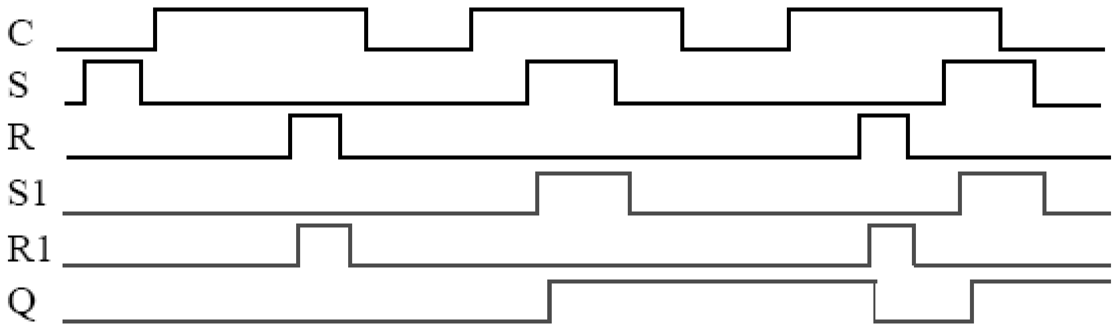
3.2(b,c)
b) 1/100,000,000 = 10 ns
c) 1/1,500,000,000 = 0.66 ns = 667 ps

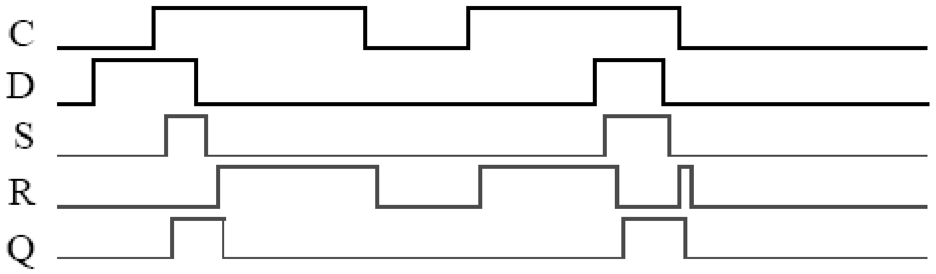
3.4(a,d)

a) 1/500ms = 2 Hz
d) 1/20ps = 50,000,000,000 Hz = 50 GHz

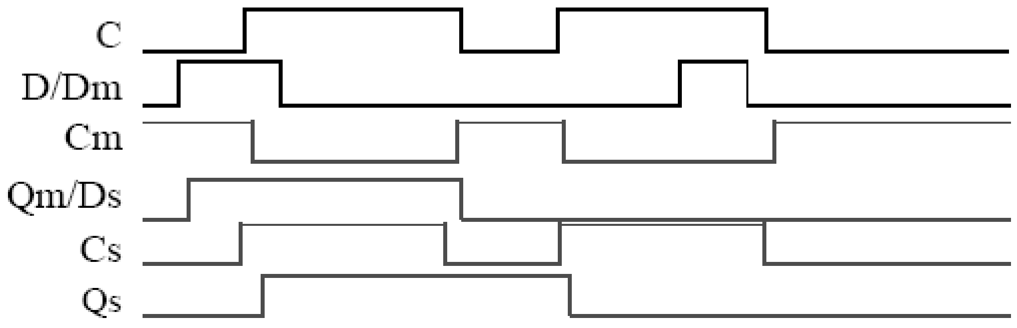
3.9



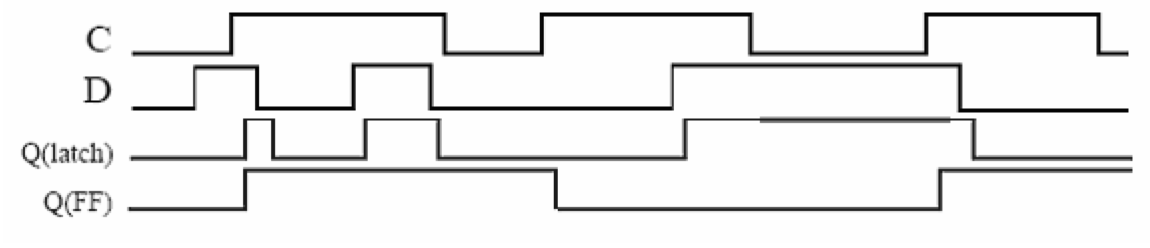
3.11



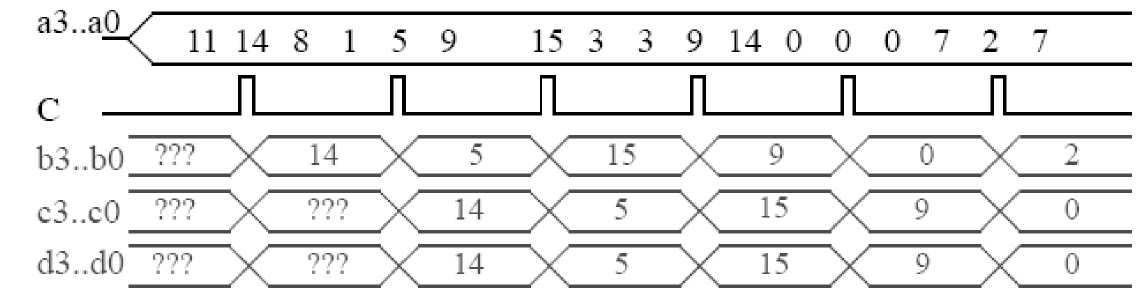
3.13



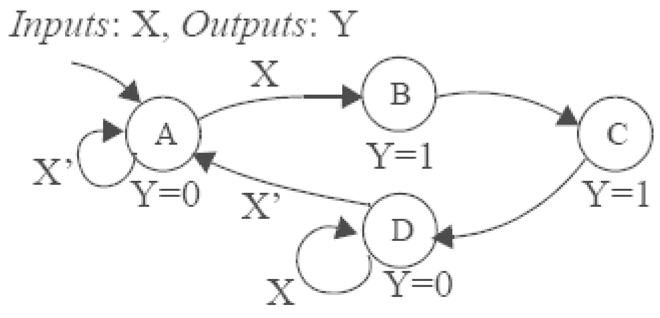
3.15



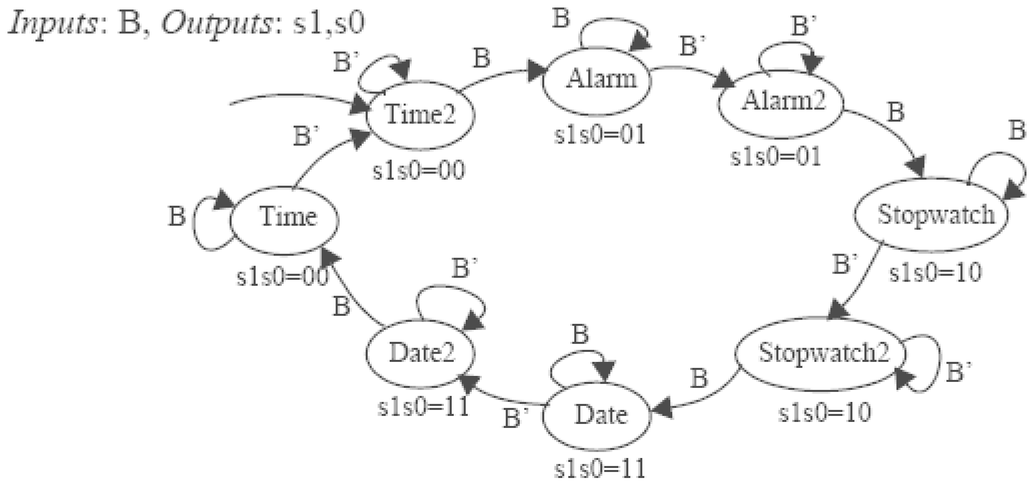
3.21



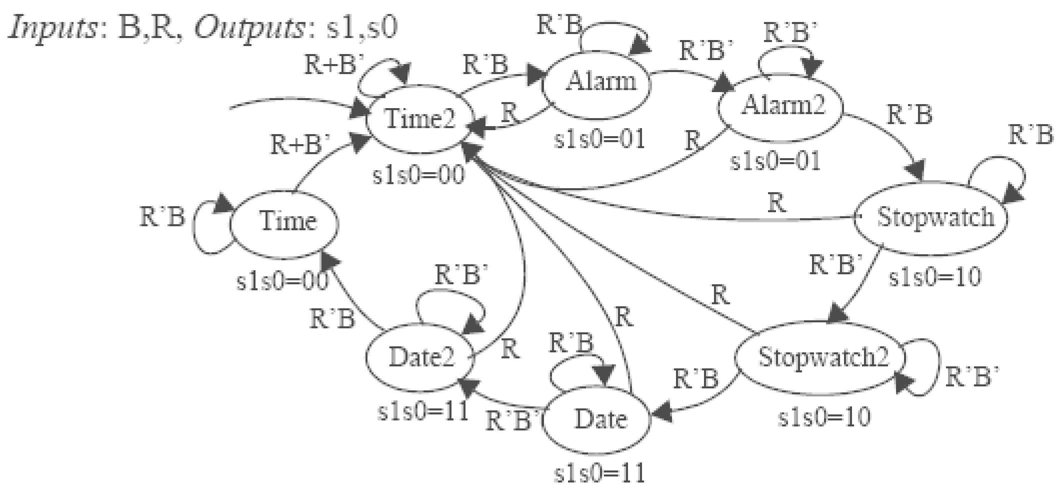
3.23



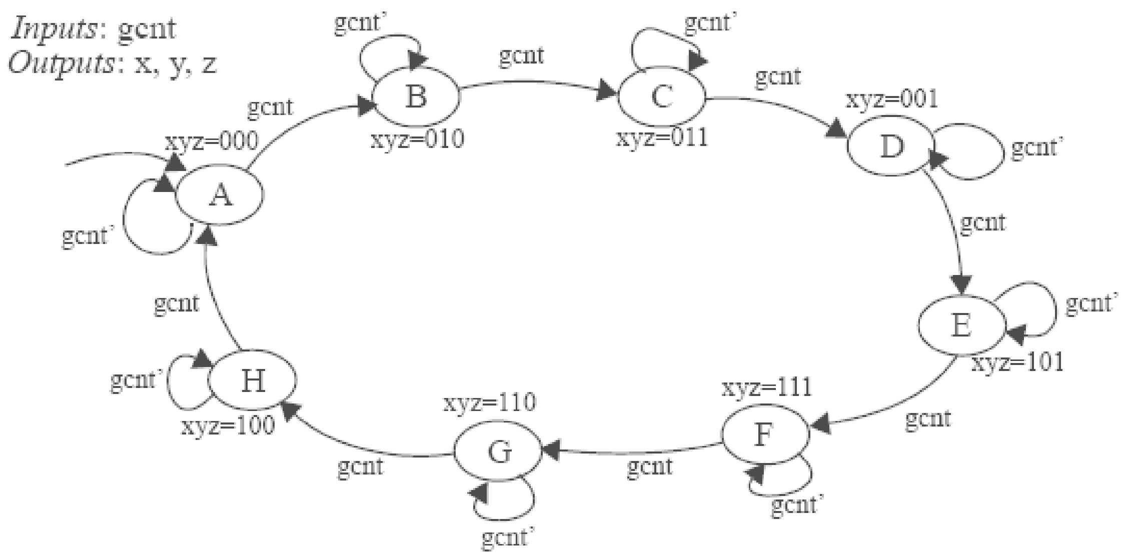
3.27



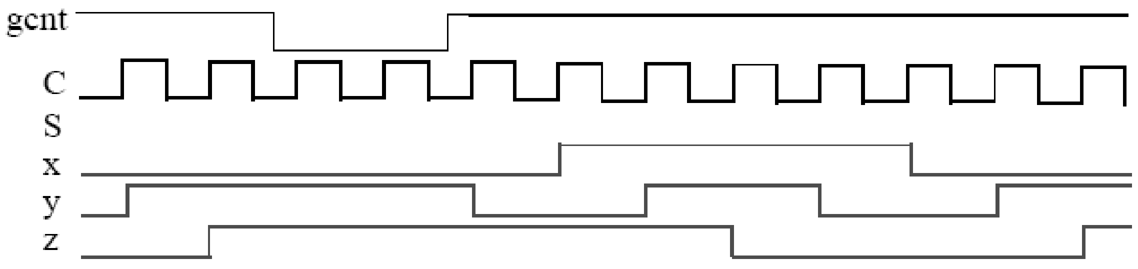
3.28


Inputs: B,R, Outputs: s1,s0
Time2
s1s0=00
Alarm
s1s0=01
Alarm2
s1s0=01
Stopwatch
s1s0=10
Stopwatch2
s1s0=10
Date
s1s0=11
Date2
s1s0=11
Time
s1s0=00
R+B'
R'B
R'B'
R

3.29


Inputs: gcnt
Outputs: x, y, z
A
xyz=000
B
xyz=010
C
xyz=011
D
xyz=001
E
xyz=101
F
xyz=111
G
xyz=110
H
xyz=100
gcnt
gcnt'

3.30


gcnt
C
S
x
y
z

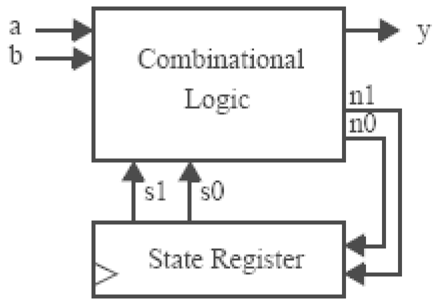
3.32

a) 2 bits
b) 3 bits
c) 4 bits
d) 5 bits
e) 10 bits

3.39

Step 1 - Capture the FSM
The appropriate FSM is given above.

Step 2 - Create the architecture



Step 3 - Encode the states
A straightforward encoding is A=00, B=01, C=10, D=11.
Step 4 - Create the state table

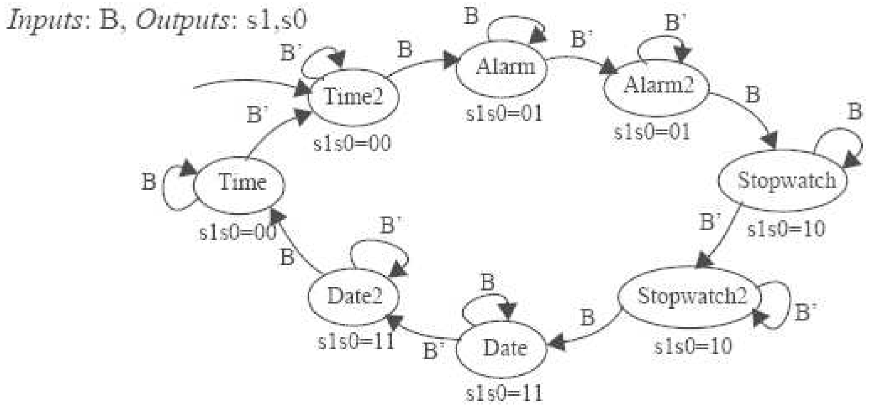
Inputs				Outputs		
s1	s0	a	b	n1	n0	y
0	0	0	0	1	0	0
0	0	0	1	0	1	0
0	0	1	0	0	0	0
0	0	1	1	0	0	0
0	1	0	0	0	1	1
0	1	0	1	0	1	1
0	1	1	0	1	0	1
0	1	1	1	1	0	1
1	0	0	0	1	0	1
1	0	0	1	1	1	1
1	0	1	0	1	0	1
1	0	1	1	1	1	1
1	1	0	0	0	0	0
1	1	0	1	0	0	0
1	1	1	0	0	0	0

1 1 1 1 0 0 0

Step 5 - Implement the combinational logic
n1 = s1’s0’a’b’ + s1’s0a + s1s0’
n0 = s1’s0’a’b + s1’s0a’ + s1s0’b
y = s1’s0 + s1s0’
Note: The above equations can be minimized further.

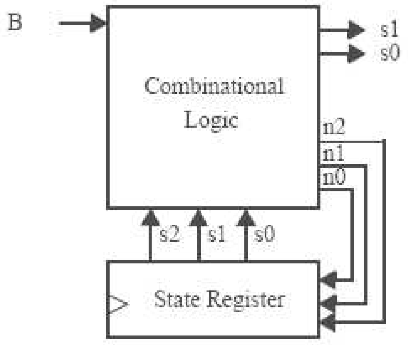
3.41

Step 1 - Capture the FSM



The FSM was created during Exercise 3.27.

Step 2 - Create the architecture



Step 3 - Encode the states
A straightforward encoding is Time2=000, Alarm=001, Alarm2=010, Stopwatch=011, Stopwatch2=100, Date=101, Date2=110, Time=111.

Step 4 - Create the state table

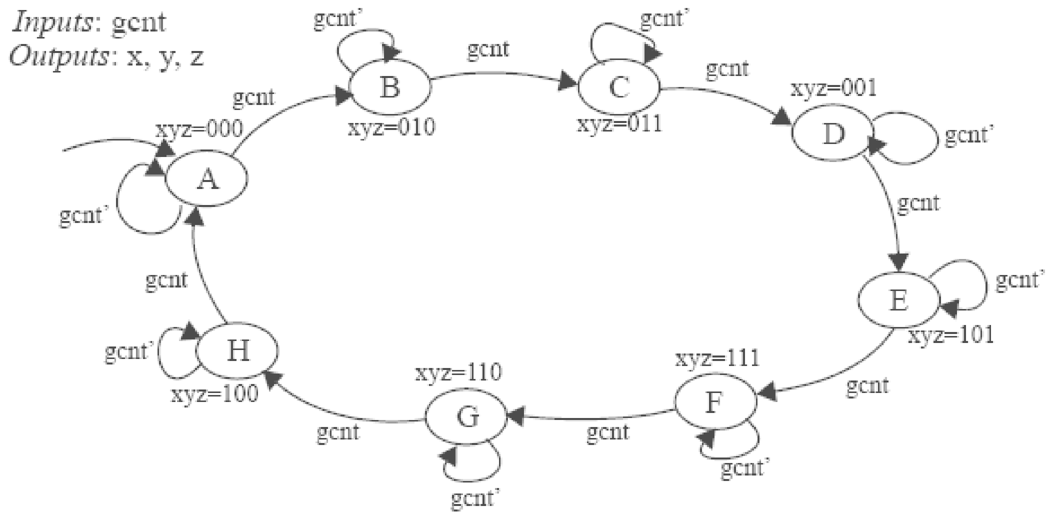
Inputs				Outputs				
s2	s1	s0	B	n2	n1	n0	s1	s0
0	0	0	0	0	0	0	0	0
0	0	0	1	0	0	1	0	0
0	0	1	0	0	1	0	0	1
0	0	1	1	0	0	1	0	1
0	1	0	0	0	1	0	0	1
0	1	0	1	0	1	1	0	1
0	1	1	0	1	0	0	1	0
0	1	1	1	0	1	1	1	0
1	0	0	0	1	0	0	1	0
1	0	0	1	1	0	1	1	0
1	0	1	0	1	0	1	1	1
1	0	1	1	1	1	0	1	1
1	1	0	0	1	1	0	1	1
1	1	0	1	1	1	1	1	1
1	1	1	0	0	0	0	0	0
1	1	1	1	1	1	1	0	0

Step 5 - Implement the combinational logic

n2 = s2’s1s0B’ + s2s1’ + s2s0’ + s2B
n1 = s1s0’ + s1B + s2s0B + s2’s1’s0B’
n0 = s0’B + s2’B + s1B + s2s1’s0B’
s1 = s2s0’ + s2s1’ + s2’s1s0
s0 = s1 XOR s0

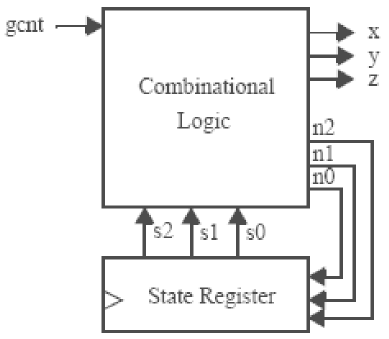
3.42

Step 1 - Capture the FSM



The FSM was created during Exercise 3.29.

Step 2 - Create the architecture



Step 3 - Encode the states

A straightforward encoding is A=000, B=001, C=010, D=011, E=100, F=101, G=110, H=111.

Step 4 - Create the state table

	Inputs				Outputs					
	s2	s1	s0	gcnt	n2	n1	n0	x	y	z
A	0	0	0	0	0	0	0	0	0	0
	0	0	0	1	0	0	1	0	0	0
B	0	0	1	0	0	0	1	0	1	0
	0	0	1	1	0	1	0	0	1	0
C	0	1	0	0	0	1	0	0	1	1
	0	1	0	1	0	1	1	0	1	1
D	0	1	1	0	0	1	1	0	0	1
	0	1	1	1	1	0	0	0	0	1
E	1	0	0	0	1	0	0	1	0	1
	1	0	0	1	1	0	1	1	0	1
F	1	0	1	0	1	0	1	1	1	1
	1	0	1	1	1	1	0	1	1	1
G	1	1	0	0	1	1	0	1	1	0
	1	1	0	1	1	1	1	1	1	0
H	1	1	1	0	1	1	1	1	0	0
	1	1	1	1	0	0	0	1	0	0

Step 5 - Implement the combinational logic

n2 = s2’s1s0gcnt + s2s1’ + s2s1s0’ + s2s1s0gcnt’

n1 = s2’s1’s0gcnt + s2’s1s0’ + s2’s1s0gcnt’ + s2s1’s0gcnt + s2s1s0’ + s2s1s0gcnt’

n0 = s2’s1’s0’gcnt + s2’s1’s0gcnt’ + s2’s1s0’gcnt + s2’s1s0gcnt’ + s2s1’s0’gcnt + s2s1’s0gcnt’ + s2s1s0’gcnt + s2s1s0gcnt’

x = s2

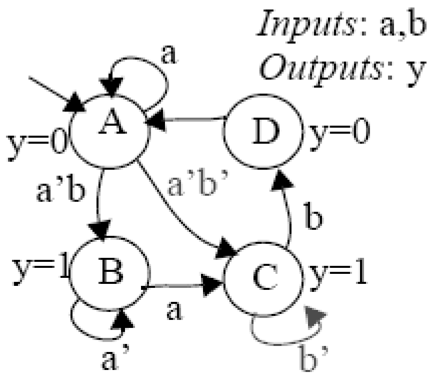
y = s2’s1’s0 + s2’s1s0’ + s2s1’s0 + s2s1s0’

z = s2’s1 + s2s1’

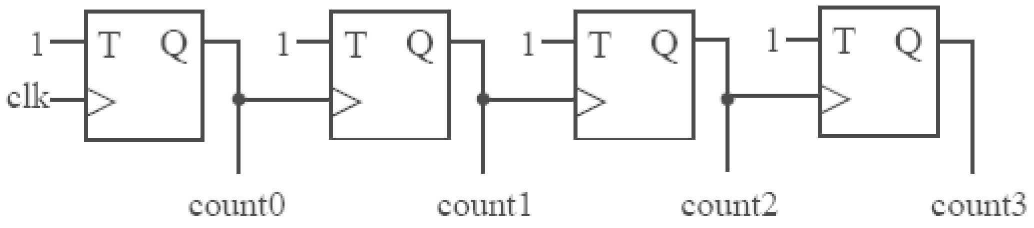
Note: The above equations can be minimized further.

3.45

If we AND each pair of transitions with each other in state *A*, we notice one pair does not evaluate to 0: a*a'b. This shoes that more than one condition can be true simultaneously.
Furthermore, if we OR all the conditions from state *C*, we notice that the expression does not evaluate to 1: b. This shoes that there may be a combination of inputs in which one condition from state *C* is not true.
We can address both of these problems with the following changes:



3.48



1
clk
T
Q
count0
count1
count2
count3