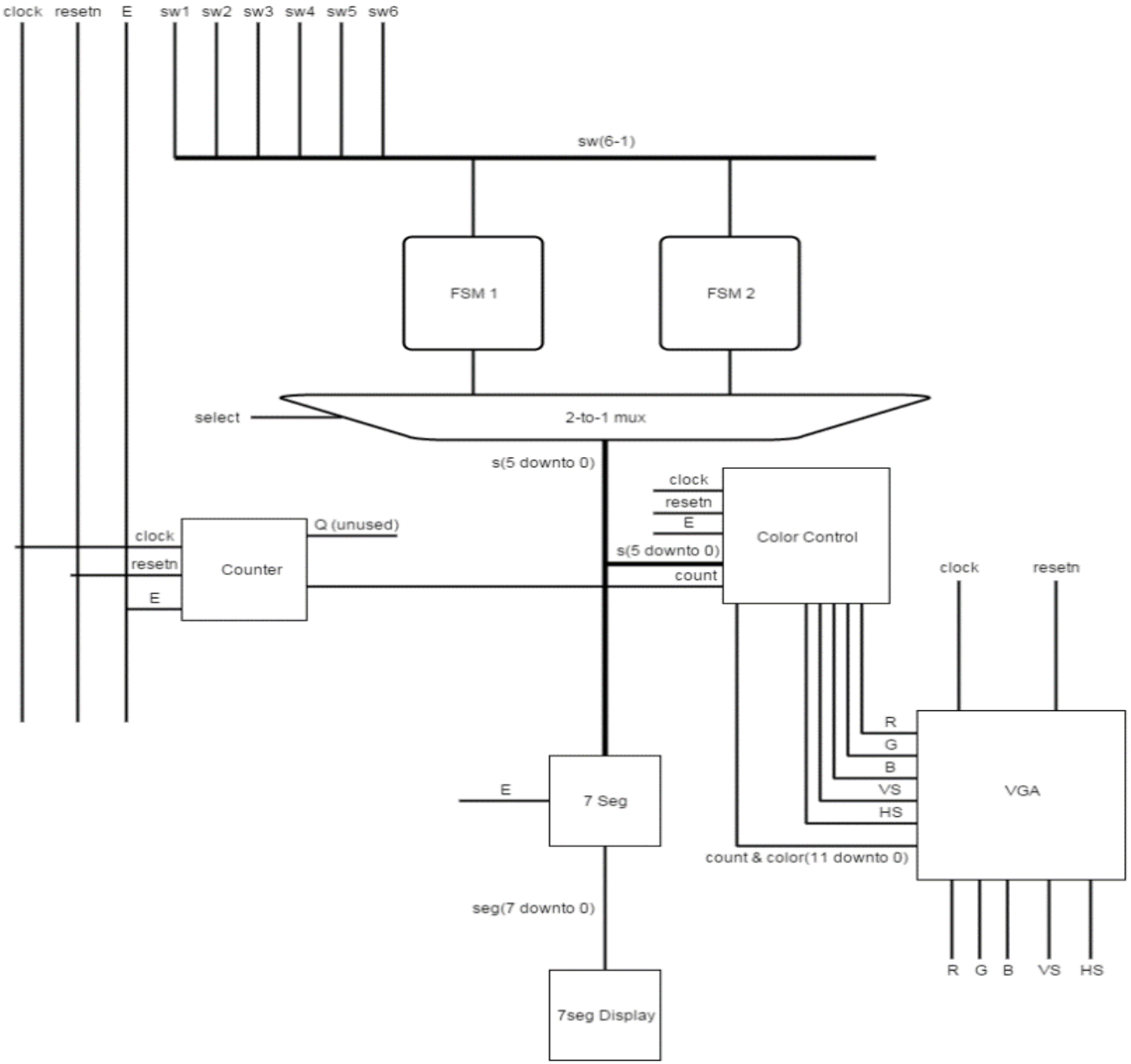
Brain Drain

By: Zarif Ghazi, Max Manley, Robert McInerney

Game Functionality

- The objective of the game is to test your memory.
- In the game, players will be initially shown a diagram of six switches, and their corresponding colors
- The players have 20 seconds to remember the colors
- The game commences and progresses through 8 different levels of varying color combinations
- The player then has to remember which switch to push up based on the colors displayed on the VGA

Top Level Architecture



Color Selection

- game_color receives the two bit output from the counter (count), which creates one second increments and combines it with the six bit output s_hold from the multiplexer
- The color sequences, for each state, are then assigned to each eight bit game_color signal

```
game_color <= count & s_hold;
   with game_color select
      color <=  "101000001100" when "00000000",--fsm1 s1
                "101000001100" when "01000000",
                "101000001100" when "10000000",
                "101000001100" when "11000000",

                "000001100000" when "00000010",--fsm1 s2
                "000001100000" when "01000010",
                "101000000000" when "10000010",
                "101000000000" when "11000010",

                "101000001100" when "00000011",--fsm1 s3
                "101000001100" when "01000011",
                "111111111111" when "10000011",
                "111111111111" when "11000011",

                "000000001100" when "00000100",--fsm1 s4
                "000000001100" when "01000100",
                "000000000000" when "10000100",
                "000000000000" when "11000100",
```

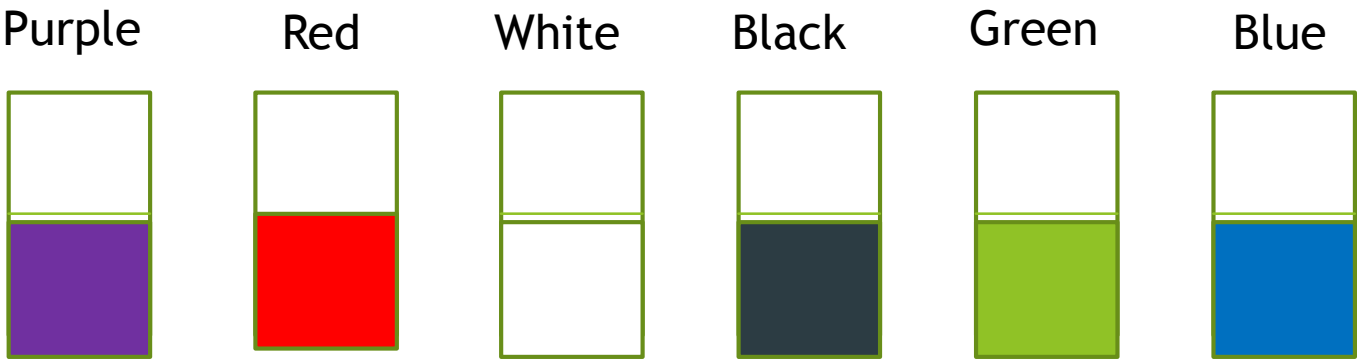
FSM Description

- The FSMs control the sequence of switch inputs. When the correct sequence is detected, it moves to the next state
- The value of s in FSM1 and FSM2 were initially the same, but in order for the color controller to tell the difference between S1 in FSM1 and S1 in FSM2, each s needed to be unique
- Only three states are shown here, each FSM has eight states (or levels) and a ninth state for game over

```
Transitions: process (resetn, clock, switch1,switch2,switch3,switch4,switch5,switch6)
begin
   if resetn = '0' then
      x <= S1;
   elsif (clock'event and clock = '1') then     -- Sequence = 1,1-2,1-3,1-6, 4-5-6-2,2-3-6
      case x is
         when S1 =>
            if switch1 = '1' then x <= S2; else x <= S1; end if;
         when S2 =>
            if switch5 = '1' then if switch2='1' then x <= S3; else x <= S2; end if;end if;
         when S3 =>
            if switch1 = '1' then if switch3='1' then x <= S4; else x <= S3; end if;end if;
```

```
Outputs: process (x)
begin
   case x is
      when S1 => s <= "000000";
      when S2 => s <= "000010";
      when S3 => s <= "000011";
```

Demonstration



Problems/Improvements

- **Problems:**
- If all six switches are all pushed up at the same time, the game immediately enters the game over state (S9)
- The game has no lose state

- **Improvements:**
- Add a lose state where if the wrong switches are pushed, the game is over
- Add more games (FSMs) and more levels (states) so the game is more robust
- Varying difficulty levels

THANKS FOR TUNING IN!