PROYECTO DE INSTALACIÓN DE APARCAMIENTO PÚBLICO

ANEXO Nº5

SISTEMA DE CONTROL DE LA OCUPACIÓN DEL PARKING

SISTEMA DE CONTROL DE LA OCUPACIÓN DEL PARKING

1.-GENERALIDADES DEL PROCESO A CONTROLAR.

Solo se realizará el control de las plantas destinadas al parking público, las plantas baja, primera y segunda, ya que la tercera y la cuarta esta destinada a plazas privadas y por tanto sólo pueden acceder a ellas los propietarios que dispongan de las llaves correspondientes.

Al llegar un vehículo a una de las dos entradas del parking, esta encontrará obstruido el paso por una barrera accionada por un motor eléctrico. Aproximadamente a la altura de la ventanilla del conductor existirá un emisor de fichas con bada magnética , el cual dispondrá de un botón y de un display que le emitirá el siguiente mensaje “pulse el botón y retire la ficha”. Automáticamente una vez que el conductor retira la ficha se abrirá la barrera quedando permitido el acceso al vehículo y se cerrará automáticamente unos segundos después tras el paso del vehículo.

Tras atravesar la barrera , el conductor entrará con su vehículo en el montacoches de ascenso que normalmente en paro estará con las puertas abiertas. Una vez dentro el conductor encontrará una botonera que dispondrá de cinco botones, uno para cada planta, que podrá activar a su elección, pero los botones de la tercera y cuarta planta dispondrán de una llave que solo tendrán los propietarios de las plazas privadas de dichas plantas.

Cuando el usuario se seleccione una planta las puertas del montacoches se cerrarán y ascenderá a la planta seleccionada .Al llegar a la planta la puerta de acceso a planta se abrirá y el conductor podrá acceder a ella.

Del mismo modo cuando un usuario desea abandonar con su vehículo cualquiera de las tres primeras plantas, este se dirigirá a uno de los dos montacoches de bajada. Al llegar alguno de ellos , encontrará la puerta de salida de planta del ascensor cerrada. Para efectuar la llamada al ascensor , accionará un pulsador que quedará aproximadamente a la altura de la ventanilla del conductor ;dicho pulsador se encuentra en un panel que dispondrá de un display ,que emitirá el siguiente mensaje al conductor “por favor espere”.Una vez que el ascensor esté libre , al recibir la llamada ,ascenderá automáticamente a la correspondiente planta. Al llegar a planta este abrirá sus puertas y el mismo display le comunicará al conductor que puede acceder al ascensor .

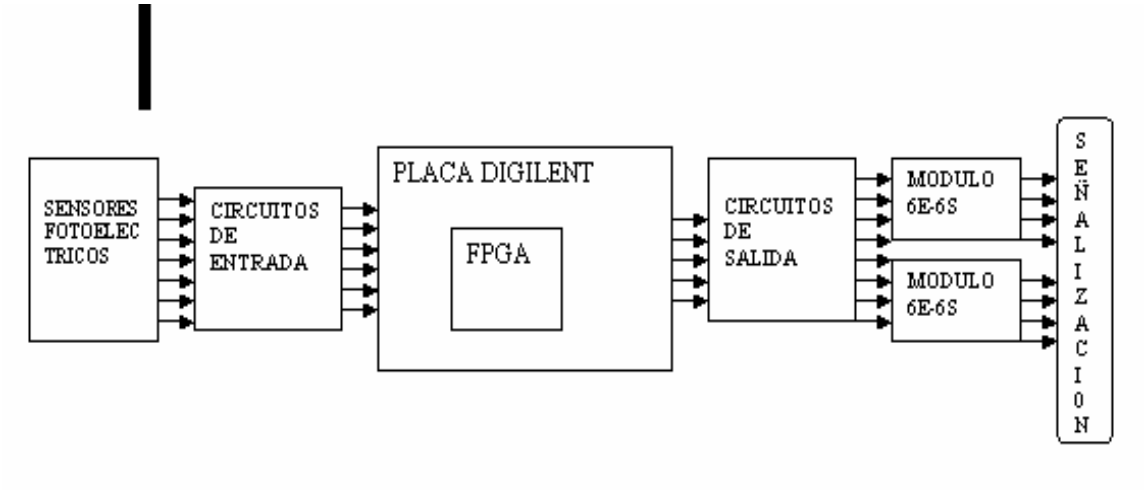
Tras entrar el conductor con su vehículo al ascensor ,encontrará un único botón para el descenso a la planta baja , que tras accionar cerrará la puerta de entrada del

montacoches (al ser de doble embarque) y comenzará la bajada. Al llegar el ascensor a la planta baja del edificio, se abrirá la puerta de salida del ascensor y el conductor podrá conducir su vehículo fuera del ascensor. Sin embargo encontrará otra vez detenido su paso al igual que a la entrada por una barrera accionada por un motor eléctrico. Aproximadamente a la altura de la ventanilla del conductor existirá un receptor de fichas con bada magnética , el cual dispondrá de un botón y de un display que le emitirá el siguiente mensaje “introduzca la ficha en la ranura”. Automáticamente una vez que el conductor introduzca la ficha, se abrirá la barrera, quedando permitido la salida y se cerrará automáticamente unos segundos después, tras la salida del vehículo. La puerta del ascensor se cerrará tras la bajada de la barrera

2.-SISTEMA DE CONTROL

El sistema que pretendemos diseñar nos deberá permitir la cuenta de todos los vehículos que entran y salen del parking en general y de cada una de las plantas. De manera que nos indique mediante una serie de luces situadas en un panel a la entrada del edificio cuando el parking esta completo o libre y lo mismo para cada uno de las plantas. (a excepción de las dos ultimas plantas reservadas para plazas privadas). La luz roja nos indicará que el parking o la plaza está completa y la verde para indicar que todavía quedan plazas.

El diagrama del sistema es el siguiente:



En términos generales este sistema presenta este los siguientes elementos:

Sensores fotoeléctricos por relé, a la salida y entrada de acceso a planta de cada ascensor de subida y bajada de coches, que nos permitirán detectar la entrada y salida de los vehículos.

Diversas fuentes de tensión continua para la alimentación de los sensores y placas de control.

Diferentes tipos de cables y conexiones para la transmisión de las señales.

Optoacopladores para la llegada de los sensores a la placa digilent.

Tarjeta de diseño programable equipada de una fpga ,que programaremos con el correspondiente softwatre para el control del sistema.

Placas electrónicas de expansión de la tarjeta interior.

Un dispositivo CONTROL6E6S: con 6 entradas digitales y 6 salidas digitales de 6 amperios conectadas fase, equipado con una serie de relés ; al que irán conectadas las salidas de la tarjeta de control y de donde saldrán las fases para la iluminación del panel de entrada.

Panel luminoso de señalización, con 8 puntos de luz dos para indicar el estado de cada planta, y otros dos para indicar el estado del parking

3.-PROGRAMACIÓN DE LA FPGA DE LA PLACA DIGILENT:

El objetivo de este apartado es crear el circuito de control que permita el control de la ocupación del parking, e implementarlo en un circuito programable FPGA de la serie SPARTAN tm de XILINX , en concreto el XCS200-PQ208, dispuesto en la tarjeta de programación DIGILENT Digilab2 tal y como la que aparece en la siguiente figura.

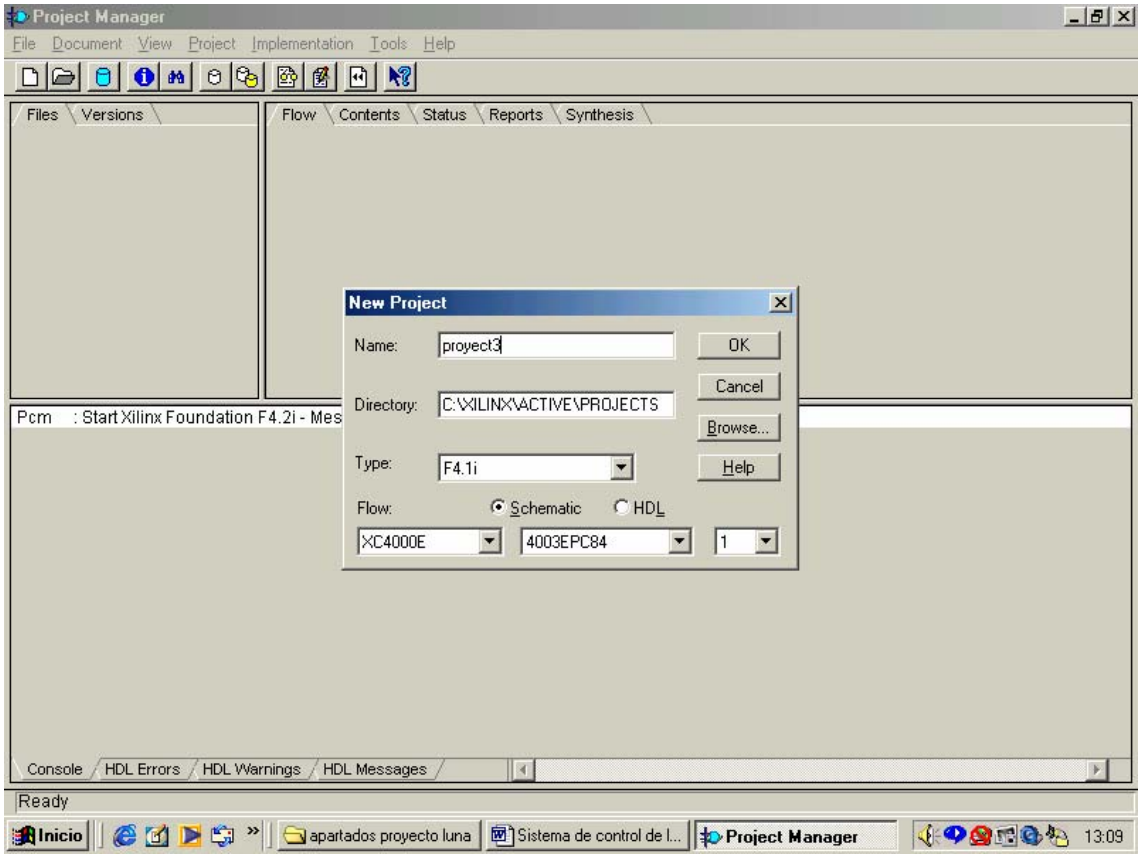
El programa utilizado para lo programación de la fpga de la tarjeta es el entorno de diseño de “XILINS FOUNDATION”.

Xilinx Foundation Series es una herramienta EDA que facilita el diseño lógico programable y que permite obtener resultados automáticamente mediante una consola basada en un diagrama de flujos, esquemáticos o lenguaje VHDL.

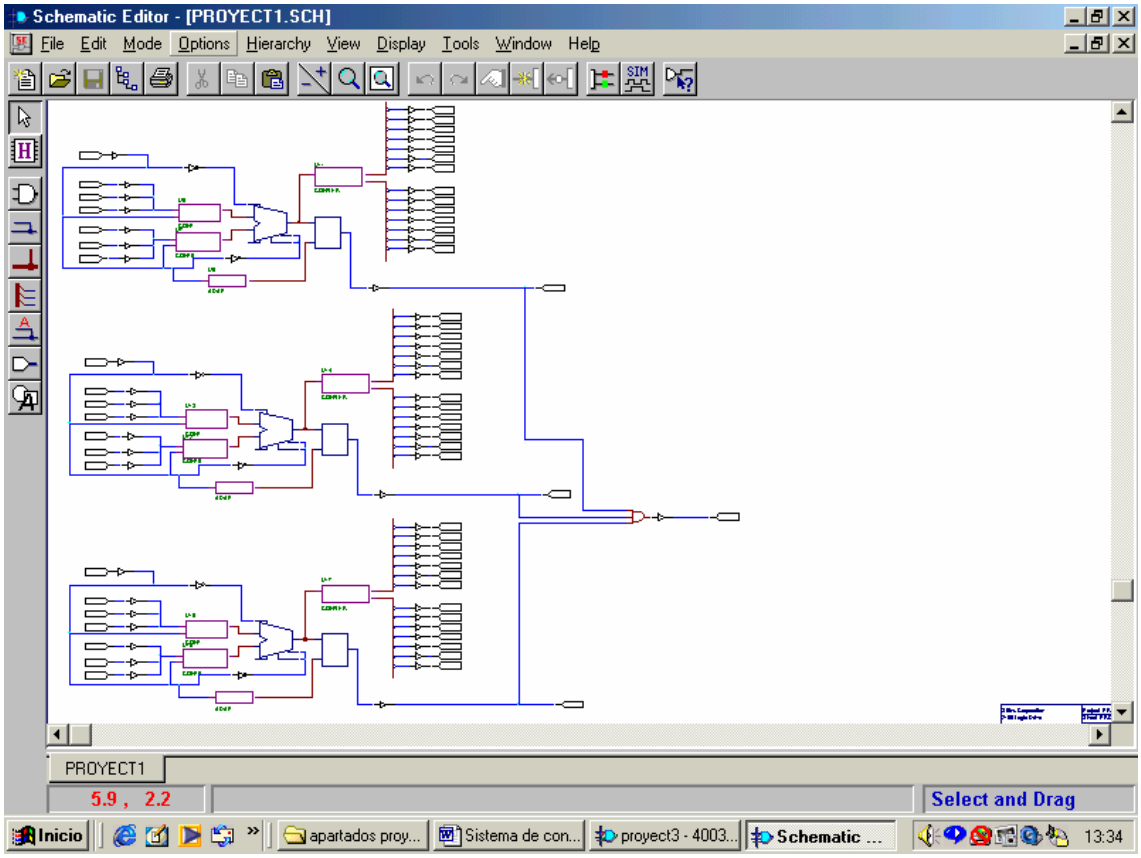
Xilinx Foundation Series incluye:

- Herramientas de captura y modelado de diseño.
- Herramientas de síntesis y optimización.
- *Hierarchy browser*. Organizador de archivos.
- *Project Flowchart.* Control de diseño.

El diseño estará basado en un proyecto basado en esquemático:

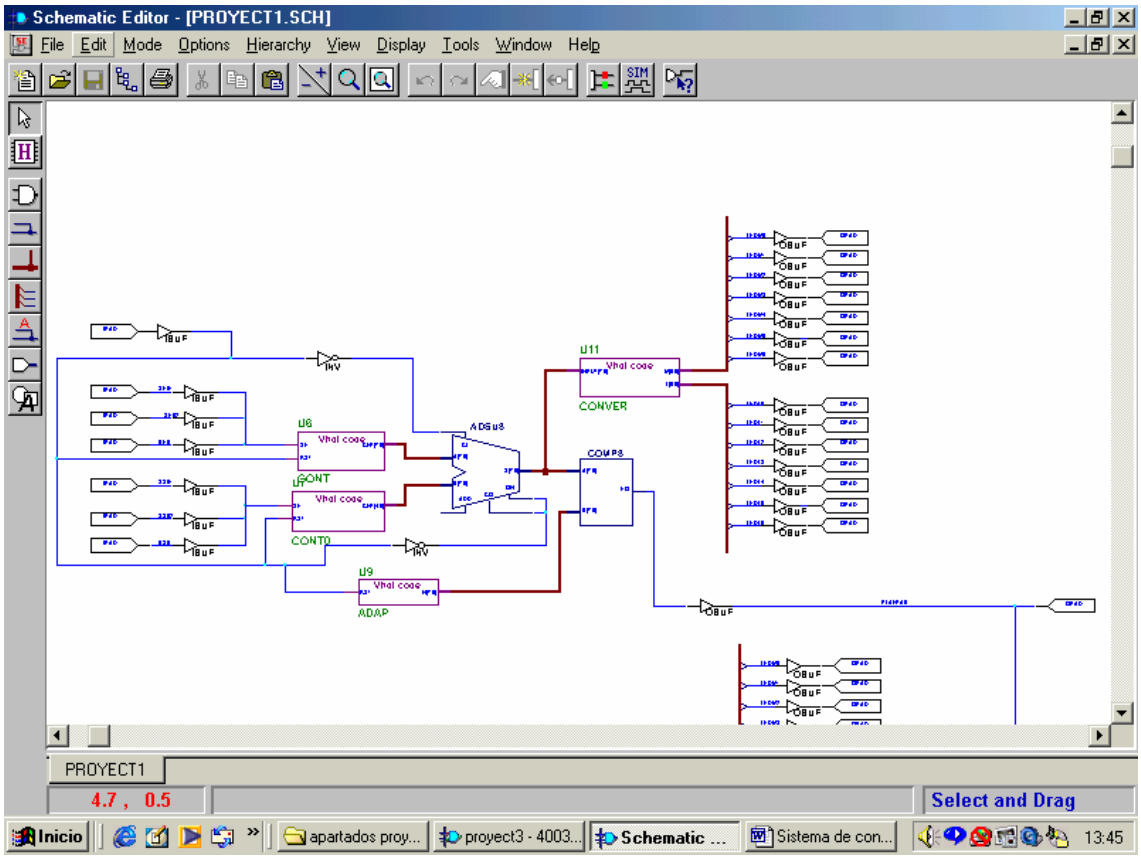


En la siguiente figura podemos observar el esquemático completo con el que implementaremos la FPGA, el modelo del sistema de control:

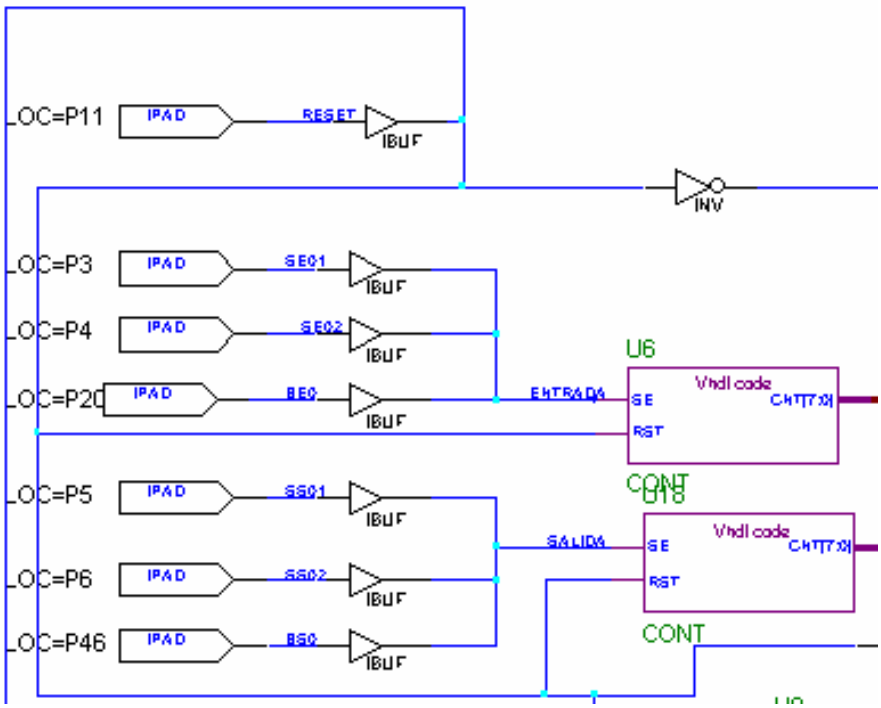


Como se puede observar el esquemático se puede dividir en tres circuitos iguales, una para el control de cada planta , cuyas salidas (para cada planta) se unen al final en una puerta and para el control del parking completo.

Pasaré a tratar uno de estos tres circuitos en profundidad; en la siguiente figura se puede observar mas detalladamente.



Comenzando por las entradas al circuito



Podemos distinguir las siguientes:

RESET: esta entrada nos permitirá el reseteado del sistema.

SEO1: señal del sensor de entrada 1 de la planta 0.

SEO2: señal del sensor de entrada 2 de la planta 0.

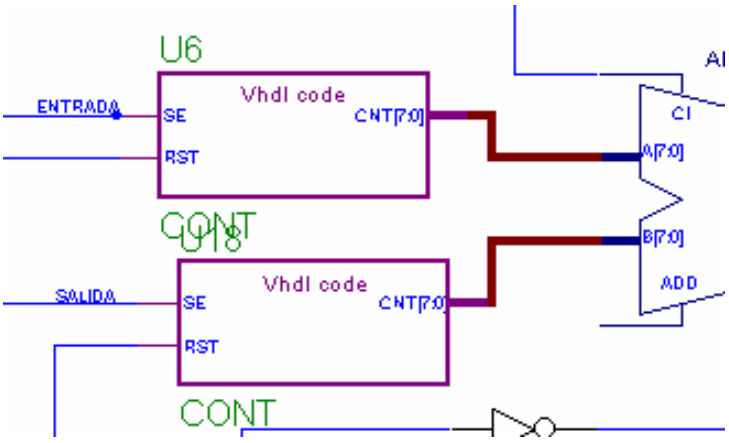
BE0: señal del pulsador de entrada para la planta0 (cuya utilidad describiré mas adelante).

SSO1: señal del sensor de sálida1 de la planta 0.

SSO2: señal del sensor de sálida 2 de la planta 0.

BS0: señal del pulsador de sálida para la planta0 (cuya utilidad describiré mas adelante).

Siguiendo con los dos primeros módulos diseñados que nos encontramos:



Ambos módulos son iguales, basados en lenguaje VHDL y nos permiten el contaje de los vehículos que entran y salen de la planta , el superior para la entrada de vehículos y el inferior para la salida.

Cada módulo dispone de dos entradas, una para la señal de reset (RST) y otra para las señal de los sensores, de entrada para el superior y de salida para el inferior. También disponen de una salida CNT por donde saldrá el resultado del contaje de los vehículos. Dichas salidas servirán de entradas para el modulo restador que hay a continuación.

El correspondiente programa con el que se ha programado es el siguiente:

```
library IEEE;
use IEEE.std_logic_1164.all;
use IEEE.std_logic_arith.all;
use IEEE.std_logic_unsigned.all;


entity CONT is
   port (
      se: in STD_LOGIC;
      rst: in STD_LOGIC;
      cnt: inout STD_LOGIC_VECTOR (7 downto 0)
   );
end CONT;

architecture CONT_arch of CONT is
begin

process (se,rst)

begin

            if rst='1' then
            cnt<="00000000";

            else

                if se='1' and se'event then

                cnt<=cnt+1;

                end if;

        end if;


end process;

end CONT_arch;
```

Pasando a explicarlo brevemente a continuación:

El programa comienza declarando las libreías necesarias

```
library IEEE;
use IEEE.std_logic_1164.all;
use IEEE.std_logic_arith.all;
use IEEE.std_logic_unsigned.all;
```

A continuación define el nombre de la entidad del programa ,con sus correspondientes puertos de entrada y salida indicandoo el tipo de cada uno de ellos.

```
entity CONT is
   port (
      se: in STD_LOGIC;
      rst: in STD_LOGIC;
      cnt: inout STD_LOGIC_VECTOR (7 downto 0)
   );
end CONT;
```

Después define e inicia la arquitectura de la entidad CONT designada con el mismo nombre.

```
architecture CONT_arch of CONT is
begin
```

Después de iniciar la arquitectura encontramos el proceso que nos permite el contaje, estableciendo como señales sensibles, la señal se y rst.

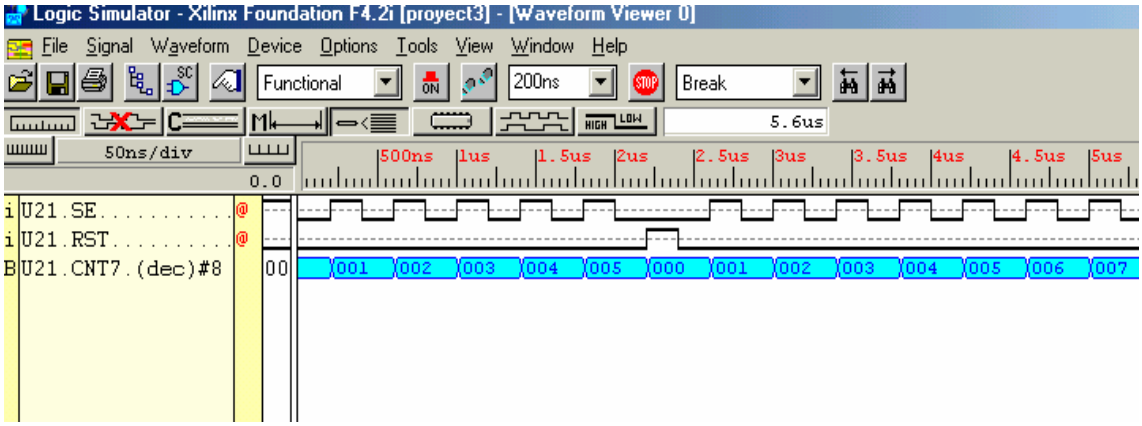
El proceso es el siguiente:

Si en la entrada RST hay un 1 lógico (por el accionamiento de la señal de reset) entonces la salida cnt se inicializará a “00000000”,si no hay un 1 sino un ‘0’ lógico y en la entrada SE se ha producido un evento y se encuentra a nivel alto (como consecuencia de la entrada de un vehículo en el módulo para la entrada, o por la salida de un vehículo en el módulo para la salida) ,entonces la salida cnt se incrementa en una unidad.

```
process (se,rst)

begin

        if rst='1' then
        cnt<="00000000";

        else

            if se='1' and se'event then

            cnt<=cnt+1;

            end if;

    end if;
```

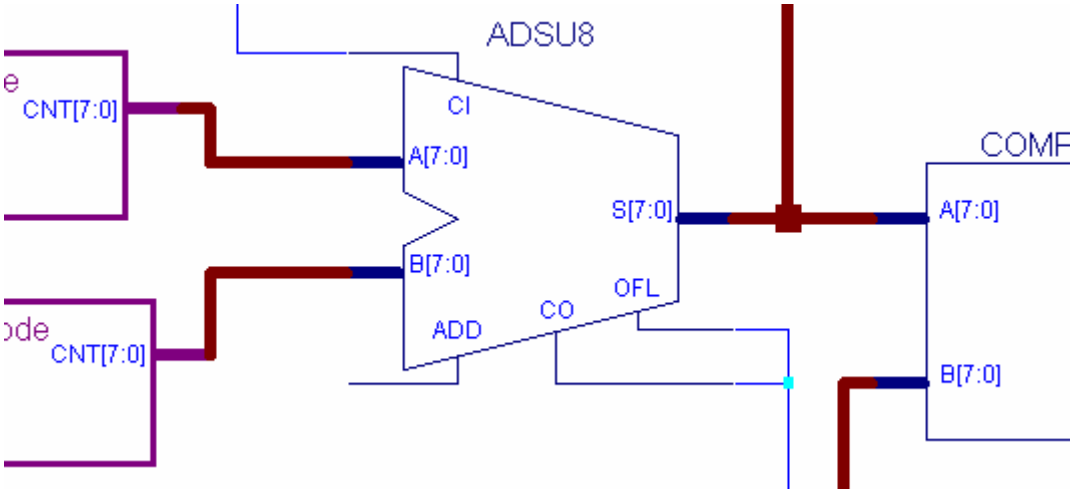
El programa termina con el fin del proceso y con el finde la arquitectura CONT.

Si pasamos a simular el módulo CONT para comprobar su funcionamiento con la herramienta de simulación del entorno de diseño de “XILINS FOUNDATION”.



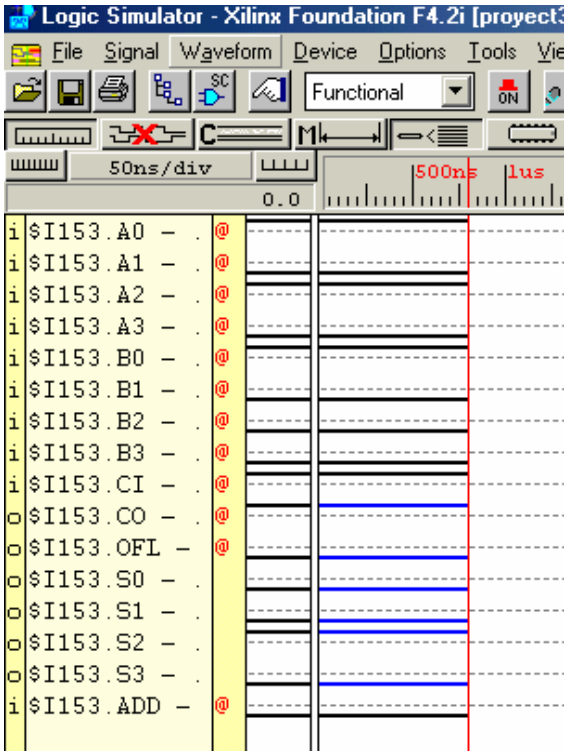
Podemos observar que su funcionamiento es correcto: cada vez que se recibe un pulso en la entrada SE por la entrada o salida del vehículo (dependiendo de si el modulo es de salida o entrada) la salida CNT se incrementa en una unidad, asimismo si activamos la entrada de reset la salida CNT se inicializa y una vez que rst vuelva a ‘o’ el módulo seguirá con su cuenta.

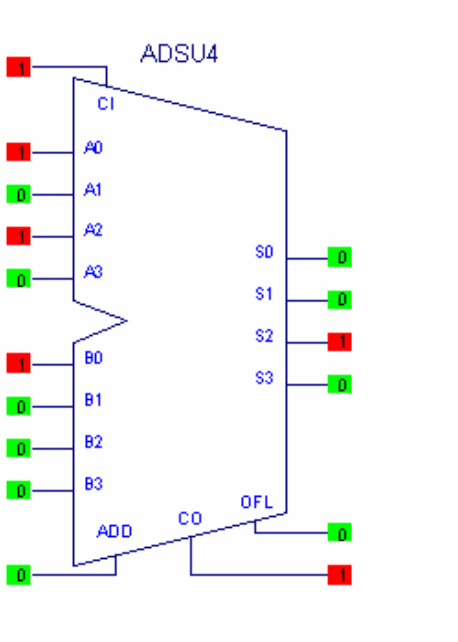
Continuando con los demás módulos del circuito, nos encontramos con el módulo sumador/restador ADSU8 encargado de restar al número de coches que han entrado a la planta con el número que han salido .



Este módulo dispone de la entrada A (a la que le llega el número que han entrado procedente de la salida del módulo CONT superior) la entrada B (a la que le llega el número de vehículos que han salido procedente de la salida del módulo CONT inferior) la entrada ADD (que dejaremos al aire) la entrada CI y las salidas CO y OFL (que deberán de estar a nivel 1 para que se produzca la resta, para lo cual le conectamos la señal de reset negada, ya que cuando esta valga 1 y por tanto tengamos un 0 en las entradas anteriores, no será necesario realizar la resta) y por último la salida S (que reflejará el número de coches en total que hay en dicha planta.

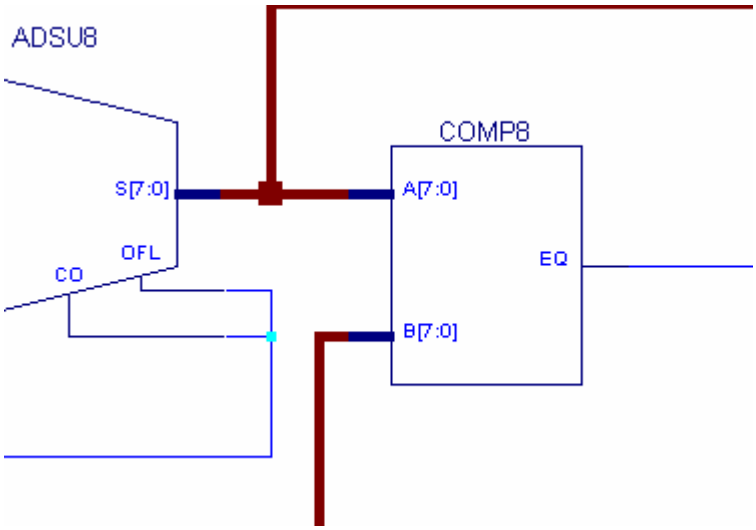
En cuanto a la simulación , realizaremos la simulación del módulo ADSU4 que posee el mismo funcionamiento y características, pero las entrada y salidas tienen la mitad de bits, por lo que su simulación s simplifica bastante.





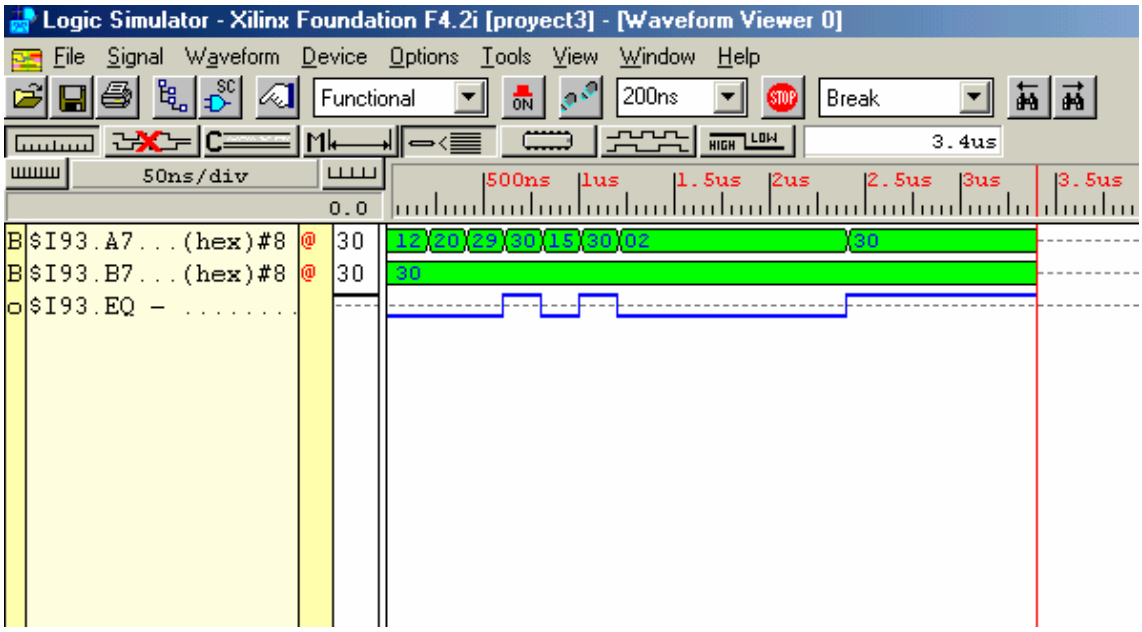
Una vez que tenemos el número total de coches de la planta, debemos compararlo con el número total de plazas (30) de manera que si es igual a 30 se activará la salida correspondiente con la que se accionará el dispositivo necesario de manera que permita el encendido de la luz roja ubicada en el panel de entrada del edificio correspondiente al llenado de dicha planta.

Para la realización de dicha simulación utilizaremos del módulo COMP8:



Como se puede observar, este módulo cuenta con dos entradas, la entrada A a la que le llegará el número total de los coches de dicha planta, procedente del restador y luego la entrada B, a la que le entrará el número treinta (necesario para la comparación) procedente de un módulo que describiré posteriormente.
Dispone de una única salida, por donde saldrá la salida correspondiente al llenado de planta en el caso de ser afirmativa la comparación.

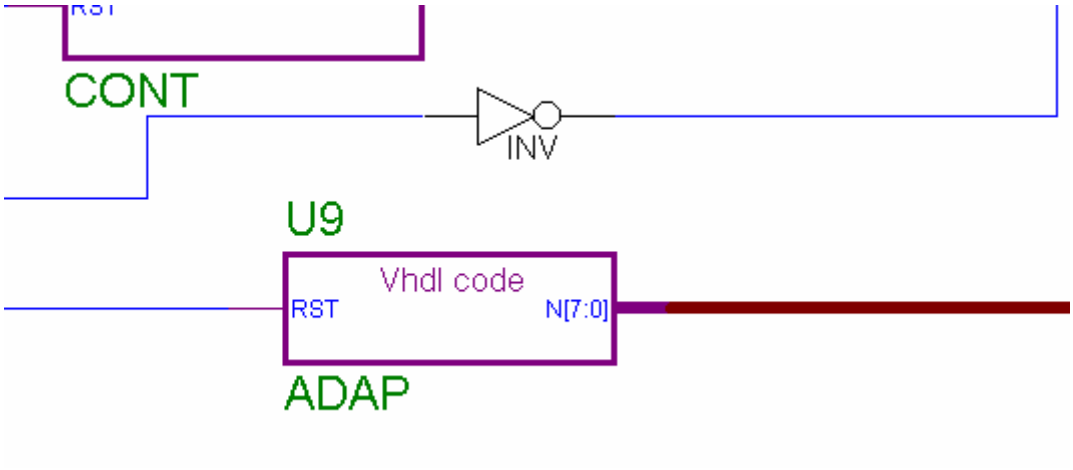
Pasando a simular su funcionamiento:



Como podemos observar su funcionamiento es correcto.

A la salida EQ del comparador va conectada la salida planta0 ubicada en LOC=P55 de la placa utilizada. Por el citado pin saldrá la señal de activación, del encendido de la luz roja que indica el llenado de la planta0.

El número 30 necesario para la comparación, llega a la entrada B del comparador, procedente del siguiente módulo (ADAP).



El programa en VHDL utilizado para la programación de este módulo es el siguiente:

```
library IEEE;
use IEEE.std_logic_1164.all;
use IEEE.std_logic_arith.all;
use IEEE.std_logic_unsigned.all;

entity ADAP is
   port (
      rst: in STD_LOGIC;
      n: out STD_LOGIC_VECTOR (7 downto 0)
   );
end ADAP;

architecture ADAP_arch of ADAP is
begin
process (rst)

begin
```

```
        if rst='1' then

        n<="00000000";


        else

         n<="00011110";


    end if;


end process;
end ADAP_arch;
```

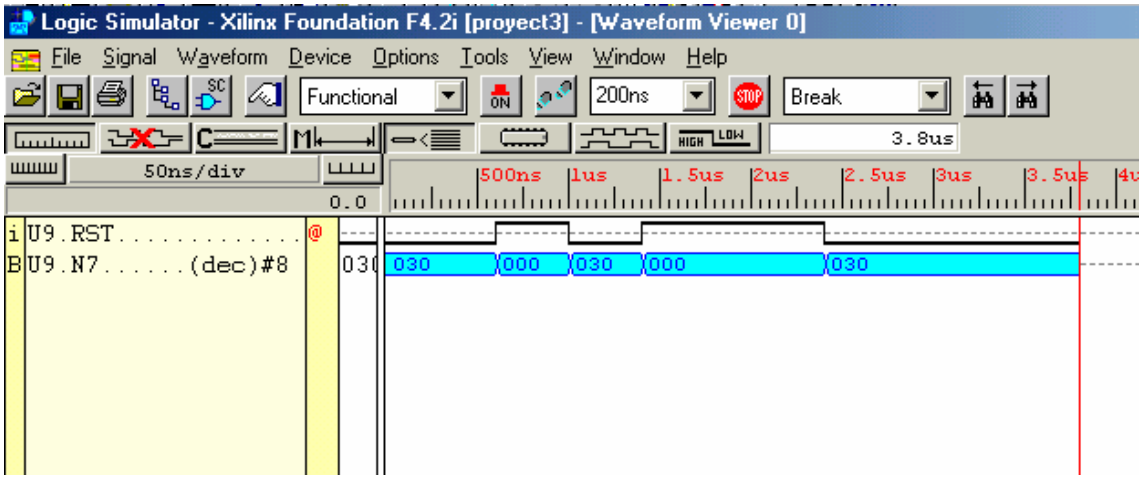
Su funcionamiento puede resumirse en lo citado a continuación:

Este de módulo dispone de la entrada RST de tipo STD_LOGIC y de la salida N de tipo STD_LOGIC_VECTOR (7 downto 0), tal como se observa en la definición de los puertos del modulo en el programa.

El proceso del programa (con rst como señal sensible) realiza lo siguiente)

Si la señal de reset se encuentra activada, a la salida saldrá el número 0 en lugar del 30 por lo que no se podrá realizar la comparación. Si por el contrario se encuentra desactivada, por la salida saldrá el número 30.(necesario para la comparación).

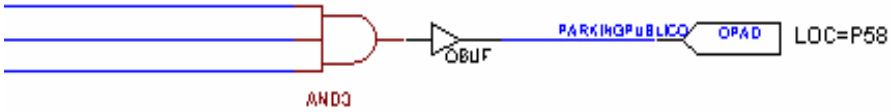
Si simulamos su funcionamiento:



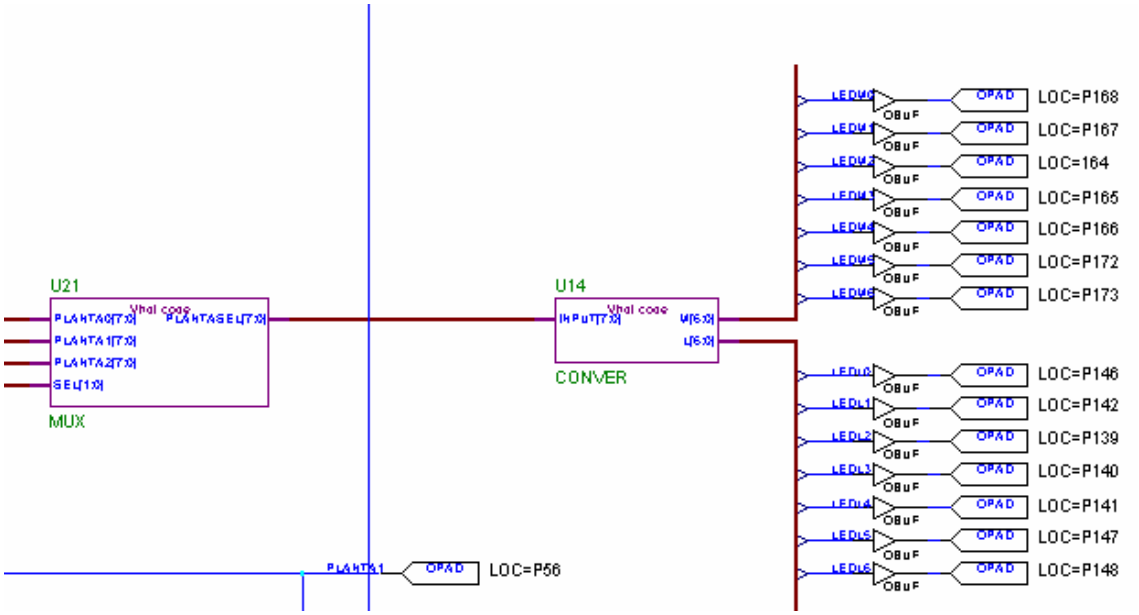
Vemos que opera correctamente

El funcionamiento de los otros dos circuitos iguales al detallado anteriormente es el mismo, uno para el control de la planta 1 y el otro para la planta 2.

Para conseguir la señal que indique que el parking público está completo, se realiza la operación and de la salida de cada uno de los tres circuitos correspondientes a cada planta. Estas salidas son PLANTA0,PLANTA1 y PLANTA2, y como resultado de esa operación and, tenemos la salida PARKING PUBLICO. De manera que cuando las tres entradas de la and estén a '1' por el llenado de cada planta, en la señal PARKING PUBLICO tendremos también un '1' indicando que el parking está completo.

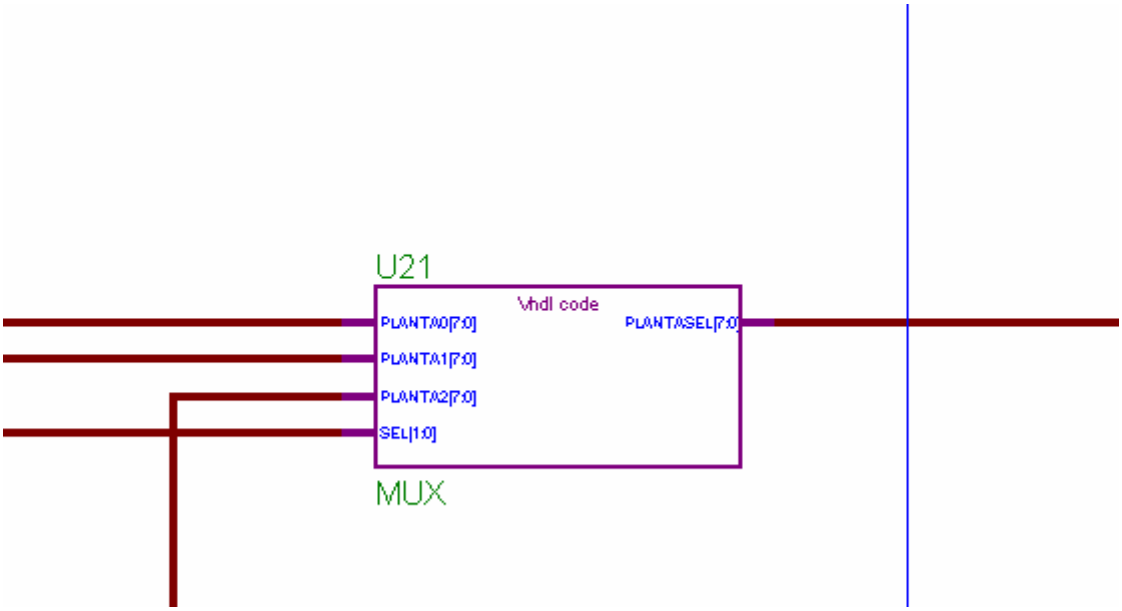


siguiendo con la descripción la parte que queda del circuito.



Esta parte se utiliza para permitir la visualización del número de vehículos de cada planta en dos displays, pudiendo el usuario seleccionar la planta de la que queremos conocer su número de vehículos.

El primer modulo que nos encontramos es el MUX(u21) reflejado a continuación:



Este módulo se utiliza para multiplexar el número de vehículos de la planta que queremos visualizar.

El módulo dispone de cuatro entradas y una salida. Por cada una de las entradas PLANTA0,PLANTA1 Y PLANTA2 entrará el número de vehículos de la planta correspondiente. Luego por la entrada SEL, entrará la señal que seleciona la planta que queremos visualizar, procedente del módulo ADAPMUX U22 que describiré posteriormente. Continuando con la salida PLANTASEL, por esta salida saldrá el número de vehículos de la planta seleccionada, que servirá de entrada al módulo CONVERT U14.

El código VHDL con el que se ha programado este módulo es el siguiente:

```
library IEEE;
use IEEE.std_logic_1164.all;
use IEEE.std_logic_arith.all;
use IEEE.std_logic_unsigned.all;

entity MUX is

   port (
      planta0: in STD_LOGIC_VECTOR (7 downto 0);
      planta1: in STD_LOGIC_VECTOR (7 downto 0);
```

```
        planta2: in STD_LOGIC_VECTOR (7 downto 0);
        sel: in STD_LOGIC_VECTOR (1 downto 0);
        plantasel: out STD_LOGIC_VECTOR (7 downto 0)
    );
end MUX;

architecture MUX_arch of MUX is

begin

process (sel,planta0,planta1,planta2)

begin


        if sel="11" then plantasel<=planta0;

        elsif sel="01" then plantasel<=planta1;

        elsif sel="10" then plantasel<=planta2;

        else plantasel<="00000000";

        end if;

end process;

end MUX_arch;
```

La descripción del proceso es la citada a continuación:

La entradas sensibles del proceso son **sel, planta0, planta1 y planta2,** de manera que si la entrada sel vale ‘11’, por la salida plantasel saldrá el número contenido en la entrada planta0;si vale ’01, saldrá el número contenido en la entrada planta1; si vale ’10’,saldrá el número contenido en la entrada planta2; y si vale ‘00’ saldrá el número” 00000000".

Como he citado anteriormente la entrada de selección procede del módulo ADAPMUX, que aparece en la siguiente figura.

Este módulo dispone de la entrada BDIS y la salida selecmux. Por la entrada bdis entrará los pulsos procedentes del pulsador de selección de planta, y por la salida SELECMUX saldrá la señal correspondiente a la planta seleccionada, que entrará a la entrada SEL del módulo citado anteriormente.

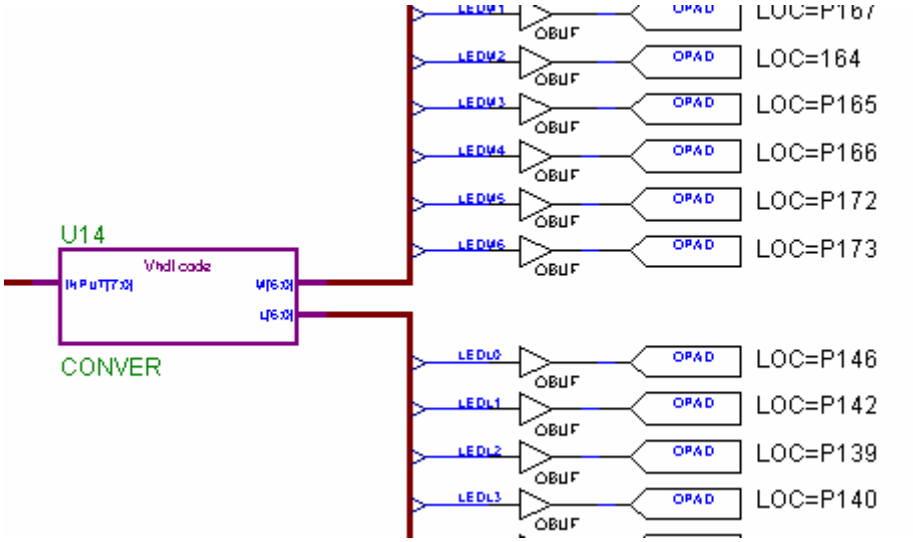
El código con el que se ha programado este módulo es el siguiente:

```
library IEEE;
use IEEE.std_logic_1164.all;
use IEEE.std_logic_arith.all;
use IEEE.std_logic_unsigned.all;

entity adapmux is
   port (
      bdis: in STD_LOGIC;
      selecmux: inout STD_LOGIC_VECTOR (1 downto 0)
   );
end adapmux;

architecture adapmux_arch of adapmux is
begin

process (bdis)

begin

            if bdis'event and bdis='1' then

             selecmux<=selecmux+1;

             selecmux<=selecmux+1;

                 end if;

end process;

end adapmux_arch;
```

El proceso es el siguiente:

Cada vez que en la entrada sensible se produzca un evento y valga ‘1’(es decir cada vez que accionemos el pulsador) la salida selecmux de 2 bits, se incrementará en una unidad, de manara que se iran alternando los cutro estados posibles ‘00’ (ninguna planta seleccionada) ‘01’ (planta1) ‘10’ (planta2) y ‘’11’ (planta0) y así sucesivamente.

Continuando con el desarrollo de la parte del circuito encargada de la visualización del número de vehículos de cada una de las tres primeras plantas, nos encontramos con el módulo CONVERT U14 reflejado en la siguiente figura:



Puesto que el número máximo de vehículos por planta es de treinta, a la hora de su visualización, una vez que supere la cifra de nueve vehículos, necesitaremos mas de un display, uno para las decenas y otro para las unidades.

Este módulo se encarga de sacar el número de vehículos directamente a la entrada del display de las unidades, cuando el número de vehículos sea inferior a 10, y cuando el número de vehículos sea mayor que diez, de llevar la cifra de las decenas del número de vehículos al display de las decenas y la correspondiente a las unidades al display de las unidades, para su correcta visualización.

Dispone de una entrada, la entrada INPUT y dos salidas, M y L. Por INPUT entrará el número de coches a visualizar en los displays; por L saldrá el código en binario necesario de siete bits, para representar las unidades en el display de las unidades, y por M saldrá el código en binario necesario de siete bits, para representar las decenas en el display de las decenas.

El código VHDL con el que se ha programado este módulo es el siguiente:

```
library IEEE;
use IEEE.std_logic_1164.all;
use IEEE.std_logic_arith.all;
use IEEE.std_logic_unsigned.all;

entity CONVER is
   port (
      INPUT: in STD_LOGIC_VECTOR (7 downto 0);
      M: out STD_LOGIC_VECTOR (6 downto 0);
      L: out STD_LOGIC_VECTOR (6 downto 0)
   );
end CONVER;
```

```
architecture CONVER_arch of CONVER is
signal aux_input: std_logic_vector(7 downto 0);
begin
with aux_input(3 downto 0) select
                L<=   "0000110" when "0001", --1
                      "1011011" when "0010", --2
                      "1001111" when "0011", --3
                      "1100110" when "0100", --4
                      "1101101" when "0101", --5
                      "1111101" when "0110", --6
                      "0000111" when "0111", --7
                      "1111111" when "1000", --8
                      "1101111" when "1001", --9

                      "0111111" when others;

        with aux_input(7 downto 4) select
                M<=   "0000110" when "0001",    --1
                      "1011011" when "0010",    --2
                      "1001111" when "0011",    --3
                      "0111111" when others;


aux_input<=(input+6) when (input > 9 and input < 20) else
        (input+12) when (input > 19) else input;



end CONVER_arch;
```

La descripción del programa es la siguiente: Se definen las librerías necesarias, la entidad convert y las correspondientes entradas y salidas.

Se define la arquitectura convert. Luego se define la señal interna **aux_input** (signal aux_input: std_logic_vector(7 downto 0);) y se inicia la arquitectura.

A comtinuación tenemos dos with select,en función de los cuatro primeros y últimos bits de la señal aux_imput. La señal aux_imput dispone de ocho bits, utilizando los cuatro bits más significativos para establecer la condición necesaria, para representación de las decenas a través de las salida M en el display de las decenas, y los cuatro bits menos significativos para la representación de las unidades a través de la salida L en el display de las unidades.

De manera que si imput es menor que diez solo necesitaremos la salida L (para el display de las unidades) y por tanto aux_imput es igual al valor de la entrada imput, y como podemos ver en el primer when select, a la salida L, se le asignará el valor correspondiente en función de los cuatro bits menos significativos (los necesarios para llegar hasta el número nueve) de la seña aux_imput.

Luego si imput es mayor que nueve y menor que veinte, aux_imput valdrá el valor de la entrada imput más seis unidades; esta suma hace que en los cuatro bits más

significativos se represente el número uno, y en el segundo when select a la salida M se le asigna el valor necesario para representación del número uno en el display de las decenas.

Pero si input es mayor que 19, aux_input valdrá el valor de la entrada input más doce; esta suma hace que en los 4 bits mas significativos de aux_input se represente el número dos, si el número de coches es menor que 30, y el tres si el número de coches es mayor que 30, asignándole a la salida M dicho valor en el segundo when select; y en los 4 bits menos significativos se representará la cifra correspondiente de las unidades, cuyo valor se le asignará a la salida L en el primer when select.

Terminada la descripción del módulo CONVERT, me queda únicamente para terminar con el desarrollo de funcionamiento del esquemático visto, explicar la función de los pulsadores BEO de cada planta.

Puesto que las plantas del parking público permanecerán cerradas durante las horas 0:00 a 7:00 de todos los dias excepto festivos, cada dia antes de la apertura del parking se reseteará el sistema con el correspondiente botón de reset, debiendo cargar después al sistema el número de vehículos que podrían haber quedado estacionados en cada una de las plantas, para lo cual utilizaremos los pulsadores BE0,BE1,BE2 BS0,BS1,BS2 (ya que cada uno actúa como entrada al contador de entrada o salida de vehícilos de su respectiva planta, como puede observarse en el esquemático) en función de la planta correspondiente.

Detalle de los pines utilizados de la placa diligent2:

LOC=P3: ---------- SE01

LOC=P4: ---------- SE02

LOC=P5: ---------- SS01

LOC=P6: ---------- SS02

LOC=P7: ---------- SE11

LOC=P8: ---------- SE12

LOC=P9: ---------- SS11

LOC=P10: ----------SS12

LOC=P15: ---------- SE21

LOC=P16: ----------SE22

LOC=P17: ---------- SS21

LOC=P18: ---------- SS22

LOC=P20: ---------- BE0

LOC=P21: ---------- BE1

LOC=P22: ---------- BE2

LOC=P23: ---------- SE01

LOC=P14: ---------- RESET

LOC=P46: ---------- BS0

LOC=P47: ---------- BS1

LOC=P48: ---------- BS2

LOC=P49: ---------- SELECT

LOC=P55: ---------- PLANTA0

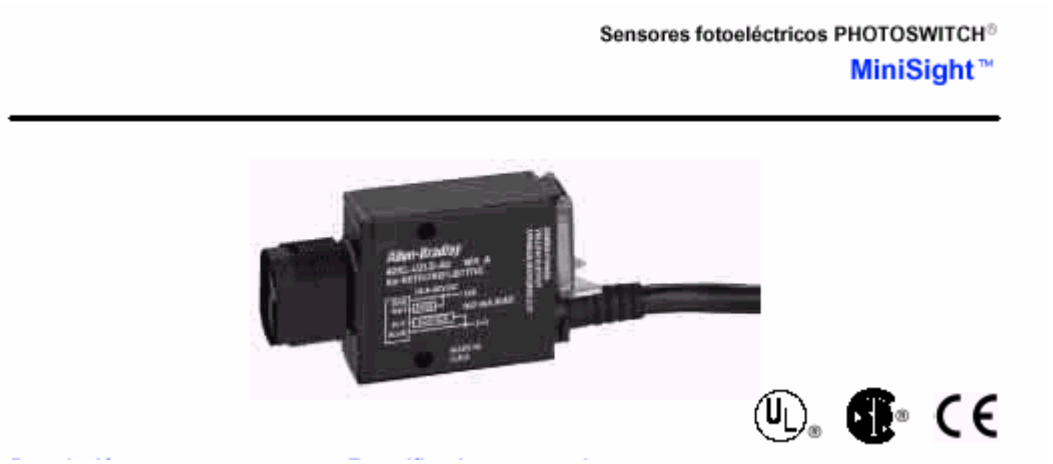
LOC=P56 ---------- PLANTA1

LOC=P57: ---------- PLANTA2

LOC=P58 ---------- PARKING PUBLICO

4.-SENSORES

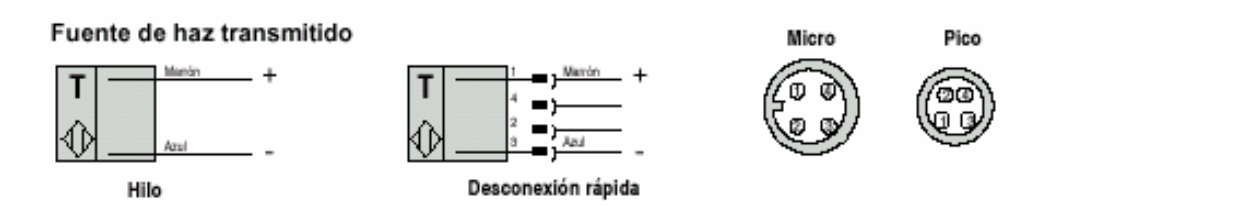
Este sistema dispondrá de doce sensores fotoeléctricos modelo “Mini Sight”, de la marca Allen-Bradley, para el funcionamiento en modo de haz transmitido (ya que el conexionado es mas sencillo), alimentación de 10.8-30Vcc y salida NPN de 100 ma, por lo que no tendremos problemas con caídas de tensión por el conexionado. Este tipo de sensor viene detallado en profundidad en el apartado X sensores, del Apéndice X “características de los productos y hojas de fabricantes” y se refleja en la siguiente figura:



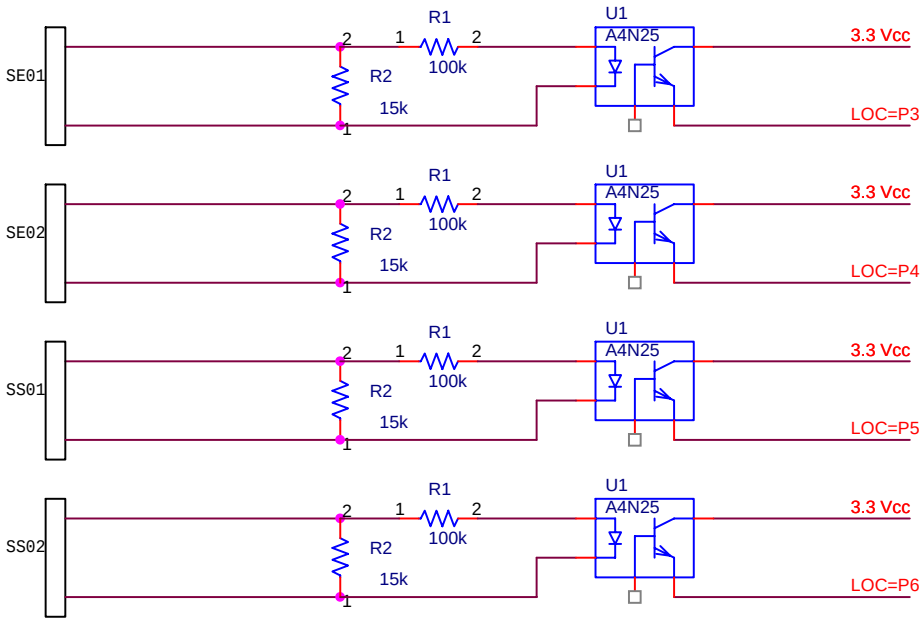
En cuanto a su ubicación, se instalarán en las puertas de acceso a planta de cada ascensor de vehículos, como se puede observar en el correspondiente **planoxxxxxxxxxxxxxxxx .**

4.1.-CONEXIÓN DE LOS SENSORES A LA PLACA DIGILENT

El transporte de la señal del sensor se realizará mediante conectores Microcc de 4 pines, de la siguiente manera:



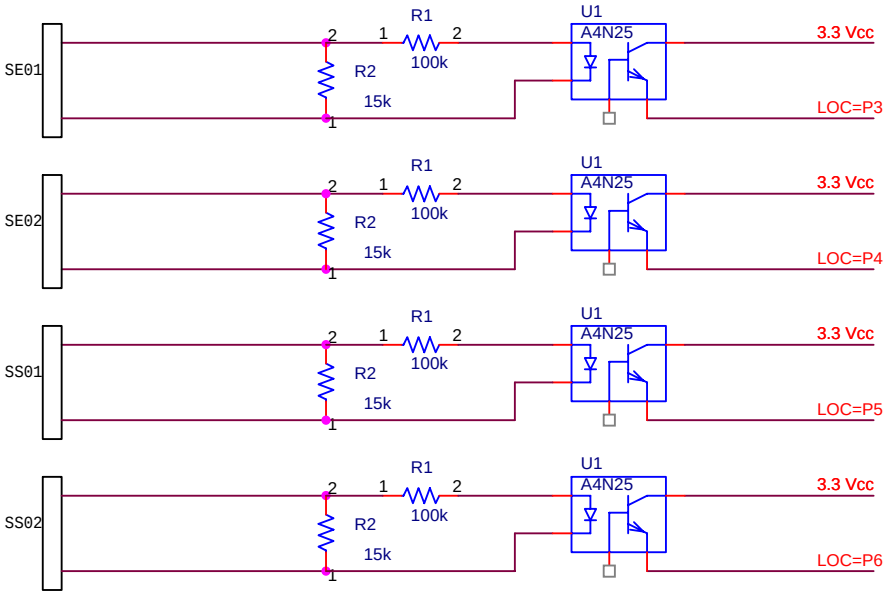
Con respecto a la llegada de la señal a la placa diligent, no procederemos con la conexión directa, sino con el fin de proteger la citada placa ante el caso de sobreintensidades por errores en la conexión o deterioro de la instalación entre otras causas, se realizará con optoacopladores 4N25 como intermediarios, de la manera detallada en la siguiente figura:



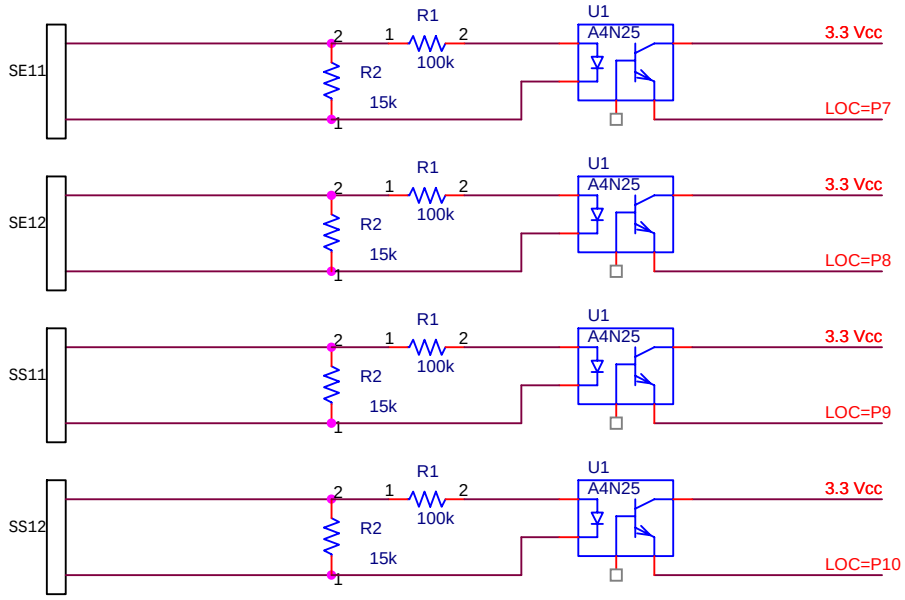
El objetivo de las resistencias R1 y R2 es el de divisor de intensidad, ya que la intensidad óptima de funcionamiento del 4N25 es de 10 ma y la máxima de 30 ma, y la que proporciona cada sensor es de 100 ma. Con este divisor la intensidad de entrada al optoacoplador será de 13 ma aproximadamente.

5.-CIRCUITOS DE ENTRADA Y SALIDA A LA PLACA DIGILENT

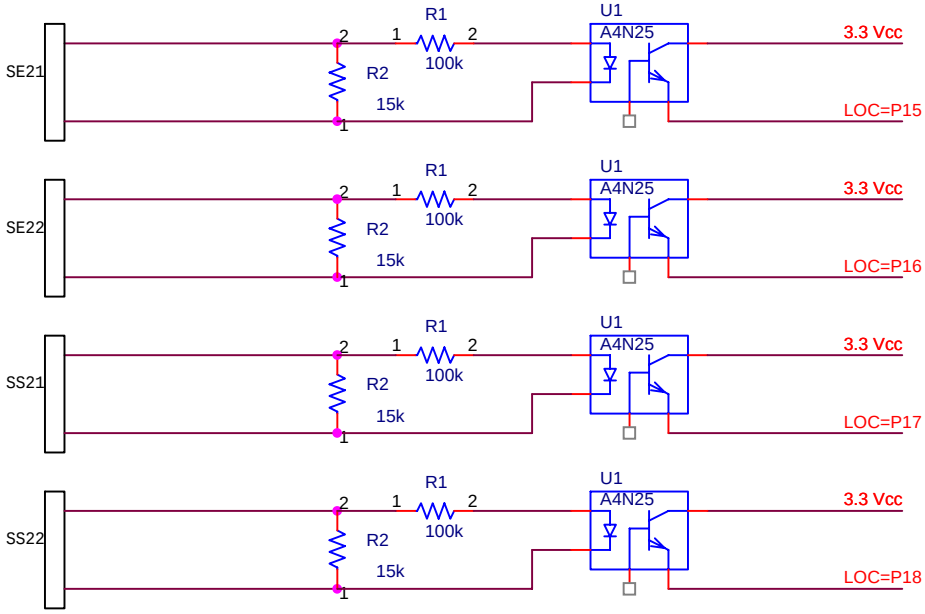
Circuito sensores planta0



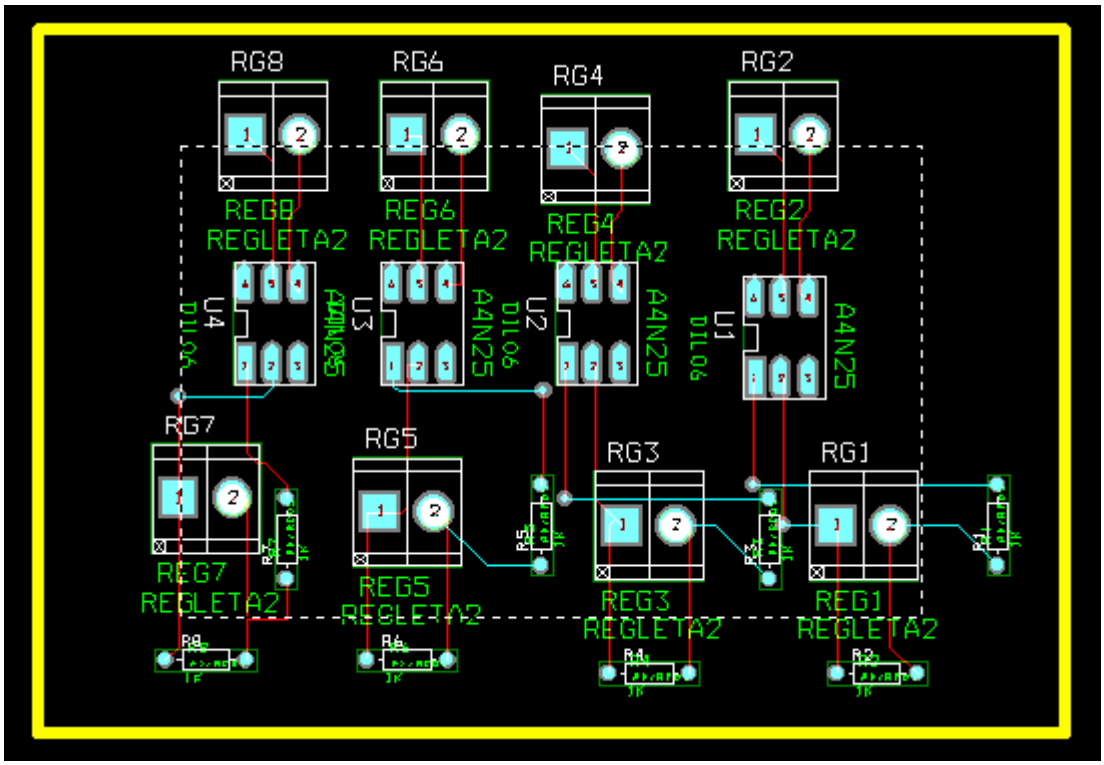
Circuito sensores planta1



Circuito sensores planta2



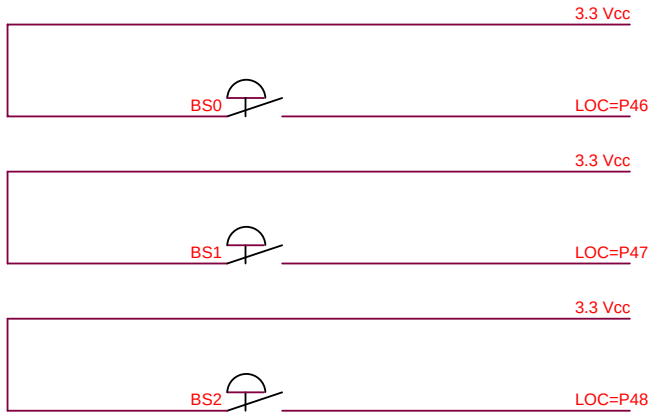
La placa a implementar para cada uno los circuitos de cada planta es le siguiente:



Pulsadores para cargar el número de coches

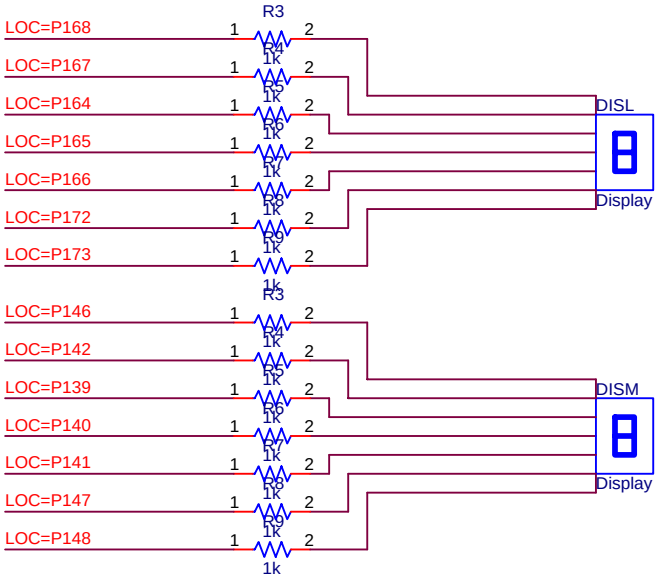
Pulsadores de entrada.

Pulsadores de salida.

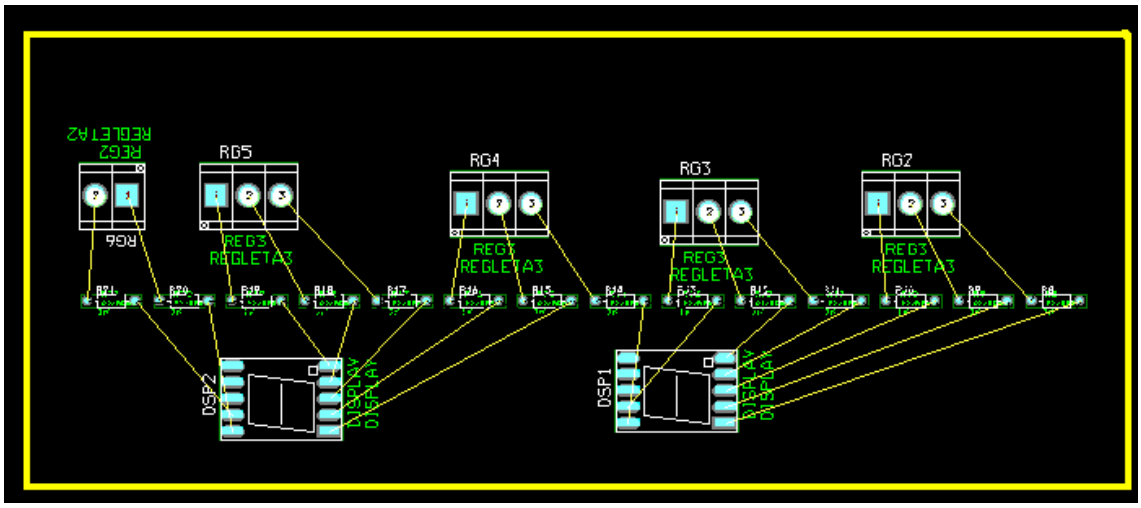


Pulsador de reset y de selección de planta a visualizar

Circuito displays



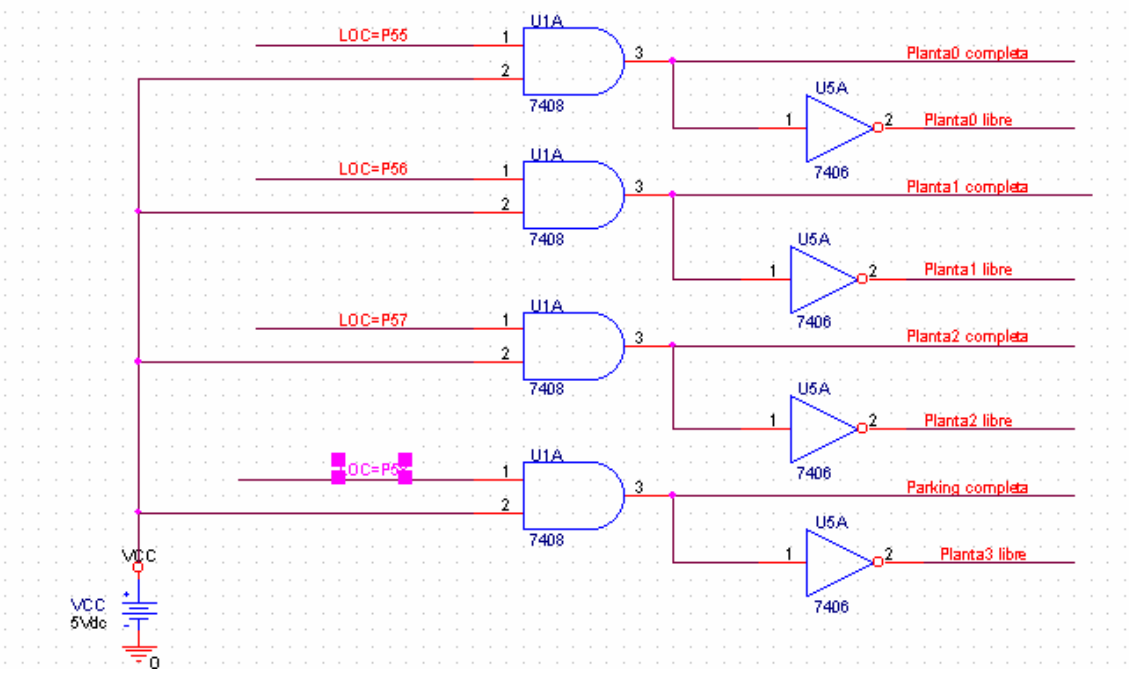
La placa a implementar para el circuito anterior es la siguiente:



Circuito de adaptación para las salidas de la placa diligent que entren al módulo 6E-6S.

Puesto que las entradas digitales al módulo de control 6E-6S han de ser d 5v, y las salidas de nuestra placa digilent son de 3.3v, puede que dicho voltaje aunque supere el mínimo valor de voltaje de 1 lógico, no sea suficiente para activar las entradas del módulo 6E-6S.

Para solucionar este problema realizarems el siguiente circuito:



Dado que los 3.3v pueden se considerados son considerados como un 1 lógico, si a cada salida le realizamos una operación AND con 5Vdc, a la salida de cada puerta AND tendremos los 5v necesarios para excitar las entradas del módulo citado anteriormente.

Como podemos observar de cada salida de cada puerta and sale otra salida pero negada mediante un inversor, de manera que tenemos dos tipos de salidas: las negadas, que servirán de entrada al primer módulo 6E-6S para el encendido de las luces de señalización que nos indican que una plaza o el parking está libre, y las no negadas, que servirán de entrada al segundo módulo 6E-6S para el encendido de las luces de señalización que nos indican que una plaza o el parking está completo.

6.-FUNCIÓN DEL MÓDULO 6E-6S.

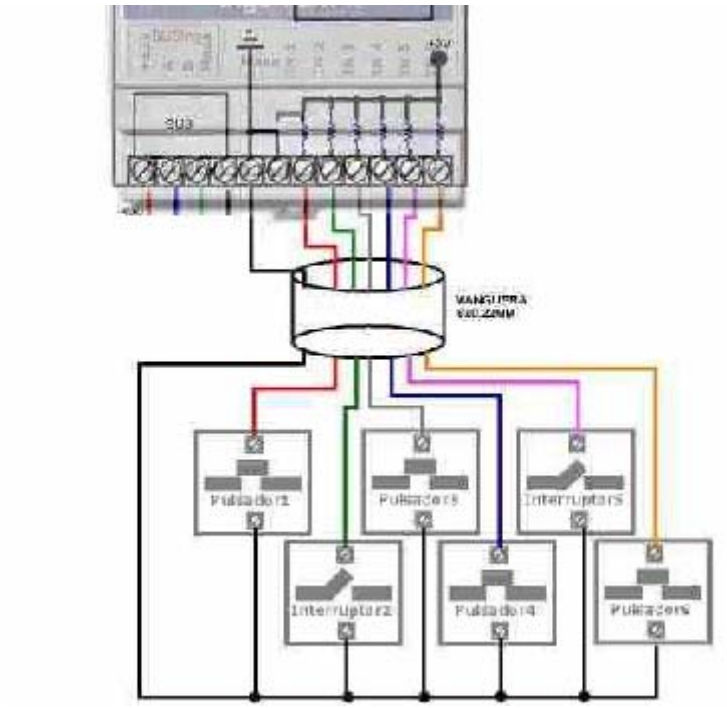
El objetivo de la activación de las salidas de la placa digilent: planta0, planta1, planta2, y parking público, es que dichas salidas accionen el sistema necesario que permita encender las luces indicadoras del estado libre o completo de cada planta o del parking en general.

Dicha función se realizará a través de dos módulos de control 6E-6S, uno para el encendido de las luces de estado completo y otro para el encendido de las luces de estado libre. Dicho módulo dispone 6 Entradas Digitales de Baja Tensión (5V) y 6 Salidas Digitales 6 Amperios referidas a Fase, y se puede obsevar en la siguiente figura:

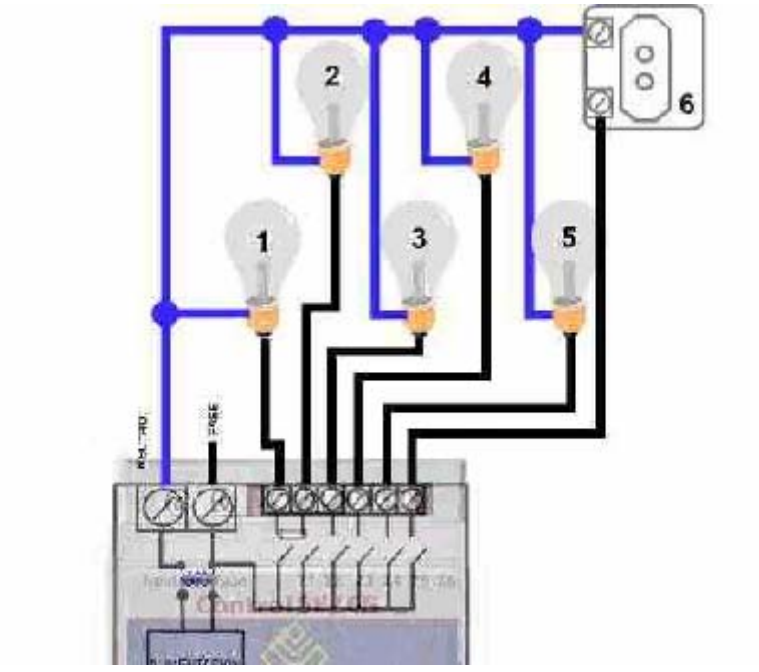
El CONTROL6E6S es un actuador provisto de 6 salidas a relé referidas a la Fase de alimentación del equipo y 6 entradas de baja tensión referidas a la masa del BUS.
Desde el sistema de desarrollo es posible asignar cadenas de 15 caracteres para identificar a cada una de las salidas o las entradas. También es posible asignar el modo de funcionamiento de cada una de las entradas (Pulsador o Interruptor), y dos eventos de BUS para cada una de las entradas (Un evento de activación y uno de desactivación), permitiendo de esta manera actuar sobre cualquier elemento de la instalación desde las entradas del equipo
Entradas:
· Entradas de baja tensión 5V, corriente mínima de activación 5mA.
· Activas cuando están conectadas a Masa.
· Distancia de cableado máxima a interruptor o pulsador 30 metros.
Salidas:
· Desactivadas: Relé abierto.
· Activadas: Relé cerrado.
· Corriente máxima de conmutación 6 Amperios.

En cuanto a la conexión delas entradas digitales de entrada al módulo, y las fases para el encendido de las luces de señalización, viene reflejada la forma de conexión en las siguientes figuras:

Entradas digitales:



Salidas en fase:



7.-DISPOSITIVO S.A.I.

Aunque el parque disponga de un grupo electrógeno de emergencia para garantizar el suministro eléctrico a todo el edificio ante el caso de corte eléctrico, para garantizar el suministro y sobre todo la calidad de la alimentación al dispositivo de control de la ocupación del parking y diferentes componentes informático ,se dispondrá de un dispositivo S.A.I , específicamente detallado a continuación.

El modelo autilizar es el de la Serie ONDA Senoidal 600 VA - 3000 VA de la marca ENDATA, reflejado en las siguiente figura:

Como principales características podemos destacar las siguientes:

Proporciona una tensión de salida segura y protege su informática de los cortes y variaciones de tensión.

Sistema de alimentación ininterrumpida (SAI) que protege su informática de los cortes de variaciones de tensión de la red eléctrica.

By-Pass automático.

Dispositivo de protección electrónica contra sobrecargas o cortocircuitos.

Dirección de posición de fase.

Control de microprocesador.

Paquete de software que funciona con los sistemas operativos MS-DOS, Windows NT, etc.

Visualiza en su ordenador los parámetros de funcionamiento del equipo.

Realiza el arranque automático mediante su sistema informático.

Ejecuta el paro automático SHUT DOWN.

Salvado de datos, shutdown del sistema y parada de la UPS.

Envío de mensajes e-mail.

Comunicación SNMP y USB.

TEST: Estado de la batería y eficacia.
Condiciones de tensión

Protección total, asegura un elevado rango de fluctuación de la tensión sin descagar la energía de la batería.

Monitorización del funcionamiento – Función de arranque en frio.

Control de la batería débil / agotada.

Numerosos estados de alarma.

El puerto interfase serie DB9 permite al UPS estar conectado al sistema informático para su monitorización y diálogo mediante señales RS232 (estado de la carga, UPS, batería, programa de apagado y encendido, salvado automático de ficheros y Shut-Down del sistema informático).