# Mini Project (Design and Implementation in Logisim and HDL)
# Parking Management System

**Team Members:**

1. Konatham Naga Mukesh
   221CS132
   konathamnagamukesh.221cs132@nitk.edu.in
   8639550996
2. Lakkimsetti Sreyas
   221CS134
   mrlakkimsettisreyas.221cs134@nitk.edu.in
   9611275188
3. Tanay Praveen Shekokar
   221CS159
   tanayshekokar.221cs159@nitk.edu.in
   7022420056

## Abstract:

With the growing issues of urbanisation and limited parking resources, the development of a user-friendly Parking Management System is critical. This project provides a comprehensive system that combines parking slot occupancy detection and real-time slot availability display, with the user experience and parking facility efficiency as the top priorities.

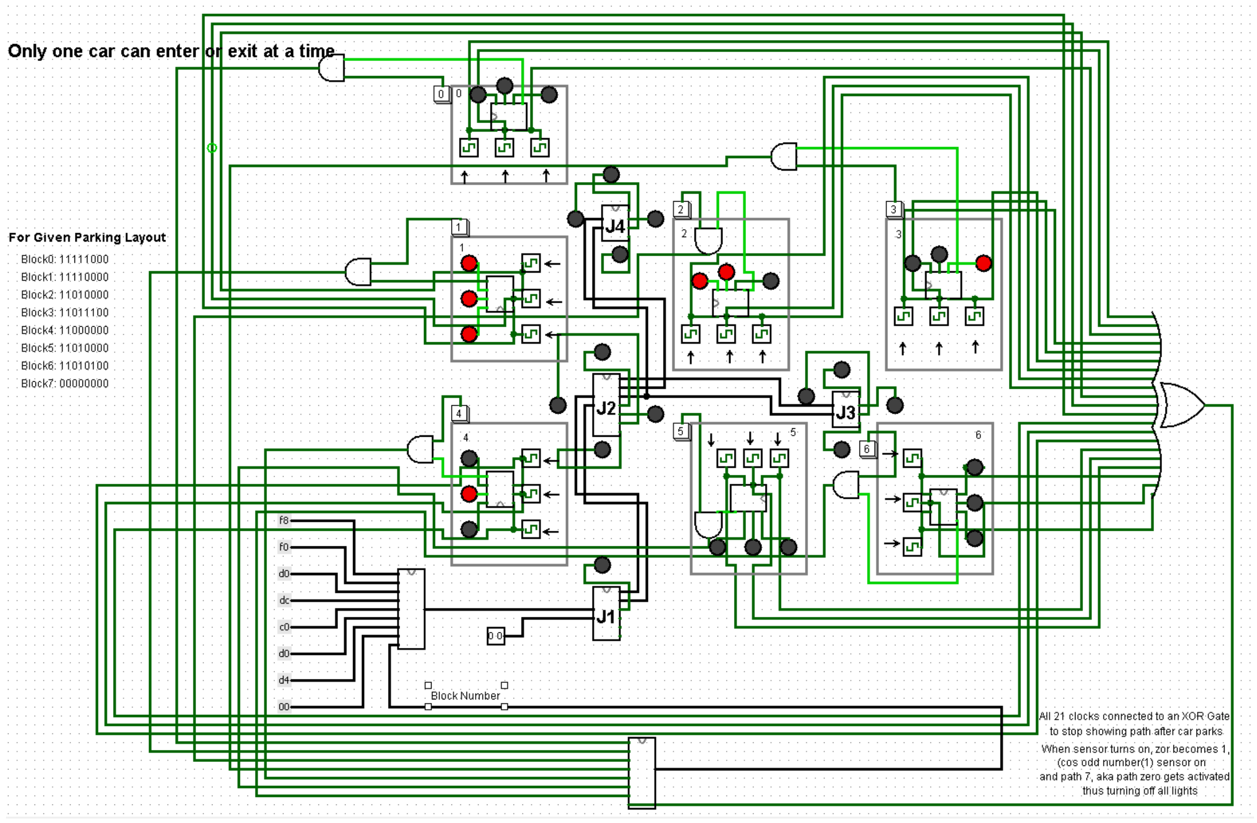
The goal behind this method is to address frequent urban difficulties such as traffic congestion and long parking search periods, which cause annoyance among vehicle owners. By utilising modern occupancy sensors, central control units, and digital displays, we hope to simplify the parking experience. It detects car presence reliably, updates real-time slot availability, and communicates this information to users, minimising the time and effort required to find a parking spot.

Our unique contribution is the seamless integration of these components, which improves customer pleasure while optimising space utilisation. The system keeps track of entry and exit times, which can be utilised for security and auditing. Furthermore, the obtained data can be used to guide future improvements and data-driven decision-making.

Our Parking Management System, by focusing on improving user experience and facility management, provides a realistic answer to the issues faced by urban parking, ultimately leading to more efficient and user-centric urban transportation solutions.

## Brief Description
## (Simulation to describe the user interaction, i.e. Output for various inputs)

Let us say this is how the screen looks originally



Only one car can enter or exit at a time

**For Given Parking Layout**

Block0: 11111000
Block1: 11110000
Block2: 11010000
Block3: 11011100
Block4: 11000000
Block5: 11010000
Block6: 11010100
Block7: 00000000

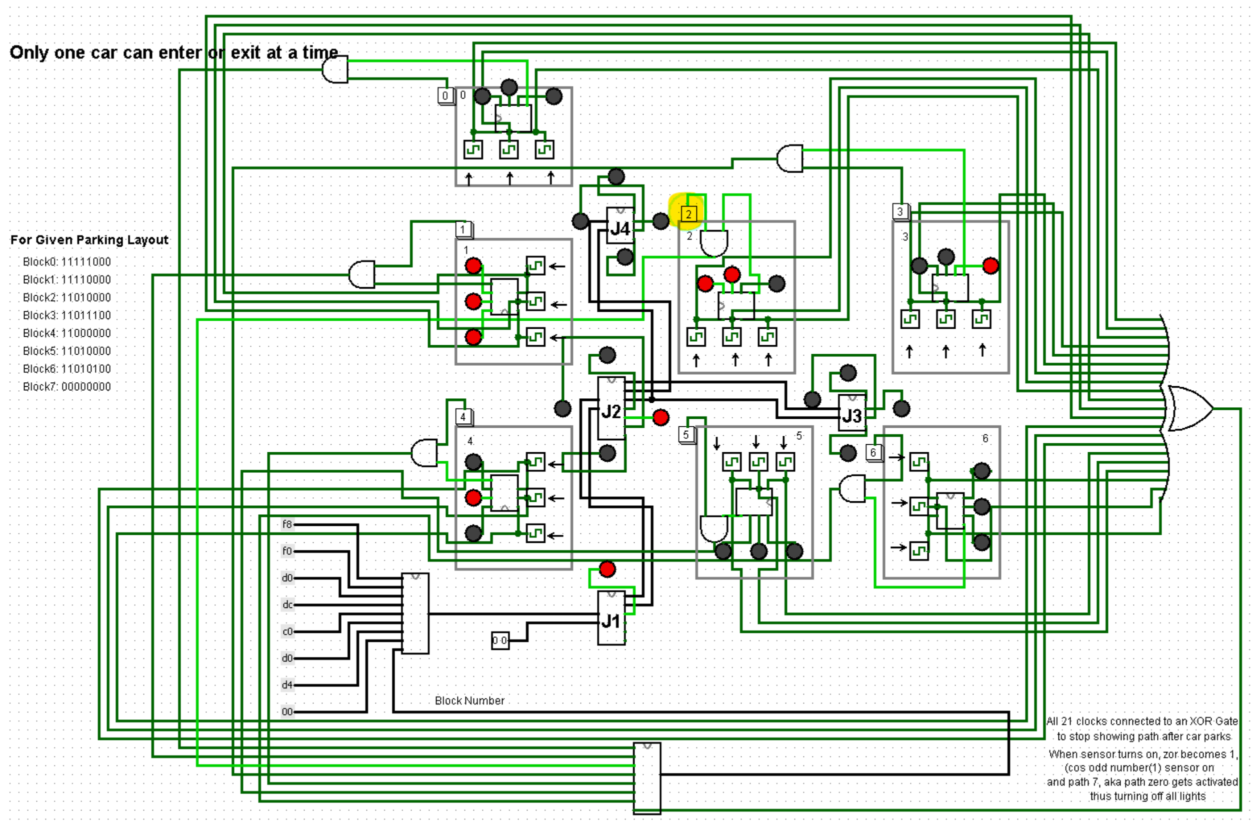
All 21 clocks connected to an XOR Gate
to stop showing path after car parks

When sensor turns on, zor becomes 1,
(cos odd number(1) sensor on
and path 7, aka path zero gets activated
thus turning off all lights

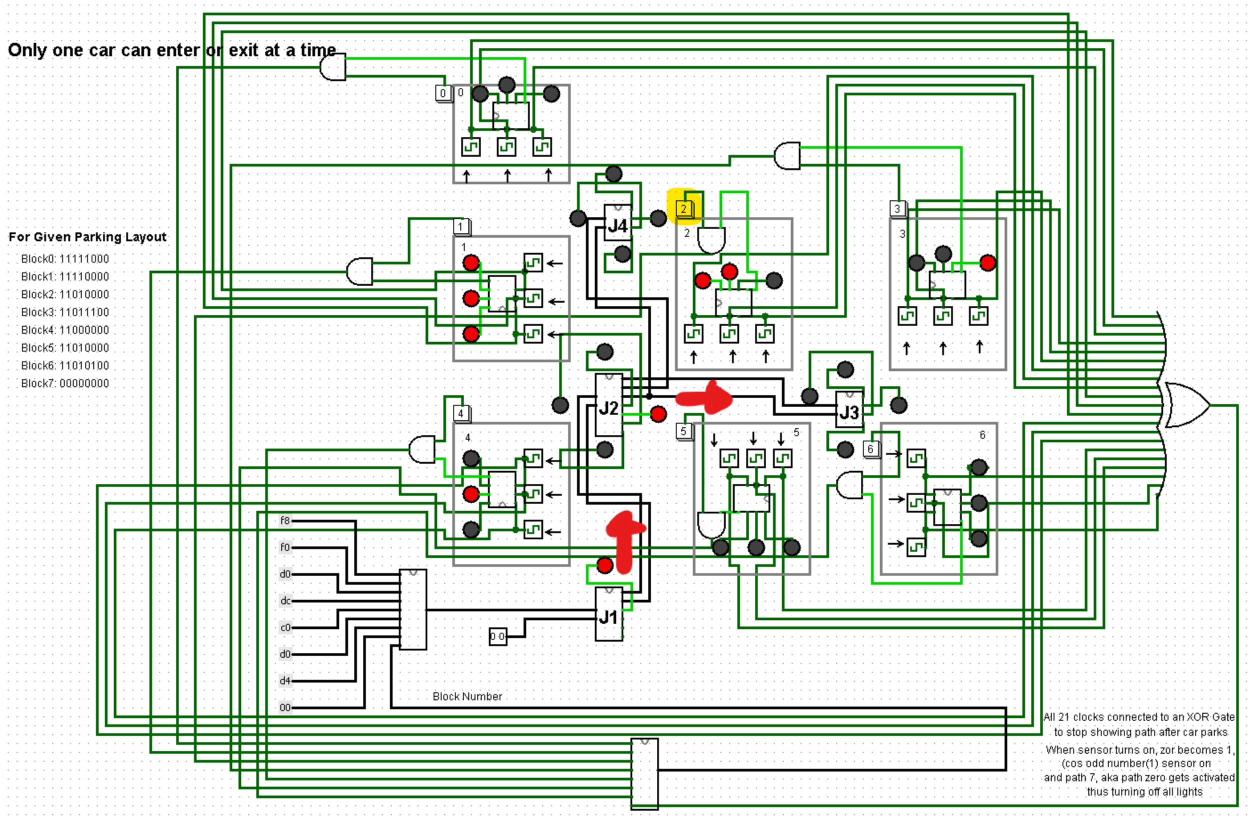
Each Block how a button labelled 0,1,2…. Depending on block number
Over here, each block has 3 LEDs
If the LED is switched on, it means that particular parking slot in that block is filled
Looking at this diagram, the user can choose which block he wants to park in

NOTE: Junction1 is the entrance

Let us say, the user wants to park in block2
He presses the button for block2


Only one car can enter or exit at a time
For Given Parking Layout
Block0: 11111000
Block1: 11110000
Block2: 11010000
Block3: 11011100
Block4: 11000000
Block5: 11010000
Block6: 11010100
Block7: 00000000
J4
J2
J3
J1
Block Number
f8
f0
d0
dc
c0
d0
d4
00
All 21 clocks connected to an XOR Gate
to stop showing path after car parks
When sensor turns on, zor becomes 1,
(cos odd number(1) sensor on
and path 7, aka path zero gets activated
thus turning off all lights

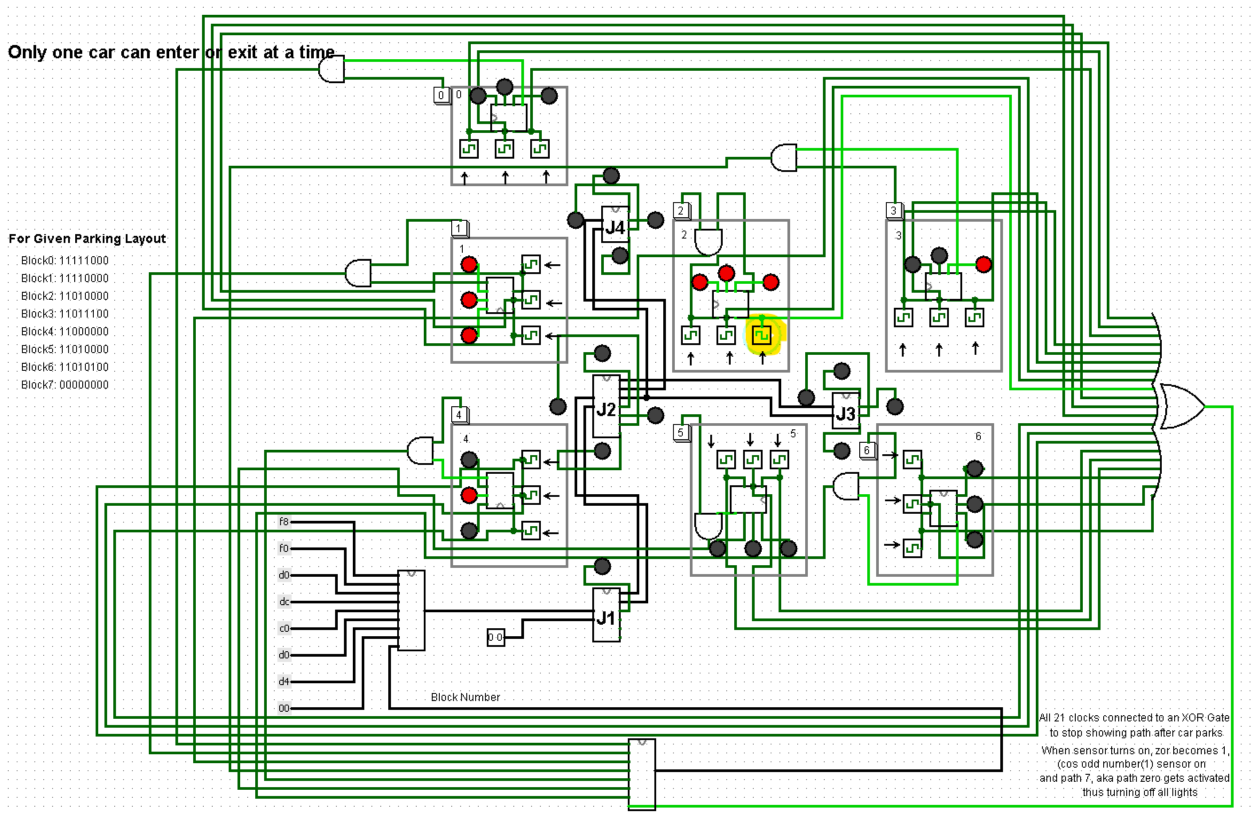
and releases it


Only one car can enter or exit at a time
For Given Parking Layout
Block0: 11111000
Block1: 11110000
Block2: 11010000
Block3: 11011100
Block4: 11000000
Block5: 11010000
Block6: 11010100
Block7: 00000000
J4
J2
J3
J1
Block Number
f8
f0
d0
dc
c0
d0
d4
00
All 21 clocks connected to an XOR Gate
to stop showing path after car parks
When sensor turns on, zor becomes 1,
(cos odd number(1) sensor on
and path 7, aka path zero gets activated
thus turning off all lights

**NOTE: Junction1 is the entrance**

Now, as we can see, the path to Junction2 is generated
At junction1, the north led is on, which means at junction1, the user must go straight
At junction2, the west led is on, which means at junction2, the user must go right
If the user follows the above instructions, he will reach the road from which he can enter the parking slot

Now, when the user enters the 3rd slot at block2
The third sensor(clock), which is present at the entrance of the slot turns on
Which means, the person is going to enter in that slot

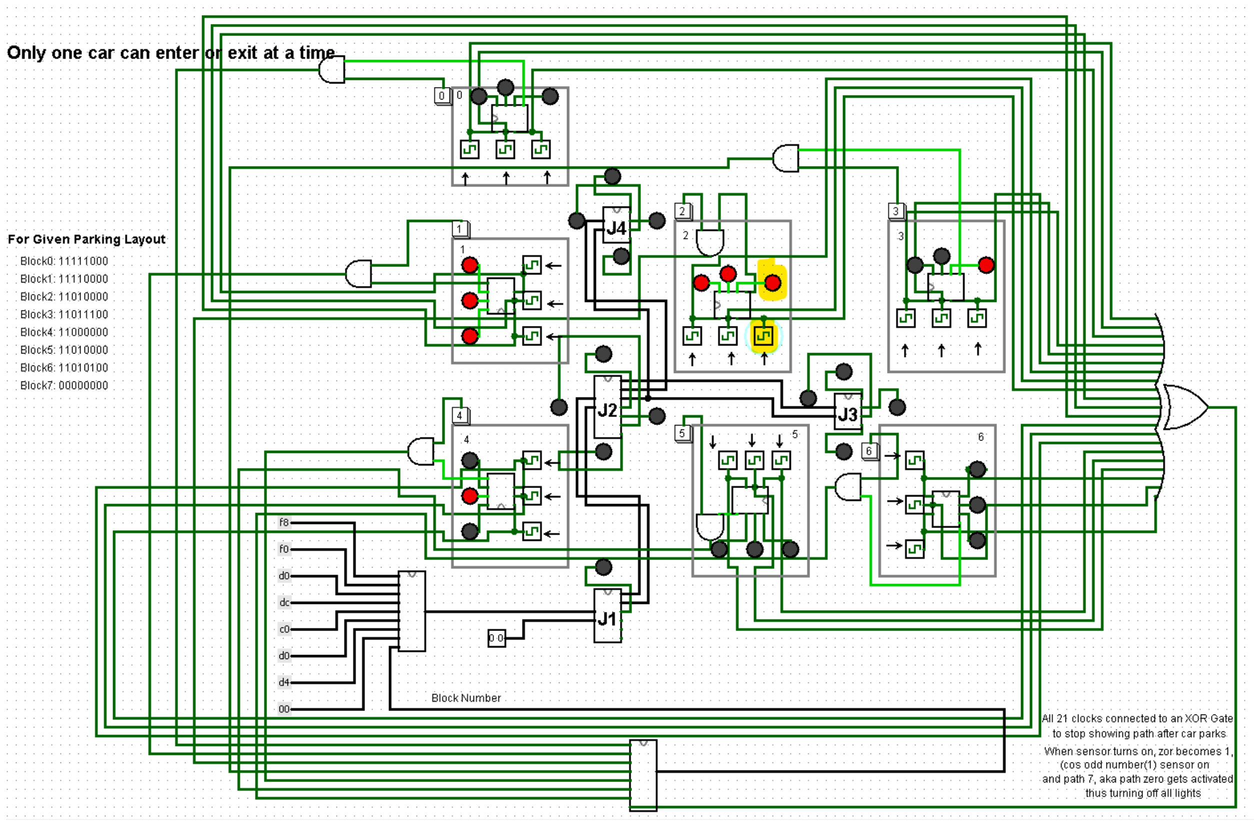


For Given Parking Layout
Block0: 11111000
Block1: 11110000
Block2: 11010000
Block3: 11011100
Block4: 11000000
Block5: 11010000
Block6: 11010100
Block7: 00000000

All 21 clocks connected to an XOR Gate
to stop showing path after car parks
When sensor turns on, zor becomes 1,
(cos odd number(1) sensor on
and path 7, aka path zero gets activated
thus turning off all lights

Now that this clock is turned on, the path towards block2 disappears
Bcos the car will be parked in this slot now, and thus doesnt need the directions anymore

Once the car parks in that slot
He will cross the entrance of that slot
Thus, the sensor(clock) for that slot becomes 0 again



Only one car can enter or exit at a time

For Given Parking Layout

Block0: 11111000
Block1: 11110000
Block2: 11010000
Block3: 11011100
Block4: 11000000
Block5: 11010000
Block6: 11010100
Block7: 00000000

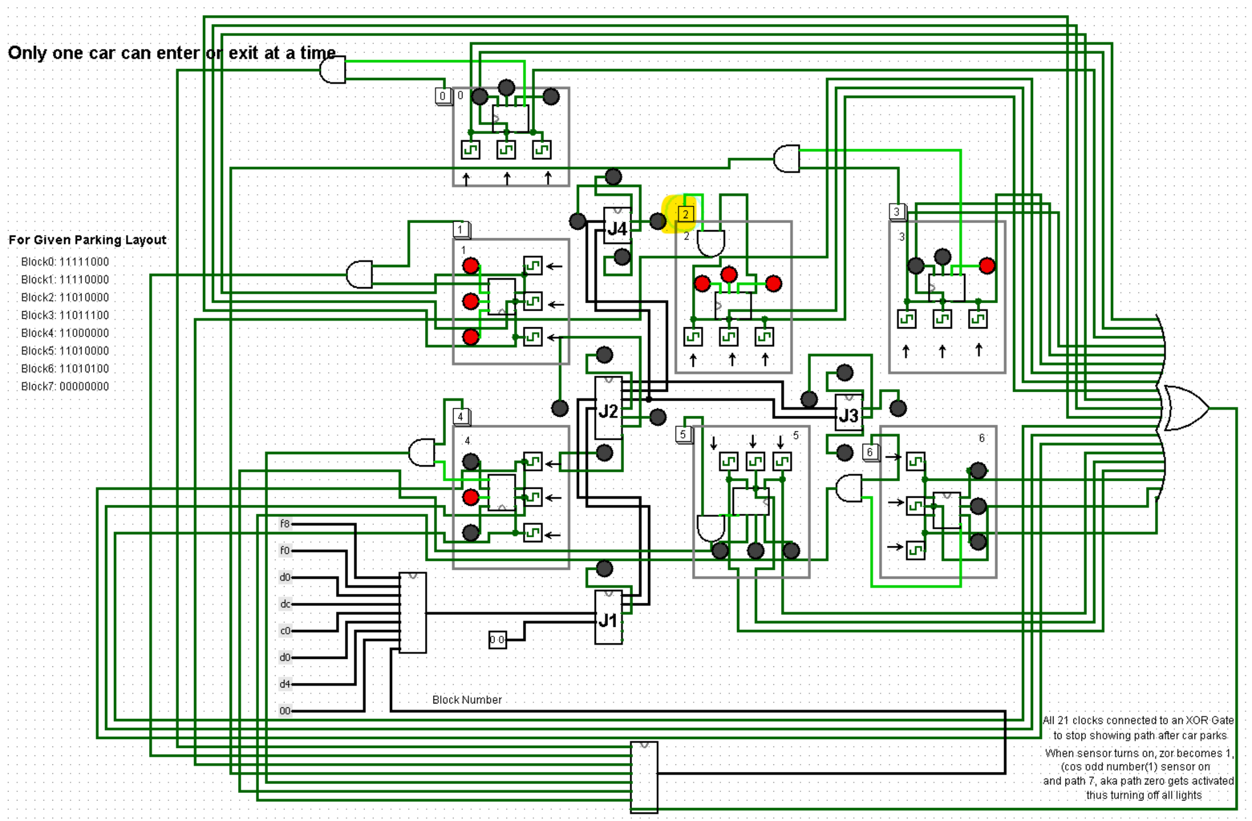
All 21 clocks connected to an XOR Gate
to stop showing path after car parks

When sensor turns on, zor becomes 1,
(cos odd number(1) sensor on
and path 7, aka path zero gets activated
thus turning off all lights

As we can see,
Now, on the screen we can see that the third slot of block 2 which was previously empty is now full as displayed by the LED after the current user parked in that slot

Now, let us see what happens in the case of the next user

Let us say, the user presses the block2 button and releases it


Only one car can enter or exit at a time

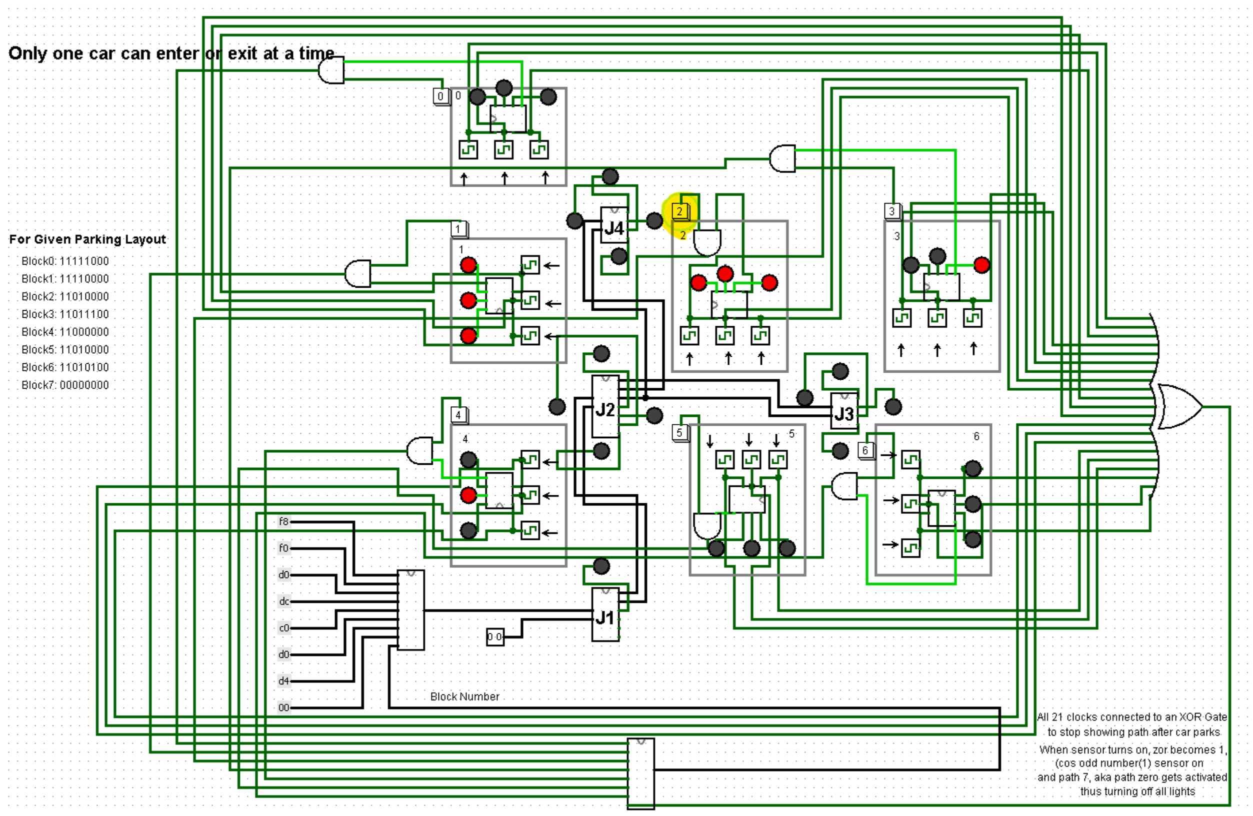
For Given Parking Layout

Block0: 11111000
Block1: 11110000
Block2: 11010000
Block3: 11011100
Block4: 11000000
Block5: 11010000
Block6: 11010100
Block7: 00000000

All 21 clocks connected to an XOR Gate to stop showing path after car parks

When sensor turns on, zor becomes 1, (cos odd number(1) sensor on and path 7, aka path zero gets activated thus turning off all lights

And releases it


Only one car can enter or exit at a time

For Given Parking Layout

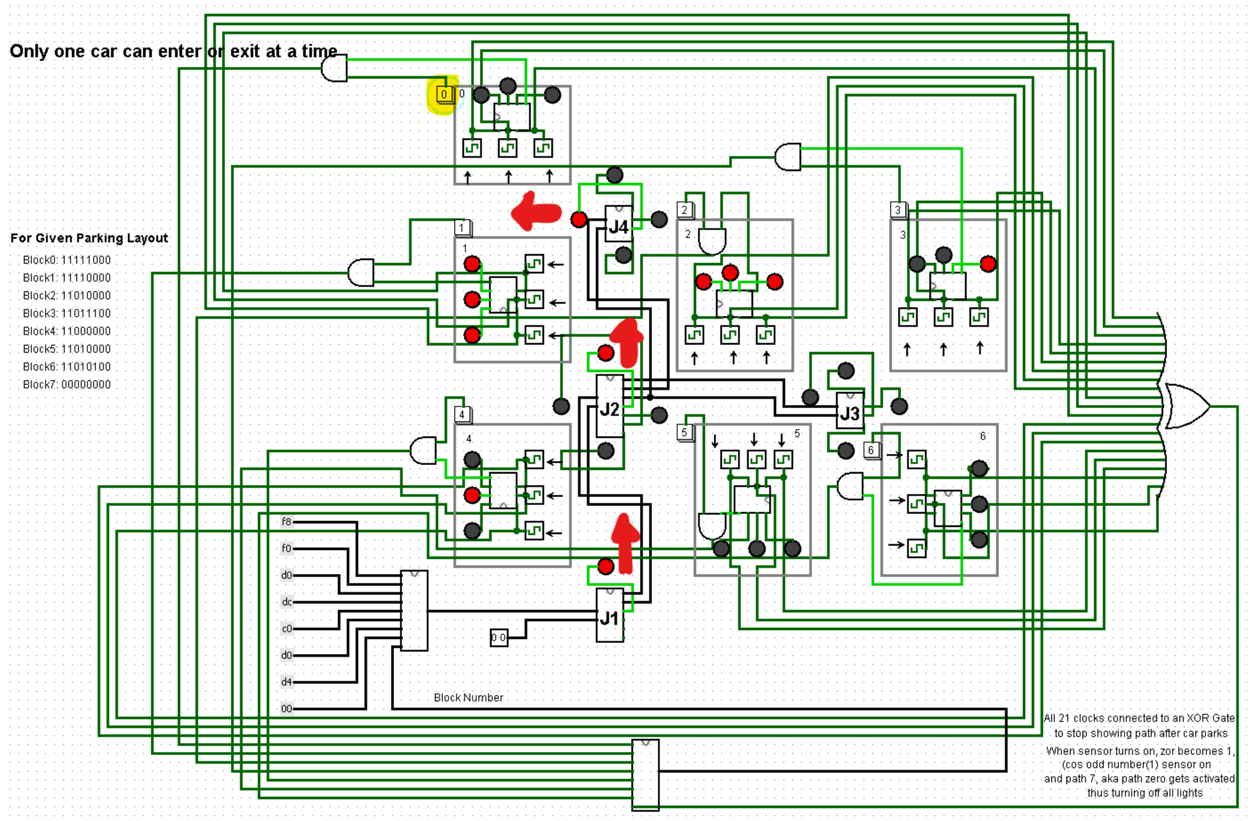
Block0: 11111000
Block1: 11110000
Block2: 11010000
Block3: 11011100
Block4: 11000000
Block5: 11010000
Block6: 11010100
Block7: 00000000

All 21 clocks connected to an XOR Gate to stop showing path after car parks

When sensor turns on, zor becomes 1, (cos odd number(1) sensor on and path 7, aka path zero gets activated thus turning off all lights

even though there are no vacant parking slots
No path to that block will be shown bcos, that block has no vacant slots

So, the user must select another block
Let us assume the user chooses block 0 now
He presses and releases the button for block 0


Only one car can enter or exit at a time

For Given Parking Layout

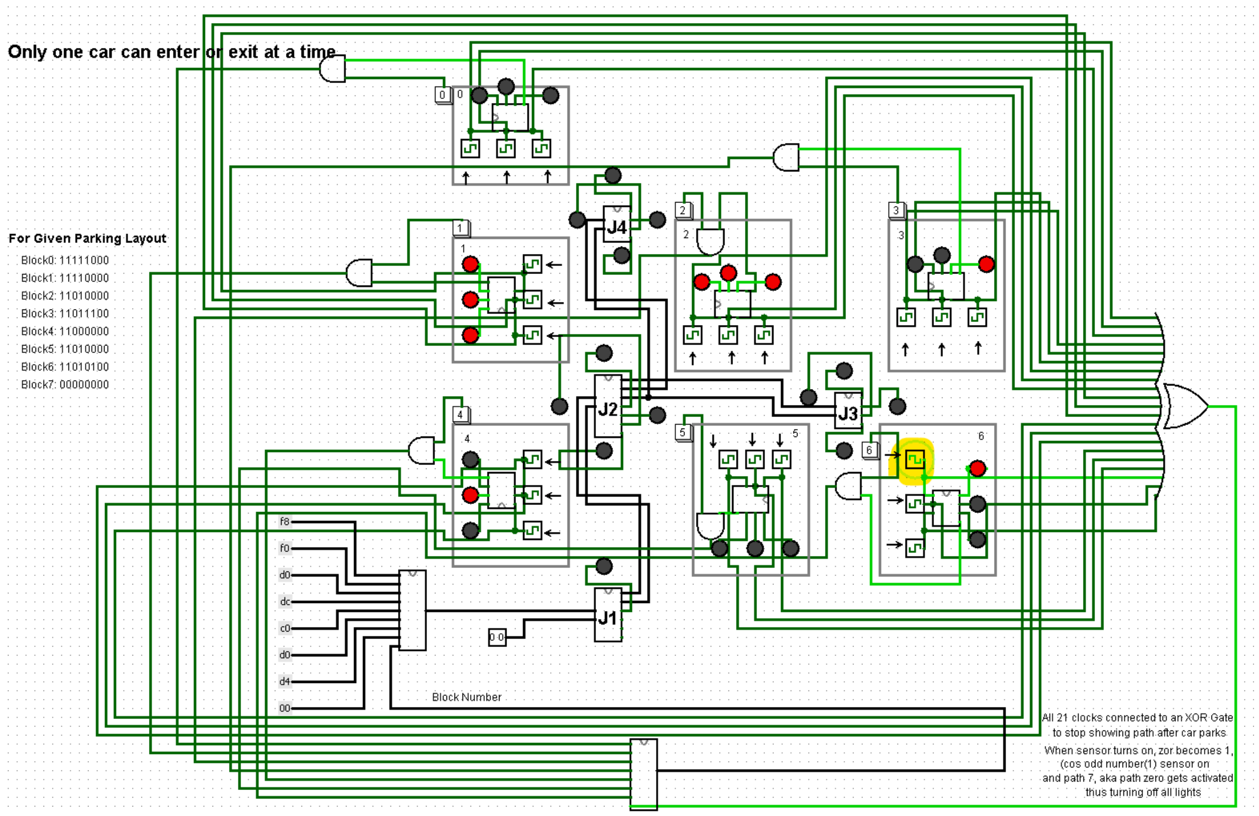
Block0: 11111000
Block1: 11110000
Block2: 11010000
Block3: 11011100
Block4: 11000000
Block5: 11010000
Block6: 11010100
Block7: 00000000

All 21 clocks connected to an XOR Gate
to stop showing path after car parks
When sensor turns on, zor becomes 1,
(cos odd number(1) sensor on
and path 7, aka path zero gets activated
thus turning off all lights

Now, the path for junction 0 is displayed as we can see from the code
Go straight from junction1
Go straight at junction2
Take a left at junction3

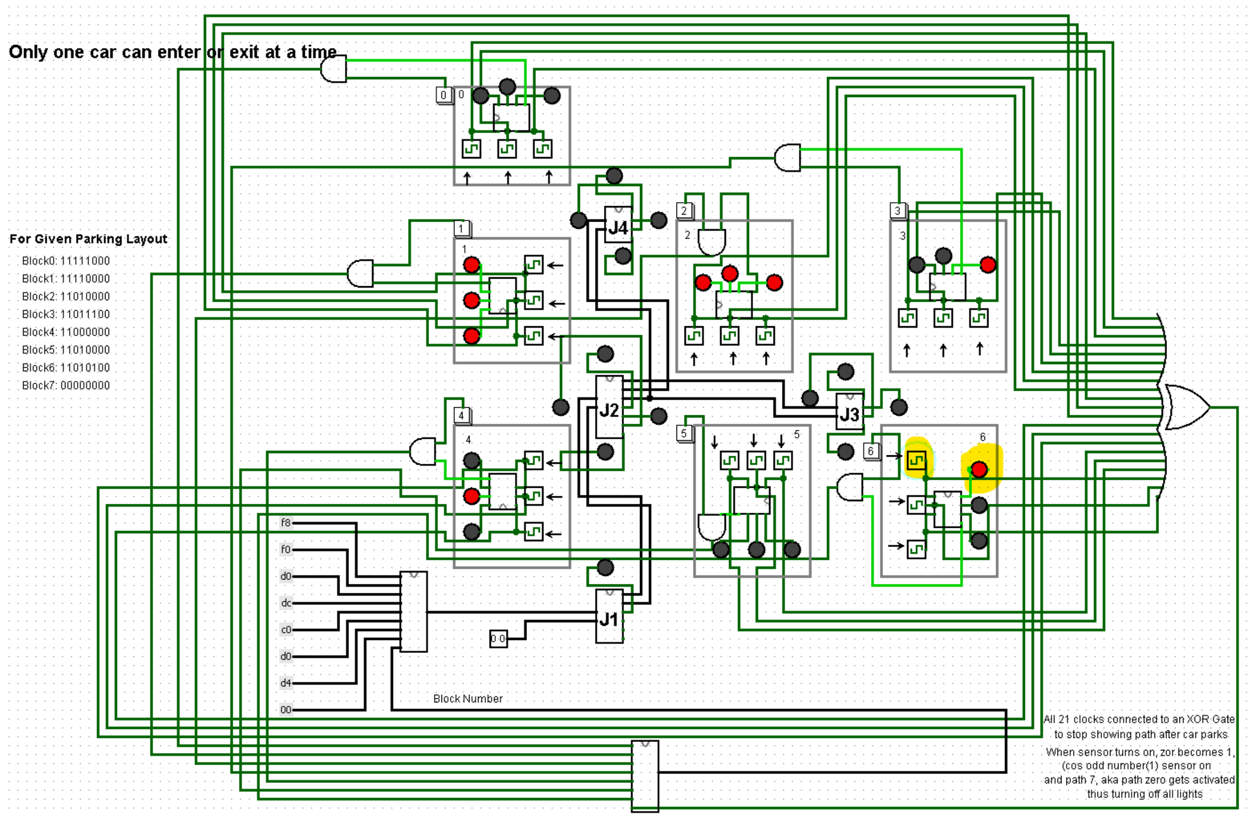
But, let us say, the user doesnt follow the path and goes to block 6
And parks in the first slot

Now, when the user enters the 1st slot at block6
The first sensor(clock), which is present at the entrance of the slot turns on
Which means, the person is going to enter in that slot


Only one car can enter or exit at a time
For Given Parking Layout
Block0: 11111000
Block1: 11110000
Block2: 11010000
Block3: 11011100
Block4: 11000000
Block5: 11010000
Block6: 11010100
Block7: 00000000
J1
J2
J3
J4
f8
f0
d0
dc
c0
d0
d4
00
Block Number
All 21 clocks connected to an XOR Gate
to stop showing path after car parks
When sensor turns on, zor becomes 1,
(cos odd number(1) sensor on
and path 7, aka path zero gets activated
thus turning off all lights

Now that this clock is turned on, the path towards block0 disappears
Bcos the car will be parked in this slot now, and thus doesnt need the directions anymore

Once the car parks in that slot
He will cross the entrance of that slot
Thus, the sensor(clock) for that slot becomes 0 again


Only one car can enter or exit at a time

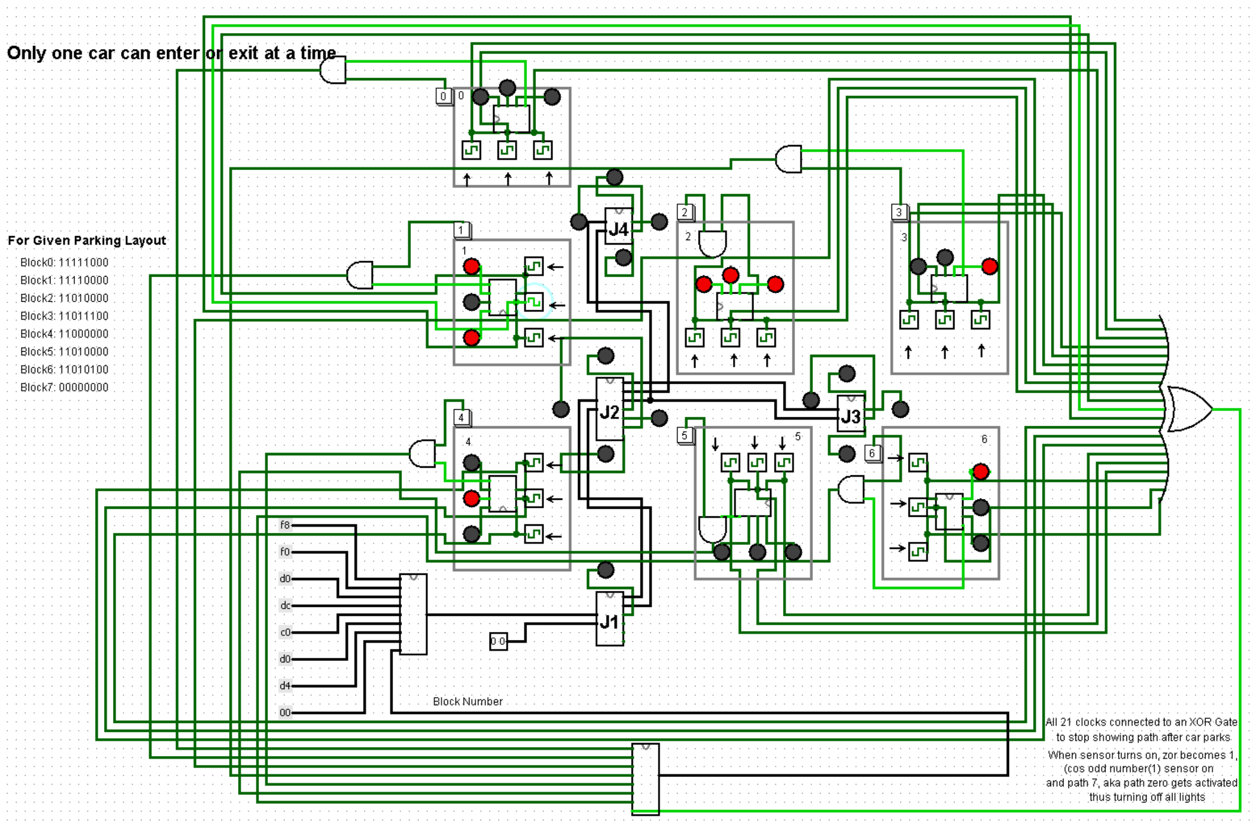
For Given Parking Layout

Block0: 11111000
Block1: 11110000
Block2: 11010000
Block3: 11011100
Block4: 11000000
Block5: 11010000
Block6: 11010100
Block7: 00000000

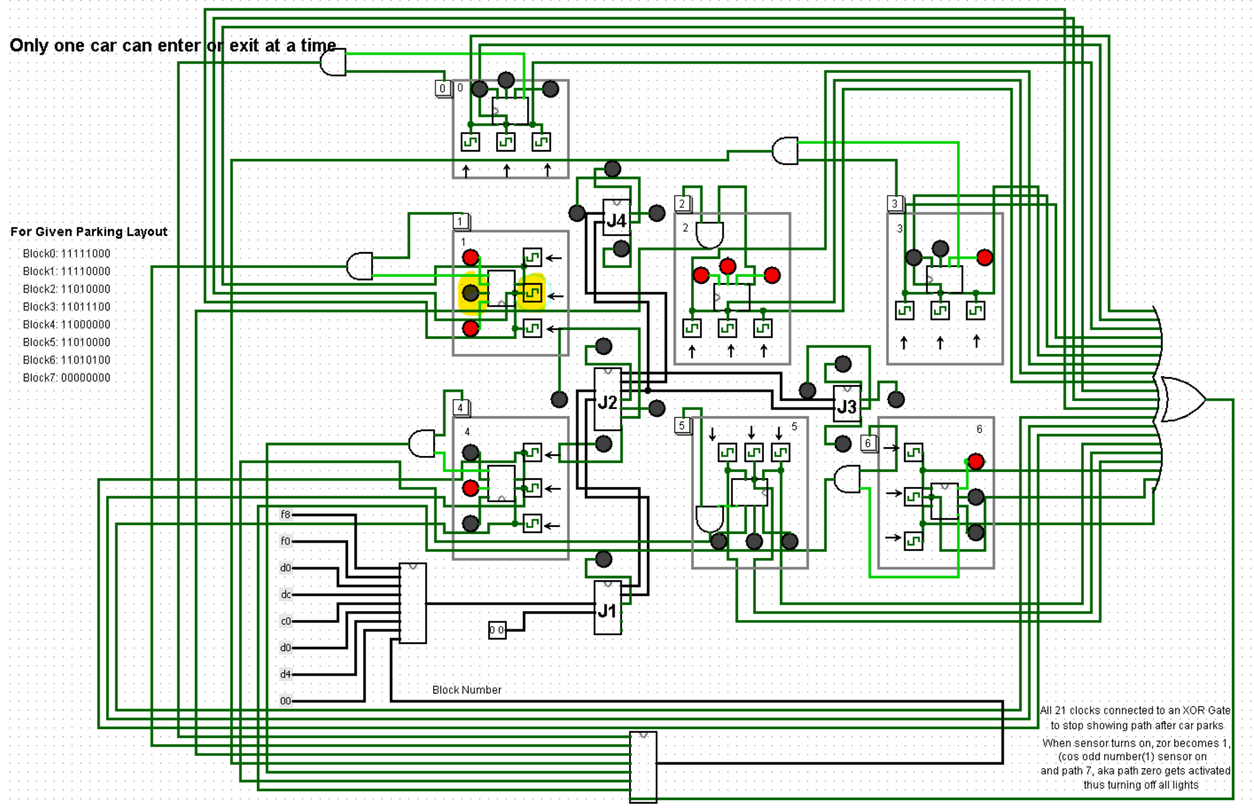
All 21 clocks connected to an XOR Gate
to stop showing path after car parks
When sensor turns on, zor becomes 1,
(cos odd number(1) sensor on
and path 7, aka path zero gets activated
thus turning off all lights

As we can see,
Now, on the screen we can see that the first slot of block 6 which was previously empty is now full as displayed by the LED after the current user parked in that slot

Now let us say, the car in the 2nd slot from block 1 is leaving
Bcos, the car is leaving, it passes through the entrance of that slot
So that sensor(clock) becomes 1


Only one car can enter or exit at a time
For Given Parking Layout
Block0: 11111000
Block1: 11110000
Block2: 11010000
Block3: 11011100
Block4: 11000000
Block5: 11010000
Block6: 11010100
Block7: 00000000
J1
J2
J3
J4
f8
f0
d0
dc
c0
d0
d4
00
Block Number
All 21 clocks connected to an XOR Gate
to stop showing path after car parks
When sensor turns on, zor becomes 1,
(cos odd number(1) sensor on
and path 7, aka path zero gets activated
thus turning off all lights

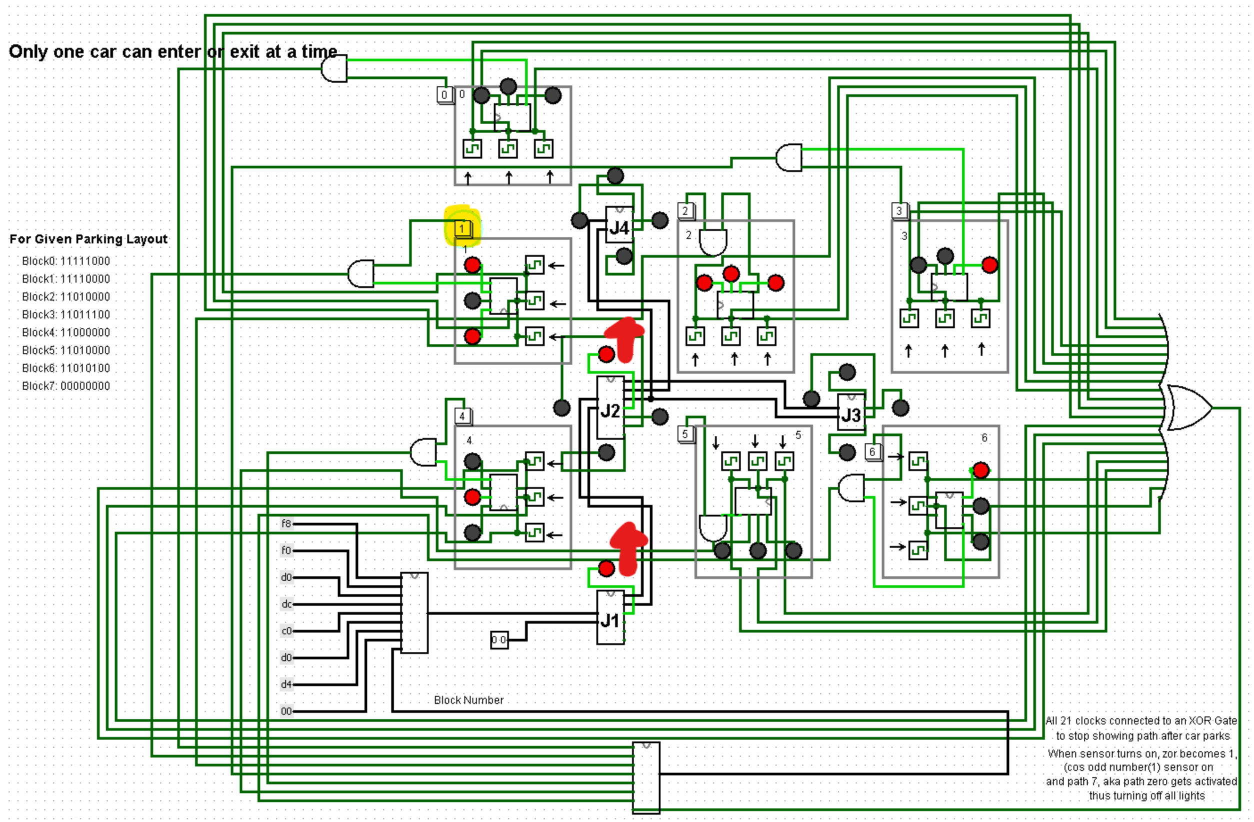
Now when he leaves that slot,
The clock sensor becomes 0 again


Only one car can enter or exit at a time
For Given Parking Layout
Block0: 11111000
Block1: 11110000
Block2: 11010000
Block3: 11011100
Block4: 11000000
Block5: 11010000
Block6: 11010100
Block7: 00000000
J1
J2
J3
J4
f8
f0
d0
dc
c0
d0
d4
00
Block Number
All 21 clocks connected to an XOR Gate
to stop showing path after car parks
When sensor turns on, zor becomes 1,
(cos odd number(1) sensor on
and path 7, aka path zero gets activated
thus turning off all lights

As we can see,

The 2nd slot of block1, which was previously filled is now empty
Thus, there is a vacant spot available in block1 now

So, if the next user presses the block1 button, the path to block1 will be displayed


Only one car can enter or exit at a time
For Given Parking Layout
Block0: 11111000
Block1: 11110000
Block2: 11010000
Block3: 11011100
Block4: 11000000
Block5: 11010000
Block6: 11010100
Block7: 00000000
J4
J2
J3
J1
Block Number
All 21 clocks connected to an XOR Gate
to stop showing path after car parks
When sensor turns on, zor becomes 1,
(cos odd number(1) sensor on
and path 7, aka path zero gets activated
thus turning off all lights

The aforementioned scenario seamlessly amalgamates all potential possibilities and eloquently articulates the ensuing outcomes therewithin.

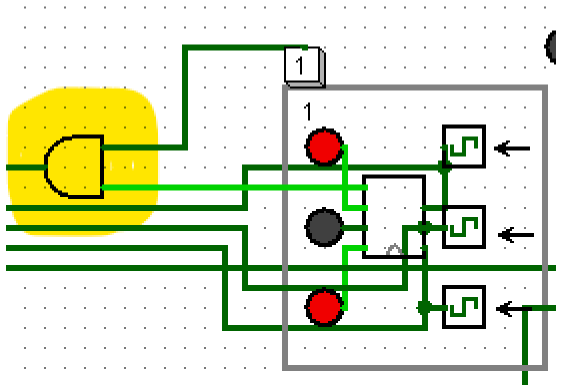
And this thus, explains what the project does
We will soon explain the working in the next section

**Working:**

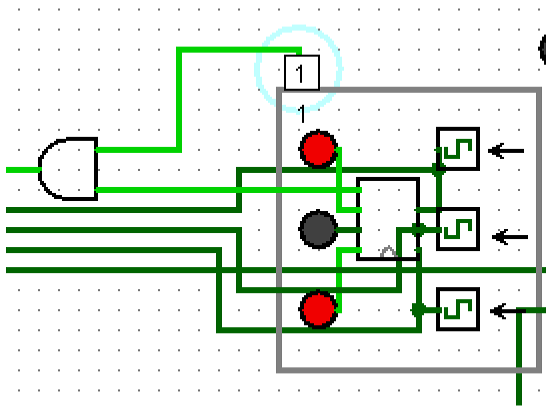
Let us go in a sequential order, starting from pressing a button until parking in a slot
Where the working will be explained step by step,
and each component will be explained on its first occurrence based on the above order

The user starts by pressing a button for a particular block
The way a button works it
When it is pressed, it is considered that the input is 1

The input for the next component will be the AND of the button input (which is 1) as long as it is pressed and the availability of free slot in that block
If a free slot is available, it returns 1, so when the button is pressed, input will be 1&1 which is 1

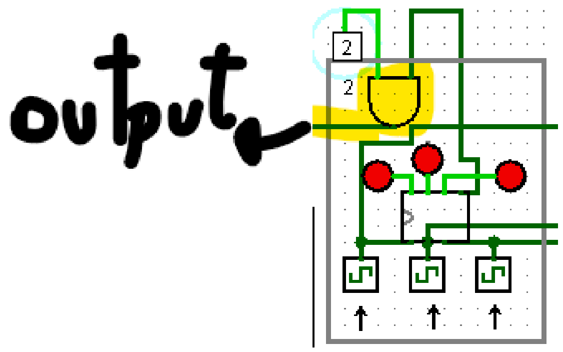


As we can see, only 3 slots are filled as per the leds
Which means a free slot is available
And when the button is pressed, the output is 1



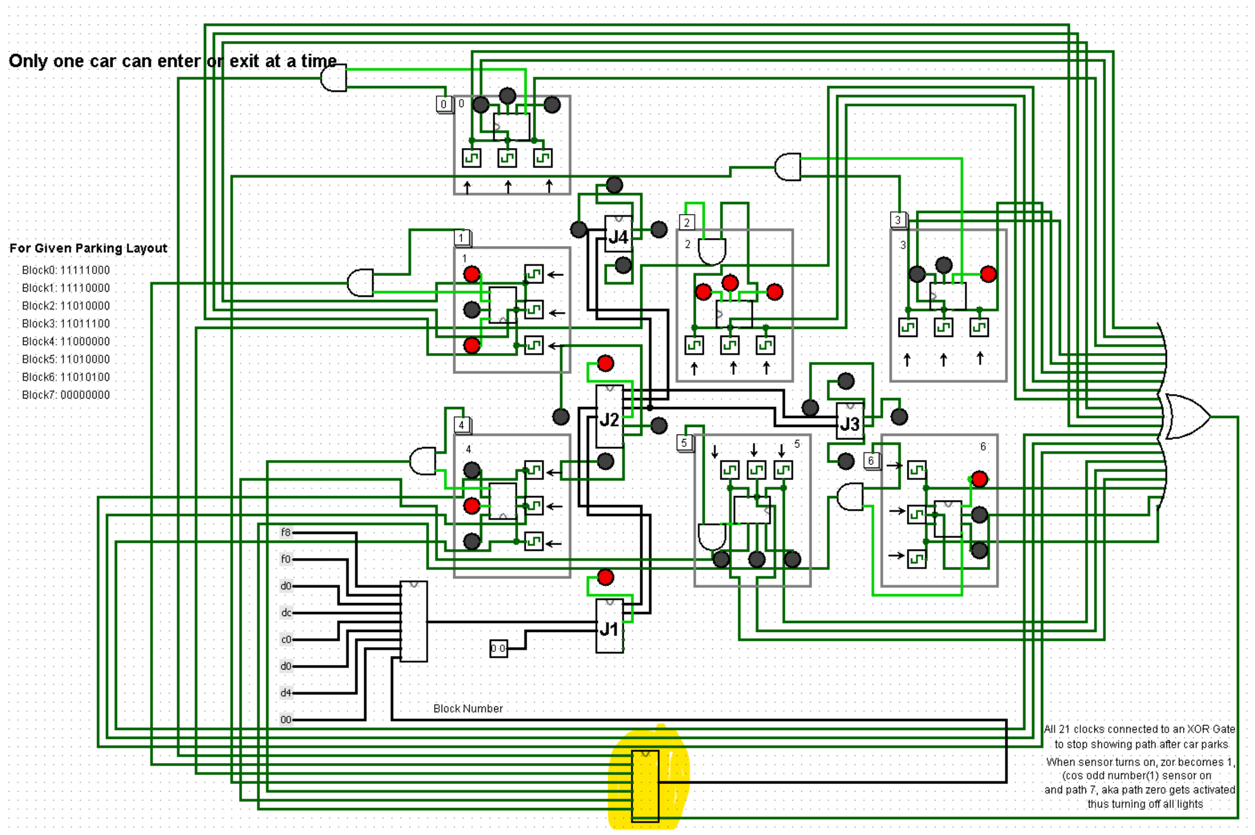
What will happen after this button press is that the path to this block will be displayed by the junctions which is triggered by the 1 output which we get when the button is pressed.

If no free slot is available, it returns 0, so when the button is pressed, input will be 1&0 which is 0, i.e. there is no change in the circuit and input from this side will remain 0.
By default, the input from this block will be 0.



As we can see, bcos all 3 leds are blinking, no free slot is available, so even though the button is pressed, the output will be 0.
Bcos, there is not change in the output, pressing the button in this situation wont change anything
Bcos this block is filled, a path to this block wont be shown

This input is connected to the BlockChooser
The input from Block i is connected to the ith port of BlockChooser
The Block Chooser will eventually return the BlockNumber of the most recent button which is pressed.

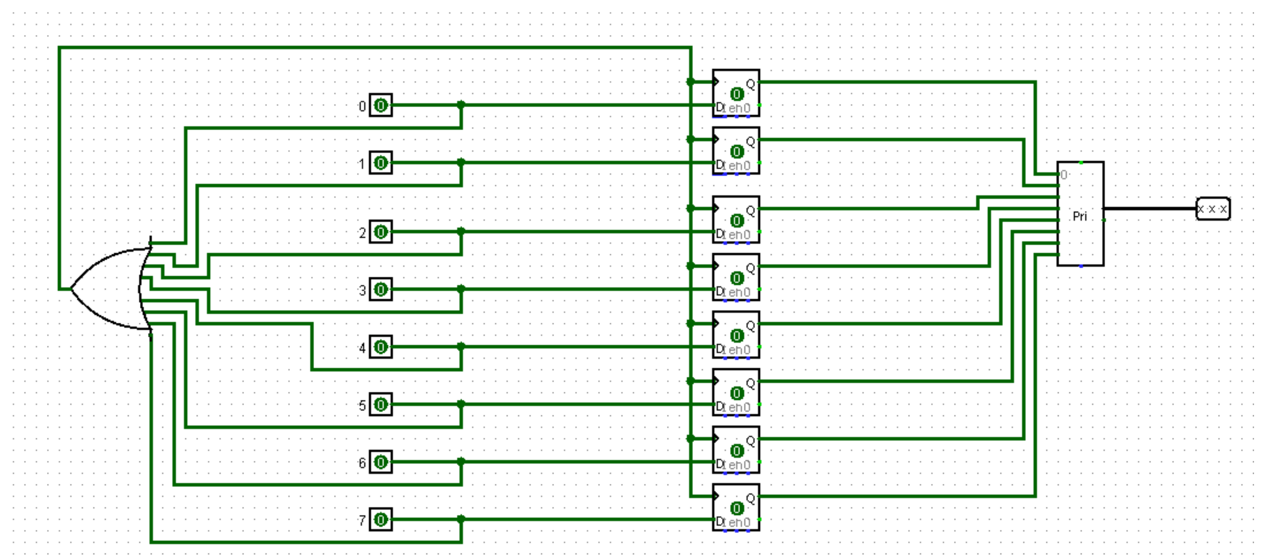


Only one car can enter or exit at a time

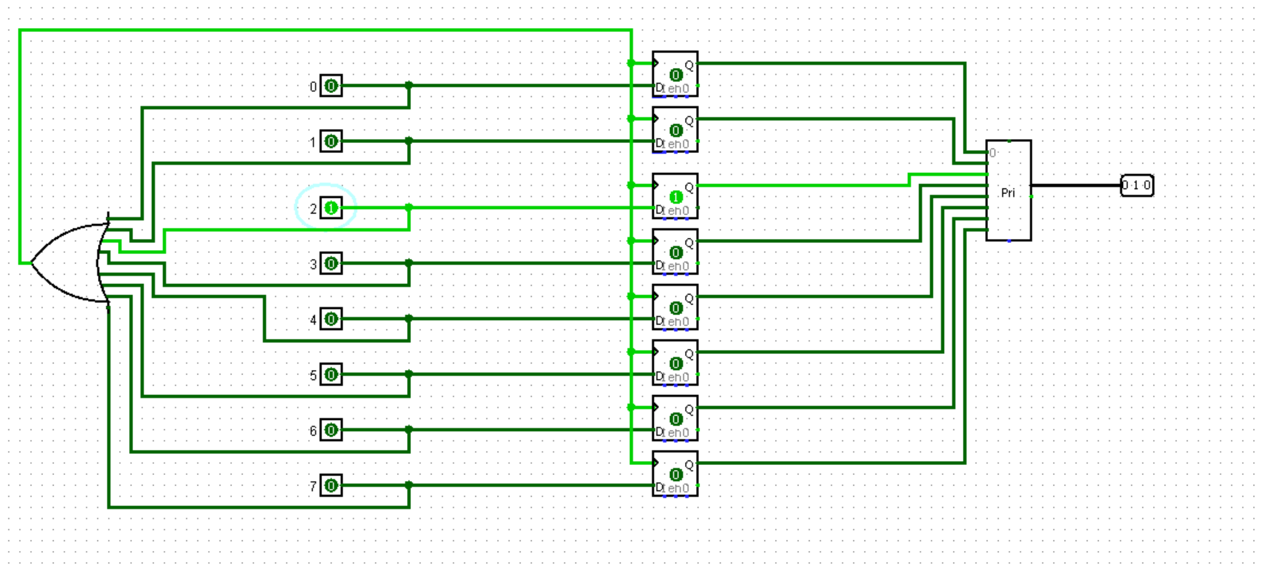
For Given Parking Layout

Block0: 11111000
Block1: 11110000
Block2: 11010000
Block3: 11011100
Block4: 11000000
Block5: 11010000
Block6: 11010100
Block7: 00000000

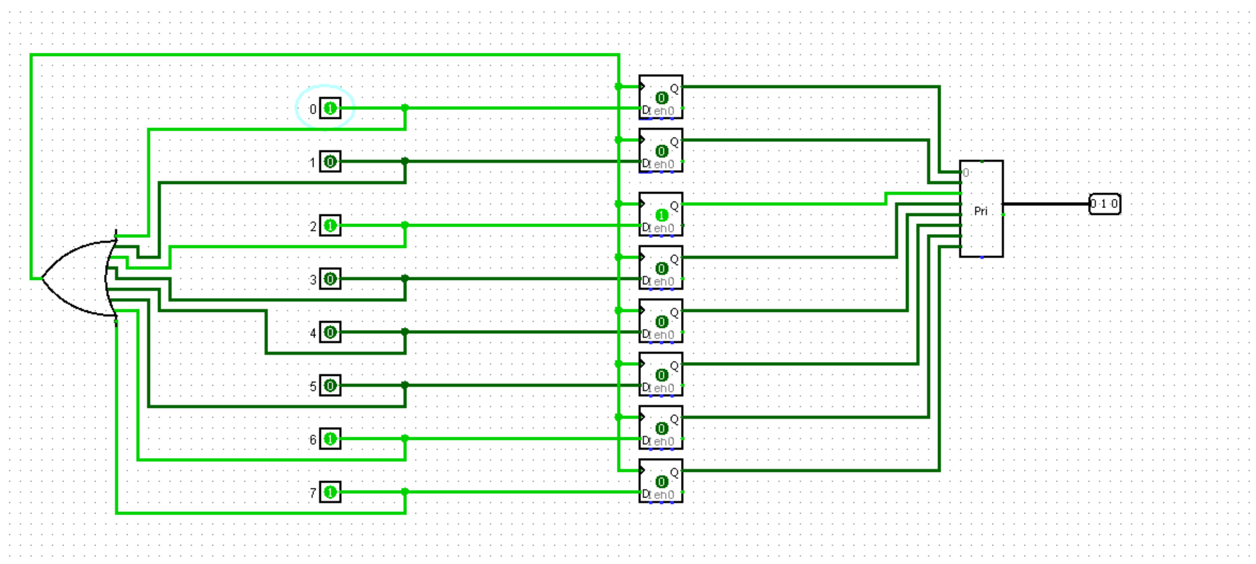
All 21 clocks connected to an XOR Gate
to stop showing path after car parks
When sensor turns on, zor becomes 1,
(cos odd number(1) sensor on
and path 7, aka path zero gets activated
thus turning off all lights



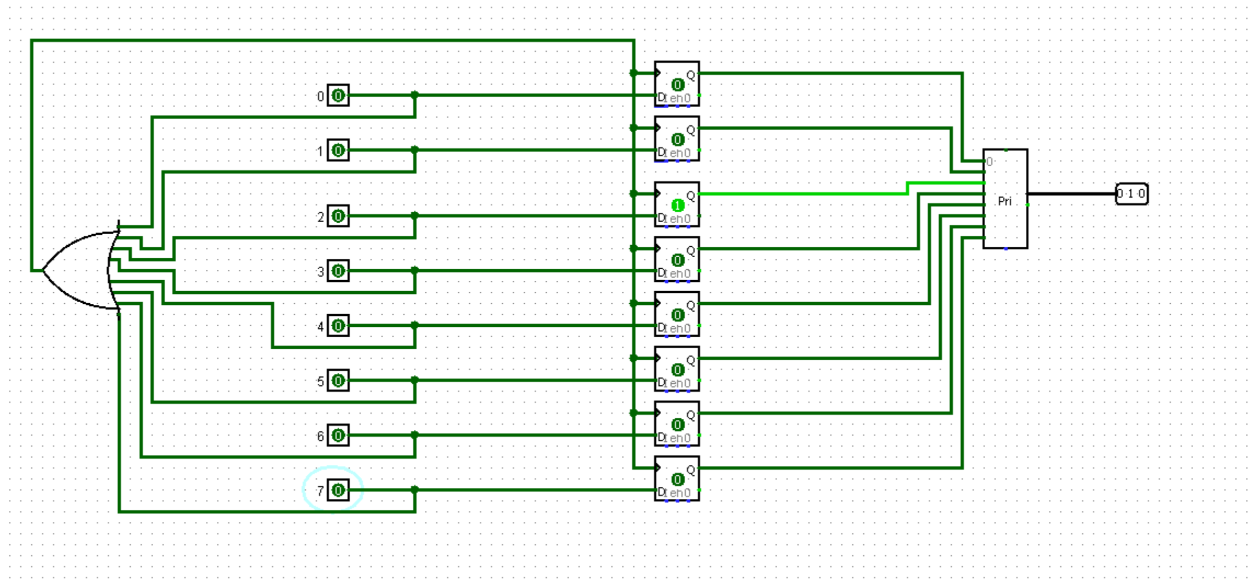
How this works is
If button 2 was pressed, the input 2 becomes 1
And simultaneously the clock tick for each flipflop changes from 0 to 1
Bcos the OR gate whose output was initially 0
Returns 1 now
And the output for the flipflop connected to input 2, becomes 1



Now, bcos the clock is not ticking and fixed at 1, even if the user presses multiple buttons, only the 2nd flipflop output will be considered,
bcos until the next clock tick, the previous output will be displayed,
bcos the OR gate still remains at 1 for the added 1 inputs,
the clock tick doesnt change and is still at 1



Now, after the path is generated, which will be explained later
The user will leave,
So he is not pressing any button now
after releasing the buttons
The OR gate returns 0 bcos all inputs are 0
So the next clock tick is activated
So the previous input which is block 2 will be shown until another button is pressed

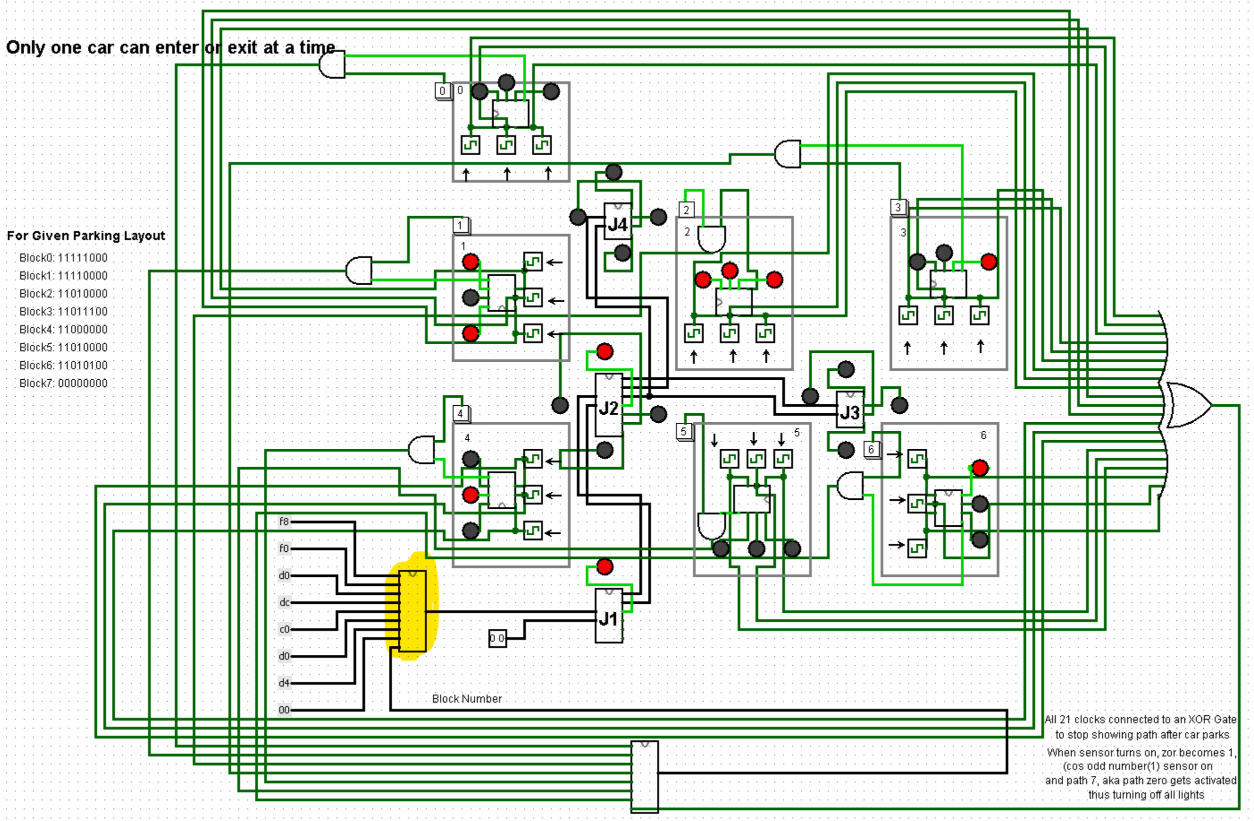


8 outputs connected from 8 D flipflops corresponding to their 8 respective inputs as per the above diagram will be taken as the corresponding inputs for the Priority Encoder
The Component marked Pri, is a Prioirty Encoder
Block 0 is input 0 in Pri
Block 1 is input 1 in Pri, ans so on……
Note: Block 7  is a special button whose functionality will be explained later on

The Priority Encoder gives Block Number in binary form as output

As we can see, after pressing the button for Block2, The block number is 010 which is the binary representation for Block2
The Block2 will be the output until another clock tick/change, i.e. another button is pressed, so the code will keep displaying the path for block2 until another button is pressed (which will be button 7, reason will be explained later)

Now, this output block number will be taken as input for Path Generator



**For Given Parking Layout**

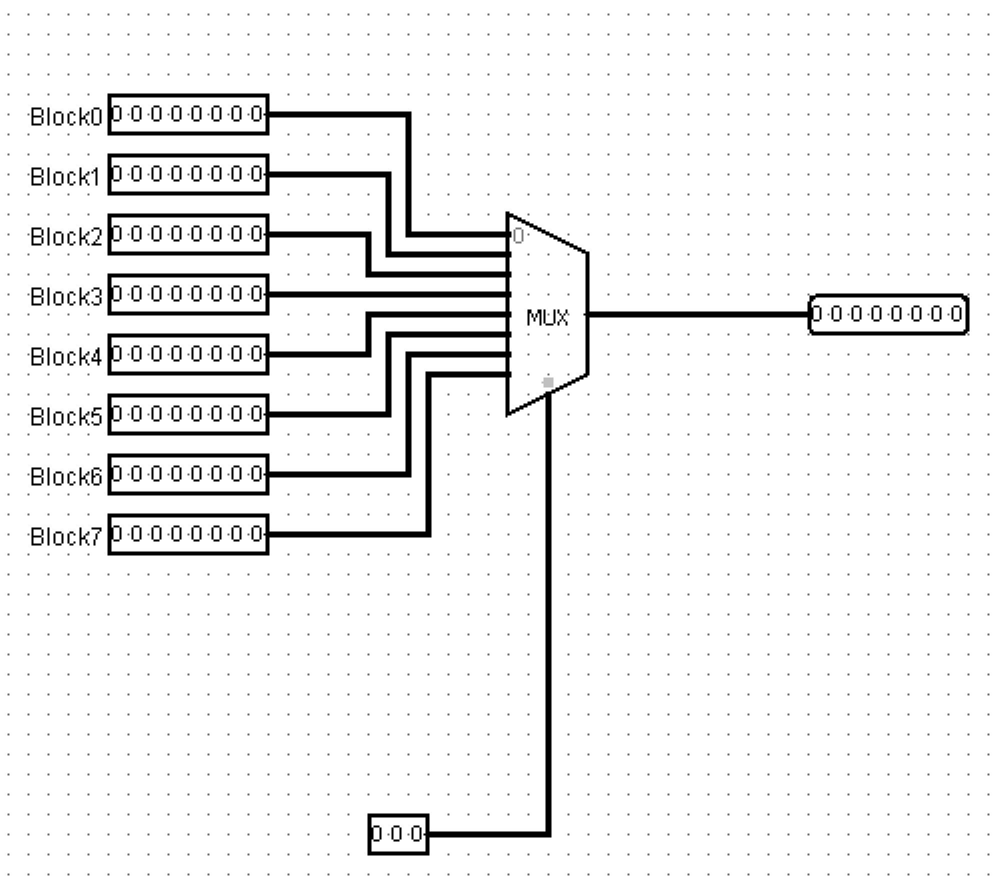
Block0: 11111000
Block1: 11110000
Block2: 11010000
Block3: 11011100
Block4: 11000000
Block5: 11010000
Block6: 11010100
Block7: 00000000

All 21 clocks connected to an XOR Gate to stop showing path after car parks

When sensor turns on, zor becomes 1, (cos odd number(1) sensor on and path 7, aka path zero gets activated thus turning off all lights

The highlighted component is PathGenerator

The PathGenerator will give PathData as output which will be used by the junctions to display the path, i.e. which direction to go towards at each potential junction the user will encounter to go to that path.



This operation is preformed using a multiplexer
The Input at port k will be the PathData for Block Number k
And the select input will be block number
So, depending on the block number, the path is given
If the block number is 3, the PathData to reach Block 3 is an output
Same for all blocks.
This operation is preformed using a multiplexer

Now, let me explain what PathData means and what each junction is supposed to do and does for particular PathData. After doing so, i'll explain the junction layout for the above example.

The Path Generator will generate PathData to reach the particular Block Number
PathData is essentially a 8 bit data, which gives information that represents a path.

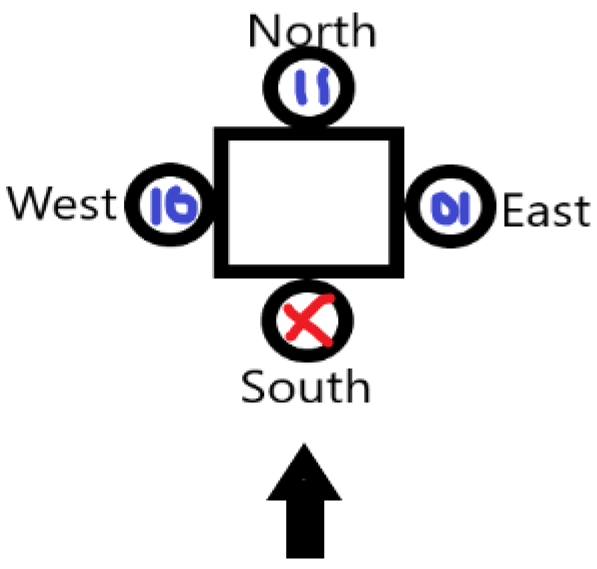
The first 2 bits represent the direction to take at the first encountered junction
The next 2 bits represent the direction to take at the second encountered junction
The next 2 bits represent the direction to take at the third encountered junction
And so on….

Note: At each junction,
We can extract the first 2 bits from PathData and call it direction
From the first 2 bits of PathData
11 represents front direction
10 represents left direction
01 represents right direction
00 is used for termination
(bcos we dont want the user to take a u-turn to reach their block, we dont use 00 to represent back direction. And after we reached all necessary junctions, the remaining PathData will be 00______, reason will be explained later, so we dont want anymore junctions to display anything)
And left shift by 2, so that at next junction, we can use next 2 bits for direction

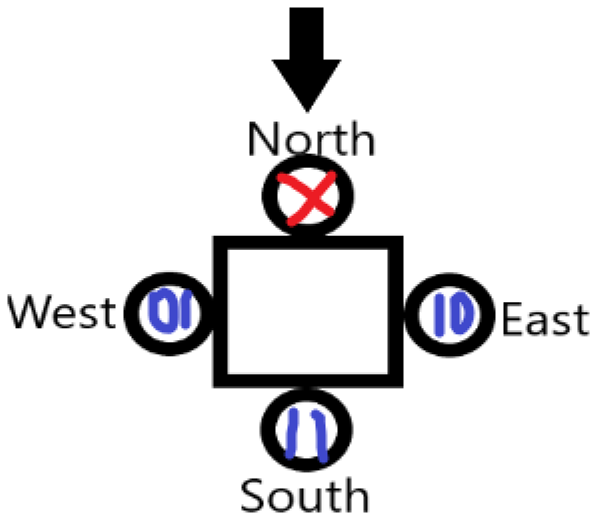
Let input direction signify the direction from which, each junction receives the input
With regards to input direction at the junction, let
00 represent south
01 represent west
10 represent east
11 represent north

Here is a truth table signifying which light (NORTH, SOUTH, EAST or WEST) of the junction must be displayed depending on input direction and

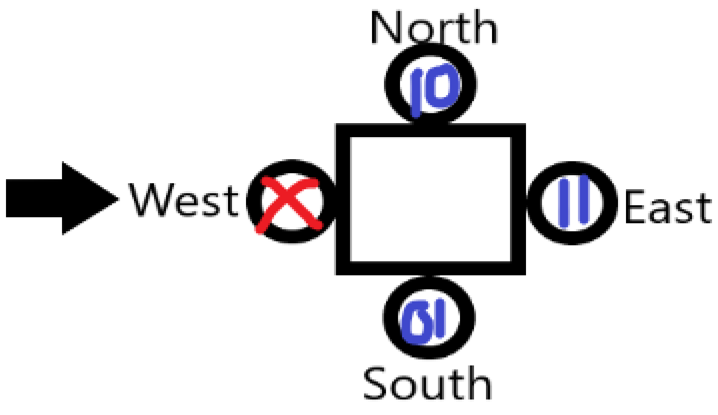
Depending on direction of encountering the junction and the direction displayed based on the 2 data input, here is a truth table



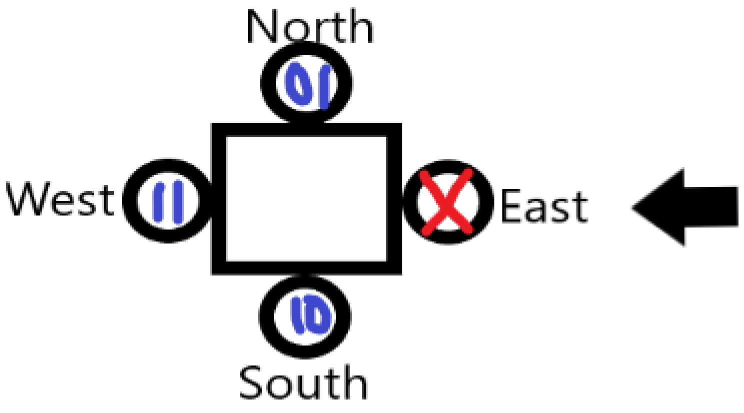
In this direction, input direction is 00, i.e. south
If input is 11, we must go front from south direction which signifies north, north light is on
If Input is 10, we must go left from south direction which signifies west, west light is on
If Input is 01, we must go right from south direction which signified east, east light is on
If input is 00, no light will be on



In this direction, input direction is 11, i.e. north
If input is 11, we must go front from north direction which signifies south, south light is on
If Input is 10, we must go left from north direction which signifies east, east light is on
If Input is 01, we must go right from north direction which signified west, west light is on
If input is 00, no light will be on



In this direction, input direction is 10, i.e. west
If input is 11, we must go front from west direction which signifies east, east light is on
If Input is 10, we must go left from west direction which signifies north, north light is on
If Input is 01, we must go right from west direction which signified south, south light is on
If input is 00, no light will be on



In this direction, input direction is 10, i.e. east
If input is 11, we must go front from east direction which signifies west, west light is on
If Input is 10, we must go left from east direction which signifies south, south light is on
If Input is 01, we must go right from east direction which signified north, north light is on
If input is 00, no light will be on

Let us assume display is 0 when direction from pathdata is 00 to make simplification easier,
We can and each light with display so that nothing is displayed when display is 0
So dis= dir[0] & dir[1];

Here is a truth table signifying the same

| | | | | | |
|---|---|---|---|---|---|
| Input Direction | Direction from PathData | North Light | South Light | East Light | West Light |
| _00 | _00 | 0 | 1 | 0 | 0 |
| _00 | _01 | 0 | 0 | 1 | 0 |
| _00 | 10 | 0 | 0 | 0 | 1 |
| _00 | 11 | 1 | 0 | 0 | 0 |
| 11 | _00 | 1 | 0 | 0 | 0 |
| 11 | _01 | 0 | 0 | 0 | 1 |
| 11 | 10 | 0 | 0 | 1 | 0 |
| 11 | 11 | 0 | 1 | 0 | 0 |
| _01 | _00 | 0 | 0 | 1 | 0 |
| _01 | _01 | 1 | 0 | 0 | 0 |
| _01 | 10 | 0 | 1 | 0 | 0 |
| _01 | 11 | 0 | 0 | 0 | 1 |
| 10 | _00 | 0 | 0 | 0 | 1 |
| 10 | _01 | 0 | 1 | 0 | 0 |
| 10 | 10 | 1 | 0 | 0 | 0 |
| 10 | 11 | 0 | 0 | 1 | 0 |

Using the same conventions for north as 11, south as 00, east as 01 ans west as 10
I want the Output Direction be the input direction for the next junction
So, that if there is a junction with multiple outputs, we know what is the input direction for the next junction directly, so we dont need to initialize Input Direction for further junction except at Junction1 for entrance
So the light which must be displayed must be reverse to Output Direction

Therefore
North Light for Output Direction 00
South Light for Output Direction 11
East Light for Output direction 10
West Light for Output Direction 01

And i can decide whether the particular light must be outputed using this

```
    assign NL = dis && (!OutputDir[1] && !OutputDir[0]);
    assign SL = dis && (OutputDir[1] && OutputDir[0]);
    assign EL = dis && (OutputDir[1] && !OutputDir[0]);
    assign WL = dis && (!OutputDir[1] && OutputDir[0]);
```

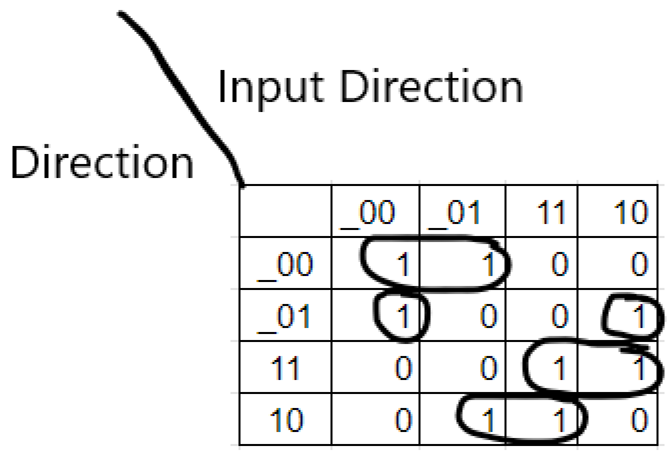
If we dont encounter 00, dis will be 1
So the above and condition wont be afftected

If we encounter a 00, no light is displayed
Cos all lights will give output 0 with dis as 0 in and gate
And, later on we’ll see that if no light is displayed, the path is complete,
thus we need not display anything,
Thus the next junction need not display anythin
so we can set the first two inputs of path data as 00 for the next junction
So that the next junction doesnt display anything

So truth table for Output Direction is opposite to the the Light which should be shown
Truth Table for Output Direction

| | | Output Direction | |
|---|---|---|---|
| Input Direction | Direction from PathData | OutputDir[1] | OutputDir[0] |
| _00 | _00 | 1 | 1 |
| _00 | _01 | 1 | 0 |
| _00 | 10 | 0 | 1 |
| _00 | 11 | 0 | 0 |
| 11 | _00 | 0 | 0 |
| 11 | _01 | 0 | 1 |
| 11 | 10 | 1 | 0 |
| 11 | 11 | 1 | 1 |
| _01 | _00 | 1 | 0 |
| _01 | _01 | 0 | 0 |
| _01 | 10 | 1 | 1 |
| _01 | 11 | 0 | 1 |
| 10 | _00 | 0 | 1 |
| 10 | _01 | 1 | 1 |
| 10 | 10 | 0 | 0 |
| 10 | 11 | 1 | 0 |

K-Map for OutputDir[1] is



Input Direction

Direction

| | _00 | _01 | 11 | 10 |
|---|---|---|---|---|
| _00 | 1 | 1 | 0 | 0 |
| _01 | 1 | 0 | 0 | 1 |
| 11 | 0 | 0 | 1 | 1 |
| 10 | 0 | 1 | 1 | 0 |

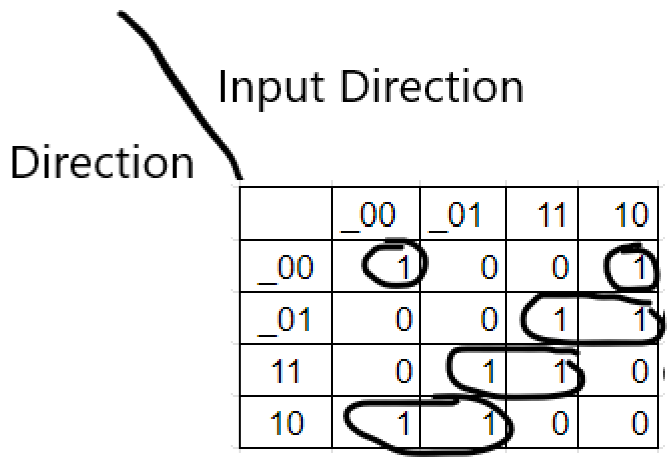
Note: if dir is 10, dir[1] is 1 & dir[0] is 0
The function is

```
    assign OutputDir[1] = (!dir[1] && !dir[0] && !InputDir[1])
                || (!dir[1] && dir[0] && !InputDir[0])
                || (dir[1] && dir[0] && InputDir[1])
                || (dir[1] && !dir[0] && InputDir[0]);
```

Based on how the function is,
We can use a multiplexer
So the effective code is

```
mux m1(!InputDir[1],!InputDir[0],InputDir[0],InputDir[1],dir,OutputDir[1]);
```

Similarly, for OutputDirection[0]
As per truth table, the K-map is



Input Direction

Direction

| | _00 | _01 | 11 | 10 |
|---|---|---|---|---|
| _00 | 1 | 0 | 0 | 1 |
| _01 | 0 | 0 | 1 | 1 |
| 11 | 0 | 1 | 1 | 0 |
| 10 | 1 | 1 | 0 | 0 |

function is

```
    assign OutputDir[0] = (!dir[1] && !dir[0] && !InputDir[0])
```

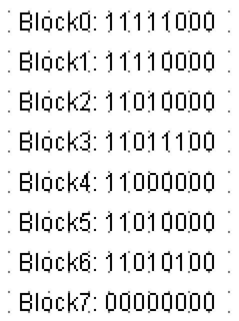
```
                || (!dir[1] && dir[0] && InputDir[1])
                || (dir[1] && dir[0] && InputDir[0])
                || (dir[1] && !dir[0] && !InputDir[1]);
```

Based on how the function is,
We can use a multiplexer
So the effective code is
mux m2(!InputDir[0],InputDir[1],!InputDir[1],InputDir[0],dir,OutputDir[0]);

After this, if the next junction is at the left
For the junction at left
The code returns the shifted path data, after left shift of 2
assign NewPath = PathData << 2;

Note that, each junction can receive input from only one direction.
We are doing so to optimize the project by the usage of a lesser number of components.
To do so, we need to ensure that the chosen PathDatas adhere with the above conditions.
So, what we do is, when we choose the path data, we ensure that, in every PathData, if a junction is to be encountered, the path to that junction will be unique.
Which means, the input direction to that junction will be unique,
Therefore, only one input direction for each junction.

This is the path data for each block for the parking layout

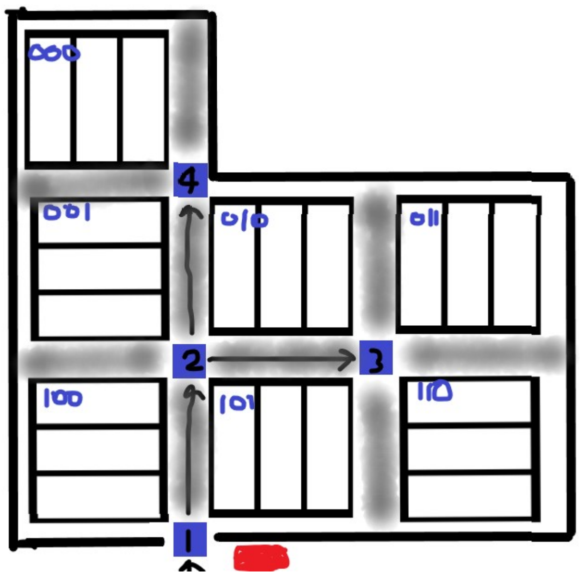


```
Block0: 11111000
Block1: 11110000
Block2: 11010000
Block3: 11011100
Block4: 11000000
Block5: 11010000
Block6: 11010100
Block7: 00000000
```

And this is the parking layout:

| Junc | Path from Junc |
|---|---|
| 2 | 11 |
| 3 | 1101 |
| 4 | 1111 |

| Block | Path |
|---|---|
| 0 | 11111000 |
| 1 | 11110000 |
| 2 | 11010000 |
| 3 | 11011100 |
| 4 | 11000000 |
| 5 | 11010000 |
| 6 | 11010100 |



Now for our layout
Junction 1 passes New Pathdata to Junction 2 from south direction
If north light is displayed, it means we are supposed to go to junction 2
So the first 2 characters will remain the same in NewPath, so for junction1
(We make first 2 characters 00 if we are not supposed to go in that direction, in this case, when North Light is 00)

```
Junction1 j1(InputDir,PathData,NL1,SL1,EL1,WL1,OutputDir1,NewPath1);
    assign InputDir2 = OutputDir1;
    assign PathData2[7] = NewPath1[7] && NL1;
    assign PathData2[6] = NewPath1[6] && NL1;
    assign PathData2[5:0] = NewPath1[5:0];
```

Junction 2 passes New Pathdata to Junction 3 from west direction is east light is on
And Junction4 from north direction if north light is on
 If East light is on, the next junction we must go to is Junction 3
So we can and the first two digits of NewPath of Junction3 with the East Light so that for junction3, pathdata will give output based on next two digits in Path data
Because we are doing and with east lights,
If the north light was switched on and east light was switched off, we were not supposed to go to junction3
So the NewPath for Junction3's first 2 digits will be 00
Which means, nothing will be displayed at Junction3

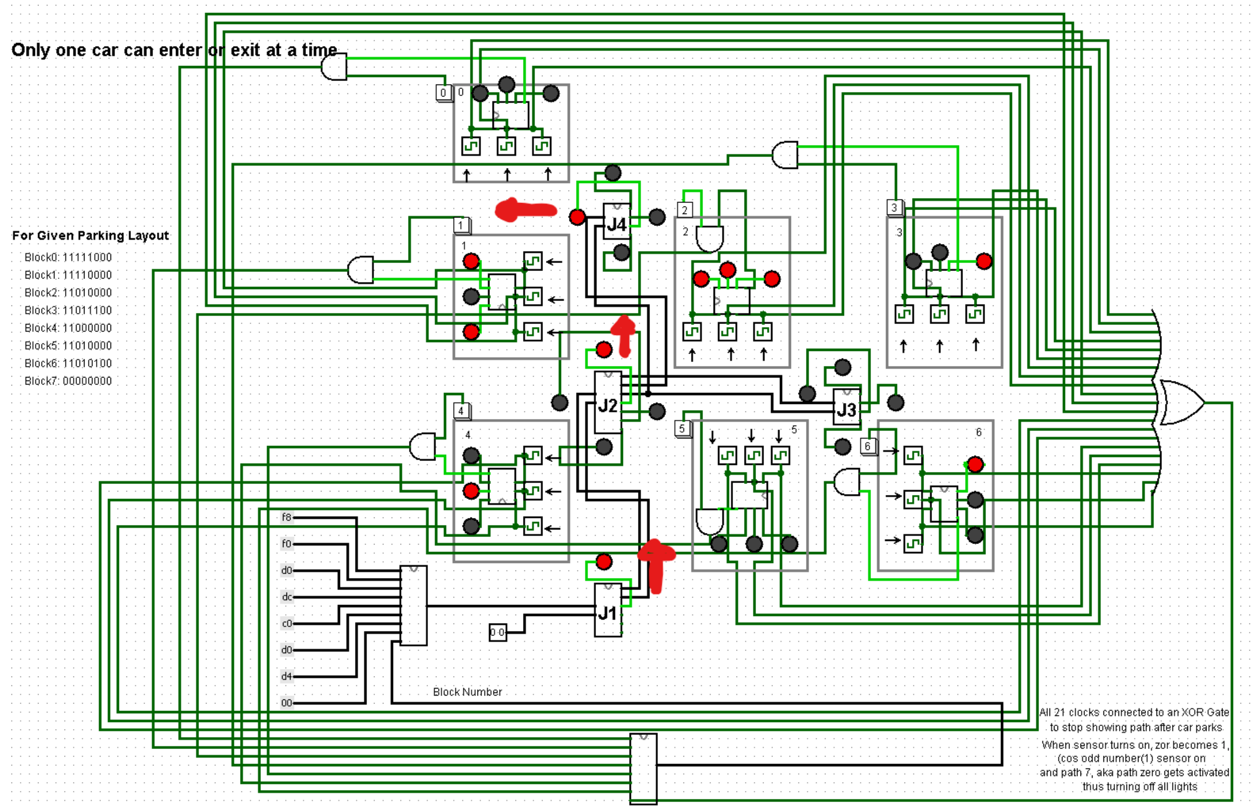
We similarly do the same with North Light for Junction4

```
    Junction2 j2(InputDir2,PathData2,NL2,SL2,EL2,WL2,OutputDir2,NewPath2);
    assign InputDir3 = OutputDir2;
    assign PathData3[7] = NewPath2[7] && EL2;
    assign PathData3[6] = NewPath2[6] && EL2;
    assign PathData3[5:0] = NewPath2[5:0];
    assign InputDir4 = OutputDir2;
    assign PathData4[7] = NewPath2[7] && NL2;
    assign PathData4[6] = NewPath2[6] && NL2;
    assign PathData4[5:0] = NewPath2[5:0];
```

Junction 3 and 4 doesnt pass New Pathdata to any junction
So the code for them is

```
    Junction3 j3(InputDir3,PathData3,NL3,SL3,EL3,WL3);
    Junction4 j4(InputDir4,PathData4,NL4,SL4,EL4,WL4);
```

Let us assume the PathData is 11111000



**Only one car can enter or exit at a time**

**For Given Parking Layout**

Block0: 11111000
Block1: 11110000
Block2: 11010000
Block3: 11011100
Block4: 11000000
Block5: 11010000
Block6: 11010100
Block7: 00000000

All 21 clocks connected to an XOR Gate
to stop showing path after car parks
When sensor turns on, zor becomes 1,
(cos odd number(1) sensor on
and path 7, aka path zero gets activated
thus turning off all lights

In this case, the PathData to Block0 is 11111000

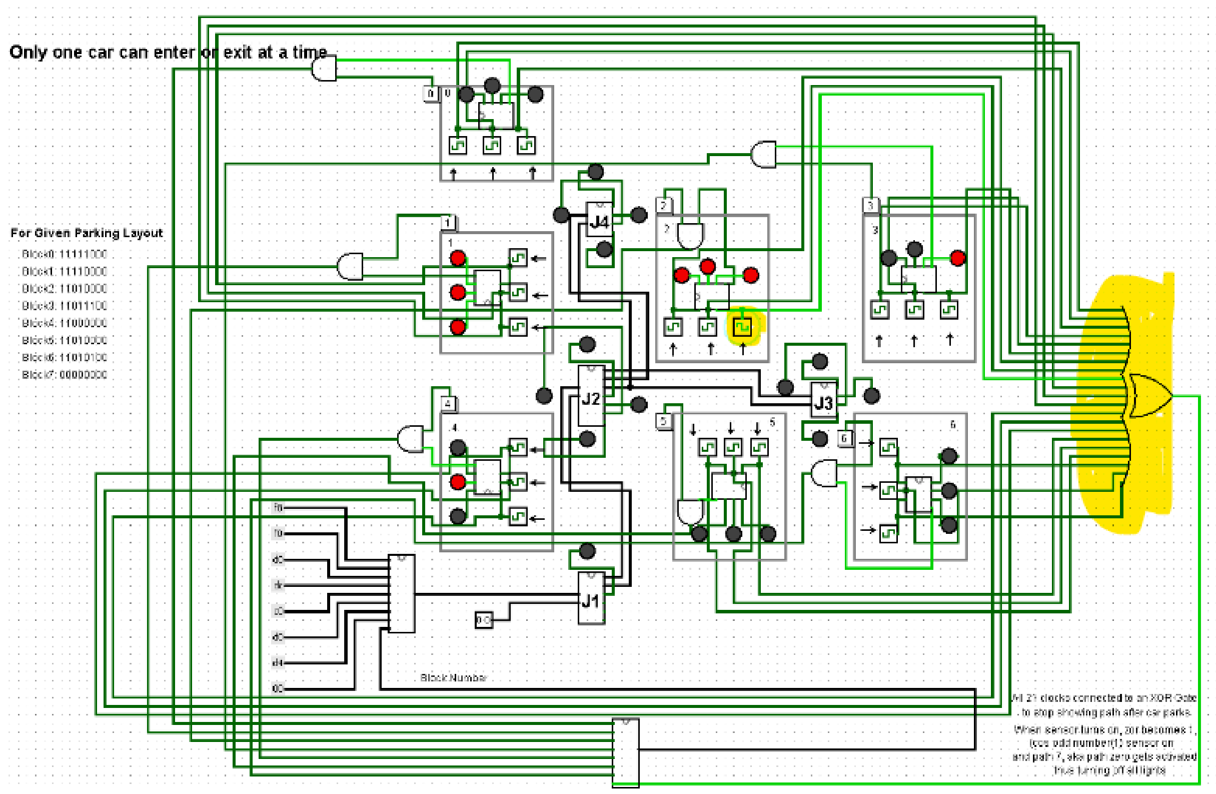
As we can see
We will encounter junction1 from the south, so input direction is 00 and at junction1, we consider the first 2 bits from PathData which is 11
Junction displays north light
After left shift, new pathdata is 11100000 for junction 2

We will encounter junction2 from the south, so input direction is 00 and at junction2, we consider the first 2 bits from PathData which is 11
Junction displays north light
Because we chose north light
We pass data to junction 4 and 0s to junction 3
Due to which junction 3 doesnt display anything
After left shift, new pathdata is 10000000 for junction 4

As we can see
We will encounter junction1 from the south, so input direction is 00 and at junction1, we consider the first 2 bits from PathData which is 10
Junction displays east light
There are no mor junction, so code ends

Now, for the Car sensors,

Let us take an example
Now, when the user enters the 3rd slot at block2
The third sensor(clock), which is present at the entrance of the slot turns on
Which means, the person is going to enter in that slot


Only one car can enter or exit at a time
For Given Parking Layout
Block0: 11111000
Block1: 11110000
Block2: 11010000
Block3: 11011100
Block4: 11000000
Block5: 11010000
Block6: 11010100
Block7: 00000000
J1
J2
J3
J4
Block Number
All 21 clocks connected to an XOR Gate to stop showing path after car parks.
When sensor turns on, xor becomes 1, (cos odd number(1) sensor on and path 7, aka path zero gets activated thus turning off all lights

Now that this clock is turned on, the path towards block2 disappears
Bcos the car will be parked in this slot now, and thus doesnt need the directions anymore

This is done by using button7 which will always display path 00000000 throught which none of the junctions display anything
This button input is connected for port7 in block chooser

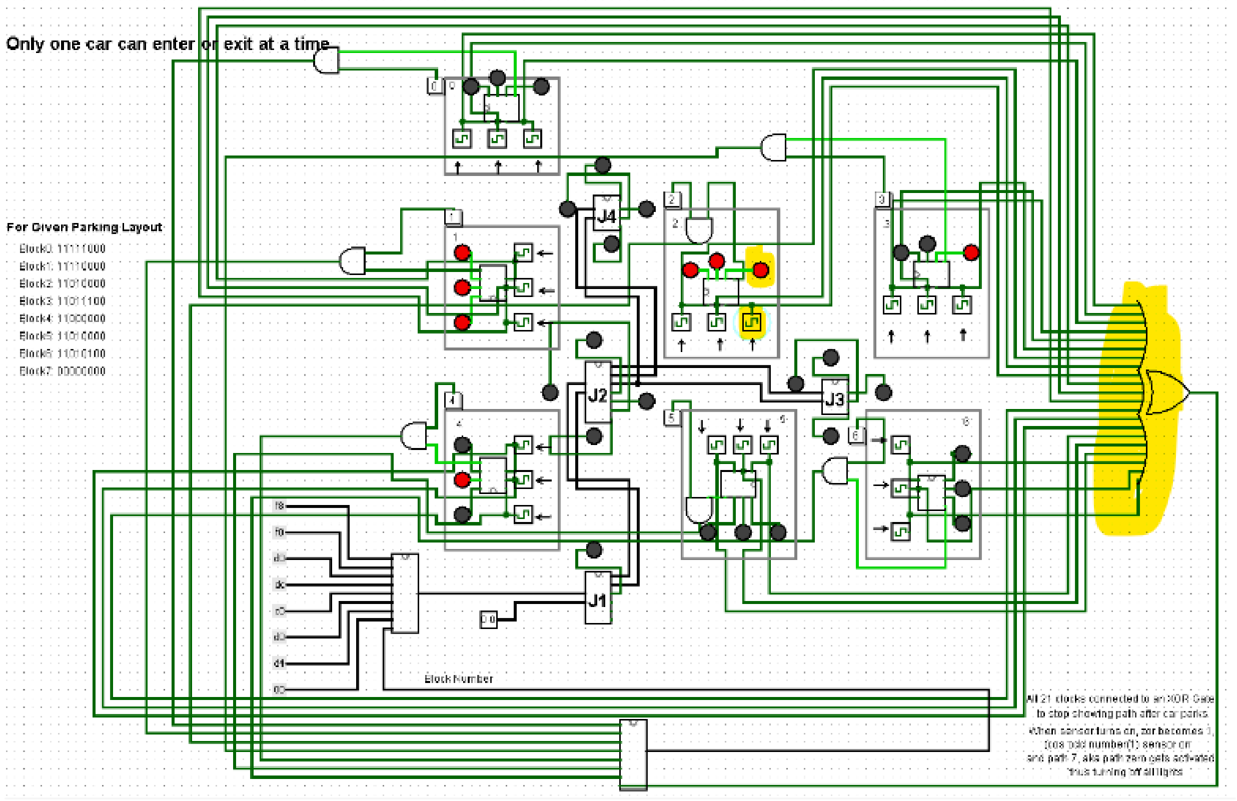
Button 7 is an xor gate to all 21 clocks
Earlier all clocks were 0, so 0 1s, even parity, so XOR returns 0
So button 7 is not chosen

NOTE: just saying button, but it is not realy a button

But when that clock is turned on while car is entering
1 clock shows 1, so 1 1 times, odd parity, so XOR returns 1
So button 7 is chosen, and as mentioned earlier, all junctions dont display anything

Once the car parks in that slot
He will cross the entrance of that slot
Thus, the sensor(clock) for that slot becomes 0 again

Button 7 is an xor gate to all 21 clocks
again all clocks were 0, so 0 1s, even parity, so XOR returns 0
So button 7 is not chosen
So as per the PathChooser code, the output is block7 for which all junctions still display nothing



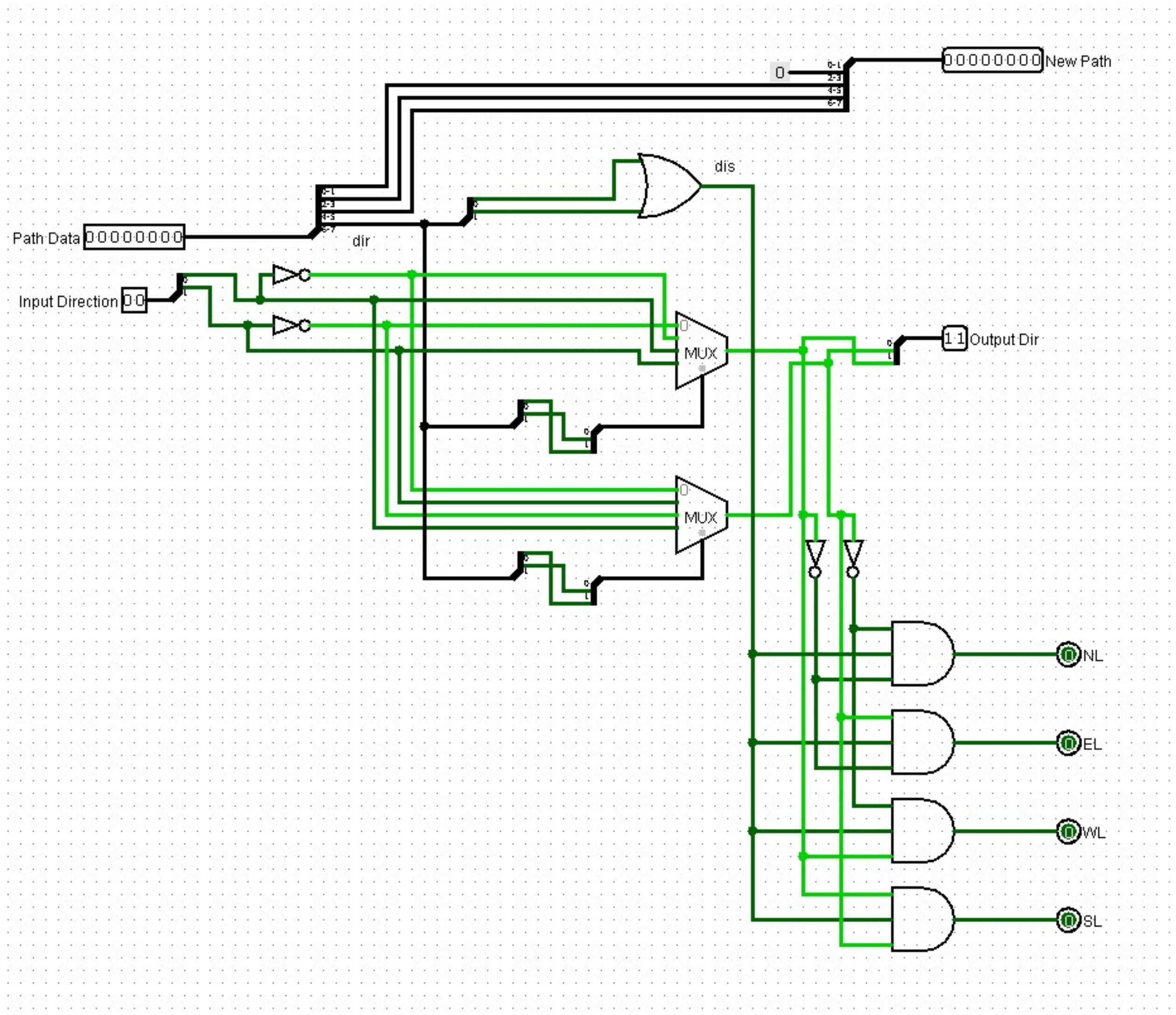
For Given Parking Layout
Block0: 11111000
Block1: 11110000
Block2: 11010000
Block3: 11011100
Block4: 11000000
Block5: 11010000
Block6: 11010100
Block7: 00000000

All 21 clocks connected to an XOR Gate
to stop showing path after car parks
When sensor turns on, xor becomes 1,
(cos odd number(1) sensor on
and path 7, aka path zero gets activated
thus turning off all lights

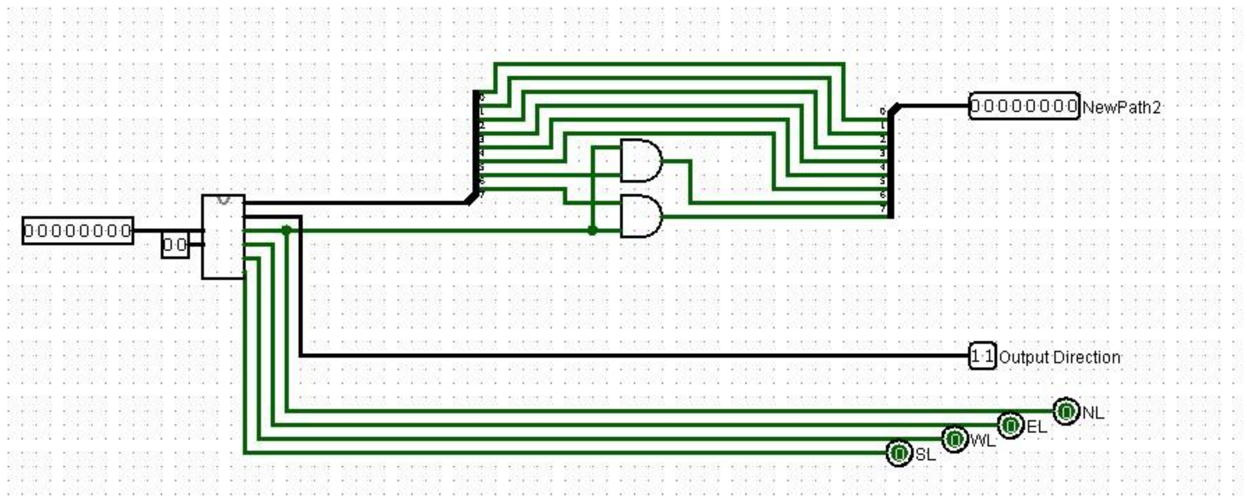
As we can see,
Now, on the screen we can see that the third slot of block 2 which was previously empty is now full as displayed by the LED after the current user parked in that slot

**Logisim Circuit Diagram:**

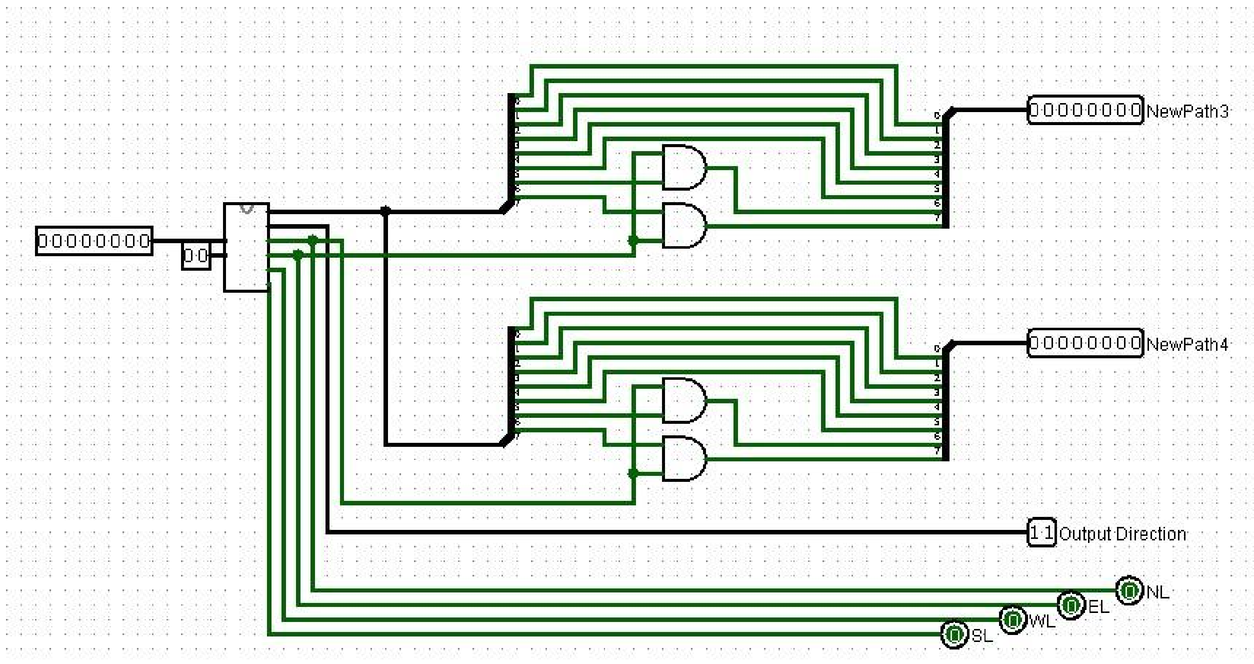
Basic Junction Circuit



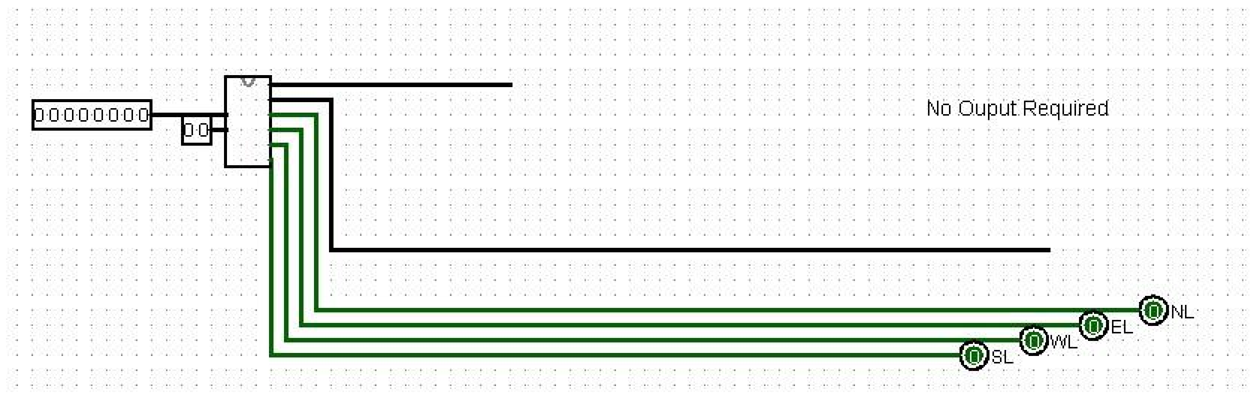
Junction1



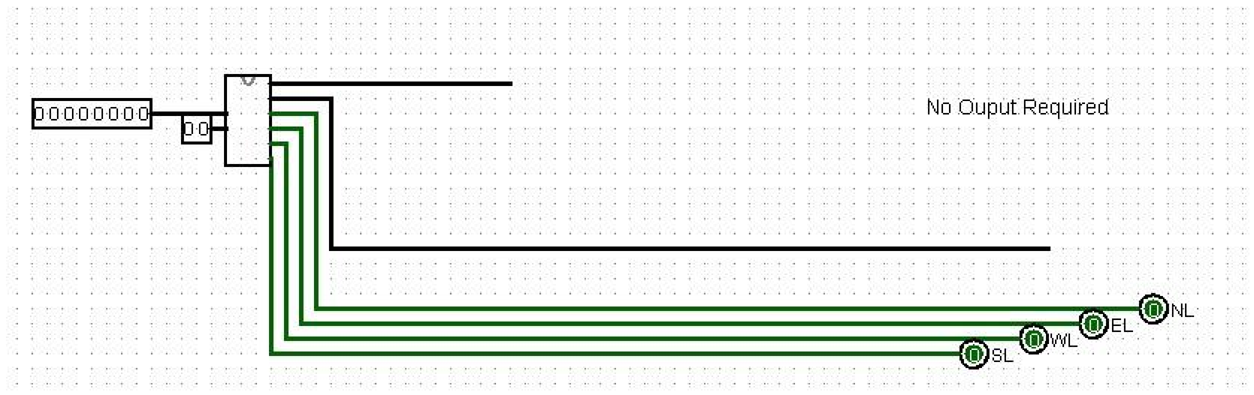
## Junction2



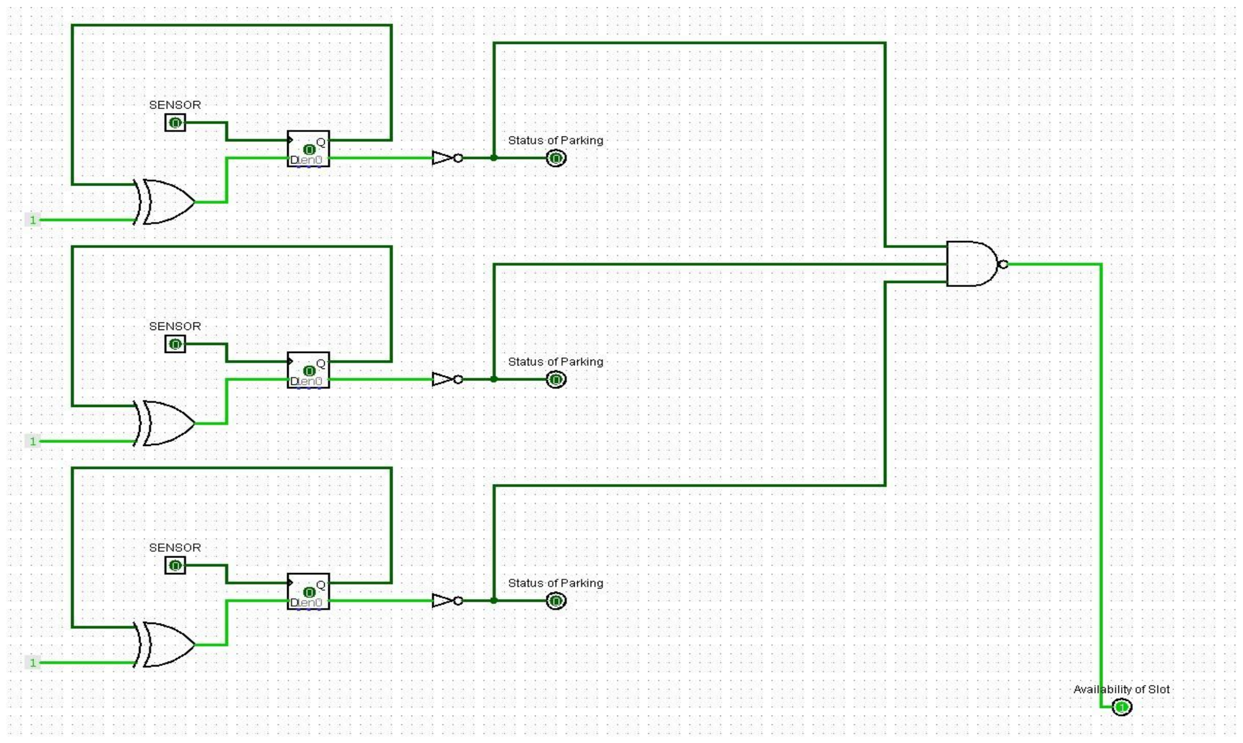
## Junction3



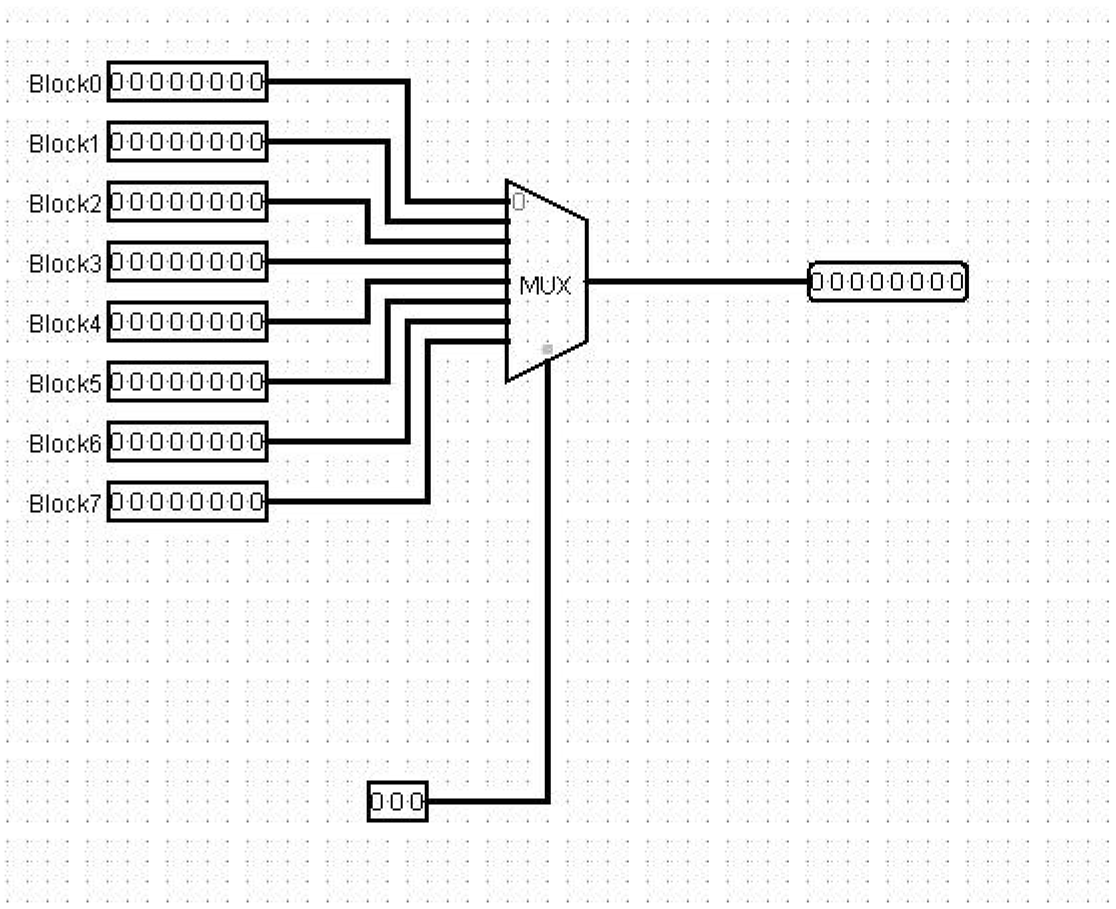
## Junction4



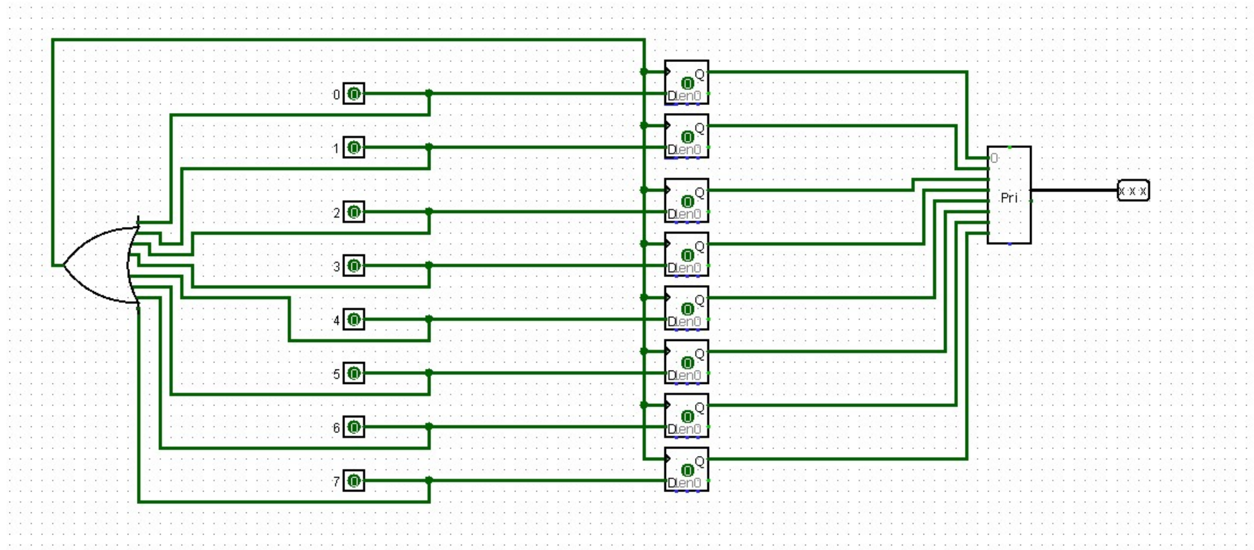
## Car Sensor



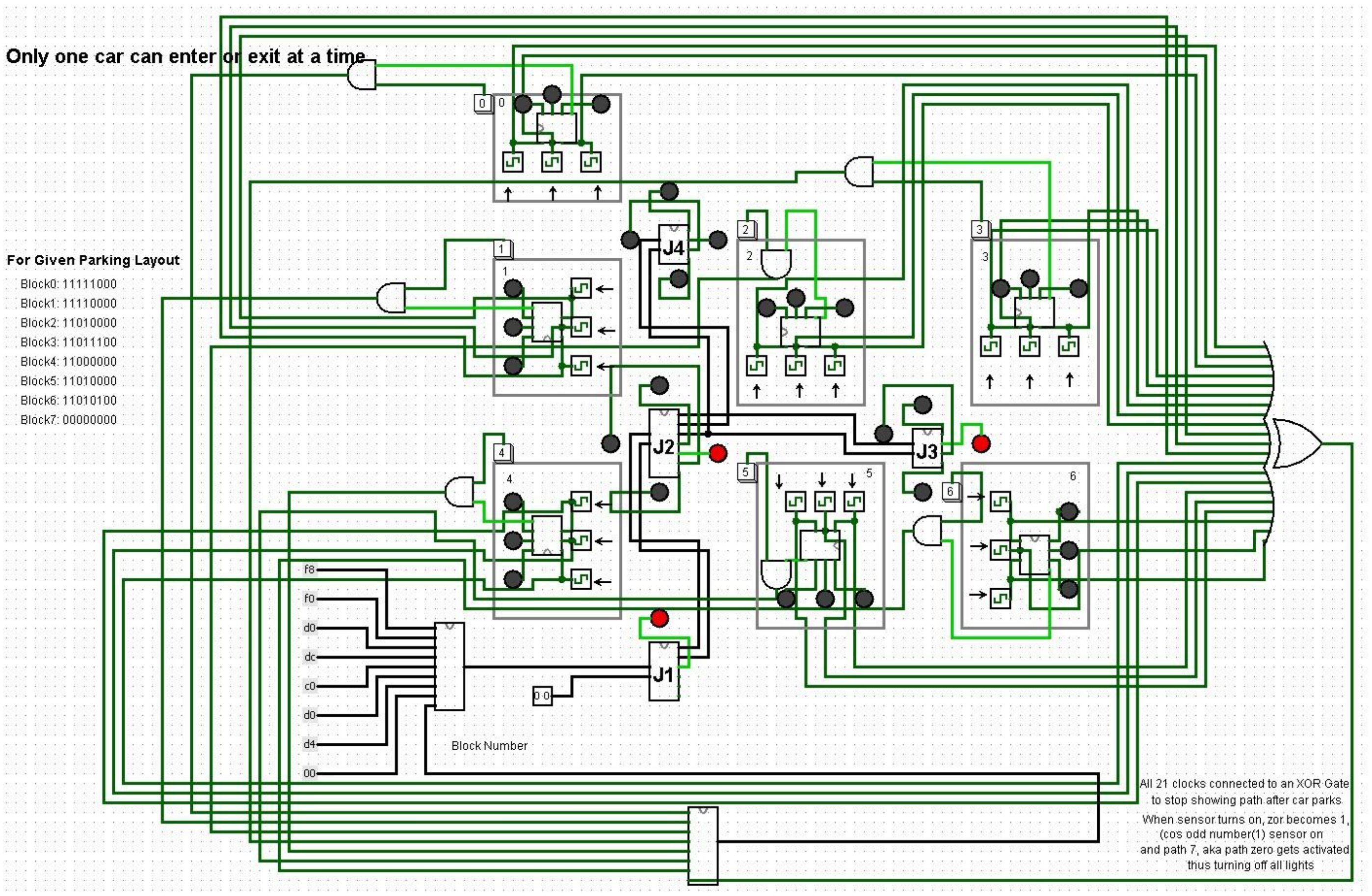
## PathGenerator



BlockChooser



Final Circuit Diagram


Only one car can enter or exit at a time
For Given Parking Layout
Block0: 11111000
Block1: 11110000
Block2: 11010000
Block3: 11011100
Block4: 11000000
Block5: 11010000
Block6: 11010100
Block7: 00000000
J1
J2
J3
J4
f8
f0
d0
dc
c0
d0
d4
00
Block Number
All 21 clocks connected to an XOR Gate
to stop showing path after car parks
When sensor turns on, zor becomes 1,
(cos odd number(1) sensor on
and path 7, aka path zero gets activated
thus turning off all lights

## Verilog Codes:

### Junction Code

```
module mux(a,b,c,d,sel,out);   // 4*1 mux

input a,b,c,d;
input [1:0]sel;
output out;

assign out = (!sel[1] && !sel[0] && a)
             || (!sel[1] && sel[0] && b)
             || (sel[1] && !sel[0] && c)
             || (sel[1] && sel[0] && d);
```

```
endmodule


module Junction4(InputDir,PathData,NL,SL,EL,WL);
   input [1:0]InputDir;
   output [1:0]OutputDir;
   input [7:0]PathData;
   output [7:0]NewPath;
   output NL,SL,EL,WL;

   wire [1:0]dir;
   assign dir = PathData[7:6];

   wire dis;
   assign dis = dir[0] || dir[1];

   mux m1(!InputDir[1],!InputDir[0],InputDir[0],InputDir[1],dir,OutputDir[1]);
   mux m2(!InputDir[0],InputDir[1],!InputDir[1],InputDir[0],dir,OutputDir[0]);


   assign NL = dis && (!OutputDir[1] && !OutputDir[0]);
   assign SL = dis && (OutputDir[1] && OutputDir[0]);
   assign EL = dis && (OutputDir[1] && !OutputDir[0]);
   assign WL = dis && (!OutputDir[1] && OutputDir[0]);

   assign NewPath = PathData << 2;


endmodule


module Junction4_tb;

   reg [1:0] InputDir;
   reg [7:0] PathData;
   wire NL, SL, EL, WL;
   wire [1:0] OutputDir;

   Junction4 uut (InputDir,PathData,NL,SL,EL,WL);


   initial begin
```

```
        $dumpfile("Junction4.vcd");
        $dumpvars(0, Junction4_tb);

        $display("-------------------------------------------------");
        $display("| InputDir | PathData    | NL  | SL  | EL  | WL  |");
        $display("-------------------------------------------------");
        $monitor("|    %b    |   %b  |  %b  |  %b  |  %b  |  %b  |",
                 InputDir, PathData, NL, SL, EL, WL);


        InputDir = 2'b00;
        PathData = 8'b00000000;
        #10 PathData = 8'b01000000;
        #20 PathData = 8'b10000000;
        #30 PathData = 8'b11000000;

        #40
        InputDir = 2'b11;
        PathData = 8'b00000000;
        #50 PathData = 8'b01000000;
        #60 PathData = 8'b10000000;
        #70 PathData = 8'b11000000;

        #80
        InputDir = 2'b01;
        PathData = 8'b00000000;
        #90 PathData = 8'b01000000;
        #100 PathData = 8'b10000000;
        #110 PathData = 8'b11000000;

        #120
        InputDir = 2'b10;
        PathData = 8'b00000000;
        #130 PathData = 8'b01000000;
        #140 PathData = 8'b10000000;
        #150 PathData = 8'b11000000;

        #1000 $display("-------------------------------------------------");

    end
endmodule
```

**Junction Simulation Code**

```
//We choose path data such that there is only one unique input direction to each
junction
// Eg. The only path to Junction3 is 1101, 11 at Junction1 and 01 at Junction2
// Junction3 can only recieve input from Junction2

//Block input will be given from the Button code

module
Display(Block,PathData,NL1,SL1,EL1,WL1,NL2,SL2,EL2,WL2,NL3,SL3,EL3,WL3,NL4,
SL4,EL4,WL4);

   input [2:0]Block;
   wire [1:0]InputDir;
   output [7:0]PathData;

   assign InputDir=2'b00;
   PathFinder m1(Block,PathData);

   output NL1,SL1,EL1,WL1,NL2,SL2,EL2,WL2,NL3,SL3,EL3,WL3,NL4,SL4,EL4,WL4;

   wire [1:0]InputDir2,InputDir3,InputDir4;
   wire [1:0]OutputDir1,OutputDir2;
   wire [7:0]NewPath1,NewPath2;
   wire [7:0]PathData2,PathData3,PathData4;

   Junction1 j1(InputDir,PathData,NL1,SL1,EL1,WL1,OutputDir1,NewPath1);
   assign InputDir2 = OutputDir1;
   assign PathData2[7] = NewPath1[7] && NL1;
   assign PathData2[6] = NewPath1[6] && NL1;
   assign PathData2[5:0] = NewPath1[5:0];

   Junction2 j2(InputDir2,PathData2,NL2,SL2,EL2,WL2,OutputDir2,NewPath2);
   assign InputDir3 = OutputDir2;
   assign PathData3[7] = NewPath2[7] && EL2;
   assign PathData3[6] = NewPath2[6] && EL2;
   assign PathData3[5:0] = NewPath2[5:0];
   assign InputDir4 = OutputDir2;
   assign PathData4[7] = NewPath2[7] && NL2;
```

```
    assign PathData4[6] = NewPath2[6] && NL2;
    assign PathData4[5:0] = NewPath2[5:0];

    Junction3 j3(InputDir3,PathData3,NL3,SL3,EL3,WL3);
    Junction4 j4(InputDir4,PathData4,NL4,SL4,EL4,WL4);

endmodule

//Multiplexer for each out index required for PathFinder
module mux1(a,i0,i1,i2,i3,i4,i5,i6,i7,o);

input [2:0]a;
input i0,i1,i2,i3,i4,i5,i6,i7;
output o;

assign o = (!a[2] && !a[1] && !a[0] && i0) ||
   (!a[2] && !a[1] && a[0] && i1) ||
   (!a[2] && a[1] && !a[0] && i2) ||
   (!a[2] && a[1] && a[0] && i3) ||
   (a[2] && !a[1] && !a[0] && i4) ||
   (a[2] && !a[1] && a[0] && i5) ||
   (a[2] && a[1] && !a[0] && i6) ||
   (a[2] && a[1] && a[0] && i7);

endmodule


module PathFinder(a,out);        //Like 8*1 mux

input [2:0]a;
output [7:0]out;
wire [7:0]a0,a1,a2,a3,a4,a5,a6,a7;

assign a0=8'b11111000;
assign a1=8'b11110000;
assign a2=8'b11010000;
assign a3=8'b11011100;
assign a4=8'b11000000;
assign a5=8'b11010000;
assign a6=8'b11010100;
assign a7=8'b00000000;
```

```
mux1 m7(a,a0[7],a1[7],a2[7],a3[7],a4[7],a5[7],a6[7],a7[7],out[7]);
mux1 m6(a,a0[6],a1[6],a2[6],a3[6],a4[6],a5[6],a6[6],a7[6],out[6]);
mux1 m5(a,a0[5],a1[5],a2[5],a3[5],a4[5],a5[5],a6[5],a7[5],out[5]);
mux1 m4(a,a0[4],a1[4],a2[4],a3[4],a4[4],a5[4],a6[4],a7[4],out[4]);
mux1 m3(a,a0[3],a1[3],a2[3],a3[3],a4[3],a5[3],a6[3],a7[3],out[3]);
mux1 m2(a,a0[2],a1[2],a2[2],a3[2],a4[2],a5[2],a6[2],a7[2],out[2]);

assign out[1]=1'b0;
assign out[0]=1'b0;


endmodule



module mux(a,b,c,d,sel,out);   // 4*1 mux

input a,b,c,d;
input [1:0]sel;
output out;

assign out = (!sel[1] && !sel[0] && a)
             || (!sel[1] && sel[0] && b)
             || (sel[1] && !sel[0] && c)
             || (sel[1] && sel[0] && d);

endmodule


module Junction1(InputDir,PathData,NL,SL,EL,WL,OutputDir,NewPath);
   input [1:0]InputDir;
   output [1:0]OutputDir;
   input [7:0]PathData;
   output [7:0]NewPath;
   output NL,SL,EL,WL;

   wire [1:0]dir;
   assign dir = PathData[7:6];

   wire dis;
   assign dis = dir[0] || dir[1];
```

```
    mux m1(!InputDir[1],!InputDir[0],InputDir[0],InputDir[1],dir,OutputDir[1]);
    mux m2(!InputDir[0],InputDir[1],!InputDir[1],InputDir[0],dir,OutputDir[0]);


    assign NL = dis && (!OutputDir[1] && !OutputDir[0]);
    assign SL = dis && (OutputDir[1] && OutputDir[0]);
    assign EL = dis && (OutputDir[1] && !OutputDir[0]);
    assign WL = dis && (!OutputDir[1] && OutputDir[0]);

    assign NewPath = PathData << 2;

endmodule

module Junction2(InputDir,PathData,NL,SL,EL,WL,OutputDir,NewPath);
    input [1:0]InputDir;
    output [1:0]OutputDir;
    input [7:0]PathData;
    output [7:0]NewPath;
    output NL,SL,EL,WL;

    wire [1:0]dir;
    assign dir = PathData[7:6];

    wire dis;
    assign dis = dir[0] || dir[1];

    mux m1(!InputDir[1],!InputDir[0],InputDir[0],InputDir[1],dir,OutputDir[1]);
    mux m2(!InputDir[0],InputDir[1],!InputDir[1],InputDir[0],dir,OutputDir[0]);


    assign NL = dis && (!OutputDir[1] && !OutputDir[0]);
    assign SL = dis && (OutputDir[1] && OutputDir[0]);
    assign EL = dis && (OutputDir[1] && !OutputDir[0]);
    assign WL = dis && (!OutputDir[1] && OutputDir[0]);

    assign NewPath = PathData << 2;

endmodule


module Junction3(InputDir,PathData,NL,SL,EL,WL);
    input [1:0]InputDir;
    output [1:0]OutputDir;
    input [7:0]PathData;
    output [7:0]NewPath;
    output NL,SL,EL,WL;

    wire [1:0]dir;
    assign dir = PathData[7:6];

    wire dis;
    assign dis = dir[0] || dir[1];

    mux m1(!InputDir[1],!InputDir[0],InputDir[0],InputDir[1],dir,OutputDir[1]);
    mux m2(!InputDir[0],InputDir[1],!InputDir[1],InputDir[0],dir,OutputDir[0]);


    assign NL = dis && (!OutputDir[1] && !OutputDir[0]);
    assign SL = dis && (OutputDir[1] && OutputDir[0]);
    assign EL = dis && (OutputDir[1] && !OutputDir[0]);
    assign WL = dis && (!OutputDir[1] && OutputDir[0]);

    assign NewPath = PathData << 2;


endmodule


module Junction4(InputDir,PathData,NL,SL,EL,WL);
    input [1:0]InputDir;
    output [1:0]OutputDir;
    input [7:0]PathData;
    output [7:0]NewPath;
    output NL,SL,EL,WL;

    wire [1:0]dir;
    assign dir = PathData[7:6];

    wire dis;
    assign dis = dir[0] || dir[1];

    mux m1(!InputDir[1],!InputDir[0],InputDir[0],InputDir[1],dir,OutputDir[1]);
    mux m2(!InputDir[0],InputDir[1],!InputDir[1],InputDir[0],dir,OutputDir[0]);
    assign NL = dis && (!OutputDir[1] && !OutputDir[0]);
    assign SL = dis && (OutputDir[1] && OutputDir[0]);
    assign EL = dis && (OutputDir[1] && !OutputDir[0]);
    assign WL = dis && (!OutputDir[1] && OutputDir[0]);

    assign NewPath = PathData << 2;


endmodule




module Display_tb;

    reg [2:0] Block;
wire [7:0] PathData;
    wire NL1, SL1, EL1, WL1;
    wire NL2, SL2, EL2, WL2;
    wire NL3, SL3, EL3, WL3;
    wire NL4, SL4, EL4, WL4;

    Display
d1(Block,PathData,NL1,SL1,EL1,WL1,NL2,SL2,EL2,WL2,NL3,SL3,EL3,WL3,NL4,SL4,
EL4,WL4);

    initial begin
        $dumpfile("Display.vcd");
        $dumpvars(0, Display_tb);


$display("-----------------------------------------------------------------------------------------------------
------------");
        $display("| Block | PathData | NL1 | SL1 | EL1 | WL1 | NL2 | SL2 | EL2 | WL2 | NL3
| SL3 | EL3 | WL3 | NL4 | SL4 | EL4 | WL4 |");

$display("-----------------------------------------------------------------------------------------------------
------------");
        $monitor("|  %b  | %b |  %b  |  %b  |  %b  |  %b  |  %b  |  %b  |  %b  |  %b  |  %b  |
%b  |  %b  |  %b  |  %b  |  %b  |  %b  |  %b  |",
            Block, PathData, NL1, SL1, EL1, WL1, NL2, SL2, EL2, WL2, NL3, SL3, EL3,
WL3, NL4, SL4, EL4, WL4);

Block=3'b000;
      repeat(7)
   begin
   #10 Block= Block + 3'b001;
  end


      #1000
$display("-----------------------------------------------------------------------------------------------------
------------");

   end
endmodule
```

**Block Circuit Code**

```
module dflipflop (input D, input CKT, output reg Q);
   always @(posedge CKT) begin
      Q <= D;
   end
endmodule

module block (input D1, input D2, input D3, input CKT1, input CKT2, input CKT3, output
Q1, output Q2, output Q3, output Free);
   wire Q1_wire, Q2_wire, Q3_wire;

   dflipflop s1 (.D(D1), .CKT(CKT1), .Q(Q1_wire));
   dflipflop s2 (.D(D2), .CKT(CKT2), .Q(Q2_wire));
   dflipflop s3 (.D(D3), .CKT(CKT3), .Q(Q3_wire));

   assign Q1 = Q1_wire;
   assign Q2 = Q2_wire;
   assign Q3 = Q3_wire;

   nand m1(Free, Q1,Q2,Q3);
```

```
endmodule


module block_tb;

reg D1,D2,D3,CKT1=0,CKT2=0,CKT3=0;
wire Q1,Q2,Q3,free;
block d(D1,D2,D3,CKT1,CKT2,CKT3,Q1,Q2,Q3,free);

always begin
   CKT1=~CKT1;
   #10;
end

always begin
   CKT2=~CKT2;
   #20;
end

always begin
   CKT3=~CKT3;
   #30;
end

initial begin
   $dumpfile("block.vcd");
   $dumpvars(0,block_tb);
end
initial begin
   $display("|Q1 |Q2 |Q3 |Free?|");
   $monitor("| %b | %b | %b |  %b  |",Q1,Q2,Q3,free);
   D1=1'b0;D2=1'b0;D3=1'b0;

   #20 D1=1'b1;
   #40 D2=1'b1;
   #60 D3=1'b1;
   #80 D1=1'b0;
   #100 D1=1'b1;
   #120 D1=1'b0;
   #140 D3=1'b0;
   #160 D2=1'b0;
end
```

```
initial #5000 $finish;

endmodule
```

**Block Chooser Code**

```
//NOTE: Input B7 will be XOR of all sensors(clocks) present for each slot
//The output block number will be the button number which is pressed last (when slot is
empty, which is handled in logisim)
//Bk is actually is button pressed & if free slot is available(from BlockCircuit code)
//Note, simulation may not seem perfect since it is not that very feasible to simulate
buttons through veriolog

module dflipflop (input D, input CKT, output reg Q);
   always @(posedge CKT) begin
      Q <= D;
   end
endmodule

module PriorityEncoder(i,y);
   input [7:0]i;
   output [2:0]y;

   assign y[2]=i[4] | i[5] | i[6] | i[7];
   assign y[1]=i[2] | i[3] | i[6] | i[7];
   assign y[0]=i[1] | i[3] | i[5] | i[7];

endmodule

module BlockChooser(B,F);
   input [7:0]B;
   wire CKT;
   wire [7:0]W;
   output [2:0]F;

   or O1(CKT,B[0],B[1],B[2],B[3],B[4],B[5],B[6],B[7]);

   dflipflop s0 (.D(B[0]), .CKT(CKT), .Q(W[0]));
   dflipflop s1 (.D(B[1]), .CKT(CKT), .Q(W[1]));
   dflipflop s2 (.D(B[2]), .CKT(CKT), .Q(W[2]));
```

```
    dflipflop s3 (.D(B[3]), .CKT(CKT), .Q(W[3]));
    dflipflop s4 (.D(B[4]), .CKT(CKT), .Q(W[4]));
    dflipflop s5 (.D(B[5]), .CKT(CKT), .Q(W[5]));
    dflipflop s6 (.D(B[6]), .CKT(CKT), .Q(W[6]));
    dflipflop s7 (.D(B[7]), .CKT(CKT), .Q(W[7]));

    PriorityEncoder p1(B,F);

endmodule


module BlockChooser_tb;

reg [7:0]B;
wire [2:0]F;

BlockChooser b1(B,F);

initial begin
    $dumpfile("BlockChooser.vcd");
    $dumpvars(0,BlockChooser_tb);
end
initial begin
    $display("|B0 |B1 |B2 |B3 |B4 |B5 |B6 |B7 | BlockNumber |");
    $monitor("| %b | %b | %b | %b | %b | %b | %b | %b |     %b
|",B[0],B[1],B[2],B[3],B[4],B[5],B[6],B[7],F);

    //Let B and be something intially
    B=8'b10000000;

    //let Button 5 be pressed now
    #110 B=8'b00100000;
    //button is released now

    //For a while no button is pressed
    //But still, path to block 5 is shown

    //When car is parked, button 7 will get activated
    //Can be understood from logisim code
    //When car is entering, only 1 out of the 21 sensors(clocks) is 1 (odd parity, so xor
gives 1 as output)
    //So the xor part will give output 1 which is button 7
    //So button 7 is pressed technically
    //Once the car is parked
    //That sensor(clock) will return 0
    //So xor of all will become 0 again, as 0 sensors are 1 (0 is even parity, so xor gives 0
as output)
    //So button is released again
    //And path to block 7, empty path is shown
    #170 B=8'b10000000;

    //let Button 1 be pressed now
    #210 B=8'b00000010;
    //button is released now

    //For a while no button is pressed
    //But still, path to block 1 is shown

    //When car is parked, button 7 will get activated
    //Can be understood from logisim code
    //When car is entering, only 1 out of the 21 sensors(clocks) is 1 (odd parity, so xor
gives 1 as output)
    //So the xor part will give output 1 which is button 7
    //So button 7 is pressed technically
    //Once the car is parked
    //That sensor(clock) will return 0
    //So xor of all will become 0 again, as 0 sensors are 1 (0 is even parity, so xor gives 0
as output)
    //So button is released again
    //And path to block 7, empty path is shown
    #270 B=8'b10000000;

end

initial #10000 $finish;

endmodule
```

## References

• Morris Mano, Digital Logic and Computer Design
• https://www.flashparking.com/blog/what-is-an-automated-parkingsystem/
• www.wayleadr.com
• https://www.slideshare.net
• Sunggu Lee, Advanced Digital Logic Design: Using VHDL, State Machines, and Synthesis for FPGAs

## Link to Resource Folder

https://drive.google.com/drive/folders/1qaamQc11U_bVCfyj0aEIp9d-NobUFvID?usp=drive_link
The above link contains the Logisim Project (.circ files), Verilog codes (.v files), Screenshots of the circuits used