B.Tech. 3rd Semester

Design of Digital System (CS-201)

Mini-Project Final Report

On

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

DEPARTMENT OF COMPUTER SCIENCE AND ENNGINEERING

NATIONAL INSTITUTE OF TECHNOLOGY KARNATAKA

SURATHKAL, MANGLORE – 575025

Index

0.1 Abstract:

Hospitals are bustling hubs of healing where thousands of individuals seeking solace for their ailments flood the halls daily. From minor concerns to major afflictions, the diversity of medical needs and questions among patients and their loved ones is as vast as the human experience itself and In addition to scheduling the appointments, the reception desk plays a critical role in managing patient information and maintaining accurate records. They are responsible for collecting patient demographics, insurance information, and medical history. They also assist patients with check-in and check-out procedures, and handle billing and payments.

An automated system for managing appointments can improve the efficiency and accuracy of the scheduling process. It can also reduce the workload on receptionists, allowing them to focus on other tasks such as managing patient information and assisting patients with check-in and check-out procedures.

One of the key benefits of automating the appointment scheduling process is the ability to match patients with the most appropriate doctor based on their medical needs. For example, a patient with a broken bone would be scheduled with an orthopaedic specialist, while a

patient with a heart condition would be scheduled with a cardiologist. An automated system can also take into account the availability of the doctor and the patient's preferred time for the appointment.

Another benefit of an automated system is the ability to notify patients of their appointment details in a timely manner. This can be done via email or text message, reducing the need for patients to call the reception desk to confirm their appointment.

In short, Automating the reception process for booking appointment would streamline the process for both the hospital staff and patients, it would improve the matchmaking of patients to doctors, reduce errors and redundancy, and also provide patients with accurate and up-to-date information about their appointments.

0.2 Introduction:

The proposed automated system for managing appointments in the hospital would use a four-bit integer as input. The first two bits of the input would represent the query number of the incoming patient, which can be either 0, 1, 2, or 3. The last two bits of the input would be a system-generated input that represents the availability of the suitable consultant. The system would be designed to take into account the different types of queries and the availability of the doctors in order to assign the most appropriate consultant to the patient.

When the system receives the input, it would first check the query number of the incoming patient. If the query number is 0, the system would check if doctor A is available. If doctor A is available, the system would assign doctor A to the patient and send a two-bit output indicating that doctor A is available to consult. If doctor A is not available, the system would send a two-bit output indicating that the patient has to wait.

Similarly, if the query number is 3, the system would check if doctor B is available. If doctor B is available, the system would assign doctor B to the patient and send a two-bit output indicating that doctor B is available to consult. If doctor B is not available, the system would send a two-bit output indicating that the patient has to wait.

If the query number is 1 or 2, the system would check the availability of both doctor A and doctor B. If both doctors are available, the system would assign the patient to the first available doctor and send a two-bit output indicating which doctor is available to consult. If both doctors are not available, the system would send a two-bit output indicating that the patient has to wait.

It is important to mention that, the waiting time can be adjustable and decided by the hospital based on the current situation, patients queue, or other hospital policies. After the waiting time, the system would request the input again.

In summary, the proposed automated system would use a four-bit input to represent the query number of the incoming patient and the availability of the consultants, and would send a two-bit output indicating the availability of the suitable consultant or the wait signal. The system would be designed to match patients with the most appropriate doctor based on their medical needs and availability of the doctor, providing a more efficient and accurate scheduling process.

This system is designed to help patients receive the appropriate medical attention they need in a timely and efficient manner. By using a patient's medical query as input, the system is able to match patients with the doctors who are best equipped to address their specific needs. This can help to improve the overall quality of care that patients receive at the hospital, as patients are more likely to receive the right treatment from the right doctor.

In this small-scale scenario, the hospital is equipped with two doctors (S1 and S2) and can handle four types of medical queries, numbered 0 to 3. The system is designed so that Query 0 can only be solved by doctor S1, while Queries 1 and 2 can be solved by both doctors. Query 3 can only be solved by doctor S2. In this way, patients are automatically directed to the doctor who is best equipped to handle their specific medical requirement.

However, in the case that none of the required doctor(s) are available, the patient would be required to wait until a doctor becomes available. This wait time can be a negative aspect of the system, but it is a common practice in hospital when a specific doctor is not available at the moment. The system could improve this wait time by regularly updating the status of available doctors and providing estimated wait times to patients.

Overall, this system is an attempt to automate the process of allocating doctors to patients based on their medical needs. By doing so, the system aims to improve the efficiency and speed of allocating doctors, reducing wait times, and ensuring that patients receive the proper medical attention they need. This is especially important for patients who may be in critical condition and need prompt medical attention. This system is one of many initiatives that hospitals could take to improve patient experience and provide effective and timely healthcare.

0.3 Design:

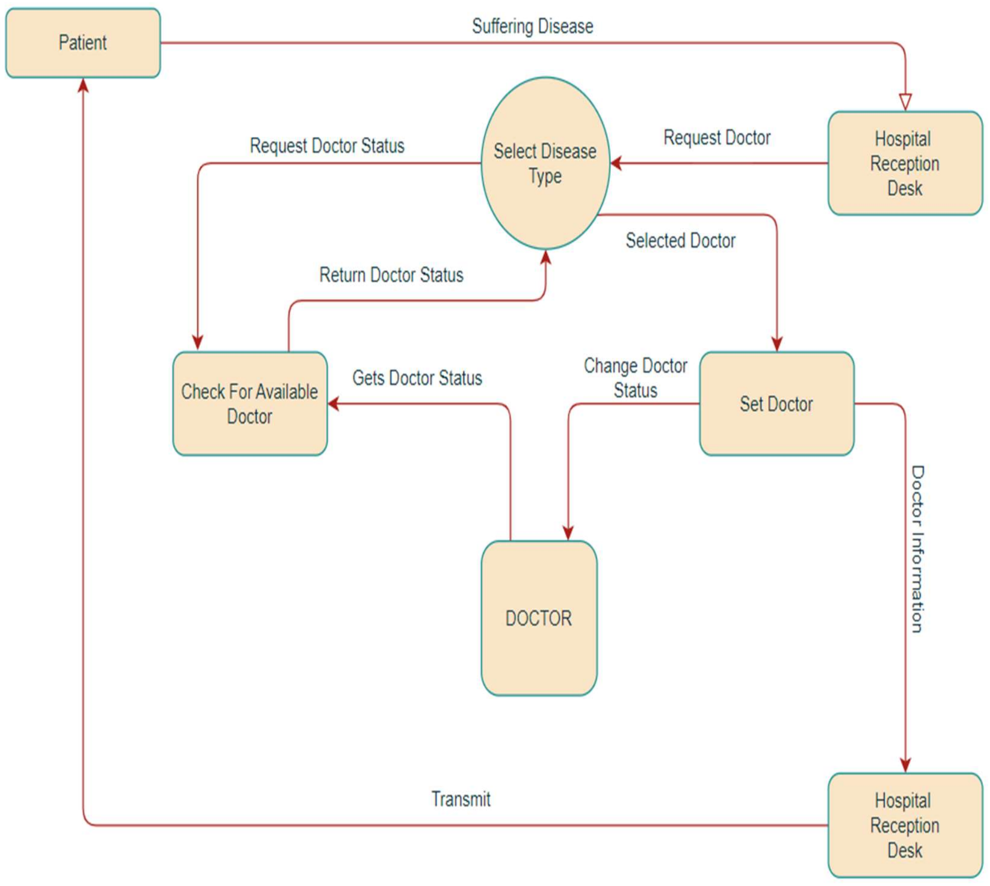
This system is designed to help patients receive the appropriate medical attention they need in a timely and efficient manner. By using a patient's medical query as input, the system is able to match patients with the doctors who are best equipped to address their specific needs. This can help to improve the overall quality of care that patients receive at the hospital, as patients are more likely to receive the right treatment from the right doctor.

In this small-scale scenario, the hospital is equipped with two doctors (S1 and S2) and can handle four types of medical queries, numbered 0 to 3. The system is designed so that Query 0 can only be solved by doctor S1, while Queries 1 and 2 can be solved by both doctors. Query 3 can only be solved by doctor S2. In this way, patients are automatically directed to the doctor who is best equipped to handle their specific medical requirement.

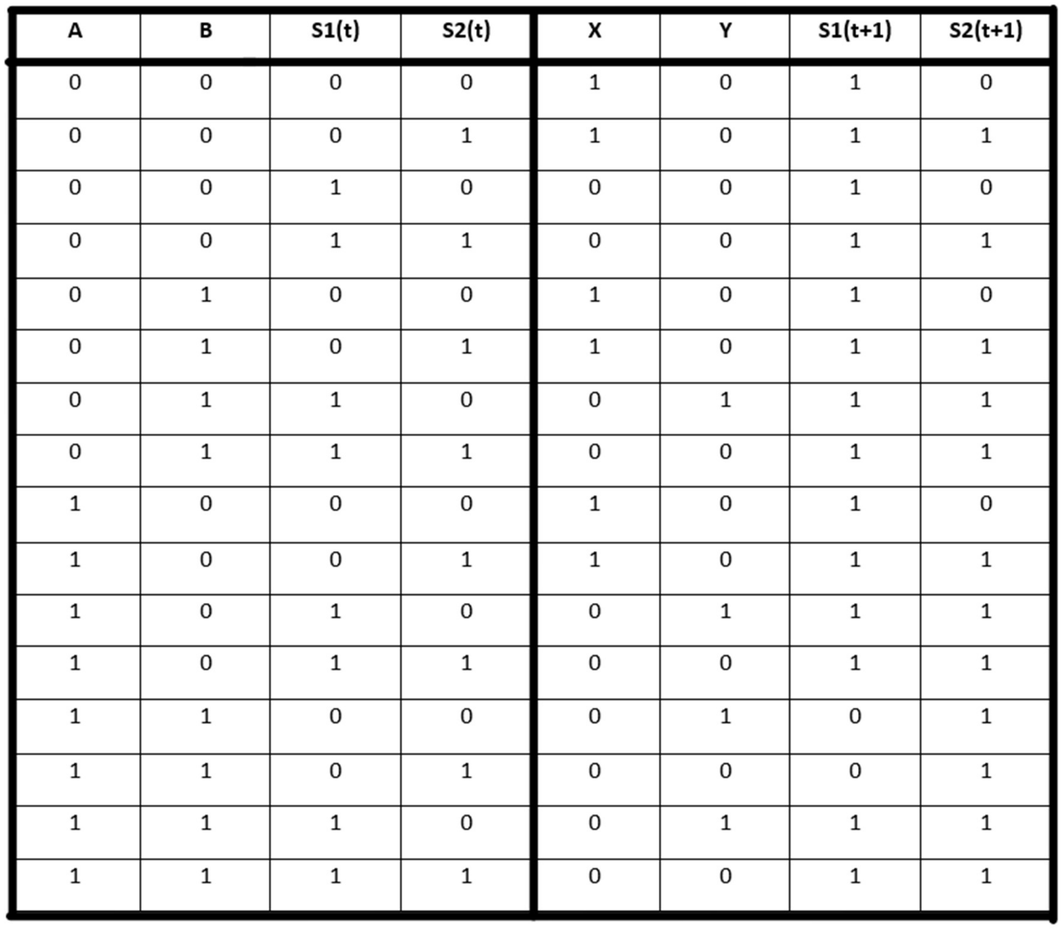
However, in the case that none of the required doctor(s) are available, the patient would be required to wait until a doctor becomes available. This wait time can be a negative aspect of the system, but it is a common practice in hospital when a specific doctor is not available at the moment. The system could improve this wait time by regularly updating the status of available doctors and providing estimated wait times to patients.

0.3.1 Flowchart:

"Efficiently match patients to the right doctor for their specific medical needs and improve the overall quality of care."



0.3.2 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

0.3.3 Simplification:

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

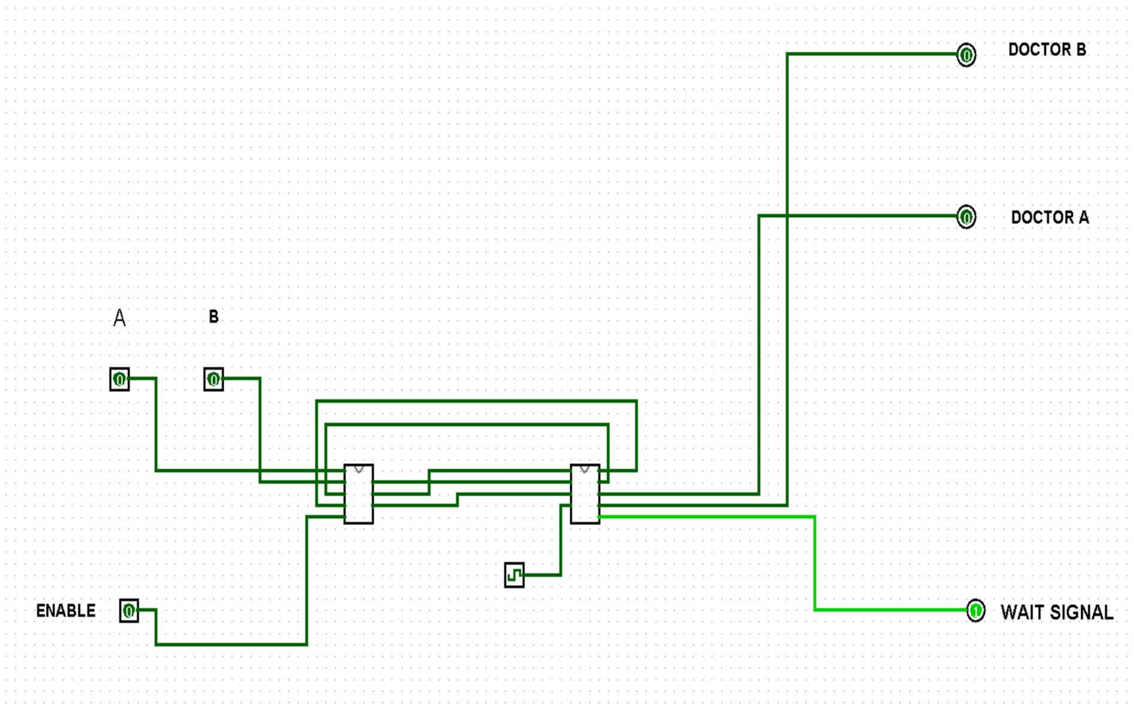
X: It will be 1 only if the doctor S1 has been allotted.

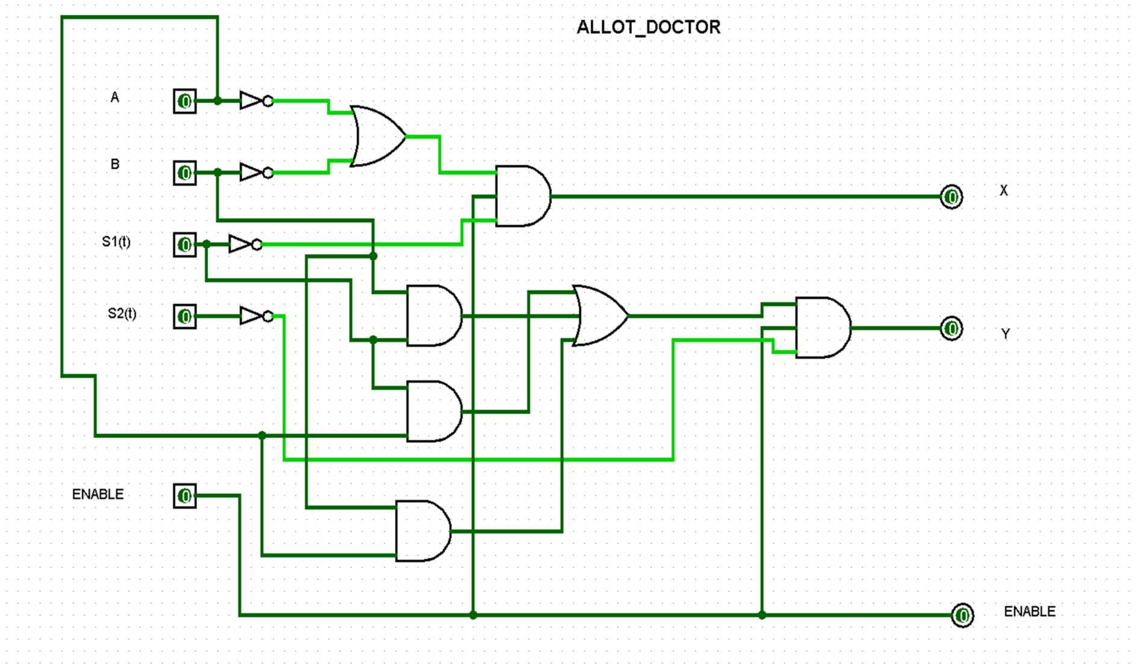
Y: It will be 0 only if the doctor S2 has been allotted.

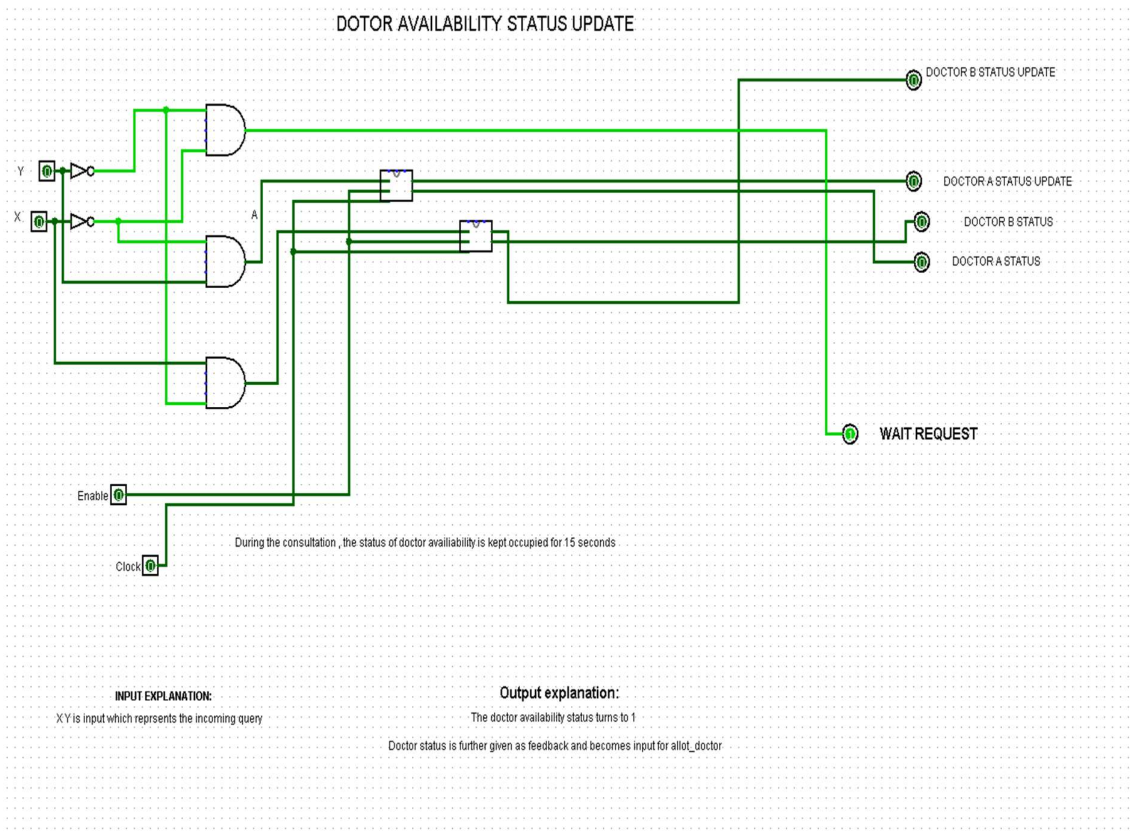
Note: Both X, Y cannot be 1 If none of the required doctors are available, then the wait signal is generated i.e., X = 0 and Y = 0.

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.

0.3.4 Logisim Circuit Diagram:







During the consultation , the status of doctor availiability is kept occupied for 15 seconds

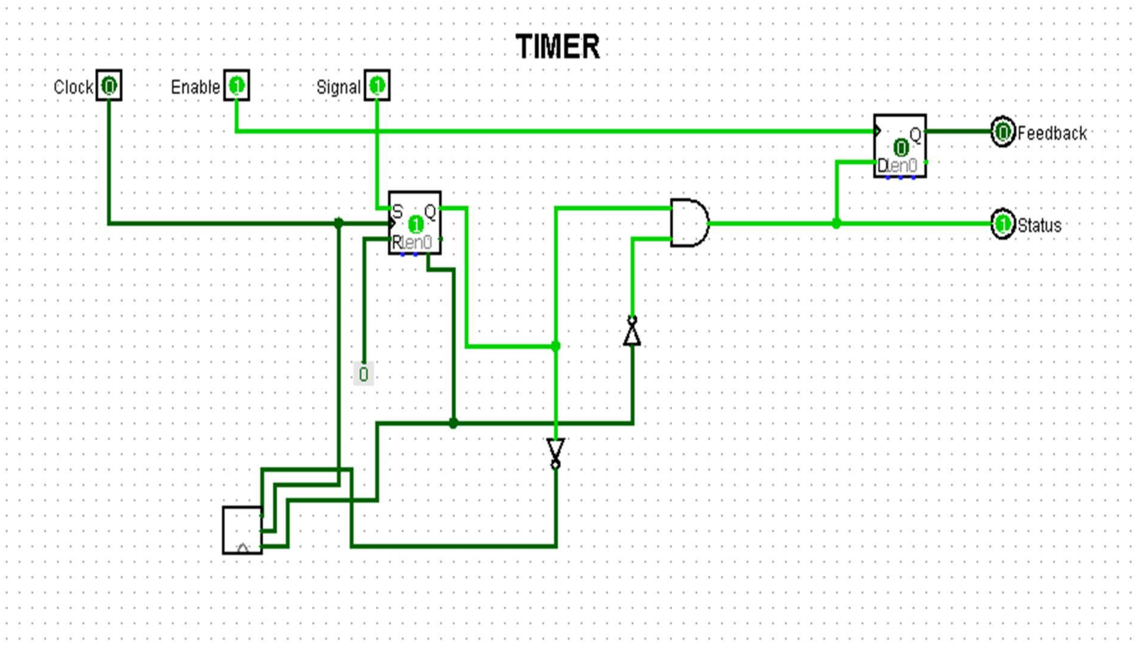
INPUT EXPLANATION:

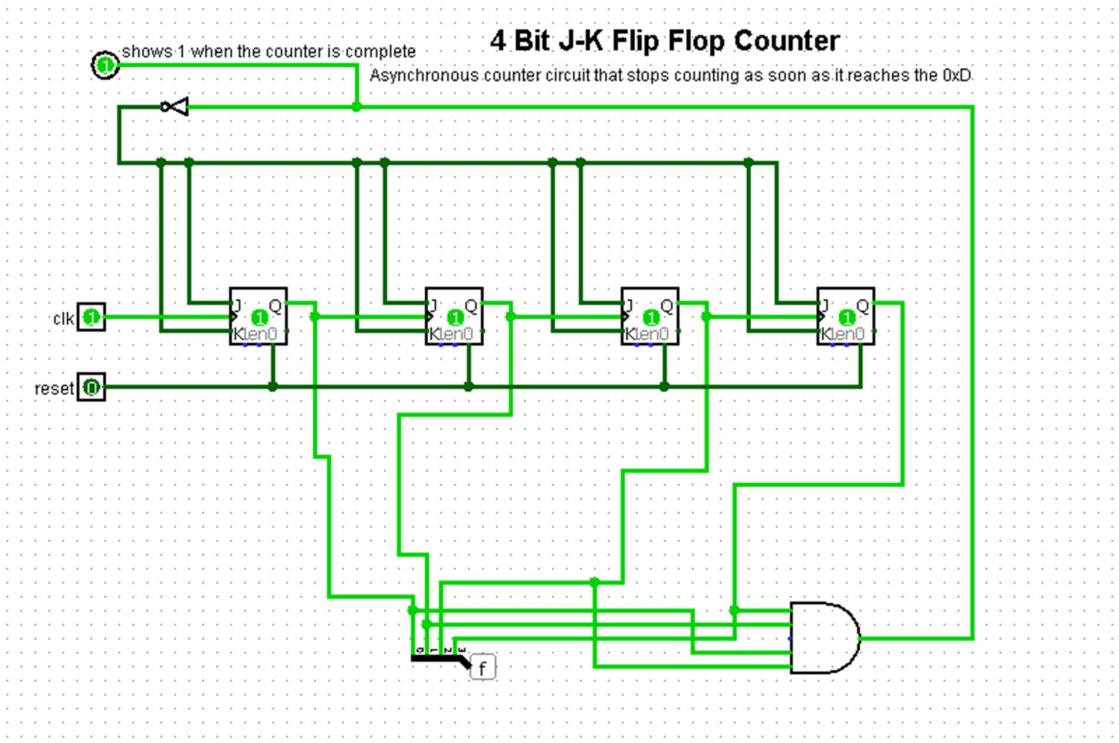
X Y is input which reprsents the incoming query

Output explanation:

The doctor availability status turns to 1

Doctor status is further given as feedback and becomes input for allot_doctor




4 Bit J-K Flip Flop Counter
shows 1 when the counter is complete
Asynchronous counter circuit that stops counting as soon as it reaches the 0xD
clk
reset
J
Q
K
len0
f

0.4 Verilog Code:

0.4.1 Gate Level:

Code File:gate_level.v

```
module gate_level(query,A,B,clk,start,msg);
    reg a,b,c,d,e,f,g,h;
    integer c_a = 0;
    integer c_b = 0;
    input [1:0]query;
    input clk,start;
    output reg [1:0]msg;
    output reg A,B;
    always@(negedge start)
    begin
        a=A;

        b=B;
        not(c,query[1]);
        not(d,query[0]);
        not(e,a);
        or(f,c,d);
```

```
            and(msg[1],f,e);

            //msg[1]= ((~query[1])|(~query[0]))&(~a);
            not(c,b);
            and(d,query[0],query[1]);
            and(e,a,query[0]);
            and(f,a,query[1]);
            or(g,d,e);
            or(h,g,f);
            and(msg[0],c,h);
            //msg[0]=
(~b)&((a&query[0])|(a&query[1])|(query[0]&query[1]));
        end
        always@(posedge clk)
        begin
            c_b=(msg==2'b01)?c_b+1:c_b+1;
            B=(c_b>5)?1:0;

            c_b=(c_b==15)?0:c_b;
        end
        always@(posedge clk)
        begin
            c_a=(msg==2'b10)?c_a+1:c_a+1;
            A=(c_a<10)?1:0;

            c_a=(c_a==15)?0:c_a;
        end
endmodule
```

Code File:gate_level_tb.v

```
`timescale 1ns/10ps
module testbench;
   reg start,clk;
   reg [1:0] query;
   wire [1:0] message;
   gate_level doctor(query,A,B,clk,start,message);
//Create an object of allot_doctor
   initial
   begin
      $monitor("                    15 seconds is equivalent to
300 time units\n                    message = 1 indicates
doctor 1 allotted and message = 2 indicates doctor 2
allotted. Message = 3 indicates wait signal ");
       clk = 1'b0;
       repeat (200);
```

```
        #10 clk = ~clk;                          //Create clock
    end
    initial
    begin
        $dumpfile("behavioural.vcd");
        $dumpvars(0,doctor);
       start = 1'b1;
        query = 2'b00;
        #10
        start = 1'b0;
        $monitor("TIME : %d || INCOMING QUERY: %d ||
MESSAGE: %d",$time, query, message);

        #100
        start = 1'b1;
        query = 2'b11;
        start = 1'b0;
        $monitor("TIME : %d || INCOMING QUERY: %d ||
MESSAGE: %d",$time, query, message);

        #100
        start = 1'b1;
        query = 2'b00;
        start = 1'b0;
        $monitor("TIME : %d || INCOMING QUERY: %d ||
MESSAGE: %d",$time, query, message);

        #20
        start = 1'b1;
        query = 2'b00;
        start = 1'b0;
        $monitor("TIME : %d || INCOMING QUERY: %d ||
MESSAGE: %d",$time, query, message);

        #200
        start = 1'b1;
        query = 2'b11;
        start = 1'b0;
        $monitor("TIME : %d || INCOMING QUERY: %d ||
MESSAGE: %d",$time, query, message);

    end
endmodule
```

0.4.2 Dataflow:

Code File:dataflow_model.v

```
module dataflow_model(query,A,B,clk,start,msg);
	reg a,b;
	integer c_a = 0;
	integer c_b = 0;
	input [1:0]query;
	input clk,start;
	output reg [1:0]msg;
	output reg A,B;
	always@(negedge start)
	begin
		a=A;

		b=B;

		msg[1]= ((~query[1])|(~query[0]))&(~a);
		msg[0]=
(~b)&((a&query[0])|(a&query[1])|(query[0]&query[1]));
	end
	always@(posedge clk)
	begin
```

```
            c_b=(msg==2'b01)?c_b+1:c_b+1;
            B=(c_b>5)?1:0;

            c_b=(c_b==15)?0:c_b;
        end
        always@(posedge clk)
        begin
            c_a=(msg==2'b10)?c_a+1:c_a+1;
            A=(c_a<10)?1:0;

            c_a=(c_a==15)?0:c_a;
        end
endmodule
```

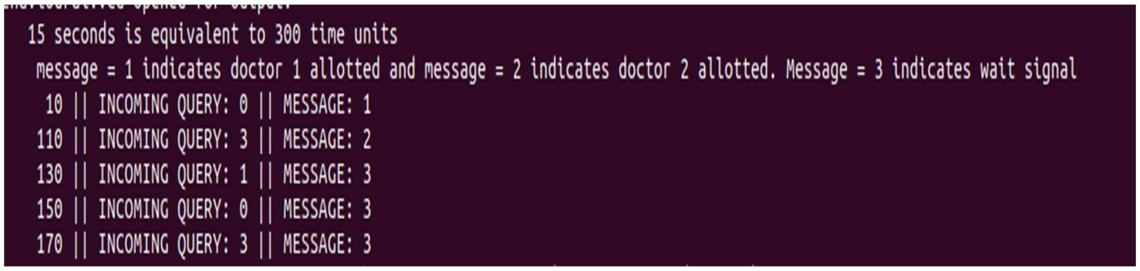
Code File:dataflow_model_tb.v

```
`timescale 1ns/10ps
module testbench;
    reg start,clk;
    reg [1:0] query;
    wire [1:0] message;
    dataflow_model doctor(query,A,B,clk,start,message);
//Create an object of allot_doctor
    initial
```

```
  begin
    $monitor("                    15 seconds is equivalent to
300 time units\n                    message = 1 indicates
doctor 1 allotted and message = 2 indicates doctor 2
allotted. Message = 3 indicates wait signal ");
     clk = 1'b0;
     repeat (200);
     #10 clk = ~clk;                              //Create clock
  end
  initial
  begin
     $dumpfile("behavioural.vcd");
     $dumpvars(0,doctor);
    start = 1'b1;
     query = 2'b00;
     #10
     start = 1'b0;
     $monitor("TIME : %d || INCOMING QUERY: %d ||
MESSAGE: %d",$time, query, message);


     #100
     start = 1'b1;
     query = 2'b11;
```

```
        start = 1'b0;
        $monitor("TIME : %d || INCOMING QUERY: %d ||
    MESSAGE: %d",$time, query, message);

        #100
        start = 1'b1;
        query = 2'b00;
        start = 1'b0;
        $monitor("TIME : %d || INCOMING QUERY: %d ||
    MESSAGE: %d",$time, query, message);

        #20
        start = 1'b1;
        query = 2'b00;
        start = 1'b0;
        $monitor("TIME : %d || INCOMING QUERY: %d ||
    MESSAGE: %d",$time, query, message);

        #200
        start = 1'b1;
        query = 2'b11;
        start = 1'b0;
        $monitor("TIME : %d || INCOMING QUERY: %d ||
    MESSAGE: %d",$time, query, message);


        end
    endmodule
```

Output:



```
[illegible]
  15 seconds is equivalent to 300 time units
   message = 1 indicates doctor 1 allotted and message = 2 indicates doctor 2 allotted. Message = 3 indicates wait signal
    10 || INCOMING QUERY: 0 || MESSAGE: 1
   110 || INCOMING QUERY: 3 || MESSAGE: 2
   130 || INCOMING QUERY: 1 || MESSAGE: 3
   150 || INCOMING QUERY: 0 || MESSAGE: 3
   170 || INCOMING QUERY: 3 || MESSAGE: 3
```

0.4.3 Behavioural Model:

Code File:allot_doctor.v

```
module allot_doctor(query,A,B,clk,start,message);
        input [1:0] query;
        input A,B,clk,start;
        output reg [1:0] message;
        reg a = 1'b0;
        reg b = 1'b0;
        integer c_a = 0;
        integer c_b = 0;
        always@(negedge start)
        begin
                if(query == 2'b00)
                        if(a == 0)
                        begin
                                message = 2'b01;
                                a = 1'b1;
                        end
                        else
                                message = 2'b11;
                if(query == 2'b01)
                        if(a == 0)
                        begin
                                message = 2'b01;
                                a = 1'b1;
                        end
                        else if(b == 0)
                        begin
                                message = 2'b10;
                                b = 1'b1;
                        end
                        else
                                message = 2'b11;
                if(query == 2'b10)
                        if(a == 0)
                        begin
                                message = 2'b01;
                                a = 1'b1;
                        end
                        else if(b == 0)
                        begin
                                message = 2'b10;
                                b = 1'b1;
                        end
                        else
                                message = 2'b11;
                if(query == 2'b11)
                        if(b == 0)
                        begin
                                message = 2'b10;
                                b = 1'b1;
                        end
                        else
                                message = 2'b11;
        end
        always@(posedge clk)
        begin
        if(a == 1)
            c_a = c_a + 1;
         if(b == 1)
            c_b = c_b + 1;
         if(c_a == 15)
            a = 0;
         if(c_b == 15)
            b = 0;
```

```
      end
endmodule
```

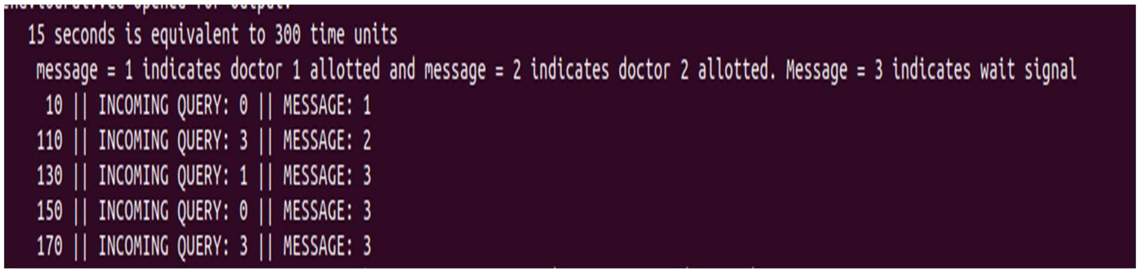
Code File:allot_doctor_tb.v

```
`timescale 1ns/10ps
module testbench;
   reg start,clk;
   reg [1:0] query;
   wire [1:0] message;
   allot_doctor doctor(query,A,B,clk,start,message);
//Create an object of allot_doctor
   initial
   begin
      $monitor("                    15 seconds is equivalent to
300 time units\n                    message = 1 indicates
doctor 1 allotted and message = 2 indicates doctor 2
allotted. Message = 3 indicates wait signal ");
       clk = 1'b0;
       repeat (100);
       #10 clk = ~clk;                              //Create clock
   end
   initial
   begin
```

```
        $dumpfile("behavioural.vcd");
        $dumpvars(0,doctor);
        #10
       start = 1'b1;
        query = 2'b00;
        start = 1'b0;
        $monitor("TIME : %d || INCOMING QUERY: %d ||
MESSAGE: %d",$time, query, message);


        #100
        start = 1'b1;
        query = 2'b11;
        start = 1'b0;
        $monitor("TIME : %d || INCOMING QUERY: %d ||
MESSAGE: %d",$time, query, message);


        #20
        start = 1'b1;
        query = 2'b01;
        start = 1'b0;
        $monitor("TIME : %d || INCOMING QUERY: %d ||
MESSAGE: %d",$time, query, message);
```

```
        #20
        start = 1'b1;
        query = 2'b00;
        start = 1'b0;
        $monitor("TIME : %d || INCOMING QUERY: %d ||
MESSAGE: %d",$time, query, message);

        #20
        start = 1'b1;
        query = 2'b11;
        start = 1'b0;
        $monitor("TIME : %d || INCOMING QUERY: %d ||
MESSAGE: %d",$time, query, message);

        #500
        start = 1'b1;
        query = 2'b11;
        start = 1'b0;
        $monitor("TIME : %d || INCOMING QUERY: %d ||
MESSAGE: %d",$time, query, message);

    end
endmodule
```

Output:



```
  15 seconds is equivalent to 300 time units
   message = 1 indicates doctor 1 allotted and message = 2 indicates doctor 2 allotted. Message = 3 indicates wait signal
    10 || INCOMING QUERY: 0 || MESSAGE: 1
   110 || INCOMING QUERY: 3 || MESSAGE: 2
   130 || INCOMING QUERY: 1 || MESSAGE: 3
   150 || INCOMING QUERY: 0 || MESSAGE: 3
   170 || INCOMING QUERY: 3 || MESSAGE: 3
```

0.5 Conclusions and Future Work:

In conclusion, automating the reception process for scheduling appointments in a hospital can bring many benefits for both patients and staff. It can improve the matchmaking of patients to doctors based on their medical needs, reduce errors and redundancy, and provide accurate and up-to-date information. It can also improve the efficiency of the scheduling process and reduce the workload on receptionists, allowing them to focus on other tasks.

The scenario presented in this write-up is an example of how an automated system could work in a hospital with two doctors and a limited set of queries. In practice, an automated system would need to be tailored to the specific needs of the hospital, taking into account the number of doctors, specialties, and types of queries.

As a **F**uture work, this system can be enhanced by using machine learning and natural language processing techniques to extract patient queries automatically, or even make a triage process. Also, incorporating the ability to handle different languages and multi-lingual customer service can be a great addition.

Additionally, integrating the system with advance electronic system can help to make the entire process seamless, and more data-driven as in Verilog is a slow Processing Language as compared to other Technologies available now a days so it can be modified above our expectations.

In any case, automating the reception process for scheduling appointments has the potential to greatly improve the patient experience and the efficiency of the hospital, and it's definitely worth considering for any healthcare organization.

0.6 References:

1. Digital Design by Morris Mano
2. https://www.fpga4student.com/2016/11/verilog-code-for-trafficlight-system.html
3. https://www.electronicstutorials.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/