Sriharsha Singam

Lab #6 Report

ECE 2031 L08

15 October 2018

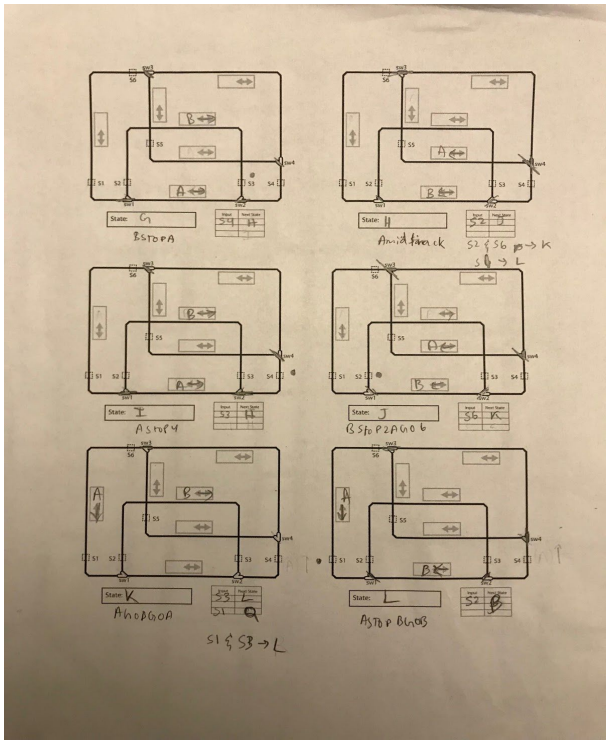

Figure 1. This is the first of three of the train worksheets that describe each state of a State Machine that allows 2 trains to run in close proximity without crashing on 4 different tracks.

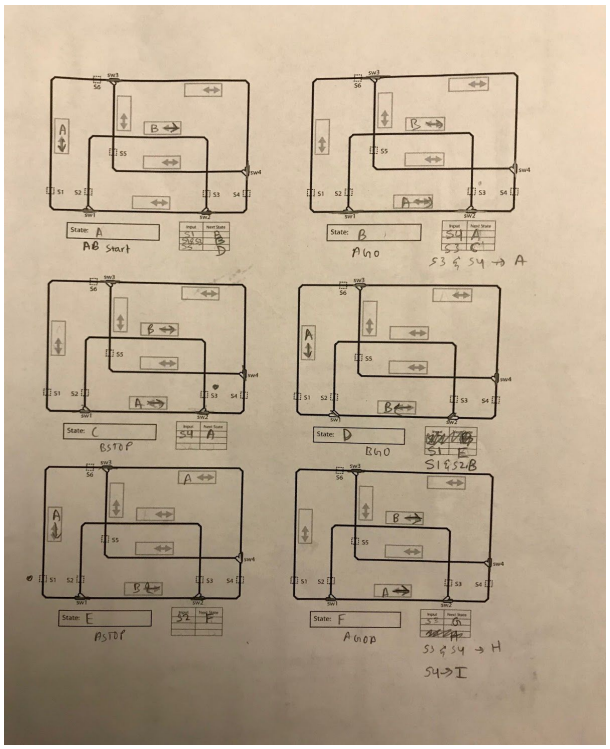

Figure 2. This is the second of three of the train worksheets that describe each state of a State Machine that allows 2 trains to run in close proximity without crashing on 4 different tracks.

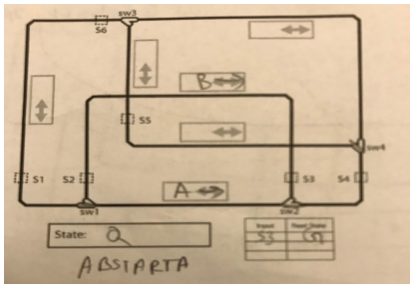

Figure 3. This the third of three of the train worksheets that describe each state of a State Machine that allows 2 trains to run in close proximity without crashing on 4 different tracks.

THE IS THE VHDL (A HARDWARE DESCRIPTION LANGUAGE) IMPLEMENTATION OF A STATE MACHINE THAT ALLOWS 2 TRAINS TO RUN IN CLOSE PROXIMITY ON 4 DIFFERENT TRACKS.

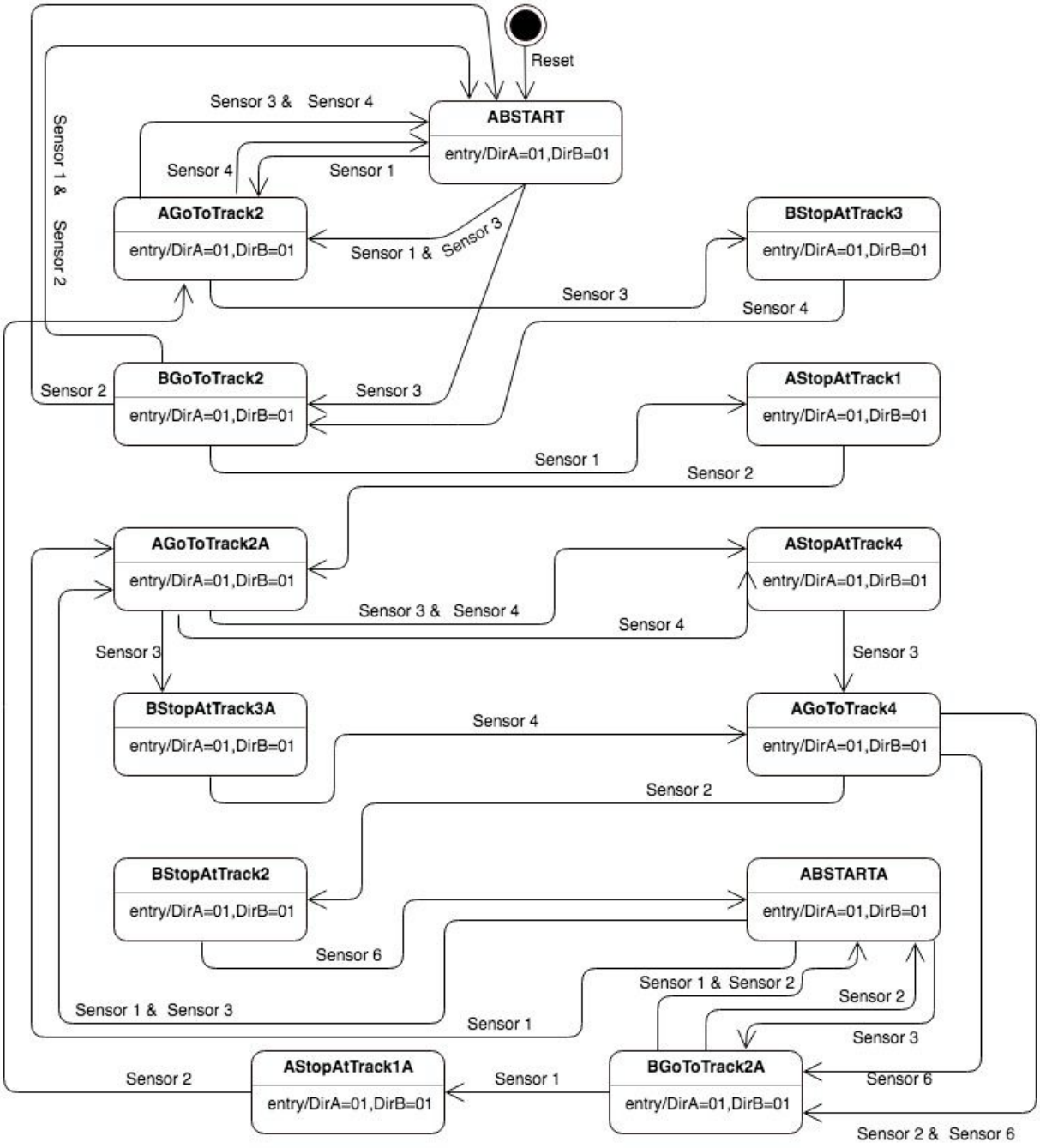

Figure 4. This is the computer-drawn UML chart for the State Machine that allows 2 trains to run in close proximity without crashing on 4 different tracks.

APPENDIX A

```
-- TCONTROL.vhd
-- Train State Machine Implementation
-- Sriharsha Singam
-- ECE 2031 L08
-- October 5, 2018

LIBRARY IEEE;
USE  IEEE.STD_LOGIC_1164.all;
USE  IEEE.STD_LOGIC_ARITH.all;
USE  IEEE.STD_LOGIC_UNSIGNED.all;


ENTITY Tcontrol IS
     PORT(
          reset, clock, sensor1, sensor2      : IN std_logic;
          sensor3, sensor4, sensor5, sensor6  : IN std_logic;
          switch1, switch2, switch3, switch4  : OUT std_logic;
          dirA, dirB                          : OUT
std_logic_vector(1 DOWNTO 0)
     );
END Tcontrol;


ARCHITECTURE a OF Tcontrol IS
     -- We create a new TYPE called STATE_TYPE that is only allowed
     -- to have the values specified here. This
     -- 1) allows us to use helpful names for values instead of
     --    arbitrary values
     -- 2) ensures that we can never accidentally set a signal of
     --    this type to an invalid value, and
     -- 3) helps the synthesis software create efficient hardware
for our design.
     TYPE STATE_TYPE IS (
          ABSTART,
        ABSTARTa,
          AGoToTrack2,
          BStopAtTrack3,
          BGoToTrack2,
          AStopAtTrack1,
        AGoToTrack2a,
        AStopAtTrack4,
        BStopAtTrack3a,
        AGoToTrack4,
        BStopAtTrack2,
        AStopAtTrack1a,
```

```
        BGoToTrack2a
    );
    -- Now we can create a signal of our new type.  Note that there
is
    -- nothing special about the names "state" or "state_type", but
it makes
    -- sense to use these names because that is how we are using
them.
    SIGNAL state                                : STATE_TYPE;
    -- Here we create some new internal signals which will be
concatenations
    -- of some of the sensor signals.  This will make CASE
statements easier.
    SIGNAL sensor12, sensor13, sensor34,sensor26        :
std_logic_vector(1 DOWNTO 0);

BEGIN
    -- A process statement is required for clocked logic, such as a
state machine.
    PROCESS (clock, reset)
    BEGIN
        IF reset = '1' THEN
            -- Reset to this state
            state <= ABSTART;
        ELSIF clock'EVENT AND clock = '1' THEN
            -- Case statement to determine next state.
            -- Case statements are a nice, clean way to make
decisions
            -- based on different values of a signal.
            CASE state IS
                WHEN ABSTART =>
                    CASE Sensor13 IS
                        WHEN "00" => state <= ABSTART;
                        WHEN "01" => state <= BGoToTrack2;
                        WHEN "10" => state <= AGoToTrack2;
                        WHEN "11" => state <= AGoToTrack2;
                        WHEN OTHERS => state <= ABSTART;
                    END CASE;
            WHEN AGoToTrack2 =>
                CASE Sensor34 IS
                        WHEN "00" => state <= AGoToTrack2;
                        WHEN "01" => state <= BGoToTrack2;
                        WHEN "10" => state <= BStopAtTrack3;
                        WHEN "11" => state <= AGoToTrack2;
                        WHEN OTHERS => state <= ABSTART;
                    END CASE;
            WHEN BStopAtTrack3 =>
                    IF Sensor4 = '1' THEN
```

```
                state <= BGoToTrack2;
            ELSE
                state <= BStopAtTrack3;
            END IF;

WHEN BGoToTrack2 =>
    CASE Sensor12 IS
                WHEN "00" => state <= BGoToTrack2;
                WHEN "01" => state <= ABSTART;
                WHEN "10" => state <= AStopAtTrack1;
                WHEN "11" => state <= ABSTART;
                WHEN OTHERS => state <= ABSTART;
            END CASE;

WHEN AStopAtTrack1 =>
            IF Sensor2 = '1' THEN
                state <= AGoToTrack2a;
            ELSE
                state <= AStopAtTrack1;
            END IF;

WHEN AGoToTrack2a =>
    CASE Sensor34 IS
                WHEN "00" => state <= AGoToTrack2a;
                WHEN "01" => state <= AStopAtTrack4;
                WHEN "10" => state <= BStopAtTrack3a;
                WHEN "11" => state <= AGoToTrack4;
                WHEN OTHERS => state <= ABSTARTa;
            END CASE;

WHEN AStopAtTrack4 =>
    IF Sensor3 = '1' THEN
                state <= AGoToTrack4;
            ELSE
                state <= AStopAtTrack4;
            END IF;
WHEN BStopAtTrack3a =>
    IF Sensor4 = '1' THEN
                state <= AGoToTrack4;
            ELSE
                state <= BStopAtTrack3a;
            END IF;

WHEN AGoToTrack4 =>
    CASE Sensor26 IS
                WHEN "00" => state <= AGoToTrack4;
                WHEN "01" => state <= BGoToTrack2a;
                WHEN "10" => state <= BStopAtTrack2;
```

```
                              WHEN "11" => state <= BGoToTrack2a;
                              WHEN OTHERS => state <= ABSTARTa;
                         END CASE;

                WHEN BStopAtTrack2 =>
                    IF Sensor6 = '1' THEN
                              state <= ABSTARTa;
                         ELSE
                              state <= BStopAtTrack2;
                         END IF;

                WHEN ABSTARTa =>
                    CASE Sensor13 IS
                              WHEN "00" => state <= ABSTARTa;
                              WHEN "01" => state <= BGoToTrack2a;
                              WHEN "10" => state <= AGoToTrack2a;
                              WHEN "11" => state <= AGoToTrack2a;
                              WHEN OTHERS => state <= ABSTARTa;
                         END CASE;


                WHEN AStopAtTrack1a =>
                    IF Sensor2 = '1' THEN
                              state <= AGoToTrack2;
                         ELSE
                              state <= AStopAtTrack1a;
                         END IF;


                WHEN BGoToTrack2a =>
                         CASE Sensor12 IS
                              WHEN "00" => state <= BGoToTrack2a;
                              WHEN "01" => state <= ABSTARTa;
                              WHEN "10" => state <= AStopAtTrack1a;
                              WHEN "11" => state <= ABSTARTa;
                              WHEN OTHERS => state <= ABSTARTa;
                         END CASE;

                END CASE;
           END IF;
     END PROCESS;

     -- Notice that all of the following logic is NOT in a process
block,
     -- and thus does not depend on any clock.  Everything here is
pure combinational
     -- logic, and exists in parallel with everything else.
```

```
-- Combine bits for the internal signals declared above.
-- ("&" operator concatenates bits)
sensor12 <= sensor1 & sensor2;
sensor13 <= sensor1 & sensor3;
sensor34 <= sensor3 & sensor4;
sensor26 <= sensor2 & sensor6;

-- The following outputs depend on the state.
WITH state SELECT Switch1 <=
   '0' WHEN ABSTART,
   '0' WHEN ABSTARTa,
   '0' WHEN AGoToTrack2,
   '0' WHEN BStopAtTrack3,
   '1' WHEN BGoToTrack2,
   '1' WHEN AStopAtTrack1,
   '0' WHEN AGoToTrack2a,
   '1' WHEN AStopAtTrack4,
   '0' WHEN BStopAtTrack3a,
   '1' WHEN AGoToTrack4,
   '1' WHEN BStopAtTrack2,
   '1' WHEN AStopAtTrack1a,
   '1' WHEN BGoToTrack2a;
WITH state SELECT Switch2 <=
   '0' WHEN ABSTART,
   '0' WHEN ABSTARTa,
   '0' WHEN AGoToTrack2,
   '0' WHEN BStopAtTrack3,
   '1' WHEN BGoToTrack2,
   '1' WHEN AStopAtTrack1,
   '0' WHEN AGoToTrack2a,
   '1' WHEN AStopAtTrack4,
   '0' WHEN BStopAtTrack3a,
   '1' WHEN AGoToTrack4,
   '1' WHEN BStopAtTrack2,
   '1' WHEN AStopAtTrack1a,
   '1' WHEN BGoToTrack2a;
WITH state SELECT Switch3 <=
   '0' WHEN ABSTART,
   '0' WHEN ABSTARTa,
   '0' WHEN AGoToTrack2,
   '0' WHEN BStopAtTrack3,
   '0' WHEN BGoToTrack2,
   '0' WHEN AStopAtTrack1,
   '1' WHEN AGoToTrack2a,
   '1' WHEN AStopAtTrack4,
   '1' WHEN BStopAtTrack3a,
   '1' WHEN AGoToTrack4,
   '1' WHEN BStopAtTrack2,
```

```
      '1' WHEN AStopAtTrack1a,
      '1' WHEN BGoToTrack2a;
   WITH state SELECT Switch4 <=
      '0' WHEN ABSTART,
      '0' WHEN ABSTARTa,
      '0' WHEN AGoToTrack2,
      '0' WHEN BStopAtTrack3,
      '0' WHEN BGoToTrack2,
      '0' WHEN AStopAtTrack1,
      '1' WHEN AGoToTrack2a,
      '1' WHEN AStopAtTrack4,
      '1' WHEN BStopAtTrack3a,
      '1' WHEN AGoToTrack4,
      '1' WHEN BStopAtTrack2,
      '1' WHEN AStopAtTrack1a,
      '1' WHEN BGoToTrack2a;
   WITH state SELECT DirA <=
      "01" WHEN ABSTART,
      "01" WHEN ABSTARTa,
      "01" WHEN AGoToTrack2,
      "01" WHEN BStopAtTrack3,
      "01" WHEN BGoToTrack2,
      "00" WHEN AStopAtTrack1,
      "01" WHEN AGoToTrack2a,
      "00" WHEN AStopAtTrack4,
      "01" WHEN BStopAtTrack3a,
      "01" WHEN AGoToTrack4,
      "01" WHEN BStopAtTrack2,
      "00" WHEN AStopAtTrack1a,
      "01" WHEN BGoToTrack2a;
   WITH state SELECT DirB <=
      "10" WHEN ABSTART,
      "10" WHEN ABSTARTa,
      "10" WHEN AGoToTrack2,
      "00" WHEN BStopAtTrack3,
      "10" WHEN BGoToTrack2,
      "10" WHEN AStopAtTrack1,
      "10" WHEN AGoToTrack2a,
      "10" WHEN AStopAtTrack4,
      "00" WHEN BStopAtTrack3a,
      "10" WHEN AGoToTrack4,
      "00" WHEN BStopAtTrack2,
      "10" WHEN AStopAtTrack1a,
      "10" WHEN BGoToTrack2a;

END a;
```