Printout

November 28, 2022 3:53 PM

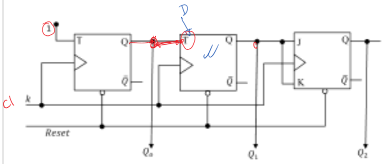
Counters → Synchronous / Asynchronous; up counters / down counters } modulo counter

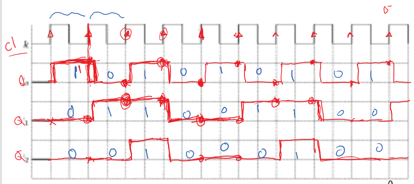
Final exam
- Counter / Register
- FSM (2 questions)
- VHDL codes (FSM)
- Small processor → Instructions will be given: PC, IR, ACC, M → Machine codes for a simple program
- Design simple processor

Counter

Q1. Following circuit is a counter

(a) Complete the timing diagram of the counter starting from $Q_2Q_1Q_0$=000.





S_1	S_0	Function
0	0	Addition
0	1	Subtraction
1	0	2's complement
1	1	Logical XOR

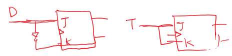
Instruction sets

ADD X, Y

SUB X − Y

$Y = \bar{X} + 1$

$F = X + Y$

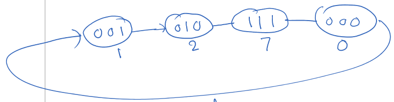


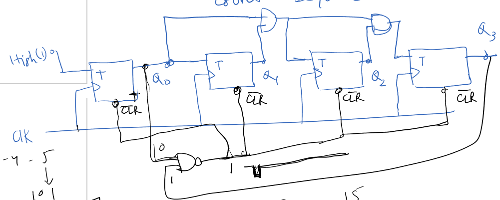
T	Q
0	NC
1	Toggle

(b) Change the design of the counter to a modulo-10 up counter.

0 − 1 − 2 − 3 − 4 − 5 − 6 − 7 − 8 − 9

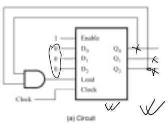
Asynchronous counter



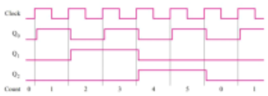


(c) Design a modulo-6 counter with synchronous reset

0 − 1 − 2 − 3 − 4 − 5

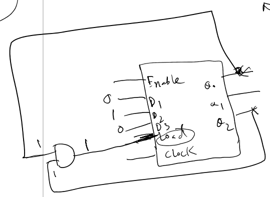


(a) Circuit



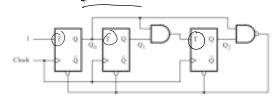
(b) Timing diagram

Figure 7.26 A modulo-6 counter with synchronous reset.

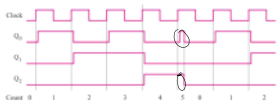


Design a counter for a sequence of 2-3-4-5

(d) Design a modulo-6 counter with asynchronous reset.



(a) Circuit



(b) Timing diagram

Figure 7.27 A modulo-6 counter with asynchronous reset.

Q2. Show how a JK flip-flop can be constructed using a T flip-flop and other logic gates.

***7.18** The circuit in Figure P7.3 looks like a counter. What is the sequence that this circuit counts in?

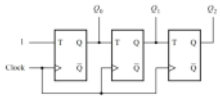


Figure P7.3 The circuit for Problem 7.18.

VHDL Code

Q1. Consider the following VHDL code:

```
LIBRARY ieee;
USE ieee.std_logic_1164.all;
ENTITY fsm IS
        PORT (Clock, Resetn, w      : IN        STD_LOGIC;
                    z               : OUT       STD_LOGIC );
END fsm;

ARCHITECTURE    Behavior    OF    fsm    IS
        TYPE  State_type    IS    (A, B, C);
        SIGNAL      y: State_type;
BEGIN
        PROCESS(Resetn, Clock)
        BEGIN
                IF Resetn = '0' THEN
                        y<=A;
                ELSEIF (Clock'EVENT AND Clock = '1') THEN
                        CASE y   IS
                                WHEN A=>
                                        IF      w='0'   THEN        y<=A;
                                        ELSE            y<=B;
                                        END   IF;
                                WHEN B=>
                                        IF w='0'        THEN        y<=A;
                                        ELSE            y<=C;
                                        END   IF;
                                WHEN C=>
                                        IF w='0'        THEN        y<=A;
                                        ELSE            y<=C;
                                        END   IF;
                        END CASE;
                ENDIF;
        END PROCESS;
        z<='1' WHEN y=C ELSE '0';
END Behavior;
```

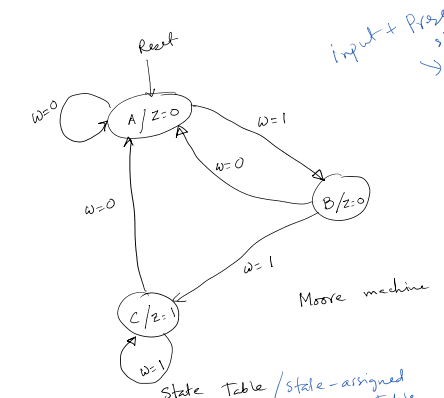
(a) Describe the behavior of the given FSM using a state diagram.

(b) Assume that an 8×4-bit EPROM is available, provide a programmable implementation of the FSM in part (a):

(i) Fill in the content of the EPROM in the table below by setting up the address of the EPROM as follows: a_2=w, a_1a_0=y_1y_2 (present sate), and explain what the contents of the EPROM represents.

(ii) Draw a schematic diagram for the implementation.

Q2.



State Table / State-assigned Table

Present state	Next state w=0	Next state w=1	Output z
A 00	A 00	B 01	0
B 01	A 00	C 10	0
C 10	A 00	C 10	1
11	00	00	0

Unused

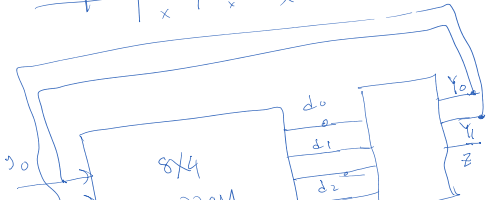
Input + Present state → Address

Data ← Next states + output

$w y_1 y_0$	d_3	$d_2(Y_1)$	$d_1(Y_0)$	$d_0(z)$
000	/	0	0	0
001	/	0	0	0
010	/	0	0	0
011	/	0 x	0 x	0 x
100	/	0	0	1
101	/	0	1	0
110	/	1	1	0
111	/	0	0	0
	x	x	x	

Undefined

Undefined



Q2.

```
LIBRARY ieee;
USE ieee.std_logic_1164.all;
ENTITY fsm IS
        PORT (Clock, Resetn, w      : IN        STD_LOGIC;
                    z               : OUT       STD_LOGIC );
END fsm;

ARCHITECTURE    Behavior    OF      fsm     IS
        TYPE  State_type    IS      (A, B, C);
        SIGNAL      y: State_type;
BEGIN
        PROCESS(Resetn, Clock)
        BEGIN
                IF Resetn ='0' THEN y<=A;
                ELSEIF (Clock'EVENT AND Clock = '1') THEN
                        CASE y  IS
                                WHEN A=>
                                        IF      w='0'  THEN         y<=B;
                                        ELSE  y<=C;
                                        END   IF;
                                WHEN B=>
                                        IF      w='0'  THEN         y<=A;
                                        ELSE  y<=C;
                                        END   IF;
                                WHEN C=>
                                        IF      w='0' THEN          y<=B;
                                        ELSE  y<=C;
                                        END   IF;
                        END CASE;
                ENDIF;
        END PROCESS;
PROCESS (y, w)
BEGIN
                CASE y  IS
                                WHEN A=>
                                        z<=0
                                WHEN B=>
                                        z<=NOT w;
                                WHEN C=>
                                        z<=w
                        END CASE;
                ENDIF;
        END PROCESS;
END Behavior;
```

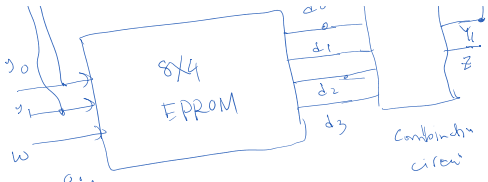
(a) Describe the behavior of the given FSM using a state diagram.

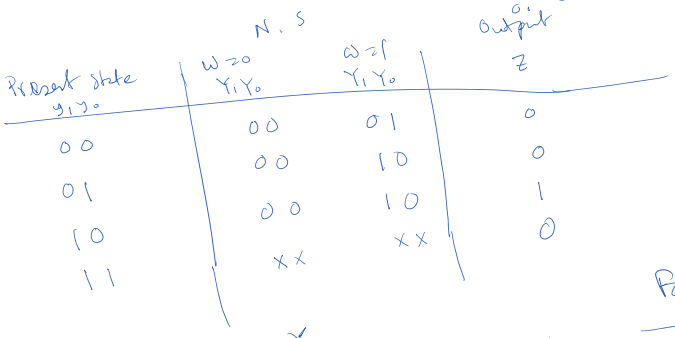
(b) Assume that an 8×4-bit EPROM is available, provide a programmable implementation of the FSM in part (a):

(iii) Fill in the content of the EPROM in the table below by setting up the address of the EPROM as follows: a_2=w, a_1a_0=y_1y_0 (present sate), and explain what the contents of the EPROM represents.

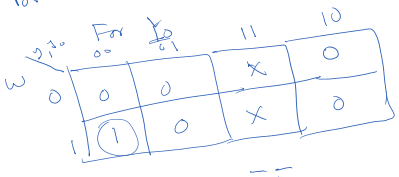
(iv) Draw a schematic diagram for the implementation.

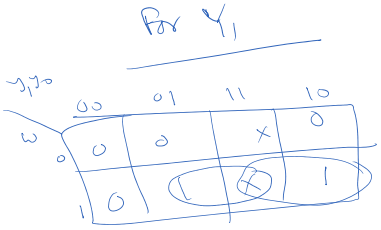
(v) Implement the FSM using D flip-flops.





$Z = y_1\bar{y}_0$





$Y_0 = w\bar{y}_1\bar{y}_0$

$Y_1 = wy_0 + wy_1$

$= w(y_0 + y_1)$