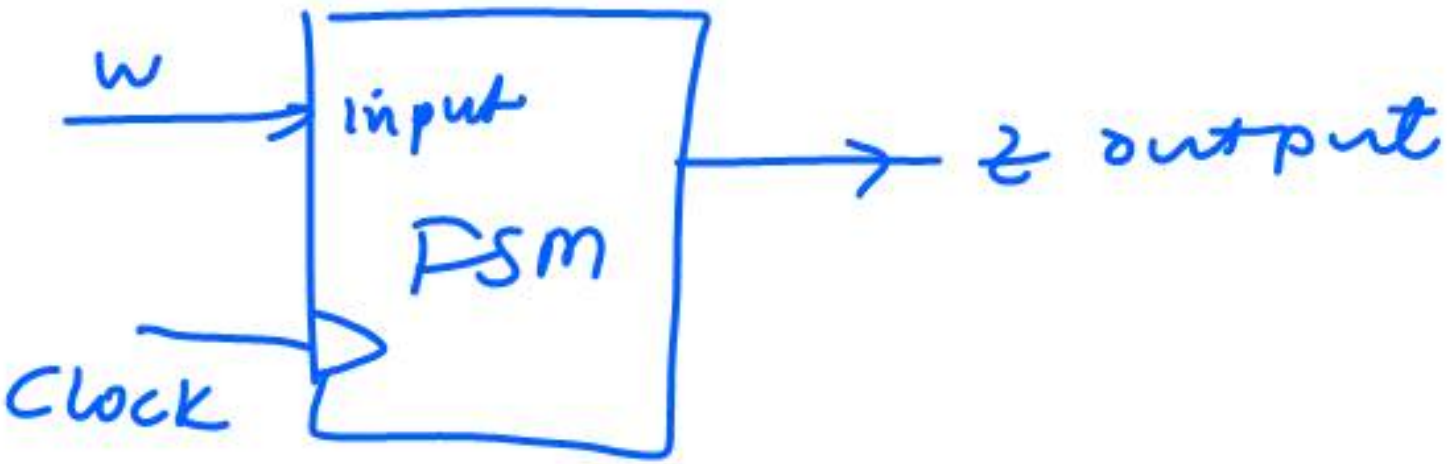
Finite State Machines

09.05.2011

Example:

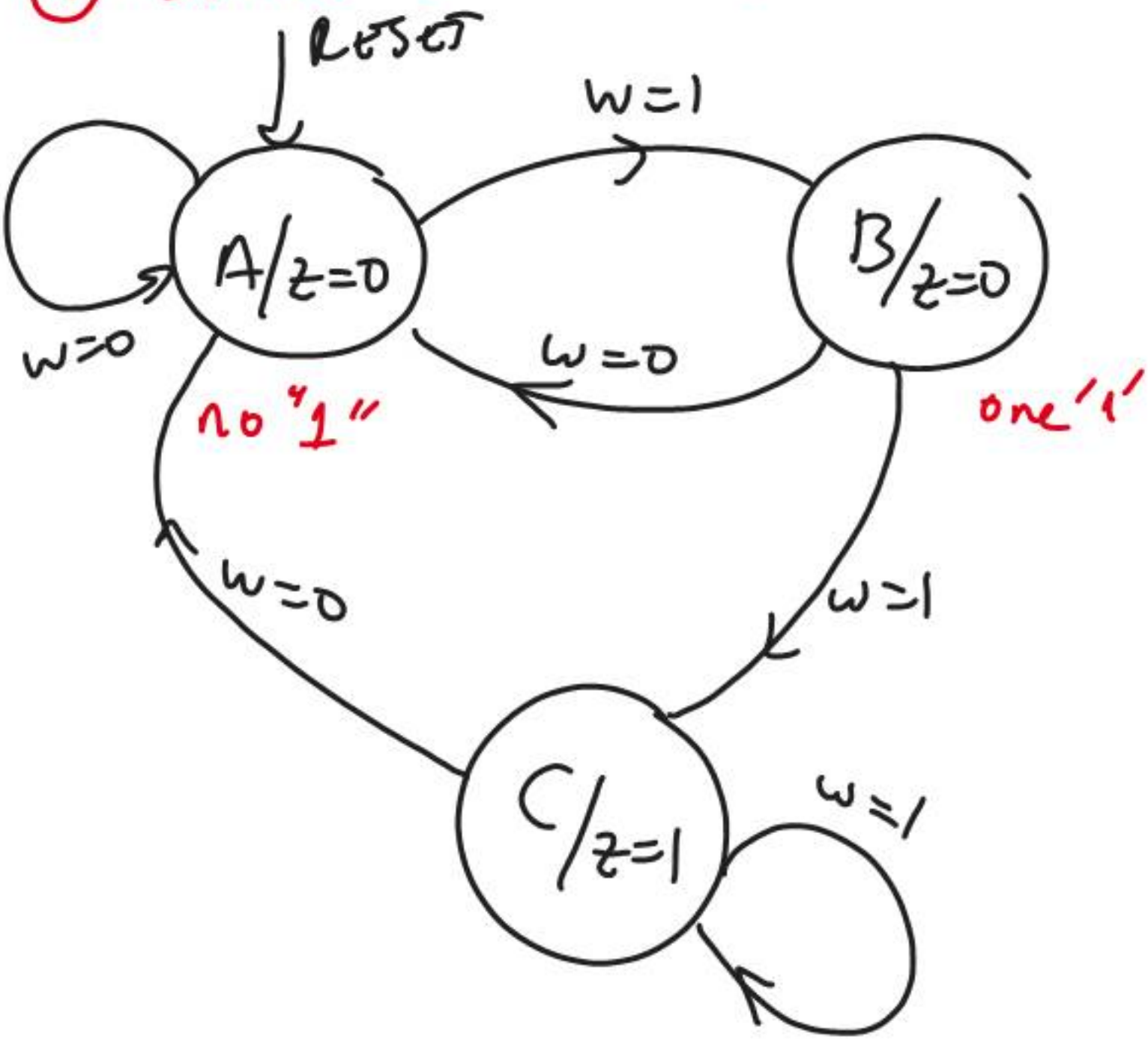


Clock cycles:	t_0	t_1	t_2	t_3	t_4	t_5	t_6	t_7	t_8	t_9
w:	0	1	0	1	1	0	1	1	1	0
z:	0	0	0	0	0	1	0	0	1	1

BASIC DESIGN STEPS of a FSM:

1. Sketch the STATE DIAGRAM
2. Generate STATE TABLE
3. Generate STATE ASSIGNED TABLE
4. FF TYPE SELECTION, DERIVATION of next STATES → (WRITE THE VHDL CODE)
5. IMPLEMENTATION on the FPGA

(f) STATE DIAGRAM



(2) STATE TABLE

Present State	Next State		Output
	w=0	w=1	z
A	A	B	0
B	A	C	0
C	A	C	1

State variables for present states
" " for next states

I need 3 states: A, B, C

to represent them with binary variables

I need minimum 2 binary digits.

For ex: 00, 01, 10, 11 2^2

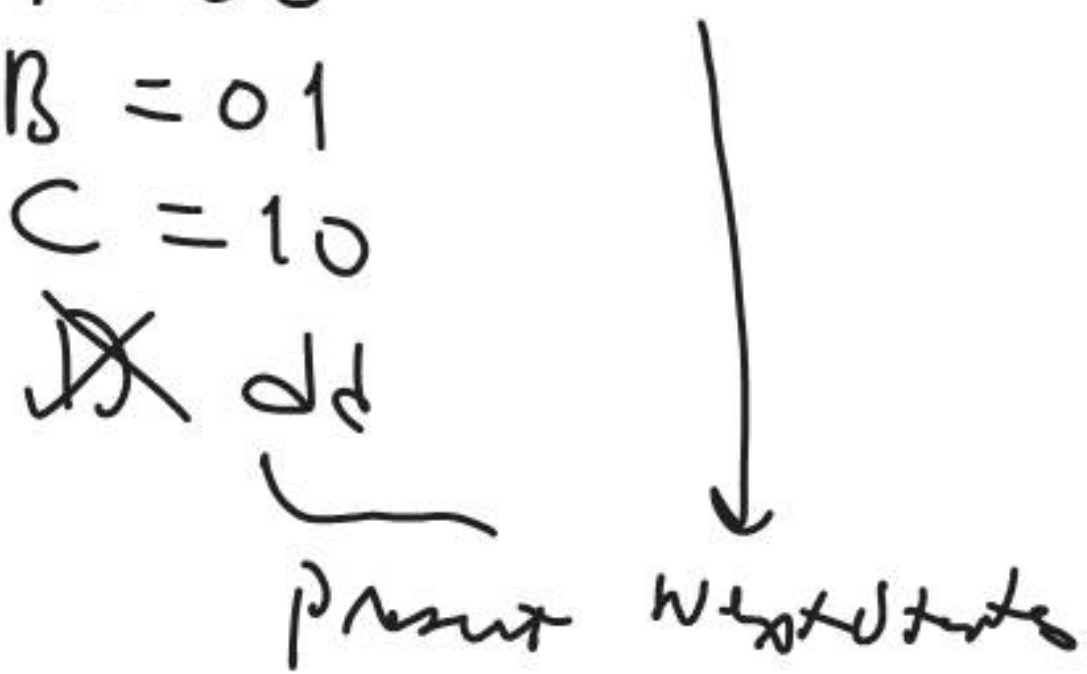
$y_2 y_1$ $Y_2 Y_1$

A = 00

B = 01

C = 10

~~D~~ dd



③ STATE ASSIGNED TABLE

	Present state $y_2\ y_1$	Next State $w=0$ $Y_2\ Y_1$	Next State $w=1$ $Y_2\ Y_1$	Output z
A	0 0	0 0	0 1	0
B	0 1	0 0	1 0	0
C	1 0	0 0	1 0	1
	1 1	d d	d d	d

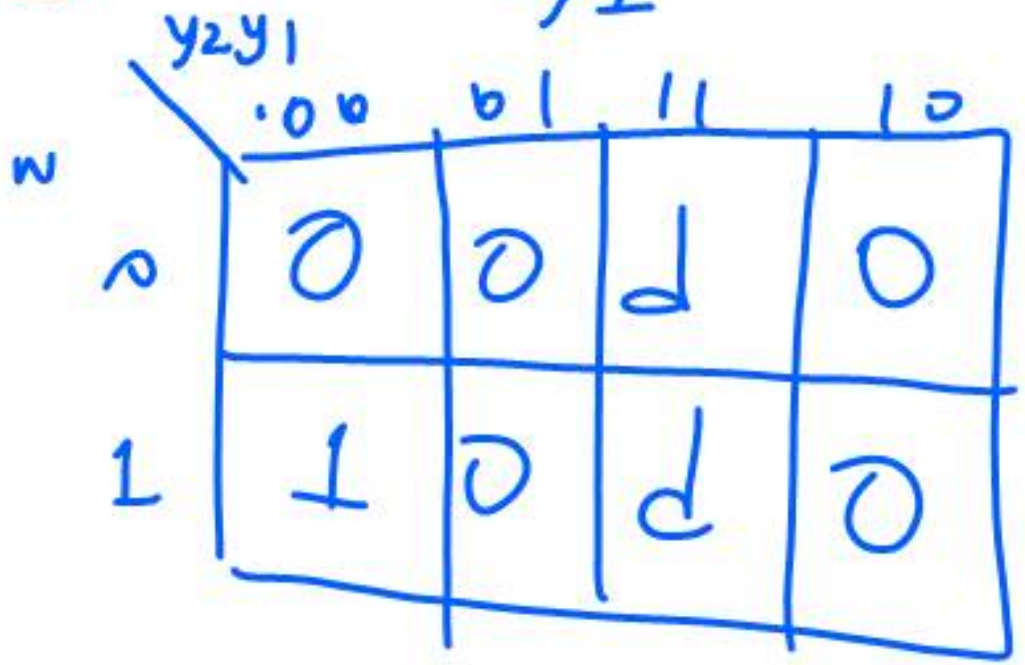
Outputs of the FFs

Inputs of the FFs

④ FF TYPE Selection & minimization

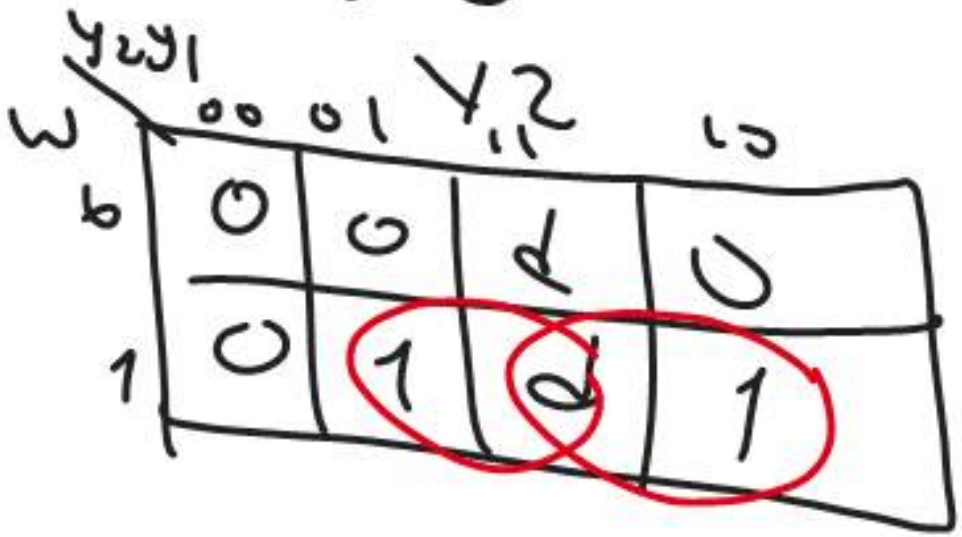
select D FFlops.

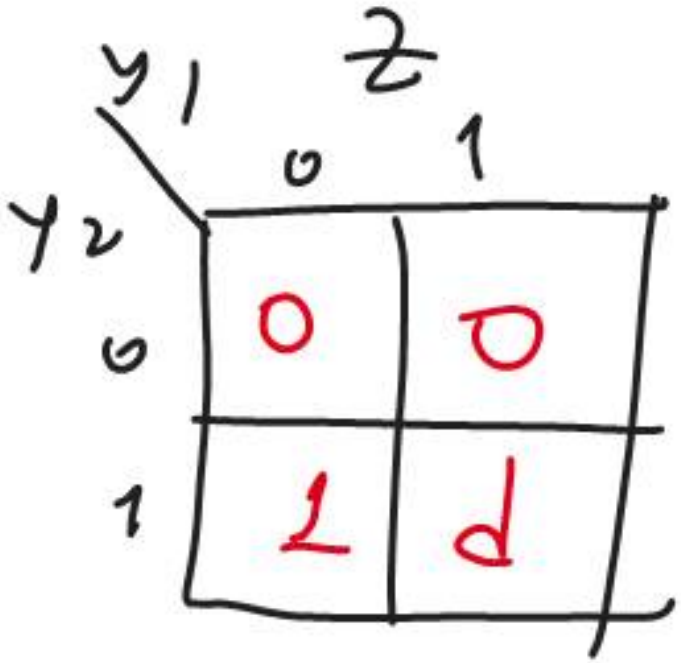
Y1



$$Y1 = w\bar{y}_2\bar{y}$$

$$Y2 = w_1y_1 + w_1y_2$$

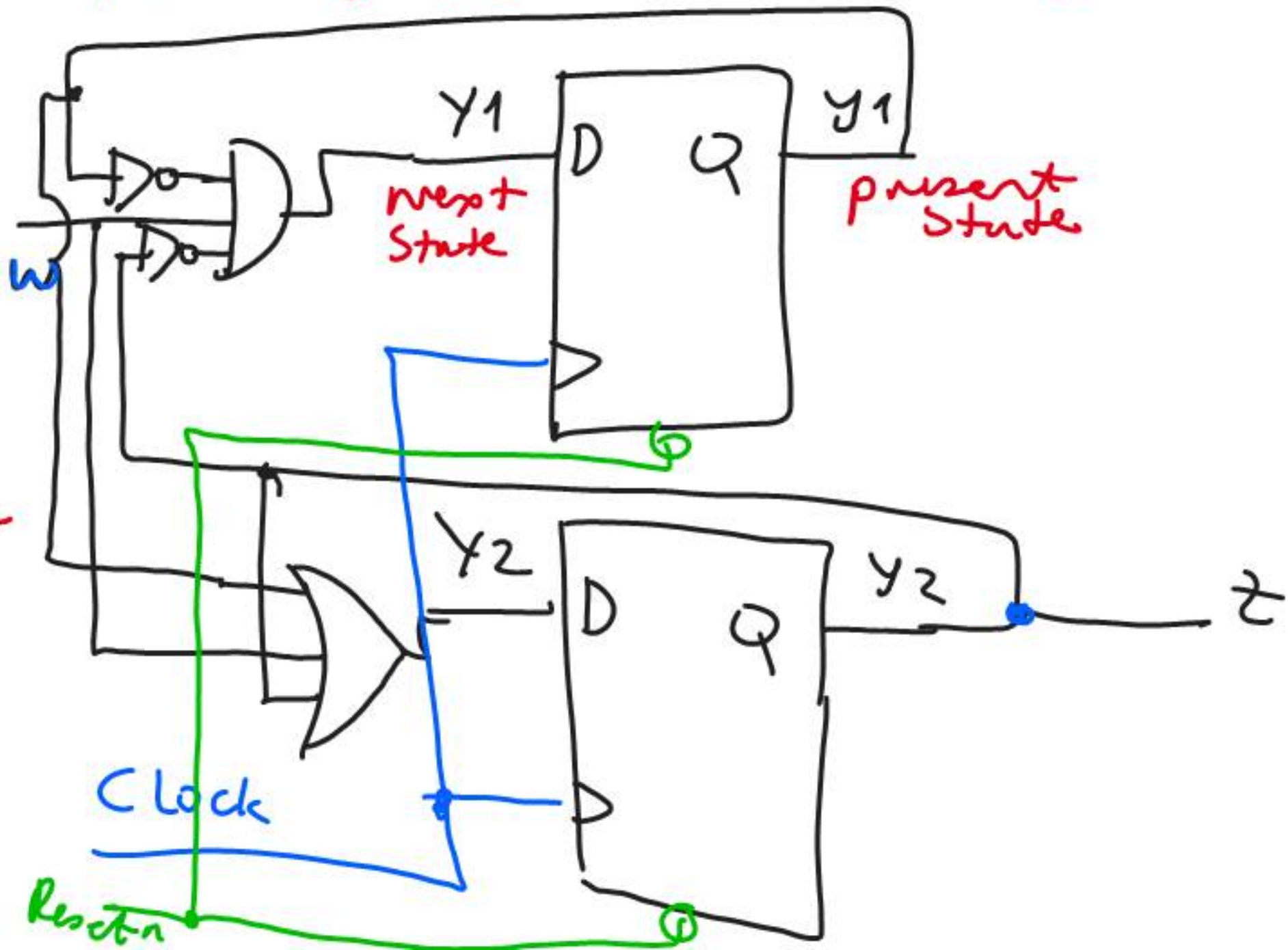




$z = y_2$

This is a MOORE type FSM, where the output is not a function of INPUTS but the STATUS.

$Y1 = w\bar{y}_1\bar{y}_2 \quad Y2 = w(y_1 + y_2) \quad z = y_2$



VHDL CODE :

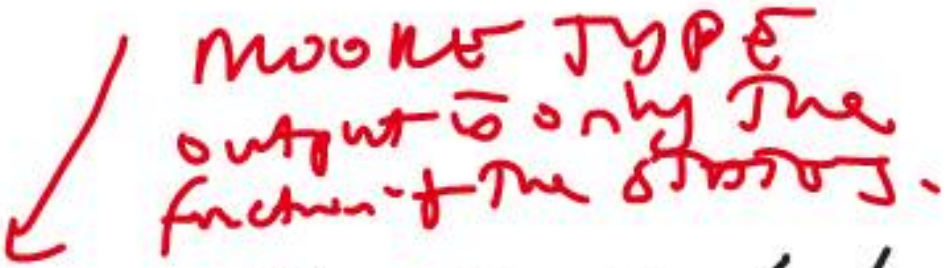
```
ENTITY Circuit_LS
PORT ( Clock, Reset_n, W : IN STD_LOGIC;
       Z                 : OUT STD_LOGIC);
END Circuit;

ARCHITECTURE Behavior OF Circuit_LS
 TYPE State_Type IS (A,B,C);
 SIGNAL y : state_type;
 BEGIN
```

```
PROCESS (Reset_n, Clock)
BEGIN
IF Reset_n = '0' THEN
   y <= A;
ELSIF (Clock'EVENT AND Clock = '1') THEN
   CASE y IS
      WHEN A =>
         IF w = '0' THEN
            y <= A;
         ELSE
            y <= B;
```

```
ENDIF;
WHEN B =>
 IF w='0' THEN
  y <= A;
 ELSE
   y<=C;
 ENDIF;
WHEN C =>
 IF w='0' THEN
   y <= A;
 ELSE y<=C;
```

```
END IF;
END CASE;
END PROCESS;
Z <= '1' WHEN y=C ELSE '0';
END Behavior;
```



Alternative code with present and next states:

```
;
ARCHITECTURE Behaviour of Circuit IS
TYPE State_type IS (A, B, C);
SIGNAL y_present, y_next : State_type;
BEGIN
 PROCESS (w, y_present)
 BEGIN
 CASE y_present IS
 WHEN A =>
```

```
IF w='0' THEN
  y_next <= A;
ELSE
  y_next <= B;
END IF;

WHEN B =>
IF w='0' THEN
  y_next <= A;
ELSE
  y_next <= C;
END IF;

WHEN C =>
IF w='0' THEN
  y_next <= A;
ELSE
  y_next <= C;
END IF;
END CASE;
END PROCESS;

PROCESS (Clock, Reset)
BEGIN
IF Reset='0' THEN
  y_present <= A;
```

```
ELSIF (clock'EVENT AND clock = '1') THEN
    y_present <= y_next

END IF;
END PROCESS;
z <= '1' WHEN y_present = C_LAST '0';
END Behavior;
```