

Proyecto I – Circuit Designer

Instituto Tecnológico de Costa Rica
Área Académica Ingeniería en Computadores
Algoritmos y Estructuras de Datos I (CE 1103)
Primer Semestre 2019
Valor 25%



Objetivo General

- Implementar una aplicación que permita el diseño de circuitos lógicos.








Objetivos Específicos

- Implementar listas enlazadas y algunas variaciones de estas.
- Desarrollar una aplicación en el lenguaje de programación Java.
- Investigar acerca de programación orientada a objetos en Java.
- Aplicar **patrones de diseño** en la solución de un problema.
- Utilizar UML para modelar la solución de un problema.

Descripción del Problema

Se solicita implementar una aplicación desktop, que permita la edición de diagramas lógicos construidos a partir de compuertas lógicas ofrecidas por el sistema y otras pre-construidas por los mismos usuarios.

Los componentes básicos ofrecidos por este sistema son:

AND	NAND	OR	NOR	NOT	XOR	XNOR
						
Dos o más entradas. Una salida	Dos o más entradas. Una salida	Dos o más entradas. Una salida	Dos o más entradas. Una salida	Una entrada y una salida	Dos o más entradas. Una salida	Dos o más entradas. Una salida

La aplicación permite al usuario arrastrar y conectar las compuertas disponibles. La salida de una compuerta se puede convertir en la entrada de otra. Es decir, las compuertas se pueden interconectar entre sí.

Las entradas se etiquetan con una <número> y las salidas con una <número>. Todas las líneas deberán tener asociado un color aleatorio. Una vez terminado el sistema podrá calcular la tabla de verdad del circuito y decir cuál sería el resultado que enviará con determinadas entradas.

El modelo creado por el usuario puede guardarse como un nuevo componente en la paleta de compuertas de la aplicación

El programa debe ser capaz de realizar la simulación de la salida para una determinada entrada. Se le debe permitir al usuario insertar los valores para las entradas y con esto la aplicación debe calcular el resultado que daría el circuito.

A continuación, se presenta un ejemplo de un circuito lógico y su respectiva tabla de verdad.

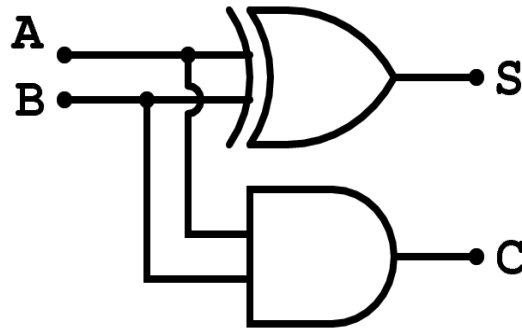
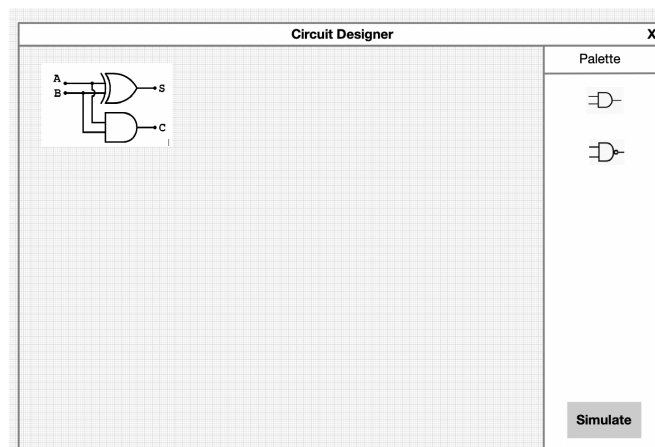


Fig. 1. Esquema lógico de un sumador medio (half adder)

A	B	C	S
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

Fig 2. Tabla de verdad del circuito anterior

A continuación, se muestra un esquema general de la aplicación por construir:



Para representar el estado del circuito en memoria, el estudiante debe diseñar una lista que permita representar las compuertas y sus interconexiones. Es decir, conforme se agregan compuertas al diseño, se agrega un nuevo nodo en una lista. Dicho nodo puede tener uno o más enlaces a otros nodos dependiendo de las conexiones que soporta dicha compuerta.

No está permitido el uso de librerías para la implementación de las estructuras de datos que se utilicen en el proyecto. Cualquier estructura de datos que se requiera debe ser implementada por el estudiante. Para aspectos como la interfaz gráfica si está permitido el uso de librerías.

Documentación requerida

1. Internamente, el código se debe documentar utilizando JavaDocs.
2. Dado que el código se deberá mantener en GitHub, la documentación externa se hará en el Wiki de GitHub. El Wiki deberá incluir:
 - a. Breve descripción del problema
 - b. **Planificación y administración del proyecto:** se utilizará la parte de project management de GitHub para la administración de proyecto. Debe incluir:
 - Lista de features e historias de usuario identificados de la especificación.
 - Distribución de historias de usuario por criticalidad y secuencia.
 - Minimal System Span
 - Plan de iteraciones que agrupen cada bloque de historias de usuario de forma que se vea un desarrollo incremental
 - Descomposición de cada user story en tareas.
 - c. Modelo de Dominio en formato JPEG o PNG.
 - d. Diagrama de Clases en formato JPEG o PNG.
 - e. Descripción de las estructuras de datos desarrolladas.
 - f. Descripción detallada de los algoritmos desarrollados.
 - g. Problemas encontrados en forma de bugs de *github*. En esta sección se detalla cualquier problema que no se ha podido solucionar en el trabajo.

Aspectos operativos y evaluación:

1. **Fecha de entrega: De acuerdo con el cronograma del curso.**
2. El proyecto tiene un valor de 25% de la nota del curso.
3. El trabajo es **individual**.
4. Es obligatorio utilizar un GitHub.
5. Es obligatorio integrar toda la solución.
6. El código tendrá un valor total de 75%, la documentación externa 20% y la defensa 5%.
7. De las notas mencionadas en el punto anterior se calculará la Nota Final del Proyecto.
8. Se evaluará que la documentación sea coherente, acorde a la dificultad/tamaño del proyecto y el trabajo realizado, se recomienda que realicen la documentación conforme se implementa el código.
9. La nota de la documentación externa es proporcional a la completitud del proyecto, y viceversa.
10. La documentación se revisará según el día de entrega en el cronograma.
11. Las citas de revisión oficiales serán determinadas por el profesor durante las lecciones o mediante algún medio electrónico.
12. Los estudiantes pueden seguir trabajando en el código hasta 15 minutos antes de la primera cita de revisión oficial.
13. Aún cuando el código y la documentación tienen sus notas por separado, se aplican las siguientes restricciones:
 - a. Si no se entrega documentación externa, automáticamente se obtiene una nota de 0.

- b. Si no se utiliza un manejador de código se obtiene una nota de 0.
 - c. Si la documentación no se entrega en la fecha indicada se obtiene una nota de 0.
 - d. Sí el código no compila se obtendrá una nota de 0, por lo cual se recomienda realizar la defensa con un código funcional.
 - e. El código debe ser desarrollado en Java, en caso contrario se obtendrá una nota de 0.
 - f. Si no se siguen las reglas del formato de email se obtendrá una nota de 0.
 - g. La nota de la documentación debe ser acorde a la completitud del proyecto.
14. La revisión de la documentación será realizada por parte del profesor, no durante la defensa del proyecto. El único requerimiento que se consultará durante la defensa del proyecto es el diagrama de clases, documentación interna y la documentación en el manejador de código.
 15. Cada estudiante tendrá como máximo 15 minutos para exponer su trabajo al profesor y realizar la defensa de éste, es responsabilidad de los estudiantes mostrar todo el trabajo realizado, por lo cual se recomienda tener todo listo antes de ingresar a la defensa.
 16. Cada excepción o error que salga durante la ejecución del proyecto y que se considere debió haber sido contemplada durante el desarrollo del proyecto, se castigará con 2 puntos de la nota final del proyecto.
 17. Durante la revisión únicamente podrán participar el estudiante, asistentes, otros profesores y el coordinador del área.