ELO 329 - Documentacion
Tarea 3: Compuertas Digitales como Objetos de Software en C++

Daniel Cárdenas: 2921033-0
Roberto Muñoz: 2821033-5
Luis Muñoz: 2821033-7

a) Para el código entregado analice el código del método simulate() de la clase Simulator. Haga los gráficos de las señales para ver el efecto de dejar sólo un llamado a log(currentTime) o dos (uno antes y otro después de procesar los eventos). ¿Cuál es la ventaja de poner dos llamados en lugar de uno?. ¿Podría usted armar este circuito sin usar instancias de Wire?

Analizando el código, el método simulate() crea un vector de prioridades donde almacena objetos de tipo ChangeEvent, después usa el método writeHeader de la misma clase para escribir la cabecera en los archivos de salida de los meter. Posteriormente entra a un ciclo iterativo en donde saca el elemento más alto del vector, esto lo puede hacer ya que el vector usa el template priority_queue en el cual con el cual a través del metodo operator() definido en la clase ChangeEvent define cual clase es mayor que otra usando en este caso el tiempo ordenando de mayor a menor, donde el elemento más alto es que tiene menos tiempo. Posteriormente escribe en el archivo de salida el tiempo del evento elegido, después entra a otro ciclo el cual está desarrollado para implementar eventos simultáneos, eso lo hace revisando el tiempo del siguiente evento y comparando si los tiempo son iguales (o la lista esta vacía), de esta manera agrega elemento al vector simultEvent en donde estarán todos los eventos en el mismo tiempo y el vector pq ya no tendrá estos eventos simultáneos. Posteriormente actualiza todas las componentes asociadas a los eventos simultáneos, los propaga y los elimina.

Ejecutando el avance entregado (el cual presenta 2 llamados a log) de la tarea se observa el siguiente resultado:

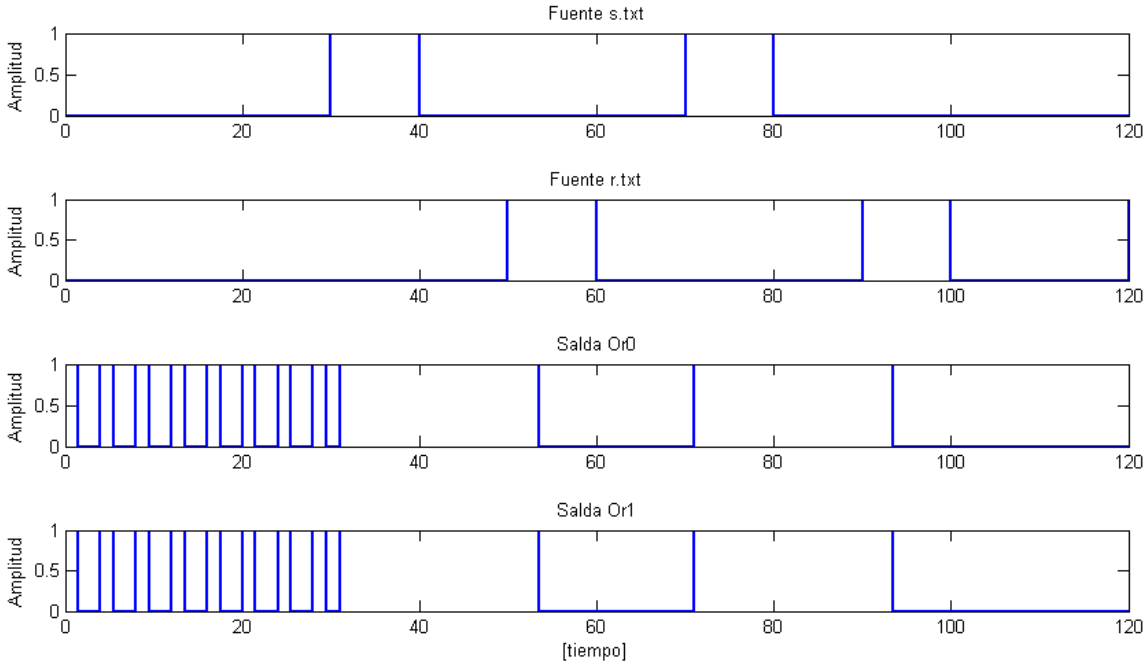

Figure 1: Simulación con 2 log

El resultado con un llamado a log es le siguiente:

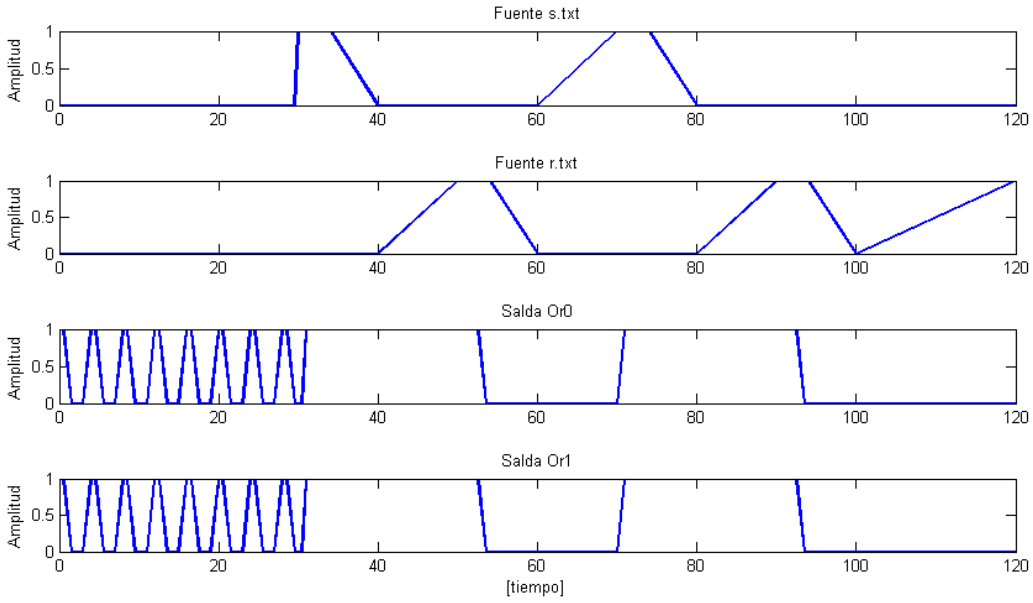

Figure 2: Simulación con 1 log

Por los resultados se observa que al realizar 2 llamados a log, se presentan los cambios de valores en instantes inmediatos, esto se ve a que se presenta el valor antes de que se propage el evento y despues de que se propague el evendo, de no hacer esto, el resultado se interpreta como que el tiempo cambió en el momento en que comienza el siguiente evento y no el momento en que termina el mismo evento.

Si se puede conectar sin implementar la clase wire, esto se debe a que los métodos implementados para conectar las compuertas piden que la entrada implemente la clase Outsignal o Insignal. Editando la clase DigitalCircuit.cpp se comprueba que la simulación se puede realizar y que el resultado es correcto:

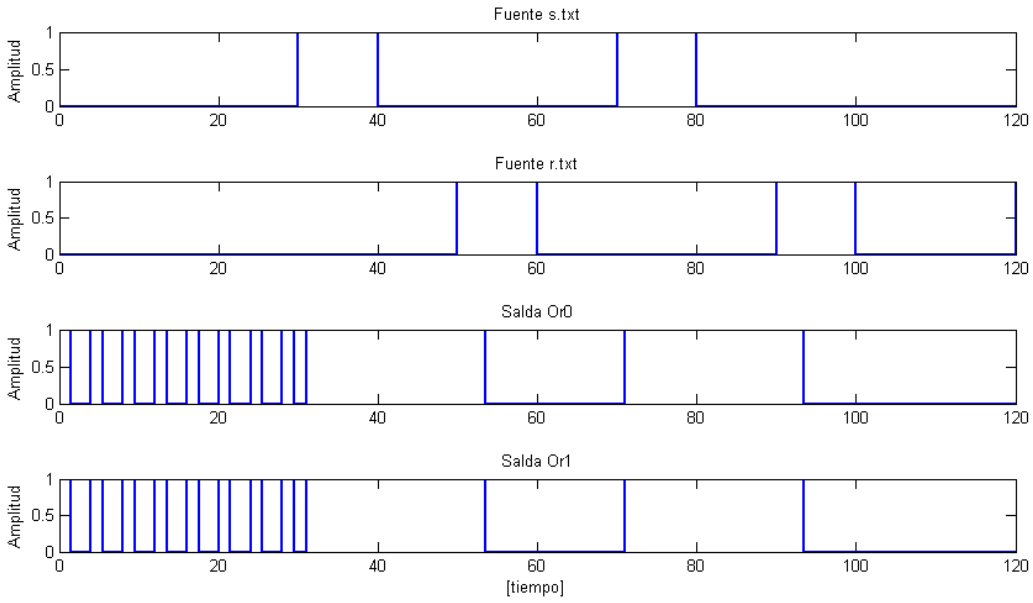

Figure 3: Simulación sin usar clase wire

b) Cambie y el archivo DigitalCircuit1.cpp para simular el circuito de la Figura 1. Incorpore un contador de instancias activas de ChangeEvent. Su objetivo es identificar si todos los objetos creados en el Heap son luego destruidos. Este programa no tiene archivos de entrada, sólo pida el archivo de salida por consola. Además pida un límite para el tiempo de simulación. Al término de su programa muestre por pantalla en número máximo de instancias creadas y el valor final de instancias activas (debería ser cero en circuitos que lleguen a estados estables). Haga los cambios necesarios si no es el caso. Utilice 2 [ns] (en realidad las unidades de tiempo son irrelevantes) como retardo de cada inversor. Muestre un diagrama temporal para la salida de ambos inversores.

Para incorporar lo solicitado, se realizaron los siguientes cambios:

1) Se agregó 2 parámetros estáticos a la clase ChageEvent, los cuales cuentan el numero de instancias actualice, y el numero de instancias activas, este numero aumenta en el constructor de la clase y se descuenta en el destructor de esta mista, con esto se asegura que los objetos son debidamente destruidos y no se presenta una fuga de memoria.
2) Se agrego 1 parámetros para el máximo tiempo de simulación para la clase Simulate, este se ingresa al constructor de la clase, y realiza un break al momento se simular (Si se cumple la condición debida) lo que provoca que se detenga la simulación.
3) Se cambió el código de DigitalCircuit.cpp para que implementara el cirucito de la figura1 del enunciado de la tarea, además se implementó el ingreso del tiempo se simulación por consola, además de que para su llamado solo se presente el archivo de salida.

Con esto cambios se obtuvo el siguiente resultado:

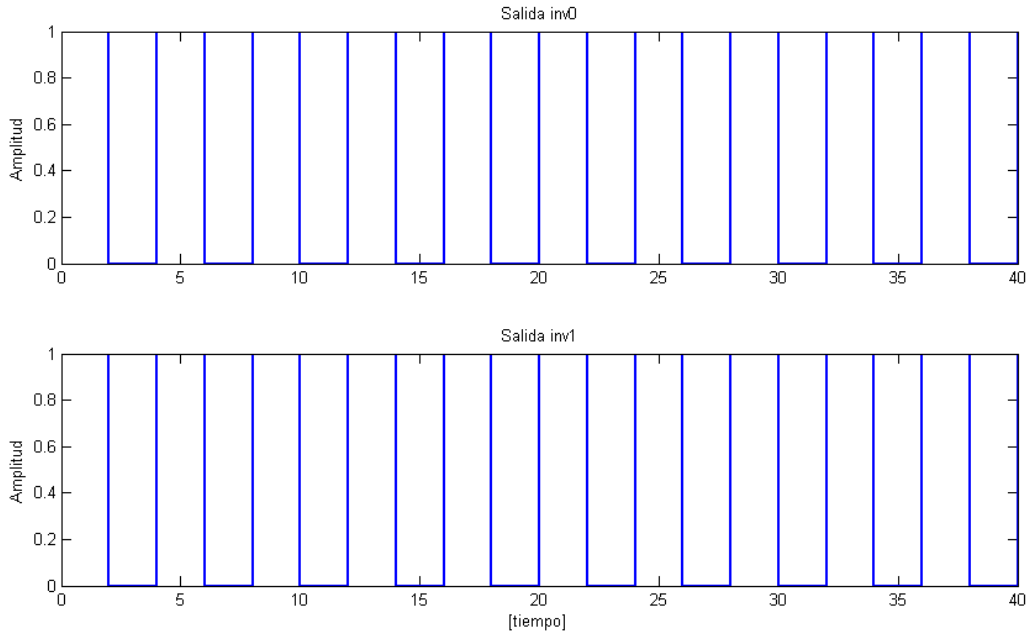

Figure 4: Salida DigitalCircuit.cpp del circuito de la Figura 1

c) Cree la clase And. Utilice los mismos tipos de constructores que en Or

Se crearon las clases solicitadas, estas son muy similares a la clase Or, con la diferencia de cambiar el retorno del método computeOutput().

d) Cree la clase Nand. Muestre el diagrama temporal generado cuando el reloj es de frecuencia igual a 20 veces el mayor retardo por usted considerado y la señal Data también es cuadrada y de la mitad de la frecuencia del reloj. Muestre 1,5 periodos de la señal D. No se espera que su programa genere el diagrama temporal. Puede usar el dibujo aquí incluido y el gráfico lo puede generar con una planilla de cálculo a partir de los datos de salida de su programa.

Generando DigitalCircuit2.cpp se obtuvo el siguiente resultado:

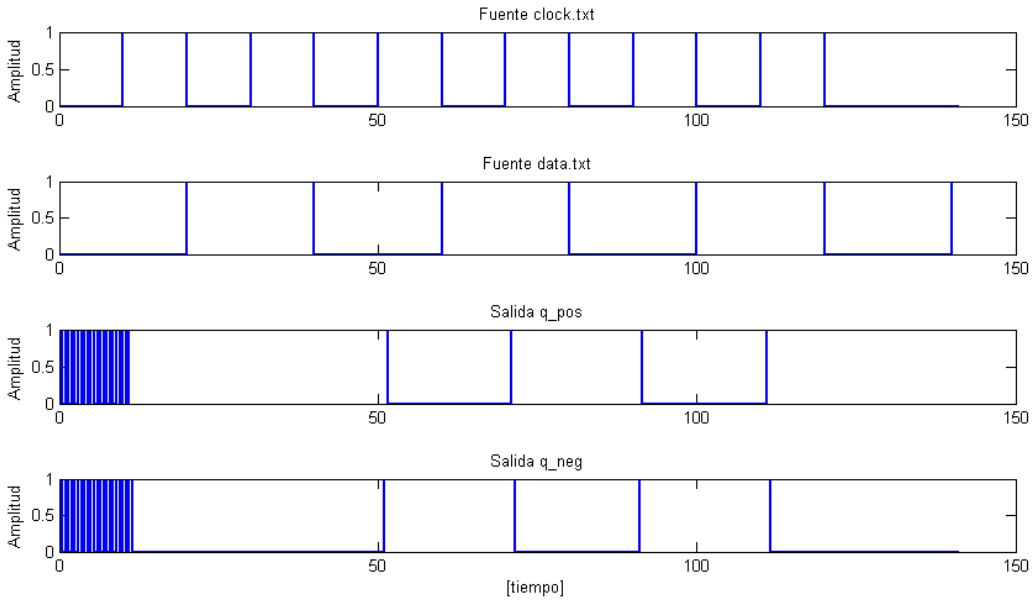

Figure 5: Salida DigitalCircuit2.cpp del circuito de la Figura 2

DIFICULTADES ENCONTRADAS

1) Se presentó como dificultad la implementación de una makefile que compilará los 2 digitalcircuit, ya que no se puede crear el ejecutable de un conjunto de archivos que contengan mas de un main, esto se solucionó utilizando 2 etiquetas distintas para las distintas clases, y se creo el ejecutable por separado.
2) Se presentó una dificultad el la lectura de archivos en c++ la cual incorpora como último carácter 0x0a (ASCII Line Feed, nueva línea) al final de los archivos, esto se arreglo utilizando la recomendación mencionada por el profesor, utilizar como condición si el archivo esta al final y de ser así salirse.
3) Se presentó un problema al momento se conectar los cables, al momento de colocar un input la clase de la compuerta llama a inputvaluechange() en el método connectInput() el cual pide a la compuerta conectada computar su salida y si esta no tiene antes entradas el programa retorna un error, esto se solucionó agregando primero conectando primero las salidas y después las entradas.