Design Example:

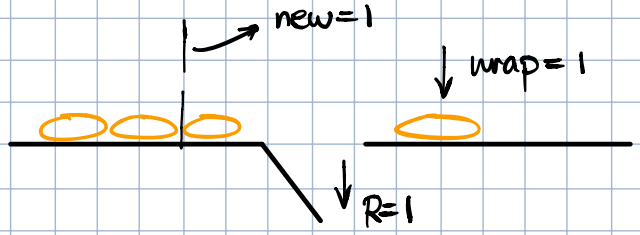
A factory bakes cookies that move down a conveyer belt. There are 3 sensors examine each cookie:

S_0: gives 1 iff a cookie is burned.

S_1: gives 1 iff a cookie is too light in weight.

S_2: gives 1 iff a cookie is not rounel.

- A logic fuction $R=1$ rejects only cookie for which two or more sensors produce 1. Cookies not rejected get wrapped.



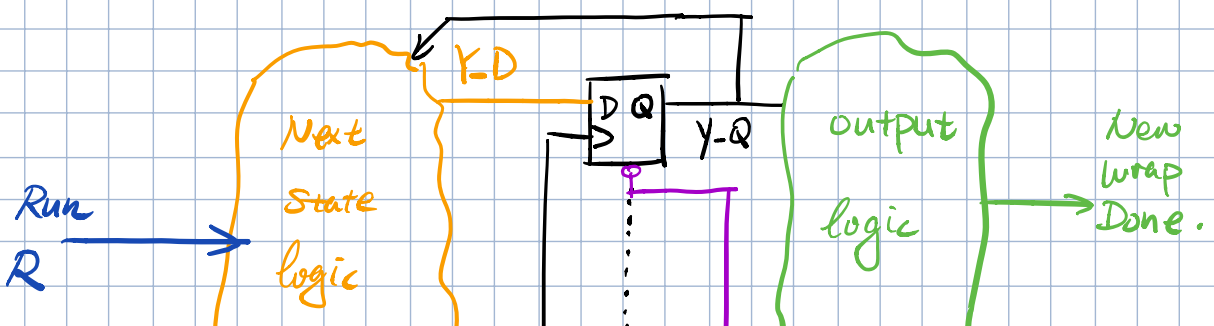
- Design a finite state machine (controller) that does the following. synchronized by a clock signal:

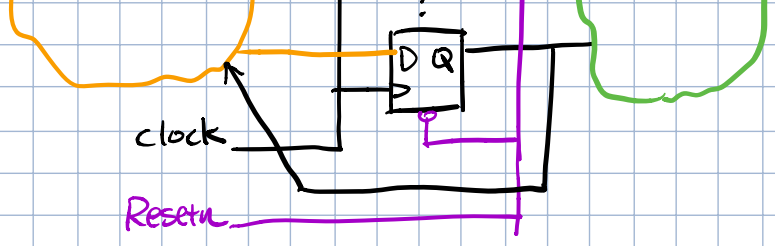
- When an input Run=0, don't do anything. But when Run=1, request a cookie by setting New=1 for one clock cycle.

(*) - wait a clock cycle to allow the sensors to produce R.

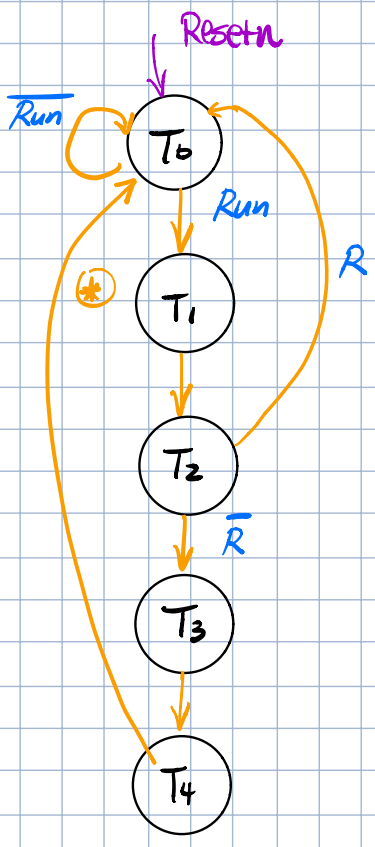
- if the cookie gives R=1, don't wrap it. Set Done=1

- but if the cookies gives R=0, set wrap=1 for two clock cycles In the second of these cycles set Done=1





State Diagram.



	T_0	T_1	T_2	T_3	T_4
$R=1$	New=Run		Done=1		
$R=0$	New=Run			Wrap=1	Done=1 Wrap=1

Implement the FSM on the DE1-SoC Board.:

- connect Run to SW9
- Resetn to KEY0
- Clock to KEY1
- Sensors to SW2, SW1, SW0

Verilog.

```
module  FSM_control (SW, KEY, LEDR);
    ... declare ports & wires
    reg [2:0] y_Q, Y_D;
    wire [1:0] sum;
        ⋮
    parameter  T0 = 3'b000, T1 = 3'b001, .... T4 = 3'b100;
    //FSM state table
    always @ (*)
        case (y_Q)
```

```
T0: if(Run) Y_D = T1;
    else Y_D = T0;
T1: Y_D = T2;
T2: if(R) Y_D = T0;
    else Y_D = T3;
T3: Y_D = T4;
T4: Y_D = T0;
default: Y_D = 3'bxxx;
endcase

// FSM outputs.
always @ (*)
  begin
    Done = 1'b0; New = 1'b0; Wrap = 1'b0
    case (y_Q)
      T0: New = Run;
      T1:
      T2: if(R) Done = 1'b1;
      T3: Wrap = 1'b1;
      T4: begin
          Wrap = 1'b1; Done = 1'b1;
          end.
always@ (posedge KEY[1])
    if (KEY[0]==0) y_Q<=T0;
    else y_Q <= Y_D;
assign sum = sw[2] + sw[1] + sw[0];
assign R = sum[1];
```

means don't care, but we have to specifie it. to avoid latches.

default values. avoid latches.

```
assign  LEDR[9] = SW[9];
        ⋮
assign  LEDR[0] = SW[0];

endmodule.
```