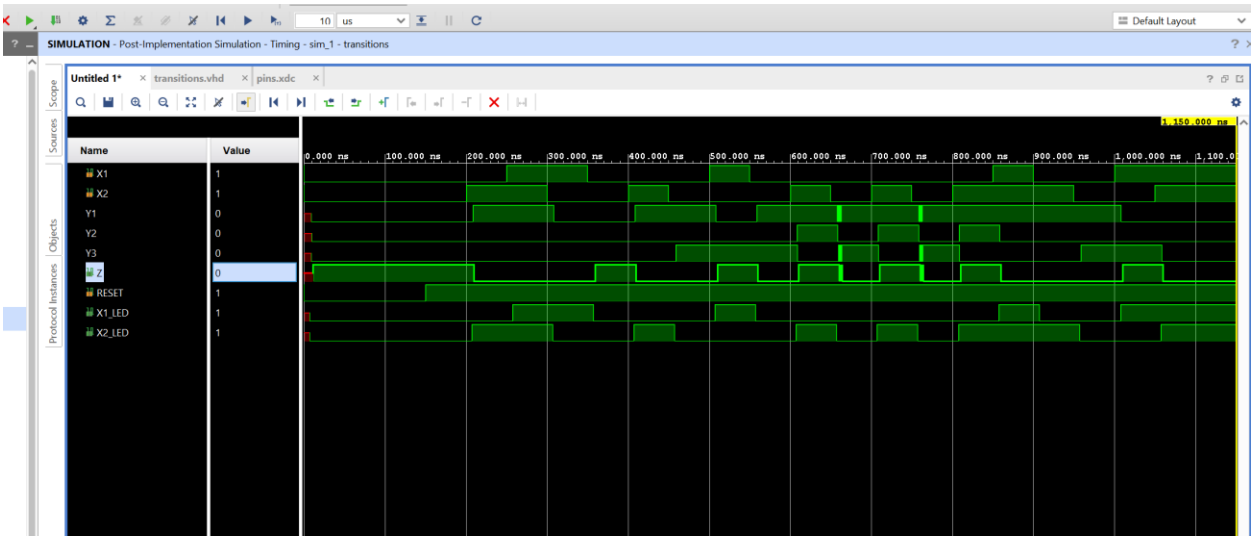
ECE 4525

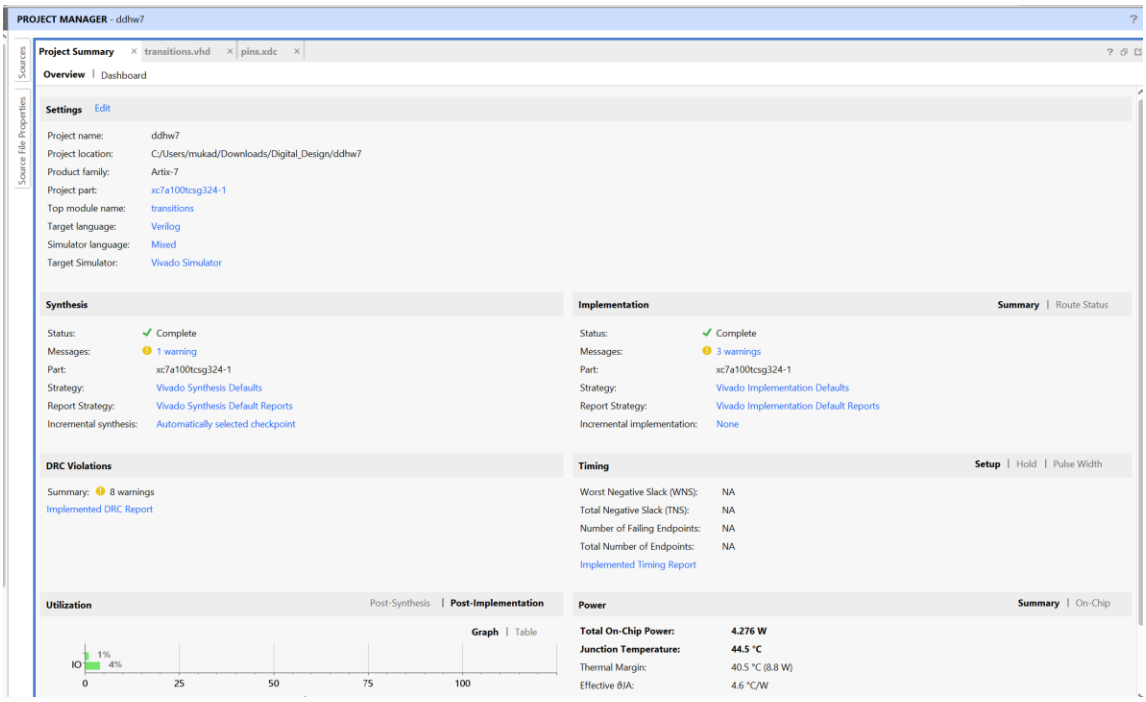
Homework 7

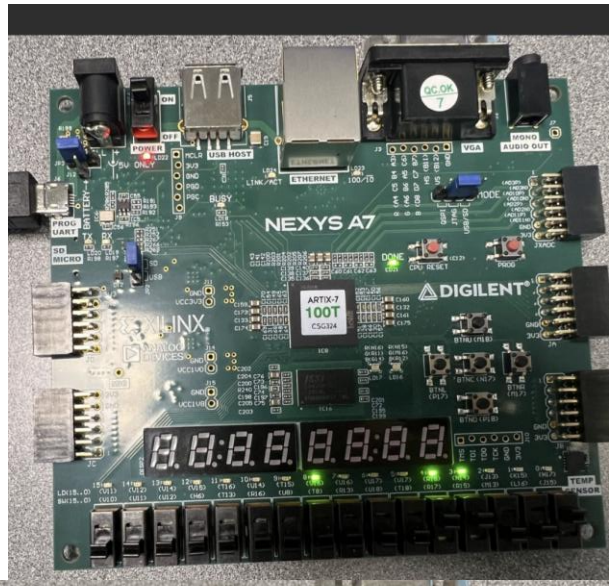
Mack Usmanova

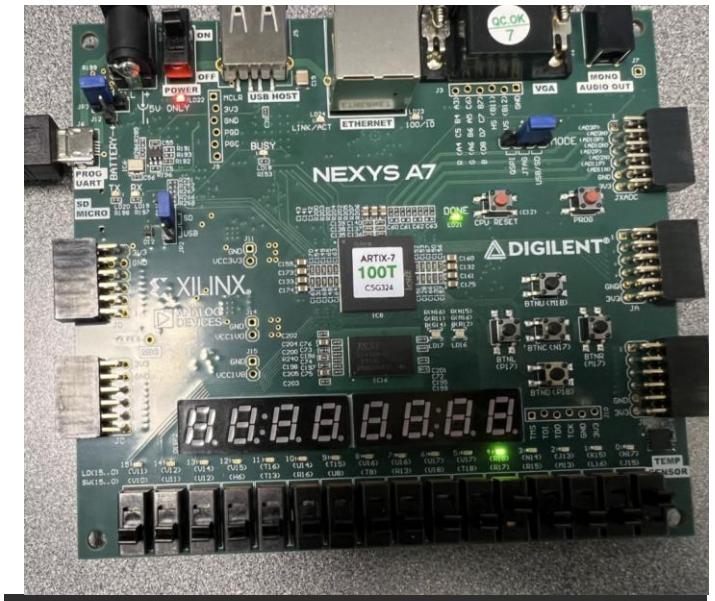
Computer engineer

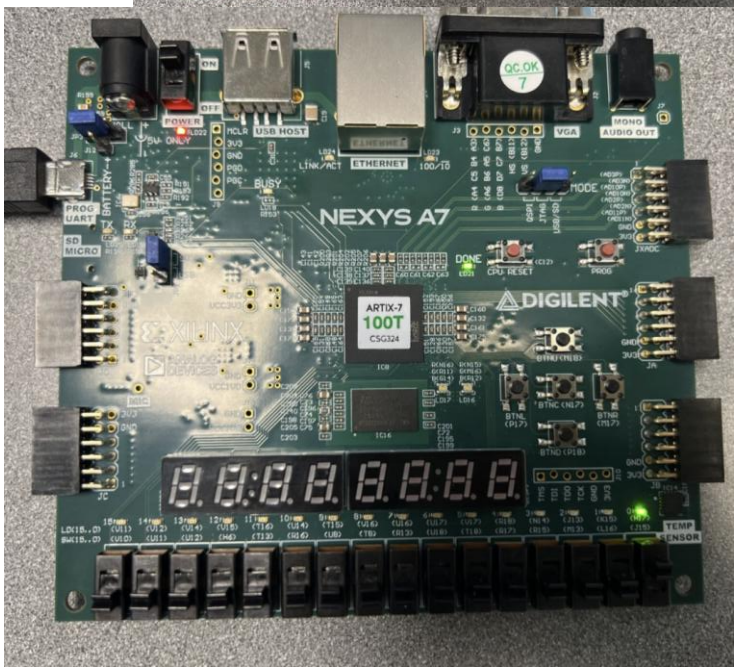


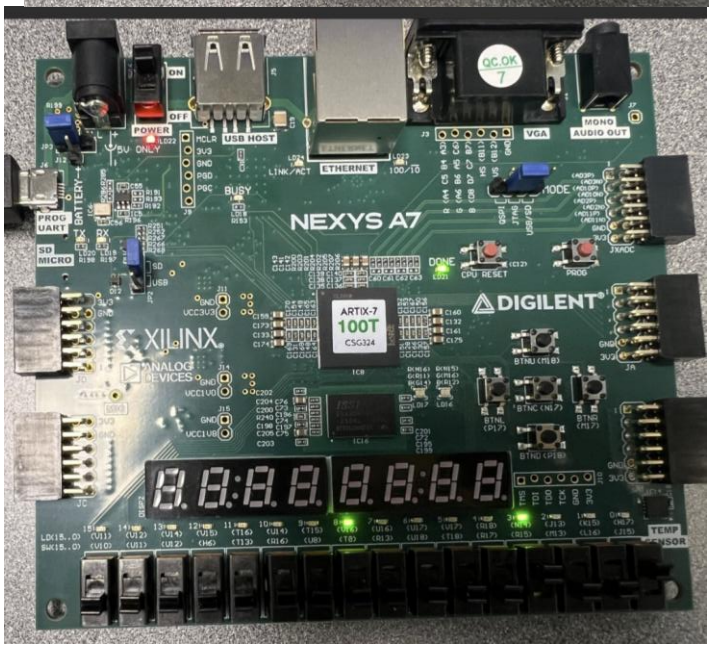
I would like to point out, the reason there seems to be a glow near 650ns is because it is meant to switch to a(000) but it also has the option of switching to g(101), ideally I had hoped it would jump to 000(a), but it seemed to have chosen to do that for only a split second. Although that seems to mess with pattern I expected, in my video, it switched to a (000) as needed. If you see my video, I have a pattern created in a notecard that I followed on my fpga, I followed the same pattern as in the video in this implementation and my outputs are exact according to my expectations. Overall the only issue I seemed to have was figuring out a way to output y1 y2 and y3 as they are “inout”s and I have not much experience with inouts.

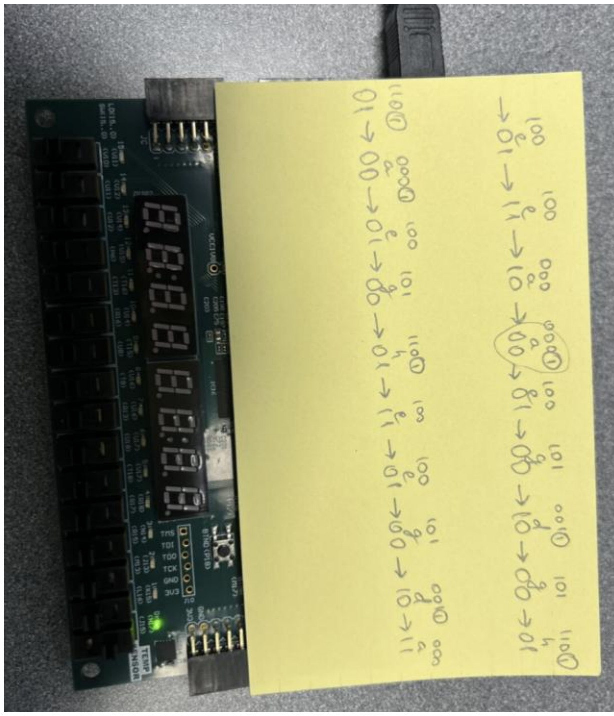
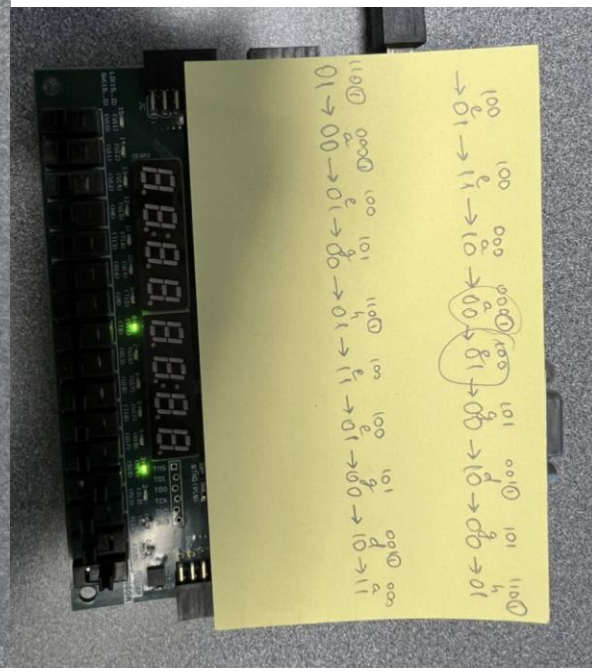
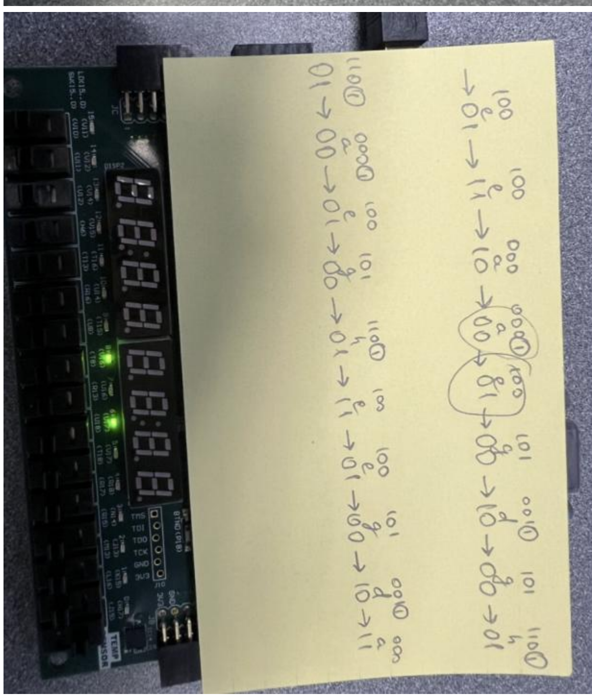
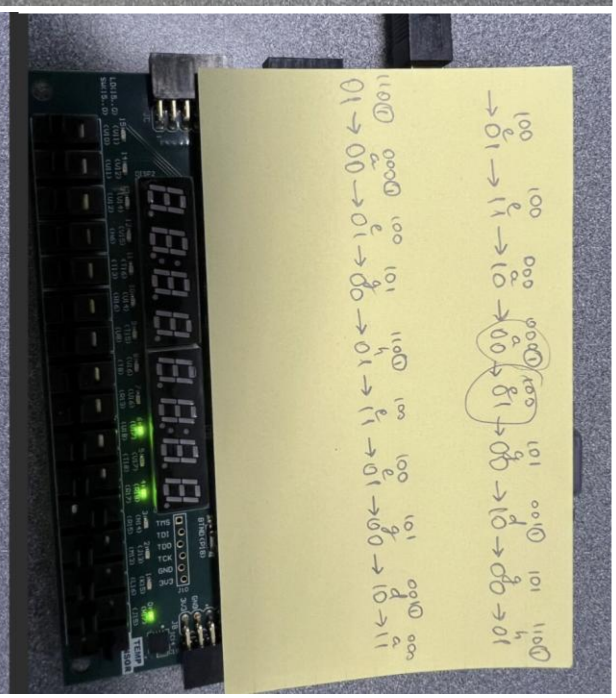


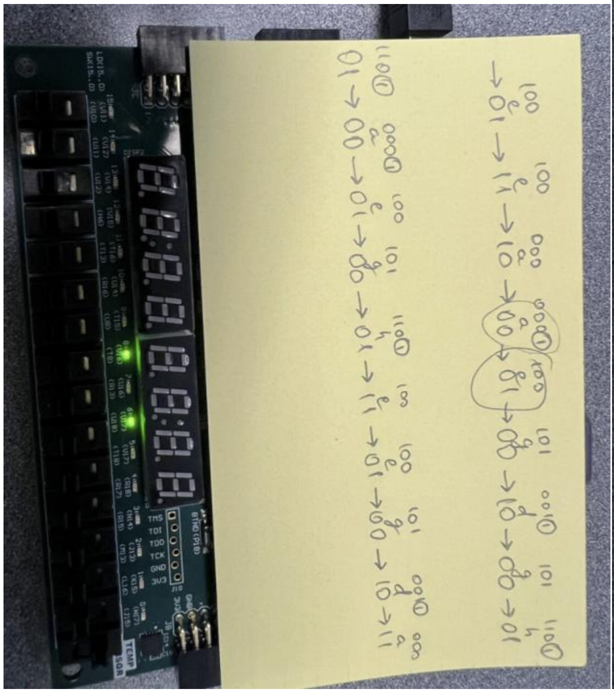

NEXYS A7
DIGILENT
XILINX
ARTIX-7
100T
CSG324

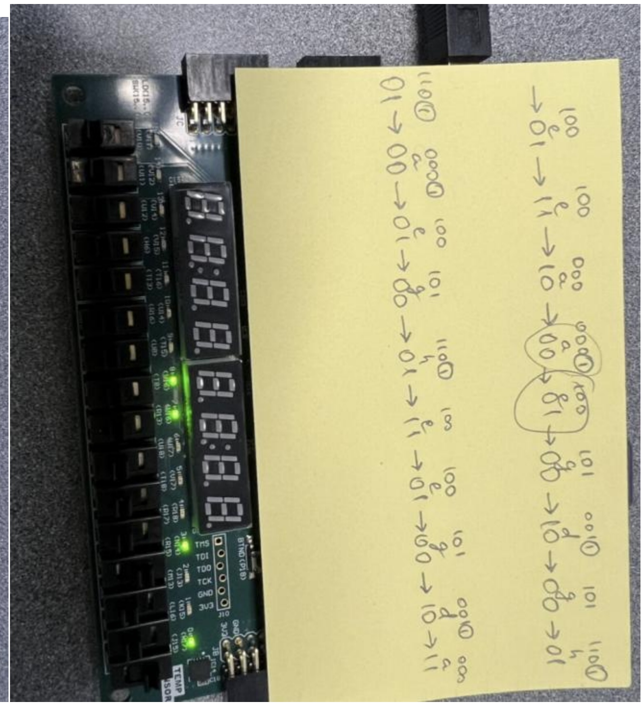

NEXYS A7
DIGILENT
XILINX
ARTIX-7
100T
CSG324

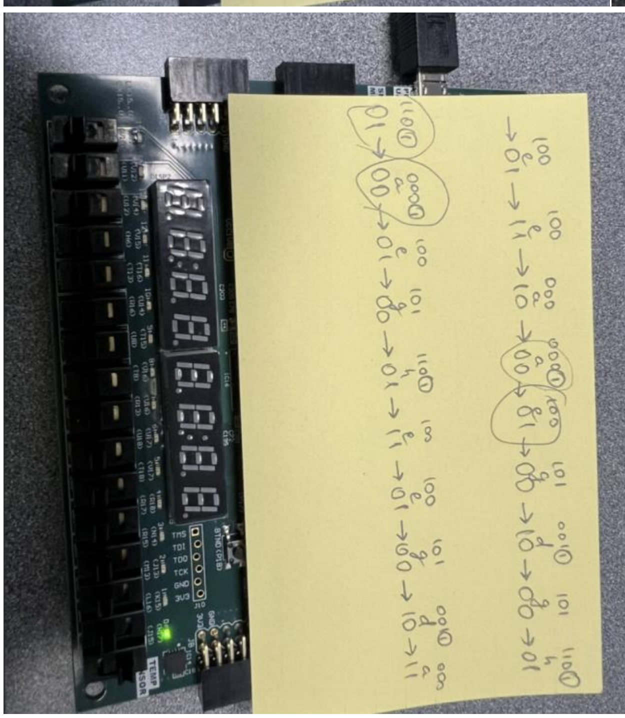

NEXYS A7
DIGILENT
XILINX
ARTIX-7
100T
CSG324

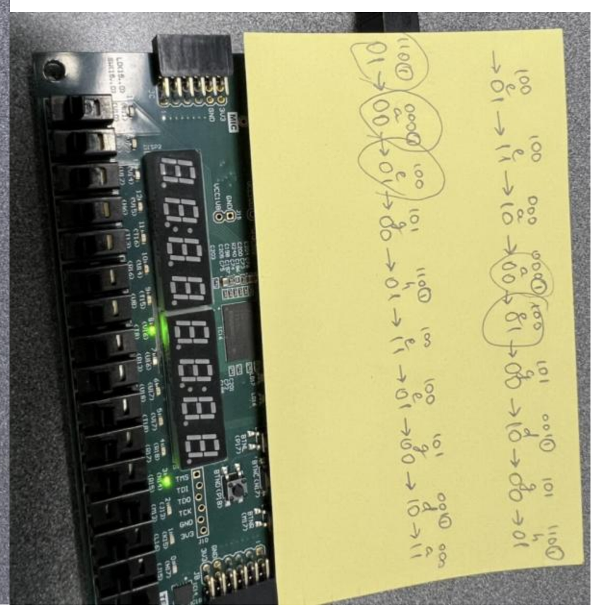

NEXYS A7
DIGILENT
XILINX
ARTIX-7
100T
CSG324

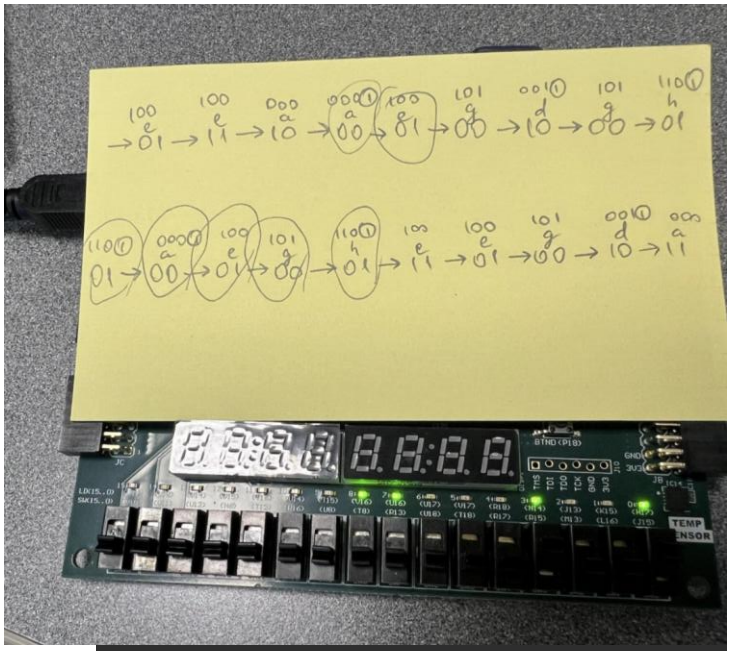





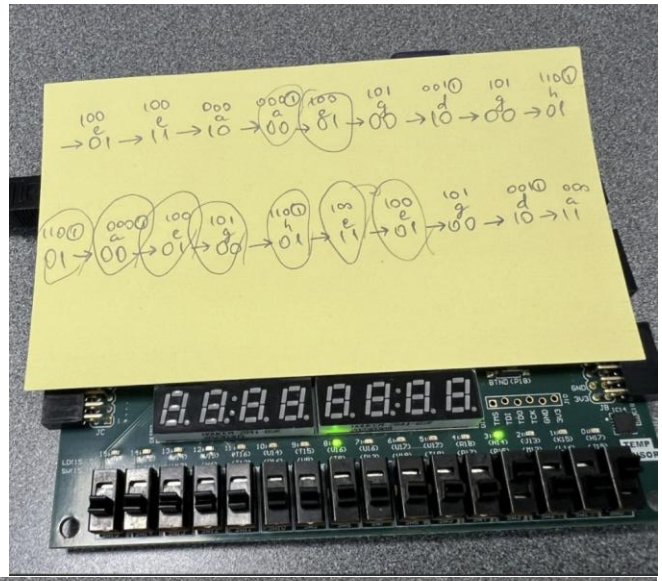


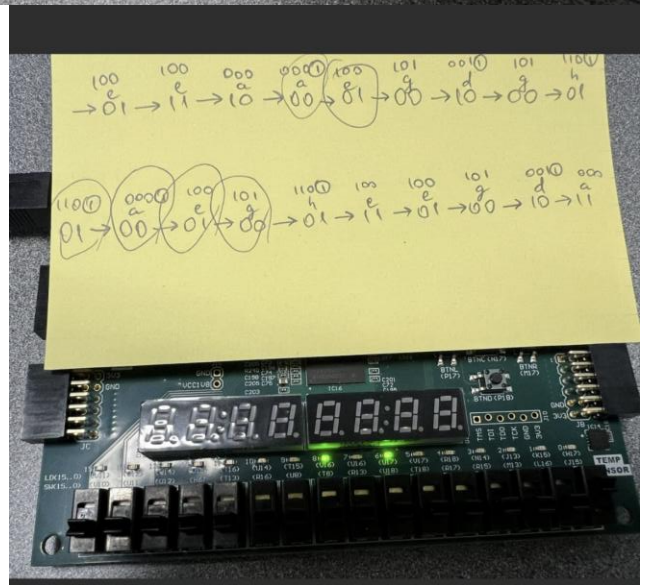


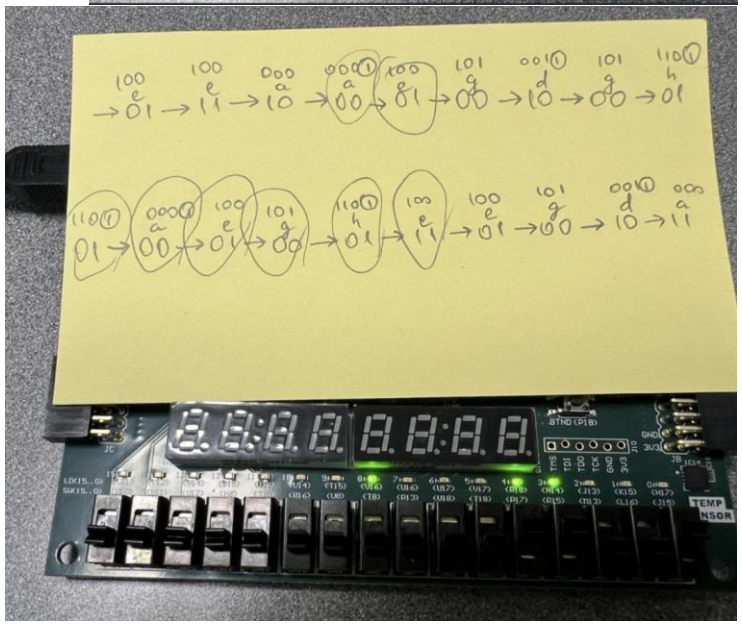


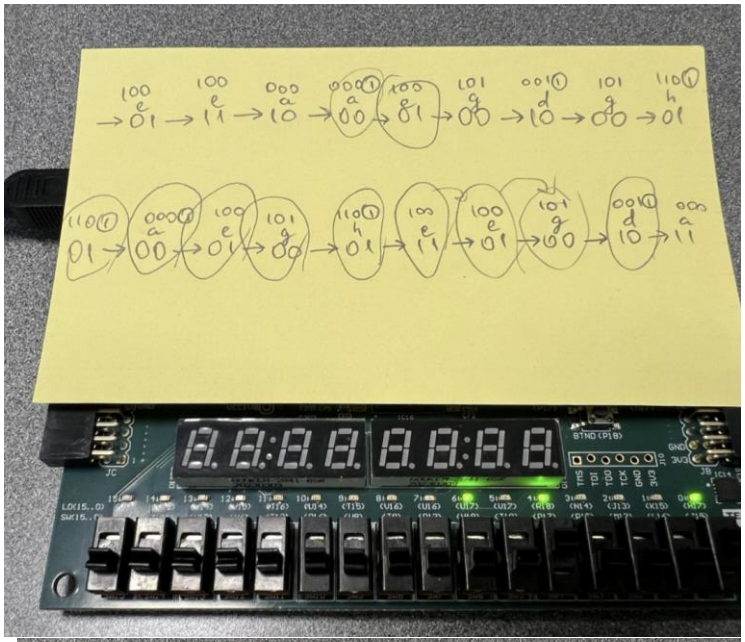


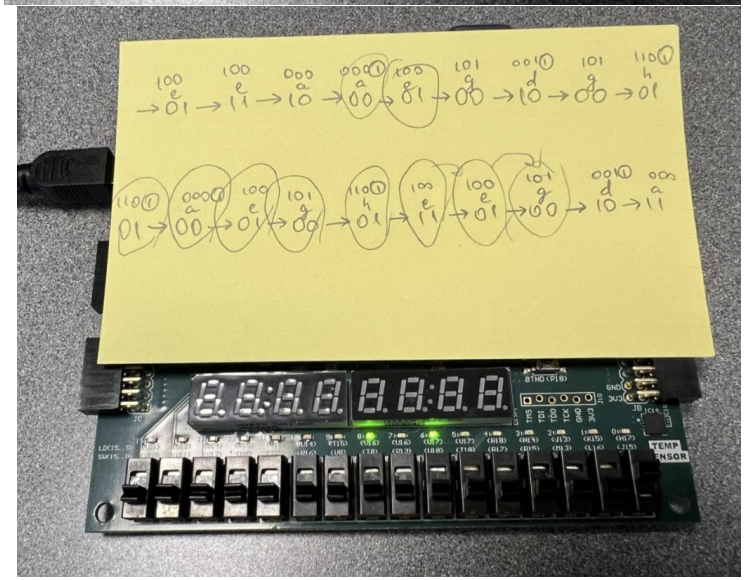


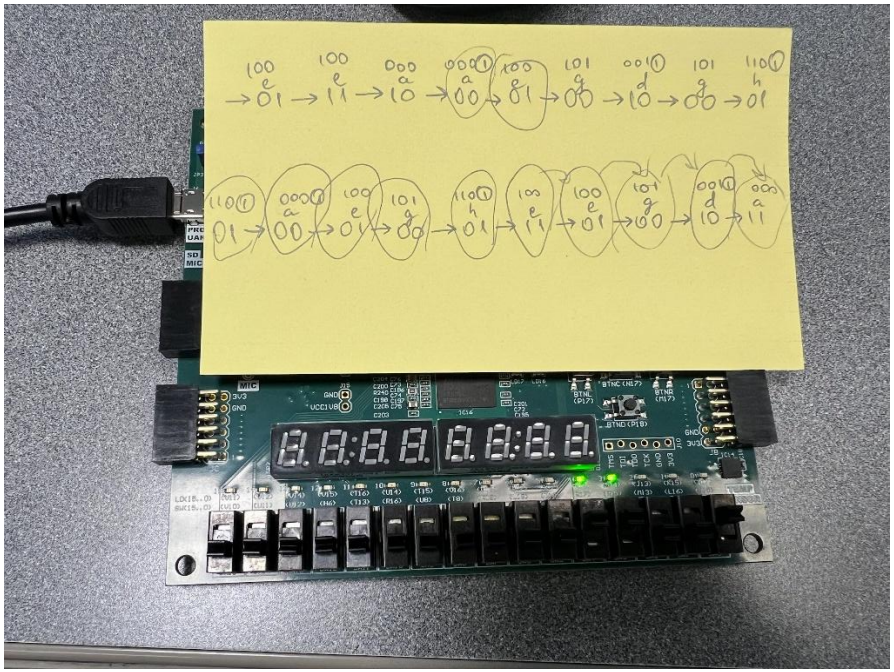












Vhd

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

-- Uncomment the following library declaration if using
-- arithmetic functions with Signed or Unsigned values
--use IEEE.NUMERIC_STD.ALL;

-- Uncomment the following library declaration if instantiating
-- any Xilinx leaf cells in this code.
--library UNISIM;
--use UNISIM.VComponents.all;

entity transitions is
  Port (
        X1, X2, RESET: in std_logic;
        Y1, Y2, Y3: inout std_logic;
        X1_LED, X2_LED, Z: out std_logic
  );
end transitions;

architecture Behavioral of transitions is

begin

     Y1 <= ((y1 and y2 and y3)         as an inout, this variable may have itself in its case,
            or (x2 and y1)
            or (x1 and y1 and y2)
            or (not(x1) and y1 and y3)
            or (not(x1) and not(x2) and not(y1) and y3)
            or (not(x1) and x2 and not(y2) and not(y3))
            or (not(x1) and y1 and not(y2))) and reset;


     Y2 <= ((not(x1) and x2 and y1 and y3)
            or (not(x1) and y1 and y2 and not(y3))
            or (not(x1) and x2 and y1 and y2)) and reset;


     Y3 <= ((not(x2) and y3)
            or (x1 and y2 and y3)
            or (not(x1) and not(y1) and y3)
            or (not(x1) and not(y2) and y3)
            or (not(y1) and y2 and y3)
            or (not(x1) and not(x2) and y1 and not(y2))) and reset;


     Z <= (not(y1) and y3)
            or (y1 and y2 and not(y3))
            or (not(x1) and not(x2) and not(y1))
            or (not(x1) and not(x2) and y2 and not(y3));

x1_led <= x1;
x2_led <= x2;


end Behavioral;
```

Xdc

```
##Switches
set_property -dict { PACKAGE_PIN J15 IOSTANDARD LVCMOS33 } [get_ports { RESET }];
#IO_L24N_T3_RS0_15 Sch=sw[0]
set_property -dict { PACKAGE_PIN R15 IOSTANDARD LVCMOS33 } [get_ports { X2 }];
#IO_L13N_T2_MRCC_14 Sch=sw[3]
```

```
set_property -dict { PACKAGE_PIN R17 IOSTANDARD LVCMOS33 } [get_ports { X1 }];
#IO_L12N_T1_MRCC_14 Sch=sw[4]

set_property -dict { PACKAGE_PIN H17 IOSTANDARD LVCMOS33 } [get_ports { Z }];
#IO_L18P_T2_A24_15 Sch=led[0]
set_property -dict { PACKAGE_PIN N14 IOSTANDARD LVCMOS33 } [get_ports { X2_LED }];
#IO_L8P_T1_D11_14 Sch=led[3]
set_property -dict { PACKAGE_PIN R18 IOSTANDARD LVCMOS33 } [get_ports { X1_LED }];
#IO_L7P_T1_D09_14 Sch=led[4]
set_property -dict { PACKAGE_PIN U17 IOSTANDARD LVCMOS33 } [get_ports { Y3 }];
#IO_L17P_T2_A14_D30_14 Sch=led[6]
set_property -dict { PACKAGE_PIN U16 IOSTANDARD LVCMOS33 } [get_ports { Y2 }];
#IO_L18P_T2_A12_D28_14 Sch=led[7]
set_property -dict { PACKAGE_PIN V16 IOSTANDARD LVCMOS33 } [get_ports { Y1 }];
#IO_L16N_T2_A15_D31_14 Sch=led[8]

set_property ALLOW_COMBINATORIAL_LOOPS TRUE [get_nets Y1*];
set_property ALLOW_COMBINATORIAL_LOOPS TRUE [get_nets Y2*];
set_property ALLOW_COMBINATORIAL_LOOPS TRUE [get_nets Y3*];
```

Tcl

```
#inputs:
# X1, X2, RESET STD_LOGIC

#outputs:
# Y1, Y2, Y3, Z, X1_LED, X2_LED STD_LOGIC

restart

# apply active low RESET
add_force RESET {0 0ns}

# initialize inputs
add_force X1 {0 0ns}
add_force X2 {0 0ns}

# wait out 150ns
run 150ns

# set active low RESET back to inactive state
add_force RESET {1 0ns}
run 50ns

# go to e
add_force X2 {1 0ns}
run 50ns

# go to e
add_force X1 {1 0ns}
run 50ns

# go to a
add_force X2 {0 0ns}
run 50ns

# go to a
add_force X1 {0 0ns}
run 50ns

# go to e
add_force X2 {1 0ns}
run 50ns

# go to g
add_force X2 {0 0ns}
run 50ns
```

```
# go to d
add_force X1 {1 0ns}
run 50ns

# go to g
add_force X1 {0 0ns}
run 50ns

# go to h
add_force X2 {1 0ns}
run 50ns

# go to a
add_force X2 {0 0ns}
run 50ns

# go to e
add_force X2 {1 0ns}
run 50ns

# go to g
add_force X2 {0 0ns}
run 50ns

# go to h
add_force X2 {1 0ns}
run 50ns

# go to e
add_force X1 {1 0ns}
run 50ns

# go to e
add_force X1 {0 0ns}
run 50ns

# go to g
add_force X2 {0 0ns}
run 50ns

# go to d
add_force X1 {1 0ns}
run 50ns

# go to a
add_force X2 {1 0ns}
run 100ns
```