Question 1

Designers, reviewers, review leader, scribe. All relevant design documents.

Question 2

Preliminary Design Review. Systems Overview - Sensor sub-sytem - Mechanical - Comms - Power - Sofyware - Integration and test - Requirements compliance - Management

Question 3

in – input
out – output
inout - bidirectional

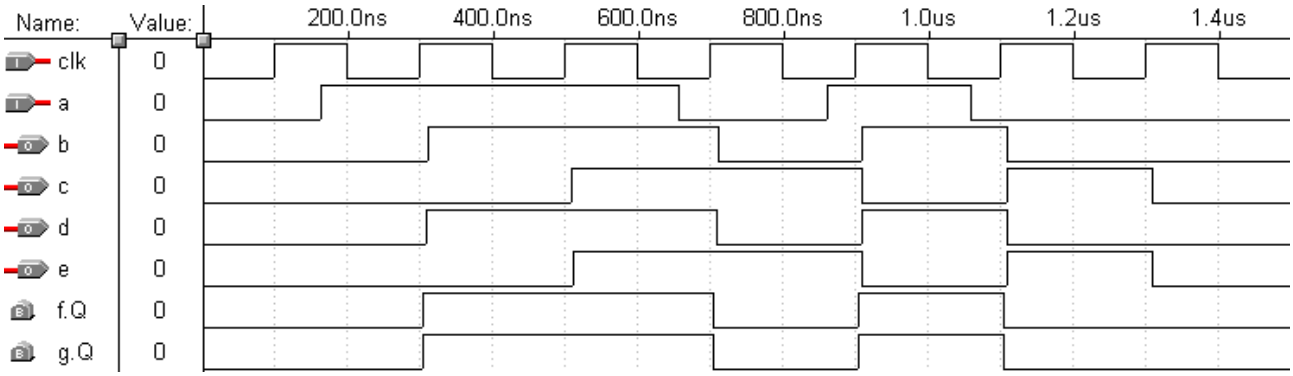
Question 4

When we use synchronous design constructs, such as processes.

Question 5

State machines in VHDL use synchronous design – meaning that all signals (especially the input signals) relevant to the FSM should be synchronized to the system clock. This prevents spurious state changes caused by race conditions in the next state logic.

Question 6:



Question 7

```
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;

ENTITY vraag3 IS
   PORT( clk          : IN STD_LOGIC;
         slow_clock   : BUFFER STD_LOGIC );
END vraag3;

ARCHITECTURE logic OF vraag3 IS
   constant divider : integer range 0 to 255 := 55;

BEGIN
```

```
   PROCESS( clk )
      VARIABLE counter : integer range 0 to 255;
   BEGIN

      IF rising_edge(clk) then

         IF counter = divider THEN
            counter := 0;
            slow_clock <= not slow_clock;
         ELSE
            counter := counter + 1;
         END IF;

      END IF;

   END PROCESS;
END ARCHITECTURE logic;
```

Question 8:

```
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;

ENTITY vraag6 IS
  PORT( clk, start, reset  : IN STD_LOGIC;
        button           : IN STD_LOGIC_VECTOR( 3 downto 0 );
        winner           : OUT STD_LOGIC_VECTOR( 3 downto 0 ) );
END vraag6;

ARCHITECTURE logic OF vraag6 IS
  type state_type is (wait_start, test, wait_reset);
  signal state : state_type;
BEGIN

  PROCESS(clk, start, reset, button) is
  BEGIN
    IF rising_edge(clk) THEN
      CASE state is

        WHEN wait_start =>
          winner <= "0000";
          IF start = '1' THEN
            state <= test;
          END IF;

        WHEN test =>
          IF button /= "1111" then
            winner <= not button;
            state <= wait_reset;
          END IF;
```

```
        WHEN wait_reset =>
          IF reset = '1' then
            state <= wait_start;
          END IF;
      END CASE;
    END IF;
  END PROCESS;

END ARCHITECTURE logic;
```