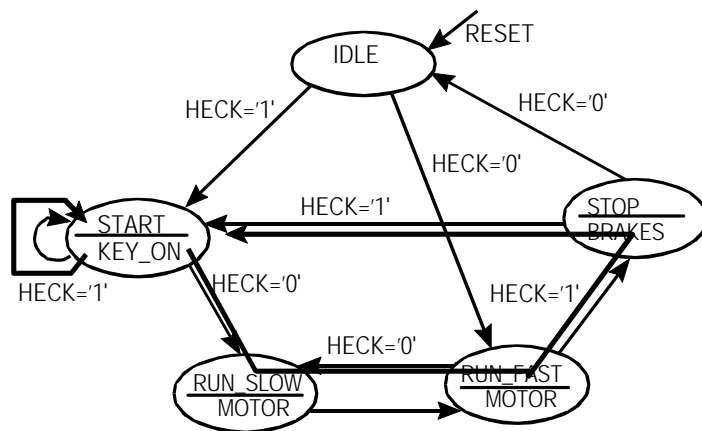
So far, then, we have identified the following initial test sequence

- 1) Apply RESET to get machine is state IDLE. Check to see that it is in this state.
- 2) Apply HECK= 0 and clock the machine. Check to see that it is in state RUN_FAST and that MOTOR is asserted.
- 3) Apply hECK= 1 and clock the machine. Check to see that it is in state STOP and that BRAKE is asserted.
- 4) Apply HECK= 0 and clock the machine. Check to see that it is state IDLE.

Now we are ready to test the other transition from state IDLE.

- 5) Apply HECK= 1 and clock the machine. Check to see that it is in state START and that KEY_ON is asserted.

State IDLE has now been tested We are ready to consider state START. The HECK= 1 is the shortest test - one test leaves it in that state. The HECK= 0 transition requires four tests to return to this state. Hence the test continues with:

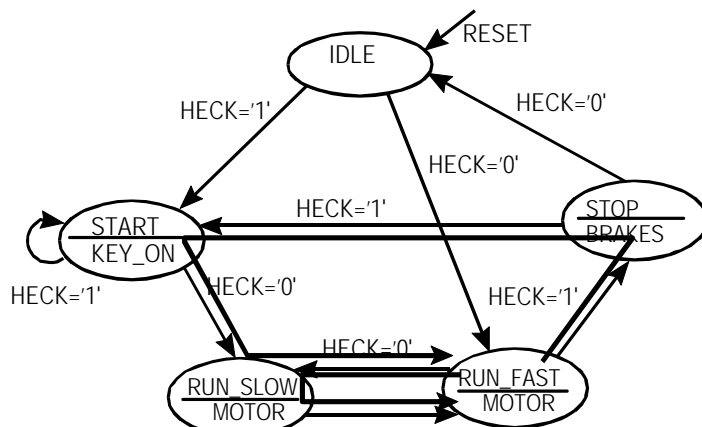


- 6) Apply HECK= 1 and clock the machine. Check to see it is still in state START.
- 7) Apply HECK= 0 and clock the machine. Check to see that it is in state RUN_SLOW and that MOTOR is asserted.

This completes all of the tests for state START. State RUN_SLOW has only one transition so we take that.

- 8) Clock the machine and check to see that it is in state RUN_FAST and the MOTOR is asserted.

Now we have state RUN_FAST. With HECK= 0 it returns to RUN_SLOW and back to RUN_FAST - two tests. The HECK= '1' transition requires four tests. So now we have



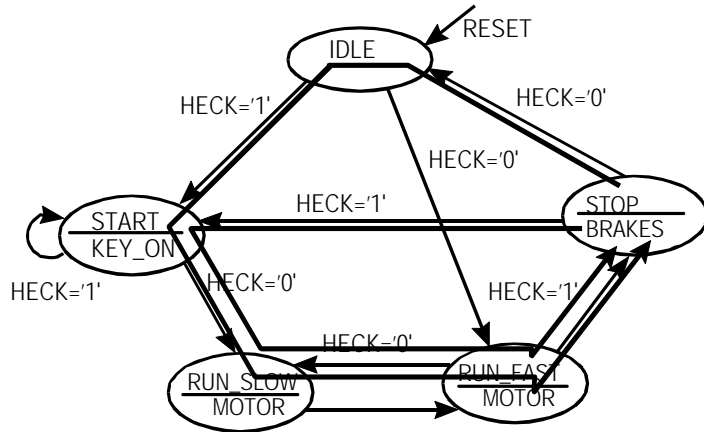
- 9) Set HECK= 0 and clock the

- machine. Check to see that it has returned to RUN_SLOW.
- 10) Clock the machine (This is an Auxillary test to get back to RUN_FAST.
 - 11) Clock the machine with HECK= 1 to get to state STOP. This is also an Auxillary test.

We have only one state to check, STOP. The shortest test is with hECK= 1 (4 tests vs. 5) BUT, there is really only one choice because the HECK= 0 transition has already been tested.

- 12) Clock the machine with HECK= 1 and verify that it reaches state START.

This completely tests the machine. Ten essential tests were performed and two auxiliary tests. It is highly unlikely that less than this is possible.



Note that, in order to perform these tests, you have to have access to the state variables. You should refer to the lab notes on how to make sure they are available. There are test procedures which may be used when you do not have access to the state variables - they are very much more difficult to perform and will not be covered here.

ALTERNATE NOTES ON TESTING

TESTING WITH STATE CAD

AN AUTOMATED TEST UNIT