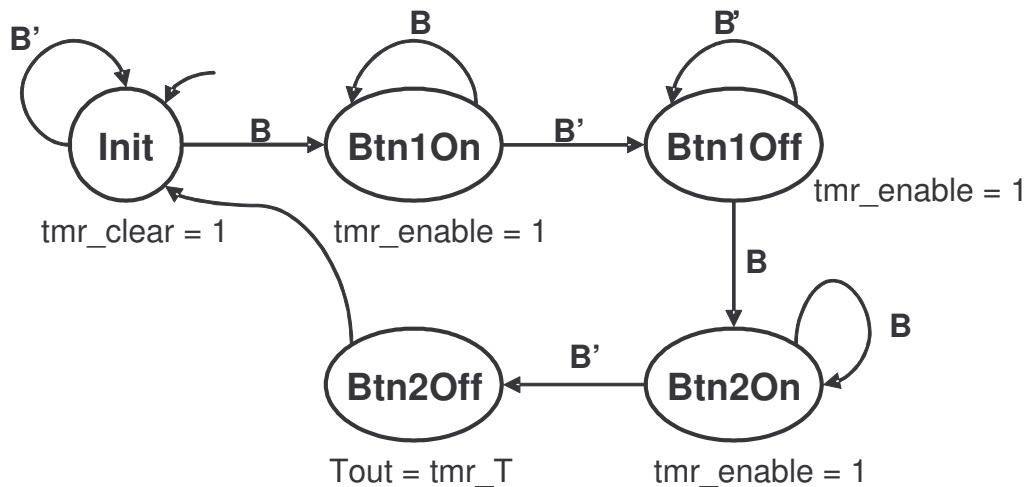


1. Create a system that measures the milliseconds between two successive button presses (detected as input $B=1$, which may last more than one clock cycle), and writes the measured value to an output $Tout$. IGNORE bounce issues.

- (a) Describe the system's behavior using an FSM. Assume access to a timer with control inputs tmr_clear (clears timer to 0 when 1), and tmr_enable (enables the timer to count when 1), and an output tmr_T (the timer's present value, in milliseconds).



- (b) Capture the system's behavior using a single VHDL process. Assume access to the same timer as in (a). TO SAVE SPACE, Just show the first state and last state, and use "... " for The states in between.

```

process (reset, clock, B)
  type state_t is (S_Init, S_Btn1On, S_Btn1Off, S_Btn2On, S_Btn2Off);
  variable state: state_t := S_Init;
begin
  if (reset = '1') then
    state := S_Init;
  elsif (clock = '1' and clock'event) then

    case state is
      when S_Init =>
        tmr_clear <= '1';
        tmr_start <= '0';
        if (B = '1') then
          state := S_Btn1On;
        else
          state := S_Init;
        end if;
      ...
      when S_Btn2Off =>
        tmr_clear <= '0';
  
```

```

    tmr_clear <= '0';
    Tout <= tmr_T;
    state := S_Init;
end case;
end if;
end process;

```

- (c) **Capture the system using C code executing on a microcontroller. The microcontroller has a timer that is cleared by calling `Tmr_Clear()`, is started by calling `Tmr_Start()`, is stopped by calling `Tmr_Stop()`, and whose value in milliseconds is returned when calling `Tmr_Read()`. (Note: This timer is different from the one in lecture). TO SAVE SPACE, just show the first state and last state, and use `"..."` for the states in between.**

```

typedef enum states_s {
    S_Init, S_Btn1On, S_Btn1Off, S_Btn2On, S_Btn2Off
} states_t;

int main() {
    states_t state = S_Init;
    while (1) {
        switch (state) {
            case S_Init:
                Tmr_Clear();
                if (B == 1) {
                    state = S_Btn1On;
                } else {
                    state = S_Init;
                }
                break;
            ...
            case S_Btn2Off:
                Tout = tmr_T;
                state = S_Init;
            }
        }
    }
    return 0;
}

```