STT SUMMER 2022

REPORT ON CRUISE CONTROLLER MODEL BASED TESTING

STUDENT DETAILS

Student Name: Syed Ahtsham

ASSIGNMENT DETAILS

Course: Software Testing Techniques - STT

Submitted To: Dr. Mudassar Azam Sindhu

Cruise Controller Model Based Testing Using NuSMV

Embedded System: 5-bit Cruise Controller (CC)

Description:

Consider Figure 1 in which a simplified cruise controller is modelled as a deterministic Kripke structure. The bit-vector Bk has a value of k = 5 in this case. The first two bits of the bit-vector represent the mode of the cruise controller. The cruise controller can be in manual mode(bit encoding 00), cruise mode (bit encoding 01) and disengaged mode(bit encoding 10). The second two bits represent the speed of the cruise controller. The speed below the cruising speed is encoded by bits 00, the cruising speed by bits 01 and above the cruising speed is represented by the bits 10. The last bit represents the state of the button used to turn on or off the cruise controller. A bit value of 1 represents the cruise controller is turned on and a bit value of 0 shows that it is turned off. The input symbols for this cruise controller consist of {brake, acc, dec, gas, button} where brake represents the brake input when it is pressed, acc represent the external input due to which the vehicle can gain speed while going downhill and dec represents the external input when the vehicle will decelerate when going uphill. The gas input represents the acceleration gained by pressing the gas pedal or accelerator. The button input represents the input given from the button to turn on or off the cruise controller.

Kripke Structure of 5-bit Cruise Controller:

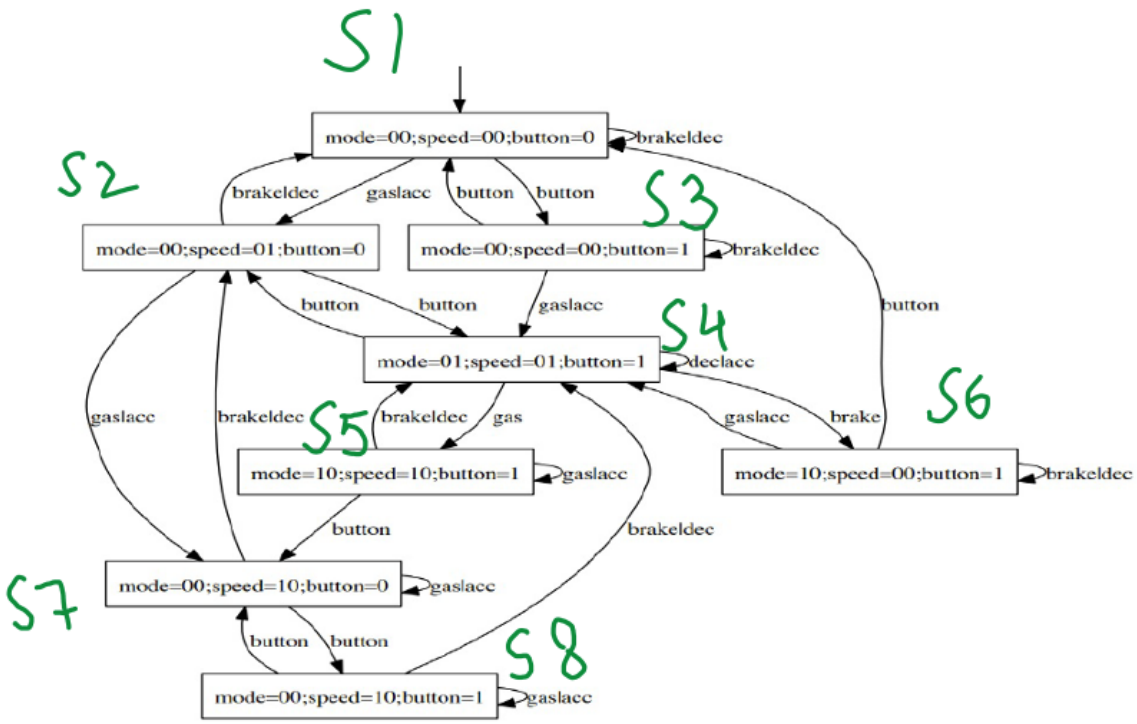

Figure 1: 5-bit Cruise Controller

Modeling and Encoding in NuSMV:

I annotated the Kripke structure of Cruise Controller (CC) with the states names S1, S2, S3, ….. , S8 to the eight states in the kripke structure respectively. After that I encoded the CC model in NuSMV. I wrote the next function for state, modeBit1, modeBit2, speedBit1, speedBit2, and buttonBit. Moreover, I took an enumeration for input symbols defined as input : {brake, acc, dec, gas, button}. The input symbol enum has 5 symbols in it. There are total 40 Transitions since 8 * 5 = 40. And from each state there are 5 transitions for 5 different input symbols.

The code is given below:

NuSMV code for Cruise Controller (CC):

```
-- by Syed Ahtsham (04071813015)
-- STT Summer 2022

MODULE main


VAR
state : {S1, S2, S3, S4, S5, S6, S7, S8};
modeBit1 : boolean;
modeBit2 : boolean;
speedBit1 : boolean;
speedBit2 : boolean;
buttonBit : boolean;

IVAR

input : {brake, acc, dec, gas, button};



ASSIGN
init(state) := S1;

init(modeBit1) := FALSE;
init(modeBit2) := FALSE;
init(speedBit1) := FALSE;
init(speedBit2) := FALSE;
init(buttonBit) := FALSE;



next(state):= case


        state = S1 & input = brake:  S1;
        state = S1 & input = acc:  S2;
        state = S1 & input = dec:  S1;
        state = S1 & input = gas:  S2;
        state = S1 & input = button:  S3;
```

state = S2 & input = brake: S1;
state = S2 & input = acc: S7;
state = S2 & input = dec: S1;
state = S2 & input = gas: S7;
state = S2 & input = button: S4;

state = S3 & input = brake: S3;
state = S3 & input = acc: S4;
state = S3 & input = dec: S3;
state = S3 & input = gas: S4;
state = S3 & input = button: S1;

state = S4 & input = brake: S6;
state = S4 & input = acc: S4;
state = S4 & input = dec: S4;
state = S4 & input = gas: S5;
state = S4 & input = button: S2;

state = S5 & input = brake: S4;
state = S5 & input = acc: S5;
state = S5 & input = dec: S4;
state = S5 & input = gas: S5;
state = S5 & input = button: S7;

state = S6 & input = brake: S6;
state = S6 & input = acc: S4;
state = S6 & input = dec: S6;
state = S6 & input = gas: S4;
state = S6 & input = button: S1;

state = S7 & input = brake: S2;
state = S7 & input = acc: S7;
state = S7 & input = dec: S2;
state = S7 & input = gas: S7;
state = S7 & input = button: S8;

state = S8 & input = brake: S4;
state = S8 & input = acc: S8;
state = S8 & input = dec: S4;
state = S8 & input = gas: S8;
state = S8 & input = button: S7;

```
        --TRUE : state;
esac;


next(modeBit1):= case

        state = S1 & input = brake:  FALSE;
        state = S1 & input = acc:  FALSE;
        state = S1 & input = dec:  FALSE;
        state = S1 & input = gas:  FALSE;
        state = S1 & input = button:  FALSE;

        state = S2 & input = brake:  FALSE;
        state = S2 & input = acc:  FALSE;
        state = S2 & input = dec:  FALSE;
        state = S2 & input = gas:  FALSE;
        state = S2 & input = button:  FALSE;

        state = S3 & input = brake:  FALSE;
        state = S3 & input = acc:  FALSE;
        state = S3 & input = dec:  FALSE;
        state = S3 & input = gas:  FALSE;
        state = S3 & input = button:  FALSE;

        state = S4 & input = brake:  TRUE;
        state = S4 & input = acc:  FALSE;
        state = S4 & input = dec:  FALSE;
        state = S4 & input = gas:  TRUE;
        state = S4 & input = button:  FALSE;

        state = S5 & input = brake:  FALSE;
        state = S5 & input = acc:  TRUE;
        state = S5 & input = dec:  FALSE;
        state = S5 & input = gas:  TRUE;
        state = S5 & input = button:  FALSE;

        state = S6 & input = brake:  TRUE;
        state = S6 & input = acc:  FALSE;
        state = S6 & input = dec:  TRUE;
        state = S6 & input = gas:  FALSE;
        state = S6 & input = button:  FALSE;

        state = S7 & input = brake:  FALSE;
        state = S7 & input = acc:  FALSE;
```

```
        state = S7 & input = dec:  FALSE;
        state = S7 & input = gas:  FALSE;
        state = S7 & input = button:  FALSE;

        state = S8 & input = brake:  FALSE;
        state = S8 & input = acc:  FALSE;
        state = S8 & input = dec:  FALSE;
        state = S8 & input = gas:  FALSE;
        state = S8 & input = button:  FALSE;

        --TRUE : modeBit1;
esac;


next(modeBit2):= case

        state = S1 & input = brake:  FALSE;
        state = S1 & input = acc:  FALSE;
        state = S1 & input = dec:  FALSE;
        state = S1 & input = gas:  FALSE;
        state = S1 & input = button:  FALSE;

        state = S2 & input = brake:  FALSE;
        state = S2 & input = acc:  FALSE;
        state = S2 & input = dec:  FALSE;
        state = S2 & input = gas:  FALSE;
        state = S2 & input = button:  TRUE;

        state = S3 & input = brake:  FALSE;
        state = S3 & input = acc:  TRUE;
        state = S3 & input = dec:  FALSE;
        state = S3 & input = gas:  TRUE;
        state = S3 & input = button:  FALSE;

        state = S4 & input = brake:  FALSE;
        state = S4 & input = acc:  TRUE;
        state = S4 & input = dec:  TRUE;
        state = S4 & input = gas:  FALSE;
        state = S4 & input = button:  FALSE;

        state = S5 & input = brake:  TRUE;
        state = S5 & input = acc:  FALSE;
        state = S5 & input = dec:  TRUE;
        state = S5 & input = gas:  FALSE;
```

```
        state = S5 & input = button:  FALSE;

        state = S6 & input = brake:  FALSE;
        state = S6 & input = acc:  TRUE;
        state = S6 & input = dec:  FALSE;
        state = S6 & input = gas:  TRUE;
        state = S6 & input = button:  FALSE;

        state = S7 & input = brake:  FALSE;
        state = S7 & input = acc:  FALSE;
        state = S7 & input = dec:  FALSE;
        state = S7 & input = gas:  FALSE;
        state = S7 & input = button:  FALSE;

        state = S8 & input = brake:  TRUE;
        state = S8 & input = acc:  FALSE;
        state = S8 & input = dec:  TRUE;
        state = S8 & input = gas:  FALSE;
        state = S8 & input = button:  FALSE;

        --TRUE : modeBit2;
esac;


next(speedBit1):= case

        state = S1 & input = brake:  FALSE;
        state = S1 & input = acc:  FALSE;
        state = S1 & input = dec:  FALSE;
        state = S1 & input = gas:  FALSE;
        state = S1 & input = button:  FALSE;

        state = S2 & input = brake:  FALSE;
        state = S2 & input = acc:  TRUE;
        state = S2 & input = dec:  FALSE;
        state = S2 & input = gas:  TRUE;
        state = S2 & input = button:  FALSE;

        state = S3 & input = brake:  FALSE;
        state = S3 & input = acc:  FALSE;
        state = S3 & input = dec:  FALSE;
        state = S3 & input = gas:  FALSE;
        state = S3 & input = button:  FALSE;
```

```
        state = S4 & input = brake:  FALSE;
        state = S4 & input = acc:  FALSE;
        state = S4 & input = dec:  FALSE;
        state = S4 & input = gas:  TRUE;
        state = S4 & input = button:  FALSE;

        state = S5 & input = brake:  FALSE;
        state = S5 & input = acc:  TRUE;
        state = S5 & input = dec:  FALSE;
        state = S5 & input = gas:  TRUE;
        state = S5 & input = button:  TRUE;

        state = S6 & input = brake:  FALSE;
        state = S6 & input = acc:  FALSE;
        state = S6 & input = dec:  FALSE;
        state = S6 & input = gas:  FALSE;
        state = S6 & input = button:  FALSE;

        state = S7 & input = brake:  FALSE;
        state = S7 & input = acc:  TRUE;
        state = S7 & input = dec:  FALSE;
        state = S7 & input = gas:  TRUE;
        state = S7 & input = button:  TRUE;

        state = S8 & input = brake:  FALSE;
        state = S8 & input = acc:  TRUE;
        state = S8 & input = dec:  FALSE;
        state = S8 & input = gas:  TRUE;
        state = S8 & input = button:  TRUE;

        --TRUE : speedBit1;
esac;


next(speedBit2):= case

        state = S1 & input = brake:  FALSE;
        state = S1 & input = acc:  TRUE;
        state = S1 & input = dec:  FALSE;
        state = S1 & input = gas:  TRUE;
        state = S1 & input = button:  FALSE;

        state = S2 & input = brake:  FALSE;
        state = S2 & input = acc:  FALSE;
```

```
        state = S2 & input = dec:  FALSE;
        state = S2 & input = gas:  FALSE;
        state = S2 & input = button:  TRUE;

        state = S3 & input = brake:  FALSE;
        state = S3 & input = acc:  TRUE;
        state = S3 & input = dec:  FALSE;
        state = S3 & input = gas:  TRUE;
        state = S3 & input = button:  FALSE;

        state = S4 & input = brake:  FALSE;
        state = S4 & input = acc:  TRUE;
        state = S4 & input = dec:  TRUE;
        state = S4 & input = gas:  FALSE;
        state = S4 & input = button:  TRUE;

        state = S5 & input = brake:  TRUE;
        state = S5 & input = acc:  FALSE;
        state = S5 & input = dec:  TRUE;
        state = S5 & input = gas:  FALSE;
        state = S5 & input = button:  FALSE;

        state = S6 & input = brake:  FALSE;
        state = S6 & input = acc:  TRUE;
        state = S6 & input = dec:  FALSE;
        state = S6 & input = gas:  TRUE;
        state = S6 & input = button:  FALSE;

        state = S7 & input = brake:  TRUE;
        state = S7 & input = acc:  FALSE;
        state = S7 & input = dec:  TRUE;
        state = S7 & input = gas:  FALSE;
        state = S7 & input = button:  FALSE;

        state = S8 & input = brake:  TRUE;
        state = S8 & input = acc:  FALSE;
        state = S8 & input = dec:  TRUE;
        state = S8 & input = gas:  FALSE;
        state = S8 & input = button:  FALSE;

        --TRUE : speedBit2;
esac;
```

```
next(buttonBit):= case

        state = S1 & input = brake:  FALSE;
        state = S1 & input = acc:  FALSE;
        state = S1 & input = dec:  FALSE;
        state = S1 & input = gas:  FALSE;
        state = S1 & input = button:  TRUE;

        state = S2 & input = brake:  FALSE;
        state = S2 & input = acc:  FALSE;
        state = S2 & input = dec:  FALSE;
        state = S2 & input = gas:  FALSE;
        state = S2 & input = button:  TRUE;

        state = S3 & input = brake:  TRUE;
        state = S3 & input = acc:  TRUE;
        state = S3 & input = dec:  TRUE;
        state = S3 & input = gas:  TRUE;
        state = S3 & input = button:  FALSE;

        state = S4 & input = brake:  TRUE;
        state = S4 & input = acc:  TRUE;
        state = S4 & input = dec:  TRUE;
        state = S4 & input = gas:  TRUE;
        state = S4 & input = button:  FALSE;

        state = S5 & input = brake:  TRUE;
        state = S5 & input = acc:  TRUE;
        state = S5 & input = dec:  TRUE;
        state = S5 & input = gas:  TRUE;
        state = S5 & input = button:  FALSE;

        state = S6 & input = brake:  TRUE;
        state = S6 & input = acc:  TRUE;
        state = S6 & input = dec:  TRUE;
        state = S6 & input = gas:  TRUE;
        state = S6 & input = button:  FALSE;

        state = S7 & input = brake:  FALSE;
        state = S7 & input = acc:  FALSE;
        state = S7 & input = dec:  FALSE;
        state = S7 & input = gas:  FALSE;
        state = S7 & input = button:  FALSE;
```

```
        state = S8 & input = brake:  TRUE;
        state = S8 & input = acc:  FALSE;
        state = S8 & input = dec:  TRUE;
        state = S8 & input = gas:  FALSE;
        state = S8 & input = button:  FALSE;

        --TRUE : buttonBit;
esac;
```

Part 1:

You are required to encode this model in NuSMV and then write LTL properties for NC, EC and CC (condition coverage). Find the trace sequences you after executing these properties in NuSMV.

Solution:

Model is encoded in the NuSMV. The code is included in the above section.

LTL Properties and Execution Traces for Node Coverage (NC):

The LTL properties for NC i.e., for the eight states are given below.

-- Node Coverage - NC

-- S1 = 00000
--G!(!modeBit1 & !modeBit2 & !speedBit1 & !speedBit2 & !buttonBit)

```
Trace Type: Counterexample
  -> State: 1.1 <-
    state = S1
    modeBit1 = FALSE
    modeBit2 = FALSE
    speedBit1 = FALSE
    speedBit2 = FALSE
    buttonBit = FALSE
  -> Input: 1.2 <-
    input = acc
```

-- S2 = 00010
--G!(!modeBit1 & !modeBit2 & !speedBit1 & speedBit2 & !buttonBit)

```
Trace Type: Counterexample
  -> State: 1.1 <-
    state = S1
    modeBit1 = FALSE
    modeBit2 = FALSE
    speedBit1 = FALSE
    speedBit2 = FALSE
```

```
    buttonBit = FALSE
  -> Input: 1.2 <-
    input = acc
  -> State: 1.2 <-
    state = S2
    speedBit2 = TRUE
  -> Input: 1.3 <-
    input = button
```

-- S3 = 00001

--G!(!modeBit1 & !modeBit2 & !speedBit1 & !speedBit2 & buttonBit)

```
Trace Type: Counterexample
  -> State: 1.1 <-
    state = S1
    modeBit1 = FALSE
    modeBit2 = FALSE
    speedBit1 = FALSE
    speedBit2 = FALSE
    buttonBit = FALSE
  -> Input: 1.2 <-
    input = button
  -> State: 1.2 <-
    state = S3
    buttonBit = TRUE
  -> Input: 1.3 <-
    input = acc
```

-- S4 = 01011

--G!(!modeBit1 & modeBit2 & !speedBit1 & speedBit2 & buttonBit)

```
Trace Type: Counterexample
  -> State: 1.1 <-
    state = S1
    modeBit1 = FALSE
    modeBit2 = FALSE
    speedBit1 = FALSE
    speedBit2 = FALSE
    buttonBit = FALSE
  -> Input: 1.2 <-
    input = acc
  -> State: 1.2 <-
    state = S2
    speedBit2 = TRUE
  -> Input: 1.3 <-
    input = button
  -> State: 1.3 <-
```

```
    state = S4
    modeBit2 = TRUE
    buttonBit = TRUE
  -> Input: 1.4 <-
    input = gas
```

-- S5 = 10101
--G!(modeBit1 & !modeBit2 & speedBit1 & !speedBit2 & buttonBit)

```
Trace Type: Counterexample
  -> State: 1.1 <-
    state = S1
    modeBit1 = FALSE
    modeBit2 = FALSE
    speedBit1 = FALSE
    speedBit2 = FALSE
    buttonBit = FALSE
  -> Input: 1.2 <-
    input = acc
  -> State: 1.2 <-
    state = S2
    speedBit2 = TRUE
  -> Input: 1.3 <-
    input = button
  -> State: 1.3 <-
    state = S4
    modeBit2 = TRUE
    buttonBit = TRUE
  -> Input: 1.4 <-
    input = gas
  -> State: 1.4 <-
    state = S5
    modeBit1 = TRUE
    modeBit2 = FALSE
    speedBit1 = TRUE
    speedBit2 = FALSE
  -> Input: 1.5 <-
    input = brake
```

-- S6 = 10001
--G!(modeBit1 & !modeBit2 & !speedBit1 & !speedBit2 & buttonBit)

```
Trace Type: Counterexample
```

```
  -> State: 1.1 <-
    state = S1
    modeBit1 = FALSE
    modeBit2 = FALSE
    speedBit1 = FALSE
    speedBit2 = FALSE
    buttonBit = FALSE
  -> Input: 1.2 <-
    input = acc
  -> State: 1.2 <-
    state = S2
    speedBit2 = TRUE
  -> Input: 1.3 <-
    input = button
  -> State: 1.3 <-
    state = S4
    modeBit2 = TRUE
    buttonBit = TRUE
  -> Input: 1.4 <-
    input = brake
  -> State: 1.4 <-
    state = S6
    modeBit1 = TRUE
    modeBit2 = FALSE
    speedBit2 = FALSE
  -> Input: 1.5 <-
    input = acc
```

-- S7 = 00100

--G!(!modeBit1 & !modeBit2 & speedBit1 & !speedBit2 & !buttonBit)

```
Trace Type: Counterexample
  -> State: 1.1 <-
    state = S1
    modeBit1 = FALSE
    modeBit2 = FALSE
    speedBit1 = FALSE
    speedBit2 = FALSE
    buttonBit = FALSE
  -> Input: 1.2 <-
    input = acc
  -> State: 1.2 <-
    state = S2
    speedBit2 = TRUE
  -> Input: 1.3 <-
  -> State: 1.3 <-
    state = S7
```

```
    speedBit1 = TRUE
    speedBit2 = FALSE
  -> Input: 1.4 <-
    input = dec
```

-- S8 = 00101

--G!(!modeBit1 & !modeBit2 & speedBit1 & !speedBit2 & buttonBit)

```
Trace Type: Counterexample
  -> State: 1.1 <-
    state = S1
    modeBit1 = FALSE
    modeBit2 = FALSE
    speedBit1 = FALSE
    speedBit2 = FALSE
    buttonBit = FALSE
  -> Input: 1.2 <-
    input = acc
  -> State: 1.2 <-
    state = S2
    speedBit2 = TRUE
  -> Input: 1.3 <-
  -> State: 1.3 <-
    state = S7
    speedBit1 = TRUE
    speedBit2 = FALSE
  -> Input: 1.4 <-
    input = button
  -> State: 1.4 <-
    state = S8
  -> Input: 1.5 <-
    input = brake
```

LTL Properties and Execution Traces for Edge Coverage (EC):

The LTL properties for EC i.e., for the 40 Edges are given below.

-- Edge Coverage - EC

-- Edges outgoing from S1

--G(state = S1 & input = brake -> X !(state = S1))

```
Trace Type: Counterexample
  -> State: 1.1 <-
    state = S1
    modeBit1 = FALSE
    modeBit2 = FALSE
    speedBit1 = FALSE
    speedBit2 = FALSE
    buttonBit = FALSE
  -> Input: 1.2 <-
    input = brake
  -- Loop starts here
  -> State: 1.2 <-
  -> Input: 1.3 <-
    input = dec
  -- Loop starts here
  -> State: 1.3 <-
  -> Input: 1.4 <-
  -- Loop starts here
  -> State: 1.4 <-
  -> Input: 1.5 <-
  -> State: 1.5 <-
```

--G(state = S1 & input = dec -> X !(state = S1))

```
Trace Type: Counterexample
  -> State: 1.1 <-
    state = S1
    modeBit1 = FALSE
    modeBit2 = FALSE
    speedBit1 = FALSE
    speedBit2 = FALSE
    buttonBit = FALSE
  -> Input: 1.2 <-
    input = brake
  -- Loop starts here
  -> State: 1.2 <-
  -> Input: 1.3 <-
    input = dec
  -- Loop starts here
  -> State: 1.3 <-
  -> Input: 1.4 <-
  -- Loop starts here
  -> State: 1.4 <-
  -> Input: 1.5 <-
  -> State: 1.5 <-
```

--G(state = S1 & input = acc -> X !(state = S2))

```
Trace Type: Counterexample
  -> State: 1.1 <-
    state = S1
    modeBit1 = FALSE
    modeBit2 = FALSE
    speedBit1 = FALSE
    speedBit2 = FALSE
    buttonBit = FALSE
  -> Input: 1.2 <-
    input = acc
  -> State: 1.2 <-
    state = S2
    speedBit2 = TRUE
  -> Input: 1.3 <-
    input = button
```

--G(state = S1 & input = gas -> X !(state = S2))

```
Trace Type: Counterexample
  -> State: 1.1 <-
    state = S1
    modeBit1 = FALSE
    modeBit2 = FALSE
    speedBit1 = FALSE
    speedBit2 = FALSE
    buttonBit = FALSE
  -> Input: 1.2 <-
    input = gas
  -> State: 1.2 <-
    state = S2
    speedBit2 = TRUE
  -> Input: 1.3 <-
    input = button
```

--G(state = S1 & input = button -> X !(state = S3))

```
Trace Type: Counterexample
  -> State: 1.1 <-
    state = S1
    modeBit1 = FALSE
    modeBit2 = FALSE
    speedBit1 = FALSE
    speedBit2 = FALSE
    buttonBit = FALSE
  -> Input: 1.2 <-
    input = button
```

```
  -- Loop starts here
  -> State: 1.2 <-
    state = S3
    buttonBit = TRUE
  -> Input: 1.3 <-
    input = dec
```

-- Edges outgoing from S2

--G(state = S2 & input = brake -> X !(state = S1))

```
Trace Type: Counterexample
  -> State: 1.1 <-
    state = S1
    modeBit1 = FALSE
    modeBit2 = FALSE
    speedBit1 = FALSE
    speedBit2 = FALSE
    buttonBit = FALSE
  -> Input: 1.2 <-
    input = dec
  -> State: 1.2 <-
  -> Input: 1.3 <-
    input = acc
  -> State: 1.3 <-
    state = S2
    speedBit2 = TRUE
  -> Input: 1.4 <-
    input = brake
  -- Loop starts here
  -> State: 1.4 <-
    state = S1
    speedBit2 = FALSE
  -> Input: 1.5 <-
    input = dec
```

--G(state = S2 & input = dec -> X !(state = S1))

```
Trace Type: Counterexample
  -> State: 1.1 <-
    state = S1
    modeBit1 = FALSE
    modeBit2 = FALSE
    speedBit1 = FALSE
    speedBit2 = FALSE
    buttonBit = FALSE
```

```
  -> Input: 1.2 <-
    input = dec
  -> State: 1.2 <-
  -> Input: 1.3 <-
    input = acc
  -> State: 1.3 <-
    state = S2
    speedBit2 = TRUE
  -> Input: 1.4 <-
    input = dec
  -- Loop starts here
  -> State: 1.4 <-
    state = S1
    speedBit2 = FALSE
  -> Input: 1.5 <-
```

--G(state = S2 & input = acc -> X !(state = S7))

```
Trace Type: Counterexample
  -> State: 1.1 <-
    state = S1
    modeBit1 = FALSE
    modeBit2 = FALSE
    speedBit1 = FALSE
    speedBit2 = FALSE
    buttonBit = FALSE
  -> Input: 1.2 <-
    input = acc
  -> State: 1.2 <-
    state = S2
    speedBit2 = TRUE
  -> Input: 1.3 <-
  -- Loop starts here
  -> State: 1.3 <-
    state = S7
    speedBit1 = TRUE
    speedBit2 = FALSE
  -> Input: 1.4 <-
```

--G(state = S2 & input = gas -> X !(state = S7))

```
Trace Type: Counterexample
  -> State: 1.1 <-
    state = S1
    modeBit1 = FALSE
    modeBit2 = FALSE
```

```
    speedBit1 = FALSE
    speedBit2 = FALSE
    buttonBit = FALSE
  -> Input: 1.2 <-
    input = acc
  -> State: 1.2 <-
    state = S2
    speedBit2 = TRUE
  -> Input: 1.3 <-
    input = gas
  -- Loop starts here
  -> State: 1.3 <-
    state = S7
    speedBit1 = TRUE
    speedBit2 = FALSE
  -> Input: 1.4 <-
    input = acc
```

--G(state = S2 & input = button -> X !(state = S4))

```
Trace Type: Counterexample
  -> State: 1.1 <-
    state = S1
    modeBit1 = FALSE
    modeBit2 = FALSE
    speedBit1 = FALSE
    speedBit2 = FALSE
    buttonBit = FALSE
  -> Input: 1.2 <-
    input = acc
  -> State: 1.2 <-
    state = S2
    speedBit2 = TRUE
  -> Input: 1.3 <-
    input = button
  -- Loop starts here
  -> State: 1.3 <-
    state = S4
    modeBit2 = TRUE
    buttonBit = TRUE
  -> Input: 1.4 <-
    input = acc
```

-- Edges outgoing from S3

--G(state = S3 & input = brake -> X !(state = S3))

```
Trace Type: Counterexample
  -> State: 1.1 <-
```

```
    state = S1
    modeBit1 = FALSE
    modeBit2 = FALSE
    speedBit1 = FALSE
    speedBit2 = FALSE
    buttonBit = FALSE
  -> Input: 1.2 <-
    input = button
  -> State: 1.2 <-
    state = S3
    buttonBit = TRUE
  -> Input: 1.3 <-
    input = brake
  -- Loop starts here
  -> State: 1.3 <-
  -> Input: 1.4 <-
    input = dec
```

--G(state = S3 & input = dec -> X !(state = S3))

```
Trace Type: Counterexample
  -> State: 1.1 <-
    state = S1
    modeBit1 = FALSE
    modeBit2 = FALSE
    speedBit1 = FALSE
    speedBit2 = FALSE
    buttonBit = FALSE
  -> Input: 1.2 <-
    input = button
  -> State: 1.2 <-
    state = S3
    buttonBit = TRUE
  -> Input: 1.3 <-
    input = brake
  -- Loop starts here
  -> State: 1.3 <-
  -> Input: 1.4 <-
    input = dec
  -- Loop starts here
  -> State: 1.4 <-
  -> Input: 1.5 <-
  -- Loop starts here
  -> State: 1.5 <-
  -> Input: 1.6 <-
  -> State: 1.6 <-
```

--G(state = S3 & input = acc -> X !(state = S4))

```
Trace Type: Counterexample
  -> State: 1.1 <-
    state = S1
    modeBit1 = FALSE
    modeBit2 = FALSE
    speedBit1 = FALSE
    speedBit2 = FALSE
    buttonBit = FALSE
  -> Input: 1.2 <-
    input = button
  -> State: 1.2 <-
    state = S3
    buttonBit = TRUE
  -> Input: 1.3 <-
    input = acc
  -- Loop starts here
  -> State: 1.3 <-
    state = S4
    modeBit2 = TRUE
    speedBit2 = TRUE
  -> Input: 1.4 <-
```

--G(state = S3 & input = gas -> X !(state = S4))

```
Trace Type: Counterexample
  -> State: 1.1 <-
    state = S1
    modeBit1 = FALSE
    modeBit2 = FALSE
    speedBit1 = FALSE
    speedBit2 = FALSE
    buttonBit = FALSE
  -> Input: 1.2 <-
    input = button
  -> State: 1.2 <-
    state = S3
    buttonBit = TRUE
  -> Input: 1.3 <-
    input = gas
  -- Loop starts here
  -> State: 1.3 <-
    state = S4
    modeBit2 = TRUE
    speedBit2 = TRUE
```

```
  -> Input: 1.4 <-
    input = acc
```

--G(state = S3 & input = button -> X !(state = S1))

```
Trace Type: Counterexample
  -> State: 1.1 <-
    state = S1
    modeBit1 = FALSE
    modeBit2 = FALSE
    speedBit1 = FALSE
    speedBit2 = FALSE
    buttonBit = FALSE
  -> Input: 1.2 <-
    input = dec
  -> State: 1.2 <-
  -> Input: 1.3 <-
    input = button
  -> State: 1.3 <-
    state = S3
    buttonBit = TRUE
  -> Input: 1.4 <-
  -- Loop starts here
  -> State: 1.4 <-
    state = S1
    buttonBit = FALSE
  -> Input: 1.5 <-
    input = dec
```

Similarly for the following edges, I found the correct counter examples when I executed them in the NuSMV.

-- Edges outgoing from S4

--G(state = S4 & input = brake -> X !(state = S6))
--G(state = S4 & input = dec -> X !(state = S4))
--G(state = S4 & input = acc -> X !(state = S4))
--G(state = S4 & input = gas -> X !(state = S5))
--G(state = S4 & input = button -> X !(state = S2))

-- Edges outgoing from S5

--G(state = S5 & input = brake -> X !(state = S4))
--G(state = S5 & input = dec -> X !(state = S4))

--G(state = S5 & input = acc -> X !(state = S5))
--G(state = S5 & input = gas -> X !(state = S5))
--G(state = S5 & input = button -> X !(state = S7))

-- Edges outgoing from S6

--G(state = S6 & (input = brake) -> X !(state = S6))
--G(state = S6 & input = dec -> X !(state = S6))
--G(state = S6 & input = acc -> X !(state = S4))
--G(state = S6 & input = gas -> X !(state = S4))
--G(state = S6 & input = button -> X !(state = S1))

-- Edges outgoing from S7

--G(state = S7 & input = brake -> X !(state = S2))
--G(state = S7 & input = dec -> X !(state = S2))
--G(state = S7 & input = acc -> X !(state = S7))
--G(state = S7 & input = gas -> X !(state = S7))
--G(state = S7 & input = button -> X !(state = S8))

-- Edges outgoing from S8

--G(state = S8 & input = brake -> X !(state = S4))

```
Trace Type: Counterexample
  -> State: 1.1 <-
    state = S1
    modeBit1 = FALSE
    modeBit2 = FALSE
    speedBit1 = FALSE
    speedBit2 = FALSE
    buttonBit = FALSE
  -> Input: 1.2 <-
    input = acc
  -> State: 1.2 <-
    state = S2
    speedBit2 = TRUE
  -> Input: 1.3 <-
  -> State: 1.3 <-
    state = S7
    speedBit1 = TRUE
    speedBit2 = FALSE
  -> Input: 1.4 <-
```

```
    input = button
  -> State: 1.4 <-
    state = S8
  -> Input: 1.5 <-
    input = brake
  -- Loop starts here
  -> State: 1.5 <-
    state = S4
    modeBit2 = TRUE
    speedBit1 = FALSE
    speedBit2 = TRUE
    buttonBit = TRUE
  -> Input: 1.6 <-
    input = acc
```

--G(state = S8 & input = dec -> X !(state = S4))

```
Trace Type: Counterexample
  -> State: 1.1 <-
    state = S1
    modeBit1 = FALSE
    modeBit2 = FALSE
    speedBit1 = FALSE
    speedBit2 = FALSE
    buttonBit = FALSE
  -> Input: 1.2 <-
    input = acc
  -> State: 1.2 <-
    state = S2
    speedBit2 = TRUE
  -> Input: 1.3 <-
  -> State: 1.3 <-
    state = S7
    speedBit1 = TRUE
    speedBit2 = FALSE
  -> Input: 1.4 <-
    input = button
  -> State: 1.4 <-
    state = S8
  -> Input: 1.5 <-
    input = brake
  -- Loop starts here
  -> State: 1.5 <-
    state = S4
    modeBit2 = TRUE
    speedBit1 = FALSE
    speedBit2 = TRUE
    buttonBit = TRUE
  -> Input: 1.6 <-
    input = acc
```

--G(state = S8 & input = acc -> X !(state = S8))

```
Trace Type: Counterexample
  -> State: 1.1 <-
    state = S1
    modeBit1 = FALSE
    modeBit2 = FALSE
    speedBit1 = FALSE
    speedBit2 = FALSE
    buttonBit = FALSE
  -> Input: 1.2 <-
    input = acc
  -> State: 1.2 <-
    state = S2
    speedBit2 = TRUE
  -> Input: 1.3 <-
  -> State: 1.3 <-
    state = S7
    speedBit1 = TRUE
    speedBit2 = FALSE
  -> Input: 1.4 <-
    input = button
  -> State: 1.4 <-
    state = S8
  -> Input: 1.5 <-
    input = acc
  -- Loop starts here
  -> State: 1.5 <-
  -> Input: 1.6 <-
    input = gas
  -- Loop starts here
  -> State: 1.6 <-
  -> Input: 1.7 <-
```

--G(state = S8 & input = gas -> X !(state = S8))

```
Trace Type: Counterexample
  -> State: 1.1 <-
    state = S1
    modeBit1 = FALSE
    modeBit2 = FALSE
    speedBit1 = FALSE
    speedBit2 = FALSE
    buttonBit = FALSE
  -> Input: 1.2 <-
    input = acc
  -> State: 1.2 <-
    state = S2
```

```
    speedBit2 = TRUE
  -> Input: 1.3 <-
  -> State: 1.3 <-
    state = S7
    speedBit1 = TRUE
    speedBit2 = FALSE
  -> Input: 1.4 <-
    input = button
  -> State: 1.4 <-
    state = S8
  -> Input: 1.5 <-
    input = acc
  -- Loop starts here
  -> State: 1.5 <-
  -> Input: 1.6 <-
    input = gas
  -- Loop starts here
  -> State: 1.6 <-
  -> Input: 1.7 <-
```

--G(state = S8 & input = button -> X !(state = S7))

```
Trace Type: Counterexample
  -> State: 1.1 <-
    state = S1
    modeBit1 = FALSE
    modeBit2 = FALSE
    speedBit1 = FALSE
    speedBit2 = FALSE
    buttonBit = FALSE
  -> Input: 1.2 <-
    input = acc
  -> State: 1.2 <-
    state = S2
    speedBit2 = TRUE
  -> Input: 1.3 <-
  -> State: 1.3 <-
    state = S7
    speedBit1 = TRUE
    speedBit2 = FALSE
  -> Input: 1.4 <-
    input = button
  -> State: 1.4 <-
    state = S8
  -> Input: 1.5 <-
  -- Loop starts here
  -> State: 1.5 <-
    state = S7
  -> Input: 1.6 <-
```

```
    input = acc
```

LTL Properties for Condition Coverage (CC):

The LTL properties and Execution Traces for CC are given below:

-- Condition Coverage - CC

--G(state = S1 & input = brake -> X !(state = S1))

```
Trace Type: Counterexample
  -> State: 1.1 <-
    state = S1
    modeBit1 = FALSE
    modeBit2 = FALSE
    speedBit1 = FALSE
    speedBit2 = FALSE
    buttonBit = FALSE
  -> Input: 1.2 <-
    input = brake
  -- Loop starts here
  -> State: 1.2 <-
  -> Input: 1.3 <-
    input = dec
  -- Loop starts here
  -> State: 1.3 <-
  -> Input: 1.4 <-
  -- Loop starts here
  -> State: 1.4 <-
  -> Input: 1.5 <-
  -> State: 1.5 <-
```

--G(state = S1 & !(input = brake) -> X !(state = S1))

```
Trace Type: Counterexample
  -- Loop starts here
  -> State: 1.1 <-
    state = S1
    modeBit1 = FALSE
    modeBit2 = FALSE
    speedBit1 = FALSE
    speedBit2 = FALSE
    buttonBit = FALSE
  -> Input: 1.2 <-
    input = dec
  -- Loop starts here
```

--G(state = S1 & input = dec -> X !(state = S1))

```
Trace Type: Counterexample
  -> State: 1.1 <-
    state = S1
    modeBit1 = FALSE
    modeBit2 = FALSE
    speedBit1 = FALSE
    speedBit2 = FALSE
    buttonBit = FALSE
  -> Input: 1.2 <-
    input = brake
  -- Loop starts here
  -> State: 1.2 <-
  -> Input: 1.3 <-
    input = dec
  -- Loop starts here
  -> State: 1.3 <-
  -> Input: 1.4 <-
  -- Loop starts here
  -> State: 1.4 <-
  -> Input: 1.5 <-
  -> State: 1.5 <-
```

--G(state = S1 & !(input = dec) -> X !(state = S1))

```
Trace Type: Counterexample
  -- Loop starts here
  -> State: 1.1 <-
    state = S1
    modeBit1 = FALSE
    modeBit2 = FALSE
    speedBit1 = FALSE
    speedBit2 = FALSE
    buttonBit = FALSE
  -> Input: 1.2 <-
    input = brake
  -- Loop starts here
  -> State: 1.2 <-
  -> Input: 1.3 <-
```

--G(state = S1 & input = acc -> X !(state = S2))

```
Trace Type: Counterexample
  -> State: 1.1 <-
    state = S1
    modeBit1 = FALSE
```

```
    modeBit2 = FALSE
    speedBit1 = FALSE
    speedBit2 = FALSE
    buttonBit = FALSE
  -> Input: 1.2 <-
    input = acc
  -> State: 1.2 <-
    state = S2
    speedBit2 = TRUE
  -> Input: 1.3 <-
    input = button
```

--G(state = S1 & !(input = acc) -> X !(state = S2))

```
Trace Type: Counterexample
  -- Loop starts here
  -> State: 1.1 <-
    state = S1
    modeBit1 = FALSE
    modeBit2 = FALSE
    speedBit1 = FALSE
    speedBit2 = FALSE
    buttonBit = FALSE
  -> Input: 1.2 <-
    input = gas
  -> State: 1.2 <-
    state = S2
    speedBit2 = TRUE
  -> Input: 1.3 <-
    input = dec
  -> State: 1.3 <-
    state = S1
    speedBit2 = FALSE
```

--G(state = S1 & input = gas -> X !(state = S2))

```
Trace Type: Counterexample
  -> State: 1.1 <-
    state = S1
    modeBit1 = FALSE
    modeBit2 = FALSE
    speedBit1 = FALSE
    speedBit2 = FALSE
    buttonBit = FALSE
  -> Input: 1.2 <-
    input = gas
  -> State: 1.2 <-
    state = S2
```

```
    speedBit2 = TRUE
  -> Input: 1.3 <-
    input = button
```

--G(state = S1 & !(input = gas) -> X !(state = S2))

```
Trace Type: Counterexample
  -- Loop starts here
  -> State: 1.1 <-
    state = S1
    modeBit1 = FALSE
    modeBit2 = FALSE
    speedBit1 = FALSE
    speedBit2 = FALSE
    buttonBit = FALSE
  -> Input: 1.2 <-
    input = acc
  -> State: 1.2 <-
    state = S2
    speedBit2 = TRUE
```

--G(state = S1 & input = button -> X !(state = S3))

```
Trace Type: Counterexample
  -> State: 1.1 <-
    state = S1
    modeBit1 = FALSE
    modeBit2 = FALSE
    speedBit1 = FALSE
    speedBit2 = FALSE
    buttonBit = FALSE
  -> Input: 1.2 <-
    input = button
  -- Loop starts here
  -> State: 1.2 <-
    state = S3
    buttonBit = TRUE
  -> Input: 1.3 <-
    input = dec
```

--G(state = S1 & !(input = button) -> X !(state = S3))

This is True

--G(state = S2 & input = brake -> X !(state = S1))

```
Trace Type: Counterexample
  -> State: 1.1 <-
    state = S1
    modeBit1 = FALSE
    modeBit2 = FALSE
    speedBit1 = FALSE
    speedBit2 = FALSE
    buttonBit = FALSE
  -> Input: 1.2 <-
    input = dec
  -> State: 1.2 <-
  -> Input: 1.3 <-
    input = acc
  -> State: 1.3 <-
    state = S2
    speedBit2 = TRUE
  -> Input: 1.4 <-
    input = brake
  -- Loop starts here
  -> State: 1.4 <-
    state = S1
    speedBit2 = FALSE
  -> Input: 1.5 <-
    input = dec
```

--G(state = S2 & !(input = brake) -> X !(state = S1))

```
Trace Type: Counterexample
  -> State: 1.1 <-
    state = S1
    modeBit1 = FALSE
    modeBit2 = FALSE
    speedBit1 = FALSE
    speedBit2 = FALSE
    buttonBit = FALSE
  -> Input: 1.2 <-
    input = dec
  -> State: 1.2 <-
  -> Input: 1.3 <-
    input = acc
  -> State: 1.3 <-
    state = S2
    speedBit2 = TRUE
  -> Input: 1.4 <-
    input = dec
  -- Loop starts here
  -> State: 1.4 <-
    state = S1
    speedBit2 = FALSE
  -> Input: 1.5 <-
```

--G(state = S2 & input = dec -> X !(state = S1))

```
Trace Type: Counterexample
  -> State: 1.1 <-
    state = S1
    modeBit1 = FALSE
    modeBit2 = FALSE
    speedBit1 = FALSE
    speedBit2 = FALSE
    buttonBit = FALSE
  -> Input: 1.2 <-
    input = dec
  -> State: 1.2 <-
  -> Input: 1.3 <-
    input = acc
  -> State: 1.3 <-
    state = S2
    speedBit2 = TRUE
  -> Input: 1.4 <-
    input = dec
  -- Loop starts here
  -> State: 1.4 <-
    state = S1
    speedBit2 = FALSE
  -> Input: 1.5 <-
```

--G(state = S2 & !(input = dec) -> X !(state = S1))

```
Trace Type: Counterexample
  -> State: 1.1 <-
    state = S1
    modeBit1 = FALSE
    modeBit2 = FALSE
    speedBit1 = FALSE
    speedBit2 = FALSE
    buttonBit = FALSE
  -> Input: 1.2 <-
    input = dec
  -> State: 1.2 <-
  -> Input: 1.3 <-
    input = acc
  -> State: 1.3 <-
    state = S2
    speedBit2 = TRUE
  -> Input: 1.4 <-
    input = brake
```

--G(state = S2 & input = acc -> X !(state = S7))

```
Trace Type: Counterexample
  -> State: 1.1 <-
    state = S1
    modeBit1 = FALSE
    modeBit2 = FALSE
    speedBit1 = FALSE
    speedBit2 = FALSE
    buttonBit = FALSE
  -> Input: 1.2 <-
    input = acc
  -> State: 1.2 <-
    state = S2
    speedBit2 = TRUE
  -> Input: 1.3 <-
  -- Loop starts here
  -> State: 1.3 <-
    state = S7
    speedBit1 = TRUE
    speedBit2 = FALSE
  -> Input: 1.4 <-
```

--G(state = S2 & !(input = acc) -> X !(state = S7))

```
Trace Type: Counterexample
  -> State: 1.1 <-
    state = S1
    modeBit1 = FALSE
    modeBit2 = FALSE
    speedBit1 = FALSE
    speedBit2 = FALSE
    buttonBit = FALSE
  -> Input: 1.2 <-
    input = acc
  -> State: 1.2 <-
    state = S2
    speedBit2 = TRUE
  -> Input: 1.3 <-
    input = gas
  -- Loop starts here
  -> State: 1.3 <-
    state = S7
    speedBit1 = TRUE
    speedBit2 = FALSE
  -> Input: 1.4 <-
    input = acc
```

--G(state = S2 & input = gas -> X !(state = S7))

```
Trace Type: Counterexample
  -> State: 1.1 <-
    state = S1
    modeBit1 = FALSE
    modeBit2 = FALSE
    speedBit1 = FALSE
    speedBit2 = FALSE
    buttonBit = FALSE
  -> Input: 1.2 <-
    input = acc
  -> State: 1.2 <-
    state = S2
    speedBit2 = TRUE
  -> Input: 1.3 <-
    input = gas
  -- Loop starts here
  -> State: 1.3 <-
    state = S7
    speedBit1 = TRUE
    speedBit2 = FALSE
  -> Input: 1.4 <-
    input = acc
```

--G(state = S2 & !(input = gas) -> X !(state = S7))

```
Trace Type: Counterexample
  -> State: 1.1 <-
    state = S1
    modeBit1 = FALSE
    modeBit2 = FALSE
    speedBit1 = FALSE
    speedBit2 = FALSE
    buttonBit = FALSE
  -> Input: 1.2 <-
    input = acc
  -> State: 1.2 <-
    state = S2
    speedBit2 = TRUE
  -> Input: 1.3 <-
  -- Loop starts here
  -> State: 1.3 <-
    state = S7
    speedBit1 = TRUE
    speedBit2 = FALSE
  -> Input: 1.4 <-
  -- Loop starts here
```

```
  -> State: 1.4 <-
  -> Input: 1.5 <-
```

--G(state = S2 & input = button -> X !(state = S4))

```
Trace Type: Counterexample
  -> State: 1.1 <-
    state = S1
    modeBit1 = FALSE
    modeBit2 = FALSE
    speedBit1 = FALSE
    speedBit2 = FALSE
    buttonBit = FALSE
  -> Input: 1.2 <-
    input = acc
  -> State: 1.2 <-
    state = S2
    speedBit2 = TRUE
  -> Input: 1.3 <-
    input = button
  -- Loop starts here
  -> State: 1.3 <-
    state = S4
    modeBit2 = TRUE
    buttonBit = TRUE
  -> Input: 1.4 <-
    input = acc
```

--G(state = S2 & !(input = button) -> X !(state = S4))

This is True

Similarly for the following Conditions Coverage LTLs, I found the correct counter examples when I executed them in the NuSMV.

--G(state = S3 & input = brake -> X !(state = S3))
--G(state = S3 & !(input = brake) -> X !(state = S3))
--G(state = S3 & input = dec -> X !(state = S3))
--G(state = S3 & !(input = dec) -> X !(state = S3))
--G(state = S3 & input = acc -> X !(state = S4))
--G(state = S3 & !(input = acc) -> X !(state = S4))
--G(state = S3 & input = gas -> X !(state = S4))
--G(state = S3 & !(input = gas) -> X !(state = S4))

--G(state = S3 & input = button -> X !(state = S1))
--G(state = S3 & !(input = button) -> X !(state = S1))

--G(state = S4 & input = brake -> X !(state = S6))
--G(state = S4 & !(input = brake) -> X !(state = S6))
--G(state = S4 & input = dec -> X !(state = S4))
--G(state = S4 & !(input = dec) -> X !(state = S4))
--G(state = S4 & input = acc -> X !(state = S4))
--G(state = S4 & !(input = acc) -> X !(state = S4))
--G(state = S4 & input = gas -> X !(state = S5))
--G(state = S4 & !(input = gas) -> X !(state = S5))
--G(state = S4 & input = button -> X !(state = S2))
--G(state = S4 & !(input = button) -> X !(state = S2))

--G(state = S5 & input = brake -> X !(state = S4))
--G(state = S5 & !(input = brake) -> X !(state = S4))
--G(state = S5 & input = dec -> X !(state = S4))
--G(state = S5 & !(input = dec) -> X !(state = S4))
--G(state = S5 & input = acc -> X !(state = S5))
--G(state = S5 & !(input = acc) -> X !(state = S5))
--G(state = S5 & input = gas -> X !(state = S5))
--G(state = S5 & !(input = gas) -> X !(state = S5))
--G(state = S5 & input = button -> X !(state = S7))
--G(state = S5 & !(input = button) -> X !(state = S7))

--G(state = S6 & input = brake -> X !(state = S6))
--G(state = S6 & !(input = brake) -> X !(state = S6))
--G(state = S6 & input = dec -> X !(state = S6))
--G(state = S6 & !(input = dec) -> X !(state = S6))
--G(state = S6 & input = acc -> X !(state = S4))
--G(state = S6 & !(input = acc) -> X !(state = S4))
--G(state = S6 & input = gas -> X !(state = S4))
--G(state = S6 & !(input = gas) -> X !(state = S4))
--G(state = S6 & input = button -> X !(state = S1))
--G(state = S6 & !(input = button) -> X !(state = S1))

--G(state = S7 & input = brake -> X !(state = S2))

--G(state = S7 & !(input = brake) -> X !(state = S2))

--G(state = S7 & input = dec -> X !(state = S2))
--G(state = S7 & !(input = dec) -> X !(state = S2))
--G(state = S7 & input = acc -> X !(state = S7))
--G(state = S7 & !(input = acc) -> X !(state = S7))
--G(state = S7 & input = gas -> X !(state = S7))
--G(state = S7 & !(input = gas) -> X !(state = S7))
--G(state = S7 & input = button -> X !(state = S8))
--G(state = S7 & !(input = button) -> X !(state = S8))

--G(state = S8 & input = brake -> X !(state = S4))

```
Trace Type: Counterexample
  -> State: 1.1 <-
    state = S1
    modeBit1 = FALSE
    modeBit2 = FALSE
    speedBit1 = FALSE
    speedBit2 = FALSE
    buttonBit = FALSE
  -> Input: 1.2 <-
    input = acc
  -> State: 1.2 <-
    state = S2
    speedBit2 = TRUE
  -> Input: 1.3 <-
  -> State: 1.3 <-
    state = S7
    speedBit1 = TRUE
    speedBit2 = FALSE
  -> Input: 1.4 <-
    input = button
  -> State: 1.4 <-
    state = S8
  -> Input: 1.5 <-
    input = brake
  -- Loop starts here
  -> State: 1.5 <-
    state = S4
    modeBit2 = TRUE
    speedBit1 = FALSE
    speedBit2 = TRUE
    buttonBit = TRUE
  -> Input: 1.6 <-
```

```
    input = acc
```

--G(state = S8 & !(input = brake) -> X !(state = S4))

```
race Type: Counterexample
  -> State: 1.1 <-
    state = S1
    modeBit1 = FALSE
    modeBit2 = FALSE
    speedBit1 = FALSE
    speedBit2 = FALSE
    buttonBit = FALSE
  -> Input: 1.2 <-
    input = acc
  -> State: 1.2 <-
    state = S2
    speedBit2 = TRUE
  -> Input: 1.3 <-
  -> State: 1.3 <-
    state = S7
    speedBit1 = TRUE
    speedBit2 = FALSE
  -> Input: 1.4 <-
    input = button
  -> State: 1.4 <-
    state = S8
  -> Input: 1.5 <-
    input = dec
  -- Loop starts here
  -> State: 1.5 <-
    state = S4
    modeBit2 = TRUE
    speedBit1 = FALSE
    speedBit2 = TRUE
    buttonBit = TRUE
  -> Input: 1.6 <-
    input = acc
```

--G(state = S8 & input = dec -> X !(state = S4))

```
Trace Type: Counterexample
  -> State: 1.1 <-
    state = S1
    modeBit1 = FALSE
    modeBit2 = FALSE
    speedBit1 = FALSE
    speedBit2 = FALSE
    buttonBit = FALSE
```

```
  -> Input: 1.2 <-
    input = acc
  -> State: 1.2 <-
    state = S2
    speedBit2 = TRUE
  -> Input: 1.3 <-
  -> State: 1.3 <-
    state = S7
    speedBit1 = TRUE
    speedBit2 = FALSE
  -> Input: 1.4 <-
    input = button
  -> State: 1.4 <-
    state = S8
  -> Input: 1.5 <-
    input = brake
  -- Loop starts here
  -> State: 1.5 <-
    state = S4
    modeBit2 = TRUE
    speedBit1 = FALSE
    speedBit2 = TRUE
    buttonBit = TRUE
  -> Input: 1.6 <-
    input = acc
```

--G(state = S8 & !(input = dec) -> X !(state = S4))

```
Trace Type: Counterexample
  -> State: 1.1 <-
    state = S1
    modeBit1 = FALSE
    modeBit2 = FALSE
    speedBit1 = FALSE
    speedBit2 = FALSE
    buttonBit = FALSE
  -> Input: 1.2 <-
    input = acc
  -> State: 1.2 <-
    state = S2
    speedBit2 = TRUE
  -> Input: 1.3 <-
  -> State: 1.3 <-
    state = S7
    speedBit1 = TRUE
    speedBit2 = FALSE
  -> Input: 1.4 <-
    input = button
  -> State: 1.4 <-
    state = S8
```

```
  -> Input: 1.5 <-
    input = brake
  -- Loop starts here
  -> State: 1.5 <-
    state = S4
    modeBit2 = TRUE
    speedBit1 = FALSE
    speedBit2 = TRUE
    buttonBit = TRUE
  -> Input: 1.6 <-
    input = acc
```

--G(state = S8 & input = acc -> X !(state = S8))

```
Trace Type: Counterexample
  -> State: 1.1 <-
    state = S1
    modeBit1 = FALSE
    modeBit2 = FALSE
    speedBit1 = FALSE
    speedBit2 = FALSE
    buttonBit = FALSE
  -> Input: 1.2 <-
    input = acc
  -> State: 1.2 <-
    state = S2
    speedBit2 = TRUE
  -> Input: 1.3 <-
  -> State: 1.3 <-
    state = S7
    speedBit1 = TRUE
    speedBit2 = FALSE
  -> Input: 1.4 <-
    input = button
  -> State: 1.4 <-
    state = S8
  -> Input: 1.5 <-
    input = acc
  -- Loop starts here
  -> State: 1.5 <-
  -> Input: 1.6 <-
    input = gas
  -- Loop starts here
  -> State: 1.6 <-
  -> Input: 1.7 <-
```

--G(state = S8 & !(input = acc) -> X !(state = S8))

```
Trace Type: Counterexample
  -> State: 1.1 <-
    state = S1
    modeBit1 = FALSE
    modeBit2 = FALSE
    speedBit1 = FALSE
    speedBit2 = FALSE
    buttonBit = FALSE
  -> Input: 1.2 <-
    input = acc
  -> State: 1.2 <-
    state = S2
    speedBit2 = TRUE
  -> Input: 1.3 <-
  -> State: 1.3 <-
    state = S7
    speedBit1 = TRUE
    speedBit2 = FALSE
  -> Input: 1.4 <-
    input = button
  -> State: 1.4 <-
    state = S8
  -> Input: 1.5 <-
    input = gas
  -> State: 1.5 <-
  -> Input: 1.6 <-
    input = dec
```

--G(state = S8 & input = gas -> X !(state = S8))

```
Trace Type: Counterexample
  -> State: 1.1 <-
    state = S1
    modeBit1 = FALSE
    modeBit2 = FALSE
    speedBit1 = FALSE
    speedBit2 = FALSE
    buttonBit = FALSE
  -> Input: 1.2 <-
    input = acc
  -> State: 1.2 <-
    state = S2
    speedBit2 = TRUE
  -> Input: 1.3 <-
  -> State: 1.3 <-
    state = S7
    speedBit1 = TRUE
    speedBit2 = FALSE
  -> Input: 1.4 <-
    input = button
```

```
  -> State: 1.4 <-
    state = S8
  -> Input: 1.5 <-
    input = acc
  -- Loop starts here
  -> State: 1.5 <-
  -> Input: 1.6 <-
    input = gas
  -- Loop starts here
  -> State: 1.6 <-
  -> Input: 1.7 <-
```

--G(state = S8 & !(input = gas) -> X !(state = S8))

```
Trace Type: Counterexample
  -> State: 1.1 <-
    state = S1
    modeBit1 = FALSE
    modeBit2 = FALSE
    speedBit1 = FALSE
    speedBit2 = FALSE
    buttonBit = FALSE
  -> Input: 1.2 <-
    input = acc
  -> State: 1.2 <-
    state = S2
    speedBit2 = TRUE
  -> Input: 1.3 <-
  -> State: 1.3 <-
    state = S7
    speedBit1 = TRUE
    speedBit2 = FALSE
  -> Input: 1.4 <-
    input = button
  -> State: 1.4 <-
    state = S8
  -> Input: 1.5 <-
    input = acc
  -> State: 1.5 <-
  -> Input: 1.6 <-
    input = dec
  -- Loop starts here
  -> State: 1.6 <-
    state = S4
    modeBit2 = TRUE
    speedBit1 = FALSE
    speedBit2 = TRUE
    buttonBit = TRUE
  -> Input: 1.7 <-
```

```
    input = acc
  -> State: 1.7 <-
```

--G(state = S8 & input = button -> X !(state = S7))

```
Trace Type: Counterexample
  -> State: 1.1 <-
    state = S1
    modeBit1 = FALSE
    modeBit2 = FALSE
    speedBit1 = FALSE
    speedBit2 = FALSE
    buttonBit = FALSE
  -> Input: 1.2 <-
    input = acc
  -> State: 1.2 <-
    state = S2
    speedBit2 = TRUE
  -> Input: 1.3 <-
  -> State: 1.3 <-
    state = S7
    speedBit1 = TRUE
    speedBit2 = FALSE
  -> Input: 1.4 <-
    input = button
  -> State: 1.4 <-
    state = S8
  -> Input: 1.5 <-
  -- Loop starts here
  -> State: 1.5 <-
    state = S7
  -> Input: 1.6 <-
    input = acc
```

--G(state = S8 & !(input = button) -> X !(state = S7))

This is True

Formalizing Behavioral Requirements into LTL properties:

-- 1. Whenever the brake pedal or gas pedal is pressed in the cruise mode the cruise controller will be disengaged.

--G(state = S4 & (input = gas | input = brake) -> X(state = S5 | state = S6))

-- OR

--G((modeBit1 = FALSE & modeBit2 = TRUE) & (input = gas | input = brake) -> X (modeBit1 = TRUE & modeBit2 = FALSE))

--Trap Property:

--F(state = S4 & (input = gas | input = brake) & X!(state = S5 | state = S6))

-- OR

--F((modeBit1 = FALSE & modeBit2 = TRUE) & (input = gas | input = brake) & X !(modeBit1 = TRUE & modeBit2 = FALSE))

```
Trace Type: Counterexample
  -> State: 1.1 <-
    state = S1
    modeBit1 = FALSE
    modeBit2 = FALSE
    speedBit1 = FALSE
    speedBit2 = FALSE
    buttonBit = FALSE
  -> Input: 1.2 <-
    input = acc
  -> State: 1.2 <-
    state = S2
    speedBit2 = TRUE
  -> Input: 1.3 <-
    input = button
  -> State: 1.3 <-
    state = S4
    modeBit2 = TRUE
    buttonBit = TRUE
  -> Input: 1.4 <-
    input = gas
```

```
  -- Loop starts here
  -> State: 1.4 <-
    state = S5
    modeBit1 = TRUE
    modeBit2 = FALSE
    speedBit1 = TRUE
    speedBit2 = FALSE
  -> Input: 1.5 <-
    input = acc
```

-- 2. If the cruise controller is in cruise mode it will remain in the cruise mode even if the vehicle is going uphill or downhill.

--G(state = S4 & (input = dec | input = acc) -> X (state = S4))

-- OR

--G((modeBit1 = FALSE & modeBit2 = TRUE) & (speedBit1 = FALSE & speedBit2 = TRUE) & (input = dec | input = acc) -> X (speedBit1 = FALSE & speedBit2 = TRUE))

--Trap Property:
--F(state = S4 & (input = dec | input = acc) & X !(state = S4))

-F((modeBit1 = FALSE & modeBit2 = TRUE) & (speedBit1 = FALSE & speedBit2 = TRUE) & (input = dec | input = acc) & X !(speedBit1 = FALSE & speedBit2 = TRUE))

```
Trace Type: Counterexample
  -> State: 1.1 <-
    state = S1
    modeBit1 = FALSE
    modeBit2 = FALSE
    speedBit1 = FALSE
    speedBit2 = FALSE
    buttonBit = FALSE
  -> Input: 1.2 <-
    input = acc
  -> State: 1.2 <-
    state = S2
    speedBit2 = TRUE
  -> Input: 1.3 <-
    input = button
  -> State: 1.3 <-
    state = S4
    modeBit2 = TRUE
```

```
    buttonBit = TRUE
  -> Input: 1.4 <-
    input = acc
  -> State: 1.4 <-
  -> Input: 1.5 <-
    input = brake
  -- Loop starts here
  -> State: 1.5 <-
    state = S6
```

-- 3. The cruise controller will eventually stop when the brake is applied no matter which mode is it in currently.

--F(((modeBit1 = FALSE & modeBit2 = FALSE) | (modeBit1 = FALSE & modeBit2 = TRUE) | (modeBit1 = TRUE & modeBit2 = FALSE)) & input = brake -> X(buttonBit = FALSE))

--Trap Property:

--G(((modeBit1 = FALSE & modeBit2 = FALSE) | (modeBit1 = FALSE & modeBit2 = TRUE) | (modeBit1 = TRUE & modeBit2 = FALSE)) & input = brake & X!(buttonBit = FALSE))

```
Trace Type: Counterexample
  -> State: 1.1 <-
    state = S1
    modeBit1 = FALSE
    modeBit2 = FALSE
    speedBit1 = FALSE
    speedBit2 = FALSE
    buttonBit = FALSE
  -> Input: 1.2 <-
    input = brake
  -> State: 1.2 <-
  -> Input: 1.3 <-
    input = acc
  -- Loop starts here
  -> State: 1.3 <-
    state = S2
    speedBit2 = TRUE
  -> Input: 1.4 <-
  -> State: 1.4 <-
    state = S7
    speedBit1 = TRUE
```

```
    speedBit2 = FALSE
  -> Input: 1.5 <-
    input = dec
  -> State: 1.5 <-
    state = S2
```

-- 4. The cruise controller once in the disengaged mode will remain in this mode no matter even if it received acceleration via the gas pedal or external acceleration while going downhill.

--G((state = S5) & (input = gas | input = acc) -> X(state = S5))

--OR

--G((modeBit1 = TRUE & modeBit2 = FALSE) & (speedBit1 = TRUE & speedBit2 = FALSE) & (input = acc | input = gas) -> X(modeBit1 = TRUE & modeBit2 = FALSE))

--Trap Property:

--F((state = S5) & (input = gas | input = acc) & X!(state = S5))

--OR

--F((modeBit1 = TRUE & modeBit2 = FALSE) & (speedBit1 = TRUE & speedBit2 = FALSE) & (input = acc | input = gas) & X!(modeBit1 = TRUE & modeBit2 = FALSE))

```
Trace Type: Counterexample
  -> State: 1.1 <-
    state = S1
    modeBit1 = FALSE
    modeBit2 = FALSE
    speedBit1 = FALSE
    speedBit2 = FALSE
    buttonBit = FALSE
  -> Input: 1.2 <-
    input = acc
  -> State: 1.2 <-
    state = S2
    speedBit2 = TRUE
  -> Input: 1.3 <-
```

```
  input = button
-> State: 1.3 <-
  state = S4
  modeBit2 = TRUE
  buttonBit = TRUE
-> Input: 1.4 <-
  input = gas
-> State: 1.4 <-
  state = S5
  modeBit1 = TRUE
  modeBit2 = FALSE
  speedBit1 = TRUE
  speedBit2 = FALSE
-> Input: 1.5 <-
  input = acc
-> State: 1.5 <-
-> Input: 1.6 <-
  input = dec
-- Loop starts here
```

```
-- 1. Whenever the brake pedal or gas pedal is pressed in the cruise mode the cruise controller will be
disengaged.

    --G(state = S4 & (input = gas | input = brake) -> X(state = S5 | state = S6))

    -- OR

    --G((modeBit1 = FALSE & modeBit2 = TRUE) & (input = gas | input = brake) -> X (modeBit1 = TRUE &
    modeBit2 = FALSE))


    --Trap Property:

    --F(state = S4 & (input = gas | input = brake) & X!(state = S5 | state = S6))

    -- OR

    --F((modeBit1 = FALSE & modeBit2 = TRUE) & (input = gas | input = brake) & X !(modeBit1 = TRUE &
    modeBit2 = FALSE))



-- 2. If the cruise controller is in cruise mode it will remain in the cruise mode even if the vehicle
is going uphill or downhill.

    --G(state = S4 & (input = dec | input = acc) -> X (state = S4))

    -- OR

    --G((modeBit1 = FALSE & modeBit2 = TRUE) & (speedBit1 = FALSE & speedBit2 = TRUE) & (input = dec |
    input = acc) -> X (speedBit1 = FALSE & speedBit2 = TRUE))


    --Trap Property:
    --F(state = S4 & (input = dec | input = acc) & X !(state = S4))

    --F((modeBit1 = FALSE & modeBit2 = TRUE) & (speedBit1 = FALSE & speedBit2 = TRUE) & (input = dec |
    input = acc) & X !(speedBit1 = FALSE & speedBit2 = TRUE))



-- 3. The cruise controller will eventually stop when the brake is applied no matter which mode is it in
currently.


    --F(((modeBit1 = FALSE & modeBit2 = FALSE) | (modeBit1 = FALSE & modeBit2 = TRUE) | (modeBit1 = TRUE
    & modeBit2 = FALSE)) & input = brake -> X(buttonBit = FALSE))


    --Trap Property:

    --G(((modeBit1 = FALSE & modeBit2 = FALSE) | (modeBit1 = FALSE & modeBit2 = TRUE) | (modeBit1 = TRUE
    & modeBit2 = FALSE)) & input = brake & X!(buttonBit = FALSE))
```

```
-- 4. The cruise controller once in the disengaged mode will remain in this mode no matter even if it
received acceleration via
--the gas pedal or external accelaration while going downhill.


    --G((state = S5) & (input = gas | input = acc) -> X(state = S5))

    --OR

    --G((modeBit1 = TRUE & modeBit2 = FALSE) & (speedBit1 = TRUE & speedBit2 = FALSE) & (input = acc |
    input = gas) -> X(modeBit1 = TRUE & modeBit2 = FALSE))


    --Trap Property:

    --F((state = S5) & (input = gas | input = acc) & X!(state = S5))

    --OR

    --F((modeBit1 = TRUE & modeBit2 = FALSE) & (speedBit1 = TRUE & speedBit2 = FALSE) & (input = acc |
    input = gas) & X!(modeBit1 = TRUE & modeBit2 = FALSE))
```