# Índice de contenido

# *TP Final - 7542 - 2do cuat. 2009*

# *Circuitos Lógicos Distribuidos*

## Objetivos

El presente trabajo tiene como objetivo poner en práctica los conocimientos adquiridos hasta el momento, articulándolos en un trabajo grupal que permita el diseño y la implementación de aplicaciones con un aceptable estándar de calidad y eficiencia y en los plazos planteados. Las aplicaciones elegidas para este trabajo combinan, entre otros, conceptos de comunicaciones, programación concurrente e interfaces gráficas.

Cabe destacar que la aprobación del mismo requiere la construcción de las aplicaciones, las cuales deben ser efectivas (cumplir todos los objetivos funcionales) y razonablemente eficientes. Además, deberá formar parte de la entrega toda la documentación técnica correspondiente, los manuales de usuario y todos los documentos, productos y subproductos (cronogramas, diagramas, evidencias de pruebas, etcétera) confeccionados durante la etapa de desarrollo que permitan demostrar, junto a la explicación verbal de los alumnos, que la elaboración fue original.

## Introducción

Una Empresa de Ingeniería Electrónica ha decidido fabricar chips de alta integración destinados a diversos usos de origen civil y militar. Para el proceso de desarrollo, la Empresa requiere una aplicación que asista en el diseño de circuitos y la simulación de funcionamiento. Debido a las cambiantes condiciones mundiales de mercado y al diverso origen geográfico de los pedidos se ha decidido enfrentar el desafío del diseño y simulación en forma de Grupos de Ingeniería Distribuidos. Cada grupo estará a cargo de diseñar uno o más circuitos. Cada uno será confeccionado y puesto a disposición del resto de los Grupos para su uso en otros diseños y para la simulación combinada.

## Nociones previas

Las puertas lógicas básicas son: AND, OR, NOT, XOR de un bit. A continuación se muestran sus representaciones gráficas típicas y sus definiciones a través de una tablas de entradas-salidas.

# La Solución

Para la implementación de los requerimientos funcionales se pide la construcción de dos aplicaciones:

- Una para el diseño y visualización de la simulación;
- Otra para la publicación (exportación de características de diseño y datos de simulación).

# La aplicación de diseño y simulación

Se trata de una aplicación gráfica que despliegue un tablero blanco sobre el cual se combinarán ENTRADAS (puntos de entrada a nuestro circuito), SALIDAS (Salidas de nuestro circuito) PUERTAS básicas, PISTAS de conexión y CIRCUITOS preconfeccionados (por éste u otros Grupos de Ingeniería) para lograr un nuevo circuito. Esta aplicación deberá poseer las siguientes características:

- La interfaz gráfica debe permitir la edición de múltiples circuitos simultáneamente.
- Tablero de trabajo cuadriculado para facilitar el posicionamiento de componentes.
- Barra de herramientas (para ENTRADAS, SALIDAS, PUERTAS y CIRCUITOS, PISTAS)
- Selección de uno o más componentes simultáneamente.
- Posibilidad de borrar, rotar (en pasos de 90 grados), invertir (vertical/horizontalmente) o desplazar ENTRADAS, SALIDAS, PUERTAS, CIRCUITOS, PISTAS individuales.
- Posibilidad de borrar o desplazar selecciones.
- Grabación y carga de circuitos utilizando archivos XML.
- Posibilidad de visualización (no edición) de circuitos publicados en otros servidores (usando otro documento MDI)
- Diálogos para fijar los atributos de cada componente, a saber:

  ENTRADAS: Nombre

  SALIDAS: Nombre

  PUERTAS: Tiempo de respuesta

  PISTAS: ----

  CIRCUITOS: IP del servidor que lo contiene

  Nombre del circuito

Cabe destacar que estas puertas lógicas son de muy rápida respuesta. Sin embargo, por insignificante que parezca, las puertas requieren un tiempo de conmutación o transición. Este tiempo siempre es considerado en los diseños, a los efectos de lograr circuitos robustos.

Finalmente, esta aplicación deberá ofrecer una pantalla que genere una tabla con la combinatoria de entradas y sus respectivas salidas y tiempos de transición. Para esta tarea, la aplicación podrá requerir conexiones a otros servidores a fin de obtener las salidas y tiempos de transición de los circuitos de menor nivel utilizados.

**La tabla y los circuitos se podrán imprimir en modo gráfico.**

# El servidor de publicación

Los circuitos almacenados podrán **publicarse** al resto de los Grupos de Ingeniería utilizando arquitectura SOAP. Esta publicación implica 2 servicios:

- Ofrecer características de diseño para quienes quieran utilizar el circuito como componente de sus propios diseños:
  - Tamaño (el circuito será visualizado por el usuario como una "caja negra")
  - Posición de cada ENTRADA y cada SALIDA
  - Detalle de diseño (sólo lectura)

- Simular el funcionamiento en tiempo real, devolviendo el estado de todas las SALIDAS (y sus respectivos tiempos de transición) a partir de las ENTRADAS suministradas por el cliente remoto. Es factible que el servidor tenga que conectarse a otros servidores de publicación para obtener los resultados de los circuitos incluidos en aquel que se está simulando.

La aplicación de simulación debe ser capaz de ofrecer estos servicios a uno o más clientes, para uno o más circuitos, y todo en forma simultánea.

# Modalidad de trabajo

## Grupos de trabajo

El grupo de trabajo del TP debe estar integrado por tres (3) alumnos regulares de la cátedra. Sólo se contemplarán grupos de distinta cantidad de integrantes por razones de fuerza mayor y bajo expresa autorización del cuerpo docente.

## Bibliotecas / Librerías

Toda funcionalidad que no figure en la siguiente lista deberá ser implementada por los alumnos:

- ISO C++
- STL

- Bibliotecas específicas del SO para el manejo de threads y *sockets*. Los sockets deben ser BSD compatibles (es decir, bloqueantes).
- Para la interfaz gráfica y para el instalador se puede usar GTK o gtkmm.

Si el grupo desea usar bibliotecas no incluidas en la lista, deberá consultar con los docentes (se recomienda solicitar la autorización por escrito).

## Entrega

El software deberá constar de instalador. El código fuente deberá contener comentarios claros y estar impreso en un apéndice del informe.

Detalle del informe:

**Definición del proyecto**

- Enunciado
- Cronograma
- División de tareas
- Reporte de avance
- Mejoras
- Conclusión

**Documentación Técnica**

- Análisis (pantallas, descripciones)
- Diseño (notas técnicas, estructuras de datos (UML), interfaces, políticas adoptadas)

**Documentación del usuario**

- Forma de instalación
- Forma de uso (pantallas, explicación )

# Manual de usuario

## Simulador de circuitos lógicos distribuidos

## *Introducción*

El simulador de Circuitos lógicos distribuidos es una herramienta, que tiene como principal objetivo la asistencia en el diseño de circuitos lógicos, la simulación del funcionamiento del mismo y la posibilidad de publicarlos, dejándolos disponibles para el uso en otros diseños y logrando una simulación combinada.

La aplicación permitirá la descarga de cualquier circuito publicado, permitiendo la incorporación del mismo en el diseño propio y otorgando la posibilidad de previsualización y simulación en tiempo real, devolviendo el estado de todas sus salidas y tiempos de simulación.

## *Requerimientos básicos*

Librerías necesarias:

- Xerces C++
- GTKMM

## *Instalación*

Cliente:

Extraer el archivo cliente.tar.gz utilizando el comando “tar -xvvf cliente.tar.gz”. Luego ejecutar el comando “make” y se genera el ejecutable “cliente”.

Para ejecutar ingrese “./cliente”.

Servidor:

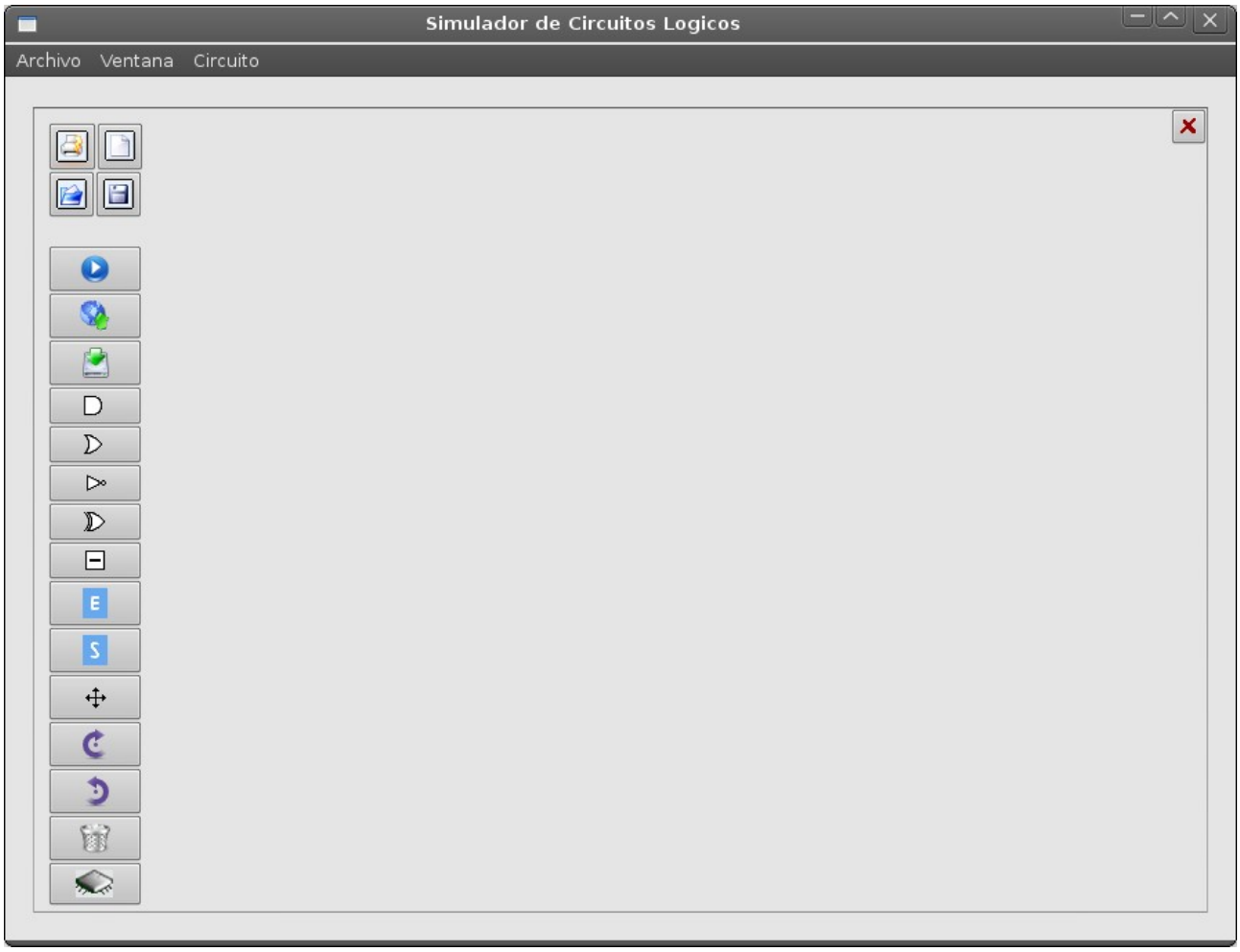
Extraer el archivo servidor.tar.gz utilizando el comando “tar -xvvf servidor.tar.gz”. Luego ejecutar el comando “make” y se genera el ejecutable “servidor”.

Para ejecutar ingrese “./servidor *param1*” donde *param1* es el numero de puerto del servidor.

## *Aplicación*

Al abrir la aplicación nos encontraremos con la ventana principal del programa, la cual se divide entres partes principales:

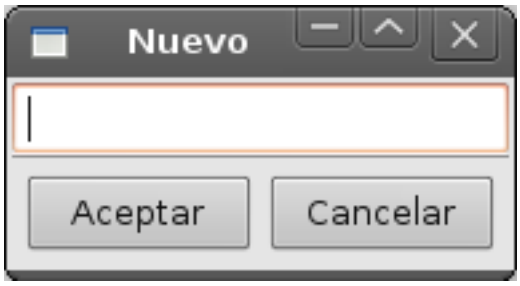
- La barra de control, en la cual se ubica el menú.
- La barra de herramientas con opciones de archivo.
- La barra de herramientas con opciones de diseño.



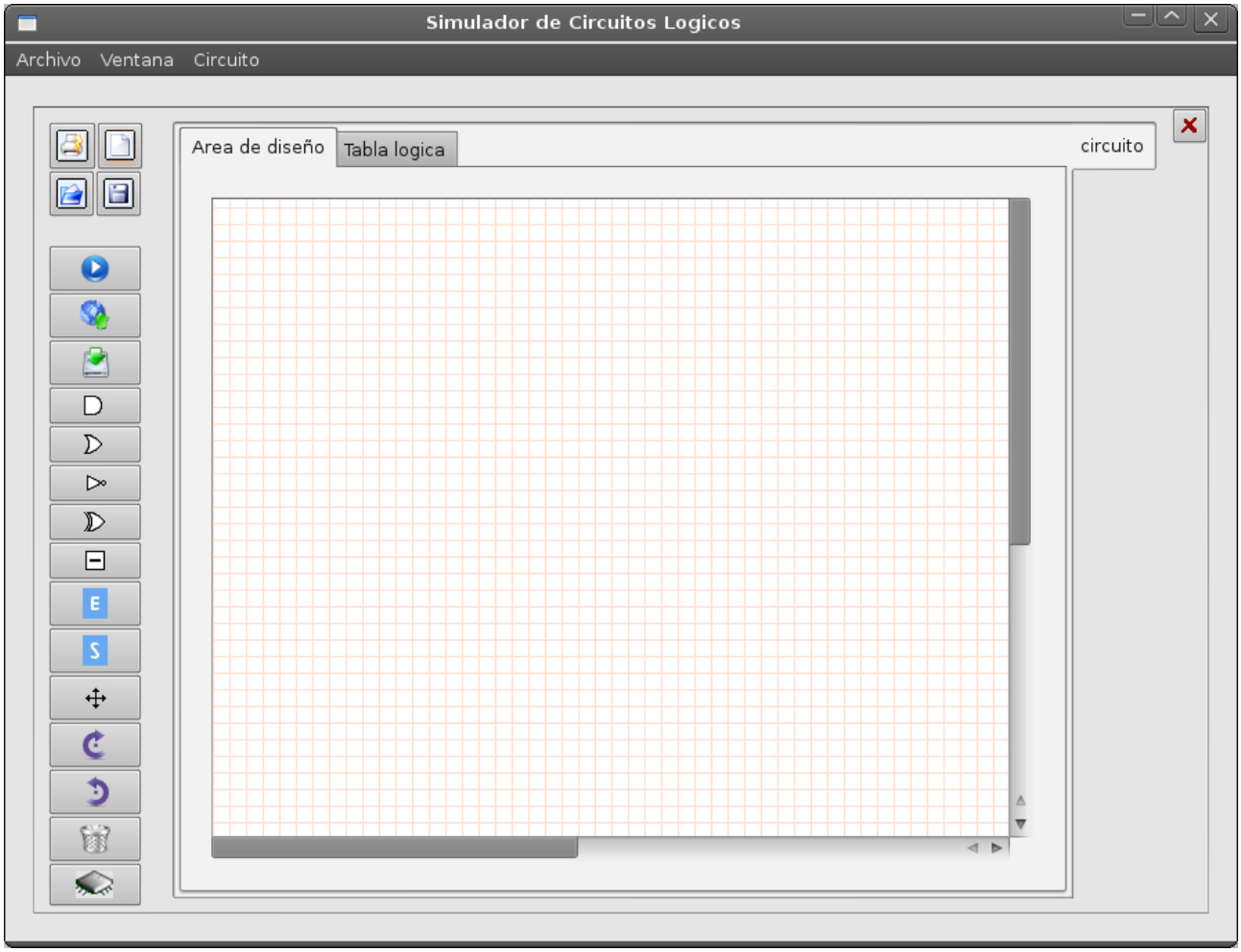

Primero comenzaremos describiendo las herramientas que se encuentran en la barra de opciones de archivo.

## - Opciones de Archivo

**Nuevo circuito:** esta herramienta se encarga de la opción que nos permite comenzar con el diseño de un nuevo circuito, al presionar este botón aparecerá la siguiente ventana:




En esta ventana tendremos la opción de ingresar el nombre con el cual identificaremos al circuito que nos proponemos a diseñar, y será el nombre con el que se guardará el mismo en caso de requerirlo. Luego de presionar aceptar, aparecerá el área de diseño donde podremos comenzar a construir nuestro circuito.




**Imprimir circuito:** esta herramienta es la encargada de la impresión del circuito actual. Al seleccionarla se abrirá las opciones de impresión.

**Abrir circuito:** esta herramienta es la encargada de recuperar un circuito que hayamos guardado con anterioridad. Al presionar esta opción nos aparecerá la siguiente ventana:


Esta es la ventana donde elegiremos el circuito a recuperar. Los mismos se deben encontrar en la carpeta saves, que es en la cual se guardan al ejecutar la opción de guardado.

**Guardar circuito:** esta es la opción que nos permite guardar el circuito que hemos diseñado. Este circuito se guardará por default en la carpeta saves y con el nombre que elegimos al abrir un nuevo circuito.

**Cerrar pestaña:** Por ultimo incluiremos la descripción de la herramienta para cerrar la pestaña actual, a pesar de no encontrarse en esta barra de herramientas.

La función de esta herramienta es la de cerrar la pestaña de diseño en uso.

## - Opciones de Diseño

**Incorporación de compuertas:** Las herramientas para la inclusión de nuevas compuertas son bastante intuitiva. Seleccionando la compuerta que queremos incorporar, simplemente tendremos que presionar con el click izquierdo del mouse el sector del área de diseño donde queremos incluir la nueva compuerta, siempre y cuando dicha posición no este ocupada con otro componente se la incluirá.

La posibilidad de incluir el componente seleccionando, estará activa hasta la elección de una nueva herramienta de la aplicación.

**Incorporación de pistas:** esta herramienta sirve para la inclusión de pistas al área de diseño. Su utilización es muy similar a la incorporación de una compuerta, con la salvedad que una pista tiene la opción de ocupar el mismo sector del área que otro componente pista. Esta salvedad ocurre siempre y cuando, las mismas se crucen perpendicularmente, y por lo menos una, corte a la otra por su centro, o sea por la mitad del otro componente.

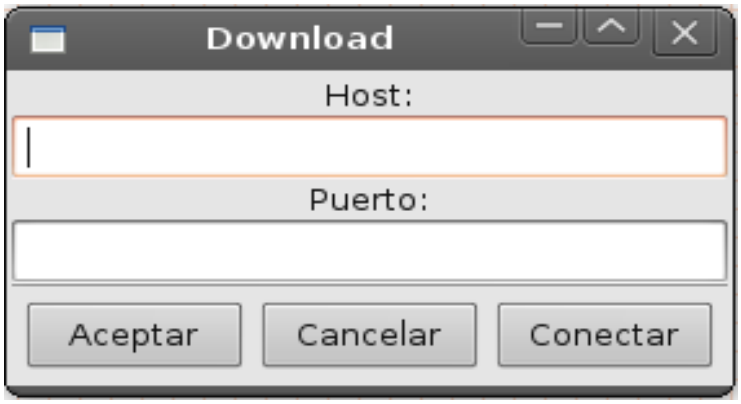
**Incorporación de Entradas y Salidas:** esta herramienta sigue el paso de las anteriores para la incorporación de estos dos componente, que al igual que las compuertas solo podrán ocupar un lugar del área de diseño que no se haya ocupado con anterioridad. Una particularidad que presentan estos componentes es la de la identificación de los mismos con un nombre, el cual será posible incluir en la ventana que mostraremos a continuación, la cual aparecerá, luego de presionar el botón correspondiente a dicha herramienta.



Otra propiedad de esta opción, es que luego de haber apretado el botón para la incorporación de una entrada o salida, se podrá agregar un solo componente, que será identificado con el nombre escrito en la ventana mostrada anteriormente, y para la incorporación de uno nuevo, se deberá seleccionar la opción nuevamente.

**Manejo de componentes:** para el manejo de los componentes se tendrán las opciones de mover, rotar , y borrar. Las cuales después de seleccionar la herramienta que se desea utilizar, presionando encima del componente al cual queremos que afecte dicha opción se llevará a cabo.

**Incorporar un circuito publicado:** la aplicación tiene la opción de incorporar a nuestro diseño un circuito que haya sido publicado en el servidor por otro grupo de trabajo. Al seleccionar esta herramienta aparecerá en pantalla la siguiente ventana:

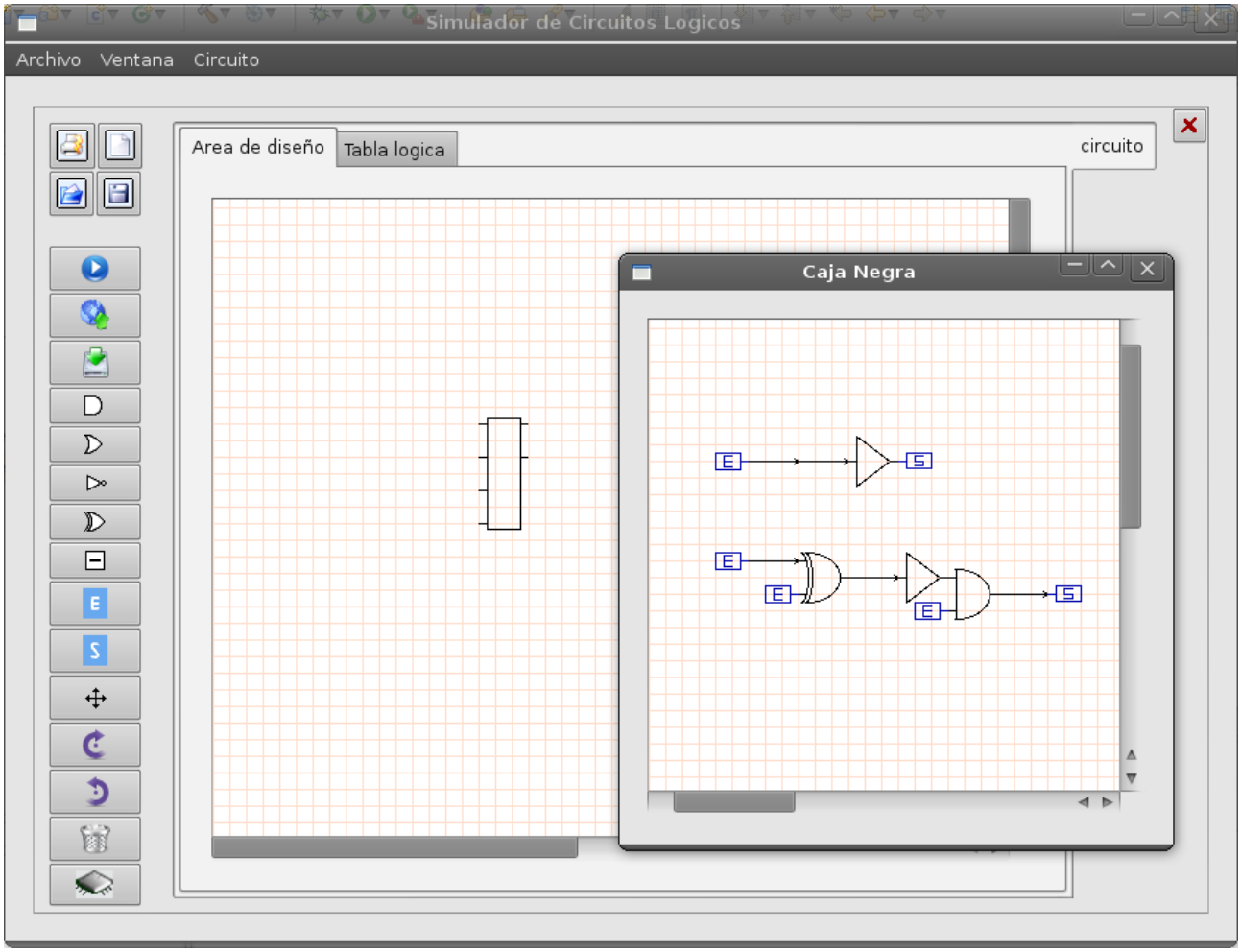



En esta ventana, tendremos que incluir la dirección y el puerto del servidor del cual queremos obtener los circuitos publicados para una posible descarga. Luego tendremos que presionar el botón de conectar para lograr la comunicación con el servidor.


Luego de realizar la conexión, en este ejemplo usamos la dirección 127.0.0.1 y el puerto 1234, aparecerán los circuitos que el servidor posee. En este caso el servidor tiene disponible el circuito llamado prueba1, y en el caso de querer utilizarlo, deberemos seleccionarlo y apretar el botón aceptar.

A partir de este momento, podremos incorporar el circuito descargado y representado por una caja negra, a nuestro diseño, presionando el botón izquierdo del mouse en algún sector disponible del área de diseño.

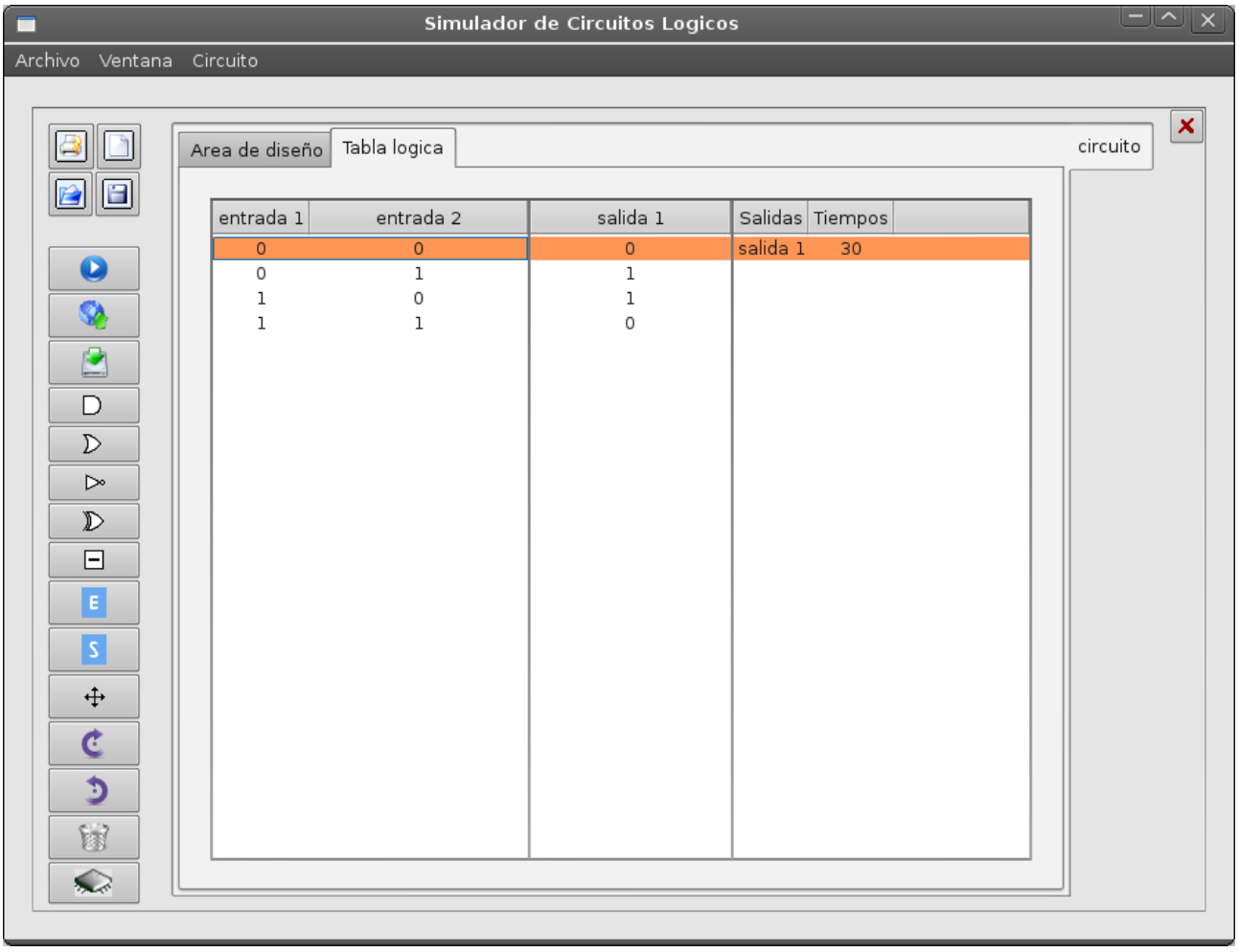
**Pre-view:** luego de a ver incorporado alguna caja negra a nuestro diseño, vamos a tener la posibilidad de visualizar su composición , pulsando la herramienta de previsualización y presionando el botón izquierdo del ratón en el componente deseado. A continuación mostraremos un ejemplo de esta acción:



**Publicar un circuito:** otra herramienta que posee el programa es la de publicar un circuito para que otro grupo de trabajo pueda utilizarlo. Al seleccionar esta opción aparecerá la siguiente pantalla:


En esta ventana, tendremos que incluir la dirección y el puerto del servidor en el cual queremos publicar nuestro circuito, tendremos que seleccionar el circuito al cual queremos realizarle esta acción. Solamente se podrán publicar circuitos que hayan sido guardados previamente.

**Simulación:** Esta herramienta ejecuta la simulación del circuito. Un circuito se puede simular en el caso de que se encuentre completo y no haya componentes desconectados. Al realizar la simulación, se completa la tabla lógica, a la cual se puede acceder seleccionando la pestaña de la tabla correspondiente al circuito que simulamos. A continuación se muestra un ejemplo de la tabla conseguida a partir la de simulación de un circuito.



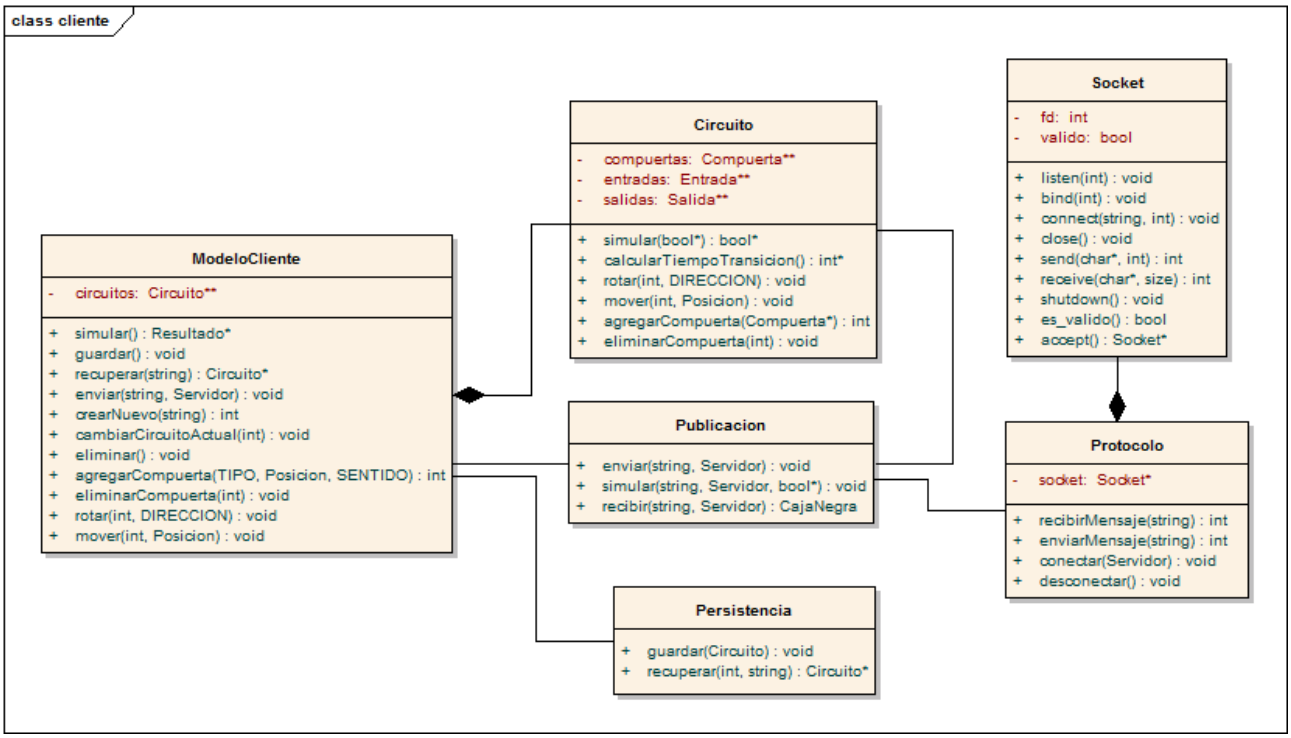

## - Barra de control (menú)

En la barra de control se podrán encontrar todos las herramienta explicadas anteriormente y con las mismas condiciones de uso, por lo cual no nos detendremos en la explicación de la misma, pero a continuación se muestra la ubicación de estas herramientas en la aplicación:

# Manual Técnico

## Simulador de circuitos lógicos distribuidos

## *ModeloCliente*




## *Circuito*

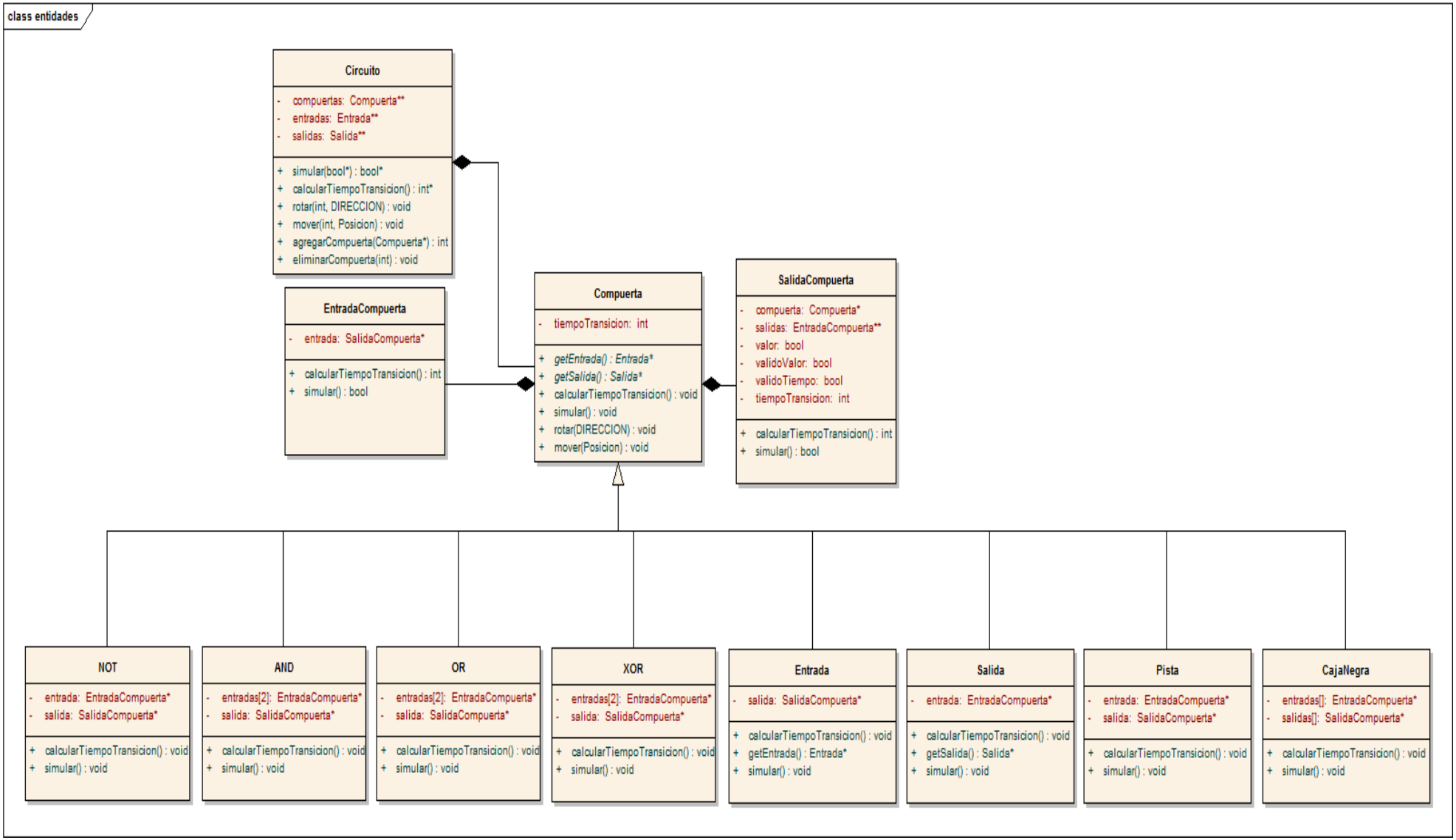
El circuito esta formado por las clase Circuito, Compuerta, EntradaCompuerta y SalidaCompuerta.

La clase Compuerta es abstracta y heredan de ella las compuertas especificas (NOT,AND,OR,XOR,PISTA,ENTRADA,SALIDA,CAJA NEGRA). Cada compuerta esta formada por EntradaCompuerta y SalidaCompuerta. Por ejemplo la AND tiene dos EntradaCompuerta y una SalidaCompuerta y la NOT tiene una de cada uno.

Las clases EntradaCompuerta y SalidaCompuerta se utilizan para poder llevar a cabo la conexión entre las compuertas. Una EntradaCompuerta se puede conectar a una unica SalidaCompuerta, pero una SalidaCompuerta puede conectarse a varias EntradaCompuerta.

Las compuertas tienen un id, para poder ser identificadas desde el módulo vista/controlador, una posición y un sentido. Con estos últimos dos atributos se pueden llevar a cabo las conexiones entre compuertas y además son utilizados por la vista para dibujar a las compuertas.

Por último la clase Circuito contiene una lista de Compuertas, y tiene el comportamiento de administrarlas.


class entidades
Circuito
- compuertas: Compuerta**
- entradas: Entrada**
- salidas: Salida**
+ simular(bool*) : bool*
+ calcularTiempoTransicion() : int*
+ rotar(int, DIRECCION) : void
+ mover(int, Posicion) : void
+ agregarCompuerta(Compuerta*) : int
+ eliminarCompuerta(int) : void
EntradaCompuerta
- entrada: SalidaCompuerta*
+ calcularTiempoTransicion() : int
+ simular() : bool
Compuerta
- tiempoTransicion: int
+ getEntrada() : Entrada*
+ getSalida() : Salida*
+ calcularTiempoTransicion() : void
+ simular() : void
+ rotar(DIRECCION) : void
+ mover(Posicion) : void
SalidaCompuerta
- compuerta: Compuerta*
- salidas: EntradaCompuerta**
- valor: bool
- validoValor: bool
- validoTiempo: bool
- tiempoTransicion: int
+ calcularTiempoTransicion() : int
+ simular() : bool
NOT
- entrada: EntradaCompuerta*
- salida: SalidaCompuerta*
+ calcularTiempoTransicion() : void
+ simular() : void
AND
- entradas[2]: EntradaCompuerta*
- salida: SalidaCompuerta*
+ calcularTiempoTransicion() : void
+ simular() : void
OR
- entradas[2]: EntradaCompuerta*
- salida: SalidaCompuerta*
+ calcularTiempoTransicion() : void
+ simular() : void
XOR
- entradas[2]: EntradaCompuerta*
- salida: SalidaCompuerta*
+ calcularTiempoTransicion() : void
+ simular() : void
Entrada
- salida: SalidaCompuerta*
+ calcularTiempoTransicion() : void
+ getEntrada() : Entrada*
+ simular() : void
Salida
- entrada: EntradaCompuerta*
+ calcularTiempoTransicion() : void
+ getSalida() : Salida*
+ simular() : void
Pista
- entrada: EntradaCompuerta*
- salida: SalidaCompuerta*
+ calcularTiempoTransicion() : void
+ simular() : void
CajaNegra
- entradas[]: EntradaCompuerta*
- salidas[]: SalidaCompuerta*
+ calcularTiempoTransicion() : void
+ simular() : void

## *Simulación*

Para la simulación se creo una clase Simulador que se encarga de generar las entradas y simular el circuito para cada combinación de entradas.

Para llevar a cabo la simulación cada compuerta tiene un método simular, el cual llama al simular de las compuertas conectadas a sus entradas obteniendo los valores de entrada y así transformarlos en la salida. De esta manera el circuito va a recorrer la lista de Salida llamando al simular de cada uno. Cada salida va a llamar al simular de la compuerta conectada a su entrada, y ésta va a llamar al simular de la compuerta conectada a su entrada y así sucesivamente hasta llegar a la Entrada.

Para implementar el método simular se utilizó el patrón Template. La clase Compuerta va a recorrer las entradas llamando al simular y luego va a llamar al método actuarSimular que va a estar definido en las clases concretas y se va a encargar de transformar las entradas en la salida

Persistencia

La persistencia esta desarrollada utilizando la librería de Apache Xerces C++, que es la encargada de guardar los archivos en formato XML.

La clase Persistencia es la encargada de cargar todos los archivos a memoria y de cualquier accion que requiera de identificar elementos en formato XML en un archivo. Por lo tanto para levantar un circuito en memoria, se utiliza la funcion *recuperarCircuito(),* la cual parsea el archivo, crea el circuito y devuelve un puntero al mismo.

A la hora de guardar un circuito en memoria, la funcion *guardarCircuito()* crea el documento y llama al metodo *guardar()* de la clase *Circuito*. Este metodo recorre la lista de compuertas y llama al *guardar()* de cada una, implementado polimorficamente ya que la clase *Compuerta* lo define como virtual puro. Por lo tanto cada compuerta sabe como debe persistirse y la clase *Persistencia* queda desligada de esta obligacion.

En la clase *Persistencia* tambien se encuentran los metodos para generar el XML en formato SOAP, para cumplir asi el protocolo y poder adjuntarlo al codigo HTML.

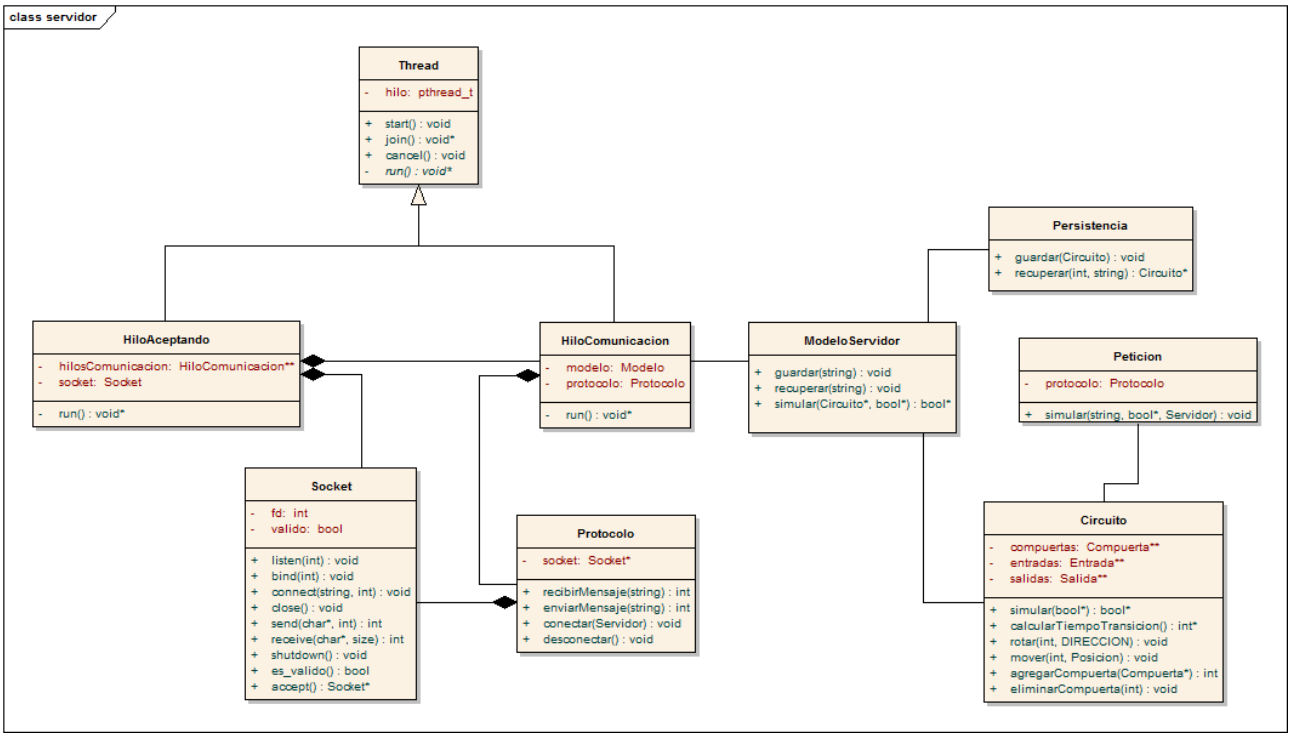
Publicación

La *Publicacion* esta compuesta por la clase *Mensajes, Publicacion* y *Servidor.* La clase *Mensajes* es la encargada de generar los mensajes en un elemento de formato XML, para que luego la clase *Publicacion* genere los mensajes SOAP, para enviarlos al servidor. Luego de enviado un mensaje, espera su respuesta e identifica el mensaje respuesto por el Servidor, para asi continuar con la aplicación de la forma requerida.

El *Cliente* tiene la posibilidad de publicar sus circuitos guardados, de descargar circuitos del servidor subidos por otros clientes, o sus propios circuitos. Todos los circuitos descargados se visualizan como una caja negra con sus respectivas entradas y salidas. Una vez conectada una caja negra de la forma adecuada, se procede a la simulacion.

La simulacion de una caja negra esta a cargo del servidor que la contiene, por lo tanto el *Cliente* debe enviar un mensaje por cada conjunto de entradas asignados, para poder obtener las salidas de la misma. Lo mismo sucede con los tiempos de la simulacion, pero es un solo mensaje ya que el tiempo no depende de los valores tomados en las entradas.

## *Servidor*




El *Servidor* es el encargado de responder a las peticiones de los distintos *Clientes,* siendo las distintas posibilidades: publicaciones de circuitos, solicitudes de circuitos, solicitudes de simulacion. Y a su vez debe poder funcionar como cliente para poder pedir la simulacion a otro servidor en caso de tener un circuito guardado que contenta una caja negra publicada en otro servidor o en si mismo.

Para soportar la concurrencia se utiliza un hilo por cada comunicación establecida, el *Servidor* tiene un hilo de acpetaciones, que esta constantemente aceptando peticiones y generando un hilo para cada una.

Existe un manager de archivos, el cual maneja la cantidad de archivos recibidos y enviados, asignandoles nombres distintos para que no ocurran problemas entre clientes. El manager de archivos tambien es el encargado de borrar los archivos generados temporalmente.

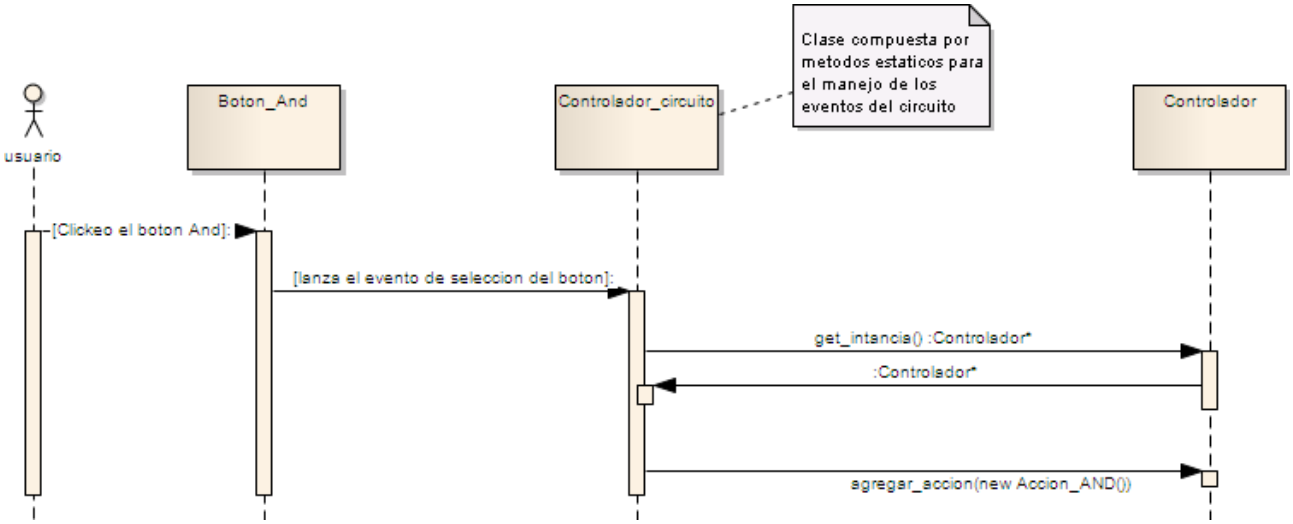
## *Controlador*

El controlador definirá el modo en que la interfaz reacciona a la entrada del usuario, por lo que será el encargado de manejar los eventos emitidos por la vista de la aplicación.

Esta clase esta compuesta principalmente por el modeloCliente, que es el objeto de la aplicación y por la fachadaVista, que es la una interfaz para la comunicación con la vista.

También entre sus atributos encuentra el modelo de la vista (Modelo_Vista_circuito), que facilitará la decisión del procedimiento a seguir a la hora de recibir un evento producido por la entrada del usuario.

El atributo acción, también juega un papel muy importante en el control del programa. Este representa la estrategia a seguir al recibir eventos provenientes del sector de la vista donde se diseñan los circuitos. Para explicar la utilidad de este componente incluimos el siguiente diagrama de secuencias:

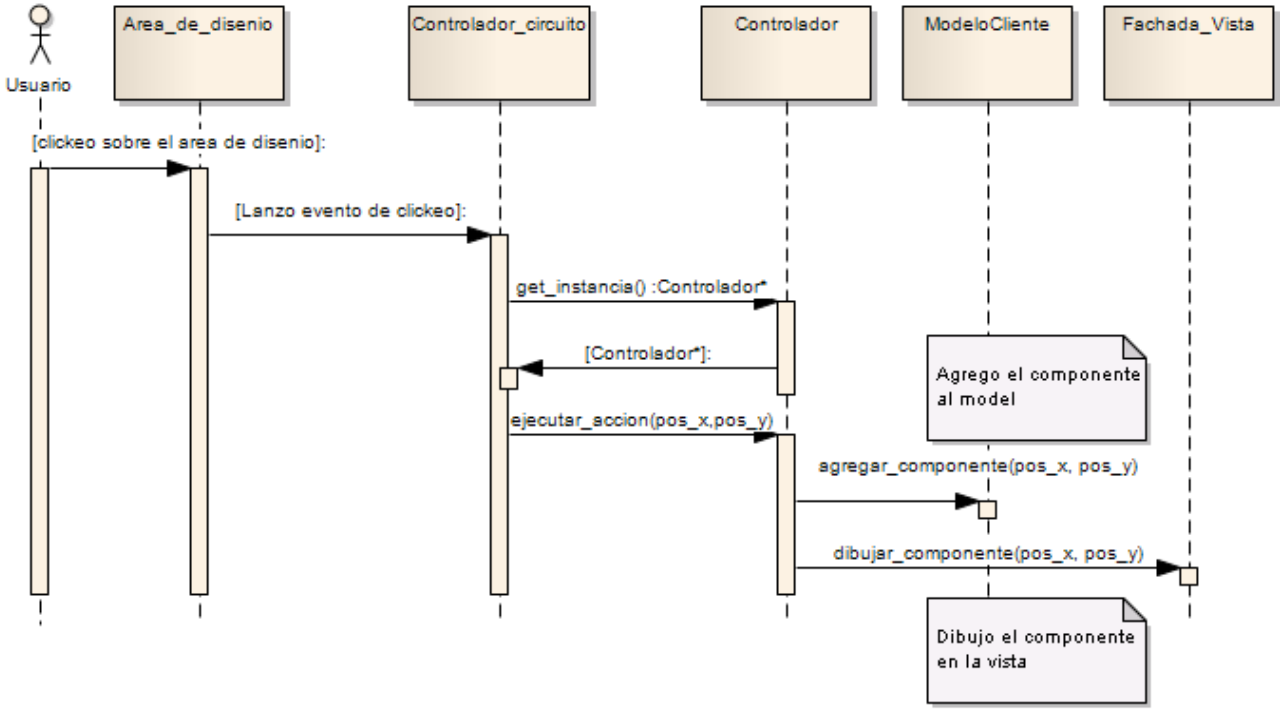



Como se puede apreciar en el diagrama, el usuario utiliza la herramienta para agregar una compuerta del tipo AND, la cual lanza un evento que es capturado por el el controlador del mismo.

*Antes de continuar con la explicación vale la pena mencionar el objetivo de la clase controlador_circuito, dado a que no se incluyo en el diagrama de clases. Esta clase contiene todos los manejadores de los eventos relacionados al diseño del circuito.*

Luego de capturar el evento, se obtiene la instancia del controlador y se ejecuta el método para agregarle una nueva acción al controlador, la acción de incluir una compuerta AND al modelo en este caso.

A partir de este momento, el controlador incorpora la estrategia a seguir al recibir la orden de ejecutar_acción, que es la requerida por el controlador del evento de clickeo sobre el área de diseño. A continuación mostraremos la secuencia que se sigue cuando el usuario clickea sobre el área de diseño.




El usuario clickea sobre el área de diseño, y el controlador_circuito captura el evento que esto produce, pide una instancia de la clase controlador e invoca al método ejecutar acción.

Debido a que anteriormente le agregamos la acción de incorporar una compuerta AND, el controlador agrega la compuerta en el modelo y la dibuja en la vista, en el caso de que se haya podido incorporar el componente con éxito.

Por ultimo, vale la pena mencionar, la decisión de implementar la clase controlador como un singleton. Esto fue principalmente debido a dos cuestiones, primero, la aplicación solo tendrá un controlador que se encargara de ser el intermediario entre el modelo del programa y sus posibles vistas, y segundo, la necesidad de que el acceso a esa única instancia sea global, para poder ser utilizada desde las funciones controladoras de eventos.

## *Vista*

La vista, es la representación de la aplicación en pantalla, con la cual el usuario va a tener la posibilidad de interactuar con el programa.

Como detalle de implementación, mencionaremos primero la utilización de la clase componente_Visual, que representará todos los componentes que se utilizaron para la vista. De esta manera podemos utilizar cualquier elemento visual como una objeto de esta clase.

Luego de definir esta clase, implementamos el patrón de diseño decorador, esto lo logramos a partir de heredar de componente visual la clase decorador, la cual tendrá como composición a otro componente_Visual. Esta decisión nos facilito decorar componentes de la vista, como por ejemplo con barras de scroll, o cajas contenedoras.

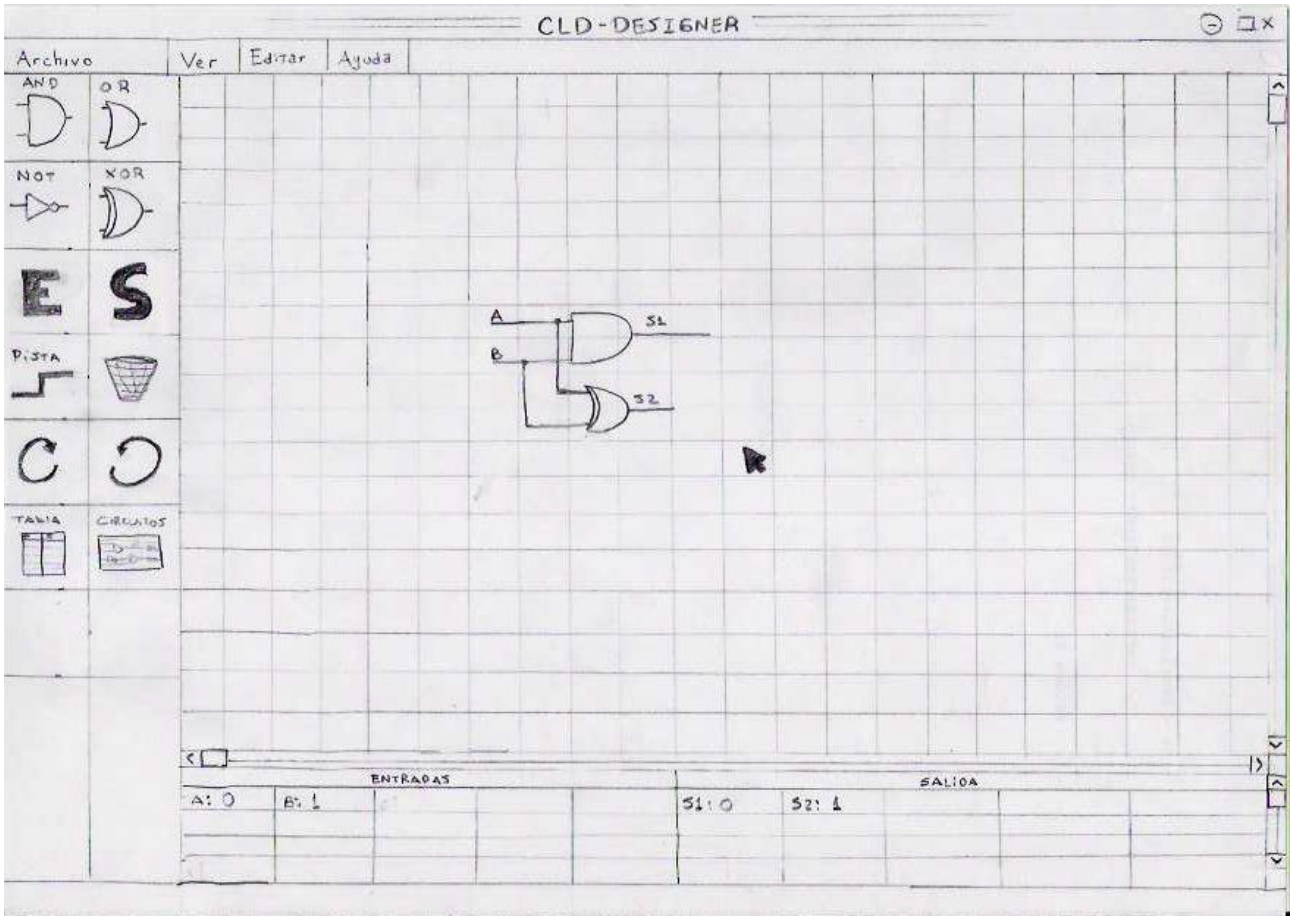
Como ultimo detalle de la vista, destacaremos el uso de la clase fachada, la cual tiene como objetivo, proporcionar una interfaz para la comunicación con cualquier objeto de la vista.

# Análisis de diseño

## Simulador de circuitos lógicos distribuidos

# Análisis de Diseño

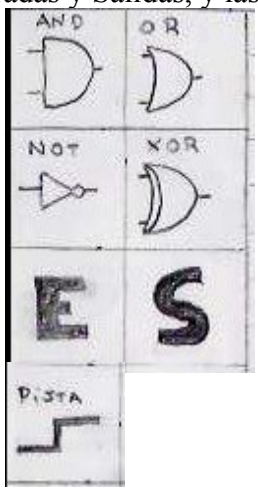
Para la realización de la aplicación hicimos gráficos estimativos de lo que seria visualmente la aplicación junto con sus funcionalidades. Los diagramas presentados, fueron los siguientes:




## Ventana del diseñador de circuitos lógicos:

La **barra lateral, de herramientas con opciones de diseño,** esta compuesta por:

- Las Compuertas, junto a las Entradas y Salidas, y las Pistas para diseñar los circuitos:




Accionando estas funciones se habilita a la colocación de las mismas en la *grilla.*

- La función Eliminar:

Accionando esta función se pueden borrar *compuertas*, clickeando sobre ellas.

- Las funciones de Rotación:

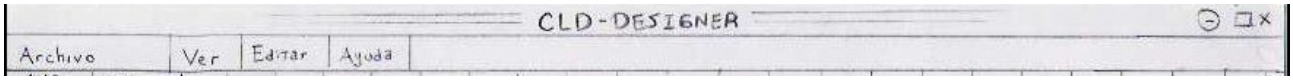
Accionando estas funciones se rota, en sentido horario o anti-horario, la *compuerta*, *entrada, salida* o *pista* seleccionada.

- La Tabla, que cumpliría la función de la Simulación:


Accionando esta función se simula el circuito y se imprimen los resultado en la *Tabla de Simulación*

- Y la función de publicaciones, que seria con la cual se conectaría al *servidor* para publicar o descargar un circuito:

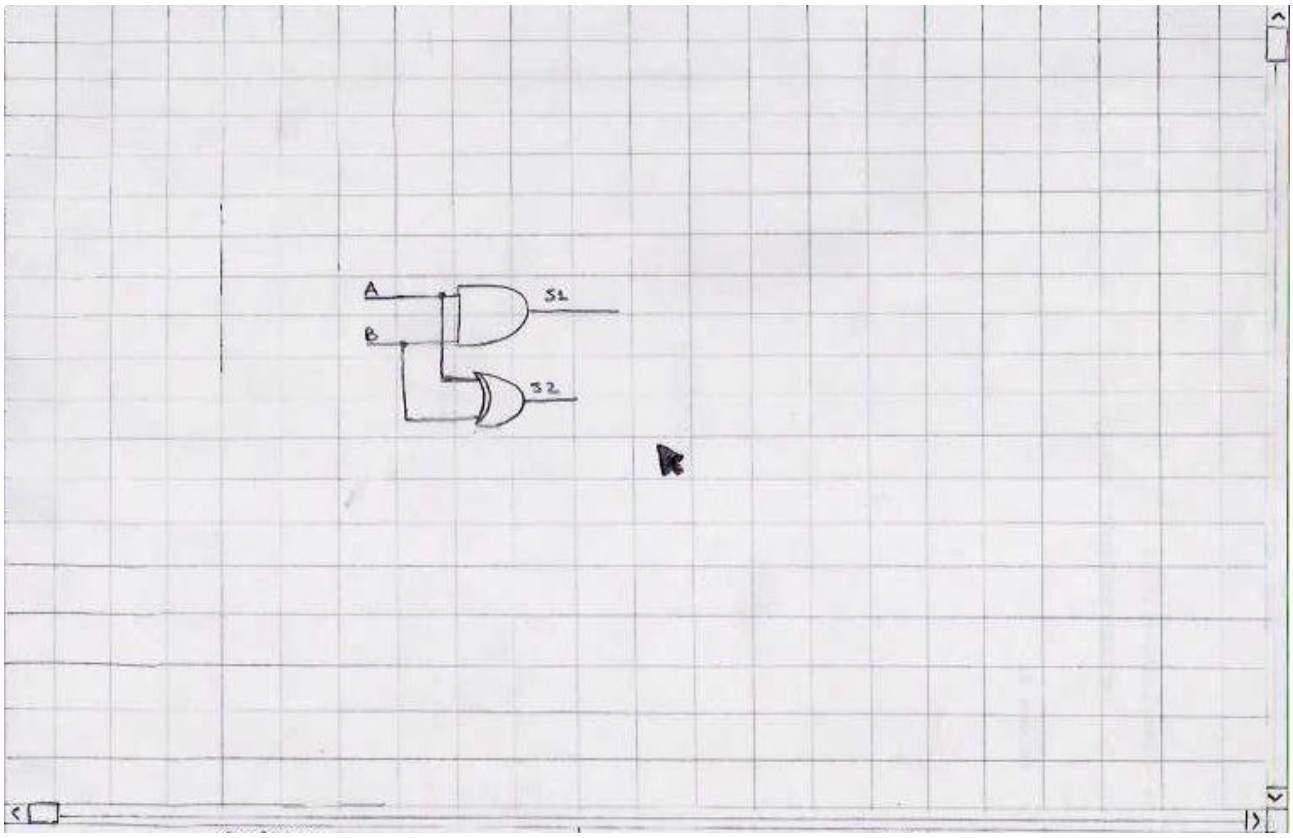

- La barra superior, de herramientas con opciones de archivo:




- El área de simulación (en tabla):

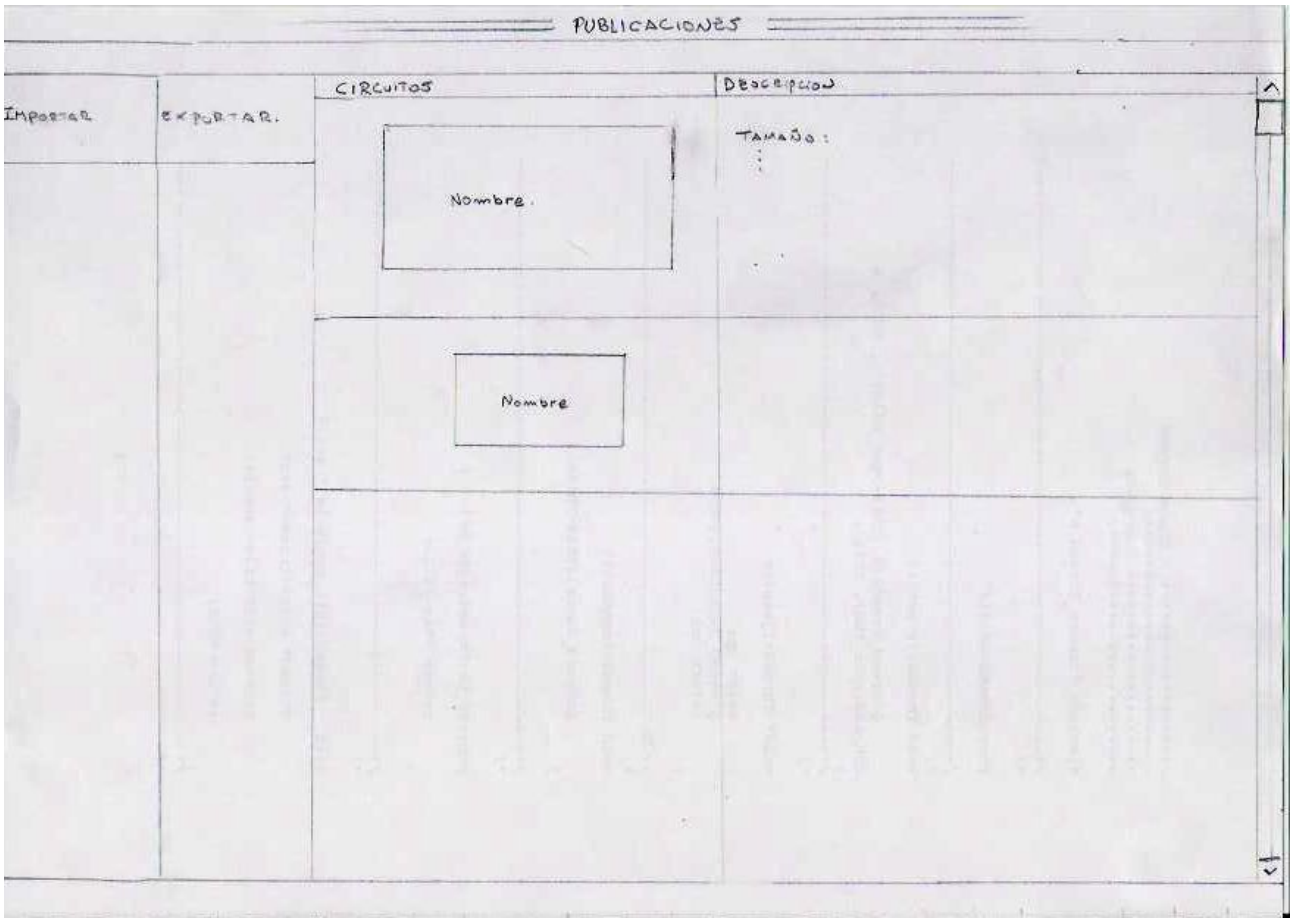

En esta tabla se representaría el resultado de la simulación del circuito graficado en la *grilla.*

- **El área de diseño:**




En esta *grilla* se podrían diseñar los circuitos lógicos previamente seleccionados en la *barra de herramientas con opciones de diseño.*

**Ventana del Servidor para Publicaciones de circuitos.**

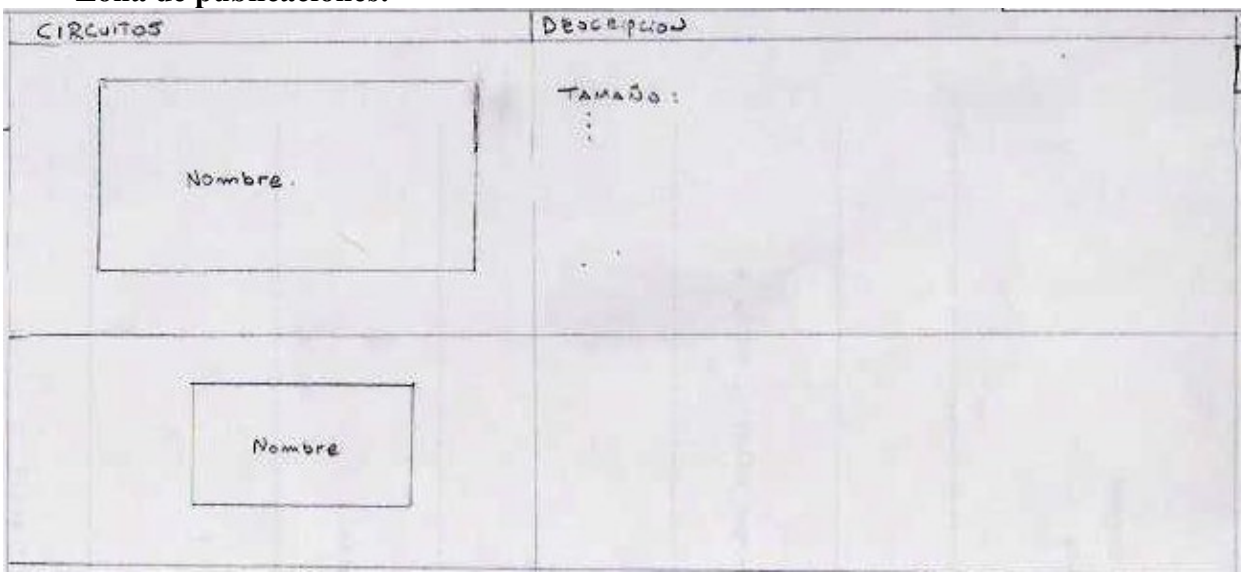



La **barra lateral, de herramientas con opciones de publicación,** esta compuesta por:

- **Publicaciones:**


Accionando la función de Importar, se descargaría el circuito seleccionado de la *zona de publicaciones.*
Accionando la función de Exportar, se publicaría el circuito diseñado en la *grilla.*

- **Zona de publicaciones:**




En la *zona de publicaciones* se visualizaran los distintos circuitos publicados por los clientes, con su nombre y descripción.

# Pruebas del proyecto

## Simulador de circuitos lógicos distribuidos

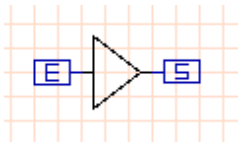
## *Introducción*

A continuación se detallaran las pruebas realizadas a la aplicación para corroborar su correcto funcionamiento. Las pruebas consistieron en:

- Verificar que las tablas de simulación sean correctas.
- Que los circuitos se publicados puedan ser descargados y simulados correctamente.
- Verificar tablas de simulación con circuitos descargados del servidor.
- Posibilidad de ver el diseño de las cajas negras.

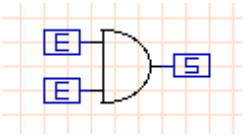
## *Compuertas individuales*

**NOT:**



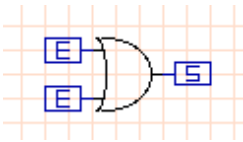

| entrada | salida | Salidas \| Tiempos \| |
|---|---|---|
| 0 | 1 | salida 5 |
| 1 | 0 | |

**AND:**



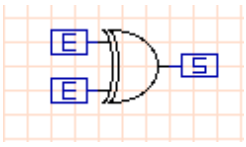

| entrada1 | entrada2 | salida | Salidas \| Tiempos \| |
|---|---|---|---|
| 0 | 0 | 0 | salida 10 |
| 0 | 1 | 0 | |
| 1 | 0 | 0 | |
| 1 | 1 | 1 | |

**OR:**




| entrada1 | entrada2 | salida | Salidas \| Tiempos \| |
|---|---|---|---|
| 0 | 0 | 0 | salida 20 |
| 0 | 1 | 1 | |
| 1 | 0 | 1 | |
| 1 | 1 | 1 | |

**XOR:**

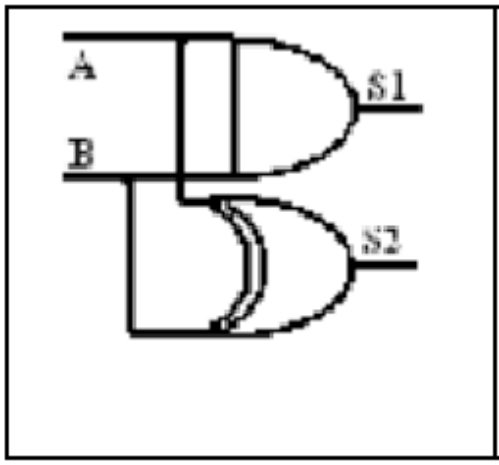



| entrada1 | entrada2 | salida | Salidas \| Tiempos \| |
|---|---|---|---|
| 0 | 0 | 0 | salida 30 |
| 0 | 1 | 1 | |
| 1 | 0 | 1 | |
| 1 | 1 | 0 | |

## *Sumador de dos bits*

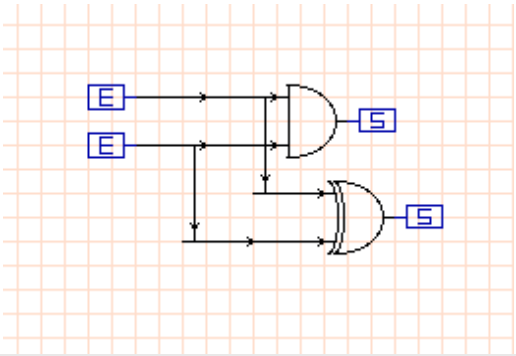
Compuesto por una compuerta AND y una XOR.

Diseño y tabla:




| A | B | S1,S2 |
|---|---|---|
| 0 | 0 | 00 |
| 0 | 1 | 01 |
| 1 | 0 | 01 |
| 1 | 1 | 10 |

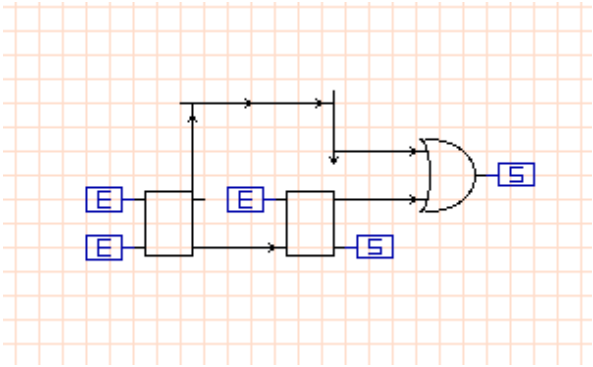
Resultados:




| entrada1 | entrada2 | salida1 | salida2 | Salidas | Tiempos |
|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | salida1 | 10 |
| 0 | 1 | 0 | 1 | salida2 | 30 |
| 1 | 0 | 0 | 1 | | |
| 1 | 1 | 1 | 0 | | |

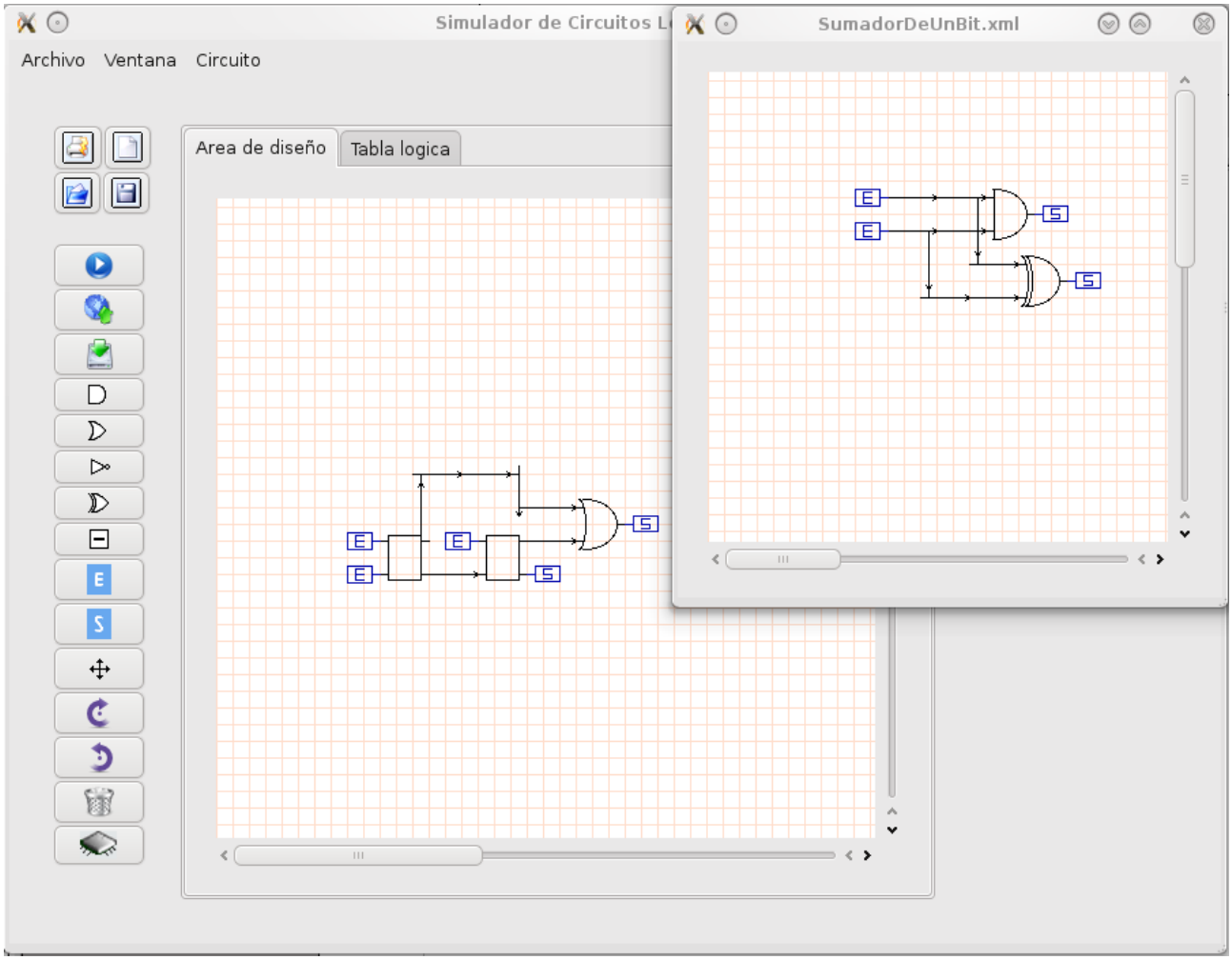
## *Sumador de tres bits con cajas negras*

Compuesto por dos **sumadores de dos bit**, que seran descargados del servidor, por lo tanto seran simulados como **cajas negras,** y una compuerta **OR** para el carry**:**



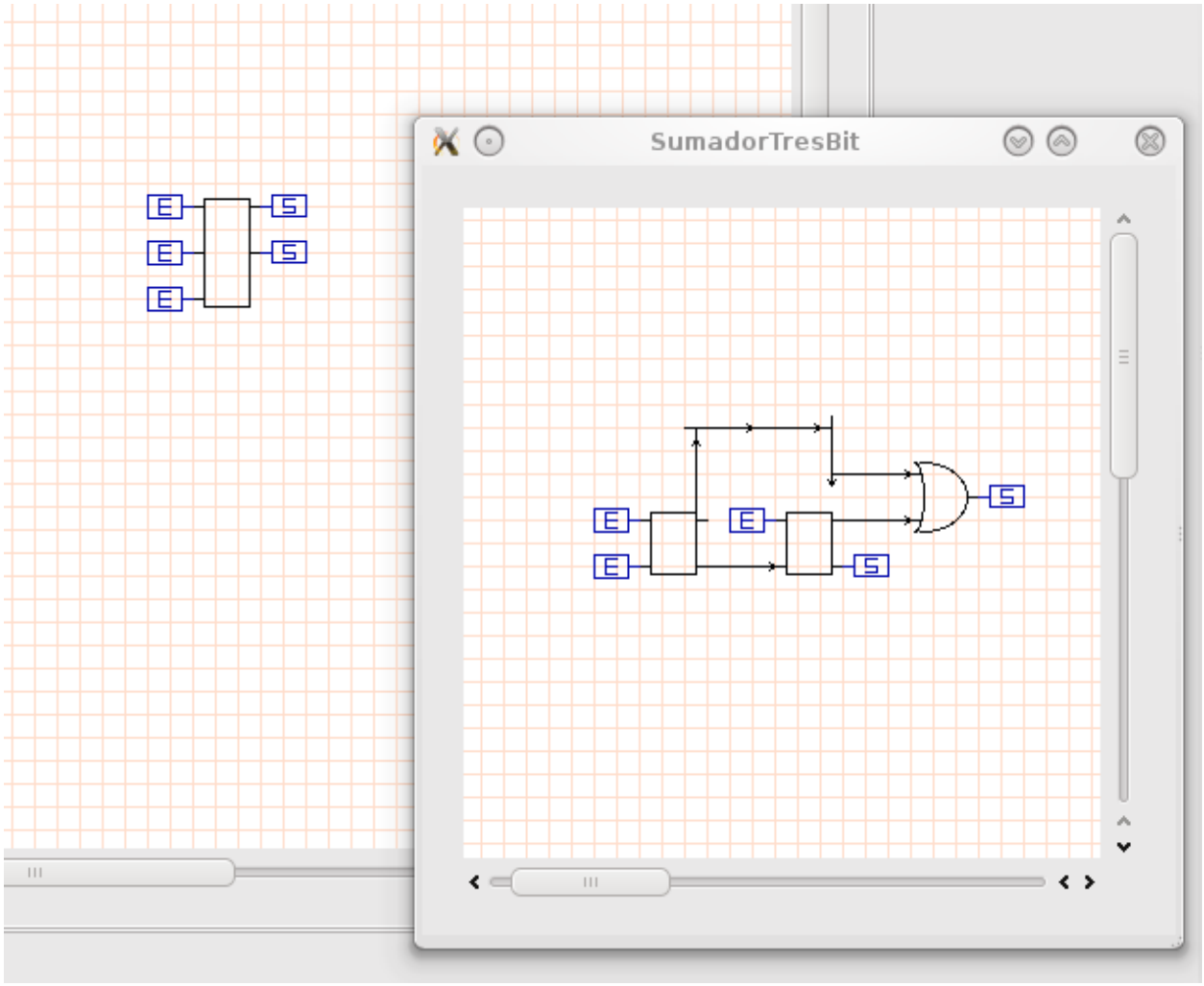

| entrada1 | entrada2 | entrada3 | Suma | carry | Salidas | Tiempos |
|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | Suma | 60 |
| 0 | 0 | 1 | 1 | 0 | carry | 60 |
| 0 | 1 | 0 | 1 | 0 | | |
| 0 | 1 | 1 | 0 | 1 | | |
| 1 | 0 | 0 | 1 | 0 | | |
| 1 | 0 | 1 | 0 | 1 | | |
| 1 | 1 | 0 | 0 | 1 | | |
| 1 | 1 | 1 | 1 | 1 | | |

## *Diseño de cajas negras*



### *Conexión servidor-servidor*

Utilizando un circuito como caja negra compuesto por  cajas negras:



# Definición del proyecto

## Simulador de circuitos lógicos distribuidos

## *Cronograma*

**10 Noviembre:**

- Pantallas
- Diagrama de clases
- División de tareas
- Socket y Thread

**17 Noviembre:**

- Circuito
- Ventanas
- Persistencia

**24 Noviembre:**

- Simulación
- SOAP
- Vista (Compuertas, Drag and Drop, rotar)
- Tabla Lógica

**1 Diciembre:**

- Servidor
- Integración
- Imprimir
- Preentrega

**8 Diciembre:**

- Pruebas
- Bugs
- Documentación

**15 Diciembre:**

- Instalador
- Documentación
- Bugs
- Entrega Final

## *División de tareas*

| | Juan Monetti | Giovanni Opromolla | Diego Saez |
|---|---|---|---|
| Circuito | | | X |
| Vista | X | | |
| Persistencia | | X | |
| Controlador | X | | |
| SOAP | | X | |
| Simulación | | | X |
| Modelo Cliente | | | X |
| Servidor | | X | |
| Socket/Thread | X | X | X |
| Impresión | X | | X |
| Integración | X | X | X |
| Documentación | X | X | X |
| Pruebas | X | X | X |

## *Reporte de avances*

**10 Noviembre:**

**17 Noviembre:**

- Socket y Thread
- Diagrama de clases
- Ventana

**24 Noviembre:**

- Vista - Compuertas
- Vista - Drag and Drop
- Simulación
- Persistencia - Guardar

**1 Diciembre:**

- Persistencia - Recuperar
- Tabla Lógica
- SOAP
- Integración

**8 Diciembre:**

- Bugs
- Servidor
- Pruebas
- Integración

**15 Diciembre:**

- Publicación
- Preview Caja Negra
- Instalador
- Documentación
- Bugs
- Impresión
- Entrega Final

## *Conclusión*

A modo de conclusión, podemos destacar la importancia que tubo el proyecto para poder poner en práctica los conocimientos adquiridos hasta el momento en el curso, como programación concurrente , conceptos de comunicación e interfaces gráficas.

Otro aspecto importante, fue la modalidad de trabajo para la construcción de la aplicación. En la cual nos encontramos con un cronograma recortado debido a la reorganización del grupo de trabajo. Pero gracias a un buen planteamiento de fechas cortas con objetivos concretos, y la buena comunicación y entendimiento que tubo el grupo, pudimos llegar a separar tareas individuales pero de fácil integración, que fue lo que nos permitió llegar a concluir el trabajo.

Por ultimo, a lo largo del proyecto, fuimos adaptando las ideas que en un principio habíamos planteado, por ejemplo en lo que tiene que ver a la parte gráfica, partimos de la base de la idea principal y la fuimos cambiando de acuerdo a los resultados que obteníamos por pantalla y las facilidades que la biblioteca nos otorgaba.