# Electronic Voting Machine (EVM)

- let us design a EVM - 10 candidate vote counter
- synch vote counter that tallies votes for candidates 0-9.
- on each rising clk edge, when not in reset, module examines the 4-bit vote i/p and increments the corresponding 8-bit counter (count 0 ... count 9)
- Reset clears all counters.
- so its a single-cycle per-vote counting and no debouncing & no-duplicating protect "
- code structure

⇒ ports : clk, rst (active high), vote [3:0] input, outputs count 0 ... count 9 (each 8 bit)

⇒ Sequential block :-

```
always @ (posedge clk or posedge rst)
  if (rst) begin
    count0 <= 0;
    count1 <= 0;
      :
    count9 <= 0;
  end
  else begin
    case (vote)
      4'd0 : count0 <= count0 + 1;
      4'd1 : count1 <= count1 + 1;
        :        :        :
      4'd9 : count9 <= count9 + 1;
    endcase
```

case X <= 0

increment when asserted.

- Behavioural code

⇒ Each clk with valid vote value increments candidate's counter once

⇒ If vote is held steady across multiple clks, counter keeps incrementing each clk

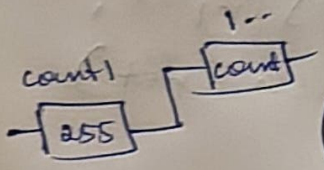
imp

⇒ Clock are 8-bit here - max 255 votes per candidate; will wrap on overflow

(for high population vote

① More bit counter can be used

② OR a trigger counter can also be added (such that whenever the counter reaches 255 then it will trigger a counter once)




↳ cascaded counter look

⇒ No i/p validation, debouncing, authentication or vote acknowledgement - All this are minimal counting core