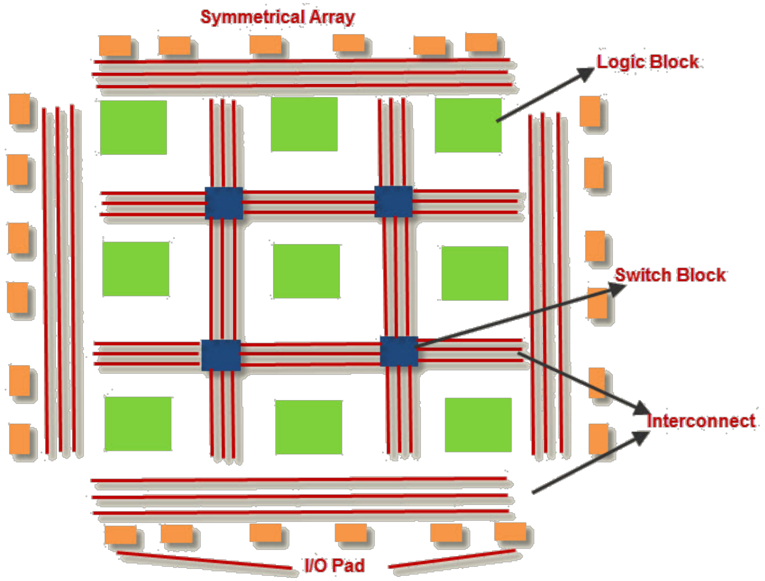
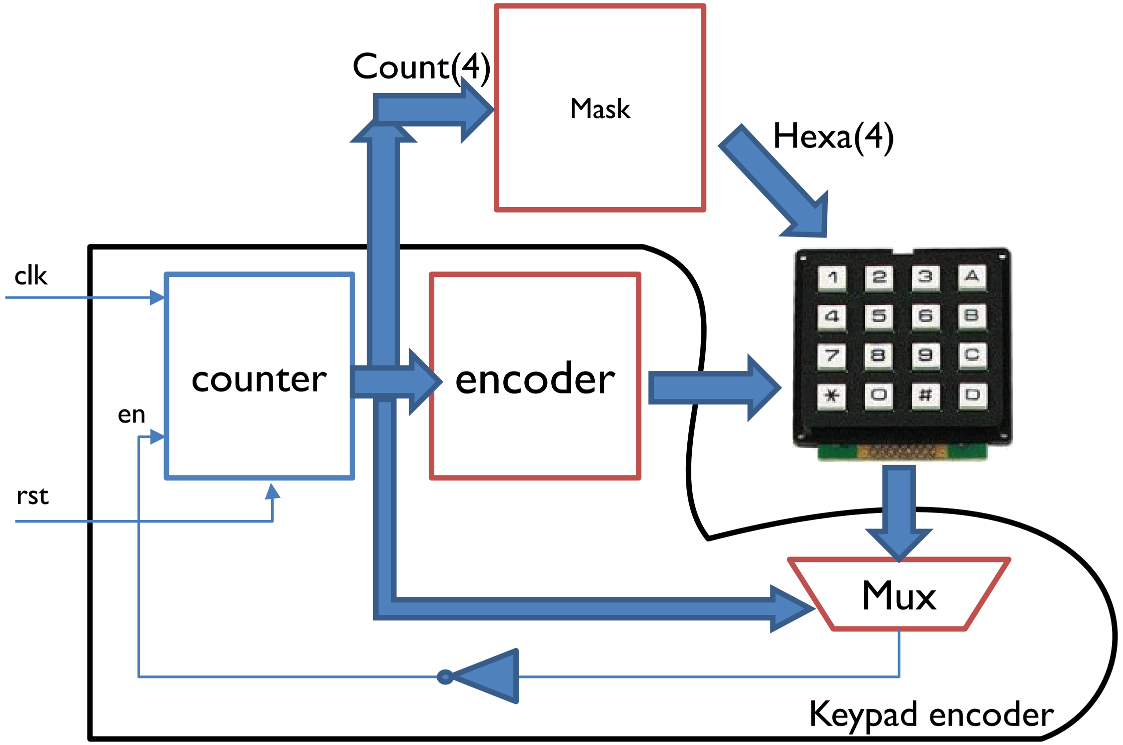
Lecture 3: Registers and FSMs

By: Emad Samuel Malki Ebeid

Summary of lecture 2





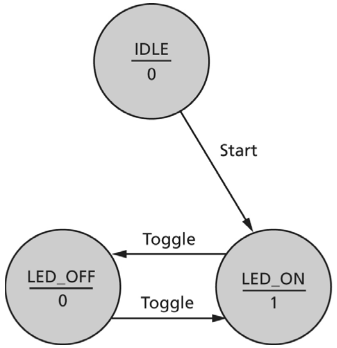
Look to the blackboard notes

- Latches & Flip flops
- Parallel & shift registers
- FSMs

  - debouncer

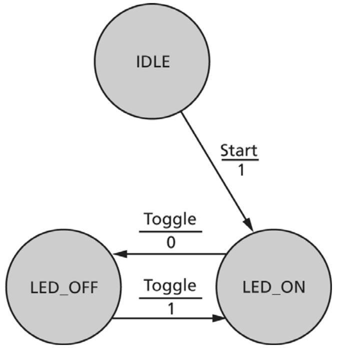
FSM

Moore State Machine



The outputs of a Moore state machine depend **only on the present state**. The <u>outputs</u> are written **only** when the state changes (on the clock edge).

Mealy State Machine

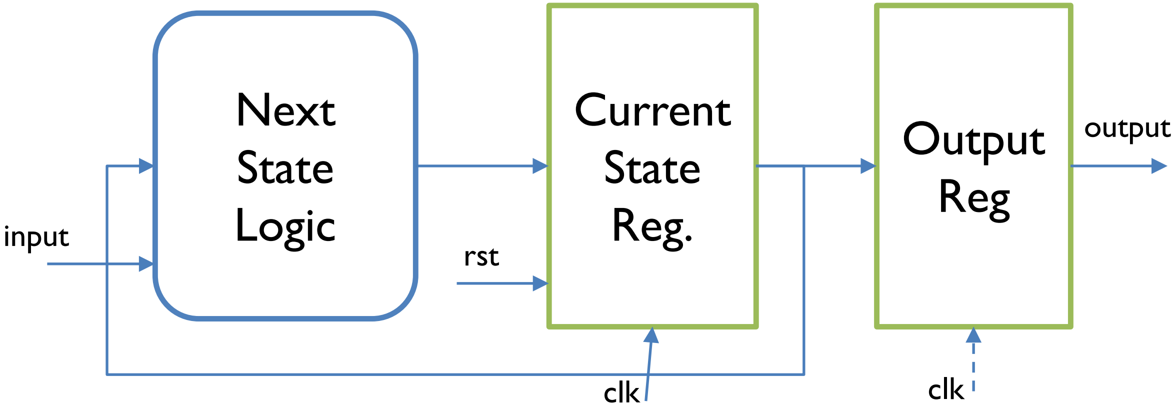


The outputs of a Mealy state machine depend on **both the inputs and the current state**. When the <u>inputs</u> change, the outputs are updated **without** waiting for a clock edge.

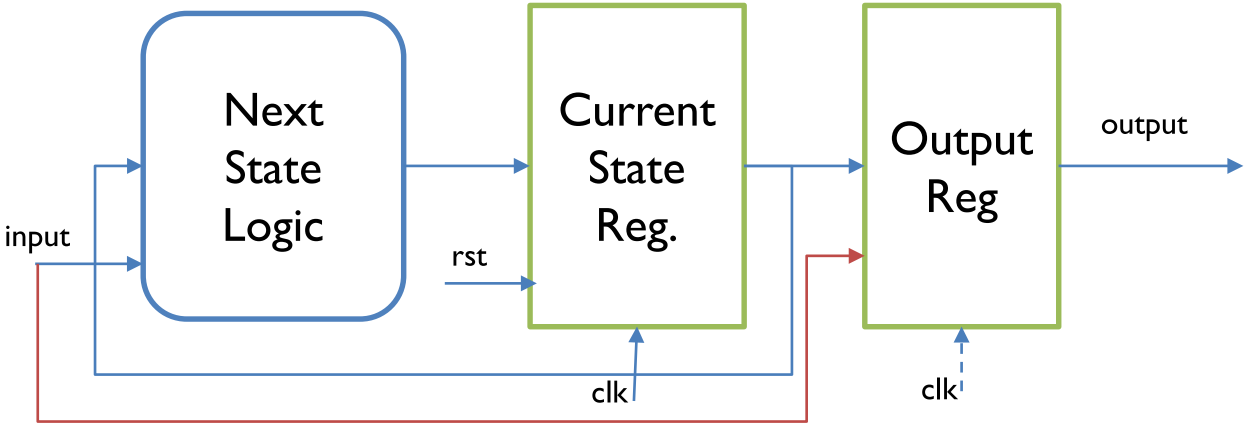
Because modern designs are generally synchronous, the **Moore** option tends to be preferred – *Finite State Machine in Hardware* book by Volnei A. Pedroni

FSM in hardware

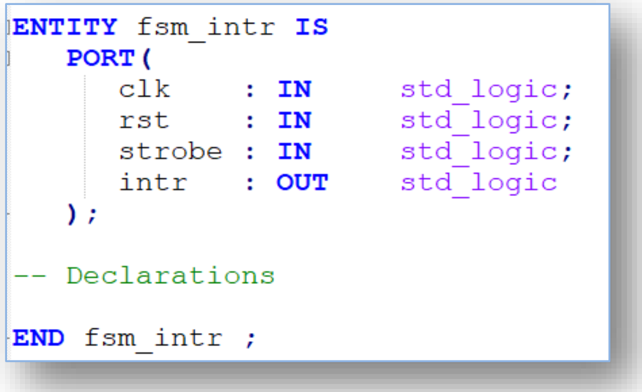
Moore



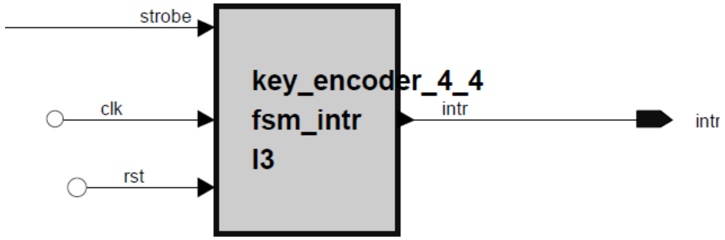
Mealy

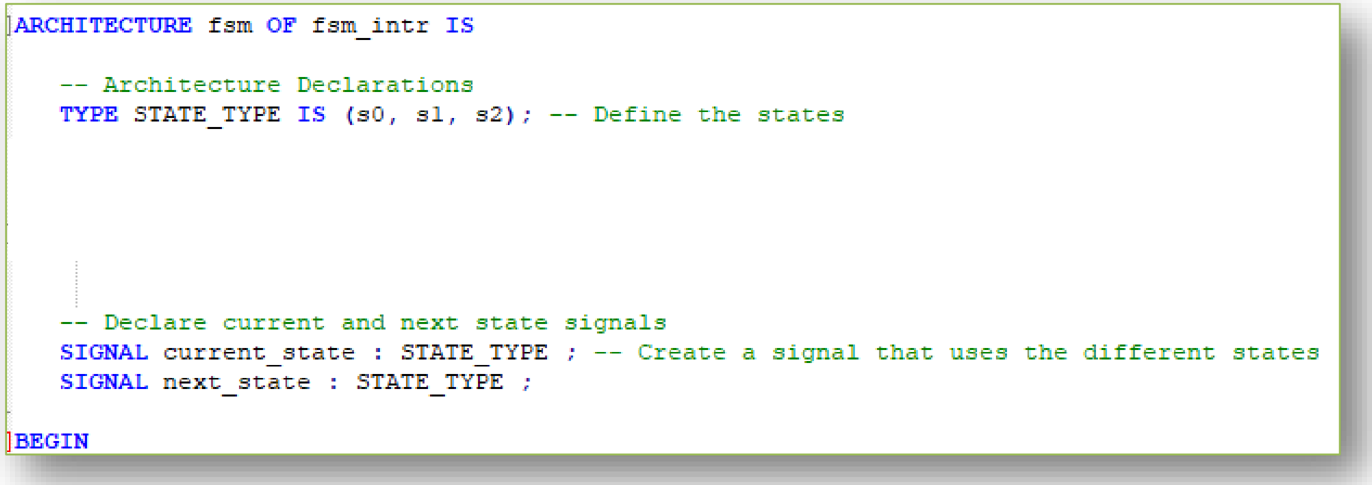


FSM examples: debouncer

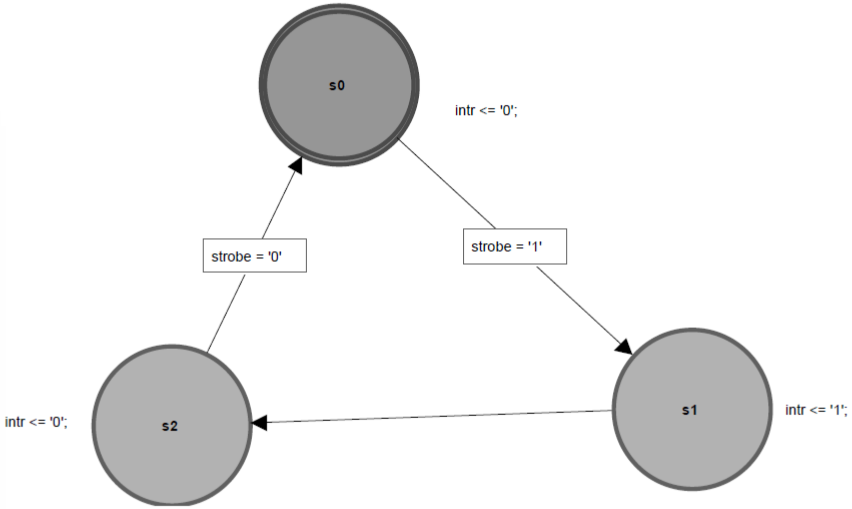


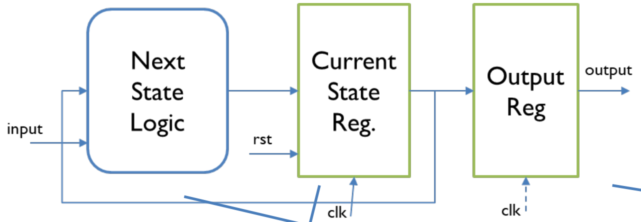
```
ENTITY fsm_intr IS
    PORT(
        clk    : IN     std_logic;
        rst    : IN     std_logic;
        strobe : IN     std_logic;
        intr   : OUT    std_logic
    );

-- Declarations

END fsm_intr ;
```

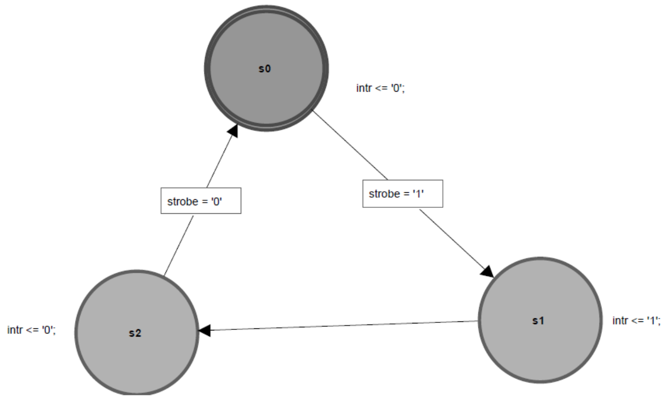


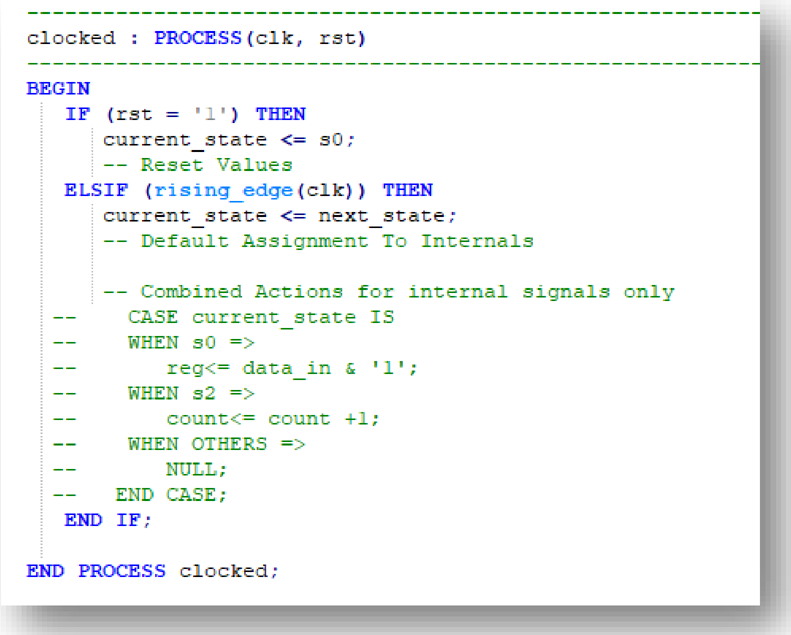


```
ARCHITECTURE fsm OF fsm_intr IS

   -- Architecture Declarations
   TYPE STATE_TYPE IS (s0, s1, s2); -- Define the states




   -- Declare current and next state signals
   SIGNAL current_state : STATE_TYPE ; -- Create a signal that uses the different states
   SIGNAL next_state : STATE_TYPE ;

BEGIN
```

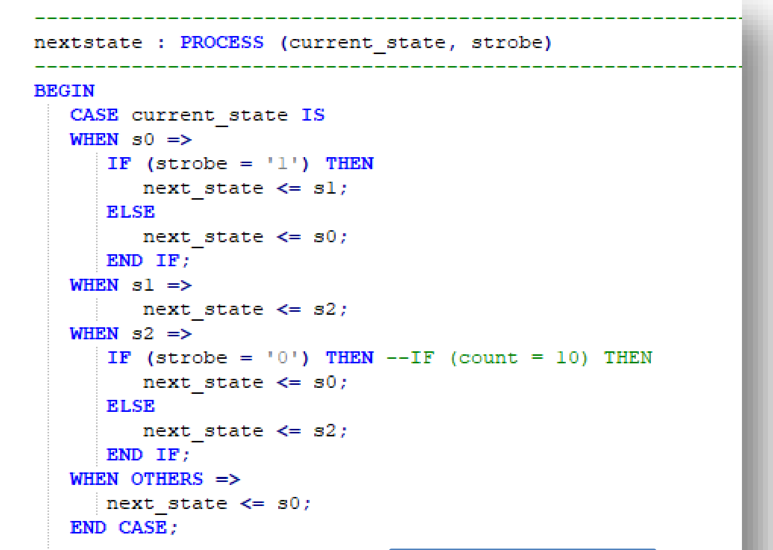




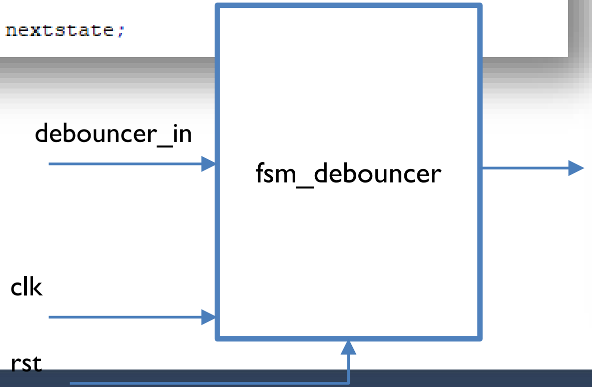


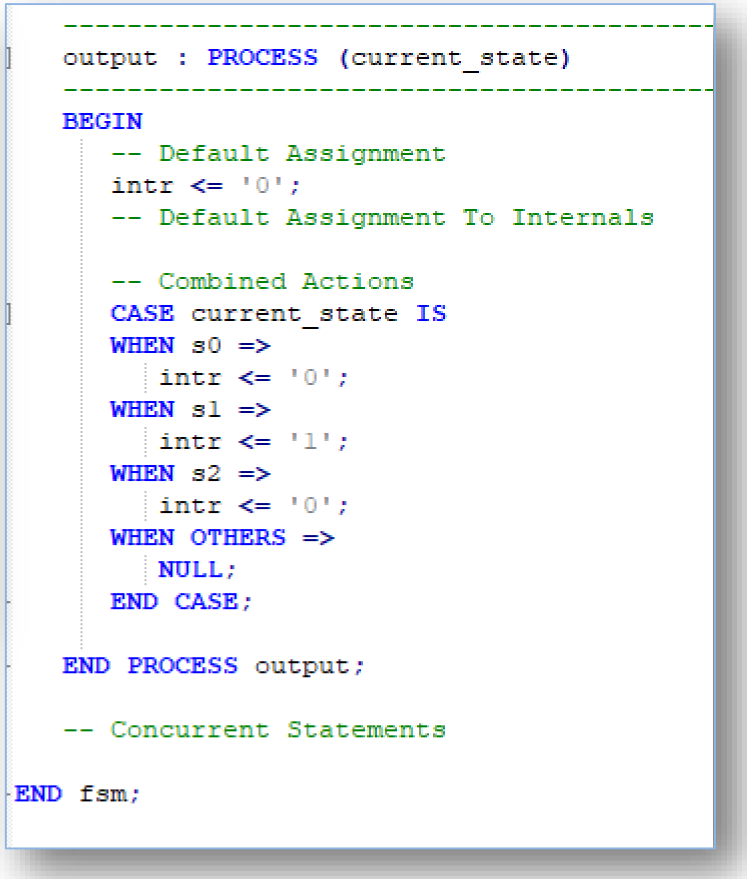


```
-----------------------------------------------------------
clocked : PROCESS(clk, rst)
-----------------------------------------------------------
BEGIN
   IF (rst = '1') THEN
      current_state <= s0;
      -- Reset Values
   ELSIF (rising_edge(clk)) THEN
      current_state <= next_state;
      -- Default Assignment To Internals

      -- Combined Actions for internal signals only
--      CASE current_state IS
--      WHEN s0 =>
--         reg<= data_in & '1';
--      WHEN s2 =>
--         count<= count +1;
--      WHEN OTHERS =>
--         NULL;
--     END CASE;
   END IF;

END PROCESS clocked;
```



```
-----------------------------------------------------------
nextstate : PROCESS (current_state, strobe)
-----------------------------------------------------------
BEGIN
   CASE current_state IS
   WHEN s0 =>
      IF (strobe = '1') THEN
         next_state <= s1;
      ELSE
         next_state <= s0;
      END IF;
   WHEN s1 =>
         next_state <= s2;
   WHEN s2 =>
      IF (strobe = '0') THEN --IF (count = 10) THEN
         next_state <= s0;
      ELSE
         next_state <= s2;
      END IF;
   WHEN OTHERS =>
      next_state <= s0;
   END CASE;

END PROCESS nextstate;
```

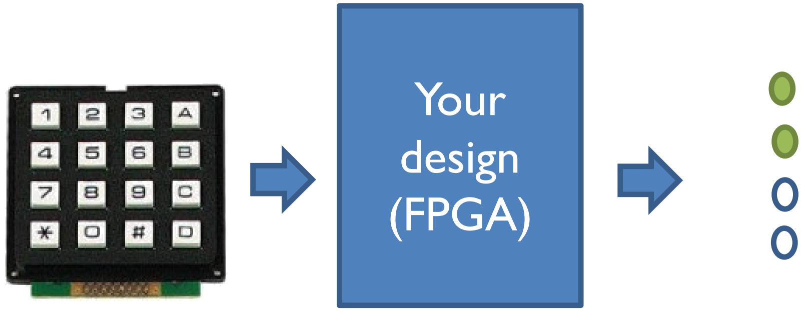




```
-----------------------------------------------------
output : PROCESS (current_state)
-----------------------------------------------------
BEGIN
   -- Default Assignment
   intr <= '0';
   -- Default Assignment To Internals

   -- Combined Actions
   CASE current_state IS
   WHEN s0 =>
      intr <= '0';
   WHEN s1 =>
      intr <= '1';
   WHEN s2 =>
      intr <= '0';
   WHEN OTHERS =>
      NULL;
   END CASE;

END PROCESS output;

-- Concurrent Statements

END fsm;
```

Lab3: FPGA configuration

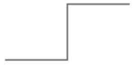
- Design a circuit in VHDL to display a pressed key from the provided 4x4 keypad.
  - Use FPGA LEDs to display the pressed key.
  - Simulate the overall circuit when number **3** is pressed (0011).
  - Configure the final design on the FPGA.

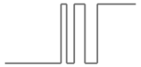


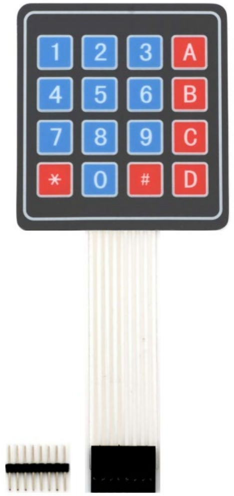
Your design (FPGA)

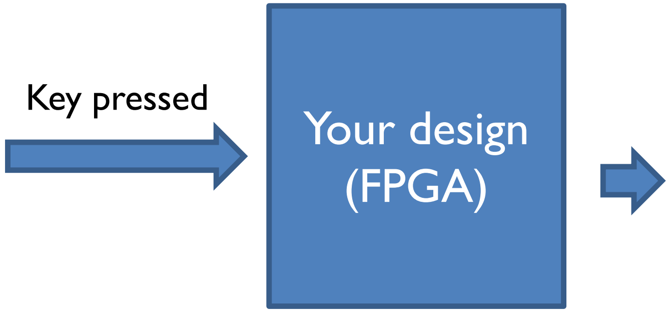
Journal 1 – delivery 27/09/2020 23:59

- Implement the keypad circuit in FPGA to display a constant pressed key (0011)
  - Use 7-segment display to display the pressed key. You can get one from the component lab (remember to add current limiting resistors (220Ω) as shown in the figure))
  - Use debouncer to avoid the bouncing effect of the contact (push button) before reaching the stable state.
  - The report should include the overall design (Vivado snapshoot), simulation results and a pnoto of the final implementation. Comment on the results.


Expected State


Current Output from Push Button





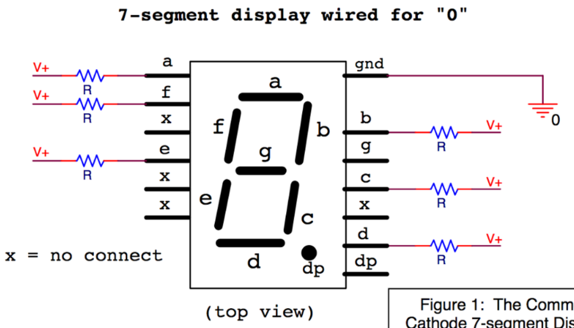


Figure 1: The Common Cathode 7-segment Display