ex: a 4-bit binary counter with parallel LOAD, using

04.05.2011

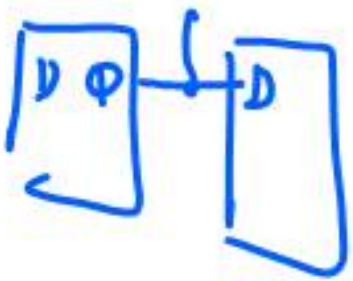
(c)

* → INTEGER SIGNALS

ex Counter that counts from 0 to 15:

```
LIBRARY ieee;
USE ieee.std_logic_1164.all;
ENTITY upcount IS
PORT (R : IN INTEGER RANGE 0 TO 15;
      Clock, Resetn, L : IN STD_LOGIC;
      Q : BUFFER INTEGER RANGE 0 TO 15);
END upcount;
```

R: IN STD_LOGIC_VECTOR (3 DOWNTO 0) a 4-bit input



HW: What is the difference between INOUT and BUFFER?

```
ARCHITECTURE Behavior OF UPCOUNT_LS
BEGIN
PROCESS (Clock, Reset_n)
 BEGIN
  IF Reset_n = '0' THEN -- asynchronous RESET!
  Q <= 0; -- because we have used INTEGER 0 TO 15
  ELSIF ( Clock'EVENT AND Clock = '1') THEN
  IF L='1' THEN
     Q <= R; -- parallel loading
```

```
ELSE -- if L /= 1
  Q <= Q+1 -- count up
END IF;
END IF;
END PROCESS;
END Behavior;
```

Ex: down counter

```
LIBRARY ieee;
USE ieee.std_logic_1164.all;
ENTITY downcount IS
GENERIC (modulus: INTEGER := 8);
PORT (Clock, L, E : IN STD_LOGIC;
      Q : OUT INTEGER RANGE
          0 TO modulus-1);
END downcount;
```

no Buffer here; Then we have to define a SIGNAL

```
ARCHITECTURE Behavior OF downcount IS
SIGNAL Count: INTEGER RANGE 0 TO modulus-1;
```

```
BEGIN
PROCESS  -- no sensitivity list here
BEGIN
WAIT UNTIL (Clock'EVENT AND clock = '1');
IF L='1' THEN
   Count <= modulo - 1;  -- biggest 3-bit number (111) to loaded for count down
ELSE
   IF E='1' THEN
      Count <= count - 1;  -- arithmetic operation because COUNT is defined as INTEGER
   END IF;
END IF;
```

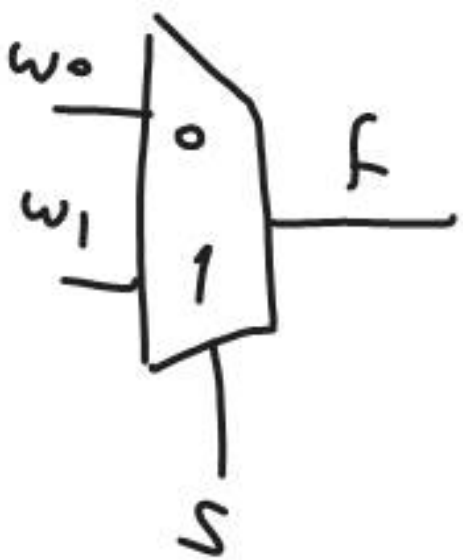
because of this

Default for the next IF

```
END PROCESS;
Q <= count;

END Behavior;
```

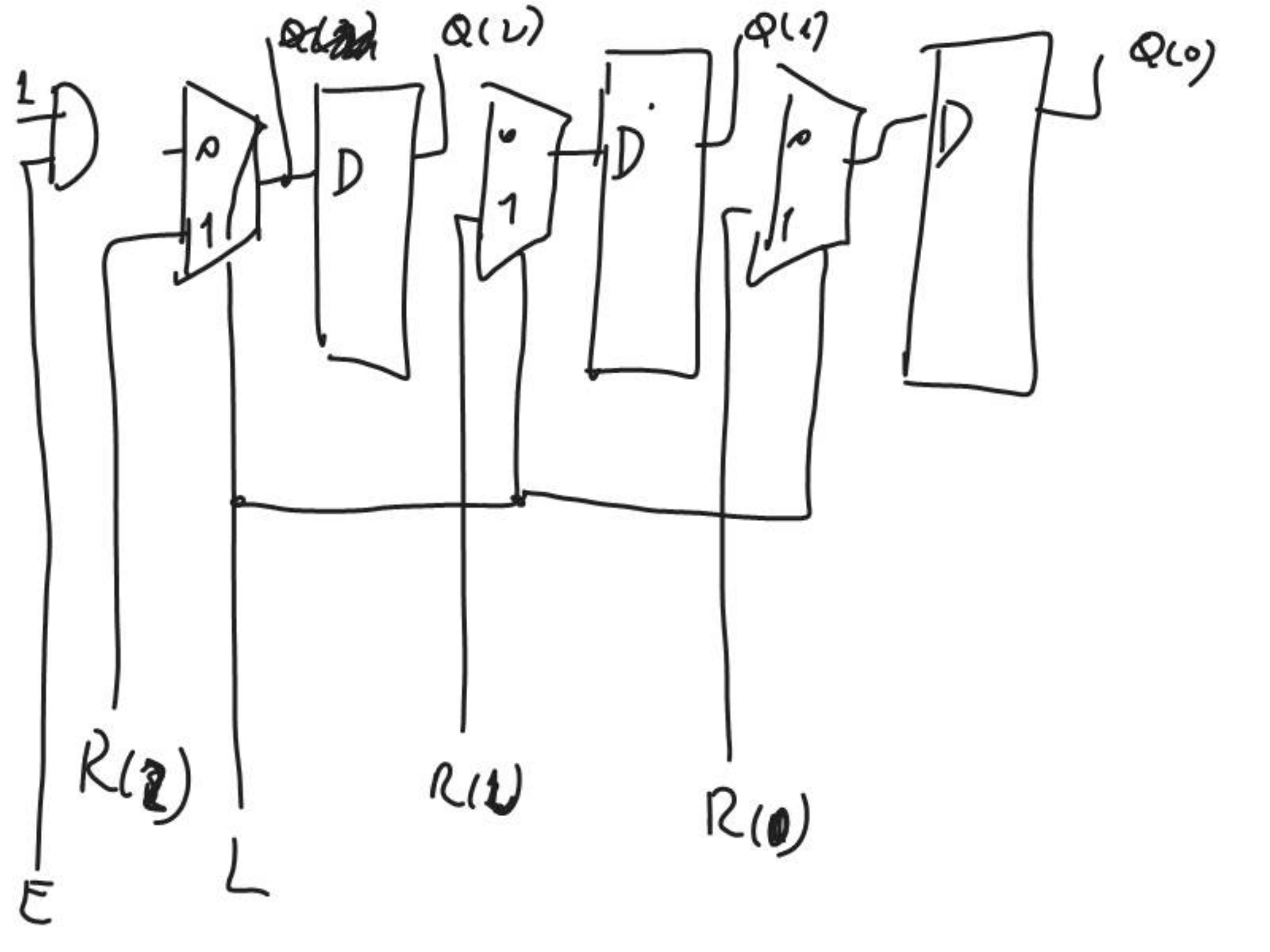
Remember:



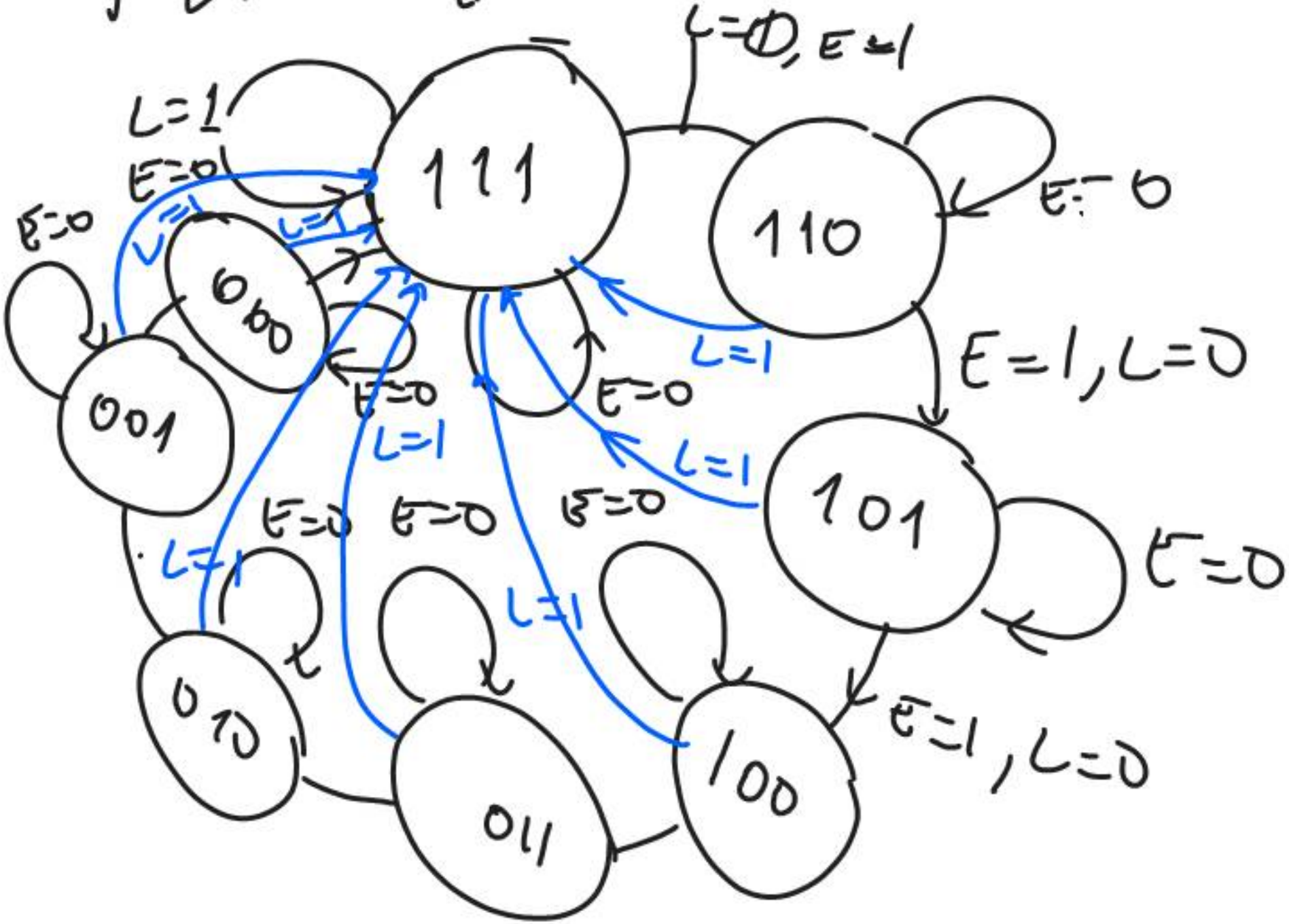

```
IF S='0' THEN
  f <= w0;
ELSE
  f <= w1;
END IF;
```

default

```
f <= w0;
IF S='1' THEN
  f <= w1;
END IF;
```


Q(2)
Q(1)
Q(1)
Q(0)
1
0
1
D
D
D
D
0
1
0
1
R(2)
R(1)
R(0)
L
E

FSM Design of modulo 7 Down-Counter

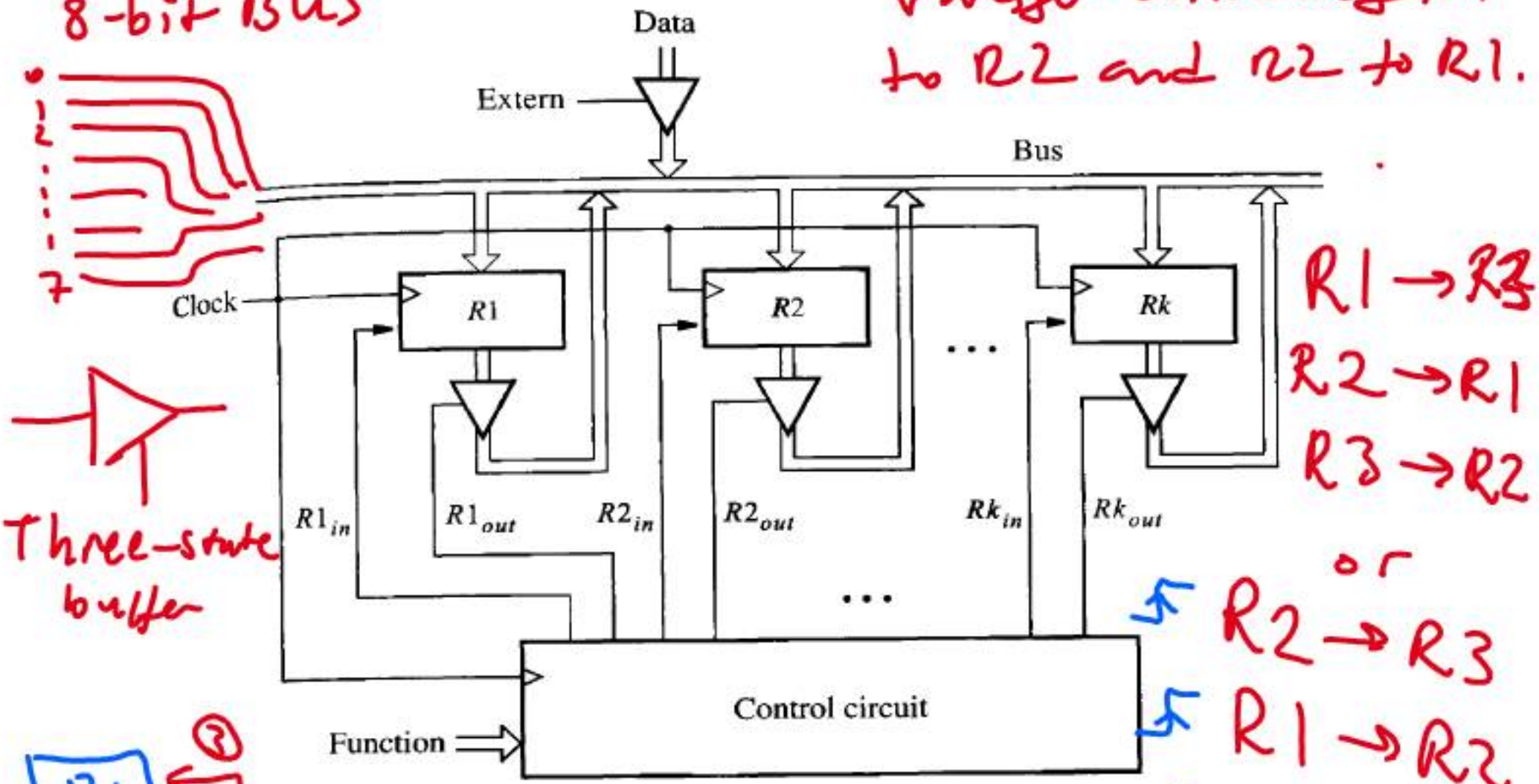


SWAP CIRCUIT:

R1, R2, R3

Transfer contents of R1 to R2 and R2 to R1.

8-bit Bus



Three-state buffer

R1 → R3

R2 → R1

R3 → R2

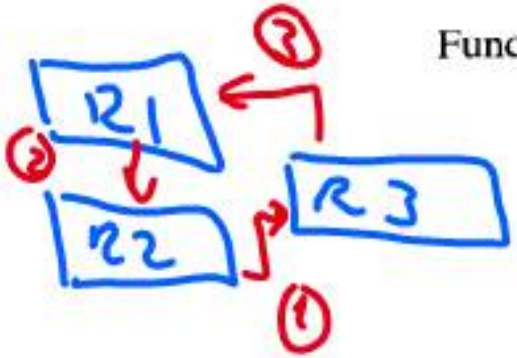
or

R2 → R3

R1 → R2

R3 → R1

R1 ⟷ R2



f $R_2 \rightarrow R3 \Longrightarrow R_{2_{out}} = 1$, $R_{3in} = 1$

f $R1 \rightarrow R2 \Longrightarrow R_{1out} = 1$, $R_{2in} = 1$

f $R3 \rightarrow R1 \Rightarrow R_{3out} = 1$, $R_{1in} = 1$

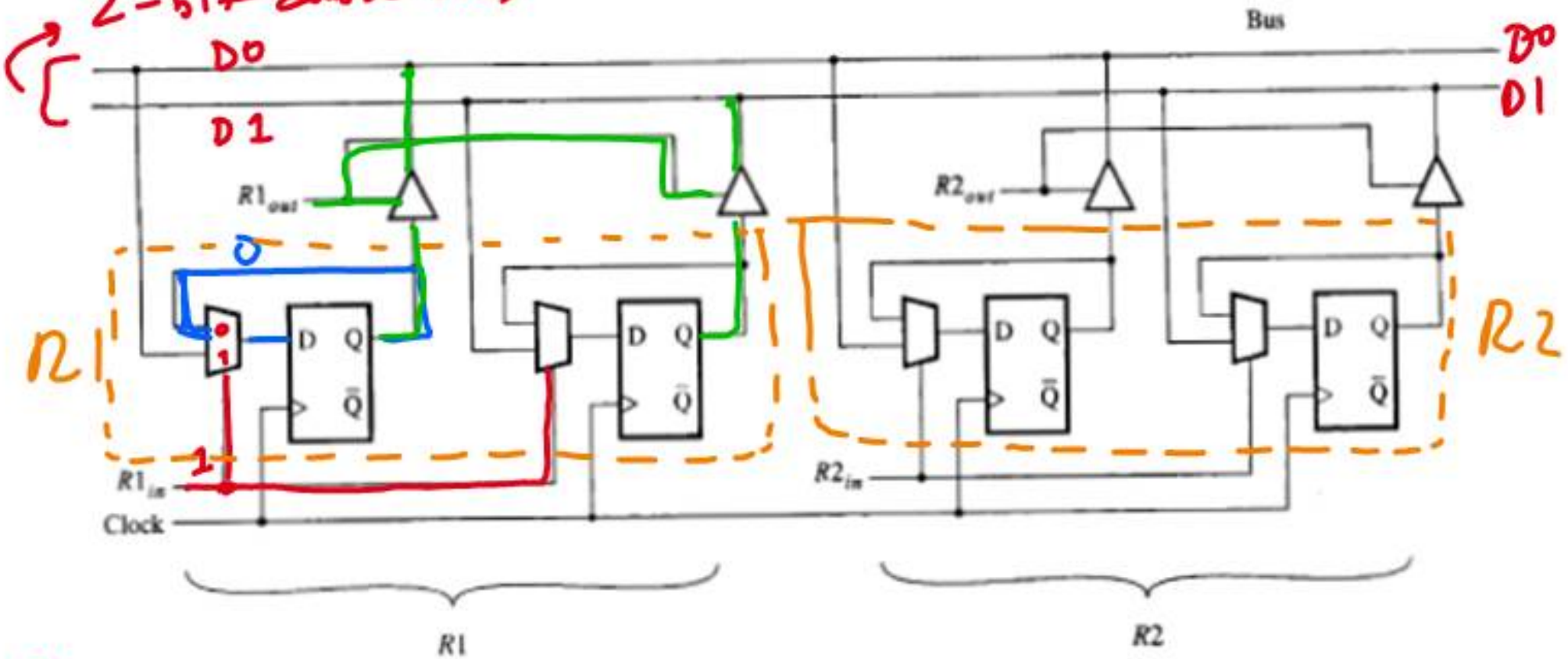
$R_1 \longleftrightarrow R2$

The rest are ϕ for each case.

	f①	f②	f③
$R1_{in}$	0	0	1
$R1_{out}$	0	1	0
$R2_{in}$	0	1	0
$R2_{out}$	1	0	0
$R3_{in}$	1	0	0
$12\emptyset_{out}$	0	0	1
	$\emptyset$	$\emptyset$	$\emptyset$

Control circuit should do this for swapping R_1 and R_2

2-bit data bus

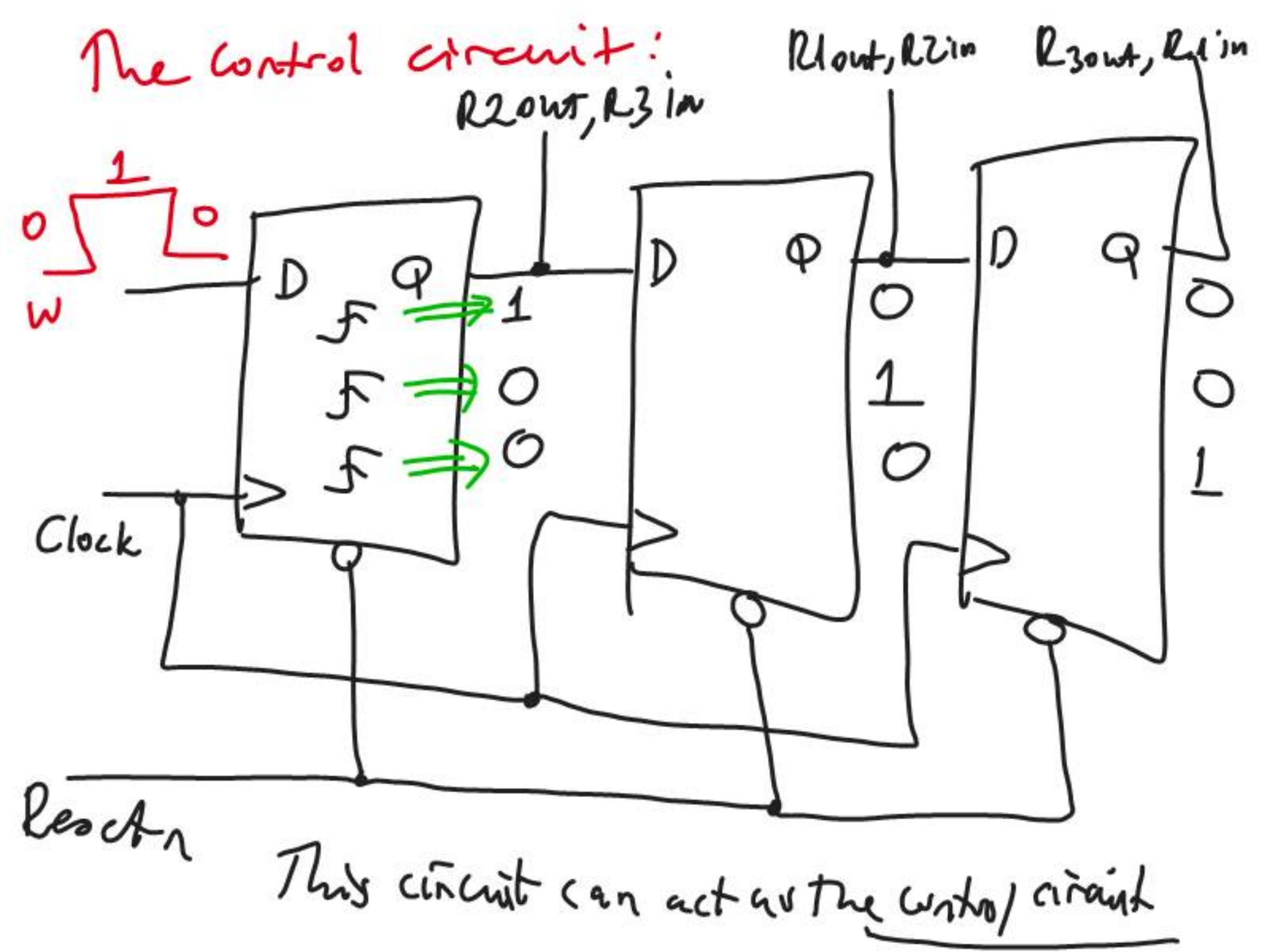


R1 = 2-bit Register

R2 = " "

Connection of 2-bit registers (R1, R2)

to the data bus of 2 bit



This circuit can act as the counter circuit

Using multiplexers in stead of 3-state BUFFERS