PROJECT 3:

Project with NuSMV

Nikolay Nikolov

Problem 2

Tasks:

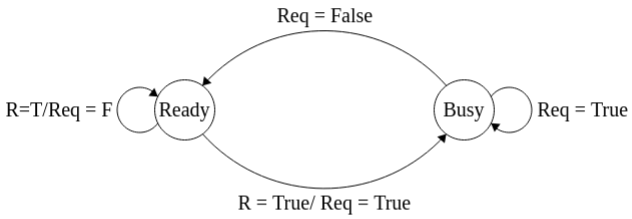
1) Read the tutorial of Nusmv 2.5 and learn the syntax, draw the state diagram of above code.

Verify formula

```
NuSMV > read_model -i short.smv
NuSMV > flatten_hierarchy
NuSMV > encode_variables
NuSMV > build_model
NuSMV > check_ctlspec
-- specification AG (request -> AF state = busy)  is true
NuSMV > check_ctlspec -p "AG(request ->AX(state=busy))
ignoring unbalanced quote ...
-- specification AG (request -> AX state = busy)  is false
-- as demonstrated by the following execution sequence
Trace Description: CTL Counterexample
Trace Type: Counterexample
  -- Loop starts here
  -> State: 1.1 <-
     request = FALSE
     state = ready
  -> State: 1.2 <-
     request = TRUE
     state = busy
  -> State: 1.3 <-
     request = FALSE
     state = ready
NuSMV >
*******  Simulation Starting From State 1.4    ********
NuSMV > show_traces -v
    <!-- ########################## Trace number: 1 ########################## -->
Trace Description: Simulation Trace
Trace Type: Simulation
  -> State: 1.1 <-
     request = FALSE
     state = ready
  -> State: 1.2 <-
     request = TRUE
     state = ready
  -> State: 1.3 <-
     request = TRUE
     state = busy
```

```
-> State: 1.4 <-
  request = FALSE
  state = ready
-> State: 1.5 <-
  request = TRUE
  state = ready
-> State: 1.6 <-
  request = FALSE
  state = busy
-> State: 1.7 <-
  request = FALSE
  state = ready
-> State: 1.8 <-
  request = TRUE
  state = ready
-> State: 1.9 <-
  request = TRUE
  state = busy
-> State: 1.10 <-
  request = FALSE
  state = busy
-> State: 1.11 <-
  request = TRUE
  state = busy
-> State: 1.12 <-
  request = FALSE
  state = ready
```

State Diagram



Come up with 3 more CTL properties and check against the model.

1)

```
NuSMV > check_ctlspec -p "AG(request -> EF(state=busy))"
-- specification AG (request -> EF state = busy)  is true
```

2)

```
NuSMV > check_ctlspec -p "EF(request -> AG(state=busy))"
-- specification EF (request -> AG state = busy)  is true
```

3)

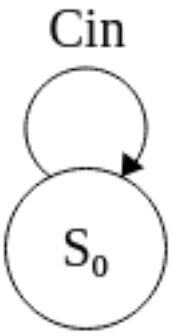
```
NuSMV > check_ctlspec -p "EF(request -> AG(state=busy))"
-- specification EF (request -> AG state = busy)  is true
```

4)

```
NuSMV > check_ctlspec -p "AG(request -> EF(state=ready))"
-- specification AG (request -> EF state = ready)  is true
```

Problem 3:

State Diagram



Trace

```
NuSMV > read_model -i counter.smv
NuSMV > flatten_hierarchy
NuSMV > encode_variables
NuSMV > build_model
NuSMV > pick_state -i

***************  AVAILABLE STATES  *************

  ================ State ================
  0) ------------------------------
  bit0.carry_out = FALSE
  bit0.value = FALSE
  bit1.carry_out = FALSE
  bit1.value = FALSE
  bit2.carry_out = FALSE
  bit2.value = FALSE

********  Simulation Starting From State 1.4  ********
NuSMV > show_traces -v
    <!-- ######################## Trace number: 1 ######################## -->
Trace Description: Simulation Trace
Trace Type: Simulation
  -> State: 1.1 <-
    bit0.value = FALSE
    bit1.value = FALSE
    bit2.value = FALSE
```

```
    bit0.carry_out = FALSE
    bit1.carry_out = FALSE
    bit2.carry_out = FALSE
-> State: 1.2 <-
    bit0.value = TRUE
    bit1.value = FALSE
    bit2.value = FALSE
    bit0.carry_out = TRUE
    bit1.carry_out = FALSE
    bit2.carry_out = FALSE
-> State: 1.3 <-
    bit0.value = FALSE
    bit1.value = TRUE
    bit2.value = FALSE
    bit0.carry_out = FALSE
    bit1.carry_out = FALSE
    bit2.carry_out = FALSE
-> State: 1.4 <-
    bit0.value = TRUE
    bit1.value = TRUE
    bit2.value = FALSE
    bit0.carry_out = TRUE
    bit1.carry_out = TRUE
    bit2.carry_out = FALSE
-> State: 1.5 <-
    bit0.value = FALSE
    bit1.value = FALSE
    bit2.value = TRUE
    bit0.carry_out = FALSE
    bit1.carry_out = FALSE
    bit2.carry_out = FALSE
-> State: 1.6 <-
    bit0.value = TRUE
    bit1.value = FALSE
    bit2.value = TRUE
    bit0.carry_out = TRUE
    bit1.carry_out = FALSE
    bit2.carry_out = FALSE
-> State: 1.7 <-
    bit0.value = FALSE
    bit1.value = TRUE
    bit2.value = TRUE
    bit0.carry_out = FALSE
    bit1.carry_out = FALSE
    bit2.carry_out = FALSE
-> State: 1.8 <-
```

```
bit0.value = TRUE
bit1.value = TRUE
bit2.value = TRUE
bit0.carry_out = TRUE
bit1.carry_out = TRUE
bit2.carry_out = TRUE
```

2) Come up with 3 more CTL property and check against above model.

1)

```
NuSMV > check_ctlspec -p "EF(bit1.value=TRUE)"
-- specification EF bit1.value = TRUE  is true
```

2)

```
NuSMV > check_ctlspec -p "EF(bit2.value=TRUE)"
-- specification EF bit2.value = TRUE  is true
```

3)

```
NuSMV > check_ctlspec -p "EF(bit0.value=TRUE)"
-- specification EF bit0.value = TRUE  is true
```

4)

```
NuSMV > check_ctlspec -p "EF(bit0.value=TRUE)"
-- specification EF bit0.value = TRUE  is true
```

5)

```
NuSMV > check_ctlspec -p "EX(bit0.value=TRUE)"
-- specification EX bit0.value = TRUE  is true
```

6)

```
NuSMV > check_ctlspec -p "EF(bit0.carry_out=FALSE)"
-- specification EF bit0.carry_out = FALSE  is true
```

7)

```
NuSMV > check_ctlspec -p "EF(bit0.value=FALSE)"
-- specification EF bit0.value = FALSE  is true
```

8)

```
NuSMV > check_ctlspec -p "AX(bit0.carry_out=TRUE)"
-- specification AX bit0.carry_out = TRUE  is true
```

Problem 4:

Tasks:

1) Understand the design and draw the transition diagram for the design

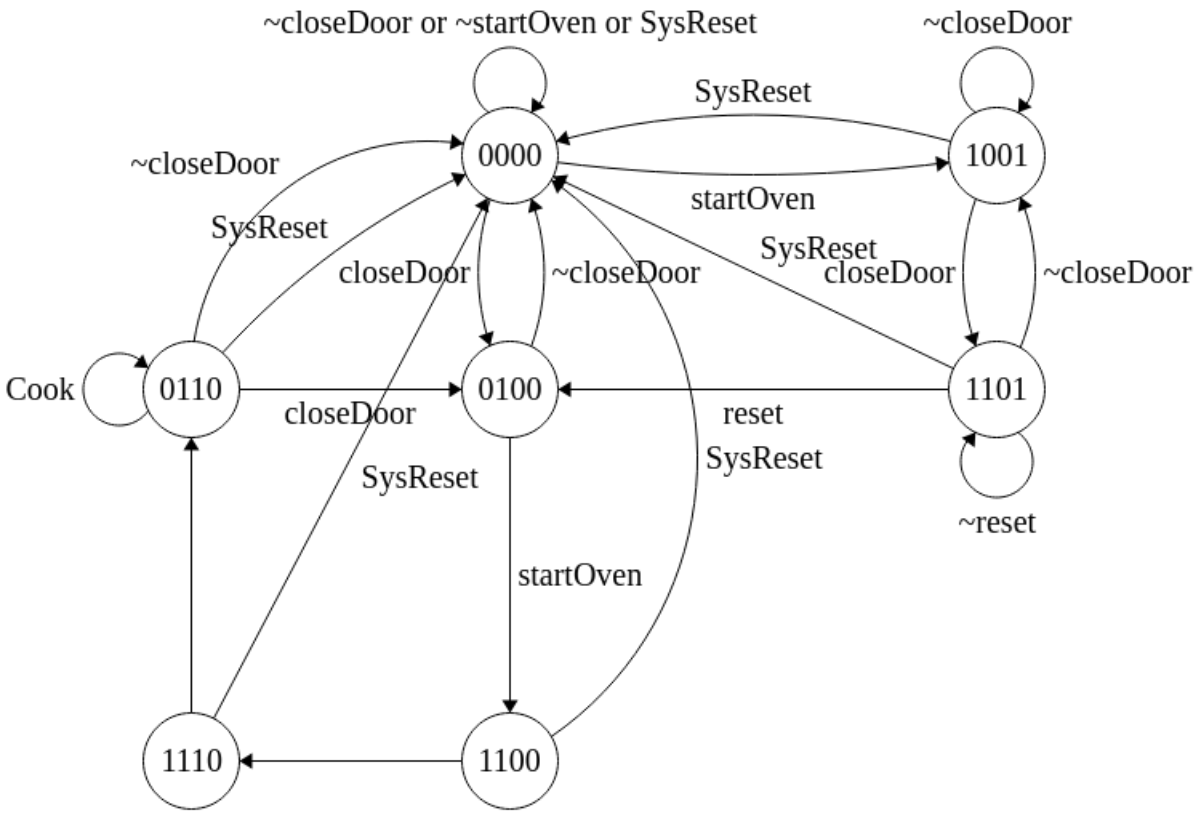


Fig. 1: State Diagram for Microwave

2) Propose 5 new CTL properties. Describe them in English first, then in CTL forms. Your properties should be different from those proposed by other students.

1. If a startOven signal occurs then there is eventually a closeDoor signal
   AG (startOven $\rightarrow$ AF closeDoor)
2. If a startOven signal occurs then there is eventually a not closeDoor signal

AG (startOven → AF ˜closeDoor)

3. If eventually a reset signal occurs then eventually a startOven signal will occur

AF (reset → EF startOven)

4. For all paths startOven signal will not occur until closeDoor occurs

AG (startOven ∪ closeDoor)

5. The signal closedDoor holds infinitely often

AF closedDoor

6. The signal startOven holds infinitely often

AF startOven

3) Create Verilog test benches and run simulation to verify your properties, justify your test result. (show the test result in script form instead of wave form)

Testbench

```
‘timescale 1 ns/10 ps

module tb_microwave();
   reg  clk, sys_reset, reset, closeDoor, startOven, done;
   wire Start, Close, Heat, Error;

   microwave DUT (
                  .clk(clk),
                  .sys_reset(sys_reset),
                  .reset(reset),
                  .closeDoor(closeDoor),
                  .startOven(startOven),
                  .done(done),
                  .Start(Start),
                  .Close(Close),
                  .Heat(Heat),
                  .Error(Error)
                  );

   initial
     begin
        #20 sys_reset = 1'b1;
        #20 sys_reset = 1'b0;
     end

   initial
     begin
        clk= 1'b0;
```

```
    end

  always
    begin
      #5 clk= ~clk;
    end

  initial
    begin
      #40 closeDoor = 1'b0; startOven = 1'b0; reset = 1'b0; done = 1'b0;
      #40 closeDoor = 1'b0; startOven = 1'b0; reset = 1'b0; done = 1'b0;
      #40 closeDoor = 1'b1; startOven = 1'b0; reset = 1'b0; done = 1'b0;
      #40 closeDoor = 1'b1; startOven = 1'b1; reset = 1'b0; done = 1'b0;
      #40 closeDoor = 1'b1; startOven = 1'b1; reset = 1'b0; done = 1'b1;
      #40 closeDoor = 1'b1; startOven = 1'b1; reset = 1'b0; done = 1'b0;
      #40 closeDoor = 1'b1; startOven = 1'b1; reset = 1'b0; done = 1'b0;
      #40 closeDoor = 1'b0; startOven = 1'b0; reset = 1'b0; done = 1'b0;
      #40 closeDoor = 1'b0; startOven = 1'b1; reset = 1'b0; done = 1'b0;
      #40 closeDoor = 1'b1; startOven = 1'b1; reset = 1'b1; done = 1'b0;
      #40 closeDoor = 1'b1; startOven = 1'b1; reset = 1'b0; done = 1'b0;
      #40 closeDoor = 1'b0; startOven = 1'b1; reset = 1'b0; done = 1'b0;
      #40 closeDoor = 1'b1; startOven = 1'b1; reset = 1'b0; done = 1'b1;
      #40 closeDoor = 1'b0; startOven = 1'b0; reset = 1'b0; done = 1'b0;
      #40 closeDoor = 1'b0; startOven = 1'b0; reset = 1'b0; done = 1'b0;
      #40 closeDoor = 1'b0; startOven = 1'b1; reset = 1'b0; done = 1'b0;
      #40 closeDoor = 1'b1; startOven = 1'b1; reset = 1'b0; done = 1'b0;
      #40 closeDoor = 1'b1; startOven = 1'b1; reset = 1'b1; done = 1'b0;
      #40 closeDoor = 1'b1; startOven = 1'b1; reset = 1'b1; done = 1'b0;
      #40 closeDoor = 1'b1; startOven = 1'b1; reset = 1'b0; done = 1'b0;
      #40 closeDoor = 1'b0; startOven = 1'b1; reset = 1'b0; done = 1'b0;

    end
  initial
    begin
      $display(" Start  Close  Heat  Error|       startOven closeDoor reset done "
      $monitor(" %1b    %1b     %1b     %1b|          %1b
%1b     %1b     %1b ", Start, Close, Heat, Error, startOven, closeDoor, reset, d
    end

  initial
    begin
      #1000 $finish;
    end

endmodule //tb_microwave
```

Trace from Testbench

```
# Loading work.microwave
run -all
#  Start Close Heat Error| startOven closeDoor reset done
#  x     x     x    x    |    x       x        x     x
#  0     0     0    0    |    x       x        x     x
#  0     0     0    0    |    0       0        0     0
#  0     0     0    0    |    0       1        0     0
#  0     1     0    0    |    0       1        0     0
#  0     1     0    0    |    1       1        0     0
#  1     1     0    0    |    1       1        0     0
#  1     1     1    0    |    1       1        0     0
#  0     1     1    0    |    1       1        0     0
#  0     1     1    0    |    1       1        0     1
#  0     1     0    0    |    1       1        0     1
#  1     1     0    0    |    1       1        0     1
#  1     1     1    0    |    1       1        0     1
#  0     1     1    0    |    1       1        0     1
#  0     1     1    0    |    1       1        0     0
#  0     1     1    0    |    0       0        0     0
#  0     0     0    0    |    0       0        0     0
#  0     0     0    0    |    1       0        0     0
#  1     0     0    1    |    1       0        0     0
#  1     0     0    1    |    1       1        1     0
#  1     1     0    1    |    1       1        1     0
#  0     1     0    0    |    1       1        1     0
#  1     1     0    0    |    1       1        1     0
#  1     1     1    0    |    1       1        1     0
#  1     1     1    0    |    1       1        0     0
#  0     1     1    0    |    1       1        0     0
#  0     1     1    0    |    1       0        0     0
#  0     0     0    0    |    1       0        0     0
#  1     0     0    1    |    1       0        0     0
#  1     0     0    1    |    1       1        0     1
#  1     1     0    1    |    1       1        0     1
#  1     1     0    1    |    0       0        0     0
#  1     0     0    1    |    0       0        0     0
#  1     0     0    1    |    1       0        0     0
#  1     0     0    1    |    1       1        0     0
#  1     1     0    1    |    1       1        0     0
#  1     1     0    1    |    1       1        1     0
#  0     1     0    0    |    1       1        1     0
#  1     1     0    0    |    1       1        1     0
#  1     1     1    0    |    1       1        1     0
```

```
#  0   1     1     0|          1     1     1     0
#  0   1     1     0|          1     1     0     0
#  0   1     1     0|          1     0     0     0
#  0   0     0     0|          1     0     0     0
#  1   0     0     1|          1     0     0     0
# ** Note: $finish    : /home/nik/Documents/582_PROJ/projects/project_3/source/tb
#    Time: 1 us  Iteration: 0  Instance: /tb_microwave
```

4) Model the design in NuSMV

Model in NuSMV

```
MODULE main
VAR
    closeDoor : boolean;
    startOven : boolean;
    reset     :boolean;
    done      :boolean;
    state : {0000,0100,1001,1100,1110,0110,1101};
ASSIGN
    init(state) := 0000;
    next (state) := case
        state = 0000 & closeDoor = TRUE: 0100;
        state = 0000 & startOven = TRUE: 1001;
        state = 0100 & startOven = TRUE: 1100;
        state = 1001 & closeDoor = TRUE: 1101;
        state = 1101 & reset = TRUE : 0100;
        state = 1100 : 1110;
        state = 1110 : 0110;
        state = 0110 & closeDoor = FALSE : 0000;
        state = 0110 & done = TRUE : 0100;
        state = 0100 & closeDoor = FALSE : 0000;
        TRUE : {0000,0100,1001,1100,1110,0110,1101};
esac;
```

Trace from NuSMV

```
NuSMV > show_traces -t
There are 2 traces currently available.
NuSMV > show_traces -v
    <!-- ########################## Trace number: 2 ########################## -->
Trace Description: Simulation Trace
Trace Type: Simulation
  -> State: 2.1 <-
    closeDoor = FALSE
    startOven = TRUE
    reset = FALSE
```

```
    done = TRUE
    state = 0
  -> State: 2.2 <-
    closeDoor = FALSE
    startOven = FALSE
    reset = TRUE
    done = FALSE
    state = 1001
  -> State: 2.3 <-
    closeDoor = FALSE
    startOven = TRUE
    reset = FALSE
    done = TRUE
    state = 1100
  -> State: 2.4 <-
    closeDoor = FALSE
    startOven = TRUE
    reset = TRUE
    done = TRUE
    state = 1110
NuSMV > simulate -r -k 3
********  Simulation Starting From State 2.4   ********
NuSMV > simulate -r -k 15
********  Simulation Starting From State 2.7   ********
NuSMV > show_traces -t
There are 2 traces currently available.
NuSMV > show_traces -v 2
    <!-- ##################### Trace number: 2 ##################### -->
Trace Description: Simulation Trace
Trace Type: Simulation
  -> State: 2.1 <-
    closeDoor = FALSE
    startOven = TRUE
    reset = FALSE
    done = TRUE
    state = 0
  -> State: 2.2 <-
    closeDoor = FALSE
    startOven = FALSE
    reset = TRUE
    done = FALSE
    state = 1001
  -> State: 2.3 <-
    closeDoor = FALSE
    startOven = TRUE
    reset = FALSE
```

```
    done = TRUE
    state = 1100
  -> State: 2.4 <-
    closeDoor = FALSE
    startOven = TRUE
    reset = TRUE
    done = TRUE
    state = 1110
  -> State: 2.5 <-
    closeDoor = FALSE
    startOven = FALSE
    reset = FALSE
    done = FALSE
    state = 110
  -> State: 2.6 <-
    closeDoor = FALSE
    startOven = FALSE
    reset = FALSE
    done = FALSE
    state = 0
  -> State: 2.7 <-
    closeDoor = FALSE
    startOven = FALSE
    reset = FALSE
    done = TRUE
    state = 1110
  -> State: 2.8 <-
    closeDoor = FALSE
    startOven = TRUE
    reset = TRUE
    done = TRUE
    state = 110
  -> State: 2.9 <-
    closeDoor = TRUE
    startOven = FALSE
    reset = TRUE
    done = TRUE
    state = 0
  -> State: 2.10 <-
    closeDoor = FALSE
    startOven = FALSE
    reset = FALSE
    done = FALSE
    state = 100
  -> State: 2.11 <-
    closeDoor = FALSE
```

```
    startOven = FALSE
    reset = TRUE
    done = FALSE
    state = 0
  -> State: 2.12 <-
    closeDoor = TRUE
    startOven = FALSE
    reset = FALSE
    done = FALSE
    state = 1001
  -> State: 2.13 <-
    closeDoor = FALSE
    startOven = TRUE
    reset = TRUE
    done = FALSE
    state = 1101
  -> State: 2.14 <-
    closeDoor = TRUE
    startOven = FALSE
    reset = FALSE
    done = FALSE
    state = 100
  -> State: 2.15 <-
    closeDoor = TRUE
    startOven = FALSE
    reset = FALSE
    done = FALSE
    state = 100
  -> State: 2.16 <-
    closeDoor = TRUE
    startOven = TRUE
    reset = FALSE
    done = FALSE
    state = 1100
  -> State: 2.17 <-
    closeDoor = TRUE
    startOven = FALSE
    reset = FALSE
    done = TRUE
    state = 1110
  -> State: 2.18 <-
    closeDoor = TRUE
    startOven = TRUE
    reset = FALSE
    done = TRUE
    state = 110
```

```
  -> State: 2.19 <-
     closeDoor = TRUE
     startOven = FALSE
     reset = FALSE
     done = FALSE
     state = 100
  -> State: 2.20 <-
     closeDoor = FALSE
     startOven = FALSE
     reset = TRUE
     done = FALSE
     state = 110
  -> State: 2.21 <-
     closeDoor = FALSE
     startOven = FALSE
     reset = TRUE
     done = TRUE
     state = 0
  -> State: 2.22 <-
     closeDoor = TRUE
     startOven = TRUE
     reset = FALSE
     done = FALSE
     state = 1001
NuSMV >
```

5) Prove/disprove them by the NuSMV package.

1)

```
uSMV > check_ctlspec -p "AG(startOven -> AF closeDoor)"
-- specification AG (startOven -> AF closeDoor)  is false
-- as demonstrated by the following execution sequence
Trace Description: CTL Counterexample
Trace Type: Counterexample
  -> State: 3.1 <-
     closeDoor = FALSE
     startOven = FALSE
     reset = FALSE
     done = FALSE
     state = 0
  -- Loop starts here
  -> State: 3.2 <-
     startOven = TRUE
  -> State: 3.3 <-
     startOven = FALSE
     state = 1001
```

```
  -> State: 3.4 <-
    startOven = TRUE
    state = 0
```

2)

```
SMV > check_ctlspec -p "AG(startOven -> AF closeDoor=FALSE)"
-- specification AG (startOven -> AF closeDoor = FALSE)  is false
-- as demonstrated by the following execution sequence
Trace Description: CTL Counterexample
Trace Type: Counterexample
  -> State: 4.1 <-
    closeDoor = FALSE
    startOven = FALSE
    reset = FALSE
    done = FALSE
    state = 0
  -- Loop starts here
  -> State: 4.2 <-
    closeDoor = TRUE
    startOven = TRUE
  -> State: 4.3 <-
    startOven = FALSE
    state = 100
  -> State: 4.4 <-
    startOven = TRUE
    state = 0
```

3)

```
NuSMV > check_ctlspec -p "AF(reset -> EF startOven)"
-- specification
AF (reset -> EF startOven)  is true
```

4)

```
NuSMV > check_ctlspec -p "EF(startOven -> closeDoor)"
-- specification EF (startOven -> closeDoor)  is true
NuSMV >
```

5)

```
uSMV > check_ctlspec -p "EF(startOven -> closeDoor)"
-- specification EF (startOven -> closeDoor)  is true
NuSMV > check_ctlspec -p "EF(startOven)"
-- specification EF startOven  is true
NuSMV > check_ctlspec -p "EF(closeDoor)"
-- specification EF closeDoor  is true
NuSMV > check_ctlspec -p "EF(reset)"
-- specification EF reset  is true
```

```
NuSMV > check_ctlspec -p "EF(done)"
-- specification EF done  is true
```

6)

```
NuSMV > check_ctlspec -p "EF(done -> closeDoor)"
-- specification EF (done -> closeDoor)  is true
NuSMV > check_ctlspec -p "EF(reset -> closeDoor)"
-- specification EF (reset -> closeDoor)  is true
NuSMV > check_ctlspec -p "EF(reset -> startOven)"
-- specification EF (reset -> startOven)  is true
```

6) Explain why the property is true or false and if the test result meets your expectation.

The results do meet my expectations. For example I would expect that in # 1 "AG(startOven -> AF closeDoor)" will be false since it not for every path startOven preceeds closeDoor. It is possible to have startOven and not closed door. Overall, all the results confirm with the expected behavior.

7) Compare the model checking to simulation, and discuss the advantages and disadvantages of the above two verification methods.

Model checking is superior. Despite that I run multiple cases in the testbench, it failed to capture a lot of the missing states that model checking was able to analyze. What is more, model checking was able to uncover many conditions that the test bench did not cover.