Design of Digital System (CS-201): Mini Project

Design Submission

Automated Reception Desk for Hospitals

Submitted to:

Prof. B. R. Chandavarkar

Design of Digital System (CS-201)

Asst. Prof., Computer Science Engineering Dept.

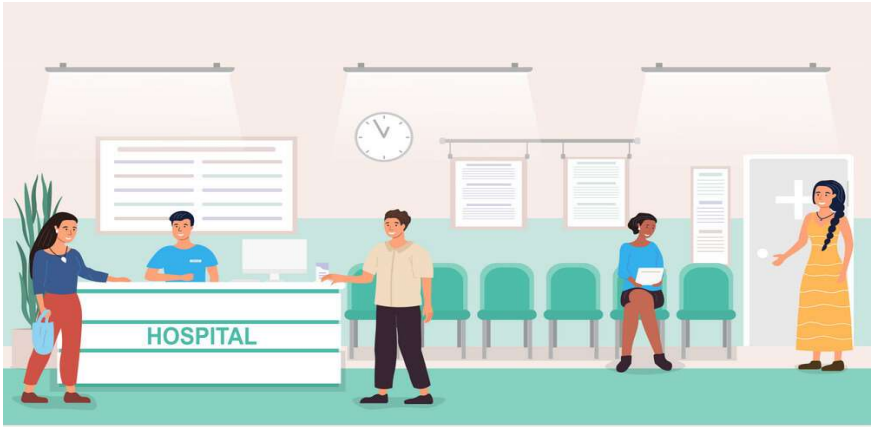
Team:

Sai Anil Poreddiwar (211CS148)

saiporeddiwar.211cs148@nitk.edu.in

Shreekara Rajendra (211CS151)

Shreekararajendra.211cs151@nitk.edu.in



PROBLEM DESCRIPTION:

We have made an attempt to automate the process of allotment of the doctors according to the incoming medical requirement of the patients.

The patients input the type of query which is fed into the system and the system generates the output informing the patient which doctor to consult. However, if none of the required doctors are vacant, the system generates the wait signal and the input request could be fed again after a certain time interval.

Here we have tried to depict our approach using a small-scale scenario:

Hospital description:

-> Equipped with two doctors namely S1 and S2

-> The incoming patient could have 4 types of queries/medical requirement which is numbered from 0 to 3.

-> Query no. 0 can only be solved by doctor S1.

-> Query no. 1 can be solved both by doctors S1 and S2.

-> Query no. 2 can be solved both by doctors S1 and S2.

-> Query no. 3 can be solved by only doctor S2.

-> If none of required doctor(s) are vacant, the patient is required to wait.

DRY RUN/ EXAMPLE:

Initially, S1 and S2 are vacant

Incoming patient query no. is 0. Doctor S1 allotted and occupied for 15 mins.

Next incoming patient query no. is 1. Doctor S2 allotted and occupied for 15 mins.

Doctor S1 becomes free.

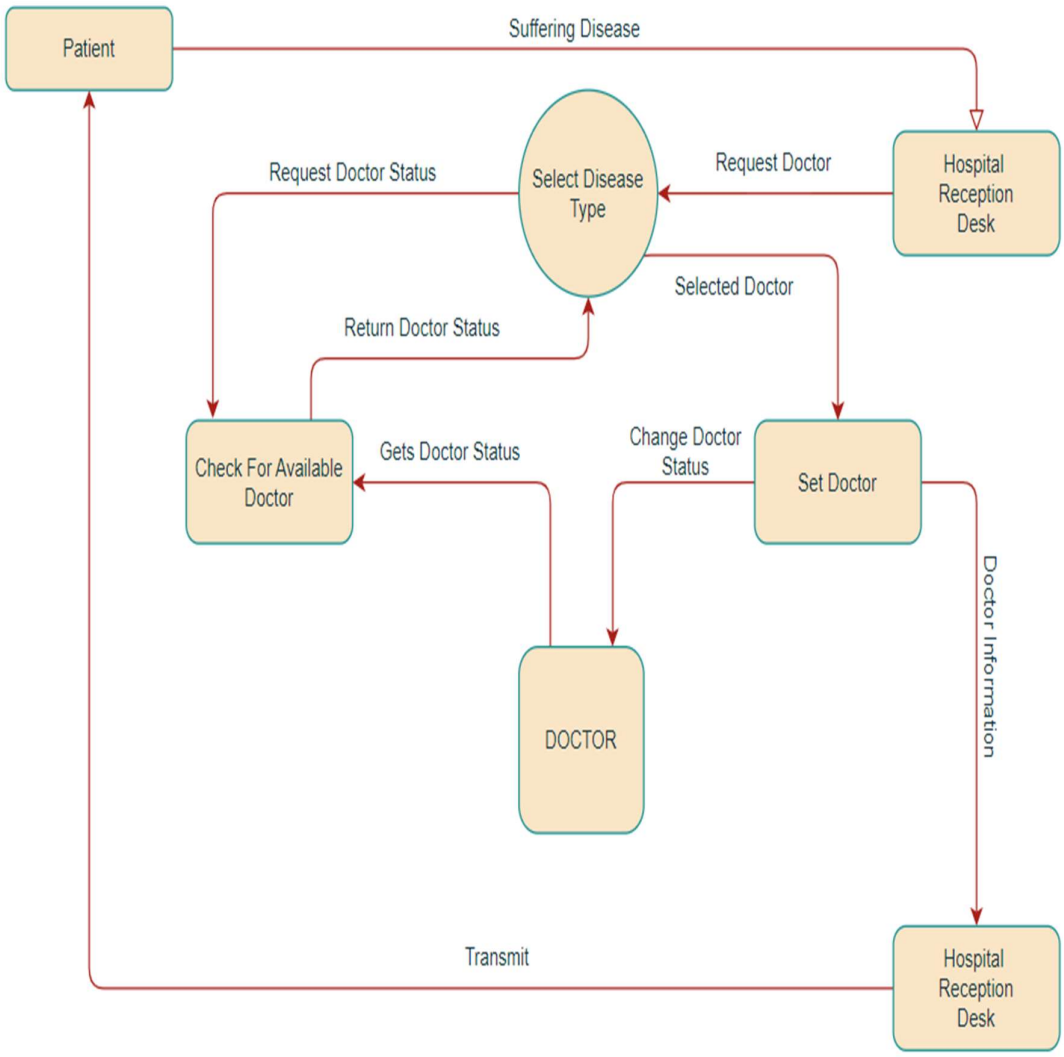
Next incoming patient query no. is 3 (can only be solved by S2). Hence requested to wait. His request will be reprocessed after 15 mins.

NOTE: If both S1 and S2 are vacant and can solve the current incoming query, then S1 is preferred over S2.

ASSUMPTIONS:

1. The input requests from the patients are assumed to be sequential, i.e., our system can't handle two requests at once.
2. Unidirectional input from patient.
3. The consulting time for every patient is considered to be the same.

FLOWCHART:



COMPONENTS REQUIRED:

1. Basic logic gates
2. Flip flops (used as register)
3. 4 bit counter

IMPLEMENTATION:

INPUT:

The input will be a four-bit integer out of which 2 bits represent the query no. of the incoming patient and the remaining 2 bits is a system generated input which represents the availability of the suitable consultant.

A: MSB of the query number

B: LSB of the query number

For example: A = 1, B = 1: represents query numbered 3

S1(t), S2(t) are present states indicating the availability of the doctors. It has been implemented using flip flops.

OUTPUT:

The output will be a two-bit integer which informs the patient whether the suitable doctor is available or if he/she is required to wait. In the former case, two bits represent the doctor to consult (A or B). In the latter case, the wait signal is given, after the waiting time, the input is requested again.

X, Y: denote if the doctor S1 or S2 have been allotted respectively

X: It will be 1 only if the doctor S1 has been allotted.

Y: It will be 0 only if the doctor S2 has been allotted.

Note: Both X, Y cannot be 1

If none of the required doctors are available, then the wait signal is generated i.e., X = 0 and Y = 0.

S1(t+1), S2(t+1) are the next or updated states after the allotment which indicates the availability of the doctors for the next input request.

➔ The truth table below depicts this outer layout of the system.

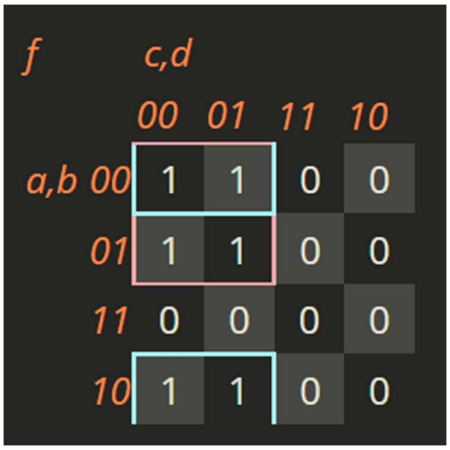
TRUTH TABLE:

A	B	S1(t)	S2(t)	X	Y	S1(t+1)	S2(t+1)
0	0	0	0	1	0	1	0
0	0	0	1	1	0	1	1
0	0	1	0	0	0	1	0
0	0	1	1	0	0	1	1
0	1	0	0	1	0	1	0
0	1	0	1	1	0	1	1
0	1	1	0	0	1	1	1
0	1	1	1	0	0	1	1
1	0	0	0	1	0	1	0
1	0	0	1	1	0	1	1
1	0	1	0	0	1	1	1
1	0	1	1	0	0	1	1
1	1	0	0	0	1	0	1
1	1	0	1	0	0	0	1
1	1	1	0	0	1	1	1
1	1	1	1	0	0	1	1

KARNAUGH MAPS (K-MAPS)

Note: in the K-maps, the variables a, b, c, d correspond to A, B, S1(t), S2(t)

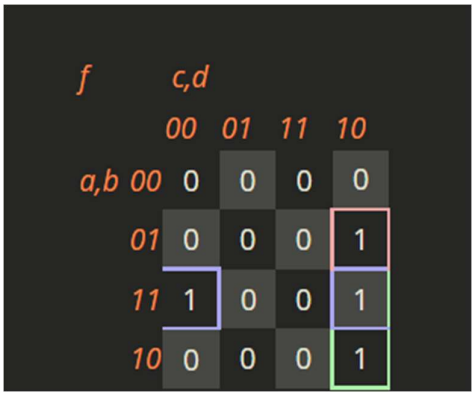
X:



BOOLEAN EXPRESSION

X = A' S1(t)' + B' S1(t)'

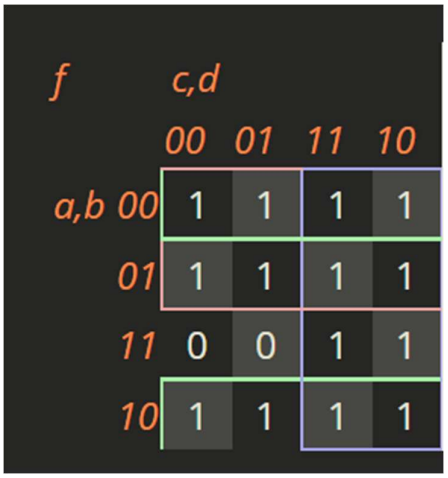
Y:



BOOLEAN EXPRESSION

Y = B S1(t) S2(t)' + A S1(t) S2(t)' + A B S2(t)'

S1(t+1):

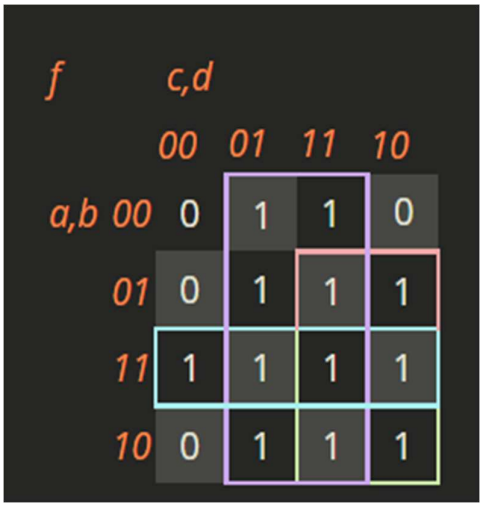


f	c,d			
	00	01	11	10
a,b 00	1	1	1	1
01	1	1	1	1
11	0	0	1	1
10	1	1	1	1

BOOLEAN EXPRESSION

S1(t+1) = A' + B' + S1(t)

S2(t+1):



f	c,d			
	00	01	11	10
a,b 00	0	1	1	0
01	0	1	1	1
11	1	1	1	1
10	0	1	1	1

BOOLEAN EXPRESSION

S2(t+1) = S2(t) + B S1(t) + A S1(t) + A

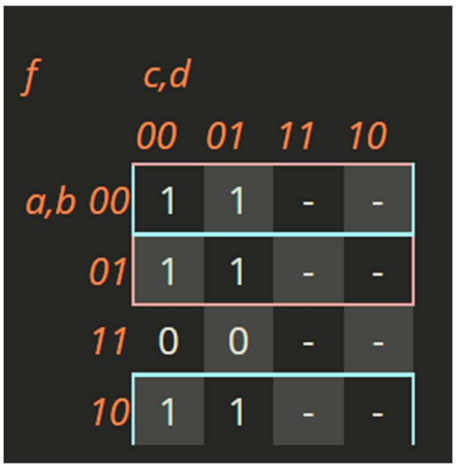
➔ The implementation of the doctor status i.e., S1(t) and S2(t) has been achieved using two JK flip flops.
Let the JK flip flop used for S1 be J1 K1
Let the JK flip flop used for S2 be J2 K2
The expression for the inputs to these flip flops has been obtained in terms of A, B, S1(t) and S2(t) by using the excitation table shown below.

A	B	S1(t)	S2(t)	S1(t+1)	S2(t+1)	J1	K1	J2	K2
0	0	0	0	1	0	1	X	0	X
0	0	0	1	1	1	1	X	X	0
0	0	1	0	1	0	X	0	0	X
0	0	1	1	1	1	X	0	X	0
0	1	0	0	1	0	1	X	0	X
0	1	0	1	1	1	1	X	X	0
0	1	1	0	1	1	X	0	1	X
0	1	1	1	1	1	X	0	X	0
1	0	0	0	1	0	1	X	0	X
1	0	0	1	1	1	1	X	X	0
1	0	1	0	1	1	X	0	1	X
1	0	1	1	1	1	X	0	X	0
1	1	0	0	0	1	0	X	1	X
1	1	0	1	0	1	0	X	X	0
1	1	1	0	1	1	X	0	1	X
1	1	1	1	1	1	X	0	X	0

K-MAPS:

Note: The variables a, b, c, d used in k-maps correspond to A, B, S1(t), S2(t).

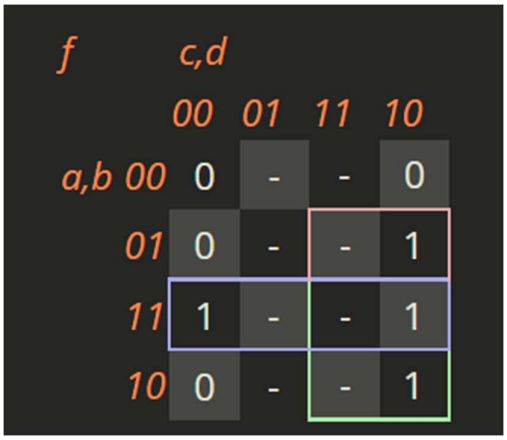
J1:



Boolean expression: **J1 = A' + B'**

K1: By simple analysis, K1 = 0

J2:



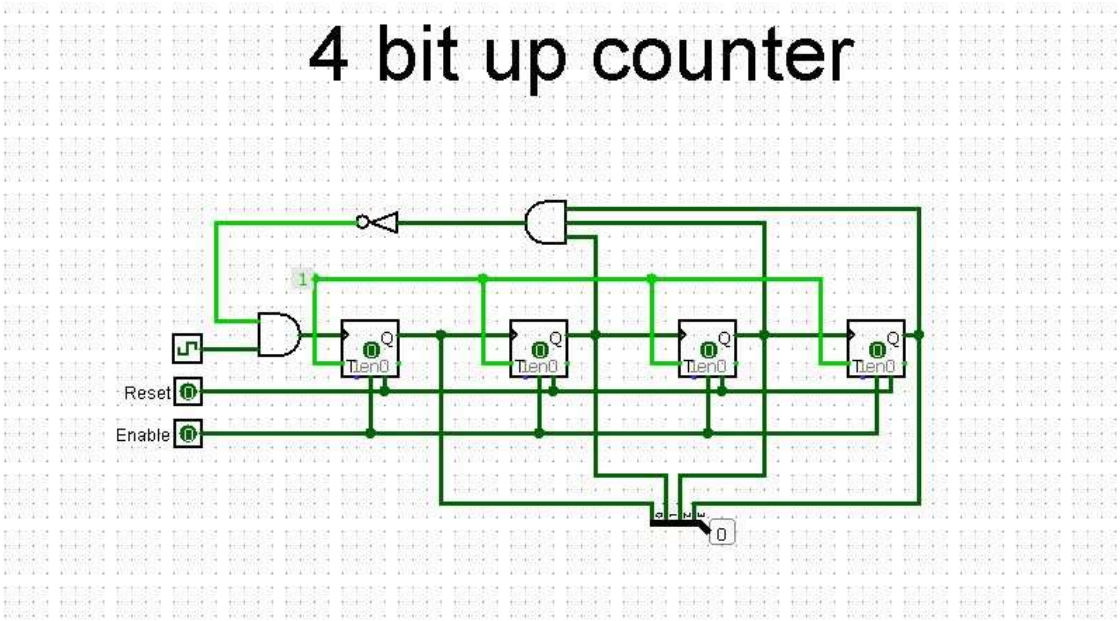
Boolean expression: **J2 = A B + B S1(t) + A S1(t)**

K2: By simple analysis, K2 = 0

COUNTER CIRCUIT

Once a particular doctor is occupied, we assume that the consultation time is constant for all patients and it is 15 mins.

Hence, a 4-bit counter is utilized for this purpose. Once the counter reaches 15, we reset the doctor status from 1 to 0



The above counter utilizes 4 T flip flops out of which first one is rising edge and last three are falling edge.

PSEUDO CODE:

//here a is status of doctor S1 and b is status of doctor S2

//count_a: counter for a

//count_b : counter for b

If(a==1) do count_a = count_a + 1;

If(b==1) do count_b = count_b + 1;

If(count_a==15) do a = 0;

If(count_b==15) do b = 0

REFERENCES:

1. Digital Design by Morris Mano
2. https://www.fpga4student.com/2016/11/verilog-code-for-trafficlight-system.html
3. https://www.electronics-tutorials.ws/sequential/seq_2.html
4. http://www.researchgate.net/
5. https://app.diagrams.net/?libs=general;flowchart
6. https://charlie-coleman.com/experiments/kmap/.